COMPUTATIONAL METHODS IN MACHINE LEARNING FOR PRIVACY PRESERVATION

A THESIS SUBMITTED TO THE AUCKLAND UNIVERSITY OF TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

March 2026

By

Bo Ma

School of Engineering, Computer & Mathematical Sciences

Abstract

A huge amount of data is currently stored online for training deep learning models. Cryptographic techniques such as fully homomorphic encryption (FHE) and secure multi-party computation (MPC) in principle, enable ML over encrypted or distributed data, but they protect the confidentiality of computation and communication and have different threat models and application scenarios. This thesis focuses on a complementary concern: protecting against inference from the *outputs* of machine learning (e.g., released models or predictions)—such as membership inference, attribute inference, and model inversion—for which differential privacy (DP) is well suited. Differential privacy adds appropriate noise according to a predetermined privacy budget to limit such inference.

But even if the differential privacy method is introduced to realize the function of privacy protection, it will have a negative impact on the learning performance of the machine learning model. The main question is, can some methods be found to measure privacy-preserving capability and machine-learning accuracy, and at the same time propose a privacy-preserving method combine with the machine-learning model to balance the trade-off between accuracy and privacy?

Based on this question and motivations, in this thesis, a privacy-preserving framework for deep learning that contributes towards solving this problem is presented. This framework consists of three layers pre-processing layer, a model layer, and an assessment layer. The approaches to the proposals consist of three stages of frameworks. The

function of the pre-processing layer is to implement privacy-preserving. This approach will generate the required privacy noises for privacy protection. In the model layer, multiple methods for natural language processing and object detection/recognition have been investigated, with the data that has noise injection for the purpose of privacy protection, in this stage, two main approaches have been proposed, they are A privacy-preserving deep transformation self-attention (PPDPTS) and PDPIFSEA algorithms.

In the third layer, the function of quality assessment verifies the quality of the model that has been trained to determine whether the output inferred by the model can reach the desired level of privacy. In this framework, the data noises for privacy protection are quantified for deep learning models through statistical-based analysis methods has been proposed, there are BUA TDA, EMPA for quantified image-based datasets, and ABAPER, PBVS algorithms. Our experimental results show that the accuracy produced by the models in this framework is higher than that using other privacy-preserving methods.

The original contributions of this thesis are: (1) A novel dynamic entropy-based noise-generating method with differential privacy approaches to improve privacy protections for federated deep learning. (2) A novel distributed stochastic gradient descent for improving the performance of privacy-preserving deep learning. (3) A privacy-preserving deep transformation self-attention (PPDPTS) method was applied to implement a self-attention mechanism on visual features of images to assist privacy preservation. (4) A method to measure the privacy boundary and privacy budget in privacy-preserving deep learning and test the privacy budget for preventing privacy leakage in the output data and a predictive reconstruction algorithm to predict the distribution of data in the privacy-preserving deep learning model, aligning the privacy budget with prediction results.

Contents

Abstract **2**

Publications **12**

Acknowledgements **13**

Attestation of Authorship **14**

1 Introduction **15**
1.1 Main Limitation and Motivation for Privacy-Preserving Related Issue in Machine Learning . . . 19
1.1.1 Main Limitations of the Current State-of-the-Art for Privacy Preservation . . . 19
1.1.2 Main Problems for Privacy Leakage in Machine Learning . . . 20
1.1.3 Motivations for Privacy-Preserving in Machine Learning . . . 22
1.2 Motivation Questions . . . 24
1.3 Research Framework and Contributions . . . 25
1.3.1 Published Work Derived from this Thesis . . . 29
1.4 Outline of the Thesis . . . 30

2 Literature Review **31**
2.1 Structure of Chapter . . . 31
2.2 Traditional Privacy Protection Methods and Differential Privacy Algorithm 32
2.3 Privacy-Preserving Federated Learning . . . 38
2.3.1 Advantage for Federated Learning . . . 39
2.3.2 Limitation for Federated Machine Learning . . . 41
2.4 Methods Related to Privacy Noise Sampling and Measurement . . . 43
2.4.1 Differential Privacy and Kullback–Leibler Divergence . . . 43
2.4.2 Differential Privacy and Reproducing Kernel Hilbert Space . . 44
2.4.3 Abstract Wiener Space . . . 46
2.4.4 Deep Belief Network with Privacy-Preserving . . . 47
2.4.5 Related Works for Label Search Methods . . . 48
2.4.6 Privacy Cost with Privacy Path Length in Spinning Tree . . . 51
2.5 Application Areas of Machine Learning for Privacy Preservation . . . 55

2.5.1 Related Works with Image Protection . . . 57
2.5.2 State-of-the-Arts with Privacy-Preserving Visual Object Detection Methods . . . 57
2.5.3 Privacy-Preservation Support Vector Machine for Text-Based Data Classification Tasks . . . 60
2.5.4 Word Embedding for Text Classification . . . 62
2.6 Description of Experimental Evaluation Metrics for Related Work . . 65
2.6.1 A Central Scheme to Experimentally Measure Privacy Boundaries and Evaluate the Capabilities of Privacy-Preserving Machine Learning Methods . . . 65
2.6.2 Metrics for Measuring Privacy-Preserving Machine Learning Model Capabilities . . . 68
2.6.3 Intersection over Union (IoU) . . . 71
2.6.4 Experimental Design with Root Mean Square Error (rMSE) . 72
2.6.5 Experimental Design with Data Quality and Noise-Related Assessment Methods . . . 73
2.6.6 The Experimental Design of Differential Privacy Noise Analyze on K-L Divergence . . . 75
2.6.7 Methods for Measuring Spanning Trees and Adjacency Matrices 77
2.6.8 Experimental Methods for Chi-Squared Test in Privacy Protection 78

3 Privacy Preservation Methods for Pre-Processing Stage 80
3.0.1 Structure of Chapter . . . 81
3.0.2 Threat Model for Privacy Leakage in Machine Learning . . . 82
3.1 Motivation with Limitations of Differential Privacy . . . 84
3.2 Background Theories . . . 88
3.2.1 Kernel Hilbert Space . . . 88
3.2.2 Abstract Wiener Space . . . 90
3.3 Measuring Information Loss with Kullback-Leibler Divergence . . . 92
3.4 Method of Generate the Privacy Noise with the Privacy Budget . . . 97
3.5 Discernibility Matrix for Noise Generation . . . 110
3.5.1 Discernible Metric . . . 117
3.5.2 Indicator for Privacy Budget Measurement . . . 121
3.5.3 Information Loss for Data Sampling Processing . . . 124
3.5.4 Top-k Deviation for Feature Extraction in Privacy Cost Measurement . . . 127
3.6 The Motivation and Detail of Modified Dynamic Conditional Random Field . . . 130
3.7 The Description of Privacy-Preserving Modified Dynamic Conditional Random Field Algorithm (MDCRF) . . . 136
3.7.1 Description of MDCRF Algorithm and Server Process for MDCRF . . . 136
3.8 Experimental Evaluations . . . 140
3.8.1 Experimental Design Principles for Object Detection Test . . . 141

3.8.2 Benchmark Datasets for Object Detection Test 143
3.8.3 Ablation Study for Privacy-Preserving Object Detection Model Tests 144
3.8.4 Privacy Noise Contrast Analysis and Ablation Study For PSNR, MMD, SSIM, and FFT 148
3.9 Summary of Chapter 157

4 Privacy-Preserving Machine Learning Models for Processing Stage 159
4.0.1 Structure of Chapter 160
4.1 Mathematical Notations for Privacy-Preserving Method 161
4.2 Privacy-Preserving Word Embedding for Multiclass Text Classification 164
4.2.1 Independence Calculation 167
4.2.2 Multiple Parallel Classifiers with Support Vector Machine . . 169
4.2.3 Privacy-Preserving Support Vector Machine 173
4.2.4 Privacy-Preserving Stochastic Gradient Descent 176
4.3 Privacy-Preserving Prediction and Independent Frequent Sub-Sequence Extraction Algorithm 178
4.4 Experimental Results for Text Classification 181
4.5 Privacy-Preserving Deep Transformation with Self-Attention 190
4.5.1 Model Architecture and Description 192
4.5.2 Loss Function for PPDPTS 195
4.6 Experimental Evaluation of PPDPTS 198
4.6.1 Comparison Performance of Visual Object Detection Models . 199
4.6.2 The Trade-Off between Privacy Budget and Object Detection Accuracy 204
4.6.3 Privacy Noise Analysis with PSNR, SSIM, MMD, and FFT in Object Detection Tasks 206
4.7 Summary of Chapter 212

5 Privacy Budget Assessment for Post-Processing Stage 215
5.0.1 Motivation of Privacy-Noise Measurement 216
5.0.2 Structure of Chapter 217
5.1 Privacy Feature Search and Privacy Budget Estimation Methods for Image Relative Tasks 218
5.1.1 Extraction of Sensitive Features with Bottom-Up Strategy . . 219
5.1.2 Extraction of Sensitive Features with Top-Down Strategy . . . 223
5.1.3 Experiments for Sensitive Classification and Clustering Algorithm with the Bottom-Up and Top-Down Strategy 228
5.2 Establish the Relationship with Privacy Feature via Expectation and Maximization Method 230
5.2.1 Motivation of the Expectation and Maximization Privacy Assessment(EMPA) Algorithm 231
5.2.2 The Process of Interference Raw Datasets with Expectation and Maximization Privacy Assessment (EMPA) Method 232

5.3 Experimental Evaluation of the EMPA with BUA, TDA Methods . . . 244
5.3.1 Ablation Study of the EMPA with BUA, TDA Methods 244
5.3.2 Horizontal Evaluation of Privacy Measurement Methods . . . 247
5.4 Summary of Chapter 253

6 Conclusions **256**

References **260**

Appendices **272**

List of Tables

1.1 The main types of privacy data and the potential risks for leakage . . . 18

2.1 Five private data protection algorithms 32
2.2 Exemplar metadata with record distance indices. 50
2.3 Basic form of confusion matrix 69

3.1 Example of frequent itemsets for label tags 115
3.2 RMSE average value for PSNR, SSIM, MMD, FFT with MDCRF algorithm 156

4.1 Comparison of classification accuracy rates of PDPIFSEA(with WECPPSVM), Autoencoder (without privacy protection), and ppSVM based on the COVID-19 training dataset 189
4.2 Accuracy of visual object detection on MOT datasets 201
4.3 RMSE value with PSNR, SSIM, MMD and FFT for PPDPTS 211

5.1 RMSE value with Chi-square test, K-L divergence, MMD, and EMPA 247

List of Figures

1.1 The similar background of digital images 23
1.2 The process for privacy-preserving in machine learning 26
1.3 Reduce the privacy leakage with incremental noise 26

2.1 The relationship between protection methods 33
2.2 The relationship between local model and global model 40
2.3 The collaborative attacks with the privacy information 42
2.4 Adjacency graph of inference results and relabelling for privacy-cost computation: (a) inference results or sensitive labels as graph G, (b) adjacency matrix A under the original labelling, (c) same graph G after relabelling, (d) adjacency matrix $\widetilde{A}$ with 1s more concentrated (band form). Relabelling reduces the bandwidth of the matrix; in this thesis, path lengths in a spanning tree on G are then used to compute privacy cost and to link to the privacy budget ε (Section 2.4.6). 53
2.5 Paragraph vector model with privacy preservation 63
2.6 Explaination for ratio with intersection over union(IOU) 71

3.1 The process of privacy-preserving machine learning model 84
3.2 The input streaming data on MDCRF 114
3.3 Outdoor images: The original image (top), the image shows the detected objects (bottom) 141
3.4 Outdoor images before hiding the sensitive features (top) and after detecting the visual objects (bottom) 142
3.5 Comparison of χ^2 values between using the original Wiener space means and private means in the Gaussian sampling with different ϵ . . 145
3.6 Visual object detection models without MDCRF 147
3.7 Visual object detection models with MDCRF 148
3.8 PSNR metric compares several privacy noises 151
3.9 SSIM metric compares several privacy noises 152
3.10 MMD metric compares several privacy noises 154
3.11 FFT-based multi-object tracking evaluation on MOT17: (a) MOTA and IDF1 comparison; (b) tracking robustness; (c) MOTA vs. speed; (d) ablation study. 155

4.1 The proposed privacy-preserving word embedding scheme 166

4.2 Category or decision boundaries of the WECPPSVM classifier in the kernel space: the figure illustrates how samples of different classes are separated and how the classifier partitions the feature space into mutually exclusive regions for the parallel classifiers discussed in this section. 171
4.3 The goal of clustering optimization 172
4.4 RMSE of the WECPPSVM with training COVID-19 datasets using with and without PDPIFSEA algorithm 183
4.5 Classification prediction matrix for text classification with WECPPSVM 184
4.6 The process of the word embedding combined with privacy-preserving support vector machine(WECPPSVM) 186
4.7 Performance comparison of PDPIFSEA for different corpora 187
4.8 Relationship between PDPIFSEA acceptance probability and threshold 188
4.9 Architecture of the PPDPTS model 193
4.10 The accuracy rates of visual object detection with training epochs (a) mAP, (b) AP_{50}, with training data volume (c) mAP, (d) AP_{50}(1) 202
4.11 Advanced visual object detection analyses: (a) mAP vs. model complexity (parameters) on MS COCO for transformer-based detectors, (b) category-wise mAP@0.5 on DOTA for several detectors, (c) AP vs. object size on MS COCO, (d) impact of data augmentation on detector accuracy. 203
4.12 Accuracy rates of PPDPTS for different privacy budgets 204
4.13 PSNR metric comparing privacy noise in PPDPTS with several other privacy noises 207
4.14 SSIM metric comparing privacy noise in PPDPTS with several other privacy noises 208
4.15 MMD metric comparing privacy noise in PPDPTS with several other privacy noises 209
4.16 FFT metric comparing privacy noise in PPDPTS with several other privacy noises 210

5.1 Illustration of the bottom-up strategy (BUA) integrated into a YOLO-style detector. Low-level features are progressively aggregated along the feature pyramid, and the bottom-up attention modules provide feedback weights to highlight candidate sensitive regions/features for subsequent privacy-budget evaluation. 221
5.2 Illustration of the top-down strategy (TDA) with PPDPTS-style tracking. High-level feature representations are analysed via a topological pipeline (e.g., point-cloud construction and persistent homology) to produce compact summaries (persistence diagrams) that identify structure-sensitive components; these components are then traced back to the underlying features/regions for privacy-feature localisation and feedback. 225
5.3 Comparison of Classification Accuracy with Top-down and Bottom-up Strategy on YOLO 229

5.4 Comparison of Classification Accuracy with Top-down and Bottom-up Strategy on PPDPTS 229
5.5 The deviation value for different images with the TDA+EMPA and BUA+EMPA algorithms 245
5.6 The privacy budget compared with the TDA+EMPA and BUA+EMPA algorithms. 249
5.7 Membership inference attack (MIA) accuracy. Vertical axis: MIA accuracy (0 = always wrong, 1 = always correct); horizontal axis: random baseline (0.5), TSDCRF (non-DP), and PPEDCRF (DP). Error bars show standard deviation over runs. Lower accuracy indicates better privacy (closer to random guess). 252
5.8 Ablation metrics for TSDCRF and PPEDCRF. Left: rMSE (root mean square error); right: member–non-member distance. 253

Publications

Ma, B., Wu, J., *et al.* (2020). Combating hard or soft disasters with privacy-preserving federated mobile buses-and-drones-based networks. *Proceedings of the IEEE International Conference on Information Reuse and Integration for Data Science (IRI 2020)*. Chapter 1.1, Chapter 2.3.1.

Ma, B., Wu, J., Song, S., & Liu, W. (2020). Assuring privacy-preservation in mining medical text for COVID-19: NLP perspective. *Open Journal of Internet of Things*, *6*(1). Chapter 4.

Ma, B., Wu, J., Lai, E., & Hu, S. (2021). PPDTSA: Privacy-preserving deep transformation self-attention for object detection. *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2021)*. Chapter 4, Chapters 4.5–4.6.

Ma, B., Yan, W. Q., Lai, E. M.-K., & Wu, J. (2021). New noise-generating method based on Gaussian sampling for privacy preservation. *ISGV 2021*. Chapter 3, Chapters 3.1–3.7.

Ma, B., Wu, J., Lai, E., & Yan, W. Q. (2023). Privacy-preserving word-embedding text classification based on privacy boundary constructed by DBN. *Multimedia Tools and Applications*. Chapter 4, Chapters 4.2–4.3.

Ma, B., Wu, J., Yan, W. Q., & Zhang, X. Y. (2024). JudPriNet: Video transition detection based on semantic relationship and Monte Carlo vision sampling. *Intelligent and Converged Networks*. Chapter 3, Chapter 3.8.

Ma, B., Wu, J., *et al.* (2026). Time series dynamic conditional random field for privacy-preserving object tracking. *IEEE Transactions on Computational Social Systems*. Manuscript submitted for publication. Chapter 3.4, Chapter 3.

Ma, B., Wu, J., *et al.* (2026). Privacy-enhanced dynamic conditional random field for sequence image segmentation and detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*. Manuscript submitted for publication. Chapter 3, Chapters 3.4 and 3.8.

Acknowledgements

I am very pleased to acknowledge the assistance and encouragement I have received from everyone in the production of this thesis. First and foremost, my thanks go to my Ph.D. supervisors, Catherine Shi, Minh Nguyen and Wei Qi Yan at the Auckland University of Technology, Jinsong Wu at the Guilin University of Electronic Technology, China, and the University of Chile, Chile. I appreciate their advice on my research proposal and for their advice on processing the data including the selection and use of suitable statistical techniques. I appreciate the help and advice in designing the experiments and guiding me in writing my thesis.

Secondly, I would like to thank my wife Ms. Elsa Deng, and my parents for supporting and encouraging me to overcome problems and challenges during my Ph.D., encouraged and helping me when I was sad.

Thirdly, I would like to thank everyone who helped me during the research phase, including William Liu, Sean Zhang and Peter Chong, from AUT, Shuolin Hu from Northeastern University, USA, Lingjie Shu from the University of Science and Technology of China, Hongjiang Wei from Guilin University of Electronic Science and Technology of China.

Finally, I would like to thank the Vice-Chancellor of AUT for awarding me the Vice-Chancellor's Doctoral Scholarship. This scholarship has given doctoral students in my studies. At the same time, I would like to thank all the leaders, staffs and students of AUT. Thank AUT for remaining working during the epidemic time. Thank AUT for approving my scholarship.

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Bo Ma

March 2026

Signature of candidate

Chapter 1

Introduction

In cloud computing, service providers currently store data in online social networks, and public or private cloud storage offered by private enterprises and government organizations, for commercial or private purposes. This data can belong to private clients or contain sensitive information that can potentially be mined and exploited by attackers.

The term "privacy leakage" is defined as the intentional or inadvertent disclosure of privacy-related information to a third party (Shabtai, Elovici & Rokach, 2012). When sensitive information related to privacy is stolen or lost by hackers, it can be illegally obtained and used for fraudulent purposes. The key distinction between a user and an attacker lies in their access to data. A user has legal access as permitted by the data owner, while an attacker gains access or acquires the data without the owner's consent.

Most publicly reported incidents of privacy leakage involve the exposure of individuals' private information, such as social security numbers and credit card details. These privacy breaches encompass various issues, including the unauthorized use of facial information from social networks, the compromise of sensitive government-issued identification documents, and inadvertent data breaches resulting from data theft on commercial websites. The disclosure of such private information can leave ordinary

citizens vulnerable to telecommunications fraud and other malicious activities. Furthermore, the reputation damage caused by the publicity surrounding these incidents can often be more detrimental than the actual loss of data itself (Ram & Liu, 2006). Therefore, in reality, the problem of privacy leakage is more severe than many people might realize. The following three examples illustrate how private information can be compromised in the context of machine learning.

- In recent years, there has been a growing trend in the healthcare research industry to develop and deploy artificial intelligence systems for use in the healthcare sector. These systems utilize deep learning models to automatically predict and analyze various aspects of patient cases, providing valuable insights (D. Chen et al., 2019). However, it is important to acknowledge that clinical records, user profiles, and biomedical images contain highly sensitive information, giving rise to significant privacy concerns. When utilizing personal privacy data to develop deep learning projects, project providers must ensure strict privacy guarantees are in place. Unfortunately, in many deep learning models that involve clinical trial data, little to no privacy protection methods are implemented, leaving healthcare providers exposed to potential legal actions under HIPAA/HITECH laws (Gostin, 1994). It is crucial for healthcare organizations to prioritize privacy considerations and implement appropriate privacy safeguards when working with sensitive healthcare data.

- On July 7, 2020, security researcher Jeremiah Fowler made a discovery regarding two computer folders of medical records hosted by the artificial intelligence company Cense AI. These records were being used to train their AI model, and the folders contained the personal medical data of numerous users. It was reported that this medical data was being used as training data in artificial intelligence models, but the model itself included features that could reveal sensitive information

through its prediction results (Humphries, 2020). It is important to note that the personal medical data used in Cense AI's AI model was provided by users from EU countries. The European Union has established clear regulations regarding the protection of privacy data through the General Data Protection Regulation (GDPR) (Regulation, Protection, 2018). As a result, Cense AI is potentially at risk of violating the GDPR, which could have legal and regulatory implications for its operations.

- According to the 2021 Google Maps® report (Mohammed, Ayesh & Boiten, 2020), the Google Maps app boasts over 1 billion monthly active users and a combined total of more than 20 petabytes of aerial and Street View imagery. This amount of data is equivalent to approximately 266 years of high-definition (HD) video. These images captured by Google Maps include various elements such as pedestrian faces, street signs, license plates, and landmarks. To address privacy concerns, Google employs privacy-preserving machine learning algorithms to automatically identify and blur privacy-invading features like license plates. However, it is still possible for hackers to exploit remaining identifiable features within Street View images to gain access to private information in the surrounding areas.

  In fact, hackers have been known to identify potential military bases using satellite images. In an effort to address potential privacy breaches, Google has established a dedicated privacy protection team to tackle such issues. While Google offers users privacy protection features, these functionalities do not provide users with the ability to evaluate the extent to which their personal privacy is safeguarded by the company's artificial intelligence models (Mohammed et al., 2020).

These incidents highlight several potential privacy breaches in the context of machine learning. The first case emphasizes the importance of using appropriate

privacy-protection methods to avoid legal consequences: failure to implement or inadequate privacy-protection measures can expose organisations to the risk of legal action.

The second case demonstrates that privacy can be compromised if data are stored and transmitted without proper safeguards in an unprotected AI model, leading to potential legal risks. Because the parameters and features within a deep-learning model are difficult to encrypt, the risk of privacy leakage increases when such AI models are employed.

In the third case, a service provider constructs an AI model using user data, but users are unable to assess the effectiveness of the privacy protection implemented by the provider. Large volumes of Street View images are publicly accessible through this application, including identifiable landmarks and their corresponding geographical information. If an individual takes a selfie that includes their facial features and recognisable landmarks in a city, privacy attackers could determine the person's location through Google Maps, resulting in a privacy breach. Because Street View images are publicly available, preserving privacy through image semantics is particularly challenging.

Table 1.1: The main types of privacy data and the potential risks for leakage

Format of data	Words	Images	Numerical Data
Source of risk	Personal information and descriptions	Objects in images	Personal data
Data Volume	Variable	Huge	Variable
Risk of data privacy leakage via machine learning	High	Medium	High

In addition, the aforementioned examples shed light on different data types associated with personal privacy information. These types mainly include text (words),

images (including videos), and pure numerical data. Exploring the various data formats and their privacy implications is a crucial aspect addressed in this thesis, and it serves as a significant motivation for the research.

Table 1.1 presents an overview of data formats commonly used in machine learning applications. It also provides estimates of the data volume for each format and assesses the risk level of privacy breaches associated with each data type. The risk levels and data-format categories are informed by prior work on privacy-preserving data mining and risk assessment (Aggarwal & Philip, 2008; Shabtai et al., 2012).

1.1 Main Limitation and Motivation for Privacy-Preserving Related Issue in Machine Learning

1.1.1 Main Limitations of the Current State-of-the-Art for Privacy Preservation

Dwork (Dwork, McSherry, Nissim & Smith, 2006) proposed the use of the Laplacian mechanism to protect privacy, while McSherry (Chawla, Dwork, McSherry, Smith & Wee, 2005) introduced the exponential mechanism for privacy protection by outputting discrete features. However, privacy protection based on these mechanisms in machine learning tasks still faces some challenges. The level of privacy protection is determined by the amount of noise added through the Laplacian mechanism. However, this addition of noise can also reduce the utility or accuracy of the predicted results with machine learning. Striking the right balance between privacy and utility poses a major challenge. Additionally, excessive or inappropriate noise and perturbation can lead to inaccurate or unavailable results. In some cases, these mechanisms may overprotect privacy, resulting in distorted or unusable outcomes. For example, Figure 1.3 shows the recognition of

the same street sign under different noise levels. Assuming that the street name on the road sign belongs to private information, adding noise to the street sign pictures can prevent people who want to steal private information from obtaining the street name information. However, if too much noise is added, as shown in the street sign at the bottom, normal machine recognition cannot extract any information from the image and human beings cannot read any words from the street sign.

Moreover, the Laplacian mechanism and the exponential mechanism themselves have high computational complexity, which prolongs the processing time of the tasks. Consequently, neither the Laplacian mechanism nor the exponential mechanism can be directly applied to machine learning models (J. Zhang, Zhang, Xiao, Yang & Winslett, 2012). On the other hand, the Gaussian mechanism inherently provides approximate differential privacy (Dong, Roth & Su, 2022), making it compatible with a broader range of data used in machine learning models. To address the trade-off between privacy and practicality, it is crucial to analyze the privacy boundaries of sensitive features in order to strike a balance between accuracy and privacy in machine learning tasks.

1.1.2 Main Problems for Privacy Leakage in Machine Learning

Data leakage incidents such as those described in Chapter 1 continue to occur. Current privacy protection methods assume that the attack is performed on the user's local machine. For example, the Tensorflow framework (Seetala, Birdsong & Reddy, 2019) which is widely used to train and operate machine learning (ML) models assumes that it is used on a user's local computer. As a result, it is not concerned about leaking the user's sensitive data. However, most of the commercial machine learning training is performed in the cloud, and the prediction results are also released to the public. Without privacy protection, an attacker could potentially mine private information either through the training process or from the trained models. Data uploaded from the local

machine to the cloud for training via the cloud infrastructure has been identified as a possible source of privacy leakage (X. Zhang et al., 2017). The lack of a common measure to assess the quality of data privacy protection has also been noted.

Another issue of concern is the data provenance from the user's perspective. Millions of users engage in posting messages and receiving online advertisements on social networks. Companies often leverage this information to analyze user interests and make product recommendations. They may also aggregate customer data from various sources, unbeknownst to the customers themselves. Consequently, users are unaware of when, where, and how their private data might be leaked. Data provenance plays a vital role in determining whether sensitive information is adequately protected (Keller, Mikkelsen & Rupp, 2012), making it a crucial factor in measuring the quality of privacy preservation. Privacy-preserving methods need to consider the provenance of users' raw data and select the proper privacy protection methods during the machine learning model training to identify the impact on original information and enable the model to trace the cause of any issues.

The reason is that privacy-preserving methods will slow down the training speed (Gao et al., 2021). However, increasingly, information is stored and processed in the Cloud. At the same time, in industry and academia, many scholars and experts have started to pay attention to federated learning (Q. Zhang, Yang & Chen, 2015), because this method can replace the existing centralized machine learning methods that are easy to leak privacy, and federated learning is also a design model that conforms to the distributed system architecture, which makes edge devices useful in machine learning. At the same time, the machine learning model can directly convert the data in the data flow into the workflow through the federated learning framework without worrying about privacy being leaked (Sakuma & Kobayashi, 2010).

To achieve this privacy-preserving goal, this thesis introduces several techniques, including privacy noise sampling, enhancing machine learning models to incorporate

privacy-preserving methods, and measuring levels of privacy preservation. These methods work collaboratively to find the right equilibrium between protecting privacy and maintaining the normal functioning of machine learning tasks.

1.1.3 Motivations for Privacy-Preserving in Machine Learning

A major application of machine learning and deep learning is natural language processing (NLP). For instance, it has been extensively used for sentiment analysis, text summaries, and question answers. The content of the training data sets or even feature sets within the trained model may contain private information that could be leaked.

In natural language processing, texts are transformed into numerical data through word embedding. A high-dimensional vector encodes each word in the text. The vectors (word embeddings) for each word are obtained by training a model with a large number of texts, and it takes into account the surrounding context (words) that the word used. The word2vec (Church, 2017) is a popular algorithm for training word embedding models (Mnih & Kavukcuoglu, 2013). The attacker could target the word embedding models. In 2019, it showed that the membership attack method could be used to attack the personal data sets via some weakness in the word embedding process (Melis, Song, De Cristofaro & Shmatikov, 2019).

Another main application of machine learning is computer vision. In particular, visual object detection is able to identify objects from images and videos. Visual object detection can cause privacy issues. For example, in a visual object detection system, the images and videos may be uploaded to the Cloud for classification. Attackers may identify the locations via the objects identified in the images. Figure 1.1 shows two pictures having a similar background. The image on the left is a visual object detection result from an image taken from a vehicle on the road, which shows the relative spatial position of the object in the image. The one on the right is obtained from Google

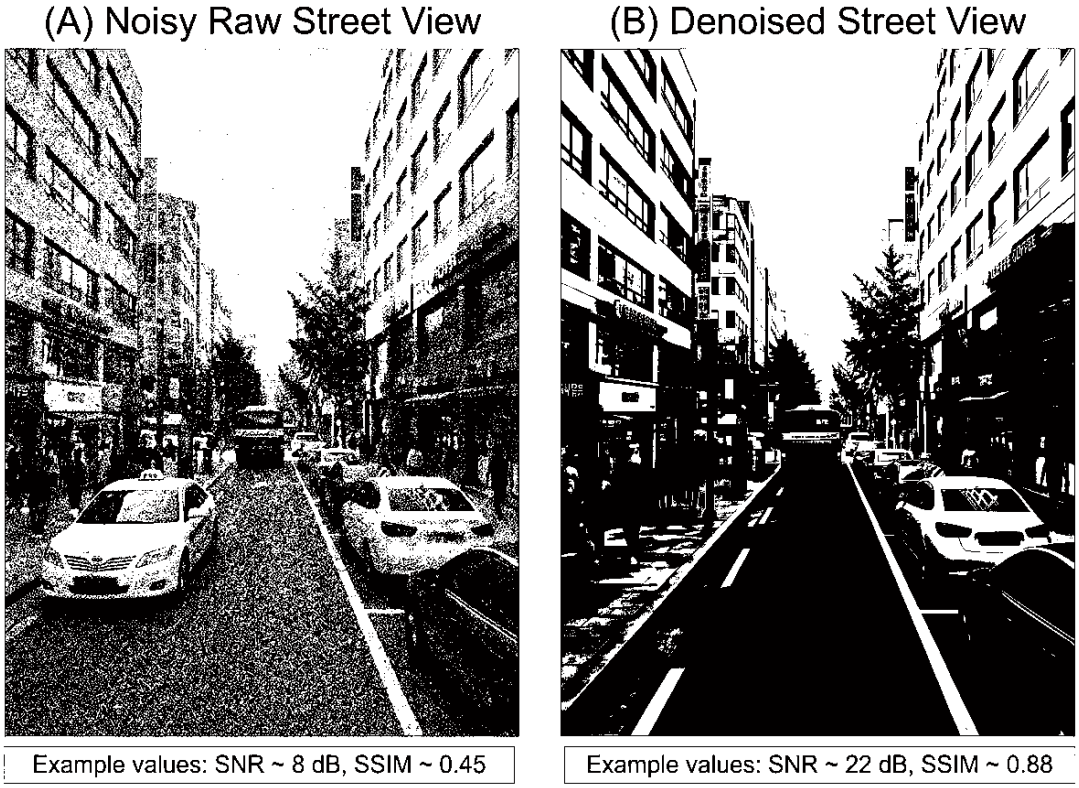


Figure 1.1: The similar background of digital images

Street View. An attacker could match the two images and determine where the vehicle is. Additionally, privacy attackers obtain private information by locating the vehicle's location and attacking with background knowledge. This story shows that privacy leakage from the videos of obstacle detection is a serious concern. However, owing to the sensitivity of visual objects embedded in the background of images, removing the objects will not be a viable option.

Improving the level of privacy preservation in machine learning is a challenging task. Although privacy preservation methods have been extensively studied (Y. Wang et al., 2013; Speciale, Schonberger, Kang, Sinha & Pollefeys, 2019), some fundamental problems remain to be solved. These include quantifying the level of privacy in machine

learning models, quantifying the impact of privacy, and how to protect privacy using some new models.

In the above discussion, machine learning models in different fields are involved with the privacy protection issue, and the tasks that these models need to complete are different. Most importantly, the privacy protection methods for different tasks under different models are also different (Gao et al., 2021). In this thesis, I try to propose some privacy protection methods based on different tasks of different models. For example, some methods can realize the protection of privacy features in machine vision models.

1.2 Motivation Questions

The main aim of this research is to design a privacy-preservation framework for machine-learning operations so that an adequate level of protection can be guaranteed. In other words, this thesis asks: how can differential-privacy-based mechanisms provide privacy-preserving machine-learning models that strike an appropriate trade-off between model performance and privacy protection? The main question to be investigated can be broken down into the following research questions:

1. How to protect data privacy in machine learning models by generating the noise needed for privacy with differential privacy and being able to track or control the addition of privacy noise?

2. How do privacy-preserving methods function in training and predicting machine learning models for different tasks? How to protect training and prediction data for different task types using differential privacy-based privacy-preserving methods?

3. How to adjust the privacy budget with differential privacy according to the sensitivity of the data label and the feedback output of the machine learning

model, so that privacy-preserving methods can adjust their protection level by determining the privacy budget?

The three research questions are related and together support the thesis. Research Question 1 (RQ1) addresses the *pre-processing* stage: how to generate and control privacy noise. Research Question 2 (RQ2) addresses the *model* stage: how privacy-preserving methods operate in training and prediction for different tasks (text and vision). Research Question 3 (RQ3) addresses the *assessment* stage: how to link the privacy budget to data sensitivity and model feedback. The answers to RQ1–RQ3 form a closed loop: noise is generated (RQ1), used in models (RQ2), and the resulting outputs are assessed against the budget (RQ3); if the assessment indicates insufficient privacy, the process returns to the pre-processing stage to adjust the noise. Thus the three questions collectively contribute to the main aim of designing a privacy-preserving framework that balances model performance and privacy protection. Figure 1.2 illustrates this three-layer structure and the flow between pre-processing, model, and assessment.

1.3 Research Framework and Contributions

The framework that is proposed for privacy-preserving machine learning that addresses the research questions is depicted in Figure 1.2. It can be broken down into three layers.

The first layer is the preprocessing layer. Its primary function is to analyze and sample the input data in preparation for training and generate the necessary privacy noise for privacy protection through sampling. A three-step preprocessing method will be investigated. The first step involves Gaussian noise sampling. Gaussian sampling is applied to the original data, and the sampled data is saved for further processing in the next step. In the second step, the Kullback-Leibler divergence is used as an estimator to measure the difference between the distribution of the sampled data and the original data. This measurement serves as an indicator of information loss for subsequent

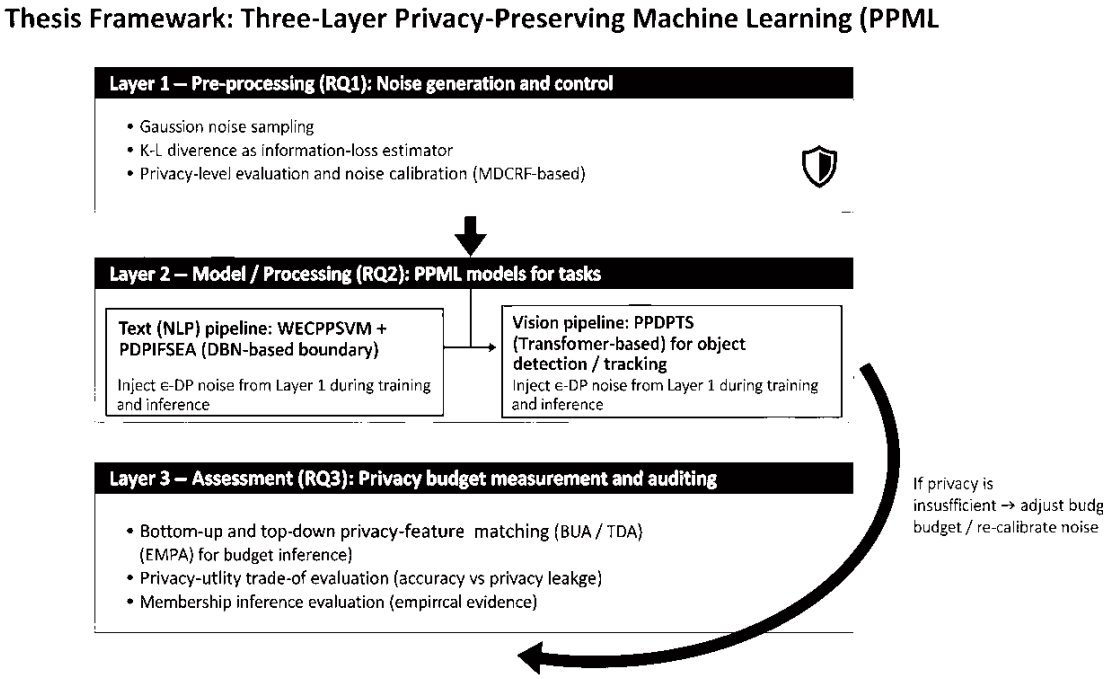


Figure 1.2: The process for privacy-preserving in machine learning

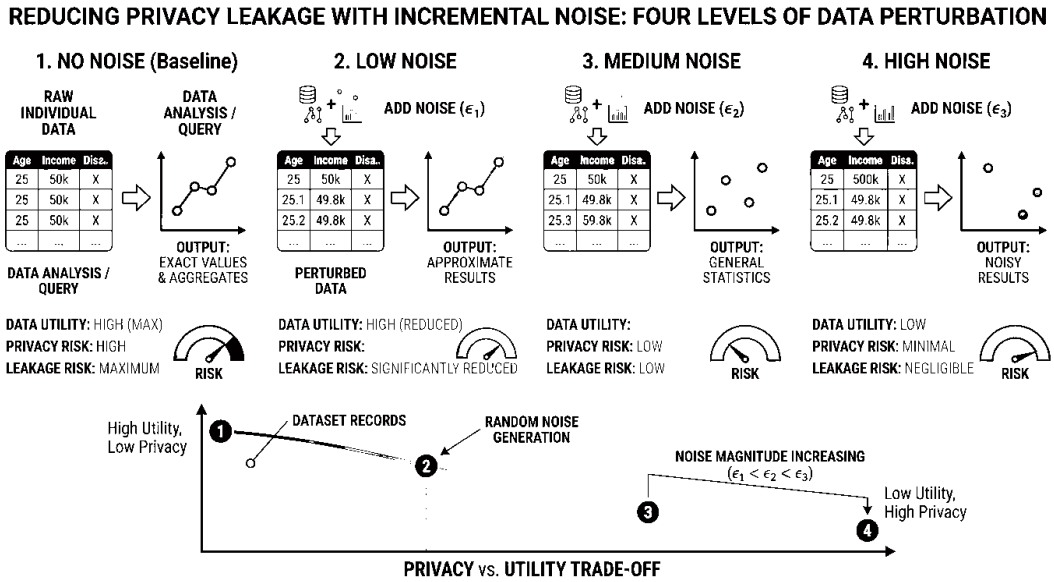


Figure 1.3: Reduce the privacy leakage with incremental noise

training. The third step involves calculating the noise estimate and evaluating the privacy level using the modified dynamic conditional random field. Through these processing steps, the preprocessing layer constructs the privacy noise, thereby forming the privacy-preserving structure utilized in machine learning.

The second layer is the processing layer. In this layer, we will investigate and propose two methods corresponding to two different tasks: natural language processing and computer vision. For natural language processing, this paper introduces a word embedding classification method that combines the privacy-preserving support vector machine (WECPPSVM) algorithm and the privacy-preserving prediction and independent frequent subsequence extraction algorithm (PDPIFSEA) (Ma, Wu, Yan & Lai, 2023). In computer vision tasks, this paper presents a privacy-preserving method based on a model called "PPDPTS" (Ma, Wu, Lai & Hu, 2021). Both of these methods utilize the ϵ-differential privacy mechanism, which enables the models to incorporate privacy noise data generated by the first layer and inject it into the protected data to achieve privacy protection.

The third layer is dedicated to privacy-preserving privacy level measurement and output. Its purpose is to assess the quality of model inference and determine whether the output of the model meets the desired privacy level. This thesis proposes several approaches to measure the privacy level within a specific privacy budget. These approaches include both bottom-up and top-down strategies, where algorithms search for privacy features within groups of sensitive labels. Additionally, the labels of sensitive classes are used to verify the presence of sensitive features. Furthermore, this thesis introduces the use of the expectation and maximization privacy assessment (EMPA) method to infer the final output. This method calculates the actual privacy budget from the privacy evaluation and verifies whether the model achieves the desired level of privacy protection. If the aforementioned method detects that the degree of privacy protection does not meet the previously defined privacy budget, it can revert back to the

noise generation and sampling stage in the first layer to make appropriate adjustments. The three-step process is then repeated until the prediction result aligns with the privacy budget requirements.

Layers 1, 2, and 3 form a dynamic, cyclic process that adjusts noise samples, and rechecks the output to ensure that the data is secure enough to defend against privacy attacks.

There are four main contributions of this research:

1. A novel method is proposed to generate differential-privacy noise and enhance tracking capabilities for sensitive groups in machine-learning frameworks. This is achieved by calculating a dynamic conditional random field during the learning process (Chapter 3).

2. The thesis utilises deep belief networks to optimise text classification. The relevance between sensitive texts is computed using a privacy-preserving prediction and independent frequent subsequence extraction algorithm (PDPIFSEA), and a privacy-preserving text-classification method combining word embeddings with a privacy-preserving support vector machine (WECPPSVM) is also proposed (Chapter 4).

3. A privacy-preserving deep transformation self-attention (PPDPTS) method is developed to protect privacy in object-detection tasks. This approach leverages the Transformer model to adjust scanning regions using a self-attention mechanism, thereby preserving sensitive information in images (Chapter 4).

4. Two privacy-feature matching methods and a privacy-level detection method are introduced, along with a label-based predictive reconstruction approach for measuring the privacy budget. These techniques align the budget with sensitive classes and support the final assessment layer (Chapter 5).

1.3.1 Published Work Derived from this Thesis

Portions of this thesis are based on work that has been published or submitted for publication. Each publication contributes to the three-layer framework and the associated algorithms. The following list summarises these publications and the thesis chapters to which they correspond:

- Combating hard or soft disasters with privacy-preserving federated mobile buses-and-drones based networks (Ma, Wu, Liu, Chiaraviglio & Ming, 2020). Parts of this work inform the discussion of federated learning in Chapter 2.3.

- New noise generating method based on Gaussian sampling for privacy preservation (Ma, Yan, Lai & Wu, 2021). The dynamic noise-generation method and related analysis are adapted in Chapter 3.4 and Chapter 3.8.2.

- A privacy-preserving word embedding text classification model based on a privacy boundary constructed by a deep belief network (Ma et al., 2023). The PDPIFSEA and WECPPSVM text-classification pipeline is adapted from Chapters 4.2–4.4.

- PPDTSA: Privacy-preserving deep transformation self-attention framework for object detection (Ma, Wu et al., 2021). The object-detection framework and privacy-preserving self-attention design are extended in Chapter 4, particularly Chapters 4.5–4.6.

- JudPriNet: Video transition detection based on semantic relationship and Monte Carlo vision sampling (Ma, Wu & Yan, 2024). Concepts from this work influence the treatment of video datasets in Chapter 2.

- The privacy-enhanced dynamic conditional random field for sequence image segmentation and detection (under review by IEEE TCSS). Elements of this work contribute to the MDCRF-based visual model in Chapter 3.7.

1.4 Outline of the Thesis

The organization of this thesis is as follows. Chapter 2 provides a comprehensive review of existing approaches to data privacy protection. It specifically examines the research advancements in privacy-preserving federated machine learning models. The chapter also explores relevant theoretical aspects of privacy protection and discusses background knowledge related to privacy-preserving research in the field of machine learning. Furthermore, it introduces relevant background knowledge from experimental studies conducted in the context of privacy-preserving machine learning. Chapter 3, Chapter 4, and Chapter 5 respectively focus on the first, second, and third layers depicted in Figure 1.2. Finally, Chapter 6 serves as a summary of this thesis. It highlights the contributions and priorities of the current work and outlines potential directions for future research.

Chapter 2

Literature Review

2.1 Structure of Chapter

The content of this chapter is based on a description of relevant research that serves as the foundation for subsequent studies. In Chapter 2.2, it provides an elaboration and explanation of privacy protection methods, with a specific focus on the advantages and disadvantages of the differential privacy method. This discussion supports the motivation behind this thesis. Additionally, in Chapter 2.4, it delves into the methods of federated learning used in this thesis, providing further explanation. Furthermore, Chapter 2.5 explores the research and application of machine learning with privacy protection in existing areas, presenting the pros and cons of related methods. This section contributes to the research conducted in this thesis. Lastly, Chapter 2.6 elaborates on the background technology and knowledge relevant to the experiments, as well as the design principles guiding the experiments used in this thesis. This discussion supports the experimental design implemented in this research.

2.2 Traditional Privacy Protection Methods and Differential Privacy Algorithm

Table 2.1: Five private data protection algorithms

	Randomization	k-anonymity and l-diversity	Distributed privacy preservation	Downgrade effectiveness of data mining results	Differential privacy
k-anonymity (Samarati & Sweeney, 1998)		$\surd$			
l-diversity (Machanavajjhala, Kifer, Gehrke & Venkitasubramaniam, 2007)	$\surd$	$\surd$		$\surd$	
ϵ-differential			$\surd$		$\surd$
(a, d)-diversity (Q. Wang & Shi, 2009)	$\surd$			$\surd$	
t-closeness (N. Li, Li & Venkatasubramanian, 2007)	$\surd$		$\surd$		

The five categories (Aggarwal & Philip, 2008) show the existing privacy-preserving data mining methods:

- Randomization methods (distributed),
- Distributed Privacy-Preserving (distributed),
- Differential privacy (incomplete),
- Statistic based l-diversity methods (distributed),
- Hidden data method k-anonymity(hidden).

These methods conceal sensitive attributes in a way that makes it difficult to trace them back to their provenance. However, it is important to note that the final processed result of these data may still retain some portion of the original user's information.

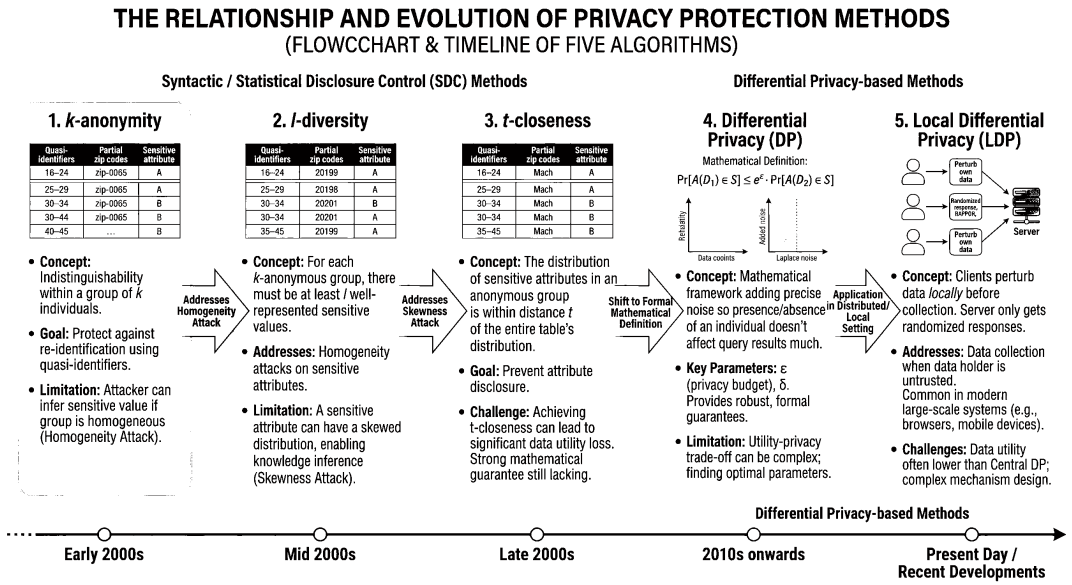


Figure 2.1: The relationship between protection methods

Table 2.1 displays the top five prominent privacy-preserving algorithms, most of which employ a combination of methods. For instance, l-diversity employs data randomization and partitioning to protect data and obfuscate records for privacy attacker tracking prevention. Figure 2.1 presents a timeline depicting the development of these five algorithms and the interrelationships among them. Four of these algorithms originated from k-anonymity, while ϵ-differential privacy was developed independently.

The statistic-based l-diversity and data-hidden-based k-anonymity algorithms operate under the assumption that no sensitive information attributes are leaked to potential attackers. Failure to meet this requirement would compromise user privacy protection.

In contrast, ϵ-differential privacy (Dwork, 2008) takes a more pessimistic approach, assuming a worst-case scenario where the attacker has access to all sensitive attribute

values except for one record. In this case, only a single record differs between the two data sources, D_1 and D_2, regardless of what the attacker is seeking, the outcome remains the same. Consequently, the attacker cannot derive any sensitive attribute values from the potentially sensitive attributes of the data resource.

The main ideas presented in the ϵ-differential privacy are defined as follows (McSherry & Talwar, 2007).

Definition 2.1 (ϵ-differential privacy)**.** *A randomised mechanism M provides ϵ-differential privacy if for all neighbouring data sets x_1, x_2 (differing in one record) and for any measurable set $L \subseteq \mathrm{Range}(M)$,*

$$\mathbb{P}(M(x_1) \in L) \leq e^{\epsilon} \cdot \mathbb{P}(M(x_2) \in L) \tag{2.1}$$

where $\epsilon \geq 0$ is the privacy loss (budget); smaller ϵ means stronger privacy. Here $\mathrm{Range}(M)$ denotes the output space of M.

Because the differential privacy mechanism is easy to implement and can be used in complex scenarios such as machine learning, this algorithm has demonstrated strong resilience since its proposal. Furthermore, significant advancements and breakthroughs have been made at the theoretical level of differential privacy algorithms.

From the perspective of theoretical progress, to make the differential privacy method have higher privacy protection performance for different data sets, the previous research (Dwork, 2008) has proposed modification. In detail, the dataset will process via a sanitization process which is essentially the addition of noise and sanitization process applying the Laplace mechanism. The result of a data search will produce $\Theta(X) = f(X) + Y(X)$ where $Y(X)$ is the added noise, and $f(X)$ is the noiseless search result. Typically, noise with a zero-mean Laplace distribution with values between $[0, \frac{\Delta f}{\epsilon}]$ is used. In this formulation, Δf denotes the *sensitivity* of the query function f (the maximum change in f when one record is added or removed), and ϵ

is the differential privacy parameter (privacy budget). The symbol θ used elsewhere in this chapter for a generic mechanism is here replaced by ϵ for consistency with Definition 2.1.

Define the noise as $Noise(b) = e^{-\frac{\omega}{b}}$ following a symmetric exponential distribution with standard deviation $\sqrt{2}b$, where $b = \Delta f/\epsilon$; the probability density function is

$$p(x) = \frac{\epsilon}{2\Delta f} e^{-\frac{|x|\epsilon}{\Delta f}} \tag{2.2}$$

Symbols in Eq. (2.2): $p(x)$ is the probability density of the Laplace noise at value x; ϵ is the privacy budget; Δf is the sensitivity of the query f; $|x|$ is the absolute value of the noise. The scale of the Laplace distribution is $b = \Delta f/\epsilon$. In Eq. (2.2), if the value of Δf is small, the algorithm performs well due to low noise. As ϵ decreases, the Laplace curve flattens and the expected noise magnitude increases.

For privacy bounds, ϵ-differential privacy(ϵ-DP) provides specific privacy guarantees that quantify the level of privacy protection offered by a machine learning model. The privacy bound is represented by the privacy budget ϵ (Dwork, Kohli & Mulligan, 2019), which determines the maximum allowable privacy loss when individual data is included or excluded from the model's training data.

Regarding ϵ-differential privacy budget ranges, $\epsilon = 0$ signifies perfect privacy, where the model's output is completely independent of any individual's data. No specific information about individuals can be inferred from the model's output. On the other hand, if $\epsilon > 0$, as ϵ increases, the level of privacy protection decreases. Larger values of ϵ result in a higher degree of privacy loss, indicating that the model's output may reveal some information about individuals in the training data. However, ϵ-differential privacy ensures that privacy risks remain within acceptable limits even when ϵ is nonzero. The choice of ϵ depends on the desired balance between privacy and utility. Smaller ϵ values provide stronger privacy guarantees but may lead to less accurate or useful models.

Conversely, larger ϵ values allow for more accurate predictions but increase the risk of privacy leakage. For example, $\epsilon = 0.01$ offers higher security than $\epsilon = 0.1$ in terms of privacy protection. This implies that, under the same privacy attack, $\epsilon = 0.01$ is less susceptible to privacy leakage.

In specific experiments and usage scenarios of this thesis, the specific privacy scope and its interpretation vary depending on the scenarios, datasets, and privacy requirements. When users select a privacy budget, they must consider the trade-off between privacy and accuracy, as well as the sensitivity of the data, in order to choose an appropriate value for ϵ. In addition, privacy-preserving machine learning schemes aim to achieve higher privacy protection ability while using lower privacy costs (referring to a larger value of ϵ). Simultaneously, they strive to maintain high model accuracy.

In machine learning tasks, differential privacy (DP) in distributed online learning scenarios has been introduced to protect data privacy during online learning (C. Li, Zhou, Xiong, Wang & Wang, 2018), thereby preventing privacy attackers from obtaining important sensitive information through its statistical means.

Compared with the traditional five methods, the progress of differential privacy is remarkable, differential privacy provides a better solution for protecting privacy in a distributed machine-learning setting because it does not break the data itself or change the structure of the datasets (Ji, Lipton & Elkan, 2014). All that is required is to measure the amount of noise in input data sets with the same or similar probability distributions.

Pros and cons for differential privacy-preserving. The concept of differential privacy, proposed by McSherry and Talwar (Chawla et al., 2005), introduces an exponential mechanism for privacy protection. Building upon this work, Dwork et al.(Dwork, McSherry et al., 2006) demonstrate that differential privacy can utilize the Laplacian mechanism to enhance privacy preservation when the output of a privacy query consists of numeric data. The exponential mechanism described in(Chawla et al., 2005), on the

other hand, is applicable when discrete privacy matching is required. Both mechanisms are suitable for input data that comprise digital features, such as pixel features in images or semantic relationship features derived from segmented and annotated text, thereby providing significant support for privacy protection in machine learning. However, neither the Laplace mechanism used by Dwork nor the exponential mechanism proposed by McSherry can be directly employed in machine learning models (J. Zhang et al., 2012). This limitation arises because the training process of machine learning models necessitates the use of regression analysis to ensure convergence. Since both mechanisms require sensitivity analysis on the protected target to determine its privacy boundary, and the input data may undergo modifications during processing within the model during machine learning, these modifications bring considerable complexity to the relationship between the input and output of the model during regression. Consequently, analyzing the privacy boundary in machine learning becomes significantly challenging.

In general, the existing differential privacy and other technologies still have the following problems in machine learning tasks:

- Existing differential privacy protection methods involve modifying the input or output data, making it challenging to assess the impact of differential privacy on data modification. Consequently, it becomes difficult to determine and evaluate the privacy boundary in machine learning tasks.

- The original Laplacian mechanism used in current differential privacy methods introduces random noise sampling to the existing data. However, this approach makes it easier to identify and remove the noise based on statistical distribution, thus compromising the effectiveness of the privacy protection mechanism and increasing the risk of privacy information leakage.

Based on the above two challenges, it is determined that in privacy-preserving

machine learning tasks, determining the privacy boundary in the privacy protection method and strengthening its statistical security in the protection method have become two important factors, and these two factors will be discussed later in research and research.

2.3 Privacy-Preserving Federated Learning

Federated or collaborative learning is a machine learning technique that trains ML models on a central server using multiple decentralized torrent tools or local data models without converting them to data models (Konečnỳ, McMahan & Ramage, 2015). This approach makes up for the lack of privacy protection in traditional centralized machine learning because centralized machine learning needs to upload all data samples to a single server. In addition, studies have shown that more classical decentralized systems can help the local data model to present a uniform distribution, and can also keep the training of each node synchronized and effective in terms of machine learning effectiveness (Y. Hu, Niu, Yang & Zhou, 2019).

Federated deep learning attempts to learn models (Delaigle & Hall, 2010) from data distributed across nodes. In federated learning, since the data is kept locally, each node usually generates data with a different distribution, and these nodes are also different in terms of model characteristics. However, although the characteristics of different models are different, there may be similarities between different models. For example, when a person uses a mobile phone, the calls between each person may be different between each mobile phone, but the same communication protocol will be used so that the call can be carried out smoothly.

2.3.1 Advantage for Federated Learning

Although federated machine learning can help preserve data privacy, there is still a privacy leak problem during node exchange because model parameters are exchanged between nodes during training. Inspired by this observation, differentially private federated deep learning has been proposed (Konečnỳ et al., 2015). It aims to build learning algorithms that provide strong (mathematically proven) privacy protection for training data, but it also suffers from privacy leaks in terms of stochastic gradients.

Specifically, deep learning models typically employ stochastic gradient descent (SGD) as a supervised training algorithm (Bottou & Bousquet, 2008). It is a form of gradient descent optimization where the true gradient (computed from all data points) is replaced by an estimate computed from a subset of the data points. Recent work has shown that distributed choice of SGD is vulnerable to inference attacks such as impersonation attacks and membership attacks using malicious servers/clients. As the public model is updated based on this private data, attack patterns are encoded in the model parameters. Therefore, individual data or statistics can be recovered in reverse if a suitable decoder can be created.

To enhance privacy protection in federated machine learning settings, Shokri and colleagues developed a Stochastic Gradient Descent (SGD) method called Distributed Selective Stochastic Gradient Descent (DSSGD) (Shokri & Shmatikov, 2015). The algorithm utilizes the data at the nodes to train a local model. The local models are then combined to produce a global model. Therefore, the private training data of the nodes are not shared; only the models trained by them are shared to avoid privacy leakage caused by data upload. After that, DSSGD sends the global model back to each node and updates the local model. Finally, DSSGD performs distributed global updates. This method proposed by Shokri solves the problem in the case of unbalanced data distribution in practical situations.

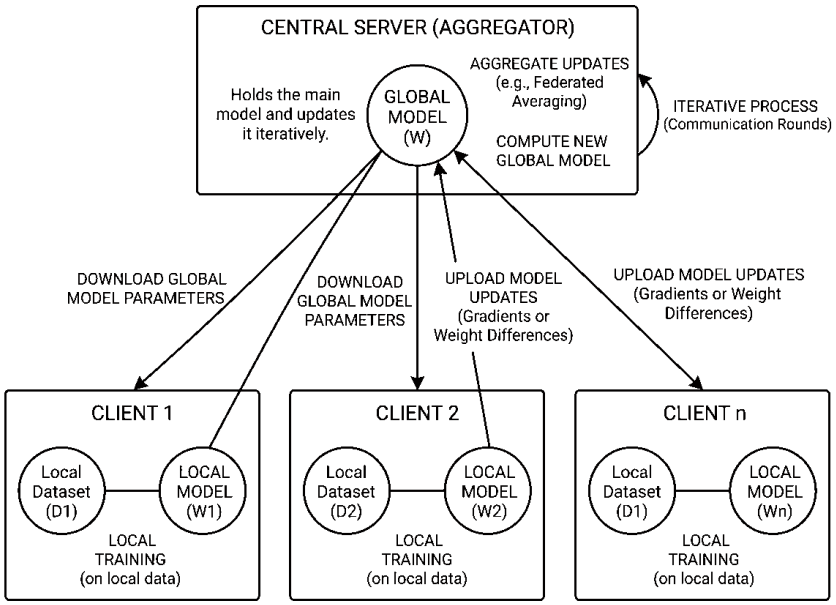


Figure 2.2: The relationship between local model and global model

The algorithm also inherits the concept of updating local parameters from the global model and local model of federated learning (Shokri & Shmatikov, 2015). In federated learning, the relationship between the global model and the local model is shown in Figure 2.2. First, the global server holds the global model, which contains global parameters. Second, the global server sends the global parameters of the global model to all local terminals, and each terminal maintains its own local model, which contains its own local parameters. Third, both the local terminal and the global server perform stochastic gradient descent (SGD) to update their models locally. Fourth, the local model sends its local parameters to the global server. Finally, the server averages the model parameters to generate new global parameters in the global model for the next round of training.

In the work of (Geyer, Klein & Nabi, 2017), Geyer introduces another way to add noise (such as differential privacy) to DSSGD. This approach enhances privacy protection by decentralizing the location of data, and at the same time, this approach

enhances protection by synchronizing client-side and server-side private information. However, noisy data can have an impact on the convergence of the training algorithm. Also, if the amount of data is small, the convergence will be slow and the resulting model will be poor.

It was discovered that pruning a hidden layer of a neural network is equivalent to adding an amount of deferentially private noise. The authors in (Y. Huang et al., 2020) analyzed the noise via Chernoff bound and Hoeffding's inequality. The work in (Y. Huang et al., 2020) connects l-sensitivity and neural network layers noise in the folded Gaussian zone and their results also prove that privacy leakage can be impeded.

2.3.2 Limitation for Federated Machine Learning

The previous section discussed the strengths of federated learning, but its model is also susceptible to malicious attacks. As for the convergence of federated learning, Wei and Li (Wei et al., 2020) mentioned in the paper that the model based on federated learning can quantify the process of calculating the loss function, which proves that the training result is linear regression. It satisfies different levels of differential privacy preservation by appropriately adapting the variance of the noise. Furthermore, they propose new scheduling strategies to optimize the balance between privacy level and data utility. However, their method still lacks test results for data leakage. In addition, the trained model is vulnerable to attacks based on background knowledge (Hitaj, Ateniese & Perez-Cruz, 2017), this thesis will try to solve this problem later.

Bagdasaryan and Poursaeed (Bagdasaryan, Poursaeed & Shmatikov, 2019) analyzed the impact of differential privacy noise introduced in SGD on the model accuracy. Their results show that privacy-preserving training methods, such as rank reduction and noise augmentation, have a disproportionate impact on underrepresented and more complex sample groups. This results in a disproportionate drop in model accuracy.

Figure 2.3 illustrates a server sending a distributed model to each client, with the IoT edge layer and IoT device layer serving each client (Ma, Wu, Song & Liu, 2020). Each individual client utilizes this distributed model to train a local model with its own data. Updates to the model are then sent back to the server, and the shared model is improved during the collaborative process. Malicious clients can deceive victims by exposing their private information (Xiong, Li, Han & Cai, 2019). Encrypting the training data is ineffective because the model won't be able to recognize the training data, causing the training to fail. Incorporating appropriate privacy noise can protect the training data and prevent privacy leaks. The privacy noise is already regulated by the privacy budget, and using an appropriate privacy budget helps control the level of privacy protection.

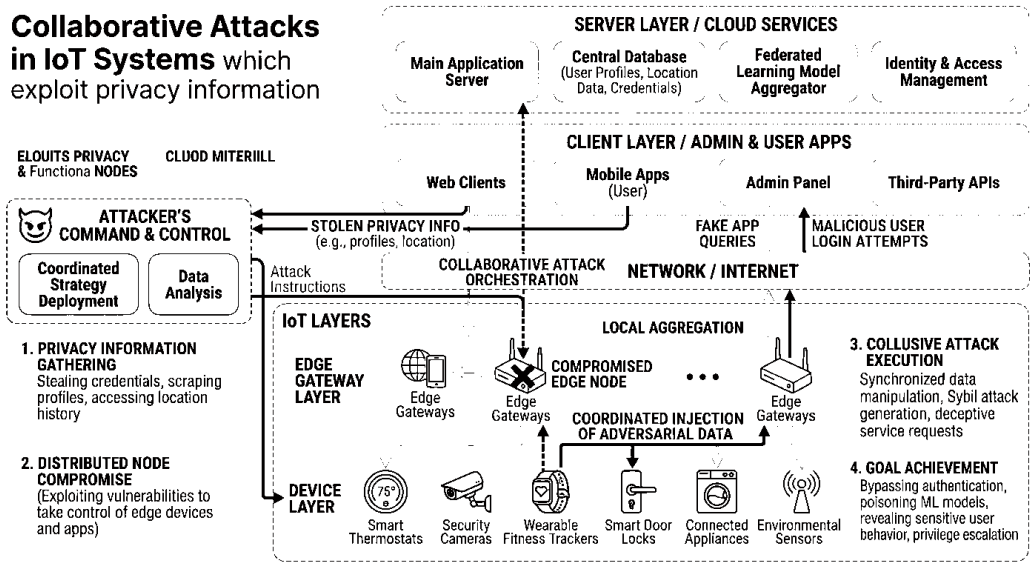


Figure 2.3: The collaborative attacks with the privacy information

All in all, differential privacy in federated deep learning is a hot research topic that is developing at a fast pace, the critical challenge is to reduce the impact of added noise on the accuracy of the trained model. The previous work (Shokri & Shmatikov, 2015) addressed the problem of deep collaborative learning with multiple participants using distributed stochastic gradient descent, however, the existing methods cannot

cope with complex network environments. Deep collaborative learning in a Cloud and local machine setting is certainly a topic of importance. From the perspective of privacy protection, the balance between privacy protection and data output, privacy protection, and training efficiency is equally important. At the same time, ensuring privacy protection during and after machine learning training is as important as the former if in a distributed machine learning environment.

2.4 Methods Related to Privacy Noise Sampling and Measurement

2.4.1 Differential Privacy and Kullback–Leibler Divergence

Ensuring data privacy without compromising the utility of privacy protection from a statistical estimation perspective is a complex issue. The bounds on mutual information and Kullback–Leibler divergence tackle this challenge that influences estimation due to the amount of privacy preserved.

Kullback–Leibler divergence between two probability distributions A and B is given by (Kullback, 1997):

$$KL(A||B) = -\Sigma_{\alpha} A(\alpha) \log \frac{A(\alpha)}{B(\alpha)} \tag{2.3}$$

for all $\alpha \in N$ where N is the sample space.

The variance of Kullback–Leibler divergence is large when the probability of an event in A is high, but the probability of the same event in B is small, this indicates that these events are quite different. On the other hand, if the probability of an event in A is small but that of the same event in B is large, then there is still a significant difference, but it is not as large as in the previous case. In short, understanding the Kullback–Leibler divergence will help to connect with the loss function in the training

process when we want to insert Laplace noise inside the machine learning period to protect privacy.

2.4.2 Differential Privacy and Reproducing Kernel Hilbert Space

(ε, δ) - differential privacy was proposed in (Hall, Rinaldo & Wasserman, 2013). Using this approach, a Gaussian process generates a covariance kernel in the Reproductive Kernel Hilbert Space (RKHS), and the exact noise level is measured through the RKHS. If the learning algorithm needs to estimate the sensitivity to additional noise to preserve privacy, the sensitivity of the specification will affect the machine learning results.

In its research, it defined $\mathcal{H}$ as a Hilbert space and $\{\varphi_i\}_{i=1}^{\infty}$ is an orthonormal basis of $\mathcal{H}$. Any function f (vector) in this space can be represented as

$$f = \sum_{i=1}^{\infty} f_i \varphi_i \tag{2.4}$$

In Eq.(2.4), it has $f = (f_1, f_2, \cdots)^T$.

In their research, it has $K(x, \cdot) = \sum_{i=1}^{\infty} \sqrt{\lambda_i}\psi_i(x)\varphi_i$, which represents a unary function or an infinite-dimensional vector of the x-th row of the matrix. That is,

$$K(x, \cdot) = (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \cdots)^T \tag{2.5}$$

and

$$K(y, \cdot) = (\sqrt{\lambda_1}\psi_1(y), \sqrt{\lambda_2}\psi_2(y), \cdots)^T \tag{2.6}$$

Thus, the inner product gives

$$\begin{aligned} \langle K(x,\cdot), K(y,\cdot)\rangle &= \{\Sigma_{i=1}^{\infty}\lambda_i\psi_i(x)\psi_i(y)\}_{\mathcal{H}} \\ &= \Sigma_{i=1}^{\infty}\lambda_i\psi_i(x)\psi_i(y)(\varphi_i)^2 \\ &= K(x,y) \end{aligned} \tag{2.7}$$

The two functions Eq.(2.5) and Eq.(2.6) are used in the kernel product, which can help to obtain the inner product of $K(y,\cdot)$ and $K(x,\cdot)$. Then, Eq.(2.8) shows the relationship between the original data $K(y,\cdot)$ and $K(x,\cdot)$. It also reveals that the original data is replicated in RKHS and connected with the feature of noisy data.

Hereinafter, the mapping relationship defined in RKHS:

$$< \Phi(x), \Phi(y) >=< K(x,\cdot), K(y,\cdot) >= K(x,y). \tag{2.8}$$

In Eq.(2.8), $\Phi(x)$ and $\Phi(y)$ are mapped to the feature space $\mathcal{H}$ can be expressed as $K(x,y)$. And $\Phi(x)$ does not need to know the space corresponding to this mapping, but $K(\cdot)$ projects an x or y into a symmetric positive definite function $< K(x,\cdot), K(y,\cdot) >$, then this mapping is established.

If $\mathbb{H}$ has been established, noise can be added to protect privacy before training. When detecting $\mathbb{H}$, the privacy level is detected by the noise as well. However, the scenario under the abstract Wiener space will also need to be considered (Osswald, 2003). This could enhance the privacy-preserving level for the Conditional Random Field (CRF) algorithm. In subsequent chapters, the condition for $\mathbb{H}$ will explore, where $\mathbb{H}$ and the input data are used to produce noise in the stochastic gradient descent(SGD) process.

2.4.3 Abstract Wiener Space

In order to calibrate the noise and privacy budget in the generated privacy noise, the computation in terms of the noise measurement can be solved using an abstract Wiener space. Abstract Wiener spaces (Dudley, Feldman & Le Cam, 1971) are the main way to obtain strong positive Gaussian measures in separable Banach spaces. Differential identities of data can be obtained from RKHS or abstract Wiener spaces if a method adds Gaussian noise to the abstraction and shows how to calibrate the noise. At the same time, RKHS and the abstract Wiener space domain can measure the entropy of the input data to be trained. In addition, the abstract Wiener space also provides a measure of privacy preservation effect when adding different levels of noise.

Define $\chi \in \mathbb{G}$ to be a Gaussian function, define a random function $\mathbb{R}$ to calculate proposals, and the distance $f(\chi)$ between its values to be a Euclidean distance. The dataset D is then defined as differentially private (DP) with a Banach space $\mathbb{B}$. $\theta : D \to \mathbb{B}$ represents the Gaussian noise achieved by differential privacy. It has $\theta_D := \theta(\mathcal{D})$. Its purpose is to output a finite number of linear functions in θ_D. Usually, a linear function is applied to a nonlinear transformation, and finally the linear function outputs a set of transformation relations. Specifically, there is $\|f\|_{B*} = sup_{\|h\|_{B*}\leq 1} f(h)$, so that the calculation will form a data space, and the result can also be It is a Sub-Banach space. The pair $(\mathbb{B}, w)$ is called an abstract Wiener space, where w is a probability measure on $\mathbb{B}$ and χ. Each Gaussian process is uniquely parameterized by the mean $\theta \in \mathbb{B}$ and $\Theta : \mathbb{B}* \to \mathbb{B}$, for each $f \in \mathbb{B}$ satisfies the Banach space $\mathbb{B}$:

$$\begin{aligned} &E[f(\chi)] = f(\theta), Cov(f(\chi), f(\theta)) = \\ &E[(f(\chi) - E(f(\chi)))(f(\theta) - E(f(\theta)))] \end{aligned} \tag{2.9}$$

Wiener space can scale the noise level of a set of data in the queue. In addition, the Wiener filter can linearly predict the recently observed background changes to

non-linearly transform the target features in the dataset for further processing.

2.4.4 Deep Belief Network with Privacy-Preserving

Deep belief networks are also widely used in natural language processing and privacy protection. Among several methods in the field of natural language processing, Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM), etc. are combined with deep belief networks. Maximize the conditional probability corresponding to the predicted object. Word semantic prediction of the text, instead of simply extracting words and sentences in the text.

For the word prediction problem using a Deep Belief Network, the improved deep belief networks(DBN) model was proposed by adding the part-Of-speech (POS) node, which samples the words in the training text through the DBN model and predicts the association between words, and Part-Of-speech (POS) is the task of labeling each word in a sentence with an appropriate part of speech within a context and part-Of-speech (POS) node inside of DBN holds the POS label with a certain word of speech (W. Li et al., 2018). From experiments, compared with the conditional random fields (CRFs) method, the accuracy rate of his method (W. Li et al., 2018) improves by 1.47%. In the privacy protection aspect, for privacy-related words, privacy attackers use the prediction method of word association to associate from the same privacy word class to obtain private(sensitive) information. The DBN model can precisely predict the sensitive words(such as clinical records) and extract that privacy information (Phan, Wu & Dou, 2017).

For text classification, Meng Wang (M. Wang, Ning, Xiao & Li, 2018) et al. proposed a word classification algorithm named information geometry deep belief networks (IGDBN). This method solves the problem of sentiment word classification with a large number of label comments in the real world by training the DBN network to predict,

associate, and classify labels and sentiment words. In terms of privacy, the user's own information mentioned in the post can also cause privacy leakage. Such posts will be used by privacy attackers to extract privacy-sensitive words and form users' privacy portraits.

2.4.5 Related Works for Label Search Methods

In supervised machine learning, data labels are a crucial determinant of model accuracy in identifying the class to which the data belongs. Labels also play an important role in identifying sensitive classes that involve privacy concerns. When measuring privacy noise or privacy budget, it is necessary to retrieve these sensitive categories to identify which parts of the raw data contain sensitive information. This allows for the quick detection and localization of privacy levels.

To retrieve label information, vector-based label pairs are often used for classifying label information. In (Behm, Ji, Li & Lu, 2009), an efficient method is presented for performing a similarity-based search for features (words) associated with a given keyword in a tag.

During a tag search, each search is represented by a vector where the frequency of a word in the question corresponds to each element. This vector representation allows for the comparison of features and searches using vector space models. The work mentioned in (Salton, Wong & Yang, 1975) highlights that the comparison process is facilitated by the application table associated with the privacy object victim, improving the efficiency of the comparison. Additionally, the comparison table contains features and searches involving target objects and their information. The weight of each search type in the comparison operation can be obtained from external knowledge and further refined. Keywords that occur less frequently in the dataset are assigned relatively higher weights in the external system because they are considered more salient features.

Furthermore, keywords such as people's names can be extracted from tags, and the information in this table can be accurately matched with these keywords. These methods help convert existing sets of labels into vectors needed for retrieval, enabling indexing and searching of these labels among other works.

For the task of label-vector indexing, various algorithms are available, including suffix trees and metric algorithms that capture the relationships between nodes in spanning trees. The Q-gram technique (Koschke, 2012) can be used to measure path lengths in spanning trees.

Furthermore, the word matching algorithm can also be employed for letter matching to achieve matching between the target text and the original text. However, dictionary and statistical searches have limitations in establishing connections between tagged features when searching for user private information. The information provided by similar words is limited, and private information involves complex semantic relationships that cannot be identified through similar words (Koschke, 2012). To overcome this limitation, Soria-Comas et al. (Soria-Comas, Domingo-Ferrer, Sánchez & Martínez, 2014) proposed the Maximum Average Record Distance (MDAV) algorithm to search and establish associations of sensitive classes, thereby improving retrieval efficiency.

The concept of MDAV can be illustrated using a specific example. The dataset is shown in Table 2.2 comprises five fields sourced from social networks. These fields include the username, gender, online purchase information, and the seller's account name. Three records from this dataset are displayed in Table 2.2. Thus, D_1, D_2, and D_3 represent the index vectors for these three records in the dataset. The features of these records are combined to obtain an average record index denoted as $\bar{d}$, which is associated with sensitive attributes.

In the MDAV method, using the records in Table 2.2, the average record is assigned to a cluster $d_1 \dots d_n$. D_1, the record furthest from the average $\bar{d}$, is assigned to cluster d_1. If D_2 is the most distant record from D_1, it is assigned to cluster D_2. The key

Table 2.2: Exemplar metadata with record distance indices.

Record distance index	Account name	Gender	Sell date	Product name	Provider
D_1	Rbcr	Female	281020-08:01:02	Switch	EB Game
D_2	ThayHuang	Female	141221-11:32:05	Choco	Root
D_3	Fray	Male	281020-12:42:13	Shoes XXL	Hai hoo
$\bar{d}$	Ut*GTF	Ma*Fe	141**-1*:*:0*	Bomb*Shirt	Rb*rs

parameter k, representing the number of distance records considered, impacts the size and quantity of the three classes.

This MDAV algorithm will label their inference results for matching similar features in the label group. The MDAV algorithm is applied to cluster the data label instead of the traditional k-nearest neighbor (kNN) method. In the measurement of relative privacy, the method of MDAV can be used to cluster the privacy classes and associate them with different privacy budgets.

In a similar research to label clustering, Khan et al. (Khan, Wu, Aggarwal & Yan, 2013) proposed clustering based on label similarity, where the method calculates the minimum cost from the query graph to the target graph to perform label matching. However, this method relies on NP-hard problems, making it challenging to calculate the approximate label value in certain cases. Iftikhar et al .(Iftikhar, Wang & Lin, 2020) proposed a framework for micro-aggregated graph anonymization, but it requires pre-anonymization of the values, which is difficult to achieve in the input data of machine learning.

In summary, MDAV provides a search method that combines labels and similarity to

quantify the privacy budget. At the same time, after the MDAV method micro-clusters the labels, it is converted into a vector, and then the distance is calculated in the spanning tree to determine the actual privacy. This value is then compared to a predefined privacy budget value.

2.4.6 Privacy Cost with Privacy Path Length in Spinning Tree

ϵ - Differential privacy ensures that differences between two datasets are preserved (Hay, Li, Miklau & Jensen, 2009). When there are correlations between labeled sensitive classes, such as in road object recognition where faces belong to both driver features and pedestrian features, the similarity can be considered based on the complete record of each corresponding label (R. Chen, Fung, Yu & Desai, 2014).

To detect privacy leaks in existing detection results, previous researchers (Hay et al., 2009) built undirected graphs of detection results to form a connection graph between them. For example, in classification tasks, inference results can be associated with their label information, represented by a matrix. This matrix builds a relational graph by feature labels and has an adjacency matrix. Network graphs that represent relationships among inference results are defined as adjacency graphs. Through this adjacency graph, graph analysis methods can be used to evaluate the effectiveness of differential privacy protection methods. It involves analyzing the privacy attack cost of some nodes through the movement of the privacy attack on the edges of the adjacency graph.

ϵ - Differential privacy ensures that differences between two datasets are preserved (Hay et al., 2009). When there is a correlation between labeled sensitive classes, such as in road object recognition where faces belong to both driver features and pedestrian features, the similarity can be considered based on the complete record of each corresponding label (R. Chen et al., 2014) .

To detect privacy leaks in existing detection results, previous researchers (Hay et

al., 2009) constructed an undirected graph of detection results, forming a connection graph between them. For example, in classification tasks, inference results can be associated with their label information, represented by a matrix. This matrix builds a relational graph through feature labels and has an adjacency matrix. Among them, for relational graphs, a network graph representing the relationship between inference results is defined as an adjacency graph. From the perspective of the relationship between differential privacy and adjacency graphs, graph analysis methods can be used to evaluate the effectiveness of differential privacy protection methods. It involves estimating the privacy budget by analyzing the privacy attack cost of some nodes by computing the distribution of nodes on the edges of the adjacency graph.

Several studies (Hay et al., 2009; Karwa, Raskhodnikova, Smith & Yaroslavtsev, 2011; Gupta, Roth & Ullman, 2012) discuss in depth the conditions and extensions of privacy noise measurement methods for differential privacy based on privacy boundaries. One approach (Hay et al., 2009) applies k edge differential privacy, where an adjacency graph consisting of an adjacency matrix with at most k edges can aggregate any subset of k edges for different sensitivity classes polymerization. By computing all k side information and other relevant information, the true privacy budget can be reverse calculated and estimated.

In detail, the following introduction mentions the generation of a connected graph $\widetilde{G}$ whose adjacency matrix $\widetilde{A}$ makes $|a_{11} - \widetilde{a_{11}}| * |a_{12} - \widetilde{a_{12}}| \ldots |a_{nn} - \widetilde{a_{nn}}|$ and $\widetilde{A}$ are minimized. If $\widetilde{A}$ is equal to A, then the matrix has $\sum_{i=1}^{|M|} \sum_{ji=1}^{|M|} |a_{ij} - widetildea_{ij}| = 0$. Or, if $\widetilde{A}$ is completely different from A, then $\sum_{i=1}^{|M|} \sum_{j=1}^{|M|} |a_{ij} - \widetilde{a_{ij}}| = |M|(|M|+1)$. Figure 2.4 illustrates this idea in a form used in this thesis: inference results or sensitive labels form a graph G; relabelling its vertices yields an adjacency matrix with nonzero entries closer to the diagonal (band form); path cost in a spanning tree on G is then used to compute privacy cost and to link to the privacy budget ε.

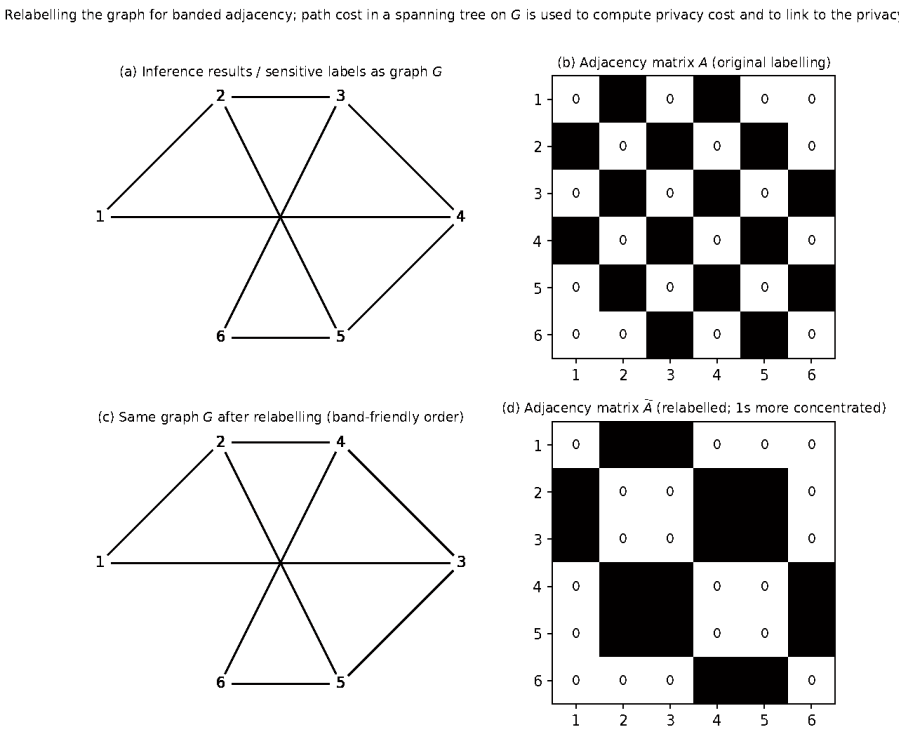


Figure 2.4: Adjacency graph of inference results and relabelling for privacy-cost computation: (a) inference results or sensitive labels as graph G, (b) adjacency matrix A under the original labelling, (c) same graph G after relabelling, (d) adjacency matrix $\widetilde{A}$ with 1s more concentrated (band form). Relabelling reduces the bandwidth of the matrix; in this thesis, path lengths in a spanning tree on G are then used to compute privacy cost and to link to the privacy budget ε (Section 2.4.6).

The difference between two graphs G and $\widetilde{G}$ can be measured by the Kullback-Leibler divergence (Kifer & Gehrke, 2006) between their respective graph distributions as Gp_{KL}. It is given by

$$Gp_{KL}(F(G), F(\widetilde{G})) = \log \frac{F(G)[t]}{F(\widetilde{G})[t]} * \sum_{t=0}^{|S|-1} F(G)_t \tag{2.10}$$

If $F(G) = F(\widetilde{G})$, then $Gp_{KL}(F(G), F(\widetilde{G}))$ is 0, so the standard value change to $0 \log 0 = 0$.

The utility loss can be measured by a relative error on the connected graph $\widetilde{G}$. In (Xiao, Bender, Hay & Gehrke, 2011), it calculate the connection edge from G, the

calculation process for relative error $relerr(Q_{S,T}(\widetilde{G}))$ is as follows,

$$relerr(Q_{S,T}(\widetilde{G})) = \frac{|Q_{S,T}(\widetilde{G}) - Q_{S,T}(G)|}{\max\{Q_{S,T}(G), S\}} \tag{2.11}$$

Among them, s has a privacy boundary, which allows s to display more search values in the search process and more choices.

When using graph theory to search and establish the relationship between tags, the graph formed by the relationship between tags usually uses dense clusters for vertical search. Among them, a dense cluster refers to a group or collection of data points that are closely packed or close to each other in a high-dimensional space. Assuming most subgraphs contain sufficient edge information, realistic graphs often exhibit characteristics of clustering results.

Matching labels to their corresponding position records in the matrix can make them easier to identify from the adjacency matrix of the graph. Accurate location records along with tag information facilitate this matching process.

Figure 2.4 describes the adjacency matrix based on the labels of the vertices in the graph, as shown in the middle. Another labeling of locations in the same graph results in a different adjacency matrix, shown on the right side of the graph. The numbers 1 on the side matrix are more concentrated than the numbers in the middle. The goal of the first step in graph theory is to relabel the location records of a graph such that the adjacency matrix exhibits such properties.

To measure the privacy level of a node, it is necessary to transform the edges of the connection graph to find the corresponding value (Kifer & Machanavajjhala, 2011; Fortunato, 2010). For example, in the paper (Cuthill & McKee, 1969), the authors use a graph search-based algorithm to traverse the correlations between edges and obtain the amount of information disclosure of nodes, indirectly determining the privacy budget. However, any change in information can significantly impact the search results.

Therefore, the aforementioned algorithms are all based on the assumption that the privacy budget fully covers all edges. Following the concept of graph search, we employ a heuristic algorithm to traverse the relationships between nodes, which enhances the effectiveness of differential privacy measures.

Finally, the privacy cost calculation can be achieved by marking the adjacency matrix formed by the edges corresponding to the fully connected graph, and the values in the matrix can be obtained by measuring the density of nodes in the connected graph. It is important to note that this metric serves a different purpose than crowd detection. If enhancing the privacy measures of certain nodes leads to an increase in overall measurement accuracy, it would suggest that improving the connectivity of these nodes is effective. It is assumed that a greedy algorithm or heuristic algorithm can be applied to traverse and select these valid nodes. In that case, those specific nodes can be labeled in the resulting adjacency matrix, enabling region labeling. Measurement precision enhancements can then be applied to regions formed by these methods.

The related work in this section is based on a brief introduction or description of several methods used in Chapter 3 and Chapter 5 or that need to be compared in experiments, because similar methods may be used for privacy sampling and measurement, so these two chapters use methods are described in this section. In the next section, we will focus on the related work of several privacy-preserving machine learning or deep learning methods in several scenarios and fields.

2.5 Application Areas of Machine Learning for Privacy Preservation

This section mainly reviews the related methods of machine learning used or mentioned later. At the same time, this section will also explain the state-of-the-art privacy

protection combined with machine learning methods involved in this thesis.

DP-based transformers and language models. Recent work has shown that large language models and pre-trained transformers can be trained or fine-tuned under differential privacy. Li et al. (X. Li, Tramèr, Liang & Hashimoto, 2022) demonstrate that large pre-trained language models can be strong DP learners via DP-SGD with suitable hyperparameters and memory-efficient gradient clipping. Yu et al. (D. Yu et al., 2022) propose parameter-efficient differentially private fine-tuning of language models, achieving utility close to non-private baselines (e.g., on MNLI). For natural language understanding with BERT, Qu et al. (C. Qu et al., 2021) apply a local DP variant (dχ-privacy) at different representation stages and use privacy-adaptive pre-training. This thesis complements these lines by focusing on an end-to-end pipeline (pre-processing, model, and assessment) and on vision transformers (PPDPTS) and word-embedding-based text classification (WECPPSVM/PDPIFSEA) rather than DP fine-tuning of BERT/RoBERTa.

Privacy auditing. Auditing differentially private machine learning is essential to verify that deployed systems meet their privacy guarantees. Jagielski et al. (Jagielski, Ullman & Oprea, 2020) use data poisoning attacks to audit DP-SGD and assess how private empirical implementations are. Steinke et al. (Steinke, Nasr & Jagielski, 2023) show that privacy can be audited with a single training run by linking multi-sample add/remove operations to DP and generalization. Nasr et al. (Nasr, Song, Thakurta, Papernot & Carlini, 2021) provide lower bounds on privacy leakage by instantiating concrete adversaries (from black-box predictions to worst-case gradient poisoning). The assessment layer in this thesis (Chapters 3–5) connects mechanism design with empirical privacy–utility curves and attack success rates, and can be compared with these auditing and lower-bound frameworks.

2.5.1 Related Works with Image Protection

Differential privacy protection for image protection (DPIP) protects sensitive information by adding noise to feature points. The classic scenario of DPIP is the application of pre-defined privacy protection of the human face (C. Liu et al., 2021). Suppose the private image U induces a feature matrix v, and the image is sent to an untrusted third party. This third party could extract sensitive (privacy) features from the image. In image processing, the DPIP method uses Gaussian noise sampling to generate a matrix $M(v)$, which records all image features $\mathcal{L}(U)$ processed with privacy noise. The third party now only has access to the modified image. Importantly, the recognition system can still identify the privacy features of the target object through its sensitive (privacy) features, which cannot be reflected on other platforms. Image differential privacy can be achieved if and only if the following condition is met (H. Huang et al., 2020):

$$Pr(\sigma^I) = \frac{\sum_{t=1}^{T} \sum_{i=1}^{|\sigma|} \frac{f(\sigma * [t][i]) - 1}{RP(M_i) - 1}}{|\sigma| * |T|}, \tag{2.12}$$

where σ is the privacy budget and $Pr(\sigma^I)$ is the operation to add the Gaussian noise into the datasets with I as an exponent of noise position in the image. $\sum_{t=1}^{T} \sum_{i=1}^{|\sigma|} \frac{f(\sigma * [t][i]) - 1}{RP(M_i) - 1}$ is to add the noise into each pixel, $RP(M_i)$ is the matrix record the ranking privacy for a sampling of the input data. T is the statistical distribution of the noise data sampled from the original data, and $|\sigma| * |T|$ is the noise generation coefficient that follows this distribution.

2.5.2 State-of-the-Arts with Privacy-Preserving Visual Object Detection Methods

The main objective of the object detection framework is to detect objects while ensuring privacy. Several deep learning methods have achieved very high accuracy and efficiency

in detecting objects in images. Among them, Fast R-CNN (Ren, He, Girshick & Sun, 2016) and Mask R-CNN (He, Gkioxari, Dollár & Girshick, 2017) extract the features of the entire image at the same time. These features are then passed to the spatial pyramid pool and the interest domain pool layer to generate regional features. Region of Interest (ROI) pooling extends the ROI align layer to reduce the error of spatial quantization (Puzicha, Held, Ketterer, Buhmann & Fellner, 2000). These are followed by Faster R-CNN (Ren, He, Girshick & Sun, 2015) which is more computationally efficient than the previous two. The Feature Pyramid Network (FPN) (T. Lin et al., 2017) is later proposed which uses the Faster R-CNN as its backbone network to replace the independent and time-consuming area select method.

In terms of speed and accuracy, the "You Only Look Once" (YOLO) model (Redmon, Divvala, Girshick & Farhadi, 2016; Farhadi & Redmon, 2018; Bochkovskiy, Wang & Liao, 2020; C. Li et al., 2022; C.-Y. Wang, Bochkovskiy & Liao, 2022) is one of the latest deep neural network object detection methods. YOLO is able to perform object detection for videos in embedded devices in real time. Furthermore, it uses the evolved deep intelligence framework to evolve the YOLO network architecture and produce an optimized architecture. Like YOLO, SSD (W. Liu et al., 2016) is also a one-stage and end-to-end object detection method. It directly classifies predefined anchors and use convolutional neural networks for detection. YOLO has incorporated the multilayer prediction function found in SSD and introduced additional context information with deconvolution operators to improve accuracy.

Inspired by the success of using multi-headed attention in Transformer architectures for natural language processing (NLP), many researchers have proposed variants of the Transformer for image classification and object detection. The detection transformer (DETR) (Carion et al., 2020) performs end-to-end object detection. It trains data with a bipartite matching loss and can be easily reproduced since it does not have any customized layers.

The vision transformer (ViT)(Dosovitskiy et al., 2021) interprets an image as a sequence of patches. It applies the encoder of the standard Transformer with an additional learnable classification token(Vaswani et al., 2017) to process the input. To address the weakness of ViT in performing pixel-level dense detection, the Pyramid Vision Transformer (PVT) (W. Wang et al., 2021) was proposed. It effectively detects dense images by replacing the backbone of CNN and utilizing fine-grained image patches to provide high-resolution representation. It incorporates both the progressive shrinking pyramid and spatial-reduction attention to acquire multi-scale feature maps while operating under limited resources.

The dense prediction transformer (DPT) (Ranftl, Bochkovskiy & Koltun, 2021) utilizes the ViT as the encoder and a convolutional network as the decoder. Notably, the per-stage ViT encoders retain spatial resolution within the original embedding.

The use of attention modules is also inspired by the Transformers architecture. There are generally two ways to apply the attention mechanism: per-channel attention and a combination of channel and spatial attention. Squeeze-and-Excite (SE)(J. Hu, Shen & Sun, 2018) demonstrates excellent potential in efficiency through dimensionality reduction. However, capturing dependencies across all channels tends to significantly increase computational complexity. Efficient Channel Attention (ECA)(Qilong et al., 2020) achieves interaction across channels through a lightweight method. Nevertheless, both of these methods disregard the spatial dimension.

The Bottleneck Attention Module (BAM) (Park, Woo, Lee & Kweon, 2018) and Convolutional Block Attention Module (CBAM) (Woo, Park, Lee & Kweon, 2018) adopt both channel and spatial attention to refine convolutional features. While Transformers can detect objects and classify images based on their characteristics, preserving privacy within a privacy budget remains a challenge. This is because it is very difficult to fit differentially private subsampling within the Transformer framework, and it is difficult to improve the accuracy of sampling (Woo et al., 2018).

The above-mentioned Transformer-based models were all released after 2020, and the privacy protection work achieved by combining these latest models is also very rare, but there are still works that have been realized. Xu (R. Xu, Joshi & Li, 2019) describes four categories of methods according to the privacy integration situation, namely data preparation, model training and evaluation, model deployment, and model inference.

Homomorphic encryption and functional encryption can prevent some privacy leaks, but extending them to deep networks and large datasets is still extremely challenging due to the high computational complexity involved (Qi, MaungMaung, Kinoshita & Kiya, 2022). In contrast, Liu (Y. Liu, Ma, Liu, Ma & Ren, 2019) implements privacy protection by combining the Faster R-CNN model, which effectively safeguards privacy targets in target detection. The methods proposed by Qi (Qi et al., 2022) and Hao (Hao et al., 2023) improve the existing VIT (Dosovitskiy et al., 2021) and DETR (Carion et al., 2020) models to achieve privacy protection.

However, reducing the privacy budget can result in the leakage of confidence information (Fredrikson, Jha & Ristenpart, 2015) and privacy gradients, while increasing the privacy budget can decrease the accuracy of the model and have a significant impact on its performance (Hatamizadeh et al., 2022). The subsequent approach of this paper aims to maintain the model's accuracy and ensure robust privacy protection by enhancing the privacy protection method in combination with existing machine vision or natural language processing models.

2.5.3 Privacy-Preservation Support Vector Machine for Text-Based Data Classification Tasks

As can be seen in the previous sections, many researchers combine differential privacy with deep learning to protect trained models from differential privacy attacks. However,

the direct application of random noise in deep learning models yields poor performance due to the high sensitivity of network output to parameters. Therefore, previous researchers proposed adding random noise to the training phase of machine learning model updates to make them privatized (X. Zhang et al., 2021). This protection method can completely eliminate the memory effect and reduce privacy leakage.

When analyzing (ε, δ)-differential privacy, the security performance (Hall et al., 2013; Liao, Yin, Chen & Qin, 2020; Liao, Yu, Li, Li & Qin, 2019; Liao, Li, Zhu & Liu, 2020) has been thoroughly investigated. When using the Gaussian process to generate a covariance kernel in the Reproducing Kernel Hilbert Space (RKHS), the correct noise level can be measured using the RKHS norm function. In machine learning, the "sensitivity" level of additional noise can be evaluated using this RKHS norm function. Additionally, the "sensitivity" level can also be applied in the kernel space within the Abstract Wiener space (Osswald, 2003).

This thesis will focus on stochastic analysis with Abstract Wiener under the stream if it is followed with Gaussian Distribution. In the concept of space, the Banach space $C_{\mathbb{R}}$ exists in the form of several subspaces in the abstract Wiener space $\mathbb{H}$, and the separable Banach space can also be densely embedded by Hilbert space $\mathbb{H}$ and Banach subspace. Using Gaussian distribution to measure $\mathbb{B}$ to form a measure ρ_1, a coefficient with a variance of 1 is obtained. Then, by derivation of Malliavin by the operator $L^2(\mathbb{B},\ rho_1)$, $L^2(\mathbb{B}, \rho_1, \mathbb{H})$ can be obtained. The Bochner square map of this result on the Hilbert space is expressed as $f : \mathbb{B} - \mathbb{H}$: (Chang, Zhao & N'Guérékata, 2011).

In the scheme proposed in this thesis, after embedding the word vector to be processed, its semantics can be spatially mapped in the word vector to reflect the part-of-word-vector association between the word vectors. After the word vector is spatially mapped, its space can also be processed into reproducing kernel Hilbert space by basis transformation. For Abstract Wiener spaces, the measure of ρ_1 on $\mathbb{B}$ can help to separate the sensitive classes from general word vectors (Ambrosio, Miranda Jr, Maniglia &

Pallara, 2010).

2.5.4 Word Embedding for Text Classification

In terms of privacy security protection of the text classification model, the classification model and its corresponding privacy protection methods will be involved. In terms of text classification models, in order to process text with neural networks, many solutions convert words in the text into numerical forms for calculation. One approach is to use a word embedding model to encode words and their context (D. Shen et al., 2018). Another most popular approach is a word embedding model called word2vec, developed by Google (Church, 2017). Word2vec has two methods for capturing contextual information, One is called continuous bag-of-words (CBOW) (Q. Wang, Xu, Chen & He, 2017) and the other is called skip-gram (Bartunov, Kondrashkin, Osokin & Vetrov, 2016). They both use artificial neural networks as their classification algorithms. In the word2vce model, the embedding of each word is a high-dimensional vector.

Since word embeddings are vectors, they can be processed using vector algebra. For example,

$$\begin{aligned} &\text{vector}('\text{Warszawa}') - \text{vector}('\text{Poland}') + \text{vector}('\text{New Zealand}') \\ &= ('\text{Wellington}') \\ &\text{vector}('\text{Boy}') - \text{vector}('\text{Man}') + \text{vector}('\text{Woman}') \\ &= ('\text{Girl}') \end{aligned} \tag{2.13}$$

where "Wellington" and "Girl" are result items respectively. Note that such relations are not obtained using prior knowledge of word semantics, such as those available in WordNet (Pedersen, Patwardhan & Michelizzi, 2004), But the relational classification of word embeddings is done purely using statistical methods. Therefore, word2vec can be thought of as a distributed representation of the vocabulary.

In word embedding methods, a vector representation of the entire document can be obtained by simply summing the embeddings of the constituent words in the document. in mathematics,

$$\vec{v}_d = \frac{1}{\sum_{a_i \in d} f_{a_i}} \sum_{a_i \in d} f_{a_i} \vec{v}_{a_i}, \tag{2.14}$$

Where $\vec{v}_d$ is the vector representing the document, $\vec{v}_i$ is the word embedding of the word a_i, and f_{a_i} is the word in the document frequency of occurrence. The computation of this document vector involves only the embedding and frequency of each word found in the document.

Vectorizing documents in a corpus can be achieved using the paragraph2Vec (Le & Mikolov, 2014)method. Its basic idea is very similar to Word2vec. There are two approaches – Distributed Bag of Words (DBOW) and Distributed Memory (DM) (Hiranandani, Kennedy & Tseng, 1992). DBOW predicts the probability of a set of random words in a given paragraph, while DM predicts the probability of a word by using the given contexts and paragraph vectors. They are illustrated in Figure 2.5.

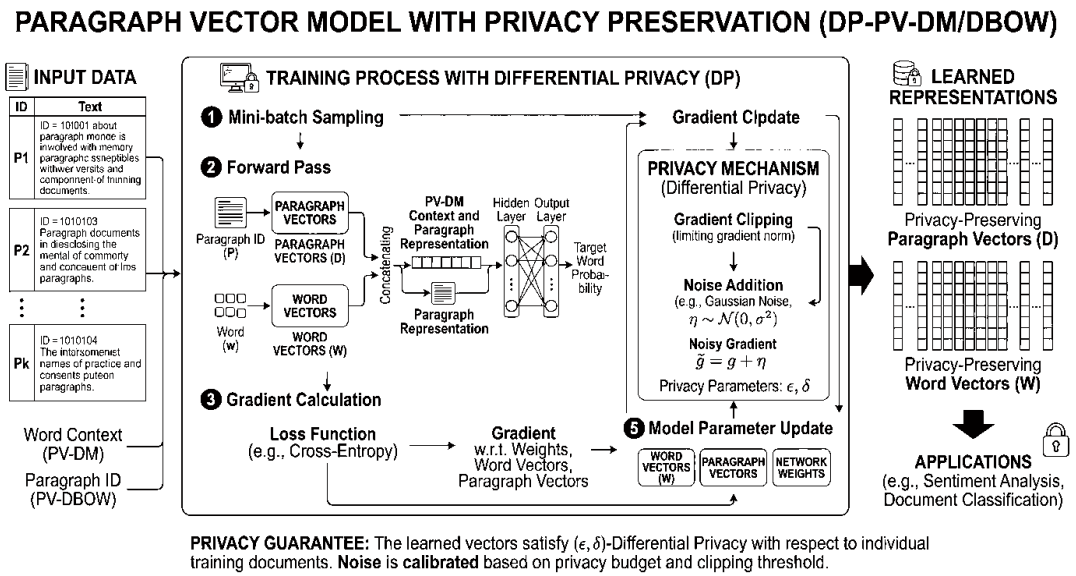


Figure 2.5: Paragraph vector model with privacy preservation

However, methods such as DBOW naturally do not consider the protection of privacy. To remedy this problem, a word embedding method for de-identification and removal of personal private information (PPI) is proposed in the work of (Hassan, Sanchez & Domingo-Ferrer, 2021). The paper (Hassan et al., 2021) found that the distance between word vectors can provide information about private associations. The distance between the privacy factors (such as a name) and an embedding vector can be used to retrieve private information. By measuring distances, word vectors form sets numerically and constitute semantic connections of texts. At the same time, the links formed between vectors will be mined as semantic links, which may cause the text processing model to leak private information. However, this function has natural limitations because the distribution of datasets in numeric format is not the same as that of datasets in text format.

However, methods such as DBOW do not naturally consider the protection of privacy. To address this issue, a word embedding method for de-identification and removal of personal private information (PPI) is proposed in the work of Hassan et al. (Hassan et al., 2021). The research found that the distance between word vectors can provide information about private associations. By measuring distances, word vectors form numerical sets and establish semantic connections within texts. However, the links formed between vectors can be mined as semantic links, potentially leading to the leakage of private information in text processing models. It's important to note that this function has inherent limitations due to the difference in distribution between numeric and text datasets. Taking this as a motivation, in the following chapters, the optimized methods will be proposed from this thesis according to this problem.

2.6 Description of Experimental Evaluation Metrics for Related Work

For privacy-preserving machine learning methods, commonly used evaluation metrics are typically involved to describe the performance of the task model. This section provides a detailed description of the evaluation metrics that may be relevant to this thesis, along with specific details for each metric.

2.6.1 A Central Scheme to Experimentally Measure Privacy Boundaries and Evaluate the Capabilities of Privacy-Preserving Machine Learning Methods

In the general empirical evaluation of machine learning, experiments typically focus on four key aspects: benchmark dataset selection, experimental setup, comparative analysis, and statistical significance. The benchmark dataset selection entails choosing a dataset that is appropriate for the task at hand. The experimental setup involves using suitable evaluation metrics and comparing the results against other advanced or baseline methods. Finally, statistical analysis is conducted based on the specific task, employing statistical tests to validate the experimental results.

For experiments on privacy protection, several concepts need to be explained first. First, a privacy budget is a concept in differential privacy that measures the total amount of privacy leakage allowed during computation or query (Dwork, McSherry et al., 2006). In differential privacy, individual privacy is preserved by introducing random noise. The privacy budget is used to control the use of this random noise, striking a balance between privacy protection and data utility. The privacy budget is usually quantified using a privacy parameter called ϵ. The value of ϵ represents the level of privacy difference between two similar datasets. A smaller value of ϵ indicates a stronger privacy guarantee,

while a larger value of ϵ implies a weaker privacy guarantee. The privacy budget can be seen as a resource that controls the total amount of privacy leakage allowed in a series of computations or queries. Each use of the differential privacy mechanism consumes a portion of the privacy budget. Once the privacy budget is depleted, the differential privacy mechanism can no longer be used to prevent excessive disclosure of personal private information.

On the other hand, the privacy boundary is a predefined limit or threshold for the level of privacy protection provided by the privacy protection mechanism. While a privacy budget quantifies the amount of privacy loss allowed within a system or algorithm, a privacy boundary sets the maximum acceptable level of privacy risk or breach. In experimental settings related to privacy protection, it is generally necessary to set the value of the privacy budget ϵ, which also represents the privacy noise level of the corresponding model. Setting a privacy budget also establishes privacy boundaries.

Managing the privacy budget aims to strike a reasonable balance between privacy preservation and data utility. A smaller privacy budget provides stronger privacy protection but may result in greater data distortion or reduced model utility. On the other hand, a larger privacy budget can enhance data utility but may have some impact on privacy preservation. Hence, careful consideration is required to achieve an appropriate balance between privacy preservation and data utility for a given privacy budget. This ensures desirable data outcomes with suitable privacy protection. Managing the privacy budget should also consider the specific requirements and risk tolerance of the particular application scenario.

Privacy cost is a broader concept that encompasses the expenses or resources involved in privacy protection. It includes various aspects such as technical costs, computational resources, time consumption, data quality loss, and more. Privacy cost is not limited to differential privacy or specific privacy protection methods but encompasses the overall costs and sacrifices associated with the privacy protection process. For

instance, applying a privacy protection technique may require higher computational resources and processing time, potentially affect data quality, and necessitate additional training and expertise. All these factors contribute to the privacy cost. There are several approaches and metrics that can help quantify the privacy cost:

- Utility Metrics With Accuracy: One way to measure the privacy cost is by evaluating the impact of privacy-preserving techniques on the utility or effectiveness of the machine learning model. Utility metrics such as accuracy, precision, recall, or F1-score can be used to compare the model's performance before and after applying privacy measures. A significant drop in utility indicates a higher privacy cost. The experimental concept for accuracy, precision, recall, or F1-score description is in Chapter 2.6.7.

- Information Loss Metrics: Information-theoretic metrics can be employed to quantify the amount of information loss caused by a privacy-preserving method. These metrics assess the privacy cost by measuring the additional knowledge an adversary gains about sensitive data due to the presence of noise or perturbations introduced during the privacy protection process. Examples of such metrics include mutual information, Kullback-Leibler divergence, or Shannon entropy.

- Privacy Budget Consumption: If you are using differential privacy, tracking the consumption of the privacy budget can provide a measure of the privacy cost. The privacy budget, typically represented by ϵ, is depleted with each use of a differentially private mechanism. Monitoring and comparing the remaining privacy budget after different computations or queries can provide insights into the privacy cost incurred by the model.

It's important to note that measuring privacy costs is often a complex task and may require a combination of multiple metrics and evaluation techniques. The specific

choice of metrics depends on the privacy-preserving method being used, the threat model, and the desired level of privacy. Additionally, the evaluation should consider the trade-off between privacy and utility, as well as the specific requirements and constraints of the application or scenario being analyzed.

In the context of differential privacy methods, privacy boundaries refer to the level of privacy guarantee provided by the chosen privacy parameters or mechanisms. The privacy boundaries are quantified using parameters such as ϵ and δ. The ϵ value represents the maximum allowable privacy loss or disclosure, while the δ value corresponds to the probability of any additional privacy breach beyond the ϵ threshold. Smaller values of ϵ and δ indicate stronger privacy guarantees, meaning that the differential privacy mechanism aims to minimize the potential privacy risks associated with the released data. By setting appropriate privacy bounds, organizations can control the level of privacy protection they want to provide to individuals and ensure compliance with privacy regulations or ethical considerations. The choice of privacy bounds depends on the specific context, privacy requirements, and trade-offs between privacy and the utility of machine learning model predictions.

2.6.2 Metrics for Measuring Privacy-Preserving Machine Learning Model Capabilities

In machine learning tasks, the confusion matrix (Heydarian, Doyle & Samavi, 2022) is usually used to summarize the prediction results of the classification model. It shows the summary of the real categories recorded in the data set and the predicted categories of the classification model in the form of a matrix. As shown in Table 2.3, it is the most basic binary classification form of the confusion matrix, where the rows of the matrix represent the real values, and the columns of the matrix represent the predicted values.

In Table 2.3, four indicators are presented: TP (True Positive), which represents

Table 2.3: Basic form of confusion matrix

Actual Value \ Predicted Value	Positive	Negative
Positive	TP	FN
Negative	FP	TN

the number of positive samples correctly predicted as positive; TN (True Negative), which represents the number of negative samples correctly predicted as negative; FP (False Positive), which indicates the number of negative samples incorrectly predicted as positive; and FN (False Negative), which represents the number of positive samples incorrectly predicted as negative. These indicators are used to evaluate the accuracy of predictive classification models.

In most machine learning models for predictive classification tasks, a higher accuracy score indicates better model performance. Ideally, the values of TP and TN mentioned earlier should be high. However, the statistics in the confusion matrix only provide classification results of the model and do not directly represent the quality of the model's predictions. Therefore, secondary indicators corresponding to the confusion matrix are derived to provide a better understanding of the results presented in the confusion matrix.

- Accuracy: Accuracy represents the correct results of the classification model, which is the proportion of TP and TN to the total number of observations. The Eq.(2.15) for accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.15}$$

In Eq.(2.15), TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives. Accuracy is calculated by dividing the sum of true positives and true negatives by the sum of all four

values (true positives, true negatives, false positives, and false negatives).

- Precision: Among all the results predicted by the model as positive, the proportion of the correct predicted result (True Positive), the precision equation is in the following.

$$Precision = \frac{TP}{TP + FP} \tag{2.16}$$

In Eq.(2.16), TP represents true positives, and FP represents false positives. Precision is calculated by dividing the number of true positives by the sum of true positives and false positives.

- Recall: In all samples where the true value is positive, the model predicts the correct result in the recall equation.

$$Recall = \frac{TP}{TP + FN} \tag{2.17}$$

In Eq.(2.17), TP represents true positives, and FN represents false negatives. Recall, also known as sensitivity or true positive rate, is calculated by dividing the number of true positives by the sum of true positives and false negatives.

In addition, this thesis also involves the indicator relevant to precision, they are:

- The error rate of classification(classification) - The error rate of classification is a performance metric that measures the accuracy of a classification model. It calculates the percentage of misclassified samples in a dataset. A lower error rate indicates a more accurate classification model.

- Mean Average Precision (mAP) - mAP is a commonly used evaluation metric in object detection tasks. It measures the precision and recall of the detected objects across multiple categories. Precision measures the accuracy of the detections, while recall measures the ability of the model to find all the relevant objects.

mAP combines precision and recall by calculating the average precision for each category and then taking the mean across all categories. It provides an overall measure of the model's performance in object detection.

- Recall of mean Average Precisions (mAP) under the IoU - This term seems to be a repetition or error in the original paragraph. Recall is a measure of the model's ability to find all the relevant objects in the dataset. It is typically calculated as the ratio of the number of true positive detections to the total number of ground truth objects. However, it is not specific to mean Average Precisions (mAP) under the IoU, as mAP already takes into account both precision and recall.

2.6.3 Intersection over Union (IoU)

In the object detection task, for the recognition of a single object, there is often a situation where the recognition area and the marked area are inconsistent. In order to recognize the overlap area, the concept of intersection over union is proposed from the paper (Rezatofighi et al., 2019).

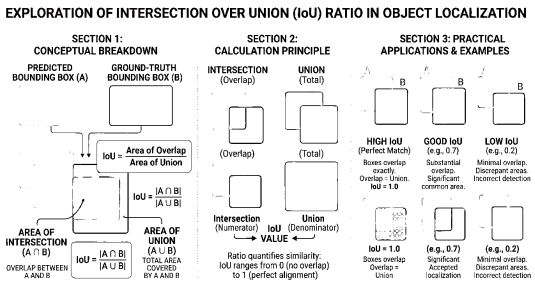


Figure 2.6: Explaination for ratio with intersection over union(IOU)

According to Figure 2.6, it is assumed in the figure that A is the detection result of the model, and B is the original marked area of the object. Then the area where A and B intersect is $A \cap B$, and the area shared by AB is $A \cup B$, so the calculation of IoU is shown in Eq.(2.18).

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.18}$$

Usually, an IoU threshold is set in the target detection task, and when the IoU is usually set to be greater than or equal to 0.5 in the program, the model can be considered to detect the object. Based on IOU, many object detection models also use generalized intersection over union (GIoU) to measure the bounding box coverage (Rezatofighi et al., 2019). The GIOU is a metric used in object detection tasks to evaluate the quality of bounding box predictions. It is an extension of the Intersection over Union (IoU) metric, which measures the overlap between the predicted bounding box and the ground truth bounding box of an object. GIoU not only considers the overlap but also takes into account the size and shape differences between the boxes. It provides a more accurate measure of localization accuracy by penalizing predictions that deviate from the ground truth in terms of size and position.

2.6.4 Experimental Design with Root Mean Square Error (rMSE)

The root mean square error (rMSE) (Y. Shen & Jin, 2014) is a measure of model performance, specifically the difference between the values predicted by the predictive model and the observed values. At the same time, rMSE can also help evaluate the detection error caused by privacy protection. Explained from a calculation point of view, the rMSE calculation equation is as follows:

$$rMSE = \sqrt{\frac{\sum_{i=1}^{n} exp(\hat{p} - o)^2}{n}} \tag{2.19}$$

In Eq.(2.19), rMSE represents the square root of the quadratic sample moment of the difference between the predicted value $\hat{p}$ and the observed value o. When the root mean square error calculation is performed on a sample of data, the deviation that occurs is

called the residual. If the rMSE is calculated using data outside the sample, it is called the prediction error. In the performance evaluation of privacy-preserving algorithms, rMSE is an important measure matrix. In the calculation for the privacy-preserving level, this evaluation method can combine the data fragment in privacy-protected data sets (such data is mixed with privacy noise) and non-privacy data sets. The magnitude of the prediction error is simultaneously mapped to the same metric space for unified evaluation, and the prediction error of different models can be described under a specific data set, rather than the prediction error between data sets. Mapping to the same space can be a good measure of the impact of existing privacy-preserving algorithms on model prediction accuracy. In addition, the results of the mapping can also show the performance difference of different privacy preservation algorithms on the same data set. In general, the lower the value of rMSE, the higher the accuracy of its model prediction.

2.6.5 Experimental Design with Data Quality and Noise-Related Assessment Methods

When testing privacy-preserving methods for tasks related to machine vision image processing, it is essential to employ tools for evaluating image quality, comparing distributions, and analyzing image frequency characteristics to verify the effectiveness of the privacy-preserving methods during the experimental stage. Here, we present four measurement methods: peak signal-to-noise ratio (PSNR) (Bovik, 2010), the structural similarity index (SSIM) (Wang, Bovik, Sheikh & Simoncelli, 2004), maximum mean discrepancy (MMD) (Gretton, Borgwardt, Rasch, Schölkopf & Smola, 2012), and fast Fourier transform (FFT) (Oppenheim, 1999). These methods are instrumental in assessing the impact of privacy noise added by privacy protection methods on image quality.

The peak signal-to-noise ratio (PSNR) (Bovik, 2010) is a widely used metric for

measuring the quality of reconstructed images combined with the original image and noise. It compares the original and reconstructed signals by calculating the ratio of the peak signal power to the mean squared error (MSE) between the original image and the image with added privacy noise. A higher PSNR value indicates that the reconstructed image features are closer to the original signal in terms of quality. PSNR is often expressed in decibels (dB). It's important to note that PSNR is a pixel-based metric and does not necessarily correlate with human perception of image quality.

The structural similarity index (SSIM) (Wang et al., 2004) is a perceptual metric that evaluates the structural similarity between two images. It considers luminance, contrast, and structure to assess the similarity between the original and distorted (privacy noise-added) images. While SSIM is mainly used for image quality assessment, it can indirectly help gauge the impact of added privacy noise. A higher SSIM value indicates greater similarity between the images, suggesting that the added noise may have less impact on the structural characteristics and perceptual quality of the image.

The maximum mean discrepancy (MMD) (Gretton et al., 2012) is a statistical measure used to compare probability distributions. It quantifies the dissimilarity between two sets of samples, such as the original dataset and the dataset with privacy noise. In the context of added privacy noise, MMD can be employed to compare the distribution of the original data with the distribution of the data after the noise is added. A higher MMD value would indicate a larger discrepancy between the distributions, implying that the added noise has a noticeable effect on the data. Additionally, MMD is often used in machine learning tasks, such as domain adaptation for privacy-preserving regions, where the goal is to align the distributions of different datasets. By minimizing the MMD, one can ensure that the distributions of the source and target domains become more similar.

Fast Fourier transform (FFT) is an algorithm used to efficiently compute the discrete

Fourier transform (DFT) of a signal or image (Oppenheim, 1999). It converts a time-domain or spatial-domain signal into its frequency-domain representation. While FFT itself does not directly measure noise or image quality, it is often used in signal and image processing techniques for tasks such as noise removal, filtering, and analysis. By applying FFT to noisy images, one can analyze the frequency components and explore the presence or characteristics of noise in the frequency domain. Analyzing the noise-added image and the original image from the perspective of FFT can also reveal the performance of privacy-preserving methods compared to random noise methods. Usually, some random noise is present in the original image, which can be easily removed by privacy attackers through FFT, thereby compromising the protection of private information.

In many cases, a good privacy protection method should also preserve the characteristics of the original data in the added data. While some evaluation methods can easily remove simple noise with inconsistent distribution, such as random noise, it is difficult for these methods to distinguish the noise added by privacy-preserving methods. Privacy-preserving methods intentionally blur the difference between the original data and the added data, making it challenging for methods such as FFT to identify them accurately.

2.6.6 The Experimental Design of Differential Privacy Noise Analyze on K-L Divergence

The privacy-preserving methods based on differential privacy proposed in this thesis will add privacy noise to the data, and these methods handle the generation and addition of privacy noise. The statistical properties of both types of data can be characterized by measuring the difference between the original and noisy data using the Kullback-Leibler (K-L) (Goldberger, Gordon & Greenspan, 2003) divergence (also known as relative

entropy). Specifically, if the K-L divergence is very small between the two distributions, indicating that the noised data is very similar to the original data, this may mean that the differential privacy method can maintain the overall statistical properties of the data when adding noise and provide more meaningful features for the machine learning model's predictions. However, if The distributions are the same or very similar, it may raise concerns about insufficient or no privacy protection.

On the other hand, if the K-L divergence is large, meaning that there is a substantial difference between the noisy data and the original data, it may imply that the differential privacy method introduces excessive or inappropriate noise, resulting in a decline in data quality or the loss of important statistical characteristics. This can potentially impact the accuracy and reliability of subsequent machine learning or data analysis tasks.

Therefore, by using the K-L divergence to compare the difference between the original data and the noise-added data, it is possible to evaluate the degree of influence of the differential privacy method on the data, as well as the magnitude and suitability of the noise. This helps to strike a balance between privacy protection and data utility by selecting an appropriate differential privacy parameter or noise level.

In theory, K-L divergence can be used to analyze the statistical characteristics between the original data and the sampled privacy noise data to calculate the distribution difference between them. From an experimental point of view, the same length defines and quantifies some situations in advance, so as to facilitate the experiment. Specifically, the mean value of several Gaussian distribution samples can be selected, and this mean value can be selected between $[-\sigma, \sigma]$. Suppose the loss function is $Loss(\varepsilon, \sigma) = \|\varepsilon - \sigma\|$. Here, $\exp(-a\|\varepsilon - \sigma\|)$ will be proportional to the sample, thus measuring the difference in differential privacy. Generally, the Gaussian mechanism is used to sample and generate data, and the original data is used as the ground truth to compare the difference between the sample and the processed data. In the Gaussian mechanism, the global

sensitivity is generally defined as 4 to standardize the comparison of the data range.

2.6.7 Methods for Measuring Spanning Trees and Adjacency Matrices

In graph theory, a spanning tree T of an undirected graph G can be recognized as a sub-graph, which is a tree containing all nodes of G(Kumar, Singh & Chakrabarti, 2005). In the privacy-preserving machine learning model, the data that needs to be inferred is input into the privacy-preserving machine learning model, and the inference results with privacy noise added will be output. For example, if there are street signs in a street map if the street signs are regarded as a sensitive class, the ideal prediction result of the privacy-preserving machine learning model is to mark all the street signs (assumed to be n street signs) and for the street signs corresponding privacy noise is injected into the feature region R to prevent privacy attackers from identifying street names.

So how do measure whether the privacy noise inside the street sign reaches the corresponding privacy level? Here, the privacy level of the data can be measured by calculating the path cost of the spanning tree. In particular, we create a spanning tree and input the corresponding identified feature regions $R_1, \ldots, R_n$ into this spanning tree, and put these feature regions into the tree node, then measure the overhead from the root to the terminal node called path. Generally, the larger the length of the path, the greater the overhead of gathering the same kind to the root node, which also means more time and space resources required for privacy protection.

But in general, the overhead from the root node to the node also depends on the number of features in the model. If the training data in the model is particularly single, the generalization performance of the model itself will be poor, so the fewer paths, the better the generalization ability of the model (Cheng, Fu & Liu, 2010). In many cases, however, the fewer the number of paths, the less entropy the model can reveal. In

addition, a shorter path length means that the probability of a dataset being successfully attacked by background knowledge is lower. Sometimes, because there are too many privacy protection features and other features, the privacy attacker can receive too much interference information and reduce the possibility of matching, so the depth of the tree itself will reduce the hit rate of the privacy attack.

In addition, privacy attackers can evaluate their privacy level through background knowledge attacks. Generally speaking, background knowledge attacks match existing external data with existing data. If some features can be matched, it is marked as a frequency. The higher the frequency, the greater the possibility of privacy leakage. Through these two indicators, the comparison between the privacy level of a privacy protection model and its own cost can generally be evaluated.

2.6.8 Experimental Methods for Chi-Squared Test in Privacy Protection

Chi-Squared test is a statistical distribution that approximately obeys the chi-square(χ^2) distribution when the null hypothesis is established. In the chi-square test experiment, the general experiment needs to divide the detected object into several mutually exclusive categories, and these categories use the same theory (or null hypothesis) to detect the model of the probability distribution that the observed value is classified into different categories. In subsequent experiments, the data to be detected can be observed, and the data to be detected (observed) can be privacy noise data. The greater the deviation between the actual observed value of the statistical sample and the theoretically inferred value, the greater the chi-square value. Conversely, the smaller the deviation between the two; if the two values are completely equal, the chi-square value is 0, indicating that the theoretical value is completely in line with the actual observed value. In the Markov process, at different time points, the theoretical and observed data The deviation of

the value may be inconsistent, so observing the deviation at different time points can describe the stability and convergence degree of the algorithm in the time series.

This chapter addresses the work related to privacy protection and machine learning. At the same time, the experiments and methods involved in this thesis are also explained. It is convenient for readers to understand the idea of experimental design in this thesis.

Chapter 3

Privacy Preservation Methods for Pre-Processing Stage

In Chapter 1, this thesis raises a fundamental question: How to protect data privacy in machine learning models by generating the noise needed for privacy and being able to track or control the addition of privacy noise? The problem focuses on safeguarding the privacy of training and prediction data in machine learning models by incorporating privacy noise. A crucial aspect is generating suitable privacy noise within a given privacy budget and subsequently applying it using that budget. Moreover, it is vital to ensure that the process of adding privacy noise to training and prediction data can be traced and controlled to achieve comprehensive privacy protection. To address this question, this chapter progressively expands on these three sub-questions and proposes solutions to address them.

This chapter specifically concentrates on the pre-processing layer of Figure 1.2, which deals with the pre-processing of input training data to preserve privacy. Three methods are introduced in this chapter, each designed to tackle different types of problems. The structure of this chapter and the function of each section are explained below (Chapter 3.0.1), followed by the threat model (Chapter 3.0.2) and the chapter

roadmap figure.

3.0.1 Structure of Chapter

The method in this chapter starts from the limitations of differential privacy and gradually explains the motivation of the privacy protection method proposed in this thesis, as described in Chapter 3.1. And combine existing problems to improve two classic mathematical theories to support several privacy protection methods proposed in subsequent sections, as described in Chapter 3.2.

In Chapter 3.3, the thesis describes the utilization of Kullbak-Leibler divergence to measure the information gap between subsets of the input data. This helps to measure the privacy noise for sub-sampling and provides the necessary parameter for the measurement of information loss.

The key application of the rest of this chapter is to prevent privacy mining by detecting features in data, and privacy-preserving machine learning based on image privacy will be the focus of the application. Figure 3.1 illustrates a method to mitigate privacy mining attacks. This approach can be employed as both a federated learning method and a local processing method. In the first and last steps, all data involving sensitive information must be processed locally, while the parameters associated with the discriminative matrix (DM) in the fourth step are either uploaded to the server (for federated learning) or processed locally (for local processing) to facilitate unified training scheduling or optimization. Sensitive classes identified in the training dataset will be addressed using a modified dynamic conditional random field approach, as explained in Chapter 3.7. This model facilitates the computation of the Discriminative Matrix (DM), as introduced in Chapter 3.5, enabling the identification of similar sensitive regions in an image. Consequently, this information allows for weight adjustments to be made during future training sessions. The method for evaluating the privacy level is described

in Chapter 3.4. All these methods are relevant to the subsequent stage of processing training data depicted in Figure 1.2, which will be discussed in Chapter 4.

In Chapter 3.8, the thesis discuss the results of these studies. In this section, the experimental studies are conducted by using videos from three publicly available datasets – MOT (Lealtaixé, Milan, Reid, Roth & Schindler, 2015), Cityscape (Cordts et al., 2015) and KITTI (Geiger, Lenz, Stiller & Urtasun, 2013). These images(videos) are street scenes that contain information by which their locations could be identified. Besides, the objects such as street names, signboards, and landmark buildings are treated as sensitive classes in those datasets.

3.0.2 Threat Model for Privacy Leakage in Machine Learning

In the threat model, there are several points to consider (R. Xu, Baracaldo & Joshi, 2021),

- Whether the data contains privacy information?
- Whether the model contains the characteristics of private information?
- Whether the model can be inverted?
- Whether differential privacy is in effect?

Among these points, it is important to note that if the data does not contain any sensitive information, there is no privacy leakage concern. Therefore, all the discussed issues here are based on the assumption that the data includes private classes.

From the second point of discussion, in a machine learning model, if the input training data contains sensitive information, the features within the network structure will also include sensitive features. Hence, privacy protection algorithms need to consider the features contained within the model itself.

The third point regarding model inversion attacks and the fourth point regarding the effectiveness of differential privacy are related, so they are discussed together. Model inversion attacks involve privacy attackers inferring sensitive information through the output prediction information, typically using background knowledge attacks to obtain privacy information. Considering two extreme cases, the first case is when the training data inputted into the model is completely inconsistent with all the background knowledge possessed by the privacy attacker. In this scenario, the privacy attacker cannot match any background knowledge information. The other case is the most challenging, where the background knowledge of the privacy attacker perfectly matches the training data inputted into the model. In such a case, without privacy protection, the privacy attacker has a chance to reconstruct the private information. In other words, the extent to which the background knowledge of the privacy attacker covers the training data significantly influences the success probability of the attacker inferring private information.

The effectiveness of the fourth point of differential privacy is that even if the background knowledge completely matches the model input data and there is a chance to reconstruct private information, differential privacy can also use different degrees of protection mechanisms to reduce the reconstruction of privacy information risk.

In subsequent experiments, the most extreme case will be used for verification, that is, it is assumed that the privacy attacker uses the original information of the training or prediction data. In this case, the original data D can be used as the ground truth data, and the noise prediction or training data can be used as D'. The test can be performed by statistical analysis or similar between D and D' to verify the effectiveness of the privacy protection method.

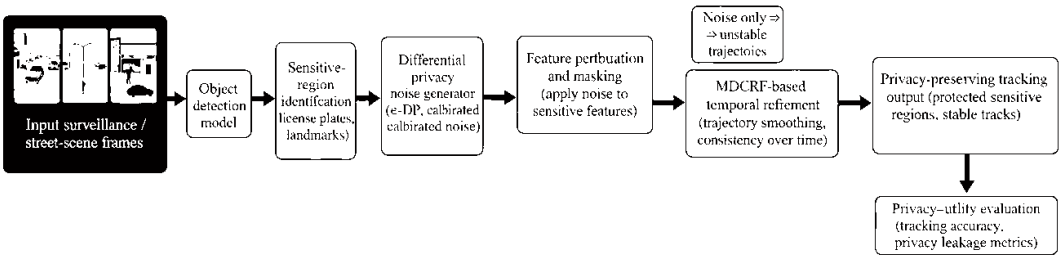


Figure 3.1: The process of privacy-preserving machine learning model

3.1 Motivation with Limitations of Differential Privacy

Existing mechanisms for achieving differential privacy utilize the randomization of original data sampling, for example, the differential privacy mechanism introduces noise generated by Laplacian or exponential distribution to protect privacy (Barthe, Köpf, Olmedo & Zanella Beguelin, 2012). However, the predictive accuracy of machine learning algorithms will suffer due to the introduction of noise. For example, in a picture, there is a human face. If the facial features are used for privacy, adding noise to the facial features in order to prevent extraction of facial noise will reduce the accuracy of machine vision model recognition. Thus, there is a trade-off between privacy protection and prediction performance (Koufogiannis, Han & Pappas, 2015). Furthermore, randomizing high-dimensional data is computationally demanding and incurs substantial practical costs (Mironov, 2017). If only a small part of the data records are analyzed, then they cannot be protected from background knowledge attacks. In

addition, protecting privacy by adding random noise is inefficient because most random noise is easy to distinguish since such noise exhibits standard statistical distributions.

The improvement of the Gaussian mechanism has been made by taking into account the statistical distribution of the data to generate differential privacy protection (Liu, 2018), however, this process is still flawed. In (Dwork, Naor, Reingold, Rothblum & Vadhan, 2009), the removing or replacing data of an individual in a dataset has a negligible effect on the distribution. Therefore, it cannot reflect whether the privacy data of individuals are well protected according to their wishes during the data aggregation process.

Gaussian mechanism (Liu, 2018) and Laplace mechanism (Dwork, McSherry et al., 2006) are two commonly used privacy protection mechanisms in differential privacy. They both aim to protect sensitive data by adding noise. Here are the similarities and differences between them:

- Objective: Both the Gaussian mechanism and the Laplace mechanism aim to provide differential privacy to protect individuals' sensitive information.
- Noise distribution: Gaussian Mechanism uses the Gaussian distribution (also known as the normal distribution) to generate noise, while Laplace Mechanism uses the Laplace distribution to generate noise.
- Noise characteristics: Gaussian noise is continuous, symmetric, and can take values across the entire real number range, allowing for arbitrary noise magnitude. Laplace noise is discrete, asymmetric, and has peaks on both sides of the real number range, limiting the magnitude of the added noise.
- Privacy protection strength: In terms of privacy protection strength, the Gaussian mechanism usually provides stronger privacy protection. Due to the larger variance range of the Gaussian distribution, it can generate larger noise values,

effectively hiding sensitive information. The noise range of the Laplace Mechanism is limited and may not provide the same level of privacy protection as the Gaussian Mechanism.

- Data handling: Gaussian mechanism typically requires normalization of the data's value range before adding noise to match the scale of the noise with the data. Laplace Mechanism usually does not require additional data handling since the generated noise can adapt to different data ranges.

- Mathematical Properties: The additive property of Gaussian noise makes it convenient to protect privacy for multiple samples by directly adding the noise values. The additive property of Laplace noise is less straightforward and requires special handling, such as using the exponential mechanism.

In order to reflect the level of privacy data, a theorem regarding (ϵ, δ)-differential privacy was introduced in privacy-preserving area (Dwork, McSherry et al., 2006). It states that for any $(\epsilon, \delta) \in (0, 1)$, the Gaussian output perturbation mechanism $\sigma \leq \frac{\Delta p_2 log \frac{1.25}{\delta}}{\upsilon}$ is (ϵ, δ)-differential privacy. The δ represents the privacy cost that quantifies the probability of a privacy breach, Δ is differential for $\frac{\Delta p_2 log \frac{1.25}{\delta}}{\upsilon}$ and $p_2\sqrt{1.25}\delta$ is differential parts for calculating privacy cost and υ is the maximum distance between a search on datasets a and the same search on the other datasets b, a and b can be train sets or real incoming inference data, δ is the probability of information accidentally being leaked and it also means the privacy cost. From the perspective of the relationship between privacy leakage and privacy cost, it has the less the probability of information leakage, the greater the consumption of privacy costs.

In terms of the value of δ in (ϵ, δ) differential privacy, two problems arise. First, in differential privacy, it needs to be determined whether the value of δ can effectively support the privacy protection algorithm to generate effective differential privacy noise with Gaussian perturbation while minimizing the utility loss caused by the privacy noise

in the newly proposed scheme. The second question is what happens when $\varepsilon \geq 1$. These questions will be addressed in this chapter. More specifically, it will display:

1. The value of δ given in this theorem is sub-optimal at high privacy settings, i.e. when $\epsilon \to 0$;

2. The problem of proving the feasibility of the strategy by analyzing the Gaussian mechanism needs to be solved.

3. If the value of the privacy budget ϵ is large, the Gaussian deviation (ϵ, δ) and the standard deviation $\sqrt{\epsilon}$ must be controlled within the valid range during the measurement.

The classical Gaussian perturbation Θ in the range of $\epsilon \in (0, 1)$ cannot exceed any bounded interval. Based on this concept, it is proposed here to classify the input data according to its entropy. Then, noise is generated in accordance with its probability distribution. The purpose of privacy protection can be achieved by injecting privacy noise into the joint stochastic gradient descent process, which will also have an impact on the next machine learning training process.

Gaussian privacy systems have bounds, especially if $\epsilon \to 0$. Using Gaussian perturbation to achieve $(0, \delta)$-differential privacy overcomes the capabilities of the classical Gaussian mechanism. Because the standard deviation of the differential singularity theorem $\sigma = \Theta(\frac{1}{\sqrt{\varepsilon}})$ grows to infinity as $\varepsilon \to 0$, $(0, \delta)$ $\sigma = \frac{\Delta}{2\delta}$ and the differential singularity of Δ differs by 2δ.

The case of $\varepsilon \to 0$ has aroused great interest in both academia and industry. For example, in (Balle & Wang, 2018), the author studies (ϵ, δ)-differential privacy in the case of $\epsilon \to 0$. The limitations of current Gaussian distribution mechanisms need to be overcome if differential privacy is to be incorporated into existing deep learning methods (Ma, Yan et al., 2021). This issue has been raised in (Dwork, Kenthapadi, McSherry, Mironov & Naor, 2006).

In the next section, the basic theory of privacy noise generation will be explored, aiming to address the limitations of the Gaussian distribution mechanism and its impact on the generation of privacy noise. The goal is to avoid the constraints of the privacy mechanism and propose a solution for the generation mechanism of privacy noise.

3.2 Background Theories

3.2.1 Kernel Hilbert Space

In Chapter 2.4.2 it is mentioned that Hall helps Gaussian process to measure privacy noise by introducing RKHS (Hall et al., 2013). Based on this idea, this thesis will also try to measure and calibrate the generated noise through RKHS to assist in the sampling and generation of privacy noise.

Here, the Reproduced Kernel Hilbert Space (RKHS) will be used for the measurements. According to Hall's research, RKHS enables the noise generated by privacy-preserving algorithms to be mapped into this space and be precisely defined and measured (Hall et al., 2013). It can define $\mathcal{H}$ as the Hilbert space of the real-valued function f, and the region of the Euclidean space as $\mathbb{R}$.

In addition, in Kadri's work (Kadri, Duflos, Preux, Canu & Davy, 2010), their research provides the contribution that, Hilbert space $\mathcal{H}$ in an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ which is linear, commutative, and obey the triangle inequality.

This thesis uses the commutative equation from Kadri's work, the thesis defines a function $K : X \times X \to \mathbb{R}$ as a kernel of $\mathcal{H}$ if there exists a map $\phi : X \to \mathcal{H}$ such that

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \qquad \forall x, y \in X \tag{3.1}$$

Furthermore, this is a reproducing kernel if

$$K(x,y) = \langle K(x,\cdot), K(y,\cdot)\rangle_{\mathcal{H}} \tag{3.2}$$

H is also known as the reproducing kernel Hilbert space (RKHS), and this kernel product is used to reconstruct the inner product of two functions $K(x,\cdot), K(y,\cdot)$.

The Eq.(3.1) and Eq.(3.2) show the relationship between the original data $K(y,\cdot)$ and $K(x,\cdot)$ with added noise to preserve privacy, the $\phi(x)$ and $\phi(y)$ are the position for pair data D in the space and so on. It also revealed that noisy data can be generated by reconstructing the original data in RKHS.

$$K(x,y) = \langle \phi(x), \phi(y)\rangle_{\mathcal{H}} = \langle K(x,\cdot), K(y,\cdot)\rangle_{\mathcal{H}}\,, \tag{3.3}$$

In Eq.(3.3), $\phi(x), \phi(y)$ are reconstruct the inner product from RKHS. Without knowing the contents of the mapping and attribute space, the calculation process only needs to know that K is asymptotically positive definite, and the feature space of mapping ϕ and $\mathcal{H}$ must be established.

If $\mathcal{H}$ aims to achieve differential privacy, privacy-preserving noise can be added before training. The privacy level can also be detected using the data feature when detecting spaces range in $\mathcal{H}$. However, we should also consider abstract Wiener space conditions, which can improve the privacy of output support vector machine (SVM) algorithms. In our work, we only focus on this case, where we can generate noise using Hilbert space and use joint stochastic gradient descent (SGD) to model the input data for the next step.

3.2.2 Abstract Wiener Space

Abstract Wiener spaces are used in some machine learning areas such as support vector machines (SVMs) and Gaussian processes. The abstract Wiener spaces are applied in modeling problems involving continuous time and randomness. In the paper by Dudley (Dudley et al., 1971), it is mentioned that strictly positive Gaussian measures in separable Banach spaces can be obtained only through the decomposition of the abstract Wiener space structure. Trigger from Dudley's work, the solution in this section adds Gaussian noise to the abstract Wiener space and shows how to tune the noise to achieve differential uniqueness in the RKHS or abstract Wiener space, then the space separability determines the tuned of noise parameters. At the same time, the differential uniqueness of the space is also the key to measuring the entropy of the input data required for training. In addition, RKHS or abstract Wiener space also plays an important role in understanding the impact of different levels of noise on privacy.

Assume that data D has differential privacy (DP), let Banach space $\mathbb{B}$ to achieve $\theta : D \to \mathbb{B}$, which can make Gaussian with realized differential privacy Noise is quantized. Introduce $\chi \in \mathbb{G}$ as data subject to Gaussian distribution, and use the Gaussian function generated by $f(\chi)$ in Euclidean space $\mathbb{R}$ as source data. We can get the definition $\theta_D := \theta(\mathcal{D})$.

Define $\chi \in \mathbb{G}$ as a Gaussian function, and define a random function in space $\mathbb{R}$, the distance between its values $f(\chi)$ is Euclidean distance. Suppose the dataset D is a dataset compliant with differentially private (DP) within Banach space $\mathbb{B}$. $\theta_D : D \to \mathbb{B}$ and θ_D represents Gaussian noise implemented by differential privacy. It has $\theta_D := \theta(\mathcal{D})$. Equipped with the norm

$$\|f\|_{\mathbb{B}^*} = sup_{\|h\|_{\mathbb{B}^*} \leq 1} f(h), \tag{3.4}$$

In Eq.(3.4), the dual space B^* of $\mathbb{B}$ is also a separable Banach space and $\|f\|_{\mathbb{B}^*}$ as random function running in $\mathbb{B}$. The pair $(\mathbb{B}, w)$ is called an abstract Weiner space, w

is the probability measure over $\mathbb{B}$ induced by χ. Every Gaussian process is uniquely parameterized by the mean, $\theta \in \mathbb{B}$, and covariance $\Theta : \mathbb{B}^* \to \mathbb{B}$, which for every $f \in \mathbb{B}$ satisfies

$$E[\chi] = f(\theta). \tag{3.5}$$

In Eq.(3.5), the $E[f(\chi)]$ means the mean for Gaussian function χ and the Gaussian noise as $f(\theta)$. From the research in this section, it can be seen that abstract Wiener space scales the Gaussian noise level from a set of data, such as training datasets. This enables the noise generated by privacy-preserving algorithms to be condensed in Wiener space and precisely defined and measured.

Error analysis: from infinite-dimensional Wiener space to discrete implementation. The foregoing development assumes an abstract Wiener space $(\mathbb{B}, w)$ with separable Banach space $\mathbb{B}$ and Gaussian measure w; in practice, implementations use finite-dimensional projections (e.g., a finite set of linear functionals $f \in \mathbb{B}^*$) and floating-point arithmetic. In practice, the approximation error arises from two sources: finite-dimensional truncation and floating-point rounding. Formally, the total approximation error can be thought of as the sum of a truncation term (due to restricting to $\mathbb{B}_n$) and a rounding term (due to finite precision); under fixed dimension and precision, both are controlled in the experiments. For a finite projection $\mathbb{B}_n$, the discrepancy from the ideal mechanism decreases as the retained covariance spectrum becomes better resolved, while the numerical error introduced by floating-point arithmetic remains bounded under a fixed implementation precision and sampling routine. Although a full theorem-level bound is left for future work, these sources of approximation are controlled in this thesis by using fixed projection settings, identical random seeds across comparative experiments, and a consistent numerical implementation pipeline. Therefore, the empirical comparisons reported in this thesis are not affected by inconsistent

numerical settings across methods, even though a complete analytical bound remains an open problem.

3.3 Measuring Information Loss with Kullback-Leibler Divergence

How to detect privacy noise and control its addition determines whether privacy noise can effectively protect privacy. The first question in Chapter 1 also highlights the significance of controlling noise addition for privacy. This section will introduce several effective methods to address this issue and achieve control and addition of privacy noise. One such method is the Kullback-Leibler (K-L) divergence method discussed in Chapter 2.4.1.

For Kullback-Leibler (K-L) divergence method, Chapter 2.4.1 discusses the method of using Kullback-Leibler (K-L) divergence to measure the difference of privacy noise distribution, which provides an important method for solving the privacy budget measurement. K-L divergence is also important for the sampling and generation of privacy noise.

On the other hand, ϵ-differential privacy is mainly used in privacy protection, also known as partial differential privacy. This method can effectively improve privacy protection in the process of machine learning. In some mainstream methods of machine learning, such as principal component analysis, support vector machine, risk reduction and continuous calculation, linear and logistic regression, etc. have been widely used (Ye & Hu, 2020). Recent results show that it is possible to combine stochastic gradients on two constraint sets to derive noise variants to minimize the Lipschitz convex function (Ma, Yan et al., 2021).

The first method is to randomize the "dropout" step to prevent over-fitting, strengthening the privacy guarantee in a simple 1-layer neural network. Furthermore, the problem of deep collaborative learning with multiple participants has been solved using distributed stochastic gradient descent (Shokri & Shmatikov, 2015). However, existing methods cannot cope with complex network environments where training is conducted using Cloud services. The challenge of this problem is how to protect the privacy of the training data both during and after training, and in such an environment, the validity of the information is guaranteed while privacy is guaranteed. Solving this problem requires a method to measure the quality and loss of information in the training stage. The measure of this is the Kullback-Leibler divergence (Kullback, 1997) discussed in Chapter 2.4.1.

Definition 3.1 (Kullback–Leibler divergence)**.** *Let P and Q be two distributions on the same sample x from space X. Kullback–Leibler divergence of Q from P is given by*

$$KL(P||Q) = -\Sigma_{x\epsilon X} P(x) \cdot \log \frac{P(x)}{Q(x)} \tag{3.6}$$

According to the description of Definition 3.1, the Kullback-Leibler divergence can be measured by using a certain loss function to evaluate the difference between the statistical distribution of privacy noise inserted in machine learning training and the distribution of the original training data. The distribution characteristics of the privacy noise and get the relationship with the privacy budget.

Before discussing data quality and building a data loss measurement model, it is first necessary to clarify the concept of data accuracy. Data accuracy is closely related to data loss caused by data processing during privacy protection. That is, less information loss means better data quality. The appropriate measure depends on the specific category of privacy protection data mining algorithms. For example, an algorithm may use perturbation or blocking techniques to hide the original and aggregated data. In this

case, the K-L divergence can be used to obtain the difference between the original dataset D and the privacy-preserving modified dataset D' to measure the information loss.

In the process of machine learning training, the pooling method of data increases the uncertainty of privacy data, which brings difficulties in the measurement of data set distribution. Based on this situation, Oliveria et al. (Oliveira & Zaiane, 2002) proposed a difference measure method based on the frequency histogram of the dataset. This method is also developed on the basis of the K-L divergence method, and its method is specifically defined as the ratio of the sum of the absolute errors of the sum of the element frequencies of the modified data set to the sum of the element frequencies of the original data set. A difference measure for the frequency histogram of a dataset expressed mathematically

$$L(A, A') = \frac{\sum_{i=1}^{n} |f_A(d_i) - f'_A(d_i)|}{\sum_{i=1}^{n} f_A(d_i)} \tag{3.7}$$

In Eq.(3.7) the data element defines as d in the resource datasets A and in this dataset the frequency is $f_A(d)$. A' is the modified dataset with privacy-preserving processing and $f_{A'}(d)$ is the degree of frequency of data cell d in cell A'.

If the multiplicative noise method is used to quantify the results of K-L divergence measurements, and the changes in these data measurements are mapped into a matrix, the calculated data distribution error depends on the disturbance generated by the mapping process. When the dimensionality of the original data is nearly equal to the dimensionality of the mapped adjacency matrix, the inner product of the mapping matrix and the matrix of the original data will also form a larger value. Here the (Liu, Kargupta & Ryan, 2006) method proposes to use data mining techniques to obtain the Hamming distance of two matrices, that is, the angle of the projection vectors of two sets of matrices. This thesis draws on the Hamming distance calculation method and

calculates the correlation coefficient by the error generated by the mining results. This method is called mixed privacy protection noise.

In the quantification of noise data, after the K-L divergence is measured, in order to reduce the measurement error m, the scheme also uses generalization or aggregation to form a spanning tree for the feature relationship, and the noise is measured through the calculation of the spanning tree. In the calculation of measuring the noise deviation, the scheme reflects the deviation of the changing data caused by each leaf node by pruning the spanning tree, so as to reflect the average accuracy of the data set trained by the privacy noise in the machine learning model. For a dataset D with N_A labels with N features inside, the information loss can be computed as

$$InformationLoss(D^*) = \frac{\sum_{i=l}^{i=N_A} \sum_{j=l}^{i=N} \frac{m}{|GD_{A_i}|}}{|D| * |N_A|} \tag{3.8}$$

In Eq.(3.8) the information loss for the spanning tree can be calculated, where D^* is the modified dataset and $\frac{m}{|GD_{A_i}|}$ represents the loss of each modified data sets GD_{A_i}.

There are other measures specifically designed for Gaussian differential privacy methods with an information loss matrix. For example, by generalizing the data and calculating the improved performance after generalization to determine the data quality of a set of data sets (Samarati, 2001). Usually before generalization, the meaning expressed by the data is more clear, and after one or several generalizations, the meaning expressed is more general and vague. This results in the loss of information, which is called the loss of entropy in information theory. Therefore, the generalization of data should reduce this information loss as little as possible to ensure the maximum utility of the data, but this k-anonymous method cannot guarantee each step of generalization, the loss of information entropy can be quantified and the loss is equal to the amount. This is also a problem that needs to be solved.

Iyengar (Iyengar, 2002) proposed a more general classification metric (CM). His

work proposes the classification metric between different classes, and the loss of classify task can be based on CM. The loss measure method can help to measure the loss of classification. And the loss measure can start with this general classification metric(CM). Trigger from his idea, we propose the Loss Measure(LM) metric. In the Loss Metric (LM), if the dataset D has n features that can be associated with similar datasets, then LM can be quantified to the sum of all the data in the dataset for each generalization average entropy loss. It can be expressed mathematically as follows:

$$LM(D^*) = \frac{\sum_{i=l}^{n} \sum_{j=l}^{|D|} \frac{f(D*[i][j])-1}{G(A_i)-1}}{|D| * |N|} \tag{3.9}$$

The Eq.(3.9) can calculate the loss measurement, where D^* is the privacy-preserving (anonymization) result of dataset D and modified data sets as $G(A_i)$. f is a function of the given data unit of D, and G returns the number of different attributes given an attribute A_i.

This section introduces the K-L divergence for measuring noisy data samples as a measure of distributional differences. The method can help existing privacy-preserving algorithms measure the difference between the private data added to training and the original data sampled. However, the measurement distribution can only measure the data differences, and these differences cannot be dynamically used to track the dependencies between privacy features. Moreover, after adding privacy noise in the privacy-preserving method, the correlation between the privacy feature and the target in the source data will be weakened, and it will become difficult to track the dependency relationship. To be able to track dependencies between sensitive objects, the next section tries to use a conditional random field approach based on observations of global variables.

3.4 Method of Generate the Privacy Noise with the Privacy Budget

In Chapter 1, the first question was raised: How to protect data privacy in machine learning models by generating the noise needed for privacy and being able to track or control the addition of privacy noise? The core of the problem is how to generate appropriate privacy noise based on the given privacy budget through noise sampling. This section will explore the properties of the noise itself, and then expand to how generating privacy noise can help privacy-preserving methods defend against privacy attacks. The sampling and generation of noise can start with the Gaussian noise of the data set.

The noise required for privacy protection can be generated by a randomized process $M : \mathbb{G} \to \mathbb{R}$ where $\mathbb{G}$ is a Gaussian distribution and $\mathbb{R}$ is the range that satisfies (ϵ, δ)-differential privacy. It will generate the noise from privacy budget ϵ. In Chapter 3.6, the output perturbation of ϵ-differential privacy will be given by $\delta \leq \frac{\Delta p_2 log\frac{1.25}{\delta}}{\varepsilon}$. A noise model n_t can be derived such that the sampled subset $D_t \leq K(x, y)$ where $\frac{\Delta p_2 log\frac{1.25}{\delta}}{\varepsilon}$ is the upper range of perturbation and $K(x, y)$ as kernel calculation with (x, y). This noise model n_t could be updated according to $\Delta n^k = n^k - n_t$ of which the n^k is the noise in RKHS, and $(\Phi(a), \Phi(b))$ are the privacy noise from sample process from training and validation data sets. For each input dataset, the privacy noise samples are obtained by

$$n_{t+1} = n_t + \frac{1}{m}\left(\Sigma_{k=0}^{m_t} \frac{\Delta n^k}{\max(< \Phi(a), \Phi(b) >, \frac{\|\Delta n^k\|_2}{P})} + \mathcal{N}(0, \epsilon^2 P^2)\right). \tag{3.10}$$

In Eq.(3.10), the term $\Sigma_{k=0}^{m_t} \frac{\Delta n^k}{\max(<\Phi(a),\Phi(b)>, \frac{\|\Delta n^k\|_2}{P})}$ to the right of the equal sign in the formula is sampling and calculating the noise from the privacy budget ϵ, which will sum the updated noise at k states with x dimension as m and m_t and the term $\mathcal{N}(0, \varepsilon^2 P^2)$ is

the noise scaled to P. m represents the number of iterations or samples used in the noise generation process. This value determines the precision and accuracy of the generated noise, m_t denotes the current iteration number. The $\max(< \Phi(a), \Phi(b) >, \frac{\|\Delta n^k\|_2}{P})$term involves the maximum between the dot product of two vectors $\Phi(a)$ and $\Phi(b)$, and the Euclidean norm of Δn^k divided by the privacy parameter P. The dot product and Euclidean norm are used to measure the similarity between the neighboring datasets, $\Phi(a)$ and $\Phi(b)$ are feature vectors representing the database entries a and b, they encode the sensitive information present in the databases and are used to compute the similarity between them. n_{t+1} and n_t represent the updated value of the privacy noise at time $t + 1$ and t separately. This value is obtained by adding the noise generated in the current iteration to the previous value of the noise. This step will generate the privacy noise from the scaled P. $\mathcal{N}(0, \epsilon^2 P^2)$ represents the Gaussian noise added to the privacy noise generation process. It follows a Gaussian distribution with mean 0 and variance $\epsilon^2 P^2$. The Gaussian noise is scaled by the privacy parameter P and the privacy budget parameter ϵ to control the level of privacy achieved.

In consecutive images, assume the generated Gaussian noise has the length as n and a standard deviation of σ. After applying the discrete Fourier transform (Oppenheim, 1999), the spectrum of the noise becomes Gaussian noise with a standard deviation of $\sqrt{N}\sigma$. The $\sigma\sqrt{N}$ represents the standard deviation of the normal distribution, which measures the spread or dispersion of the distribution. A larger standard deviation indicates a more dispersed distribution, while a smaller standard deviation indicates a more concentrated distribution. The sampled noise can be expressed in the following theorem:

Theorem 3.1. *For*

$$a[n] = x[n] + j \cdot y[n] \tag{3.11}$$

$$b[n] = y[n] + i \cdot x[n] \tag{3.12}$$

n is the value of the random variable, which has $n \in N$, and i, j are variable coefficient. the $a[n]$ and $b[n]$'s probability density function (PDF) is:

$$f_{a[n]}(x) = f_{b[n]}(y) = \frac{1}{\sqrt{2\pi N}\sigma} \exp^{-\frac{y}{2\sigma^2 N}}, \tag{3.13}$$

x or y represents the value of the random variable. In the probability density function, it computes the probability density at a given value x or y.

Proof. To prove the probability density function (PDF) for $a[n]$ and $b[n]$, it can calculate from the discrete Fourier transform. In detail, the discrete Fourier transform of this noise is:

$$A[k] = X[k] + j \cdot Y[k] = \Sigma_{n=0}^{N-1}(X[n] + j \cdot Y[n]) \cdot \exp^{-2\pi j \frac{nk}{N}} \tag{3.14}$$

In Eq.(3.14), k and n are the value of the random variable, N is range with n, and its real and imaginary parts are:

$$X[k] = \Sigma_{n=0}^{N}(x[n]cos\frac{2\pi nk}{N} + y[n]sin\frac{2\pi nk}{N}) \tag{3.15}$$

$$Y[k] = \Sigma_{n=0}^{N}(y[n]cos\frac{2\pi nk}{N} - x[n]sin\frac{2\pi nk}{N}) \tag{3.16}$$

In Eq.(3.15) and Eq.(3.16), the difference between $X[k]$ and $Y[k]$ is that $x[n]$ and $y[n]$ exchange positions with each other, and $x[n]$ takes a negative sign in here. Since the probability distributions of $x[n]$ and $y[n]$ are the same, and the Gaussian distribution is an even function, the exchange position does not change the value, that is, $X[k]$ and $Y[k]$ obey the same distribution.

Then for $X[n]$, in $x[n]cos\frac{2\pi nk}{N}$, $cos\frac{2\pi nk}{N}$ is a constant that has nothing to do with the random variable $x[n]$, and is multiplied by the number of the random variable, the

PDF of $x[n]cos\frac{2\pi nk}{N}$ is:

$$f_{x[n]cos\frac{2\pi nk}{N}}(b) = \frac{1}{|cos\frac{2\pi nk}{N}|(b)\sigma\sqrt{2\pi}} \exp^{-\frac{\sigma^2[n]}{2\sigma^2 cos^2\frac{2\pi}{nk}}} \tag{3.17}$$

In Eq.(3.17), $f_{x[n]cos\frac{2\pi nk}{N}}(b)$ represents the Laplace distribution probability density function (PDF) evaluated at the point b. It denotes the probability of observing a value b given the input $x[n]cos\frac{2\pi nk}{N}$, and $x[n]cos\frac{2\pi nk}{N}$ is the main function of interest in the Laplace Mechanism. $x[n]$ is the original data or the sensitive value that needs to be protected. This could be any numerical value or a vector and k is a parameter used to control the sensitivity of the mechanism. A higher value of k makes the mechanism less sensitive, while a lower value increases the sensitivity. N represents the total number of entries in the database or the dataset size. It is used to scale the sensitivity based on the dataset size, b is the observed or desired output value, π denotes the scale parameter of the Laplace distribution. It determines the spread or variability of the noise added to the output, n refers to the sensitivity parameter, which captures the sensitivity of the function to changes in the input data. It is used to adjust the amount of noise added to the output, exp is the exponential function used to calculate the value of the Laplace distribution, $\sqrt{2\pi}$ constant term appears in the normalization factor of the Laplace distribution. $cos\frac{2\pi nk}{N}$ term appears in both the numerator and the denominator. It captures the sensitivity of the function to changes in the input data and scales the noise accordingly. It oscillates between -1 and 1 as the input values change.

Transfer Eq.(3.17) with Laplace Mechanism, it has:

$$Laplace_{x[n]cos\frac{2\pi nk}{N}(s)} = \int f_{x[n]cos\frac{2\pi nk}{N}}(b) \cdot \exp^{s\cdot b} \Delta b = \exp^{\frac{s^2\sigma^2}{2} cos^2\frac{2\pi nk}{N}} \tag{3.18}$$

In Eq.(3.17), $Laplace_{x[n]cos\frac{2\pi nk}{N}(s)}$ represents the Laplace transform of the function, $f_{x[n]cos\frac{2\pi nk}{N}(b)}$ with respect to the variable s. It denotes the Laplace transform of the

Laplace distribution probability density function with the input $x[n]cos\frac{2\pi nk}{N}$, and s is the Laplace transform variable. It is a complex number that is part of the Laplace transformation process. Δb represents the differential or infinitesimal change in the variable b used in the integration process.

From Eq.(3.17) that the Laplace transfer for $y[n]sin\frac{2\pi nk}{N}$ is $\exp^{\frac{s^2\sigma^2}{2}sin^2\frac{2\pi nk}{N}}$. So, the $r[n]$ = $x[n]sin\frac{2\pi nk}{N}$ + $y[n]sin\frac{2\pi nk}{N}$ and its Laplace can transfer to the equation:

$$Laplace_{r[n]}(s) = \exp^{sin^2\frac{2\pi nk}{N}+cos^2\frac{2\pi nk}{N}} = exp^{\frac{s^2\sigma^2}{2}} \tag{3.19}$$

And the Laplace transfer for $X[k]$ is :

$$Laplace_{X[k]}(s) = exp^{\frac{s^2\sigma^2N}{2}} \tag{3.20}$$

In Eq.(3.20), all means of variables are the same with Eq.(3.17), $\frac{s^2\sigma^2N}{2}$ appears in the exponent of the exponential function. It captures the effect of the Laplace transform on the random variable $X[k]$ and its Laplace distribution. It depends on the Laplace transform variable S, and the discrete frequency range n. This equation represents the Laplace transform of the random variable $X[k]$. It provides a mathematical representation of the Laplace distribution in terms of its Laplace transform, incorporating the Laplace transform variable, the scale parameter, and the discrete frequency range. The Laplace transform helps analyze the properties and characteristics of the random variable $X[k]$ in a different mathematical space.

From the Eq.(3.20) the PDF for $X[k]$ is :

$$f_{X[k]}(x) = \frac{1}{\sqrt{2\pi N}\sigma}exp^{-\frac{x}{2\sigma^2N}} \tag{3.21}$$

and

$$f_{Y[k]}(y) = \frac{1}{\sqrt{2\pi N}\sigma} exp^{-\frac{y}{2\sigma^2 N}}. \tag{3.22}$$

In Eq.(3.21) and Eq.(3.22), $f_{X[k]}(x)$ and $f_{Y[k]}(x)$ represents the PDF of the random variable $X[k]$ and $Y[k]$ evaluated at the point x and y separately. It denotes the probability of observing the value x and y for the discrete random variable X and Y at the discrete frequency index k. $X[k]$ and $Y[k]$ are discrete random variables representing the value of interest at the discrete frequency index k. This could be any numerical value or a sequence of values. $-\frac{x}{2\sigma^2 N}$ and $-\frac{y}{2\sigma^2 N}$ term appear in the exponent of the exponential function. It captures the shape and behavior of the PDF. It depends on the observed value x and y, the scale parameter σ, and the discrete frequency range N. These PDFs help analyze the probabilities associated with different values of the random variables in their respective domains.

Proof end. □

From Eq.(3.21) and Eq.(3.22), the standard deviations for Gaussian noise can be generated. On this basis, sampling from differential privacy noise between origin data $X[k]$ and noise data $Y[k]$ generated from $X[k]$ are based on Gaussian noise.

From the privacy noise generation and sampling aspect, the work of the paper (Y. Qu & Cheng, 2011) shows that the level of privacy-preserving can be calculated from the Rényi Entropy of the data. This method can not only introduce the Rényi differential function (RDF) in the theoretical proof to prove the degree of privacy protection (ontology) of the algorithm to the data but also in practical applications, such as face recognition training logs and facial feature analysis and other applications arrive.

The z comes from two datasets x and y, if z contains a class or implementation triple x, y, it means z is an attribute and x, y contains an executive group sets. Its predicate is z. The degree of privacy leakage z can be called punishment degree, and because of the

equation:

$$\begin{aligned} Penalty(z) &= p(x, y|\omega) \\ &= \prod_{i=1}^{I} p(x_i, y_i|\omega) \\ &= \prod_{i=1}^{I} p(x_i|y_{i+1}, \omega)\dot{p}(a_t|b_t, \omega) \end{aligned} \tag{3.23}$$

The Eq.(3.23) is a measure that can be used to rank the privacy leakage or penalty components of the data before being used for training. ω is penalty parameter and I is whole loop times for two datasets (x_i, y_i) in i. In Eq.(3.23), $Penalty(z)$ represents the penalty function or penalty component for the data, denoted by z. It quantifies the penalty associated with certain aspects of the data under the privacy-preserving mechanism. $p(x, y|\omega)$ represents the joint probability distribution of the variables x and y gave the parameter ω. It denotes the probability of observing the values x and y simultaneously under the privacy-preserving mechanism.

If noise is added into an unknown data pair (x, y) sequentially according to (ϵ, δ)-differential privacy, then based on the Markov property,

$$\begin{aligned} p(x_i|y_{1:i-1}, x_{1:i-1}) &= p(y_i|y_{i-1}) \\ p(x_i|x_{1:y-1}, y_{1:i-1}) &= p(x_i|y_{i-1}) \end{aligned} \tag{3.24}$$

In Eq.(3.24) the property between $x_{1:i-1}$ and $y_{1:i-1}$ in status i can be calculated from Markov chain. In this subsection and the following CRF/MDCRF exposition, the symbol p denotes probability or probability density as appropriate (e.g. $p(x)$, $p(x|y)$, or $p_t(\cdot)$ for state distributions at time t).

Let's define the probability of x as $p(x)$, then assume that its conditional field is

Gaussian distributed. Then,

$$\begin{aligned} p(x) &= \frac{1}{Z}\sum_{t=1}^{T}\eta_t(x_{t-1}) \\ &= \frac{1}{Z}\sum_{t=1}^{T}\exp[-E_t(x_{t-1})] \\ &= \frac{1}{Z}\exp\sum_{t=1}^{T}E_t(x_{t-1}) \end{aligned} \tag{3.25}$$

The Eq.(3.25) can obtain the value of $p(x)$ from the expectation of x_{t-1} and the privacy leakage range Z.

$\eta_t(x_{t-1})$ term represents a function that determines the contribution or weight η_t of the previous value x_{t-1} at time step t to the overall probability distribution. The specific form of this function depends on the context and problem being addressed and $\frac{1}{Z}$ represents the normalization factor or the partition function Z used to ensure that the probability distribution integrates to 1 over the entire range of possible values of x. $E_t(x_{t-1})$ represents the cost associated with the value x_{t-1} at time step t in the conditional field. It is a measure of how well the value x_{t-1} fits or aligns with the desired distribution. Consequently, the mutually exclusive conditions $p(x|y)$ and $p(y|x)$ in the two Markov chains are deduced by the following formulas respectively,

$$p(x|y) = \frac{1}{Z}exp\sum_{t=1}^{T}[\sum_{k=1}^{K}\omega_k g_k(y_{t-1}, y_t, x_{1:t}) + \sum_{l=1}^{L}\alpha_l h_l(y_t, x_{1:t})] \tag{3.26}$$

In Eq.(3.26), $p(x|y)$ represents the conditional probability distribution of the variable x given the variable y. It denotes the probability of observing the value x given the value y in the context of a Markov chain. ω_k represents the weight or coefficient associated with the term k in the first summation. It captures the importance or contribution of the term to the overall probability distribution. $g_k(y_{t-1}, y_t, x_{1:t})$ represents a function that depends on the previous value $y - t - 1$, the current value y_t, and the history of values $x_{1:t}$ up to time step t. It captures the dynamics probabilities between states in the

Markov chain. And $h_l(y_t, x_{1:t})$ represents a function that depends on the current value y_t and the history of values $x_{1:t}$ up to time step t. It captures the influence or impact of the current state on the observed variable x in the Markov chain.

$$\begin{aligned} p(y|x) &= \frac{1}{z}\exp\sum_{u=1}^{U} F_u(X_t) \\ &= \frac{1}{Z}\exp\sum_{t=1}^{T} F(y_{t-1}, y_t, x_{1:T}) \\ &= \frac{1}{Z}\exp(\Delta y_{t-1}, x_{1:T} + \Delta y_t, x_{1:T} + \Delta y_{t-1}, y_t, x_{1:T}) \\ &= \frac{1}{Z}\exp[\sum_{t=1}^{T}\omega_t g_t(y_{t-1}, x_{1:T}) \\ &\quad + \sum_{u=1}^{U}\alpha h(y_u, x_{1:U}) + \sum^{V}\beta_v i_v(y_{v-1}, y_v, x_{1:V})] \end{aligned} \tag{3.27}$$

In Eq.(3.27), $\Delta y_{t-1}, x_{1:T} + \Delta y_t, x_{1:T} + \Delta y_{t-1}, y_t, x_{1:T}$, represents a combination of terms that involve differences or changes in y and x at each time step t in the Markov chain. α represents the weight or coefficient associated with the term h in the second summation. It captures the importance or contribution of the term to the overall conditional probability distribution. The Eq.(3.27) involving the parameters, where $\Delta y_{t-1}, x_{1:T}$ and $\Delta y_t, x_{1:T}$ are conditional part and $\Delta y_{t-1}, y_t, x_{1:T}$ is transfer part in the exponential. Note that,

$$\begin{aligned} p(T_x = t|U) &= \sum_{u_1, u_2 \ldots u_{x-1} y_{t+1} \ldots y_T} P(y|x) \\ &= \sum_{y<1,t-1>} \sum_{y<t+1,T>} \frac{1}{z}\prod_{t'=1}^{T}\Theta(y_{t'-1}, y_{t'}, x) \\ &= \frac{1}{z}\Delta X \Delta Y \end{aligned} \tag{3.28}$$

In Eq.(3.28), for $y_t \in S$, and the result of probability $p(T_x|U)$ has been deducted. $\sum_{u_1, u_2 \ldots u_{x-1} y_{t+1} \ldots y_T}$ represents the summation operation over all possible combinations

of variables $u_1, u_2 \ldots u_{x-1} y_{t+1} \ldots y_T$. It indicates the sum over all possible values of the preceding variables and subsequent variables in the sequence. $\Theta(y_{t'-1}, y_{t'}, x)$ represents a function that depends on the previous state y_{t-1}, the current statey'_t, and the value of the variable x. It captures the contribution or effect of the states and the observed variable on the overall probability distribution.

Then the ΔX and ΔY can be calculated separately,

$$\Delta X = \sum_{y<1:t-1>} \Theta_1(y_0, x_1, x)\Theta_2(y_1, y_2, x) \ldots \Theta_{i-1}(y_{t-2,y_{t-1},x})\Theta_t(y_{t-1}, y_t, x) \quad (3.29)$$

and

$$\Delta Y = \sum_{y<t+1:T>} \Theta_{t+1}(y_t = i, y_{t+1}, x) \ldots \Theta_T(y_{T+1}, y_T, x) \quad (3.30)$$

For the calculation of $p(T_x = t|U)$ in sampling process, the result of Eq.(3.29) and Eq.(3.30) can bring to the Eq.(3.28), and $\Theta_1(y_i, x_j, x)$ represent certain status of the Markov process when point out x_i and y_j and $(i, j) \in T$.

The optimal value could be computed using a gradient ascent algorithm with objective function

$$\begin{aligned}
\hat{\phi}\hat{\eta} &= \arg\max_{\phi,\eta} \sum_{i=1}^{I} p(y^{(i)}|x^{(i)}) \\
&= \arg\max_{\phi,\eta} \log \prod_{n=1}^{N} p(y^(i)|x^(i)) \\
&= \arg\max_{\phi,\eta} \sum_{n=1}^{N} \log p(y^{(i)}|x^(i)) \\
&= \arg\max_{\phi,\eta} \sum_{n=1}^{N} \{-\log C(X^i, \phi, \eta) \\
&\quad + \sum_{i=1}^{T} [\phi^T \cdot g(y_{t-1}^{(i)}, y_t^{(i)}, x^{(i)}) + \eta^T \cdot h(y_t^{(i)}, x^{(i)}))]\} \\
&= \arg\max_{\phi,\eta} L(\phi, \eta, x^{(i)})
\end{aligned} \quad (3.31)$$

In Eq.(3.31), the $\hat{\eta}$ and $\hat{\phi}$ represent the parameters of learning rate and iteration angle

in stochastic gradient descent(SGD) separately. And $\arg\max_{\phi,\eta}$ notation indicates that the SGD finding the values of $\hat{\eta}$ and $\hat{\phi}$ that maximize the objective function that follows. $\sum_{i=1}^{I} p(y^{(i)}|x^{(i)})$ represents the summation of the probability $p(y^{(i)}|x^{(i)})$ over all training instances i. It indicates the $h(y_t^{(i)}, x^{(i)}))$. It captures the influence of the features on the model's output. $\arg\max_{\phi,\eta} L(\phi, \eta, x^{(i)})$ indicates that it is finding the values of $\hat{\eta}$ and $\hat{\phi}$ that maximize the objective function $L(\phi, \eta, x^{(i)})$. The objective function represents the log-likelihood or log-loss, and the equation aim to find the parameters that yield the highest likelihood or lowest loss.

Now, the iterative gradient at a certain angle has the following result:

$$\nabla_{\phi\eta} L = \sum_{n=1}^{N} \left[\sum_{t=1}^{T} \phi^T(y_{t-1}, y_t, x^{(i)}) - \nabla_{\phi\eta} \log Z(x^{(i)}, \lambda, \eta) \right] \tag{3.32}$$

In Eq.(3.32), $\nabla_{\phi\eta} L$ denotes the gradient of the objective function L with respect to the parameters ϕ and η. It represents the vector of partial derivatives of L with respect to ϕ and η. $\phi^T(y_{t-1}, y_t, x^{(i)}) - \nabla_{\phi\eta}$ represents the dot product between the parameter vector θ and the feature vector $y_{t-1}, y_t, x^{(i)}$. It captures the influence of the features on the model's output at time step t for the nth training instance and $\nabla_{\phi\eta} \log Z(x^{(i)}, \lambda, \eta)$ represents the gradient of the logarithm of the normalization constant Z with respect to the parameters ϕ and η. The normalization constant Z is a factor that ensures the conditional probabilities sum up to 1. This term represents the influence of the parameters on the normalization constant.

In Eq.(3.32), the gradient term on the right side can be expressed as

$$\begin{aligned}
\nabla_{\phi\eta} \log Z(x^{(i)}, \phi, \eta) &= \exp\left[\sum_{t=1}^{T} g(y_{t-1}, y_t, x^{(i)})\right] \\
&= \sum_{y} p(y|x^{(i)}) \cdot \sum_{t=1}^{T} g(by_{t-1}, y_t, x^{(i)}) \\
&= \sum_{t=1}^{T} [\sum_{y} p(y|x^{(i)}) \cdot g(y_{t-1}, y_t, x^{(i)})] \\
&= \sum_{t=1}^{T} \sum_{y_{<1:t-2>}} \sum_{y_{t-1}} \sum_{y_t} \sum_{y_{<t+1,T>}} p(y|x^{(i)}) \cdot g(0) \\
&= \sum_{t=1}^{T} \sum_{y_{t-1}} \sum_{y_t} [\sum_{y_{<1,y-2>}} \sum_{y_{<t+1,T>}} p(y|x^{(i)}) g(0)] \\
&= \sum_{t=1}^{T} \sum_{y_{t-1}} \sum_{y_t} p(y_{t-1}, y_t, x^{(i)}) g(y_{t-1}, y_t, x^{(i)}) \qquad (3.33)
\end{aligned}$$

In Eq.(3.33), the gradient term in a certain direction can be calculated, and the result shows that the value of the right side of Eq.(3.32) has been affected by the status of Markov property and gradient weight decay.

In Eq.(3.33), the $\nabla_{\phi\eta}$ denotes the gradient with respect to the variables ϕ and η. $\log Z(x^{(i)}, \phi, \eta)$ represents the logarithm of the partition function Z evaluated at the input $x^{(i)}$ with parameters ϕ and η. This term typically arises in the context of probabilistic graphical models or log-linear models. $x^{(i)}$ represents the input indexed by i and y represents a variable or assignment in the model. In this equation, it represents a sequence of variables related to the model. g represents a function that depends on the variables y_{t-1}, y_t, and $x^{(i)}$. The specific form of this function is not provided in the equation. $p(y|x^{(i)})$ represents the conditional probability of the variables y given the input $x^{(i)}$. $y_{<1:t-2>}$, y_{t-1}, y_t, $y_{<t+1,T>}$ represent subsets or ranges of the variables y within the sequence. The notation $y_{<1:t-2>}$ refers to all variables y before the time step $t - 2$, y_{t-1} represents the variable at time step $t - 1$, y_t represents the variable at time step t, and $y_{<t+1,T>}$ refers to all variables y after time step $t + 1$ up to T. The gradient term in

this equation has a certain direction, and the result shows that the value of the right side of Eq.(3.32) has been affected by the status of Markov property and gradient weight decay.

Therefore,

$$\nabla_{\phi\eta} L = \sum_{n=1}^{N} \sum_{t=1}^{T} \left[g(y_{t-1}, y_t, x^{(i)}) - \sum_{y_{t-1}} \sum_{y_t} p(y_{t-1}, y_t \cdot g(y_{t-1}, y_t, x^{(i)})) \right] \tag{3.34}$$

The Eq.(3.34) represents the iterative gradient function at a certain angle. In this equation, $\nabla_{\phi\eta} L$ denotes the gradient of the loss function L with respect to the variables ϕ and η. This gradient represents the direction of the steepest ascent or descent for optimizing the loss function. N represents the whole number of samples in the dataset and n is an index that ranges from 1 to N, representing individual samples in the dataset. Then, $\sum_{t=1}^{T}$: It represents a summation over time steps. The variable T denotes the total number of time steps in the sequence and $\phi^T(y_{t-1}, y_t, x^{(i)})$ represents a function that depends on the variables y_{t-1}, y_t, and $x^{(i)}$. The specific form of this function is not provided in the equation. The superscript T suggests that it might be a parameterized function. Lastly, $\nabla_{\phi\eta} \log Z(x^{(i)}, \lambda, \eta)$ represents the gradient of the logarithm of the partition function Z with respect to the variables ϕ and η. The logarithm of the partition function is typically used in probabilistic graphical models or log-linear models. The parameters λ and η are involved in this gradient calculation. The $g(y_{t-1}, y_t, x^{(i)})$ is the gradient ascent parameter, it will calculate the gradient descent trend. The gradient ascent update equations are

$$\begin{cases} \phi^{(t+1)} = \phi^{(t)} + step \nabla \phi L(\phi^{(t)}, \eta^{(t)}) \\ \eta^{(t+1)} = \eta^{(t)} + step \nabla \eta L(\phi^{(t)}, \eta^{(t)}) \end{cases} \tag{3.35}$$

In Eq.(3.35), $\phi^{(t+1)}$ and $\phi^{(t)}$ represent the values of the variable ϕ at time step $t + 1$ and t, respectively. The variable ϕ is being updated through iterations of the gradient

ascent algorithm. $\eta^{(t+1)}$ and $\eta^{(t)}$ represent the values of the variable η at time step $t+1$ and t, respectively. Similar to ϕ, η is being updated through iterations of the gradient ascent algorithm. And $step$ represents the step size or learning rate, which determines the magnitude of the update at each iteration. It controls the speed of convergence and the stability of the optimization process. Then, $\nabla\phi L(\phi^{(t)}, \eta^{(t)})$ and $\nabla\eta L(\phi^{(t)}, \eta^{(t)})$ represent the gradients of the loss function L with respect to the variables ϕ and η, respectively. The gradients indicate the direction and magnitude of the steepest ascent for optimizing the loss function. Lastly, $L(\phi^{(t)}, \eta^{(t)})$ represents the value of the loss function L evaluated at the current values of ϕ and η at time step t. The loss function measures the discrepancy or error of the model's predictions compared to the true values. In this equation, the parameters can be calculated in each step, and the Eq.(3.31) will reflect the details of the calculation.

This section explains in detail the sampling and generation methods of privacy noise. As the sub-question mentioned in Chapter 1.2, in order to be able to track and control the process of adding machine learning to privacy noise, the next section will focus on this process explores, and attempt to propose effective solutions.

3.5 Discernibility Matrix for Noise Generation

In this section, a method is proposed to integrate K-L divergence into the MDCRF model to quantify privacy noise and monitor the process of noise addition in the machine learning process. The proposed solution aims to identify image features and locate similar sensitive regions based on sensitive classes. This novel method will be referred to as Normalized Control Penalty (NCP). NCP quantifies the information loss associated with model accuracy.

As Figure 3.1 shows, after the sensitive class is recognized via MDCRF, the next stage is to reconstruct the discernibility matrix according to the input data set. The

discernibility metric (DM) (Yao & Zhao, 2009)) or discernibility matrix (Zhou & Yang, 2003) (Terrovitis, Mamoulis & Kalnis, 2011) allow the cardinality of the sensitive feature classes to be measured. A graduated increased edge penalty (IEP) (Q. Yu & Clausi, 2008) has been proposed to control the segmentation under non-stationary conditions. Extended from IEP, a control edge penalty method is proposed (Qin & Clausi, 2010), where the multivariate iterative region grows with an increased edge penalty method to enhance segmentation efficiency.

In the first stage, the proposed solution needs to consider numerical attributes. Let D be a data source with quasi-indexs $(A_1, \ldots, A_n)$. Suppose the tuple $d = (x_1, \ldots, x_n)$ is generalized to the tuple $d' = ([y_1, z_1], \ldots, [y_n, z_n])$. Then, a normalized deterministic penalty of the tuple t for the attribute A_i is defined as

$$NCP_{A_i}(t) = \frac{\{z_i\} - \{y_i\}}{\{x_i\} - \{y_i\}} \tag{3.36}$$

In Eq.(3.36), $NCP_{A_i}(t)$ denotes the normalized deterministic penalty for the tuple t specifically related to the attribute A_i. The NCP quantifies the penalty or discrepancy of the attribute value for the tuple compared to some reference values. And z_i represents the value of the attribute A_i for the tuple t under consideration. This value is the observed or actual value associated with the tuple. y_i represents a reference or target value for the attribute A_i. This value is typically a desired or expected value for the attribute. x_i represents another reference or baseline value for the attribute A_i. This value is typically a minimum or maximum threshold value for the attribute. The NCP can be used to assess the relative penalty or bias of nodes in a spanning tree compared to some reference value. It provides a measure of how far an observed value deviates from an expected or expected value, taking into account the generalization of the corresponding

value in graph theory. Then, this equation has

$$|A_i| = \max_{t \epsilon D} t_{A_i} - \min_{t \epsilon D} t_{A_i} \tag{3.37}$$

In Eq.(3.37), $|A_i|$ represents the absolute range of values for the attribute A_i. This value indicates the maximum difference between the highest and lowest values observed for the attribute A_i in the dataset. And $\max_{t \epsilon D} t_{A_i}$ represents the maximum value observed for the attribute A_i among all tuples t in the dataset D. This value is the largest value of attribute A_i present in the dataset. $\min_{t \epsilon D} t_{A_i}$ represents the minimum value observed for the attribute A_i among all tuples t in the dataset D. This value is the smallest value of attribute A_i present in the dataset. Intuitively, $NCP_{A_i}(t)$ in Eq.(3.36) measures how many parts in the spinning tree is generalized on the numerical attribute A_i in MDCRF, and the numerical attribute A_i for the tracking objects in MDCRF can be calculated from Eq.(3.37).

For classified, sensitive attributes, a hierarchical tree can be established from the spanning tree. It specifies attribute values with different regularities. After scanning the input feature, the hierarchical tree can be searched according to its similar sensitive attributions. At position t, the attribute A_i of the dataset D has classification result v, it can be generalized to $v_1, \ldots, v_m$. In the spinning tree, let **ancestor**$(v_1, \ldots, v_m)$ denote the common ancestors of $(v_1, \ldots, v_m)$. Also, let $(l_1, \ldots, l_m)$, $(l_1, \ldots, l_n)$ be the leaf nodes of **ancestor**$(v_1, \ldots, v_m)$. The relationship among the ancestor nodes and their leaf node quantitatively measures generalization. This gives the equation:

$$NCP_{A_i}(t) = \frac{|ancestor(v_1, \ldots, v_m)|}{A_i} \tag{3.38}$$

Through $NCP_{A_i}(t)$, the data difference of matrix A_i at time t can be measured to measure the difference of its categorical attributes in Eq.(3.38).

A weighted sum of the normalized deterministic costs for all attributes can be obtained as

$$NCP(t) = \sum_{i=1}^{n}(w_i \cdot NCP_{A_i}(t)) \tag{3.39}$$

In Eq.(3.39), with the constraint $\sum_{i=1}^{n} w_i = 1$ and the normal control penalty of A_i in the position t of spinning tree T is NCP_{A_i}. Thus, the penalty on the entire dataset in the spinning tree T is the sum of the penalties for all tuples:

$$NCP(T) = \sum_{t \epsilon T} NCP(t) \tag{3.40}$$

In Eq.(3.40), the weights from $NCP(T)$ allow different utilities of an attribute for different applications to be reflected.

Assume the item feature F has feature quantity the value as f. And the n_f is its leaf quantity with spinning tree T. It has

$$NCP(f) = \begin{cases} 0, & |T_f| = 1 \\ \dfrac{|T_f|}{|F|}, & otherwise \end{cases} \tag{3.41}$$

In Eq.(3.41), $NCP(f)$ represents the normalized deterministic penalty for the feature f. This penalty quantifies the degree of penalty or discrepancy associated with the feature. And $|T_f|$ denotes the number of unique values observed for the feature f that has been generalization. In other words, it represents the count of distinct values for the feature in the dataset. $|F|$ represents the total number of tuples (or instances) in the dataset. Lastly, $|T_f|$ represents the tree that has been generalization.

$NCP(f)$ is measured by spanning the relationship between the two sets of related data sets D and D'. Each leaf node indicates that one related data in the two sets of data forms a generalization, while each generalization updates each leaf node to achieve a better generalization. Therefore, each associated data can be generalized and updated

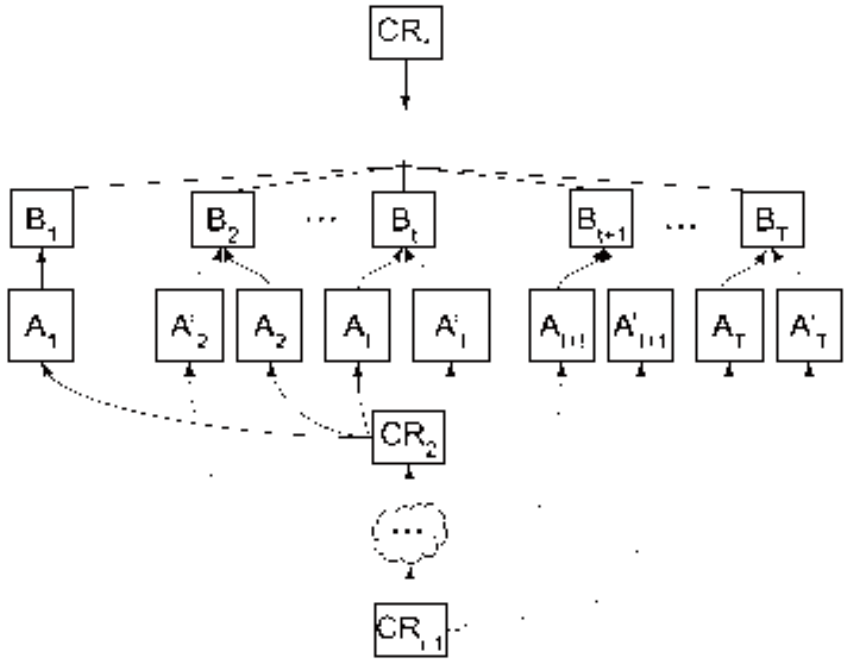


Figure 3.2: The input streaming data on MDCRF

through NCP, so that the data relationship tree can achieve a better generalization effect with lower information loss. For example, in the hierarchy shown in Figure 3.2, the information loss of A_1 is $NCP(A_1) = \frac{n}{2n} = 0.5$. Since usually two nodes A_n and A'_n generate one leaf node B_n, the number of ancestors approaches $2n$. The NCP of the entire data uses the ratio of the appearance of elements affected by the total elements in the datasets to weigh the information loss of each broad item. Define the number of occurrences of the data item p in the dataset D as C_p, then the information loss that occurs because the data items are aggregated and generalized is calculated by the following formula:

$$NCP(D) = \frac{\sum_{p \in I} C_p \cdot NCP(p)}{\sum_{p \in I} C_p} \tag{3.42}$$

In specific cases, in Eq.(3.42), the entropy loss of information can be controlled in the range $[0, 1]$, so that all privacy noise ranges can be quantified. If we scan dataset D

once, the support of all items can be used to measure any generalized information loss without accessing the datasets again. For example, the information loss due to cutting $\{a_1, a_2\} \longrightarrow A$ in Figure 3.2 is

$$\frac{A_t \times 0.5 + B_t \times 0.5 + CR_2 \times 0 + CR_1 \times 0}{t} = \frac{(A_t + B_t) \times 0.5}{t} \tag{3.43}$$

In the loss function, the NCP of A_t and B_t is 0.5, which has been obtained above. CR_1 CR_2 is zero due to both tree levels having only one node each. The loss is therefore $\frac{(A_t+B_t)\times 0.5}{t}$.

Table 3.1: Example of frequent itemsets for label tags

t_1	t_2	t_3	t_4
$\{S_1, C_1, S_1\}$	$\{S_1, C_1\}$	$\{S_1, C_1, S_2\}$	$\{S_2, S_1, C_2\}$
t'_1	t'_2	t'_3	t'_4
$\{S_3, C_1, S_2\}$	$\{S_3, C_2\}$	$\{C_2, C_3, C_1\}$	$\{S_3, C_3\}$

In order to account for the impact of anonymization on the data mining results of anonymous data, a more specialized information loss measure for NCP, known as Multi-level Normalization Loss (MNL), needs to be introduced. If a set of data is already public, then ML2 can detect the loss of information entropy after its generalization, which is more difficult than the frequency of occurrence of the original data set.

Clustering datasets at multiple levels of generalization can help track sensitive features because it allows detection of frequent association rules and frequent itemsets that might not appear in feature classes. For example, suppose we have a dataset of shop signs that appear on buildings. If we discuss the granularity of its generalization, many feature associations between the original dataset and the generalized dataset may be very abstract and cannot be detected by the above algorithms. From the abstract level of generalization, its characteristics may be observed. For example, the attributes

of goods are often associated with specific locations, such as drugs often appear in pharmacies (Liu, Liu & Sun, 2014), and such attacks through background knowledge are generalized The privacy of data is more difficult to protect. The MNL method can detect the frequency of generalized public datasets in the original dataset in data mining, so that the generalization results of the input datasets for machine learning can be effectively compared.

However, if the data set is processed with privacy protection, although its sensitive features have a certain degree of loss after generalization, the MNL method can still detect the degree of association with the original data set, so that any privacy protection algorithm can be detected by the MNL method. Detect how well it relates to the original dataset. The loss of information entropy can also be accurately quantified. Mathematically, MNL is defined as

$$MNL = 1 - \frac{\sum_i^h FI_i(D_p)}{\sum_i^h FI_i(D)} \tag{3.44}$$

In Eq.(3.44), FI_i is the number of frequent itemsets of generalization level i, and D_p and D represent the published and original data sets respectively.

In input datasets, if the information suffers from entropy loss due to generalization, the measurement of the generalization level needs to take these entropy losses into account, including using the loss function to determine the loss. For example, for the target detection task, the detected data class trucks S often appear with the data class of another class of cars C, you can get $\{S_3, C_1, S_2\}$ and $\{S_3, C_2\}$ can finally be generalized to a certain level to form the same class, and the loss of accuracy caused by truck generalization will also be considered. That is, after the truck is generalized, its confidence level drops 3.1. Since such frequent itemsets appear in level l(vehicle), the generalized form of more itemsets should appear in all levels $l' > l$. In an anonymous dataset, generalization may hide the association of $\{S_2, S_1, C_2\}$ or

$\{S_3, C_2\}$, so the itemset representing this correlation may first be generated at the level of $l+d$ generalization, giving $\{S_1, C_1, S_1\}$ and $\{C_2, C_3, C_1\}$ respectively. The difference between levels l and $l + d$ of the frequency itemset in the original data can be computed as

$$d_{MNL} = \frac{\sum_i^h \sum_{fi \epsilon FI_i(D)} d_{level}(fi, D, D_p)}{\sum_i^h FI_i(D)} \tag{3.45}$$

In particular of Eq.(3.45), after inputting the original data, D is generalized several times to generate a new generalization dataset D_p, the generalization level $d_{level}(fi, D, D_p)$ indicates that if there is a set of suspected privacy categories The difference between the number of occurrences of the original dataset D and the frequency of the dataset D_p that appears in the generalized p times of the data set D_p. In Eq.(3.45), $FI_i(D)$ indicates the privacy sensitivity of the privacy category. If several data categories containing privacy classes are generalized and aggregated together, the sensitivity level of the sensitive class generated by this generalization will be lower than that of the original sensitive data class, such as credit card numbers without specific encoding and other data that do not involve privacy. Membership card numbers without specific encoding are generalized into one category, and it is difficult for privacy attackers to determine which are credit card numbers and which are membership card numbers.

3.5.1 Discernible Metric

This thesis proposed a new idea that uses a form of classification metric to optimize the k-sensitive dataset for training classifiers is now put forward. This classification metric is defined by the sum of individual penalties for each row, normalized by the total k rows. Mathematically,

$$M(D^*) = \frac{\sum_r Pen(d)}{N} \tag{3.46}$$

In Eq.(3.46), a row d is penalized, i.e. the $Pen(d)$ is the penalty parameter, and its value is 1 if it is suppressed or its class label is not the majority class label of its group, otherwise, the penalty parameter is 0. The utility value of the Eq.(3.46) can reach a high level if the solution prepares to classify the data after the processing of sensitive data.

Bayardo (Bayardo & Agrawal, 2005) proposed a method for discernible metrics (DM). The gist of Bayardo's work is that, by transforming the dataset, if there are N tuples that cannot be further divided, label N tuples and assign a set of weights. Based on Bayardo's idea, this thesis calculates the generalization quality of the data set through iterative calculation, to obtain the correlation between sensitive data and track the input private data. In iterative computing, for a data set, the more tuples that cannot be generalized, the less likely it is to generalize. In the proposed scheme, define t as a tuple from the original dataset D, and let D^* as one of the tuples $G_T^*(c)$ indistinguishable from T^* in the sensitive data D', the scheme defines c as privacy cost, the tuple is equivalent to the sensitive value of t. The work for the proposed DM calculation has been described as:

$$DM(T^*) = \sum_{c \in T} G_{T^*}(c) \tag{3.47}$$

In Eq.(3.47), $DM(T^*)$ denotes the discernible metric for the set of tuples T^*. The discernible metric is a measure that quantifies some aspect or characteristic of the tuples in the set. And $\sum_{c \in T}$ represents the summation over each tuple c in the set T. The set T is the set of all tuples, and the summation is performed over each tuple in this set. $G_{T^*}(c)$ represents the discernibility function for a tuple c with respect to the set of tuples T^*. The discernibility function captures the contribution or impact of the tuple c towards the discernible metric for the set T^*. In summary, the equation calculates the discernible metric for a set of tuples T^* by summing up the discernibility function $G_{T^*}(c)$ for each tuple c in the set T. The discernible metric provides a measure

or score that captures a specific characteristic or property of the tuples in the set T^*. The discernibility function determines how much each individual tuple contributes to this metric. In addition, assuming that the value of the tuple is in the range t, it's $G_{T^*}(c) = T^*$. This reduces the utility of the data. The best approach is to avoid tuple values being limited to the range c.

In Eq.(3.47), if $M(T) > M(T')$, it means that the data generalization quality of T' is better than T. For the measurement of data quality, this thesis draws on two works of Nergiz (Nergiz & Clifton, 2007). Nergiz's work defined the criteria that can be used as a measure of data quality and introduces the evaluation criteria themselves in order to achieve data quality. Its work demonstrates experimentally that classification matrices are better than discriminative matrices in classification applications, but discriminative matrices perform better in searching association rules. In order to verify which scheme is more suitable for a specific scenario, data quality measurement efforts need to correlate experiments with specific tasks and scenarios of machine learning.

Bertino and Sweeney separately discuss the loss of information search resulting from privacy protection in classification matrices (Bertino & Fovino, 2005; Sweeney, 2002). However, the construction of feature classification for different scenarios needs further resolution. Currently, there are no unified rules and methods established for these two tasks. Also, if the data is used to build a classifier, these two metrics will also work fine. Kifer (Kifer & Gehrke, 2006) proposed a utility measure related to the Kullback-Leibler divergence as a utility indicator. Preliminary results from Kifer's work show that this metric works very well in practice. With the help of the utility matrix proposed by Kifer, this thesis measures the effect of privacy protection by adding a utility matrix to measure sensitive data sets and establishes associations. Due to the different data characteristics, these differences can be accurately quantified by combining the methods proposed later in this thesis.

As described in the paper (Sweeney, 2002), in statistical models for perturbations,

the information loss reflects the loss of accuracy in estimating the original distribution function for a given attribute. Trigger from Sweeney's work, the solution in the thesis measuring the estimated feature X requires validating the density function $f_X(x)$ before and after the data is reconstructed to measure the loss caused by information destruction and reconstruction. It has:

$$I(f_X, \widehat{f_X}) = \frac{1}{2} E\left[\int_{\Omega_X} \left|f_X(x) - \widehat{f_X}(x)\right| dx\right]. \tag{3.48}$$

Eq.(3.48) represents the reconstructed loss for measuring the discrepancy between the true function f_X and its reconstructed approximation $\widehat{f_X}$. When the two inputs coincide, i.e. when $\widehat{f_X} = f_X$, the integrand $|f_X(x) - \widehat{f_X}(x)|$ vanishes and the loss $I(f_X, \widehat{f_X})$ equals zero. In this equation, $I(f_X, \widehat{f_X})$ represents the reconstructed loss, which quantifies the difference or discrepancy between the true function f_X and its reconstructed approximation $\widehat{f_X}$ and $\frac{1}{2}$is a constant scaling factor that is applied to the loss function. Then, $E[\cdot]$ denotes the expectation operator, which computes the average value of the expression inside the brackets. $\int_{\Omega_X}$ represents the integration over the domain Ω_X, which is the region or space where the true function and its reconstructed approximation are defined. $f_X(x)$ denotes the true function at a specific point x in the domain. This represents the actual value or output of the true function. And $\widehat{f_X}(x)$ represents the reconstructed approximation of the true function f_X at the same point x in the domain. This is the estimated or predicted value of the function based on the reconstruction method. Lastly, dx indicates an infinitesimal element or differential element of the integration, representing the small region or interval over which the integration is performed. The equation calculates the reconstructed loss by integrating the absolute difference between the true function $f_X(x)$ and its reconstructed approximation $\widehat{f_X}(x)$ over the domain Ω_X, and then taking the expectation of this integrated difference. The reconstructed loss measures the overall discrepancy or error

between the true function and its reconstructed approximation, providing a quantitative assessment of the quality of the reconstruction. The factor of $\frac{1}{2}$ is often used for mathematical convenience or to ensure symmetry in certain optimization or training algorithms.

This section uses Discernible Metrics to track the characteristics of mixed privacy data datasets throughout the process. Based on this scheme, data sampling based on privacy noise can be better monitored and quantified.

3.5.2 Indicator for Privacy Budget Measurement

The work in this section will be used later in the data sampling processing. This part of the work can effectively connect the user-defined privacy budget in the sampling and generation of existing privacy noise.

For the privacy budget of ϵ, the allocation of the privacy budget in the three steps of the algorithm is the core problem to be solved in measuring the privacy budget. Recall that the budget for the first step is ε_{Node}, the budget for the second step is $\varepsilon_{Edge} = \varepsilon_{cnt} + \varepsilon_{Node}$, and the last step is to build the array of boundary points There is ε_M. Usually, ε_{cnt} and ε_M allocate larger budgets. There are two reasons. The first reason is that vertex relabeling search hardly constitutes a utility loss during the search. The second reason is that, as long as you get relatively accurate edge counts, you can always find denser (or sparser) subregions and thus reconstruct them more accurately. Between ε_{cnt} and ε_A, ε_{cnt} can be given more budget because it can be recovered with sufficient accuracy regardless of the privacy budget.

In the second step, since the spanning tree T is not a perfect tree, the depth of the leaf nodes has a relationship of $i < h - 1$. From the perspective of tree division, the privacy budget of the leaf nodes has become $\frac{(2^{h/3}-2^{(i+1)/3})\epsilon_{cnt}}{2^{(h+1)/3}-1}$. This will be added to ϵ_A so that the privacy budget can be fully utilized. Therefore, each leaf node in $Tree$ will

recalculate its privacy budget during the reconstruction of the leaf area, and its privacy budget conforms to ε_M.

Each internal node in the relationship tree has the feature of the four branches at each node in the quad-tree, which can be computed via the exponential mechanism. A simple implementation has runtime complexity $O(|N|^4)$. The time complexity of the data sum of its adjacency matrix can be reduced to $O(|N|^2)$, so the computation time can be reduced due to the iteration of the algorithm.

Definition 3.2 (The quantity of matrix search time)**.** *If it has an adjacency matrix* $M = \{\alpha_{ij}\}$ *as their origin graph* $G = (N, E)$*, the quantity of matrix search time* S*'s in matrix* M *has elements given by* $S[i, j] = \sum_{m=1}^{i} \sum_{m=1}^{j} \alpha_{ml}$.

This quantity of matrix search time can be constituted with the $O(|N|^2)$ as its time complexity. The entire process to search the quantity is only processed one time for each loop.

The density in a specific area can be computed from node position (p, q) in the quantity of matrix search time S of A by

$$den(A[i, j; m, n]) = \frac{\sum_{p=j}^{i} \sum_{q=m}^{n} A_{pq}}{(n - m + 1)(j = i + 1)} \tag{3.49}$$

The Eq.(3.49) $den(A[i, j; m, n])$ represents the density of the matrix A in the specific area defined by the node positions (p, q) and the search time quantity S. The density refers to the measure of how much of the area is occupied or filled by non-zero elements of the matrix. And A_{pq} represents the value of the matrix A at the position (p, q). Each element of the matrix A is denoted by A_{pq}, where p represents the row index and q represents the column index. i, j, m, n variables represent the indices or positions of the nodes within the matrix. Specifically, i and j represent the row indices for the starting and ending nodes, respectively, in the horizontal direction, and m and n represent the column indices for the starting and ending nodes, respectively, in the

vertical direction. $\sum_{p=j}^{i}\sum_{q=m}^{n} A_{pq}$ represents the summation of the matrix elements within the area defined by the node positions (p, q). The double summation iterates over the rows and columns within the specified range and sums up the corresponding matrix elements. $(n - m + 1)(j = i + 1)$ represents the total number of elements in the area defined by the node positions (p, q). The product $(n - m + 1)$ represents the number of columns, and $(j = i + 1)$ represents the number of rows. Multiplying these two values gives the total number of elements in the specified area. This equation computes the density of matrix A within a specific area by summing up the values of the matrix elements within that area and dividing it by the total number of elements in the area. The density provides a measure of how densely the specific area is filled with non-zero values, indicating the concentration or sparsity of data within that region of the matrix. Moreover, the time complexity of $O(1)$ for Eq.(3.49) and search time can be calculated from $[i, j]$ and $[m, n]$.

$$\begin{aligned} QMST(M[6,2;6,5]) &= (S[5,6] - S[5,1] - S[1,6] + S[1,1])/12 \\ &= (6 - 0 - 1 + 0)/12 \\ &= \frac{5}{12} \end{aligned}$$

Eq.(3.5.2) calculates the quality metric of a minimum spanning tree (QMST) for a specific submatrix $M[6, 2; 6, 5]$ based on the values of the search time matrix S. $QMST(M[6, 2; 6, 5])$ represents the quality metric of the minimum spanning tree for the submatrix $M[6, 2; 6, 5]$. The QMST is a measure that evaluates the quality or efficiency of a minimum spanning tree in a graph or network. The resulting value $\frac{5}{12}$ represents the quality metric of the minimum spanning tree for the specified submatrix. This metric provides a measure of the quality or efficiency of the minimum spanning tree in terms of the search time values in the submatrix. If in Eq.(3.5.2), in the worst

case, the complexity of the datasets required to calculate the privacy budget is high, the search frequency of tree branches needs to be extended to compensate for the loss of accuracy, and extended branches will incur a cost of higher time complexity.

3.5.3 Information Loss for Data Sampling Processing

In privacy protection, the noise generated by the sampled data will lead to a certain degree of information loss. Assuming that the sensitive data comes from the definition of the user, the information loss of the sensitive data class can be evaluated by the Bayesian probability model. Previous work by Xu has proposed a probabilistic approach to predict the likelihood that private data belongs to a specific sensitive class (Xu, Jiang, Wang, Yuan & Ren, 2014). The idea is that the probability of correctly selecting and removing noisy data can affect the distribution of the entire dataset. Probabilities can include the probability of selecting the wrong data from the entire dataset. Based on this, the prior probability of selecting a real dataset from the entire collection depends on the information retrieval process. At this stage, the multinormal model $p(w|d)$ correlation terms are estimated for each sensitive data d' in set C. The data loss of the entire dataset will be indexed and searched. The probability of the loss model follows the Bayesian equation:

$$p(d|q) = p(q|d)p(d) \tag{3.50}$$

In Eq.(3.50), q is the search condition and d is the privacy features.

Through the generalization process, all data are distributed into their respective sets. In Lin's work, the generalization distance was proposed (D. Lin, 1998), and its research can calculate the similarity between two data-sets, and the similarity calculation in data sampling can directly measure the sampling results. The similarity calculation is as follows:

$$Sim(A,B) = \frac{logGM(common(A,B))}{logGM(description(A,B))} \tag{3.51}$$

In Eq.(3.51), GM is the data set with classification precision, and RD is the dataset with the related classification precision for GM. This distance has been used in (Sweeney, 2002) to calculate similar accuracy rates between the GM and RD.

Let $RD(A_1, \ldots, A_{N_a})$ be resource data, $t_{P_j} \in RD$. $\frac{m}{GM(A_1,\ldots,A_{N_a})}$ is processed by a privacy-preserving method. After inputting the data into the model for training or inference, in $t_{P_j} \in PT$, GM_A represents the generalization process of the data A. The information loss rate $Loss(L)$ during the generalization process can be calculated as:

$$Loss(L, L') = \frac{|N_A||RD| - \sum_{i=1}^{N_A} \sum_{j=1}^{N} \frac{m}{|GM_{A_i}|}}{|RD||N_A|} \tag{3.52}$$

If $L = L'$, the loss of data is distributed GM_A level into the minimum level, it has $m = |GM_{A_i}||RD|$ so the height of the general level is zero, which means $Loss(L, L') = (1 - \frac{1}{N_A}) \approx 1$. On the opposite, if the level of GM_A reaches the highest level, then the tuple $|GM_{A_i}|$ has processed and pushed the data set m to the higher level of the layer. The accuracy of the entire published data contains the lowest loss, which approaches 0.001.

The following loss equation has been triggered by the related work from Chapter 2.4.6. It shows the relationship between path cost with the tree path(the equation represents DL_{CO}) from the spinning tree. The path length can describe the cost and the information loss for generalization can describe the information loss during the generalized process, the equation is:

$$DL_{CO}(x, x') = \begin{cases} \frac{(x_{\max} - x_{\min})^2}{\max\{A\} - \min\{A\}} & \max\{A\} + \min\{A\} \\ 0 & \max\{A\} - \min\{A\} \end{cases} \tag{3.53}$$

In Eq.(3.53), $A[x']$ represents the attributes of $A[x]$ after generalized processing. $A[x']_{max}$ and $A[x']_{min}$ are the maximum and minimum values of $A[x]$ respectively. Finally, $max\{A\}$ and $min\{A\}$ represent the maximum and minimum values of A.

Eq.(3.53) can not only measure the amount of information lost in the generalization process but also quantify the probability of data being statistically attacked if the privacy attacker destroys the generalization process. The conditions of these calculations include data that change in the generalized process. For example, using a spanning tree to define a school-based hierarchy, the process of generalization can be described by generalizing schools. If we define primary school as the root, the root node can generalize secondary schools as leaf nodes. Therefore, the school type has a generalized height of 2 because students typically complete elementary school before starting secondary school. If the tree now includes a university, its height in the spanning tree becomes 3. It is observed that there is information loss in the generalization process from middle school to university, and the information loss rate is 2/3. However, if the primary school has no generalized results (no university and secondary school), then the word has a height of 1 (only primary exists). If the third layer (leaf-leaf-node is the university) is removed, the loss rate becomes $\frac{1}{3}$. However, the term "university" produces no real information loss in this example, since the university does not make any generalizations. This example is a good example of the information loss generated during the generalization process, and can also help to understand a series of problems and solutions to problems generated in data sampling.

Measurements based on generated privacy budget quantities can help models correct and normalize errors during processing, and this proposed work follows that of Chapter 3.5.2. This work also links to the estimation of the privacy budget in the sample. It is given by:

$$DL_{CO}(A,B) = \begin{cases} \dfrac{h_{A+B} - h_B}{h_B - h_A} & h_{A+B} \neq 0 \\ 0 & h_A = 0 \end{cases}. \tag{3.54}$$

In Eq.(3.54), for two attributes A and B, a spanning tree can be established by establishing the connection matrix of the two attributes. h_{A+B} represents, and the height

of the tree formed by the generalization of attribute A is h_A, and so on, the height of B is h_B. Because the generalization is compared according to the attribute A, if the height of the generalization tree of attributes A is 0, the two attributes are irrelevant, so $DL_{CO}(A, B) = 0$, and for the two associated attributes, the equation for the data source is as follows:

$$\begin{aligned} DL(D, D') = & \sum_{\forall A_{CO}} \sum_{i=k}^{n} t_{A_{CO}} \bullet DL(A_{CO}[i], A_{CO}[i]') \\ & + \sum_{\forall A_{CO}} \sum_{i=1}^{n} w_{A_{CO}} \bullet DL(A_{CO}[i], A'_{CO}[i]) \end{aligned} \tag{3.55}$$

In Eq.(3.55) where A_{CO} is a numeric attribute of CO, A'_{CO} is a categorical attribute, n is an integer index of the data record, $t_{A_{co}}$ and $t_{A_{ca}}$ are the weights of A_{CO} and A'_{CO} respectively. These weights must satisfy the constraint:

$$\sum_{\forall A_{CO}} t_{A'_{CO}} + \sum_{\forall A_{CO}} t_{A_{CO}} = 1 \tag{3.56}$$

The Eq.(3.56) explains the quantitative relationship between $t_{A_{CO}}$ and $t_{A'_{CO}}$, the $t_{A'_{CO}}$ can be obtained through this equation thus brought into the calculation.

In this quantitative calculation of Eq.(3.55) and Eq.(3.56), the relationship between the privacy noise data and the original data is quantified by spanning the tree. In the generalization process, some features of the sensitive data and the original data can be extracted. For effective extraction, the next section will introduce the top-k deviation method to effectively improve the extraction efficiency.

3.5.4 Top-k Deviation for Feature Extraction in Privacy Cost Measurement

The loss in the spinning tree during generalization and the cost of generalization can be measured as described previously. However, the loss in the disassociated dataset

has not been taken into account. For the situation where information is lost in the generalization process, my work has checked the amount of information loss by mining and searching the data that needs to be processed. Here are three metrics that can help the proposed solution compare and use privacy-preserving methods based on different data transformations, generalization, and suppression. From the perspective of data characteristics, by evaluating the statistical frequency of the original data in the data that needs to be trained and disclosed, the relative error of the information loss after the generalization of the existing processed data can be evaluated.

Top-k deviation (Terrovitis, Liagouris, Mamoulis & Skiadopoulos, 2012) (so-called top_k) from Terrovitis's work, which can calculate the most k frequent items in the original data set. Our proposed solution applies and improves it to calculate the independence degree for sampling the privacy noise from the training data in a machine learning model that belongs to a sensitive class. Assume that T and T' as the most k frequent data sets in the original and the data sets belong to the sensitive class, respectively, the independence degree can measure the relationship between the frequent data sets in non-sensitive and sensitive class$T \cap T'$. The top_k is defined as,

$$top_k = 1 - \frac{|T \cap T'|}{T} \tag{3.57}$$

In Eq.(3.57), top_k is the top k frequent sets in the most k frequent time from the data belongs to the sensitive class, and $T \cap T'$ means the data sets both appear in non-sensitive and sensitive class, and rest of data sets $T - |T \cap T'|$ are only appear in one side.

Assume the separation and generalization-based methods as two features to assess, and define ts_2 as the degree of top-k loss for calculation, which is based on the s_2 parameters. Searching the top-k elements from index of the data sets at multiple degrees of normalization is an established technique (Han & Fu, 1995), which allows the detection of privacy association rules and privacy degree of datasets that may not

appear in the frequency degree of data. In my proposed work, if there is a normalization degree that decides the privacy boundary with different sensitive classes, then the data sets in those sensitive classes can be calculated for privacy noise, it can be assumed that the same degree can be used to calculate the frequency of data sets that appear in the processed and the original data at different privacy degree of abstraction. In this case, L and L' in Eq.(3.57) are sets of normalized privacy datasets, which can be traced back to the original and sensitive class data. If the privacy-normalized dataset has been generated in a higher-level generalization process during processing, this information will be lost according to the characteristics of the privacy-preserving method. Although the reconstructed data do not contain any normalized sets, if a normalized structure is given, then in theory, the information loss can be obtained by mining the frequent data sets.

A relative error indicator c_{top} has been used to measure the frequency of privacy degree with combinations in processed data (R. Chen, Mohammed, Fung, Desai & Xiong, 2011). A modified version of this indicator is used in my work. It is given by

$$c_{top} = \frac{F(ts_1(x,y), ts_2(x,y))}{\Omega(ts_1(x,y), ts_2(x,y))} \tag{3.58}$$

In Eq.(3.58), the $ts_1(x,y)$ and $ts_2(x,y)$ are privacy levels for the combination of the terms x and y in the original data and privacy noise data, respectively. Reconstructing a privacy noise data set may introduce a new portfolio of items into the published data, which does not exist in the original data. The $F(ts_1(x,y), ts_2(x,y))$ means the frequency for x and y, and $\Omega(ts_1(x,y), ts_2(x,y))$ is the parameter which can convert the frequency for dataset to privacy degree. The average value in the denominator (instead of using the original $ts_o(x,y)$) accounts for this situation.

The association of noisy sampling with building a privacy level via spanning trees

is an important component of our Modified Dynamic Conditional Random Field (MDCRF). After introducing the important components, the MDCRF approach as a whole is described and illustrated in the next section.

3.6 The Motivation and Detail of Modified Dynamic Conditional Random Field

This chapter begins with a fundamental question: How can data privacy be preserved in machine learning models by generating the necessary privacy noise and effectively tracking or controlling its addition? In addressing this question, the previous research in this chapter has made progress in addressing the generation of privacy noise. Another crucial aspect of this problem is to establish the process of tracking and controlling the addition of privacy noise to the training and prediction data. These two sections provide a detailed description of the tracking and control methods proposed in this paper, which aim to establish the relationship between an object's sensitive class labels and its sensitive features in order to solve this problem. Subsequently, privacy noise is injected into the sensitive features to protect them from the potential leakage of sensitive information.

In 2005, Dynamic Conditional Random Fields (DCRF) were proposed for efficient object segmentation and related feature tracking of image sequences (Y. Wang & Ji, 2005; Y. Wang, 2009). In (Wojek & Schiele, 2008), the DCRF model is used for joint labeling of objects and scenes. Although the DCRF model does not involve any privacy image protection, the joint labels proposed by its work can help solve the core work of privacy protection and machine learning tasks, namely the association of sensitive features. The success of these models has inspired our work combining privacy preservation and machine learning, i.e. constructing dependencies between

sensitive object class labels and probability distributions of sensitive objects.

Assuming that the data obtained at the time point t is the time observation data, the conditional random field method can be used to classify and segment the data to be segmented through the posterior probability. At the same time, these segmented labels a and b both follow the Markov chain:

$$M(p_t(a)|i_{1,\ldots,T}, p_t(b), a \neq b) = M(p_t(a)|i_{1,\ldots,T}, p_t(b), b \in F_a) \tag{3.59}$$

In Eq.(3.59), $M(p_t(a)|i_{1,\ldots,T}, p_t(b), a \neq b)$ represents at time t, the probability distribution of the variable a with the condition of the variables $i_{1,\ldots,T}$ (which could be a sequence of inputs or observations), the probability distribution $p_t(b)$ of variable b at time t, and the condition that a and b are not equal, and $M(p_t(a)|i_{1,\ldots,T}, p_t(b), b \in F_a)$ has the same definition of variables, the probability distribution $p_t(b)$ of variable b at time t, and the condition that b belongs to the set F_a. This equation essentially states that given the probability distribution of variable a and b at time t as input, the condition that a and b are not equal, is equal to the probability distribution of a at time t, given the same observations, the probability distribution of b at time t, and the condition that b belongs to the set F_a. Therefore, MDCRF is a random number with the observation data as the global condition. Through the Hammersley-Clifford theorem mentioned in the paper (Chandgotia, 2017), the Gibbs distribution can be used to determine the probability value of the conditionally segmented data. The detail as follow:

$$\begin{aligned} M(p_t(a)|i_{1,\ldots,T}) \propto \exp\{-\sum_{a \in A}[R_x(p_t(a)|i_{1,\ldots,T}) \\ + \sum_{b \in F_a} R_{a,b}(p_t(a), p_t(b))\} \end{aligned} \tag{3.60}$$

In Eq.(3.60), $M(p_t(a)|i_{1,\ldots,T}, p_t(b), a \neq b)$ represents the conditional probability of variable a at time t given the observations $i_{1,\ldots,T}$, the probability of variable b at time

t, and the condition that a and b are not equal. And $M(p_t(a)|i_{1,\ldots,T}, p_t(b), b \in F_a)$ represents the conditional probability of variable a at time t given the observations $i_{1,\ldots,T}$, the probability of variable b at time t, and the condition that b belongs to the set F_a. The set F_a contains all the possible values of b that are related to variable a in some way. In order to reduce the computation complexity, it is assumed that the pairwise constraints are independent of the observed prediction data. The following dual-domain potential imposes these spatial connectivity constraints:

$$R_{a,b}(p_t(a), p_t(b)) \propto S_{a,b}\left(p_t(x), p_t(y)\right) \tag{3.61}$$

In Eq.(3.61), the single-domain potential $R_a(s_T(X)|I_{1,\ldots,t})$ reflects the observation information of a single site from $t = 1$ to $t = T$, while the dual-domain potential $R_{a,b}(p_t(a), p_t(b))$ form a continuous observation area by applying spatial constraints to it.

$$R_{a,b}(p_t(a), p_t(b)) = \frac{1}{\| a - b \|^2}\left(1 - \sigma(p_t(a) - p_t(b))\right) \tag{3.62}$$

In Eq.(3.62), the $\| a - b \|^2$ means the Euclidean distance and $\sigma(p_t(a) - p_t(b))$ is the Kronecker delta function (Koo, Yoon & Cho, 2013). Thus, two neighborhoods are more likely to belong to the same class.

To represent the time dependence of continuous fractional fields, the state transition probabilities of fractional fields are also fitted by Gaussian and Gibbs distributions for single-domain and double-domain data sets.

$$\begin{aligned} M(p_{t+1}|p_t) \propto \exp\{-\sum_{a\in A}[R_a(p_{t+1}(a)|p_t(G_a)) \\ + \sum_{b\in F_a} R_{a,b}(p_{t+1}(a), p_{t+1}(b))\} \end{aligned} \tag{3.63}$$

In Eq.(3.63), $M(p_{t+1}|p_t)$ represents the conditional probability distribution of the variable p_{t+1} at time $t+1$ given the variable p_t at time t. It denotes the probability of the next state given the current state. And. $\exp\ldots$ is the exponential function, where the argument inside the curly braces represents the exponent. It is used to transform the sum of the terms into a positive value. $\sum_{a\in A}$ denotes the summation over all values of variable a in the set A. It represents a sum of terms over all elements in the set A. Then, $R_a(p_{t+1}(a)|p_t(G_a))$ represents a potential function or a compatibility score associated with variable a at time $t+1$ given the variable $p_t(G_a)$, which represents the relevant features or context associated with variable a at time t. It captures the relationship between the current and next state of variable a. $\sum_{b\in F_a} R_{a,b}(p_{t+1}(a), p_{t+1}(b))$ represents a sum over all values of variable b in the set F_a, which represents the set of variables that are related to variable a. It represents the pairwise potential or interaction score between variable a and its related variables at time $t+1$.

Furthermore, $R_a(p_{t+1}(a)|p_t(G_a))$ models the label state transition at a single position. And the adjacent sites at t have an impact on all adjacent domains at $(t+1)$, and G_a represents the group of domains. For a pair of data sets $R_{a,b}(p_{t+1}(a), p_{t+1}(b))$, Eq.(3.63) provides the spatial constraints. F_a is called the spatial neighbourhood, and G_a is the temporal neighbourhood. The equation represents the probability distribution over the next state p_{t+1} given the current state p_t in Modified Dynamic Conditional Random Field (MDCRF). The potential functions R_a and $R_{a,b}$ define the dependencies and relationships between variables, and the exponential term ensures that the probability distribution is normalized.

Modified Dynamic Conditional Random Field (MDCRF) models can now be derived by integrating both temporal and space dependencies into the segmentation process into a single dynamic stochastic model. The following equations govern this MDCRF

model:

$$\begin{aligned} R_a(p_{t+1}(a)|p_t(G_a)) &= \frac{1}{|F_a|}\sum_{b\in F_a} R_a(p_{t+1}(a), p_t(b)) \\ &\propto S_a(p_{t+1}(a)|p_t(b)) \\ S_a(p_{t+1}(a)|p_t(b)) &= \frac{1}{\parallel a-b \parallel^2 +\rho^2}(1-\sigma(p_{t+1}(a)-p_t(b))) \end{aligned} \tag{3.64}$$

In Eq.(3.64), $R_a(p_{t+1}(a)|p_t(G_a))$ represents a potential function or compatibility score associated with variable a at time $t+1$ given the context or features $p_t(G_a)$. It captures the relationship between the current state $p_t(G_a)$ and the next state $p_{t+1}(a)$ of variable a. The potential function R_a is defined based on the interaction between a and other related variables. And F_a represents the set of variables that are related to variable a. The summation is performed over all variables b in this set. The potential function R_a is averaged over these related variables to obtain $R_a(p_{t+1}(a)|p_t(G_a))$. $S_a(p_{t+1}(a)|p_t(b))$ represents a scaled potential function or compatibility score between variable a at time $t+1$ and variable b at time t. It measures the similarity or affinity between the states $p_{t+1}(a)$ and $p_t(b)$. $\parallel a-b \parallel^2$ represents the squared Euclidean distance between variables a and b. It measures the dissimilarity between the two variables. ρ^2 represents a parameter that adds a smoothing factor to the distance. It ensures that the denominator is not zero, preventing division by zero errors and providing stability to the computation. $\sigma(p_{t+1}(a)-p_t(b))$ represents the Sigmoid function, which maps the difference between $p_{t+1}(a)$ and $p_t(b)$ to a value between 0 and 1. It quantifies the dissimilarity between the states $p_{t+1}(a)$ and $p_t(b)$.

In addition, $R_a[p_{t+1}(a)|p_t(G_a)]$ in the potential prediction equation is used to label its temporal neighborhood through time continuity constraints to match the label and $S_a[p_{t+1}(a)|p_t(b)]$ is the method to calculate the sensitive features. Generally, if the input data changes slowly, the value of ρ is small. Overall, Eq.(3.64) defines the potential functions or compatibility scores between variables in MDCRF framework.

The potential functions capture the relationships and interactions between variables, and the Sigmoid function and distance terms quantify the dissimilarity and similarity between variable states. Assume that the spatial observations are independent and identically distributed, the observation model $q(l_t|p_t)$ can be simplified to

$$q(l_t|p_t) = \prod_{a \in A} q\left(l_t(a)|p_t(a)\right) \tag{3.65}$$

In Eq.(3.65), $q(l_t|p_t)$ represents the conditional probability distribution of the label variables l_t at time t given the input or observed variables p_t. It denotes the probability of the label variables given the input variables. $\prod_{a \in A}$ denotes the product over all values of variable a in the set A. It represents the product of terms over all elements in the set A. $q\left(l_t(a)|p_t(a)\right)$ represents the conditional probability distribution of the label variable $l_t(a)$ for a specific variable a at time t given the corresponding input variable $p_t(a)$. It captures the probability of a specific label for a specific variable given its corresponding input value. And $l_t(a)$ is the observation value of node a in the chain, and $p_t(a)$ is the probability in time t of value a, which consists of locally measured information such as its data input intensity or gradient characteristics.

In this section, the modified dynamic conditional random field(MDCRF) method is used to observe and track the features that require privacy protection globally, and all privacy-related features can be associated. But on the other hand, using this tracking method, if one privacy feature is exposed, other privacy features will also be obtained by privacy attackers. The next section will attempt to address this issue.

3.7 The Description of Privacy-Preserving Modified Dynamic Conditional Random Field Algorithm (MDCRF)

The traditional dynamic conditional random field (Y. Wang & Ji, 2005) method does not take privacy protection into consideration, and cannot associate privacy features. Using this method to associate privacy features will cause a series of privacy feature exposure problems. In order to preserve the privacy features while associating them, this thesis attempts to combine the privacy noise method of Chapter 3.4 on the basis of the MDCRF method.

The generation of privacy noise is described in Chapter 3.4. In the MDCRF algorithm, the method in this section is used to generate privacy noise, and the conditional random field method is used and improved. At the same time, the privacy noise proposed in this thesis can be used in machine learning tasks.

3.7.1 Description of MDCRF Algorithm and Server Process for MDCRF

Specifically in Algorithm 3.1, in order to detect the objects inside of images and protect the privacy(sensitive) object at the same time, the process will include four steps, which are described in Algorithm 3.1. In the input step in Algorithm 3.1, the data Θ the sensitive class $D_1, D_2, \ldots D_N$ are as input, and loss function is $L(\Theta) = \frac{1}{N}\Sigma_i \mathcal{L}(\theta, d_n)$, learning rate as η, and noise amount τ and other parameters are specified as input data for processing. Next, MDCRF generates the discernibility matrix(DM) based on the input training set Θ and sensitive categories. According to the modified dynamic conditional random field scheme described in Chapter 3.6, which uses the discernibility

matrix (DM) mentioned in Chapter 3.5, this method will be used to update its adjacency matrix x. In the third step, samples are collected according to the input privacy budget to generate privacy noise. This step uses the method described in Chapter 3.4. The fourth step algorithm will reconstruct Θ and update the corresponding DM. It is then fed into an object detection framework to train or predict image data Θ to preserve privacy. The MDCRF scheme helps the existing target detection framework to complete model training or inference by inputting privacy-related parameters and image data, ensuring that privacy will not be leaked.

A third-party object detection method is added to the MDCRF algorithm. In this case, YOLO (Bochkovskiy et al., 2020) is used as the object detection method combined with MDCRF in this thesis. When MDCRF combines object detection methods to process images, it can match sensitive objects and add privacy noise to them in order to protect privacy. Specific examples are shown in Figure 3.3 and Figure 3.4. For instance, in Figure 3.3, the original YOLO object detection model can predict the classes that already exist in all models and mark each class on the graph, such as marking vehicles and providing other relevant information. In Figure 3.4, when the user identifies the vehicle in the image as a sensitive object, the MDCRF method combined with YOLO adds quantitative privacy noise to the area of the vehicle in the image. This makes it challenging for privacy attackers to match the features of these vehicles with other vehicle pictures and obtain privacy information about the vehicle. The amount of quantitative privacy noise is determined by the privacy budget ϵ input by the user. Naturally, more noise makes it harder to match the privacy. However, if the noised pictures are used as training data, it becomes difficult for the model to extract features and predict other pictures based on the features of these pictures. Consequently, this reduces the prediction accuracy of the model.

The MDCRF algorithm can be implemented using federated learning. Algorithm 3.1 is the client's steps. The details of the process in the server will be described in

Algorithm 3.1 Privacy-Preserving Modified Dynamic Conditional Random Field(MDCRF) Algorithm

Input: Images data Θ, sensitive objects feature classes $D_1, D_2, \ldots D_N$ and class size N, adjacent matrix x, loss function $L(\Theta) = \frac{1}{N}\Sigma_i \mathcal{L}(\theta, d_n)$, learning rate η, noise budget ϵ, noise scale τ, gradient norm bound G and object detect model $Det(M)$ and federate learning opinion fl.

Output: Privacy cost (ρ, ω) and images data Θ'

Procedure: Sensitive labels D_n and adjust parameter group $P_{DCRF}(\eta, \tau, S, G)$ and discernibility matrix DM
for $n \in [\mathcal{N}]$ **do**
    Input parameter group $P_{Raw}(\eta, \tau, G)$ into dynamic conditional random field P_{DCRF}
    Estimated Parameters with dynamic conditional random field and restructure $P_{MDCRF}(\eta, \tau, S, G)$, generate DM and $P_n(\eta, \tau, G)$
    Fetch sensitive feature label D_n and the probability is $\frac{D}{N}$
    Input parameters η, τ, G into object detection model $Det(M)$
    for each $x \in L_t$ compute $g_t(d_x)$ then it has $\vartheta_{\Theta_t}\mathcal{L}(\Theta_t, d_n) \rightarrow L_t$ **do**
        $\dot{g}_t(d_i) \leftarrow \frac{g_t(d_i)}{max(1, \frac{\|g_t(d_i)\|_2}{G})}$
        Generated Gaussian noise from ϵ and τ.
        $\dot{g}_t \leftarrow \frac{1}{N}(\Sigma_i g_t(d_i) + \mathcal{N}(0, \eta^2 G^2 I))$
        $\Theta_{i+1} \leftarrow \Theta_i - \eta_i \dot{g}_i$
    end for
end for
Input the object detection framework, and retains the newly generated privacy budget ϑ_{Θ_i} as the new privacy budget, and then chooses to run one of the set of rules G.
Choose a gradient sample D_{n+1} that is bounded and below the range G in object detection framework.
Transfer the data of the output Θ_{i+1} to the sequence $\dot{g}_i(d_i)$ for sorting and return the maximum value from the object detection framework and update DM with new parameters $P_{MDCRF}(\eta, \tau, S, G)$.
Submit the $P_{fl} = P_{MDCRF}(\eta, \tau, S, G)$ to the server if using federated learning.
End procedure

Algorithm 3.2, and the server controls all clients The learning parameters of the terminal, and the training process is put into the client in Algorithm 3.1. In this way, all privacy-related datasets are kept in their own client, and avoid uploading or sharing the collected training images, but only upload the parameters $P_{MDCRF}(\eta, \tau, S, G)$.

Algorithm 3.2 Server Process for the MDCRF Algorithm

Input: Collect parameters $P_n(\eta, \tau, S, \mathbb{G})$ from each user terminal. Parameters include: Loss function $L(\Theta)$, learning rate of parameters η, noise scale τ, data size S and gradient norm $\mathbb{G}$ bounded.
Output: Latest global parameter $P_{global}(\eta, \tau, S, \mathbb{G})$

1: **Procedure** Fetch each individual terminate parameter group $P_n(\eta, \tau, S, \mathbb{G})$ from the user terminate.
2: **for** $n \in [\mathcal{N}]$ **do**
3: Set parameter group $\acute{P}_{global} = P_{global} + P_n$
4: Set $C_n := C_n + 1$
5: **end for**
6: Send $\acute{P}_{global}(\acute{\eta}, \acute{\tau}, \acute{S}, \acute{\mathbb{G}})$ back to the user terminate.
7: Let T_n be the set of index for counter C_n to select maximum for sorted list C_n.
8: **End procedure**

The federated training process for the MDCRF algorithm in this thesis involves the following three steps:

- The client downloads the training parameters from the server.
- At each stage of stochastic gradient descent training, the local client computes the gradient and the information loss $\mathcal{L}(\Theta_t, d_n)$ of a random subset of training samples ϑ_{Θ_t}. This information loss is used to calculate the average gradient descent. Finally, the privacy of the training dataset is protected by adding noise through the noise assignment method described in Chapter 3.4, and measures are implemented to address privacy-related noise sequences and improve the accuracy of model training. Furthermore, the private output is based on the data stored internally within the data-protected machine learning model, which does not fully converge its SGD descent gradient until the end of training.
- The final step involves uploading the data to the server based on two criteria. The first criterion is the maximum value of the stochastic gradient origin, while the second criterion requires the sample to be smaller than a specified threshold.

Failure to meet these two criteria will slow down the convergence of training. In addition, both criteria can be evaluated by measuring the privacy noise and privacy budget using a kernel function within an abstract Wiener space.

The server initializes the parameters and sends them back to each client when requested. The algorithm for the server is described in Algorithm 3.2.

In summary, a privacy-preserving modified dynamic conditional random field(MDCRF) algorithm combined with the federated learning framework is proposed in this chapter. In this framework, the server can control the process of SGD but it cannot read or fetch any data, and all data are saved and processed on local machines. Adding Gaussian noise in the gradient descent iteration will not disrupt the original training process, and privacy attackers will not detect the output data during the training of the machine learning model, which can protect the privacy of the data. The next section will provide the related experiments to support the theory work or algorithm from the previous sections in this chapter.

3.8 Experimental Evaluations

In the experiments in this chapter, the privacy-preserving pre-processing method before the machine learning training phase will be involved. Some of the experimental methods used come from the methods of verifying privacy attacks mentioned in the threat model in Chapter 3.0.2, including the comparison of K-L divergence between original data and processed data, and the accuracy index of model prediction, privacy errors caused by noise. The methods and design principles used in the specific experimental design are described in detail in Chapter 2.6. For the MDCRF method that needs to be tested, Chapter 3.6, elaborates in detail on how to mitigate privacy mining used by privacy attackers through objects detected in images and videos. In this study, the statistics of several types of noise in target detection tasks will be analyzed.

Figure 3.3: Outdoor images: The original image (top), the image shows the detected objects (bottom)

3.8.1 Experimental Design Principles for Object Detection Test

In terms of experimental design for privacy protection, two indicators need to be considered. The first indicator is the parameters of object detection itself, including the type and quantity of objects, and the scale detection of objects. The second indicator is the impact of privacy noise or other factors on the accuracy interference problem.

For the problem of object type and quantity, when performing object detection, the scheme needs to detect multiple different types of objects in one image, which requires that if the data set includes multiple types of objects for machine vision tasks, It contains objects of privacy-sensitive classes. For the second point, the problem of object scale, this requires that in the image data set, under the same picture, it is necessary to restore

Figure 3.4: Outdoor images before hiding the sensitive features (top) and after detecting the visual objects (bottom)

the real situation as much as possible, that is, there are cases where the object size is very large, and there are also cases where the object size is very small, and if the scale of the object is in the sequence image, the greater the change, it means that when detecting this type of object, it is more difficult to accurately identify the object. Especially in an extreme case, that is, when the object scale (that is, the object size) is very small, and its number appears in the picture, the difficulty of this type of detection also increases accordingly. Finally, the interference caused by privacy noise and other factors is also very important, because privacy protection is based on adding noise to sensitive areas to interfere with privacy attacks. To a certain extent, it may also affect the detection of other objects, such as The case where the privacy object area and the non-privacy object

area overlap. In addition, other factors will also interfere with the accuracy of detection, including lighting, occlusion of objects or overlapping areas, and pixel quality of the image itself. This requires us to take into account relevant factors in the experimental design, so as to select the data set properly.

3.8.2 Benchmark Datasets for Object Detection Test

Among the related methods proposed in this chapter, the main research purpose is the generation of privacy noise and its application in machine learning models. The MDCRF approach proposed in this chapter aims to preserve the privacy of sensitive objects while tracking features in images. Regarding conservation and tracking, the following datasets are benchmark datasets: MOT (Lealtaixé et al., 2015), Cityscape (Cordts et al., 2015) and KITTI (Geiger et al., 2013). These three public datasets contain image sequences extracted from videos, including street scenes, pedestrians, and other features. Street names, signboards, landmarks, and pedestrian faces can all be used as sources of sensitive objects. In the above-mentioned several benchmark datasets, accurate manual labeling and annotation data are provided to provide basic facts. So for all experiments using this data set, the ground truth refers to 100% accuracy.

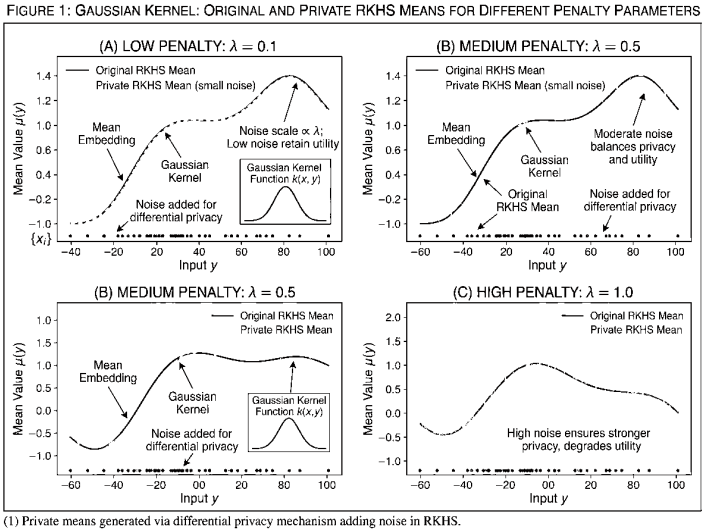
An example of the images is shown in Figure 3.3. The quality of these images is similar to that obtained using typical surveillance cameras. The top image is the original image and the bottom one is the same image with some objects (vehicles and pedestrians) that have been identified using an object detection system. Both images have a resolution of 1280×720 pixels, the equivalent of 720p. Private information that allows certain vehicles and their location to be identified includes the car license numbers, the names of the shops along the street, landmarks, logos, and faces of the pedestrians. Figure 3.4 shows the same image with all the sensitive areas covered by mosaics, which are highlighted by the green bounding boxes. It is interesting to note

that the object detection algorithm failed to detect the person on the left-hand side of the image since most of its features are destroyed by the feature hiding process when it could be detected as shown in the lower image in Figure 3.3.

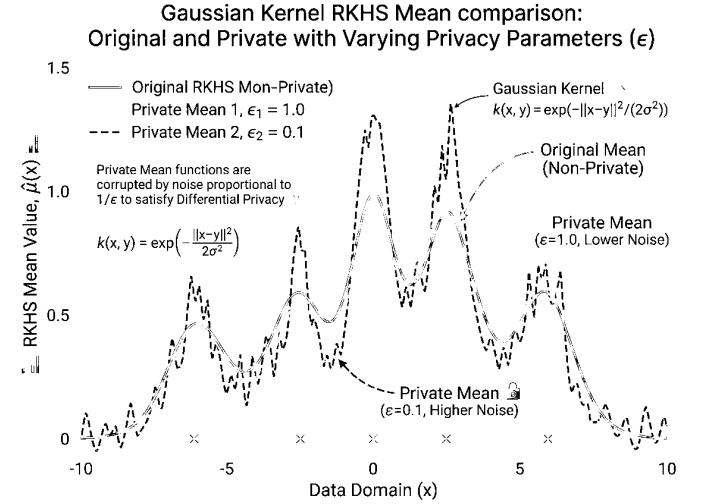
3.8.3 Ablation Study for Privacy-Preserving Object Detection Model Tests

In subsequent experiments, k-fold cross-validation was used for training and division of the data set. The data set was divided into a training set and a test set using cross-validation. The test set was set aside, and the training set was further divided into k parts. One of the k subsets was used as a validation set, while the remaining subsets were used for training. This process was repeated k times, resulting in different models. Unless otherwise specified in subsequent experiments, k = 10. The final selected model was evaluated based on the performance of the k models. The hyper-parameters that yielded the best results were chosen as the optimal hyper-parameters. Subsequently, the final model was trained using all k subsets of data as the training set.

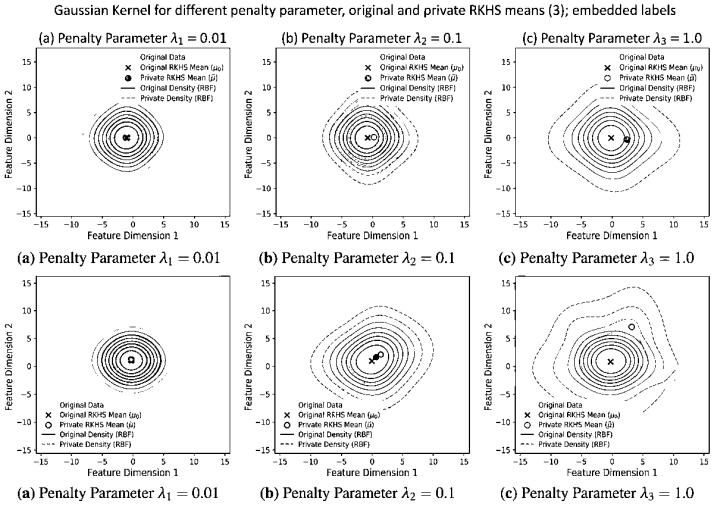
The curves in Figure 3.5 demonstrate the incorporation of privacy noise into the Wiener space of the Gaussian kernel for calculation, and they are compared using a χ^2 test. The experimental setup is based on the principle of experimental design explained in Chapter 2.6.8, which elaborates on the χ^2 test principle. In the experiments conducted in this section, the χ^2 test is employed as a statistical method to detect the generated noise under different privacy scenarios within the allocated budget. The privacy budget represents the privacy noise level of the proposed method, the details of which are explained in Chapter 3.4. The benchmark datasets used to test the privacy noise have been selected from the MOT17 datasets (Lealtaixé et al., 2015). This experiment considers a chi-square statistic of 0 as the ground truth because a chi-square statistic of 0 indicates that the observed frequency matches exactly with the expected frequency,



(1) Private means generated via differential privacy mechanism adding noise in RKHS.



EFFECT OF PRIVACY ON GAUSSIAN KERNEL RKHS MEANS ACROSS DIFFERENT PENALTY PARAMETERS



(a) Penalty Parameter $\lambda_1 = 0.01$ (b) Penalty Parameter $\lambda_2 = 0.1$ (c) Penalty Parameter $\lambda_3 = 1.0$

(a) Penalty Parameter $\lambda_1 = 0.01$ (b) Penalty Parameter $\lambda_2 = 0.1$ (c) Penalty Parameter $\lambda_3 = 1.0$

Figure 3.5: Comparison of χ^2 values between using the original Wiener space means and private means in the Gaussian sampling with different ϵ

implying no difference.

As the penalty or privacy leakage score is given by Eq.(3.23) increases, the difference between the private mean and the original mean approaches zero. In other words, the noise data generated by $K(x,\cdot) = \Sigma_{i=1}^{\infty}\lambda_i\psi_i(x)\varphi_i$ is an approximation to the original data. In terms of utility, the amount of noise decreases as ρ increases.

In this thesis, the privacy-preserving parameter is utilized. The added privacy noise data will affect the error of the machine learning result, so the scheme needs to adjust the penalty parameter according to the second and third stages of the combined machine learning process to help reduce the error. Finally, χ^2 test checks whether the noise fits within the privacy mean.

For experimental setup, the design of experiments for Figure 3.6 and Figure 3.7 are from Chapter 2.6.2 and Chapter 2.6.3. In this experiment, the MDCRF model is combined with several indicators including the prediction accuracy and GIOU with the most advanced baseline method Faster R-CNN (Ren et al., 2015) and YOLO (Bochkovskiy et al., 2020) to combine with our federated learning mentioned in Chapter 3.7.1, so as to achieve a meaningful assessment of its effectiveness and superiority. And this test applied MOT17 (Dendorfer et al., 2020) as a benchmark dataset. In Figure 3.6 and Figure 3.7, the x-axes are the time and epochs, the y-axes are the GIoU, objectness, classifiction, precision and recall separately.

For comparative analysis, the Faster R-CNN (Ren et al., 2015) and YOLO (Bochkovskiy et al., 2020) methods are used for object detection in these experiments with privacy noise added to the sensitive objects of the test image dataset, the privacy noise generation work has mentioned in Chapter 3.4. Since Faster R-CNN is a two-stage detector, the experiment epochs are doubled compared to those conducted using YOLO. Five different performance measures are used for comparison. These include six parameters that indicate the performance of object detection tasks: the predicted probability of objectiveness, the Generalized Intersection over Union (GIoU) (Rezatofighi et al., 2019), the

error rate of classification, mean average precision (mAP), and recall of mean average precisions (mAP) under the IoU, those indicators are mentioned in Chapter 2.6.2. The training results with and without using MDCRF are shown in Figure 3.6 and Figure 3.7, respectively. Because the benchmark datasets uses artificially labeled data, the ground truth for this experiment is 0 for the three experimental metrics of GIoU, objectness, and Classification error. This indicates that there is no error in these three metrics. For the two metrics of mAP@0.5 and Recall mAP@0.5:0.95, the ground truth is 100%, indicating a prediction accuracy of 100%. The default parameters for those two tests in Figure 3.6 and Figure 3.7 are 0.5 with IOU in precision tests and 0.5 and 0.9 for recall tests separately.

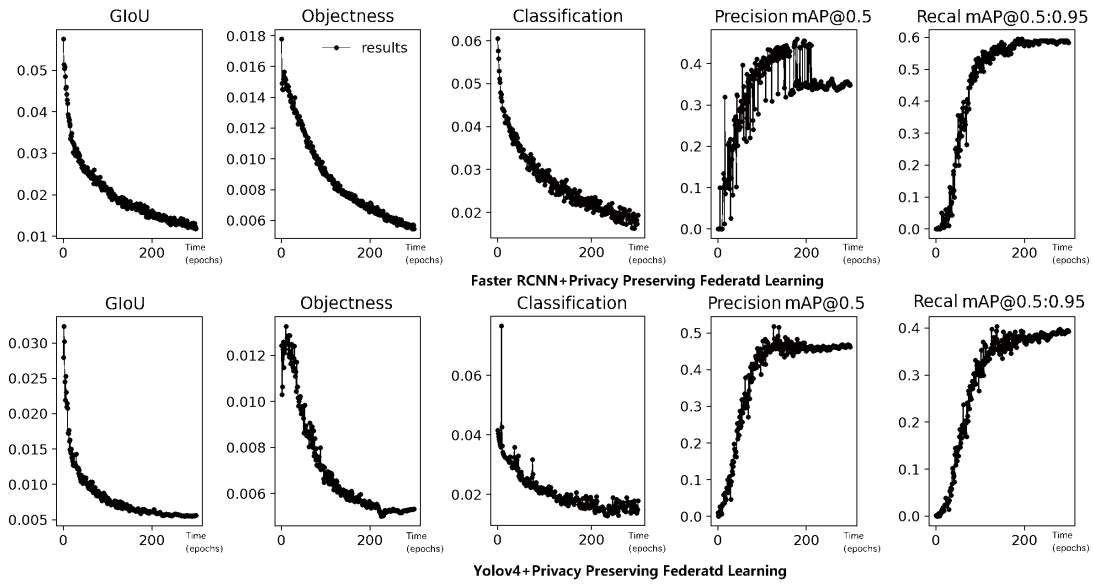


Figure 3.6: Visual object detection models without MDCRF

In Figure 3.7, the GIoU curves maintain the same downward trend as those without MDCRF, with their range increased from $0.016 - 0.022$ to $0.04 - 0.10$ during the training process. The same trend is also present in Figure 3.6, the validation GIoU graph, increasing approximately from $0.006 - 0.009$ to $0.01 - 0.07$ The processing time

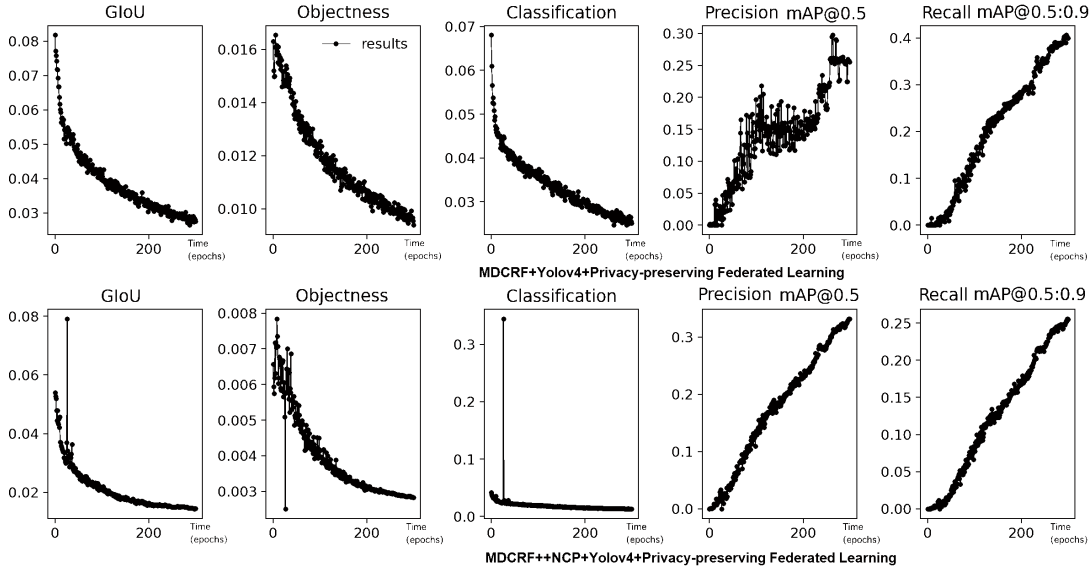


Figure 3.7: Visual object detection models with MDCRF

for objectiveness converges with the overall objectiveness has decreased from 0.016 to 0.01, and the precision of the object recognized rate increased significantly, from 0.001 to 0.3 and 0.001 to 0.37 separately. The classification performance improved when using MDCRF with the YOLOv4 detector. Notably, the predicted precision stays high as its recall increases, demonstrating better performance with the use of MDCRF. The graphs show that the mAP values present upward trends with fewer fluctuations, and the final mAP@0.5 is improved from 0.48 without MDCRF to 0.5 with MDCRF for YOLO.

3.8.4 Privacy Noise Contrast Analysis and Ablation Study For PSNR, MMD, SSIM, and FFT

For experimental setup, in this section, from Figure 3.8 to Figure 3.11 utilize the MOT17 dataset as the benchmark dataset. Since the proposed privacy-preserving method is intended for tasks involving images or videos, it measures the quality of images from

the MOT17 dataset using PSNR, MMD, SSIM, and FFT indicators. These methods calculate values by comparing the differences between the original image and its image with privacy noise. Additionally, unless otherwise specified, all tests in this section default to using a privacy budget of $\epsilon = 0.01$.At the same time, in order to compare the differences brought about by different privacy noises on these four indicators, the experiment also added DP(MDCRF)-0.001, that is, the method MDCRF in this paper also uses differential privacy and sets the privacy budget to $\epsilon = 0.001$, that is, it is hoped to verify the differences of these indicators on the same data set through a higher privacy level, so as to complete the ablation study at the same time.

Regarding the benchmark datasets, the MOT17 (Dendorfer et al., 2020) dataset is a widely recognized benchmark dataset in the field of object detection and multi-object tracking, known as MOTChallenge. MOTChallenge includes several datasets with different versions, such as MOT15, MOT16, MOT17, and MOT20. The MOT Challenge dataset is designed to evaluate the performance of multi-object tracking algorithms. It consists of video sequences captured from various real-world scenarios, including crowded public spaces, streets, and shopping malls. The dataset provides ground truth annotations that include bounding boxes around the objects of interest and unique object IDs to track individual objects across frames. Each version of the MOT dataset, including MOT17 and MOT20, introduces new challenges and improvements compared to previous versions. These challenges can involve occlusions, crowded scenes, object interactions, scale variations, and more. The dataset enables researchers to compare and evaluate different computer vision algorithms based on their accuracy, robustness, and efficiency. In these experiments, 0 means ground truth, and if an indicator is 0, it means that this indicator is exactly the same as the original data.

From Figure 3.8 to Figure 3.11, each figure uses different evaluation indicators. The design concepts of these evaluation indicators are described in Chapter 2.6.5. Specifically, in this test, each indicator serves a different purpose and those indicators

are the statistical analysis methods. The details of these four indicators are as follows:

- Firstly, the peak signal-to-noise ratio(PSNR) compares the original and reconstructed signals by calculating the ratio of the peak signal power to the mean squared error (MSE) between the original image and the image with added privacy noise. Each picture element (pixel) has a color value that can change when the image is compressed and then decompressed. The larger the PSNR value between the two images, the more similar they are in terms of pixel color values.

- Secondly, the structural similarity index (SSIM) (Wang et al., 2004) is an image quality metric that takes into account brightness, contrast, and structure to assess the similarity between the original image and the distorted image with added privacy noise. In SSIM, a decimal value between $[0, 1]$ is used to indicate the level of dissimilarity between the noisy image and the undistorted original image. Generally, the closer the SSIM value is to 1, the closer the image quality is to the original image. Conversely, as the SSIM value approaches 0, the image quality decreases, suggesting greater dissimilarity between the two images.

- Thirdly, maximum mean discrepancy (MMD) (Gretton et al., 2012) quantifies the difference between two groups of samples and is primarily used to measure the distance between two different but related distributions. The MMD value represents the dissimilarity between the distributions, with a larger value indicating a greater difference.

- Lastly, fast Fourier transform (FFT) (Oppenheim, 1999) is an algorithm for efficiently computing the discrete Fourier transforms (DFT) of signals or images. FFT is a rapid algorithm for performing DFT, enabling the transformation of a signal from the time domain to the frequency domain. Some signals are challenging to analyze in the time domain, but by converting them to the frequency

domain using FFT, their characteristics become easier to observe. By applying FFT to noisy images, it is possible to analyze their frequency components and explore the presence or characteristics of noise in the frequency domain.

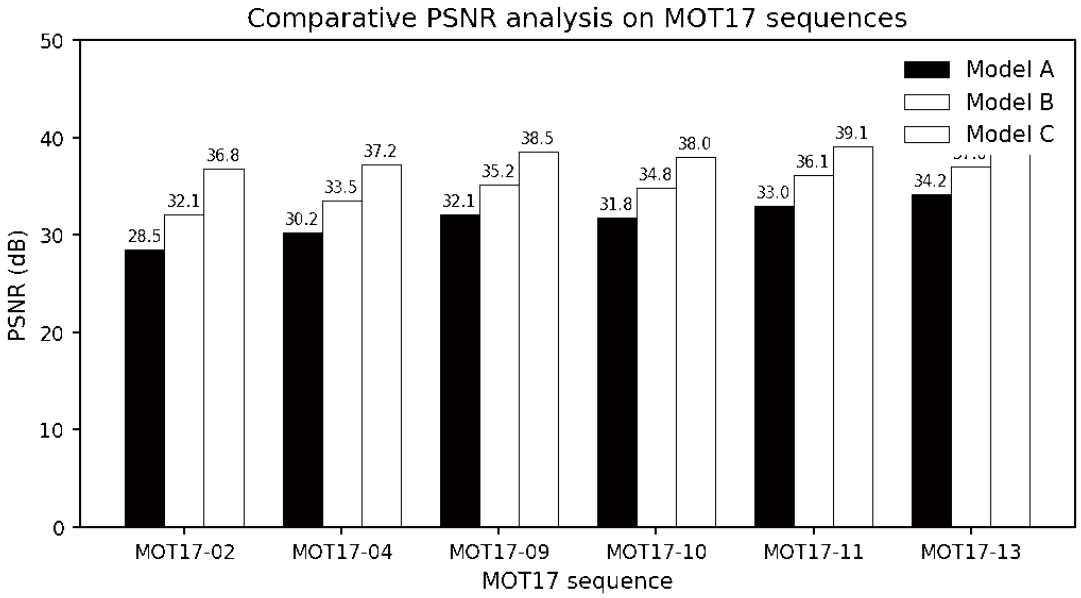
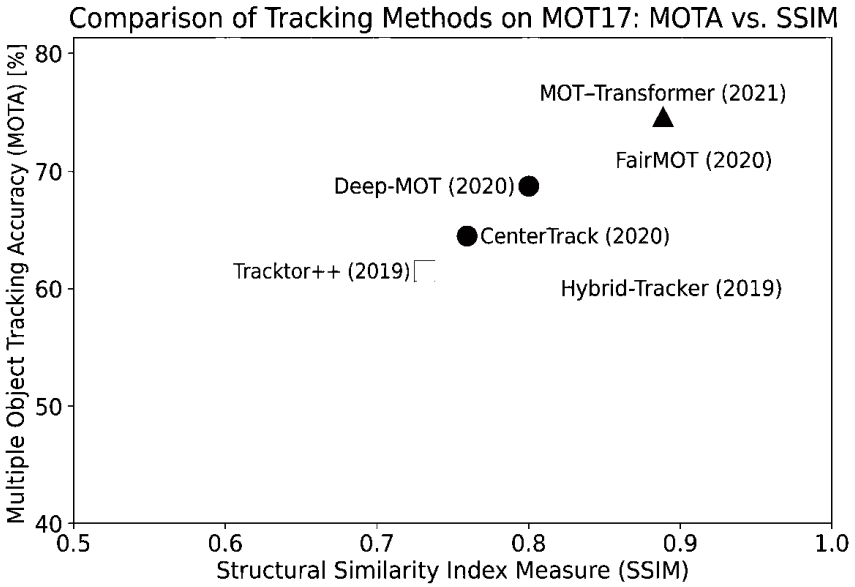


Figure 3.8: PSNR metric compares several privacy noises

For comparative analysis of each figure, Figure 3.8 displays five privacy protection methods on the x-axis, namely differential privacy by Gaussian noise (DP (Gaussian)), differential privacy by Laplace noise (DP (Laplace)), Random noise (Random), Exponential noise (exponential), and differential privacy by MDCRF (DP(MDCRF)), differential privacy by MDCRF (DP(MDCRF)) use two privacy budget to test the differences, one is $\epsilon = 0.01$, the named as DP(MDCRF-0.01), the other named DP(MDCRF-0.001) using the privacy budget$\epsilon = 0.001$. Here, the performance of MDCRF on four metrics is observed by injecting different privacy noises. The y-axis represents their Peak Signal-to-Noise Ratio Value (PSNR), while the x-axis represents the timestamp indicating different time points for the images. From Figure 3.8, it can be observed

that the lines of the five privacy protection methods coincide for the most part. As the number of runs increases, the PSNR values show a slow-increasing trend, reaching a maximum of approximately 38.7 after around 750 iterations. However, after the time stamp 1050, the PSNR values drop rapidly, reaching a minimum of about 32.7 after approximately timestamp 1100. Notably, DP (Gaussian) stands out slightly in several instances. Additionally, the DP(MDCRF)-0.01 and DP(MDCRF)-0.001 proposed in this thesis overlaps with the other lines, indicating that the difference between the privacy protection method proposed in this thesis and the other methods is minimal. MDCRF exhibits a stronger protective effect due to its good fit and is less susceptible to attacks such as filters.



MOTA vs SSIM performance for selected algorithms on the MOT17 test set. Generally, methods with higher MOTA tend to achieve higher SSIM scores, indicating a correlation between tracking quality and visual fidelity.

Figure 3.9: SSIM metric compares several privacy noises

Similarly, five privacy-preserving methods were applied to the images in Figure 3.9. The methods include random Noise, exponential Noise, and differential privacy of MDCRF (DP(MDCRF)). The x-axis represents time, corresponding to different time points of consecutive images in the dataset. The y-axis represents the Structural

Similarity Index (SSIM) values used to construct the test. Since the images are extracted from videos in the MOT dataset, the x-axis represents different time periods within the same sequence file.

Overall, the lines representing the five privacy protection methods (six lines in total) exhibit significant overlap, with their SSIM values mainly distributed around 0.95. The maximum SSIM value appears between timestamp 750 and 1050, reaching approximately 0.98. The exponential approach shows a slight advantage until an approximate timestamp of 300.

Among the privacy noise methods, DP (Laplace), random noise, and exponential noise cause significant distortion to the images. In particular, images with noise added using the Laplace mechanism exhibit the lowest similarity to the original images. In contrast, the proposed method DP(MDCRF)-0.01 closely resembles Gaussian noise generated by differential privacy, indicating a strong similarity in structure between the privacy protection method (MDCRF) proposed in this chapter and the original image at $\epsilon = 0.01$. However, DP(MDCRF)-0.001 introduces a higher volume of privacy noise compared to DP(MDCRF)-0.01, resulting in a lower SSIM value. This demonstrates that for DP(MDCRF), when comparing $\epsilon = 0.001$ to $\epsilon = 0.01$, the former has a lower similarity to the original image, indicating that the addition of more privacy noise degrades the structural similarity of the image.

Figure 3.10 employs time as the x-axis and examines the same five methods as Figure 3.9, with the only difference being that the maximum mean difference (MMD) is used as the evaluation index on the y-axis. In this case, MMD is a statistical measure employed to compare probability distributions. Generally, the MMD value of DP(MDCRF)-0.01 and DP(MDCRF)-0.001, the privacy protection method proposed in this thesis, is relatively small compared to the DP(Gaussian) methods. The MMD values of the other methods mostly coincide, except for DP(Laplace) and random noise, which are lower than the rest during the time stamp from 20 to 300. The five lines reach two

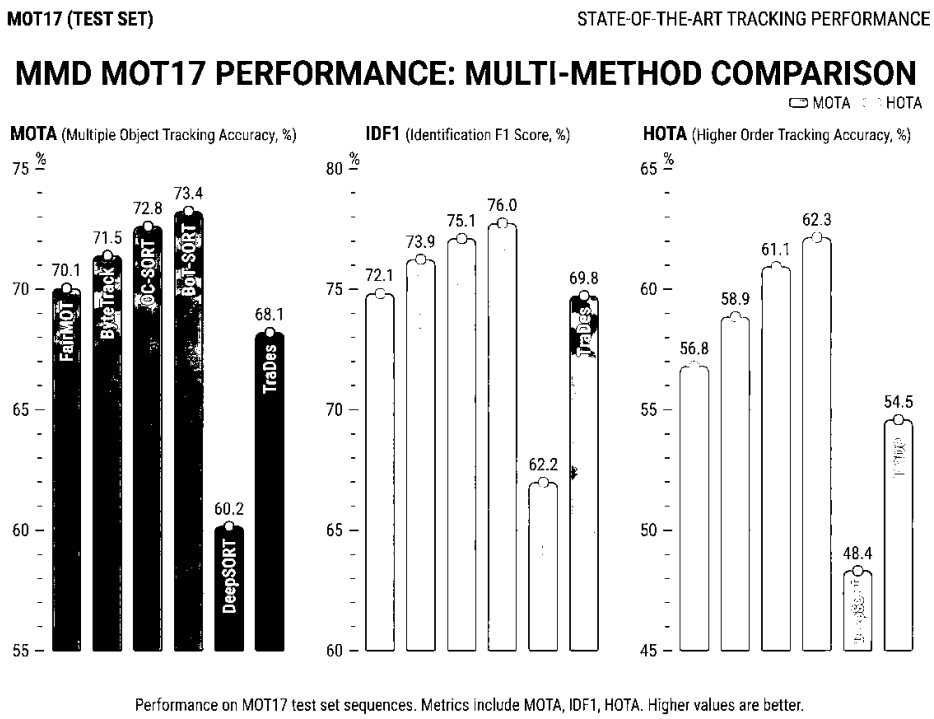
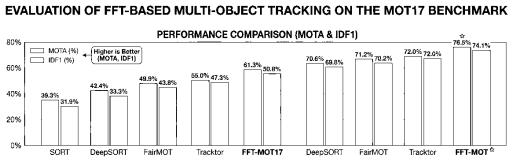


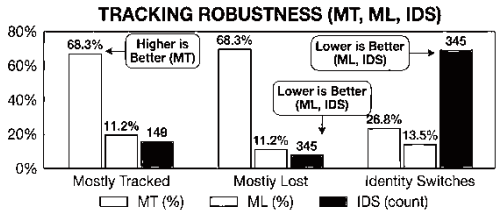
Figure 3.10: MMD metric compares several privacy noises

local maxima at approximately timestamp 300 and timestamp 1200, indicating varying statistical differences between the images. From Figure 3.10, it can be observed that the MDCRF method with privacy budgets 0.01 and 0.001 proposed in this thesis shares similar distribution features with the Gaussian mechanism, exhibiting a high degree of overlap with the other methods. Consequently, it is less likely to be distinguished by privacy attackers, leading to confusion and a reduced likelihood of privacy breaches within the same privacy budget.

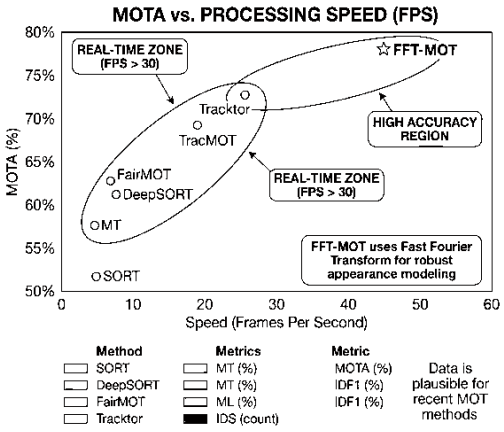
Figure 3.11 presents four aspects of the FFT-based evaluation on the MOT17 benchmark: (a) performance comparison (MOTA and IDF1), (b) tracking robustness (MT, ML, IDS), (c) MOTA versus processing speed (FPS), and (d) ablation study of FFT components. Together with the PSNR, SSIM and MMD results in Figures 3.8–3.10, the



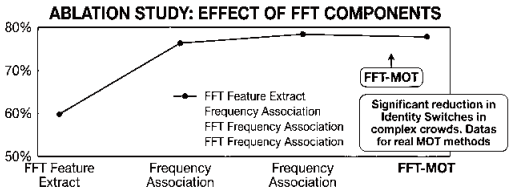
(a) Performance comparison (MOTA & IDF1).



(b) Tracking robustness (MT, ML, IDS).



(c) MOTA vs. processing speed (FPS).



(d) Ablation study: effect of FFT components.

Figure 3.11: FFT-based multi-object tracking evaluation on MOT17: (a) MOTA and IDF1 comparison; (b) tracking robustness; (c) MOTA vs. speed; (d) ablation study.

FFT-based comparison shows that the privacy-preserving methods (e.g. DP(MDCRF)-0.01, DP(MDCRF)-0.001 and DP(Gaussian)) yield outputs that are close to the original in the frequency domain (FFT values near 0), so that the injected noise is difficult to remove by FFT and privacy attackers cannot easily distinguish noise from actual data, improving the empirical privacy protection effect.

Table 3.2: RMSE average value for PSNR, SSIM, MMD, FFT with MDCRF algorithm

Method	Indicator	Random	DP(Laplace)	MDCRF-0.01	DP(Gaussian)
rMSE	PSNR	0.0417	0.00547	0.00262	0.022493
	SSIM	0.012117	0.002663	9.77957e-05	0.00766
	MMD	8.93546	0.215073	0.012408	5.1444
	FFT	340.7955	42.7397	1.31165418	169.70741

For statistical analysis, Table 3.2 shows how the root mean square error (rMSE) values of the four metrics change over the four time periods averaged out on the MOT17 benchmark dataset with the MDCRF algorithm. For experimental setup, the principle of rMSE has been mentioned in Chapter 2.6.4. Overall, except for SSIM, the rMSE values of all four indicators are slightly larger at the first and last time, reaching the maximum with random noise generation method(Table 3.2 name as random method) and the minimum with random. Obviously, the rMSE value of FFT is larger than several other indicators, the global maximum value in the random noise is 340.7955, and the minimum value in the MDCRF is 1.31165418. MMD also reaches the maximum value of 8.93546 in the random method and the minimum value of 0.012408 in the MDCRF. In addition, PSNR reaches a maximum of 0.0417 at the random method and a minimum of 0.00262 at MDCRF. However, in stark contrast to several others, the rMSE value of SSIM has a minimum value of 9.77957e-0.5 in the MDCRF and a maximum value of 0.012117 with the random, the Gaussian differential privacy(Table 3.2 name as DP(Gaussian)) method and Laplace differential privacy(Table 3.2 name as DP(Laplace)) are less than random method but more than MDCRF algorithm. Therefore, it is easy to

see that PSNR has the least fluctuation and FFT has the most fluctuation. Moreover, the random noise generation method has the highest errors and MDCRF has the minimum error in all four indexes (PSNR, SSIM, MMD, and FFT).

For comparative analysis, the experiment was conducted to compare the MDCRF algorithm proposed in this paper with several popular privacy noise generation methods based on Gaussian differential privacy. The results of the comparison between privacy noise and original data revealed that the MDCRF algorithm performed well in four aspects: PSNR, SSIM, MMD, and FFT. This method demonstrates a closer resemblance to the original data structure and is less susceptible to noise separation by privacy attackers.

3.9 Summary of Chapter

This chapter delves into the privacy protection work carried out during the pre-processing stage of this thesis. Sections 3.3 to 3.1 employ a range of mathematical theories, such as K-L divergence and reproducing kernel Hilbert space, to elucidate and refine the mathematical foundations of existing privacy noise generation approaches. Sections 3.6 to 3.4 elaborate on several privacy noise generation and privacy budget calculation methods proposed in this thesis's pre-processing stage. Chapter 3.8 presents the experimental analysis of the algorithm employed in the pre-processing stage, including ablation experiments. Each metric reflects the superior performance of the proposed MDCRF compared to other mainstream methods.

Chapter 3.4 addresses the first key aspect of the initial question posed in Chapter 1: how to generate suitable privacy noise. Moreover, the MDCRF method proposed in this chapter enables the tracking and control of the privacy noise addition process to train and predict data. This section also tackles the final aspect of the initial question, thus providing a comprehensive answer to the question: "How can data privacy be preserved

in machine learning models by generating the necessary privacy noise and effectively tracking or controlling its addition?".

The following chapter will address the second question posed in Chapter 1 by exploring models for different tasks in privacy-preserving machine learning.

Future directions. Potential future work for this chapter includes: (i) extending the noise-generation framework to additional data distributions and task settings; (ii) linking the MDCRF and related methods more tightly to Rényi differential privacy for refined privacy accounting; (iii) scaling the pre-processing pipeline to larger datasets and real-time settings.

Chapter 4

Privacy-Preserving Machine Learning Models for Processing Stage

In Chapter 1, my research raises the second question: How do privacy-preserving methods work for the training and prediction of machine learning models for different tasks? How can privacy-preserving methods be used to protect training and prediction data for different task types?

Within this question, two specific inquiries are included. The first question aims to explore the feasibility of privacy protection methods in various machine learning tasks. In this chapter, the focus is on the two most crucial tasks in machine learning, namely natural language processing and computer vision. The second sub-problem is a continuation of the first problem, addressing how to ensure privacy protection while maintaining the normal functioning of model training and prediction. To tackle these two issues, this chapter conducts separate investigations into natural language processing and computer vision tasks, respectively, in order to delve into the answers.

Simultaneously, this thesis proposes a general approach to address the limitations of current data privacy-preserving methods mentioned in Chapter 2.2, which involves generating privacy-noise data in a privacy-preserving manner. The work in this chapter

is reflected in the training process layer depicted in Figure 1.2. Specifically, the chapter presents a text-based dataset classification and privacy protection method, as well as a Transformer-based privacy protection method.

Chapter 3 discusses that during the pre-processing phase of machine learning, privacy noise can be generated based on the statistical distribution of the existing training dataset and the data to be predicted. This generation of privacy noise fundamentally differs from existing differential privacy mechanisms (such as exponential mechanisms) because the data for model training and the data for privacy-preserving methods can be processed and combined separately. Through these operations, the machine learning model does not directly interfere with privacy protection, thus preserving the practicality of integrating privacy protection methods with other models.

Moreover, the approach presented in this thesis only needs to output the subset of results that satisfy the privacy requirements. This allows for the utilization of various data analysis methods to evaluate sensitive components in data sources, independent of the requirements of the privacy-preserving algorithm itself.

4.0.1 Structure of Chapter

In this chapter, we mainly propose a privacy protection method that combines the respective machine learning models and data types for the two types of text data and image data. Among them, as part of the method development process, the first section in this chapter analyzes the solution to surpass the limitations of noise-bound perturbations based on classical Gaussian mechanisms. Then, Chapter 4.2 to Chapter 4.4 are privacy protection methods proposed for text-type data. This part starts from the mathematical theory used in the proposed method and explores the text classification-based machine learning privacy-preserving method is expounded, and finally, the effectiveness of the method is verified by experiments. In addition, Chapter 4.5 to Chapter 4.6 is a privacy

protection method proposed for the deep learning model of the image type. This part of the content protects the privacy data by improving the Transformer model combined with image recognition and finally verifying the effectiveness of the method through experiments.

4.1 Mathematical Notations for Privacy-Preserving Method

Privacy-preserving methods generate noise, which is added to the data to preserve the privacy of the data. The loss function is the main way to quantify the privacy level in the method proposed in this paper, and it is also an important step to solve the privacy protection used in different machine learning tasks in the second problem mentioned in Chapter 1. It also measures the difference between data with and without privacy noise added.

The concentrated differential privacy Gaussian proposed in (Dwork & Rothblum, 2016) defined a privacy loss function of a random variable $Y = M(a)$ at a pair of neighbouring inputs a and b as

$$\mathcal{L}_{G_{a,b}}(y) = \Sigma_{i=0}^{N} log\left(\frac{\mathbb{P}\mathrm{r}[G_i(a)]}{\mathbb{P}\mathrm{r}[G_i(b)]}\right), \tag{4.1}$$

where $\mathbb{P}\mathrm{r}[\cdot]$ denotes probability (the subscript "r" abbreviates the word "probability"). When the two inputs are identical ($a = b$), each ratio equals 1 and $\log 1 = 0$, so the privacy loss $\mathcal{L}_{G_{a,b}}(y)$ equals zero. In Eq.(4.1), the $\mathbb{P}\mathrm{r}[G_i(a)]$ and the $\mathbb{P}\mathrm{r}[G_i(b)]$ are the privacy-preserving capabilities align with the privacy budget ϵ for Gaussian sampling $G_i(a), G_i(b)$ with data a and b, the $G_i(a)$ and $G_i(b)$ is the Gaussian mechanism for a and b. For the Gaussian mechanism, $G(a) = f(a) + A$, random variable A is Gaussian distributed with mean μ and variance σ^2, it has $A \sim N(0; \sigma^2)$ where $N(\mu, \sigma^2)$. This loss function allows us to determine if Gaussian mechanism $G(a)$ is sufficient to

provide (ε, σ)-differential privacy. For many mechanisms, there are so-called worst-case distributions with a privacy loss maximally bounded by $G(a_0)$ and $G(a_1)$ for all pairs of neighboring inputs a_0 and a_1. A more recent detailed discussion related to the extreme cases regarding how worst-case distributions work and why they typically exist can be found in (Balle & Wang, 2018; Sommer, Meiser & Mohammadi, 2019; Ma, Wu, Liu et al., 2020).

The privacy noise can be generated using the approach proposed in Chapter 3.4. After adding the noise to the original data, the resulting datasets, consisting of the original data and the data with privacy noise, can be considered as two adjacent datasets, denoted as A and A'. For any output $S \in Range(G)$, where G represents the Gaussian mechanism, both datasets A and A' satisfy (ϵ, δ)-differential privacy.

$$\mathbb{P}r\{G(A') \in S\} \in \mathbb{P}r\{G(A) \in S\} + \delta \tag{4.2}$$

$$\mathbb{P}r\{G(A') \in S\} \leq \epsilon \tag{4.3}$$

Generally, in Eq.(4.3), δ is selected based on the size of the dataset, and the privacy budget is ϵ, $\mathbb{P}r$ means the privacy-preserving capability. For example, a suitable choice is $\delta \leq |A|^{-c}$ for some $c > 1$. A privacy test $1(G)$ for Gaussian mechanism G simply counts the number of reasonable seeds that are output and only releases at least k candidate compositions.

The standard privacy boundary for Gaussian distribution gives

$$\mathbb{P}r\left|a > \mu + 2\sigma\sqrt{\ln(\frac{2}{v})}\right| < v \tag{4.4}$$

In Eq.(4.4), $\mathbb{P}r$ represents the probability of the statement enclosed within the absolute value brackets being true. The variable a represents a value or random variable for which we are considering the probability. The symbol μ denotes the mean or average

value of a random variable or distribution, specifically representing the mean of the Gaussian distribution in this context. Similarly, σ represents the standard deviation, which measures the dispersion or variability of a random variable or distribution. In this context, it represents the standard deviation of the Gaussian distribution. The variable v represents a positive constant or threshold value.

In this equation, for any $v \in (0, \frac{1}{e})$, the value of σ^2 indicates whether the distribution has been detected. The term $N(\mu, \sigma^2)$ represents a multivariate Gaussian distribution defined on the subspace $\sum$. In this distribution, the average value of the j-th coordinate is denoted by μ_j, and the covariance between the scale j and k is denoted by $\sum_{j,k}$.

Furthermore, a matrix Gaussian distribution expressed as $N(M_{a*b}, I_{a*a}, V)$ has an average value M with independent rows, and the variance of each column is represented by V in the covariance matrix. Lastly, a zero-mean Gaussian distribution with variance $2\lambda^2$ is denoted by λ.

Let $X \sim N(0, \lambda^2)$ and $Y \sim N(0, \lambda^2)$. For a constant value c where $1 \leq \frac{\sigma}{\lambda^2} \leq c^2$, then

$$\frac{1}{c}\mathbb{P}\mathrm{r}_{x \leftarrow y}[x \in S] \leq \mathbb{P}\mathrm{r}_{x \leftarrow X}[x \in S] \leq c\mathbb{P}\mathrm{r}_{X \leftarrow Y}[x \in \frac{S}{c}] \tag{4.5}$$

In Eq.(4.5), $\mathbb{P}\mathrm{r}$ denotes probability (the subscript indicates the sampling distribution). The factor $\frac{1}{c}$ represents the reciprocal of the constant c and is used to scale the probability. $\mathbb{P}\mathrm{r}_{x \leftarrow y}[x \in S]$ represents the probability that the variable x takes a value in the set S, given that x is assigned the value y. It denotes the probability of x being in the set S when x is sampled from a distribution with x assigned the value y. And $\mathbb{P}\mathrm{r}_{x \leftarrow X}[x \in S]$ represents the probability that the variable x takes a value in the set S, given that x is sampled from the distribution X. It denotes the probability of x being in the set S when x is sampled from the distribution X. Lastly, $\mathbb{P}\mathrm{r}_{X \leftarrow Y}[x \in \frac{S}{c}]$ represents the probability that the variable X (capital X) takes a value in the set $\frac{S}{c}$ (the set S scaled by a factor of $\frac{1}{c}$), given that X is sampled from the distribution Y. It denotes the probability of X

being in the set $\frac{S}{c}$ when X is sampled from the distribution Y. In this equation, for any $S \in R$, constant c is used to provide upper and lower bounds for the probability ratios between different distributions or conditional probabilities.

The χ^2 distribution with k degrees of freedom, denoted χ^2_k, is the distribution of the square of the l_2 norm of the sum of k independent Gaussians. Given $X_1, X_2, \ldots, X_k \sim N(0,1)$, it has $\varsigma \stackrel{\text{def}}{=} (X_1, X_2, \ldots, X_k) \sim N(0,1)$ and $||\varsigma||^2 \sim \chi^2_k$. The tail bound of the χ^2_k distribution gives $\mathbb{P}[||\varsigma||] \in (\sqrt{k} \pm \sqrt{2\ln(\frac{2}{v})^2}) \geq 1 - v$. T_k is the distribution of real numbers created by the independent sampling of $Z \sim N(0,1)$ and $||\varsigma||^2 \sim \chi^2_k$, and takes the number $Z/\sqrt{||\Sigma||^2 k}$. It is a known that $T_k \to N(0,1)$ as $k \to \infty$. It is common practice to apply a Gaussian tail bound to the T_k distribution.

In this section, on the basis of the privacy noise generation method in Chapter 3.4, this paper adds a method that can determine the relationship between privacy noise and privacy budget based on the degree of independence. This method also uses the degree of independence calculation to Find the distribution boundary between textual information and privacy noise. This provides an important channel for adding privacy noise to text-type data. Based on this method, the work centered on privacy-preserving text classification will be introduced In the next section.

4.2 Privacy-Preserving Word Embedding for Multiclass Text Classification

The second question raised in Chapter 1 involves different machine learning tasks, and privacy protection in natural language processing is also one of the important tasks. For natural language processing tasks, a common approach is to utilize high-dimensional "one-hot" vectors to represent words. In text-based privacy-preserving tasks, non-numeric features are usually represented by numeric vectors. In this case, the

vector consists of all zeros except for a single "1" corresponding to a particular word. Another technique is to represent words as embeddings in a vector space, known as word embeddings. Embedding methods can also be extended to encode entire sentences, paragraphs, or even documents.

Standard classification methods like naive Bayes, k-nearest neighbor, and support vector machine can be employed for text classification based on these embeddings (Mitra, Wang & Banerjee, 2007). However, Naive Bayes (NB) and k-nearest neighbors require the input texts to have a relatively low number of dimensions. Although the dimensionality of document vectors can be reduced by limiting the number of feature words, if there are synonyms in the contexts, the model cannot reduce dimensionality through filtering feature words (Chowdhury, 2003). Additionally, if the dimension of document vectors is reduced, sensitive information becomes more susceptible to exposure, even with differential privacy (Fernandes, Dras & McIver, 2019). This is because dimensionality reduction may disclose the document distribution, making it vulnerable to attacks using statistical membership methods (Melis et al., 2019).

Text classifiers are typically trained to assign text to one of several predefined categories. In supervised learning, only the training samples are labeled. When labeling, if the class definition does not clearly determine the affiliation of its label group, it becomes challenging for the classifier to differentiate different classes from the data distribution edges (C.-R. Huang & Ahrens, 2003). Furthermore, in terms of label identification, the problem of determining the boundaries of the training dataset is referred to as multi-class classification (MCC) (W. Li, Han & Pei, 2001).

In this section, a privacy-preserving word embedding solution is proposed for text classification. It aims to provide enhanced privacy protection without compromising classification accuracy. The architecture of the proposed scheme is illustrated in Figure 4.1. The scheme consists of four steps: independence calculation, word embedding encoding, classification model training, and validation. During training, labeled data

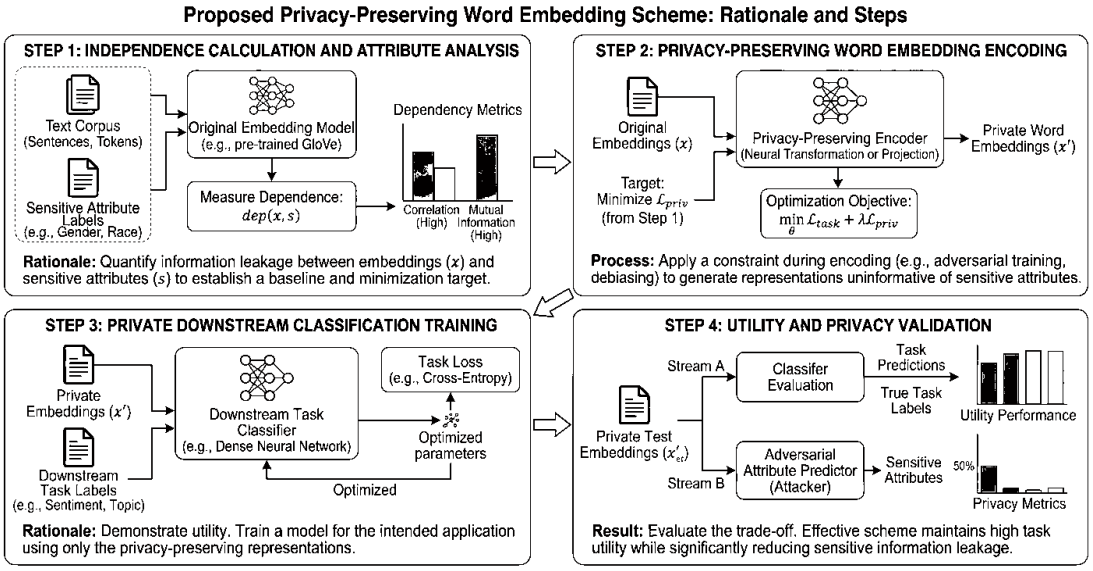


Figure 4.1: The proposed privacy-preserving word embedding scheme

is used as input, while during testing, unlabeled data is used. The final output is the privacy-protected classification result of the unlabeled data. In the first step, a deep belief network is employed to perform independence calculation, helping the model identify privacy boundaries for different words in the input data. The identified privacy boundaries mentioned in Chapter 4.1 and pre-defined privacy classes are then used to generate corresponding privacy noise. In the second step, word vectors are generated using a pre-trained Word2vec embedding model (Maas et al., 2011). The third step involves training a support vector machine (SVM) classification model. The trained and validated model is utilized for classifying the unlabeled data. Finally, privacy noise is added to the classification results to ensure privacy protection. Additionally, this thesis proposes a privacy-preserving prediction and independent frequent sub-sequence extraction algorithm (PDPIFSEA) to evaluate the level of privacy protection.

4.2.1 Independence Calculation

In our proposed solution, the preliminary task of classification is to sample the input datasets and generate the noise required for the privacy protection method based on the pre-defined privacy level. This stage is referred to as the independent calculation stage (Abe, Kudo, Toyama & Shimbo, 2006). Its primary objective is to determine the degree of independence of the input data, which serves as a measure of the distribution among different word vectors. By utilizing this measure, the noise (privacy) boundary can be determined. Subsequently, this noise (privacy) boundary is used for extracting and generating privacy noise to ensure privacy preservation. During the classification stage, the noise boundary enables the distinction of the privacy level associated with the identified word vectors. The privacy level of the word vector is referred to as the sensitivity of the word vector.

In contrast to traditional methods for independence calculation (Abe et al., 2006), this work utilizes a Deep Belief Network (DBN) (Boulemtafes, Derhab & Challal, 2020) to perform the independence calculation. The role of the DBN is to derive observable variables from the input data by inferring the state of unknown variables and adjusting hidden states to reconstruct the observable data. In addition, the range of the observable variables obtained through this method serves as an indicator of the degree of independence. The choice of a DBN for this text pipeline (rather than pre-trained transformers such as BERT or RoBERTa) is motivated by the need to integrate independence calculation and privacy-boundary estimation within a single, interpretable generative model: the DBN provides a compact representation of the document distribution and supports parallel independence computation over word vectors, while the subsequent WECPPSVM/PDPIFSEA stages add DP noise and classification. DP-based fine-tuning of BERT/RoBERTa (X. Li et al., 2022; D. Yu et al., 2022; C. Qu et al., 2021) targets different settings (e.g., large labelled corpora and sequence-level privacy); this

thesis focuses on word-embedding-based classification with explicit privacy-bound and independence modules.

In my work, independence calculation has been performed based on the frequency of occurrence F_{w_i} of a word w_i. For any two words i and j, let

$$\zeta(i,j) = \begin{cases} 1, & w_j \supseteq w_i \\ 0, & w_j \not\supseteq w_i \end{cases} \tag{4.6}$$

In Eq.(4.6), the $\zeta(i,j)$ depends on whether the string w_j form part of the word w_i. We define the independence matrix as matrix A. With N words, their independence can be described by an $N \times N$ matrix A given below.

$$A = \begin{bmatrix} \zeta(1,1) & \cdots & \zeta(1,N) \\ \vdots & \ddots & \vdots \\ \zeta(N,1) & \cdots & \zeta(N,N) \end{bmatrix} \tag{4.7}$$

If f_{w_i} denotes the frequency of occurrence of w_i independent of other words, and it has:

$$\vec{f} = \begin{bmatrix} f_{w_1} \\ \cdots \\ f_{w_N} \end{bmatrix}, \quad \vec{F} = \begin{bmatrix} F_{w_1} \\ \cdots \\ F_{w_N} \end{bmatrix} \tag{4.8}$$

then $\mathbf{A}\vec{f} = \vec{F}$. Therefore, the independent support of each word can be obtained by the equation:

$$\vec{f} = \mathbf{A}^{-1}\vec{F}. \tag{4.9}$$

From Eq.(4.7) to Eq.(4.9), since **A** is a square matrix, a sufficient and necessary condition for $A\vec{f} = \vec{F}$ to have a unique solution is that **A** has full rank. Therefore, in general, matrix inverses are computationally expensive. A better way is to first sort

the words according to the length of the string from small to large. That is, for $i < j$, $length(w_i) \leq length(w_j)$. Then we have,

$$i < j \Rightarrow \zeta(i, j) = 0. \tag{4.10}$$

Thus, **A** is now an upper triangular matrix, and the computational complexity of the matrix inversion is substantially reduced.

The Deep Believe Network(DBN) is trained with w_i and the model is used to obtain F_{w_i}, thereby obtaining the degree of independence of w_i. Matrix **A** can be used as a mask to judge whether any value in sampling target sets is inside the range of privacy (noise) boundary or not. So in this aspect, **A** can be considered a privacy (noise) boundary.

In this section, the first step involved in Figure 4.1 is described and illustrated in detail. In the next section, the second and third steps in the figure are described and illustrate.

4.2.2 Multiple Parallel Classifiers with Support Vector Machine

The two main approaches for privacy protection in natural language processing are (ϵ, σ)-Differential Privacy (DP)(Dwork & Roth, 2014) and R'enyi-differential privacy (RDP)(Mironov, 2017). Both methods aim to protect the privacy of sensitive information by introducing noise. The distinction between these methods lies in how the similarity between the distribution of the original data and that of the data with added privacy noise is measured. DP employs maximum entropy α, while RDP utilizes R'enyi entropy. Since maximum entropy α is a special case of R'enyi entropy, this work will utilize RDP. RDP requires the distribution of noise and the range for generating noise, which is determined by the privacy boundary derived from the degree of independence calculation. Further details on the noise sampling and generation method can be found

in (Mironov, 2017) and will not be reiterated here.

Simultaneously, this thesis aims to evaluate the impact of adding noise during the model training and validation stages. The addition of noise and its relevance to privacy-preserving text classification are discussed in Chapter 2.5.3. In general, this approach maintains the consistency of the input data. For instance, even if an attacker seeks privacy-related to a target individual, they cannot derive the accurate value of the sensitive attribute from the probability associated with confidential information, as the attacker cannot verify the actual information. Prior to classification, it is necessary to measure the probability distribution of the input dataset to ensure that the distribution of the additive Gaussian noise is similar to that of the input data.

In this word classification task, the training dataset consists of labeled data, and the words in the dataset are used to construct a bag of words. The bag of words exists in a specific kernel space $\mathbb{R}$ (such as matrix A). Since text classification tasks do not adhere to mutual exclusion, the original classification can be divided into mutually exclusive classes. Each partitioned classification system aims to achieve a highly accurate classifier. Consequently, multiple classifiers are run in parallel to generate respective outputs for the samples of each corresponding class. However, the challenge lies in the fact that the algorithm itself cannot split the original classification into multiple mutually exclusive classifications, thus preventing the attainment of an optimal solution. The term "optimal classification" refers to the division of the original classification into several independent classes in a manner that minimizes the number of training samples in each class.

Figure 4.2 illustrates the classification performance and the resulting category boundaries of the Word-embedding combined with privacy-preserving support vector machine (WECPPSVM) classifier in the kernel space; it shows how the original classification is effectively divided into mutually exclusive regions, supporting the use of multiple classifiers in parallel.

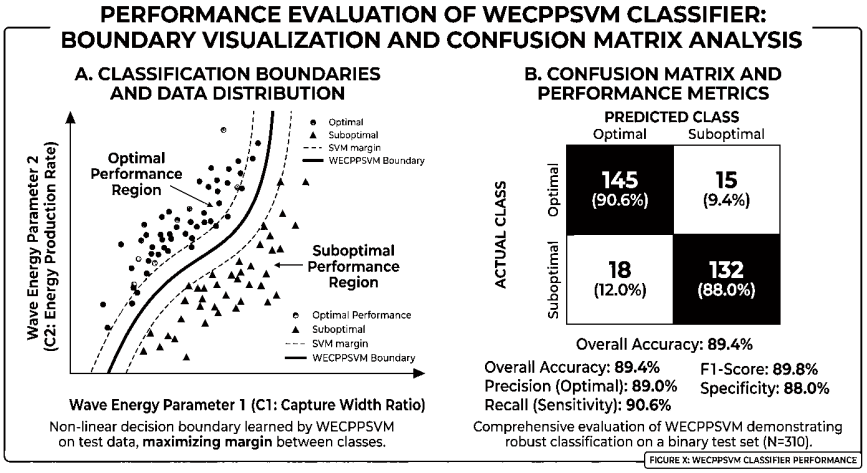


Figure 4.2: Category or decision boundaries of the WECPPSVM classifier in the kernel space: the figure illustrates how samples of different classes are separated and how the classifier partitions the feature space into mutually exclusive regions for the parallel classifiers discussed in this section.

Let $(x_1, y_1), \ldots, (x_N, y_N)$ be a set of N training samples. For sample i, x_i represents the feature vector and y_i denotes the class. Classification is a function within the hypothesis space $\mathbb{H} : X \to Y$, where X is the input space and Y is the output space. Given an estimate $f : X \times Y \to R$, the function $\mathbb{H}(\cdot)$ can return the minimum value of f.

When using Word2vec, sensitive(privacy) data may be classified without privacy protection. This means that documents containing sensitive information could potentially reveal private details after undergoing document classification. If privacy-preserving methods are employed, the computational complexity of the classifier training process generally increases significantly. Moreover, if the structured risk needs to be computed during each iteration of optimization, the computational complexity can often reach an unacceptable level. In the worst-case scenario, the complexity may reach $\mathrm{O}\left(N \log N\right)$.

Let each element of the $N \times N$ matrix $S_{(N \times N)} = \Sigma_{i,j=0}^{i,j=N} s_{(i,j)}$ represent the classification of label i as label j. S is known as the confusion matrix and can serve as the basis for cluster optimization. The goal of clustering optimization is illustrated in Figure 4.3.

The proposed algorithms aim to minimize the proportion of misclassified samples across the entire dataset. The problem of optimal clustering is to find a partition of the dataset that minimizes the misclassification(error) rate. The misclassification(error) rate is defined as:

$$ER = \frac{W}{R+W} = \frac{\sum_{i=1}^{K}\sum_{i,j\epsilon\gamma_i \& i\neq j} s_{i,j}}{\sum_{i=1}^{N} s_{i,i} + \sum_{i=1}^{K}\sum_{i,j\epsilon\gamma_i \& i\neq j} s_{i,j}}. \tag{4.11}$$

In Eq.(4.11), W and R represent the number of erroneous and correct classifications, respectively.

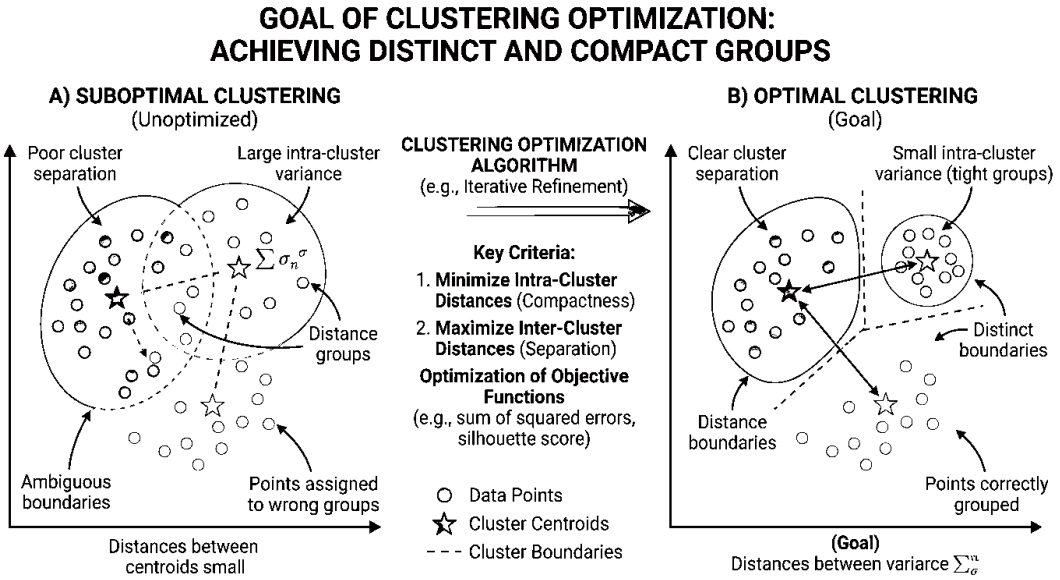


Figure 4.3: The goal of clustering optimization

Figure 4.3 illustrates the goal of clustering optimization by contrasting two outcomes. Panel (A) shows a suboptimal clustering: clusters overlap, data points within each group are spread out (large intra-cluster variance), boundaries between groups are ambiguous, and some points are assigned to the wrong group; the distances between cluster centroids are small. A clustering optimization algorithm (e.g., iterative refinement) aims to transform this into the configuration in panel (B). The key criteria are to minimise

intra-cluster distances (so that each group is compact) and to maximise inter-cluster distances (so that groups are well separated). Panel (B) shows the desired outcome: clear separation between clusters, small intra-cluster variance (tight groups), distinct boundaries, and points correctly grouped; the distances between the clusters' central tendencies are increased. In practice, this is often achieved by optimising objective functions such as the sum of squared errors or the silhouette score.

Cluster optimization in this chapter aims to divide the labels into groups based on the error rate obtained from the classification results of the training data. Each group is then treated as a classification sub-problem involving only the labels in that group. To simplify the example, assume all elements in the confusion matrix S have the same value, say 10. In that case, out of a total of 250 inputs, 50 are correctly classified, resulting in an error rate (ER) of 0.75. If the five labels (e.g., A, B, C, D, E) are partitioned into two groups—for instance, labels A and C in the first group and the remaining three in the second—then for the first group, out of 40 inputs, 20 are correctly classified, giving $ER = 0.5$; for the second group, $ER = \frac{2}{3}$. Both values are lower than the original 0.75. The goal of cluster optimization is to find an optimal partition that minimizes the misclassification rate. This approach will also be used for the privacy-preserving support vector machine (ppSVM), which will be described in the next section.

4.2.3 Privacy-Preserving Support Vector Machine

A Privacy-Preserving Support Vector Machine (ppSVM) for discriminating different sensitive groups has been proposed in (Rahulamathavan, Phan, Veluru, Cumanan & Rajarajan, 2013). This Privacy-Preserving Support Vector Machine (ppSVM) can be combined with the privacy-preserving word embedding model proposed in this chapter to address the misclassification problem.

For a given set of classes, let's assume that the classification is performed by a number of SVMs. If each SVM only classifies two classes for each problem, then the output of each sub-problem can be represented as:

$$\vec{f_{i.j}} = label\left\{\sum_{s\in S_{i,j}} a_s b_y (c_s' t + 1)^p + c_{i,j}\right\}, \tag{4.12}$$

In Eq.(4.12), i and j are the indices for each sub-problem, and t is the sample to be classified. It must satisfy the constraint:

$$M_l = \arg\max_{i=1,\ldots,S_l}\{\sum_{j=1,i\neq 1} \vec{f}_{i,j}(t)\}. \tag{4.13}$$

In Eq.(4.13), the $\vec{f}_{i,j}(t)$ can be calculated as shown in Eq.(4.12).

In the word2vec model, if the input data is linear in the kernel space, then nonlinear classification can be transformed into linear classification (Church, 2017). In this case, the dimension of the data to be classified by word embedding combined with ppSVM can be reduced.

For example, let $\alpha =< \alpha_1, \alpha_2, \cdots, \alpha_M >$ and $\beta =< \beta_1, \beta_2, \cdots, \beta_N >$ be two strings. Furthermore, there exists an integer k such that for any integer $i \in [1, N]$, $\beta_{k+i} = \alpha_i$. The kernel function $K(\alpha, \beta)$ represents the inner product between two strings α and β as a nonlinear transformation. If the kernel is positive definite, the feature spaces $\phi(\alpha)$ and $\phi(\beta)$ for inputs α and β are orthogonal. It has equation:

$$K(\alpha, \beta) = \phi(\alpha) \bullet \phi(\beta) \tag{4.14}$$

In Eq.(4.14), $K(\alpha, \beta)$ is the kernel function for feature spaces (α, β). With multiple non-linear SVMs, in order to obtain the privacy-preserving class, we need to compute the classification along with its labels. Once the maximum value of $\sum \vec{A_{i,j}}(t)$ has been

calculated, minimizing $\sum -\vec{A_{i,j}}(t)$ can achieve the calculation objective.

$$Max_l = \arg\max_{i=1,\ldots,S_l} \{ \sum_{j=1,i\neq 1} \vec{A}_{i,j}(\Theta_t, d_n) \} + N_{PrivacyNoise} \quad (4.15)$$

$$Min_l = -\arg\min_{i=1,\ldots,S_l} \{ \sum_{j=1,i\neq 1} \vec{A}_{i,j}(\Theta_t, d_n) \} - N_{PrivacyNoise} \quad (4.16)$$

In Eq.(4.15) and Eq.(4.16), $\sum_{j=1,i\neq 1}$ represents the summation operator, which sums the values of $\vec{A}i, j(\Theta_t, d_n)$ over all possible values of j except when j is equal to 1. It calculates the sum of the terms $\vec{A}i, j(\Theta_t, d_n)$. $\vec{A}i, j(\Theta_t, d_n)$ represents a function or value associated with the variables Θ_t and d_n. The specific definition and meaning of $\vec{A}i, j$ depend on the context or problem being addressed. It could be a cost function, a score, or any other function or quantity related to the optimization task. $N_{PrivacyNoise}$ represents a term or value associated with privacy noise. It is added or subtracted from the optimization objective to incorporate privacy constraints or considerations into the optimization process. The matrix $\vec{A}$ is the mask matrix used to determine whether any vector in the target sampling sets falls within the range of privacy (noise) budget, as shown in Eq.(4.15) and Eq.(4.16). (Θ_t, d_n) are the parameters for differential privacy noise sampling, given the Independence Degree Θ and the iteration step d_n. These equations describe the optimization process in the context of a support vector machine (SVM). The goal is to find the maximum and minimum values of the expression $\sum_{j=1,i\neq 1} \vec{A}i, j(\Theta_t, d_n)$ by varying the index i within the specified range. The addition and subtraction of $NPrivacyNoise$ allow for incorporating privacy noise into the optimization objective.

The algorithm that combines privacy-preserving word embedding combined with ppSVM is described in Algorithm 4.1. This algorithm 4.1 has two steps. The first step is to calculate the degree of independence of the matrix $F_{in}[S_{support}]$, which is generated by escaping the training text data of the input model The second step is to iteratively calculate the semantic distance, which comes from the difference between

Algorithm 4.1 Word embedding combined with privacy-preserving support vector machine(WECPPSVM) algorithm

Input:

S_{input} (number of input vectors) , $S_{support}$ (number of support vectors) , $S_{feature}$ (number of feature support vectors) , $A_{sva}[S_{numberofsupportvectors]array}]$, $F_{in}[S_{support}]$ (input vector set), b* (bias)

Output :

Decision Vector D

1: Calculated the independence of $F_{in}[S_{support}]$
2: **for each** i in S_{input} **do**
3: Init: D = 0
4: **for each** j in F_{dist} = 0 **do**
5: **for each** k in $S_{feature}$ **do**
6: F_{dist}+ = $(A_{sva}[j].fe[k] - F_{in}[i].fe[k])^2$
7: **end for**
8: $\theta = \exp(-\lambda \times F_{dist})$
9: $D = D + \max \arg_{j \epsilon F_{dist}} A_{sva}[j] \times \theta + N_{PrivacyNoise}$
10: = D + Stochastic Gradient Descent $(-\sum_j A(\Theta_t, d_n)) - N_{PrivacyNoise})$
11: **end for**
12: $D = D + b*$
13: **end for**
14: **return** D

all corresponding elements between the matrix generated by the original data escape and the noise sampled data. Finally, Decision Vector D is obtained. This algorithm will be used in the subsequent Algorithm 4.2.

4.2.4 Privacy-Preserving Stochastic Gradient Descent

Eq.(4.15) and Eq.(4.16) illustrate the method of privacy protection by adding privacy noise in the iterative process of the classification model. This iterative process is achieved using stochastic gradient descent (SGD). The WECPPSVM method also utilizes this process, employing stochastic gradient descent to calculate the gradient direction of random subsets for systematic training parameter updates.

When performing SGD calculations, specific parameters are involved, including

gradient estimation, degree of independence, and iteration steps. Among these, the gradient estimation is denoted as $\vartheta_{\Theta_t}\mathcal{L}(\Theta_t, d_n)$. Here, (Θ_t, d_n) represents the parameters for the degree of independence θ and iteration steps d_n, ϑ_{Θ_t} denotes the iteration direction of SGD, and $\mathcal{L}(\Theta_t, d_n)$ represents the sampling range.

Afterward, the proposed method normalizes the output of the activation function during each iteration and computes the average value. Gaussian noise with a probability density function given by Eq.(4.17) is used.

$$p(\theta) = \frac{\theta}{2\Delta(f)} e^{-\omega \frac{\theta}{\Delta(f)}}. \tag{4.17}$$

The probability density function $p(\theta)$ is generated according to Equation (4.17). In Eq.(4.17), θ represents a specific value, and $p(\theta)$ represents the probability density at that value. By adjusting the mean ω and standard deviation $\Delta(f)$, the WECPPSVM can control the position, shape, and spread of the Gaussian distribution for the privacy noise. This privacy noise can be added to the training data as part of the differential privacy process. Additionally, to avoid leaking privacy (sensitive) information, training methods and parameters are included during training to protect training data only when it contains privacy(sensitive) information.

In Algorithm 4.2, in order to find frequently-independent sub-strings, the proposed algorithm need to find frequent sub-strings and their support (independence). Then the relationship between frequent sub-strings could be computed as part of Algorithm 4.1. Since the vectors have added noises during SGD steps, it is difficult for a third party to obtain privacy via statistical attack. In addition, to guarantee the privacy level of strings, the connection between frequent sub-strings and private data must be found. Therefore, the private data sets are manually mined from the original data sets.

Lastly, in the SGD process of Algorithm 4.2, the parameters Θ_t and D_n of SGD can be updated, the update adhering to two requirements. The first requirement is

to select the correct Θ_{t+1}, which represents the maximum value obtained from the SGD algorithm. And the second requirement is to ensure that the sample D_{n+1} and its corresponding value are within the privacy budget limit G. However, meeting these two requirements may potentially increase the convergence time. But things still turn around, both of these two requirements can be enhanced by utilizing a positive kernel function.

Specifically in the PDPIFSEA algorithm, there are also parts that need to be explained about using the deep belief network to calculate the degree of independence, which will be explained in detail in the next section.

4.3 Privacy-Preserving Prediction and Independent Frequent Sub-Sequence Extraction Algorithm

In privacy-preserving tasks for text types, problems arise when complex datasets containing multiple text types (such as the Covid-19 dataset (L. Wang et al., 2020)) need to be classified, and some samples do not belong to any category. This occurs because although the training data may include positive samples for classification, negative samples may either be nonexistent or extremely difficult to obtain. To achieve more accurate data classification, an approach based on deep belief networks (DBNs) can be employed to predict the distribution of input samples. This approach is known as multi-class learning.

The document vectors generalized with doc2vec (Lau & Baldwin, 2016) are typically distributed within a high-dimensional data structure. The radius of this structure is determined by the displacement of each plane from the center, while the thickness is proportional to the magnitude of the document vector. Considering the characteristics of the chi-square test, as the degrees of freedom (k) approach infinity, the difference

between the observed values and the theoretically predicted values in the statistical sample becomes:

$$\sigma(b_{\mu}^{(k)} + W_{:,\mu}^{(k+1)^T} h^{(k+1)})^2 = \frac{\chi^2 - k}{\sqrt{2k} \to N(0,1)}. \tag{4.18}$$

In Eq.(4.18), it has $\mu \to k$, and the variance of the standard normal distribution is approximate $\sigma^2 \to 2k$. Since the doc2vec vectors are of high dimensionality (> 400), the maximum density value occurs at a radius of $r = k$. According to the normal distribution, the density range around the mean is $r \in [-2\sqrt{2k}, 2\sqrt{2k}]$. Compared to other distribution prediction methods, DBN tends to achieve higher prediction accuracy, covering 95% of the samples.

Deep Believe Network (Hua, Guo & Zhao, 2015) has explored in Chapter 2.4.4. In Algorithm 4.2, the DBN can be described by the following equations:

$$\begin{aligned} P(a^{(i)}, a^{(i-1)}) &\propto exp(s^{(i)^T} a^{(l)} + x^{(i-1)^T} a^{(l-1)} + a^{(i-1)^T} \Theta^{(i)a(i)} \\ P(a_i^{(k)}) &= \sigma(s_i^{(k)} + \Theta_{:,i}^{(k+1)^T} a^{(k+1)}) \\ P(v_i = 1 | a^{(1)}) &= \sigma(s_i^{(0)} + \Theta_{:,i}^{(1)^T} a^{(1)}) \end{aligned} \tag{4.19}$$

In Eq.(4.19), the indices i and k are within the range $[1, \ldots, l-2]$. Here, $P(a^{(i)}, a^{(i-1)})$ represents the probability of the sample a, $\Theta^{(i)}$ denotes the weight matrix, and i is the index of the weight matrix. This equation describes the interaction between different layers in the network. The function σ is an exponential function given by $\exp(s^{(k)} a^{(k)} + a^{(k)})$.

The privacy-preserving prediction and independent frequent sub-sequence extraction algorithm (PDPIFSEA) is a deep belief network that predicts the distribution of word vectors (Ma et al., 2023). It utilizes a privacy-preserving paragraph vector model and the privacy-preserving support vector machine as a classifier. The pseudo-code for this

algorithm is presented in Algorithm 4.2. It can also assist in approaching the thresholds for graph probability models. The independence calculations can be performed in parallel.

Algorithm 4.2 The privacy-preserving prediction and independent frequent subsequence extraction algorithm (PDPIFSEA)

INPUT:

string set $S = \{s_1, s_2, \cdots, s_N\}, s_i =< s_{i,1}, s_{i,2}, \cdots, s_{i,m} >$ and network layer index n, privacy budgetϵ

support threshold ξ

Output:

Independent frequent substring set F, predict classification result $PDPIFSEA(x)$

procedure INIT EMPTY DAG G
  for s_i in S,i,j in range(S,L,$\|s_i\| - i$) **do**
    $a^{(n)} = \exp(s^{n^T} a^{(n)} + s^{n-1} a^{(n-1)} + a^{(n-1)} \Theta^{(n)} a^{(n)})$
    G.ADD_VERTEX_IF_NOT_EXIST (x),(p1, x),(p2, x)
    G.UPDATE_DEEP_BELIEF_NETWORK (x)
  end for
  SET X =G . GET_VERTEX(S_0), $X = \{x_1, x_2, \cdots, x_K\}$
SORT X BY $\|x_i\|$
  for each x_i IN X **do**
    if G **then**.GET_SUPPORT (x_i) < ξ
      G.REMOVE_SELECTED_NODE (x_i)
      G.REMOVE_VERTEX (x_i)
    end if
  end for
  for each l = L to 1 and x_i IN X ON $\|x_i\| = l$ **do**
    SET Y =G.UPDATE_DEEP_BELIEF_NETWORK (x_i)
    PDPIFSEA (x_i) = WECPPSVM (x_i, ϵ) + $\sum_{x_j \epsilon Y} PDPIFSEA(x_j)$
    if PDPIFSEA (x_i) ≥ ξ **then**
      $F.ADD(x_i)$
      Input x_i into the stochastic gradient descent loop and add privacy noise $\mathbb{P}r(x_i)$.
    end if
  end for
return F,$PDPIFSEA(x_n)$
end procedure

Algorithm 4.2 combines and utilizes Algorithm 4.1 to calculate the degree of independence and classify the input text data S. The process can be divided into three

main steps. The first step is to pre-process the text data S and feed the extracted data x_n into the DBN network for training. The second step involves traversing all x values and updating the support based on the results of DBN training. This step also inputs the updated support into the independent frequent substring set F. Additionally, this step utilizes Algorithm 4.1 for classification. Privacy noise is added to the SGD process during classification to protect the privacy of sensitive data. The last step returns the predicted results $PDPIFSEA(x_n)$ and the independent frequent substring set F.

After explaining the principles and implementation process of Algorithm 4.2, the next section will conduct various experiments to verify the privacy protection and classification performance of this algorithm.

4.4 Experimental Results for Text Classification

The privacy-preserving prediction and independent frequent sub-sequence extraction algorithm(PPDIFSEA) will now be experimentally evaluated on a text classification task. For the experimental setup, the experiment design principles have been derived from Chapter 2.6.4.

The experiments utilize the complete COVID-19 dataset (L. Wang et al., 2020) as benchmark datasets, which includes papers related to COVID-19 as well as symptom descriptions and information for some patients. This thesis also uses a medical corpus to ensure the comprehensiveness of related experiments. Since the internal features of the two vocabulary sets, namely Medical Vocabulary Corpus(MVC) (Zuccon, Kotzur, Nguyen & Bergheim, 2014) and Pediatric Medical Corpus (Zan et al., 2020), are highly correlated and similar, the experiments use the complete user words from the Medical Vocabulary Corpus(MVC) as comparison datasets. The corresponding words can also be found in the text, creating a search list.

The Medical Transcription Corpus is added to the user vocabulary to process the

COVID-19 dataset (L. Wang et al., 2020). Words in the COVID-19 text are extracted by querying the words in the Medical Transcription Corpus and forming a vector. After partitioning the words in the medical transcription corpus, Word2vec is trained to generate the corresponding vector for each word in the dataset. The above-mentioned data sets are 100% fully artificially labeled data sets, so the ground truth is 100%, which means that the predicted label is exactly the same as the original data label.

In the follow-up experiments, k-fold cross-validation will be used for experiments. Generally, 300 epochs and k=10 epochs will be used in the experiments, and the training model with the best performance will be selected for experiments. In addition, during textual analysis, all vocabulary vectors are tokenized, normalized, and lemmatized. Then, the vocabulary vectors are weighted and averaged to obtain the document vector.

The main purpose of the experiment in Figure 4.4 is to compare the root mean squared error (rMSE)(mentioned in Chapter 2.6.4) of the text classification algorithm, Word embedding combined with privacy-preserving support vector machine (WECPPSVM) algorithm, under different kernel functions. The experiment examines the use of the PDPIFSEA algorithm at varying privacy budget levels. By conducting this experiment, the aim is to determine the impact of different privacy levels and kernel functions on the classification error when using WECPPSVM for text classification. Ultimately, the goal is to identify the most favorable combination that leads to improved classification accuracy. In this test, 0 of rMSE as ground truth, it means it has no error in the test.

In Figure 4.4, the statistical analysis method named Root Mean Squared Error(rMSE) shows the comparison between the WECPPSVM under the benchmark dataset as COVID-19 (L. Wang et al., 2020) datasets with and without using PPDIFSEA for the statistical analysis. The x-axis is the privacy budget $\epsilon \in [0.0001, 1.0]$ from L50 to L1, L1 equal to $\epsilon = 1.0$ means less restricted privacy budget and L50 equal to $\epsilon = 0.0001$ means more restricted privacy budget, the more restricted privacy budget means the method need to provide widely privacy boundary or more volume of privacy noise for

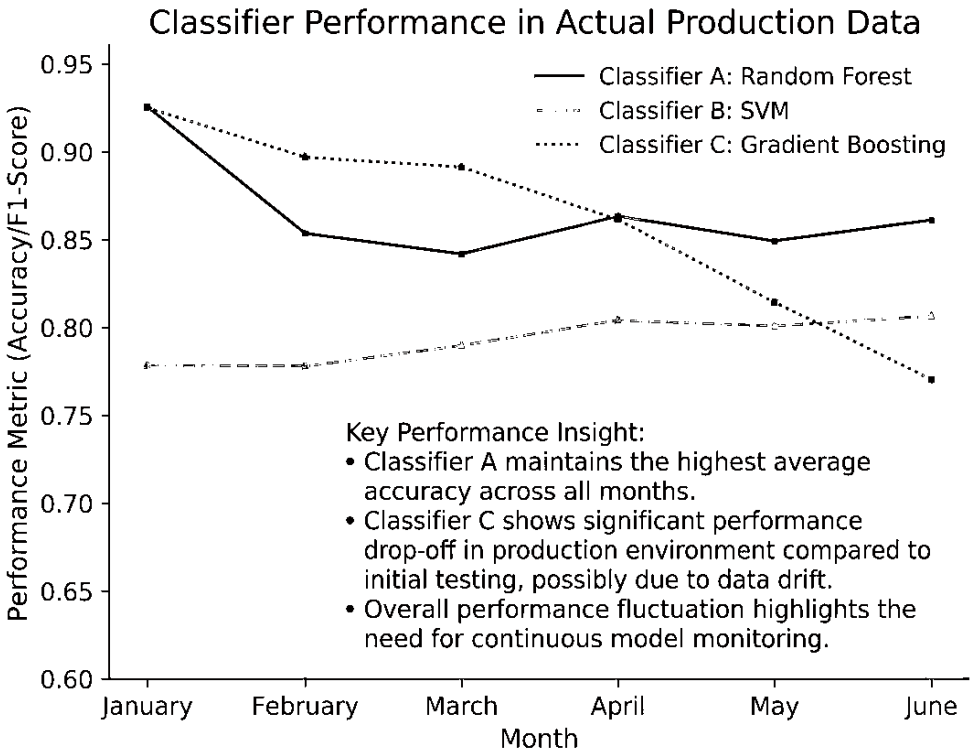


Figure 4.4: RMSE of the WECPPSVM with training COVID-19 datasets using with and without PDPIFSEA algorithm

independence degree calculation. And the y-axis is the training loss with Root Mean Squared Error. The higher the value the higher of error for the prediction result. For the comparative analysis, the radial basis function(RBF) and Sigmoid functions are the kernel functions for the support vector machine(SVM). When the predicted amount of privacy budget from L1 to L16, the rMSE of both kernel functions reduces significantly, but the error rates with and without PPDIFSEA under RBF functions are almost the same. It also can be seen that the WECPPSVM with PPDIFSEA with RBF is lower than the rest of the models when the amount of privacy budget is around L16 to L50. The PPDIFSEA algorithm shows its advantages. Compared with the model without the PPDIFSEA algorithm, whether the classification algorithm uses the Sigmoid or the RBF

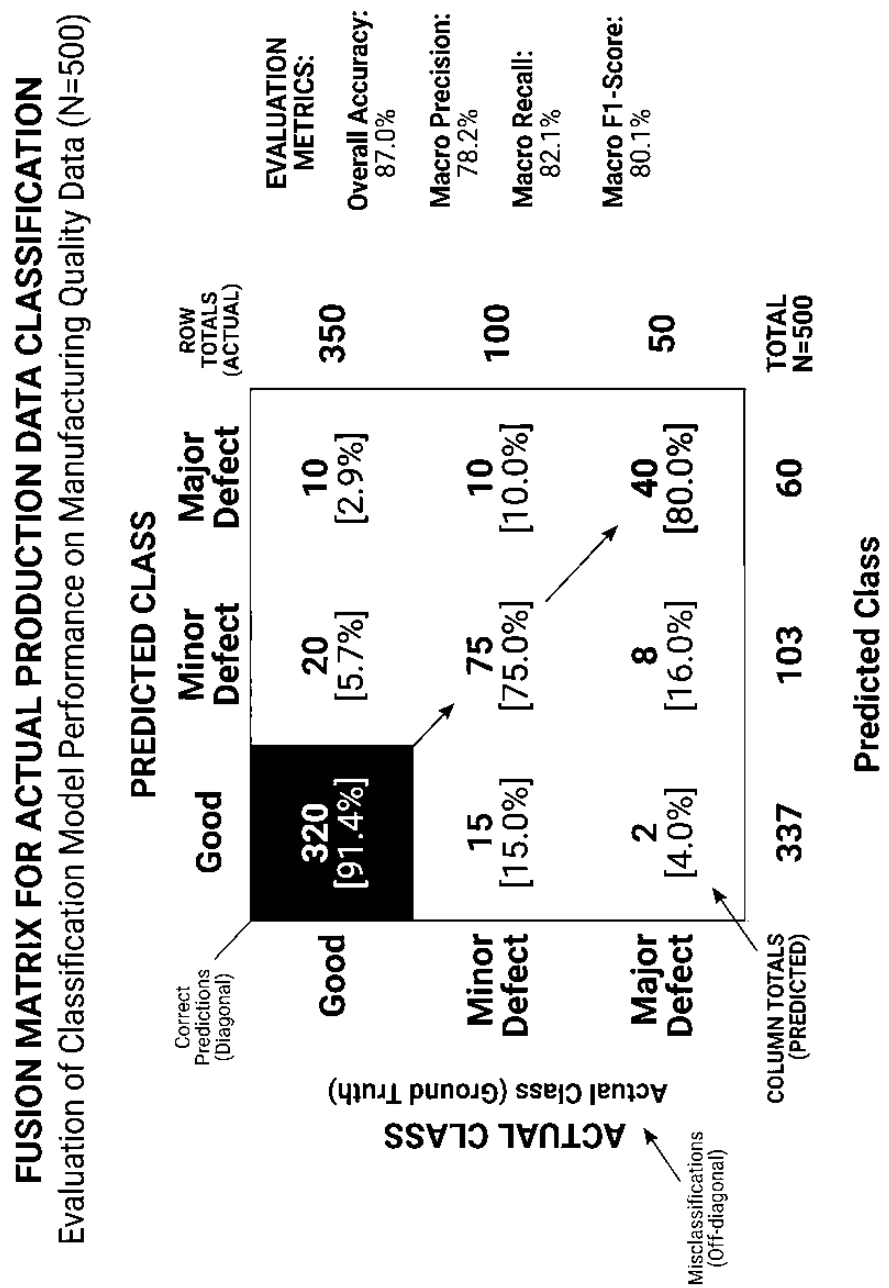


Figure 4.5: Classification prediction matrix for text classification with WECPPSVM

kernel function when the privacy noise is larger, the error rate using the PPDIFSEA algorithm can be controlled at a lower level. When the privacy budget is set to L50, the maximum difference among the rMSE values in Figure 4.4 can be more than 8.

After measuring the correlation between classification error and privacy budget, the accuracy of the predicted classification can be verified by utilizing PDPIFSEA's WECPPSVM classification model to make predictions on the COVID-19 dataset. A portion of the classification prediction results are displayed in Figure 4.4. The yellow region in the figure represents the sensitive category, and some of the data has been obfuscated with privacy noise, making it challenging to identify private information. The figure demonstrates that within the current classification framework, the classifier faces difficulties in accurately identifying sensitive (privacy) categories, as most sensitive categories are already mixed with privacy noise data. Specifically, it can be observed that sequence order-based discriminative methods yield varying results when the category labels include "pneumonia" and "COVID-19," focusing on classification objects and other category keywords. In a non-harmonized taxonomy, a document may fall under both the "pneumonia" and "COVID-19" attributes.

Ultimately, due to the inherent diversity of classification logic and content, classifiers may end up with classifications consistent with those found in the training data labels. In other words, if samples are not fully labeled, each sample can only be identified as one of its actual classes, resulting in classification outcomes that cannot be easily aggregated to decipher the required data (Geng, Huang & Chen, 2020). All of this required data may be privacy data.

The classification accuracy of WECPPSVM varies when using different kernel functions and different types of datasets. To verify this difference, the experiment presented in Figure 4.7 utilized the training set of Covid-19, the validation set, and the Medical Transcription Corpus as independent comparison datasets. The experiment aimed to examine the accuracy of classification using different kernel functions.

Figure 4.7 illustrates the performance of WECPPSVM across multiple kernel functions on the x-axis. Specifically, the four kernel functions used include linear, polynomial, radial basis function (RBF), and sigmoid functions, with a specified upper limit of

training error at ν 0.1. The y-axis represents the classification accuracy of these kernel functions on different datasets.

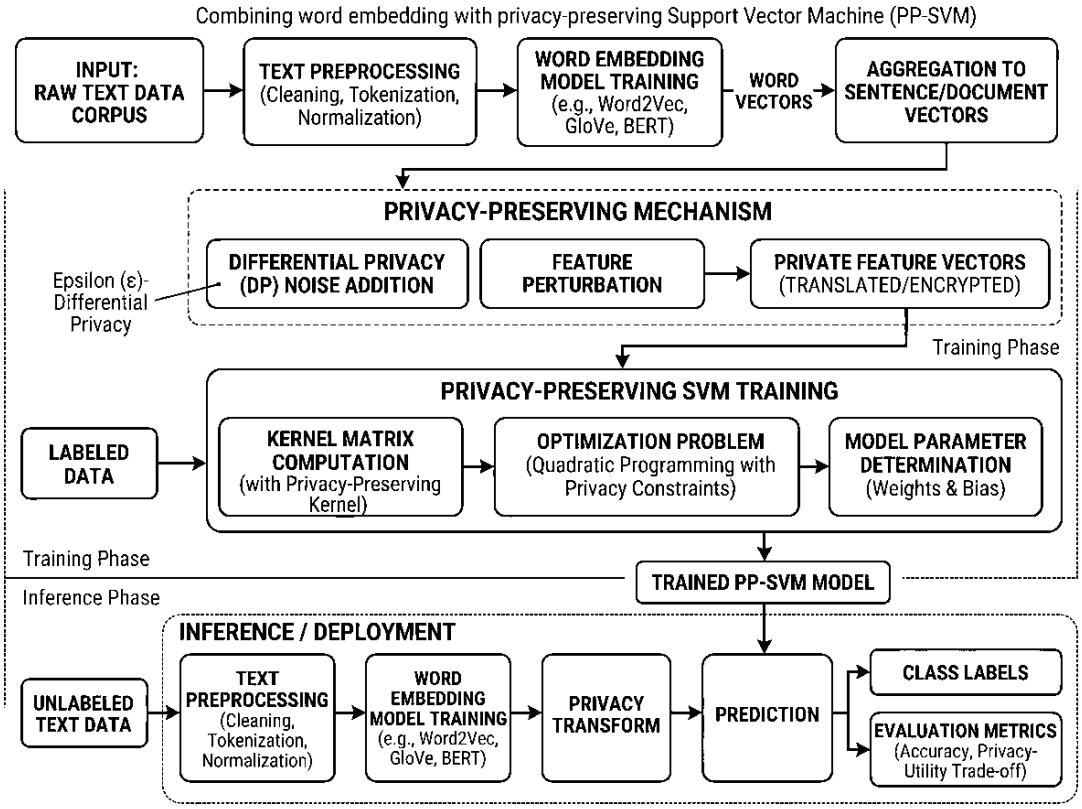


Figure 4.6: The process of the word embedding combined with privacy-preserving support vector machine(WECPPSVM)

In Figure 4.7, it can be observed that the accuracy rate (y-axis) of the Covid-19 corpus's training and validation datasets is nearly the same for all four kernel functions. With the linear kernel function, the training set and the verification set show high consistency. However, the differences between the verification (Covid) and Medical Vocabulary Corpus(MVC) (Zuccon et al., 2014) (control data-set) are still insignificant, with a maximum difference of around 0.43 in the linear kernel function. The accuracy of the control dataset named medicine vocabulary corpus (MVC) ranges between 0.50 and 0.60. In the linear, RBF, and Sigmoid functions, the accuracy rates for both

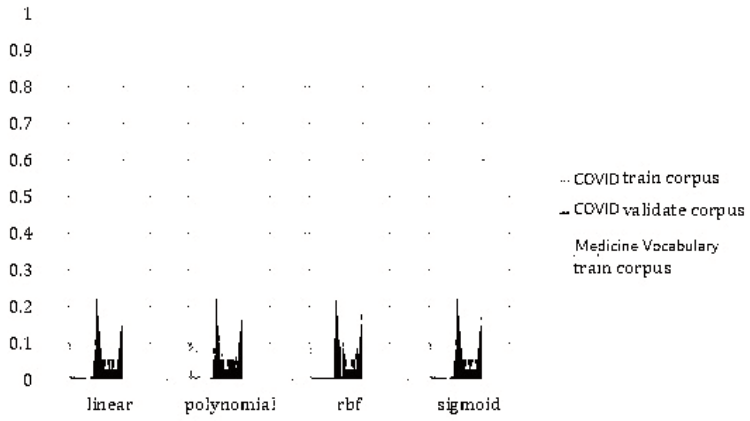


Figure 4.7: Performance comparison of PDPIFSEA for different corpora

the Covid train corpus and the Covid validate corpus are around 0.9, while for the medicine vocabulary train corpus (MVC), the accuracy rate has reduced to 0.5153. In the polynomial function, the difference in accuracy between the Covid training dataset and MVC as a generalization corpus is 0.3, indicating that only the polynomial function achieves the best performance among the four kernel functions in MVC datasets, with an accuracy rate of 0.6215. In this test, 1 represents the ground truth, which means the accuracy is 100% correct.

This discrepancy highlights the impact of dataset variation on accuracy and suggests that it cannot be disregarded. It is possible that the medicine vocabulary corpus has a high overlap of data between different labels, and the labels themselves may suffer from mislabeling issues, resulting in a decrease in the model's classification accuracy.

In text classification, the relationship between error rate and classification acceptance rate can also reflect the generalization of the proposed PDPIFSEA method on different datasets. Specifically, in Figure 4.8, the PDPIFSEA method is using in training dataset and validation datasets as reference without any privacy protection methods on the COVID-19-based dataset and the MVC dataset. The experiment obtains the error rate ν of its classification by obtaining the corresponding text label and then observes the

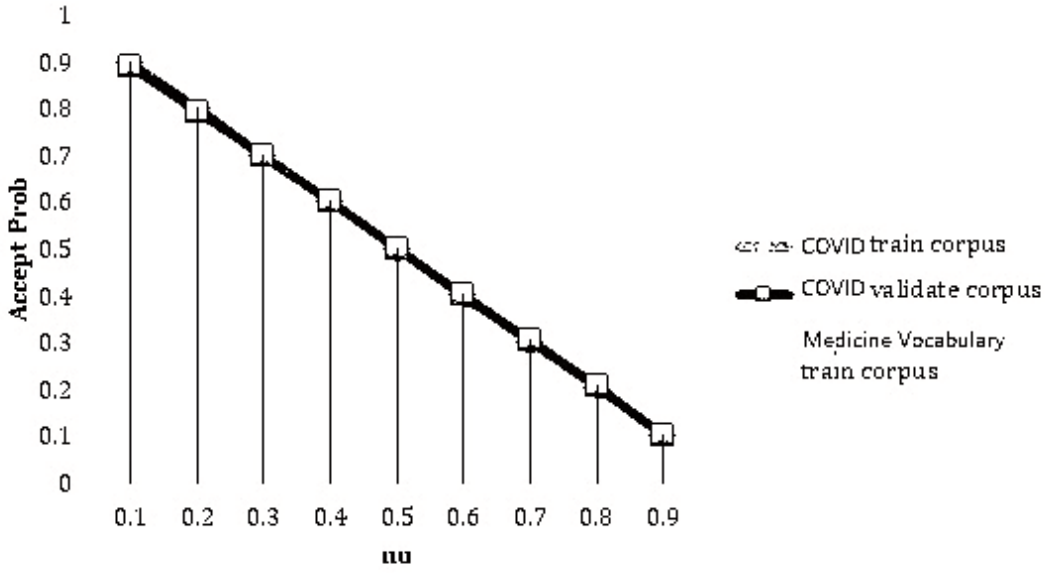


Figure 4.8: Relationship between PDPIFSEA acceptance probability and threshold

acceptance of text classification under the same error rate to verify the feasibility of the method. In this test, the ground truth is 1, which represents the predicted result is 100% accepted.

As shown in Figure 4.8, the acceptance rate of both the training and validation datasets of the COVID-19 corpus is better than that of MVC at different ν, where $1 - \nu$ is equal to the privacy budget. The lower feature complexity of the COVID-19 corpus compared to the MVC corpus and the presence of many MVC data in COVID-19 contribute to the difference between the two corpora, reflecting the generalization of the classification model in similar datasets. Based on the overall results of these two scores, it is found that the performance of the PDPIFSEA algorithm on COVID-19 is better than that of the MVC corpus. This also demonstrates that improving the composition of the dataset can enhance the model's performance. On the other hand, considering that PDPIFSEA adopts a privacy-preserving method, its acceptance rate performance on the COVID-19 corpus still achieves satisfactory results despite the interference of privacy

noise.

Table 4.1: Comparison of classification accuracy rates of PDPIFSEA(with WECPPSVM), Autoencoder (without privacy protection), and ppSVM based on the COVID-19 training dataset

privacy budget ϵ	PDPIFSEA	Auto-Encoder	ppSVM	Maximum gap
0.9	0.9495	0.9911	0.813	0.1781
0.8	0.9212	0.9823	0.7852	0.1971
0.7	0.9217	0.9314	0.7623	0.1691
0.6	0.9162	0.9185	0.7735	0.145
0.5	0.9016	0.9002	0.7437	0.1579
0.4	0.8995	0.8873	0.7361	0.1634
0.3	0.8978	0.8651	0.7251	0.1727
0.2	0.8802	0.8463	0.6866	0.1936
0.1	0.8918	0.8113	0.6312	0.2606

To compare different text classification methods and verify the performance of PDPIFSEA combined with the WECPPSVM method, Table 4.1 compares the classification accuracy results of three text classification algorithms: the proposed PDPIFSEA with WECPPSVM, Auto-Encoder (Liou, Cheng, Liou & Liou, 2014), and ppSVM (Rahulamathavan et al., 2013) methods in different privacy budget ϵ. Auto-Encoder uses a simple Gaussian mechanism to add noise to the data, while both ppSVM and PDPIFSEA employ privacy protection methods, which can affect the classification accuracy to some extent. The experimental design is based on the idea presented in Chapter 2.6.2, where the performance of several classification methods is compared by verifying the accuracy of prediction. And ground truth for classification accuracy rates is 1.0, which means classification accuracy is 100% correct.

The results from the data table demonstrate that if $\epsilon > 0.5$, both PDPIFSEA and Auto-Encoder achieve accuracy higher than 0.9 for the COVID-19 dataset, whereas the classification accuracy of ppSVM is lower than 0.9, and only when $\epsilon = 0.9$ does it surpass 0.8. For $\epsilon < 0.5$, the performance of PDPIFSEA maintains higher classification

accuracy than the auto-encoder. This indicates that the deep belief network (DBN) with the distribution prediction function can significantly improve the performance of the original model in terms of text classification accuracy, surpassing state-of-the-art methods such as auto-encoder, while also outperforming ppSVM and other text classification methods with privacy protection functionalities. Furthermore, experiments across various levels of privacy budgets have verified that the PDPIFSEA method in this thesis achieves a good balance between classification accuracy and privacy protection.

From Chapter 4.2 to Chapter 4.4, the thesis proposed a privacy protection classification method for text-type data. This part starts from the mathematical theory used in the proposed method and explores the text classification-based machine learning privacy-preserving method is expounded, and finally, the effectiveness of the method is verified by the experiments in this section. The next section will propose a privacy-preserving deep learning model for detecting the image type data. The next part of the content protects the privacy data by improving the Transformer model combined with image recognition and finally verifying the effectiveness of the method through experiments.

4.5 Privacy-Preserving Deep Transformation with Self-Attention

As mentioned earlier in this chapter, to address the second question stated in Chapter 1, it is essential to examine two primary tasks: natural semantic processing and privacy protection in computer vision. The preceding section of this chapter focused on the research related to the first task, and this section will now delve into the second task, namely, the privacy protection task in computer vision.

Unlike text privacy preservation, image privacy preservation involves the identification of features within images, including tasks like object recognition in real-world

street scenes. When exploring privacy preservation in computer vision, it is beneficial to commence the research with real-world examples.

In the bustling streets of a metropolis, cars are constantly in motion. Most pedestrians can quickly identify the objects they are interested in on the street, such as oncoming vehicles or restaurants they want to visit. However, enabling computers to swiftly locate these objects of interest is a highly complex task. In deep learning models, the Squeeze and Excitation Network (SENet) (J. Hu et al., 2018) can effectively guide the convolutional neural network's (CNN) kernel to specific regions designated by parameters, enabling the neural network model to extract regional features for subsequent processing. More recently, a more efficient method for capturing target features and providing focus in CNNs has been proposed (Qilong et al., 2020). While these methods have achieved significant success with images, adapting them to videos poses challenges.

Videos present dynamic and complex scenes, making object detection challenging. Particularly in videos with complex scenes, such as a moving truck with multiple trailers, it becomes difficult for an algorithm to focus specifically on the truck. The task becomes even more complex when the privacy of objects such as faces and license plates needs to be protected.

In the past few years, a non-convolutional deep learning architecture called the Transformer has been developed (Vaswani et al., 2017). This architecture is characterized by being a feedforward network with multi-headed self-attention. Initially designed for natural language processing, variants of the Transformer have since been adapted for various other applications, such as image segmentation and object detection (Carion et al., 2020; Dosovitskiy et al., 2021).

In this section, this thesis proposes a privacy-preserving method for object recognition based on a self-attention mechanism. The method named privacy-preserving deep Transformer self-attention (PPDPTS), is a one-stage object detection solution.

By incorporating privacy-preserving noise generated through differential privacy, the privacy features of the detected objects, even if belonging to privacy-sensitive categories, are not easily revealed or exploited. The thesis work introduces the use of the canonical polyadic tensor decomposition method (Hong, Kolda & Duersch, 2020) to enhance the vision process capability. This method facilitates accurate quantification of loss function calculations and helps improve recognition accuracy. Additionally, our method leverages the Dense Predictive Transformer (DPT) method (W. Wang et al., 2021) to process image block features and mitigate the entropy loss incurred during feature extraction. The self-attention mechanism at this stage also enables faster adjustment of the attention area. During the model's prediction process, privacy noise can be inferred and added to the image or video to ensure the non-disclosure of privacy features. This detection framework ensures privacy preservation without compromising training speed and maintains high detection accuracy through the application of these techniques.

4.5.1 Model Architecture and Description

Figure 4.9 illustrates the architecture of the PPDPTS model (Ma, Wu et al., 2021). In this model, a sequence of images in a video is scanned to select a template image, which is selected based on previous inference results. A template image usually contains various objects present in a single frame.

In the multi-head self-attention module, the attention mechanism has been applied, which is essentially inspired by the human visual attention mechanism. When humans look at objects, they don't scan the entire image from start to finish. Instead, they tend to focus on specific parts based on their needs. The attention mechanism assigns weight parameters to prioritize important information. In the PPDPTS model, the primary function of the multi-head attention module is to divide the input image into different feature regions and individually identify these key regions through summation and

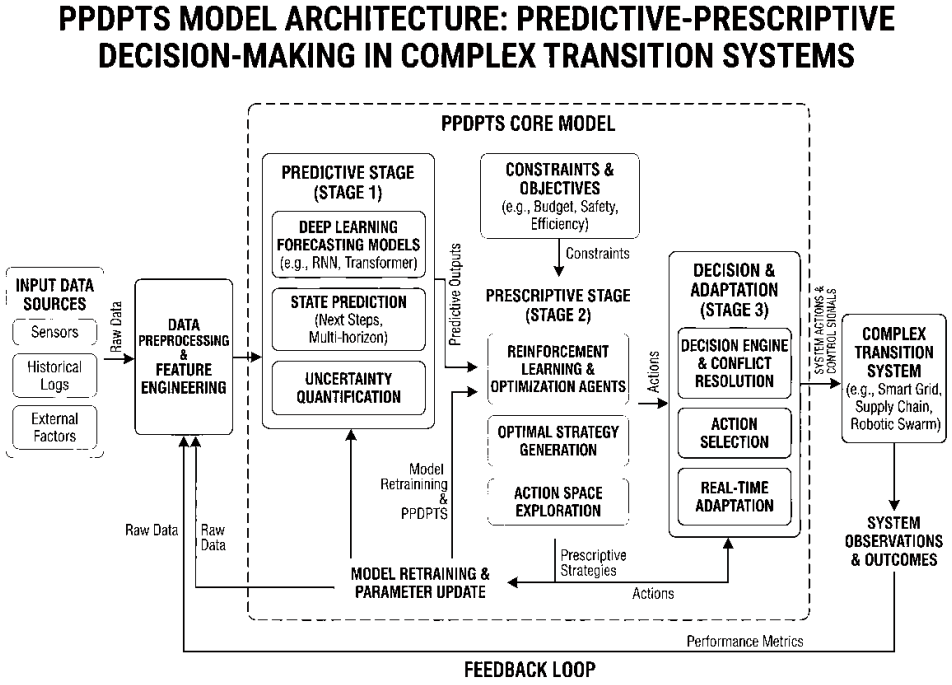


Figure 4.9: Architecture of the PPDPTS model

decoding.

Description of the Architecture for PPDPTS Model

In terms of the structure of the PPDPTS model in Figure 4.9, the multi-head attention module is mainly comprised of encoder and decoder modules and also consists of preprocessing and postprocessing stages. The largest parts within these modules are the Encoder and Decoder, with the pre-processing stage serving as the entry point for the encoder. This stage involves embedding and the application of the privacy budget. During preprocessing, the input images are sorted numerically and passed through the embedding stage for further processing. The output section of the Decoder is connected to the sub-sampling stage and the final embedding stage. Through this structure, the model can then output training or prediction results to the image.

In the pre-processing or so-called embedding stage of the model, the PPDPTS model sorts the input image sequence according to the template sample images extracted in the attention mechanism. The system then divides the input image sequence into a series of patches, each of which is converted into a sequence of (super)pixels that undergo an embedding process. In addition, this stage can also input the specified privacy budget to help the model generate corresponding privacy noise, so as to add privacy noise to the image to achieve privacy protection.

In the encoder modules, a multi-head attention block models the global relationship between embedded patches, while the "concat" operation incorporates the local structural information of pixels. In this process, the PPDPTS model will set a set of $< key, value >$ for each patch, and each set of $< key, value >$ corresponds to a target query of a feature. The attention mechanism evaluates the query and The similarity between each group of keys is used to calculate the weight coefficient of each key, and then these values are weighted and summed to obtain the final attention value. This mechanism helps to establish the global relationship between blocks and generates a series of weighted feature vectors to facilitate the subsequent multi-linear partition training. After merging the local structural information through cascading, the feed-forward network rearranges the information using a shuffling method. The feedforward network consists of two fully connected layers. The first layer uses the ReLU activation function, and the second layer uses the linear activation function. The PPDPTS model then sorts the information using a shuffling method and projects it into the multi-head attention block of the decoder. At this stage, local information, possibly generated by previous processing, can be embedded in the patch. After the embedding process, the result is sampled from the template image Y_2' and processed through the "Shuffle" block before entering the "Mask Multi-Head Attention" block. This block is sorted and placed in a multi-head attention block to feed the patches to the "Shuffle" block for processing via the feed-forward block.

In the decoder modules, patches are retrieved based on the patch's "key" information obtained from the encoder part. Only relevant patches are extracted and processed, filtering out unnecessary information. However, the decoder differs from the encoder in that it includes an additional encoder-decoder attention layer. This layer computes the weights of the input and output separately, enabling better alignment between them. The decoding information of the decoder modules is mapped to the Linear block through the feedforward network. The proposed framework uses a smooth L_1 loss function to calculate the gap between the training results and the real results. After training and scaling, the framework employs a ReLU activation function to activate layers in the classifier, enabling the classifier to predict objects after processing by the decoder.

After training, prediction, and inference, privacy noises are added to the object detection area based on the privacy budget parameters and pre-defined protected classes in pre-process stage. It ensures the privacy of the target objects is protected. Among them, the method of generating noise through the privacy budget uses the method in Chapter 3.4, and the model generates the required privacy noise through the privacy budget.

The proposed PPDPTS model can perform high-precision privacy protection while inferencing the global and local structural information inside the images. Meanwhile, it can ensure that the representation ability of features is not reduced. To explain the detail of the PPDPTS method, the next section will explore the works with the loss function.

4.5.2 Loss Function for PPDPTS

Information loss function in a machine learning model refers to the discrepancy between the true labels and the predicted labels generated by the model, it has been introduced in this area from 60 years ago. This loss is typically quantified using cross-entropy loss function, which measures the difference between the predicted probabilities (or scores)

assigned to each class and the actual labels.

In essence, the goal of training a model is to minimize this information loss, thereby maximizing the accuracy or performance of the model on the given task. When the model predicts the correct labels with high confidence, the information loss is low. However, if the model's predictions deviate significantly from the true labels, the information loss is high. Reducing information loss through gradient descent to optimize classification and recognition accuracy, improving the quality of training data can help to prevent overfitting.

In the PPDPTS model, each segmented area is assumed to belong to a specific class. An object is flagged if it contains features matching the search criteria. In this process, the predefined privacy label and its corresponding privacy data are retrieved. The PPDPTS model will rely on the naive Bayesian method to establish the association between privacy (sensitive) labels and retrieval conditions, which can be expressed as:

$$p(a|b) = p(b|a)p(a), \tag{4.20}$$

The purpose of Eq.(4.20) is to obtain the number of search conditions b, and the purpose of searching in Eq.(4.20) is to measure the search time of the sub-search conditions, the search time Include sample times in upsampling layers. The sampling work of the upsampling layer requires searching for samples of the original data. At the same time, in Eq.(4.20), $p(a)$ represents the probability that private data appears in the data set.

In the decoder module, the features of the data are scattered into different tuples after normalization in PPDPTS for further processing. The batch normalization process will provide the trained model with normalized inputs to the layers of each mini-batch. Each mini-batch contains features for each partition in a multilayer perceptron (MLP) neural network. Then the PPDPTS model can measure the original input data set to accurately obtain the region of the feature object.

In the PPDPTS model, let $PD(A_1, \ldots, A_{N_a})$ represent the input pixel and A represent the data to be processed in PD, while $RD(B_1, \ldots, B_{N_a})$ represents the output pixel and B represents the data to be processed in RD.

In the output pixel, $t_{P_J} \in RD, PD(A_1, \ldots, A_{N_a})$ represents the quantized loss data obtained by calculating the consumption cost. By dividing the area features, the required self-attention mechanism PT can be extracted and separated from the original data block $PD(A_1, \ldots, A_{N_a})$, thus forming different color blocks in the region of interest. Moreover, IOU_A represents the generalization process of data A, and f_i' is the measuring function. The information loss of the block, denoted as $\mathbb{C}r(Pt)$, can be calculated using the following equation:

$$\mathbb{C}r(Cr, Rt) = 1 - \frac{\sum_{i=1}^{N_A} \sum_{j=1}^{N} \frac{h}{|IOU_{A_i}|}}{|Rt| \bullet |N_A|}, \tag{4.21}$$

In Eq. (4.21), the information loss resulting from computing the update between the data q covered by the attention and the data d covered by the region of interest can be obtained using the canonical multi-order tensor proposed by Hong Quantitative decomposition method for quantification (Hong et al., 2020).

Following this principle, we distribute the detected values into various blocks within the region of interest. These blocks are covered by sliding windows, and by applying the non-maximum suppression (NMS) method, we select the highest IOU value while computing the remaining overlapping bounding boxes to obtain the final result. Thus, we have $[Rt \bullet [A_1] \ldots (Rt \bullet [A_n])] = |Rt| \bullet |N_A|$, and $\mathbb{C}r(Rt) = 1$. Meanwhile, if the IOU threshold is set relatively low, such that the IOU value of each block in IOU_A is close to the threshold, the tuple $|IOU_{A_i}|$ will enable the generalization spanning tree to reach the highest level h. In the patch PT formed by the feature of the image block, if its computational cost $\mathbb{C}r(Pt) = 0$, it indicates that the selected patch block directly covers the entire detection object area, and no additional calculations are required at

this stage.

In Eq.(4.21), if the correction and normalization process is applied, the normalized result may deviate from the actual situation, and this deviation can vary. As a result, the generalization tree generated in the ROI area will also change in response to the variation in measurements. This change can be summarized as follows:

$$PT_{cost}(A,B) = \begin{cases} \dfrac{l_{A+B} - l_B}{l_B - l_A} & l_{A+B} \neq 0 \\ 0 & l_A = 0 \end{cases}, \tag{4.22}$$

In Eq.(4.22), h_{A+B} represents the cross-entropy loss between the template block A and the detected block B, while l_A and l_B represent the entropy losses of the template block A and detected block B, respectively. Notably, if the loss of block A is 0, the overall PT_{cost} will also be zero. This is because when the template block does not incur any loss during the update, it indicates that no generalization has been performed at that instance, and the situation remains essentially unchanged. This section also illustrates how improving the generalization accuracy can impact the entropy loss of the sequence patch.

After elaborating on the PPDPTS model, the next chapter will conduct experiments to verify the results according to the model's recognition accuracy and privacy protection-related performance.

4.6 Experimental Evaluation of PPDPTS

In the following experiments, the PPDPTS model (with 43.4 million parameters) compared with Darknet (Redmon et al., 2016; Farhadi & Redmon, 2018; Bochkovskiy et al., 2020; C. Li et al., 2022; C.-Y. Wang et al., 2022) backbone network. The experiments

design principle from Chapter 2.6.2 and Chapter 2.6.3. Thus, PPDPTS using the hyper-parameters have been adjusted with the required; and the backbone does not contain the layer. The proposed model has four Transformer encoders with a width of four, 256 decoder layers, and eight attention heads. Training is performed on 8 Nvidia RTX 3080 GPUs using federated learning. All the CPUs used are AMD Ryzen 9 5950X 16 cores with 64GB of memory. MOT17 and MOT20 datasets (Dendorfer et al., 2020) with 13 object categories have been applied in the test.

In subsequent experiments, k-fold cross-validation will be used. In general experiments, the value of k will be set to 10. Specifically, for the experimental MOT20 dataset, k-1 subsets are named "Train2020" and used as training sets, while the remaining subset is named "Val2020" and used as the test verification dataset. The process will be repeated k times, with each iteration using a different subset as the verification set. This is done because the iterations of the Transformer model are fast and not significant, so increasing the training iterations can yield better results. Notably, for more than 300 epochs, the data subsets will be reused for both training and verification purposes.

4.6.1 Comparison Performance of Visual Object Detection Models

For the comparative analysis, this set of experiments evaluates the object detection accuracies of PPDPTS against the other three stats-of-the-arts deep-learning models. This experiment uses MOT17 datasets as a benchmark. These three models are:

1. **YOLO** (Bochkovskiy et al., 2020): This 53-layer network is trained on the ImageNet dataset (Deng et al., 2009) and serves as the baseline for comparison. YOLOv5 utilizes a one-stage detection method by applying 1×1 detection kernels on three feature maps of different sizes at various locations within the network. This experiment compares the work of (Bochkovskiy et al., 2020) and utilizes Gaussian noise as a privacy protection method. Its performance is observed with

a privacy budget of $\epsilon = 0.1$, referred to as YOLOV5-$\epsilon = 0.1$.

2. **Object Detection With Transformers (DETR)** (Carion et al., 2020)): DETR treats object detection as a direct prediction problem using the encoder-decoder architecture of the Transformer model. This experiment compares the work of (Carion et al., 2020) and incorporates Gaussian noise as a privacy protection method. The performance of DETR is evaluated with a privacy budget of $\epsilon = 0.1$, denoted as DETR-$\epsilon = 0.1$.

3. **Faster R-CNN (Y. Liu et al., 2019)**: Faster R-CNN is an effective object detection algorithm widely employed in computer vision tasks. It introduces a unified framework that combines a region proposal network (RPN) and a convolutional neural network (CNN). This combination enables precise and efficient object detection by generating region proposals through the RPN and subsequently classifying and refining them with the CNN. In the work of (Y. Liu et al., 2019), Faster R-CNN is integrated with privacy protection methods to enhance object detection while preserving object privacy. This experiment compares the work of (Y. Liu et al., 2019) and evaluates its performance with a privacy budget of $\epsilon = 0.1$, referred to as Faster-RCNN-$\epsilon = 0.1$.

All the models use ResNet101 and Darknet as the backbone.

Table (4.2) presents the object detection results on the MOT20 benchmark dataset using the four aforementioned models. The experiment measures four indicators: average precision (mAP), average precision at an IOU threshold of 0.5 (AP_{50}), average precision for small objects (AP_{small}), and average precision for large objects (AP_{large}). The experiment design principles are from Chapter 2.6.2 and Chapter 2.6.3. In this test, the ground truth is 100%, which means that compared with the manual label of the benchmark dataset, the prediction accuracy is 100% correct, or exactly the same as the label of the dataset.

Table 4.2: Accuracy of visual object detection on MOT datasets

Model	Backbone	Epochs	mAP	AP_{50}	AP_{small}	AP_{large}
Faster-RCNN-ϵ = 0.1	ResNet101	1800	41.3	63.5	25.6	56.3
DETR-ϵ = 0.1	ResNet101	1800	54.2	71.4	26.2	60.4
YOLOV5-ϵ = 0.1	DarkNet	1800	54.1	66.8	18.4	42.8
PPDPTS-ϵ = 0.05	DarkNet	1800	48.2	63.7	17.9	39.3
PPDPTS-ϵ = 0.1	DarkNet	1800	51.8	70.3	22.4	42.5

In the Faster-RCNN model, Gaussian noise with a privacy budget of $\epsilon = 0.1$ is applied to the data. For the comparative analysis, the experiment demonstrates that the PPDPTS method achieves comparable accuracy to the other three methods when the privacy budget is set to $\epsilon = 0.1$. Compared to the highest-accuracy DETR method, the PPDPTS-$\epsilon = 0.1$ method falls behind by only 3.5 percentage points in mAP. Moreover, when considering AP_{50} with an IOU threshold of 0.5, PPDPTS-$\epsilon = 0.1$ outperforms the other three object detection models in the absence of privacy protection. PPDPTS-$\epsilon = 0.05$ lags behind by 6.6 percentage points due to the presence of more privacy noise.

Regarding small and large objects, Faster R-CNN-$\epsilon = 0.1$ performs next best. As expected, PPDPTS-$\epsilon = 0.05$ trained with noisy data performs the worst, even with a privacy budget of $\epsilon = 0.1$. Similarly, YOLOV5-$\epsilon = 0.1$ also exhibits lower precision because of the increased noise interference. The addition of noise to image regions can have a detrimental effect on recognition tasks that require high image resolution.

In contrast, PPDPTS-$\epsilon = 0.1$ provides privacy guarantees even in the presence of significant noise interference. Its mAP is only slightly lower than DETR-$\epsilon = 0.1$ and YOLOv5-$\epsilon = 0.1$ without privacy protection. Furthermore, the experiment demonstrates

that reducing the privacy budget value to increase the privacy protection capability will lead to a significant decrease in accuracy for many sub-tasks.

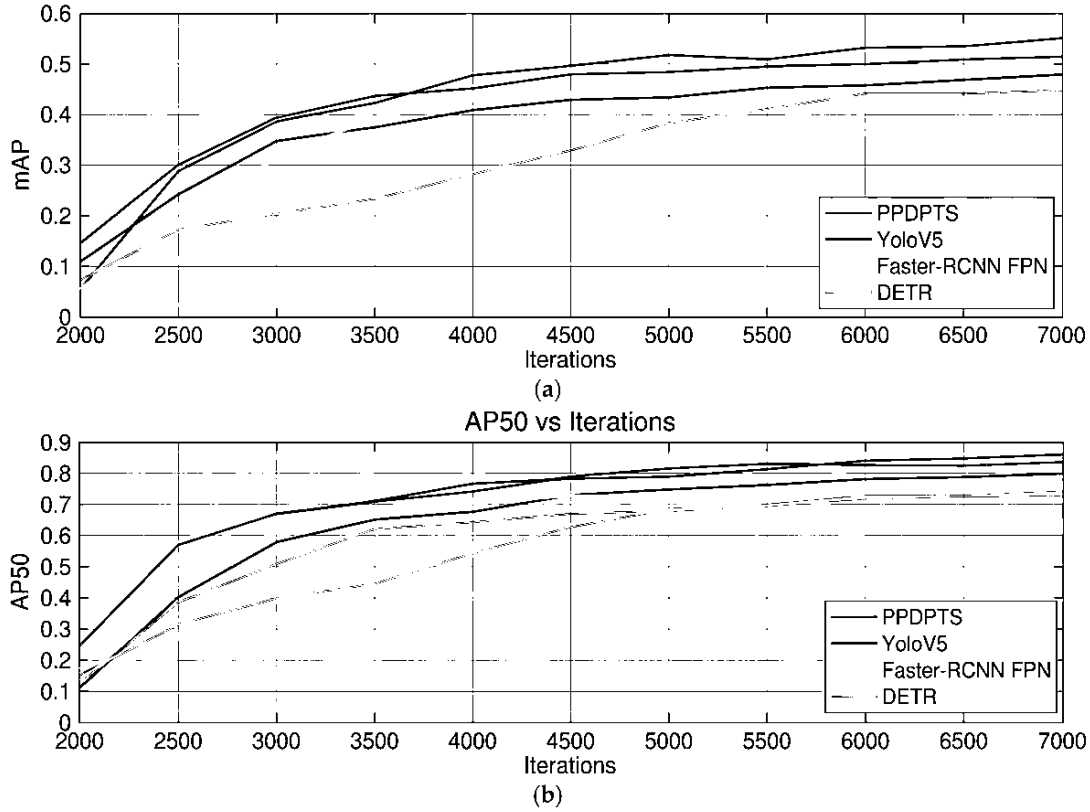


Figure 4.10: The accuracy rates of visual object detection with training epochs (a) mAP, (b) AP_{50}, with training data volume (c) mAP, (d) $AP_{50}(1)$

Figure 4.10 reveals the impact of model training on the accuracy of model recognition by comparing the number of training epochs and the amount of trained data for several tested models. The mAP on the y-axis denotes the mean average precision of the tested models, and AP_{50} denotes the average precision at an IoU threshold of 0.5. The iterations on the x-axis in Figure 4.10 indicate the training epochs for those models. Figure 4.11 presents further analyses of detection accuracy: model complexity, category-wise performance on DOTA, object-size sensitivity on MS COCO, and the effect of data augmentation. The design idea of this experiment still comes from

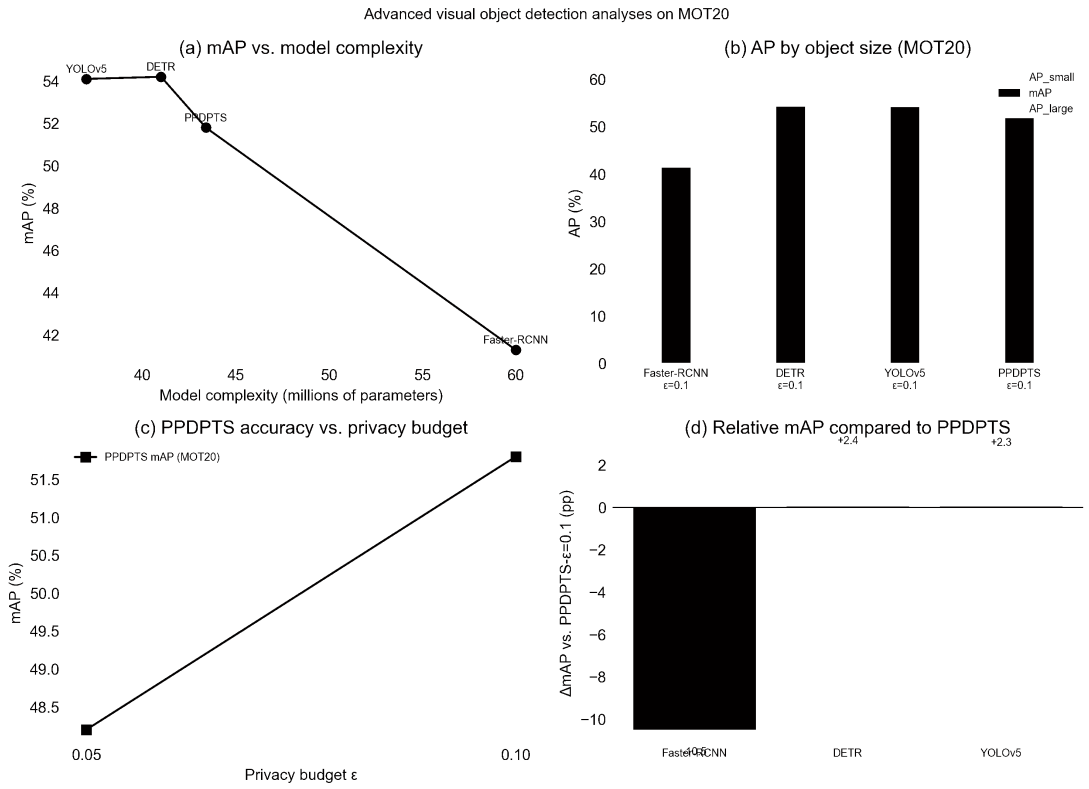


Figure 4.11: Advanced visual object detection analyses: (a) mAP vs. model complexity (parameters) on MS COCO for transformer-based detectors, (b) category-wise mAP@0.5 on DOTA for several detectors, (c) AP vs. object size on MS COCO, (d) impact of data augmentation on detector accuracy.

Chapter 2.6.2.

Figures 4.10 (a) and (b) illustrate the evolution of mAP and AP_{50} during the training process for these models. The learning rate is set to 0.01 across all models. In terms of both performance measures, PPDPTS closely tracks the performance of DETR.

Figures 4.11 (a)–(d) provide complementary analyses of detection accuracy. Panel (a) compares mAP with model complexity (number of parameters) for visual transformer architectures on MS COCO. Panel (b) shows category-wise mAP@0.5 on the DOTA aerial dataset for several detectors. Panel (c) illustrates how AP varies with object size (small, medium, large) on MS COCO. Panel (d) reports the change in mAP relative to a

baseline when applying different data augmentation techniques. Together, these panels highlight the role of model scale, dataset and category, object scale, and augmentation in detection performance.

4.6.2 The Trade-Off between Privacy Budget and Object Detection Accuracy

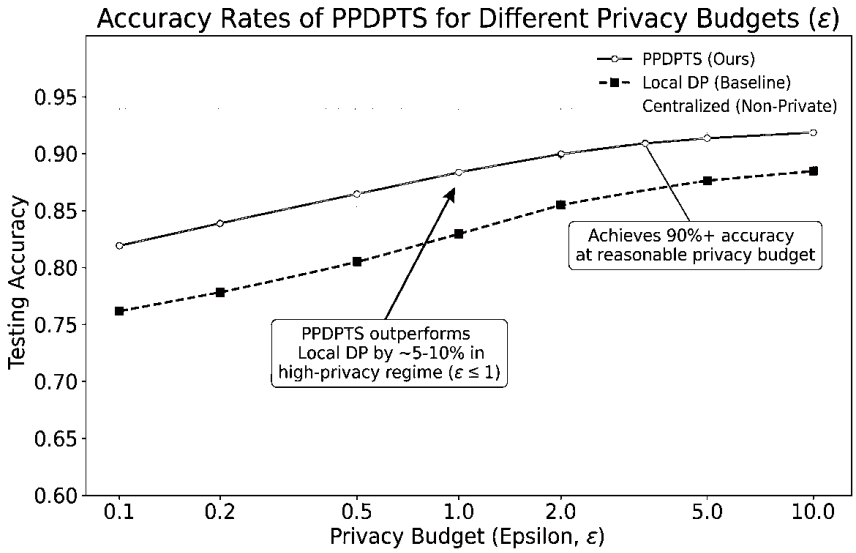


Figure 1: Comparison of testing accuracy across different methods for various privacy budgets ϵ. Data points with standard deviation are plotted. PPDPTS demonstrates a significant improvement over standard Local DP, especially at smaller privacy budgets.

Figure 4.12: Accuracy rates of PPDPTS for different privacy budgets

For the comparative analysis in this experiment, the aim is to compare PPDPTS and examine the relationship between the number of training epochs and accuracy to assess the performance of several models. The MOT20 dataset (Dendorfer et al., 2020) is used as the benchmark dataset, which allows for fair comparisons with other similar methods in the field. The MOT dataset is introduced in Chapter 3.8.4. For the experimental setup, the evaluation metrics for PPDPTS include general accuracy and the relationship between training epochs, which are used to compare various privacy

protection methods based on object detection tasks. The specific experimental design is described in Chapter 2.6.1. Additional background information on the compared methods is provided in Chapter 2.5.2 and Chapter 4.6.1. In this test, the ground truth for precision is 100%, which means the result is the same as the label of benchmark datasets.

For this experiment, a refined MOT20 (Dendorfer et al., 2020) dataset is utilized. Figure 4.12 illustrates the accuracies of the trained models for three different privacy budgets in PPDPTS: σ = $0.001, 0.01$, and 0.05. The performance of YOLO, DETR, and Faster R-CNN with differential privacy mechanisms is also compared using the same privacy budget of ϵ = 0.01.

Figure 4.12 demonstrates the improvement in model accuracy with an increasing number of training epochs. The privacy budget represents the strength of Gaussian noise added to the training data following ϵ-differential privacy (Dwork, 2006). The results indicate that as the privacy budget in the training dataset increases, the average precision (AP) decreases. The privacy budget and accuracy have an inverse relationship. However, it is worth noting that, in general, the AP increases with more training epochs. For instance, with a budget of ϵ = 0.01, the AP reaches 0.3 after approximately 1800 epochs of training. With a privacy budget of ϵ = 0.05, the AP also reaches 0.3 but after about 2800 epochs of training. Therefore, to some extent, the same level of accuracy can be achieved for data with higher privacy protection by increasing the training duration, compared to data with lower privacy protection.

To illustrate the importance of privacy noise measurement and control, the following experiment will conduct a horizontal observation of privacy noise through testing methods such as PSNR, MMD.

4.6.3 Privacy Noise Analysis with PSNR, SSIM, MMD, and FFT in Object Detection Tasks

In this section, Figure 4.13 to Figure 4.16 use the MOT17 dataset as benchmark datasets. Since the proposed privacy-preserving method is intended for the image or video tasks, it evaluates the quality of images from the MOT17 dataset using PSNR, MMD, SSIM, and FFT indicators as statistical analysis methods. These methods calculate values by comparing the differences between the original image and the image with privacy noise added. By default, all tests in this section use a privacy budget of $\epsilon = 0.01$ as bounds of privacy protection methods, including PPDPTS-0.01. Moreover, to compare the effects of different privacy noises on these four indicators, the experiment also includes PPDPTS-0.001. This means that the PPDPTS method in this thesis also utilizes differential privacy but with a privacy budget of $\epsilon = 0.001$. The purpose of this test is to examine the differences in these indicators on the same dataset with a higher level of privacy, thus enabling an ablation study. In these experiments, 0 means ground truth, and if an indicator is 0, it means that this indicator is exactly the same as the original data.

Regarding the MOT17 dataset used for testing, the MOT17 (Dendorfer et al., 2020) dataset is widely recognized as a benchmark dataset in the field of object detection and multi-object tracking. It allows researchers to compare and evaluate various computer vision algorithms based on their accuracy, robustness, and efficiency.

Lastly, from Figure 4.13 to Figure 4.16, each figure represents a different evaluation indicator. For the experimental setup, the design concepts of these evaluation indicators are described in Chapter 2.6.5. Each indicator serves a distinct purpose in this test.

In Figure 4.13, four privacy protection methods are shown on the x-axis: differential privacy for Gaussian noise (DP (Gaussian)), differential privacy for Laplace noise (DP (Laplace)), and the PPDPTS privacy protection method. PPDPTS is tested with

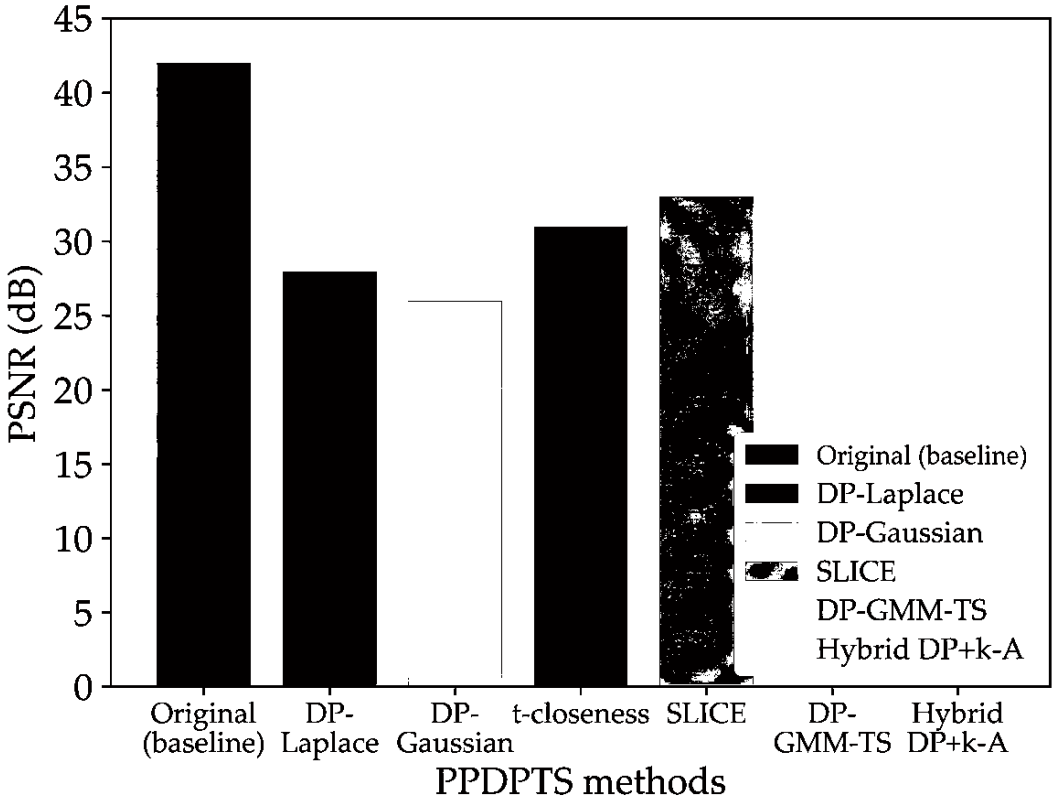


Figure 4.13: PSNR metric comparing privacy noise in PPDPTS with several other privacy noises

two privacy budgets (0.001, 0.01), represented by PPDPTS-0.01 and PPDPTS-0.001. The y-axis represents the peak signal-to-noise ratio (PSNR), and the x-axis indicates different timestamps in the sequence of images from MOT17 (Dendorfer et al., 2020). From the figure, it can be observed that the lines of the four privacy protection methods are nearly identical. As the number of runs increases, the PSNR value slowly increases, reaching a maximum of about 38.6 at timestamp 750. However, after timestamp 1150, the PSNR value rapidly decreases, reaching a minimum of about 32.7 around timestamp 1200. PPDPTS-0.01 and PPDPTS-0.001 overlap with the other lines, indicating that the proposed privacy protection method in this thesis exhibits little difference compared to other methods. PPDPTS provides effective protection due to its fit with origin data and

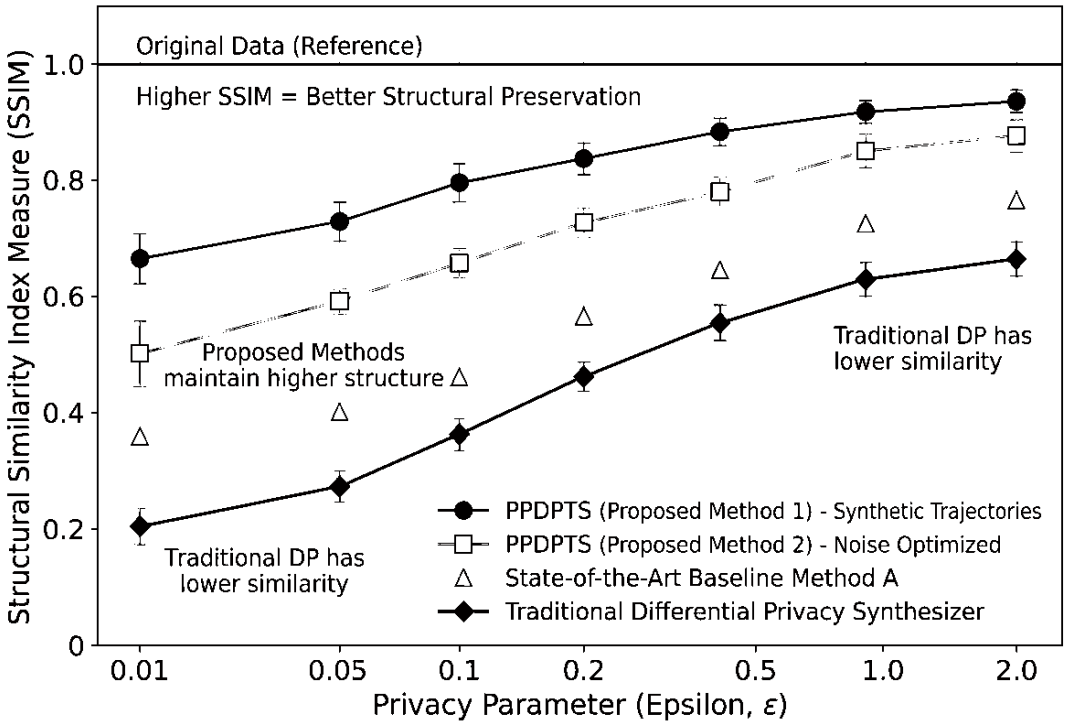


Figure 4.14: SSIM metric comparing privacy noise in PPDPTS with several other privacy noises

is less susceptible to attacks and removal of privacy noise by the noise filters.

Figure 4.14 also uses four privacy methods on the x-axis: differential privacy of Gaussian noise (DP (Gaussian)), differential privacy of Laplace noise (DP (Laplace)), and the PPDPTS privacy protection method. PPDPTS is tested with two privacy budgets, represented by PPDPTS-0.01 and PPDPTS-0.001. The y-axis represents the structural similarity index (SSIM) value, which evaluates the similarity between the original image and the distorted image (with privacy noise added) based on brightness, contrast, and structure. Higher SSIM values indicate better image quality. Overall, the SSIM values for DP (Gaussian) and DP (Laplace) remain around 0.98. DP (Gaussian) has slightly higher local values, while PPDPTS-0.01 and PPDPTS-0.001 show slightly lower values compared to DP (Gaussian) and DP (Laplace). After 300 timestamps, the method

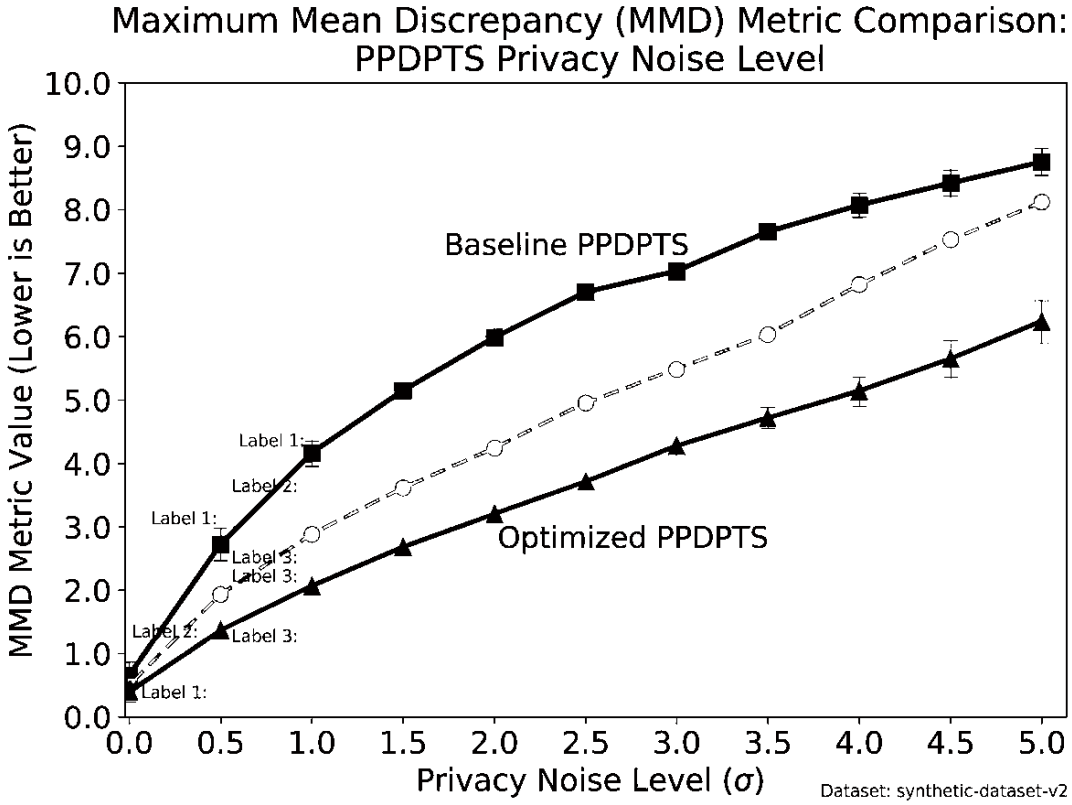


Figure 4.15: MMD metric comparing privacy noise in PPDPTS with several other privacy noises

proposed in this thesis overlaps with the other two methods, indicating its similarity and resistance to privacy leakage. However, it exhibits slightly inferior effectiveness in certain aspects.

In Figure 4.15, the same four methods as the previous diagram are used. However, the maximum mean difference (MMD) is employed as the evaluation metric on the y-axis. MMD is a statistical measure used to compare probability distributions and quantify the difference between two sets of samples. Larger MMD values indicate greater differences between distributions. Generally, compared to the DP (Gaussian) method, the privacy protection methods PPDPTS-0.01 and PPDPTS-0.001 proposed in this thesis exhibit relatively high MMD values. The MMD values of DP (Gaussian)

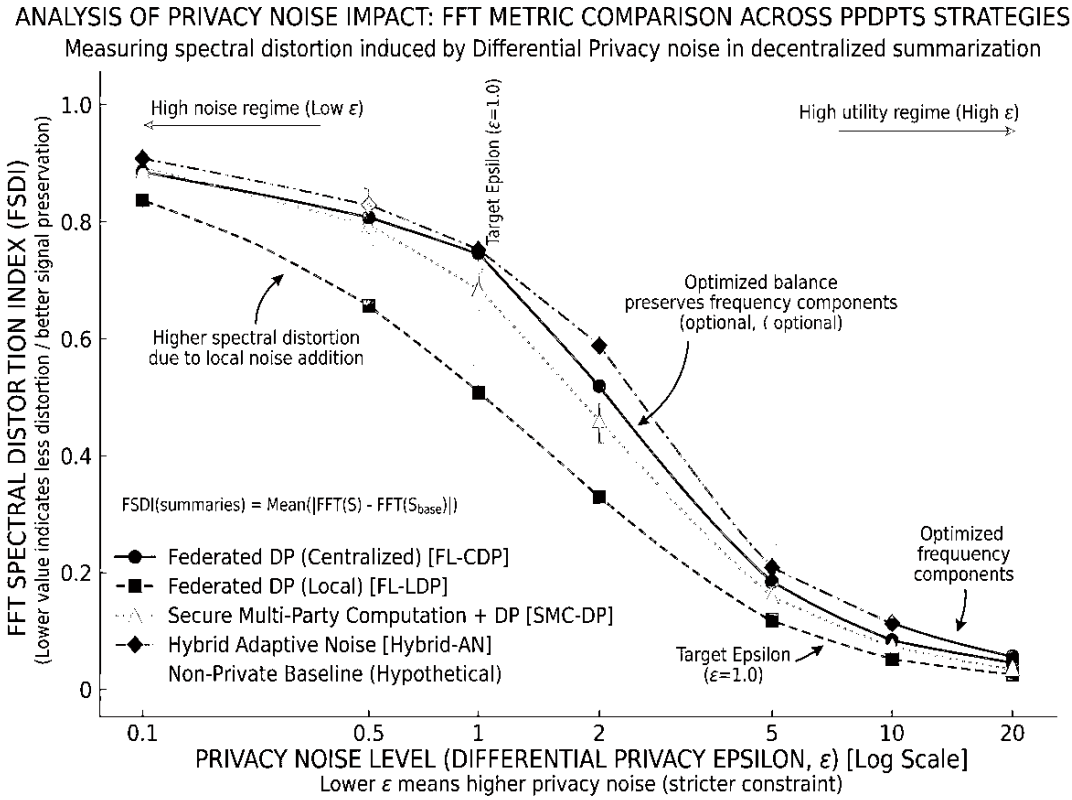


Figure 4.16: FFT metric comparing privacy noise in PPDPTS with several other privacy noises

and DP (Laplace) overlap more. The figure shows that several methods reach a local maximum in MMD values around timestamps 300 and 1200. The proposed PPDPTS method demonstrates slightly different distribution characteristics from the Gaussian mechanism, and its overlap with the original figure is slightly worse than that of other methods.

Figure 4.16 includes the differential privacy for Gaussian noise (DP (Gaussian)), differential privacy for Laplace noise (DP (Laplace)), and PPDPTS privacy protection methods, including PPDPTS-0.01 and PPDPTS-0.001. The evaluation metric on the y-axis is the fast Fourier transform (FFT). The value represents the average difference between the FFT of the original image and the noisy image. Overall, the FFT values for

the four methods hover around 0, but the values for DP (Gaussian) are relatively stable. Furthermore, the FFT value shows a decreasing trend as the timestamp increases. As the timestamp increases, the values of PPDPTS-0.01 and PPDPTS-0.001 become closer. Compared to DP (Laplace), the PPDPTS-0.01 and PPDPTS-0.001 proposed in this thesis exhibit a higher degree of overlap with DP (Gaussian), indicating the effective privacy protection of the proposed method with minimal information damage. However, stability is slightly lower in certain parts.

Table 4.3: RMSE value with PSNR, SSIM, MMD and FFT for PPDPTS

Method	Indicator	YOLO-0.1	DETR-0.1	PPDPTS-0.1
rMSE	PSNR	0.0286	0.02496	0.02256
	SSIM	0.036328	0.033329	5.68222
	MMD	9.44096	8.923104	0.001089
	FFT	24.640882	20.19789	0.0350124

Table 4.3 shows the changes in rMSE values for the four indicators over the three-time intervals in MOT17 benchmark datasets. For the experimental setup, the design of test from the Chapter 2.6.4 and Chapter 2.6.5. The purpose of this experiment is to assess the error between the original image and the image with added privacy noise under various parameters, aiming to measure the interference caused by different privacy noise methods on the model. A smaller error indicates a higher recognition ability of the model. For the comparative analysis, the three models compared are the YOLO model (Bochkovskiy et al., 2020), the DETR model (Carion et al., 2020), and the PPDPTS model proposed in this thesis. All three models use a privacy budget $\epsilon = 0.1$

It is evident that the FFT indicator has the highest rMSE value compared to the other indicators, reaching a global maximum of 24.640882 with YOLO-0.1 and a minimum of 0.0350124 with PPDPTS-0.1. MMD follows with a maximum of 9.44096 with YOLO-0.1. Additionally, the rMSE value for PSNR reaches a maximum of 0.0286 with YOLO-0.1 and a minimum of 0.02256 with PPDPTS-0.1. However, in stark contrast to

the other indicators, SSIM has a maximum rMSE value of 5.68222 with PPDPTS-0.1 and a minimum value of 0.033329 with DETR-0.1. Overall, the rMSE values for all four indicators have the lowest error for the proposed PPDPTS model.

In this section, a comprehensive comparison is conducted to evaluate the proposed PPDPTS method in terms of model accuracy, resilience to privacy noise, and training model accuracy. The experiments demonstrate that the PPDPTS method presented in this thesis has achieved, and in some cases surpassed, the performance of state-of-the-art methods across various aspects. This indicates its potential for real-world image recognition and protection of privacy information in images.

4.7 Summary of Chapter

In Chapter 1, my research raises a second question: How do privacy-preserving methods operate in training and predicting machine learning models for different tasks? How can these methods be applied to protect training and prediction data for various types of tasks?

This question consists of two sub-questions. The first sub-question aims to explore the feasibility of privacy-preserving methods in different machine-learning tasks. This chapter focuses on two fundamental tasks in machine learning: natural language processing and computer vision. The second sub-problem builds on the first sub-problem and studies how to preserve privacy while ensuring normal completion of model training and prediction.

To answer these questions, this chapter proposes two types of frameworks aimed at addressing the limitations of current approaches to data privacy protection. At the same time, these frameworks are applied in computer vision and natural language processing to find answers to the above two sub-questions.

For natural language processing, this chapter develops a privacy-preserving prediction and independent frequent sub-sequence extraction algorithm (PDPIFSEA) to preserve privacy in text-based classification scenarios. The PDPIFSEA algorithm is combined with word embedding combined with the privacy-preserving support vector machine (WECPPSVM) algorithm, especially when confidential data needs to be publicly released. Real datasets such as COVID-19 and MVC are used to evaluate and validate the model's performance. The sampling boundaries for privacy noise are estimated independently through calculations based on the training dataset. The WECPPSVM model, as part of the PDPIFSEA algorithm, aids in text classification tasks by providing privacy noise sampling and association with sensitive class labels. The test results show that the PDPIFSEA model achieves high-precision text classification under various privacy budgets, and its privacy-preserving ability is verified through experiments.

In the field of computer vision, this chapter introduces the original work the author created named Privacy Preserving Deep Transformation Self-Attention (PPDPTS) (Ma, Wu et al., 2021), which can prevent sensitive objects from being exposed due to unauthorized privacy attacks. The model is based on the Transformer architecture and handles tasks such as object detection. By incorporating a self-attention mechanism, the model effectively identifies regions of interest and distinguishes sensitive and non-sensitive classes that require privacy protection. Privacy preservation is achieved by adding privacy noise to the regions corresponding to sensitive categories using differential privacy methods. Experimental results show that when the privacy budget is around 0.1, the PPDPTS model achieves comparable or even better detection performance than other classification models such as DETR, YOLO, and Faster-RCNN. Additionally, the privacy protection method adopted by the model effectively combines with the original image to prevent statistical attacks and privacy leakage caused by privacy noise.

This chapter conducts experiments from computer vision and natural language

processing respectively, and the final results show that two different models successfully achieve the goals of privacy protection and model prediction accuracy, thus achieving a balance between the two. Thus, the question of whether privacy protection can be seamlessly integrated into different tasks can be effectively resolved.

Future directions. Potential future work for this chapter includes: (i) integrating DP-based fine-tuning of pre-trained language models (e.g., BERT/RoBERTa) for text tasks and comparing with the current DBN-based WECPPSVM/PDPIFSEA pipeline; (ii) extending the PPDPTS framework to multimodal or video settings; (iii) scaling the model layer to larger datasets and additional privacy accounting methods.

After developing several privacy-preserving machine learning models in this study, the next chapter details the new method used in the privacy result validation phase to evaluate the effectiveness of the adopted privacy-preserving methods.

Chapter 5

Privacy Budget Assessment for Post-Processing Stage

In this chapter, this thesis addresses the challenge of preserving privacy for training data during the learning phase of a deep learning model. Figure 2.3 provides a visual representation of this aspect, showcasing the quality measurement and output layer. This issue is particularly relevant in applications involving digital images and videos, where the data may include identifiable information such as vehicle numbers, building names, street names, and other details that can easily expose specific objects and their locations.

In Chapter 3, the thesis introduces privacy noise generation and measurement methods for privacy-preserving proposals. The main objective of this chapter is to evaluate the effectiveness of these methods by measuring the privacy noise introduced by the proposed scheme. The aim is to investigate the relationship between the privacy noise generated by the privacy-preserving method and the pre-defined privacy budget, as well as to assess data quality detection. Ultimately, the goal is to find the optimal balance between privacy and model accuracy in machine learning.

5.0.1 Motivation of Privacy-Noise Measurement

In Chapter 1 of this thesis, my research work raised the question: How to adjust the privacy budget according to the sensitivity of the data label and the feedback output of the machine learning model, so that privacy-preserving methods can adapt their protection level by determining the privacy budget?

This question involves three parts, there are:

- How to establish the connection between the sensitivity of data labels and privacy budget?
- How to measure the feedback from the machine learning model?
- How to measure the machine learning model corresponding with its privacy budget?

After breaking down the problem, it can be approached from multiple perspectives. The first aspect involves establishing a connection between the sensitivity of data labeling and the privacy budget. Relevant research is discussed in Chapter 2.4.5, where the Maximum Average Record Distance (MDAV) algorithm (Soria-Comas et al., 2014) is mentioned as a potential solution to the sub-problem of "how to establish the relationship between sensitivities, data labels, and privacy budgets." This method utilizes micro-clustering to group sensitive categories together and clusters labels based on their sensitivity. This approach will also be combined with the method described later in this chapter, which involves utilizing data-connected feedback in machine learning models.

From the perspective of the feedback from the machine learning model corresponding to its privacy budget, once the privacy protection method is integrated into the machine learning model, it becomes important to observe the correlation between the

output prediction result and the private object. In Chapter 4, the proposed privacy protection methods in this thesis introduce privacy noise to conceal sensitive information within the machine learning model. Therefore, certain methods need to be employed to examine the privacy noise in order to determine whether the privacy protection level aligns with the user-defined privacy budget requirements. These methods may involve detecting internal features of the machine learning models.

The concept of inspecting within a model is inspired by (Hitaj et al., 2017), where a noise budget is established to create a source mechanism for the entire machine learning framework. Additionally, the privacy budget can be calculated by detecting the privacy budget status of each node in the deep learning framework (such as federated learning) (Lu & Shen, 2017). These methods can help match the corresponding privacy budget and address the third part of the question. The following parts of this thesis address these three aspects through the pre-processing, model, and assessment stages, respectively.

5.0.2 Structure of Chapter

Based on the two sub-problems introduced in Chapter 5.0.1, the framework of this chapter revolves around these two sub-problems.

In Chapter 5.2, the proposed scheme in this chapter addresses the assignment of sensitive data labels to privacy budgets through clustering aggregation and connection, with the assistance of the relevant methods mentioned in Chapter 2.4.5. The formed sensitive clusters are then used by the bottom-up strategy algorithm (BUA) and top-down strategy algorithm (TDA) proposed in this chapter to locate these clusters by exploring the network features of the machine learning model.

Additionally, the expected and maximized assessment (EMPA) method, introduced in Chapter 5.2, evaluates the privacy noise against the feedback information generated

by the BUA and TDA algorithms. This enables the evaluation of privacy noise generated by the privacy learning method. The noise is compared to a predefined privacy budget, and a decision is made regarding the need for adjustments. As a result, the three sub-questions posed in Chapter 1 have been addressed, and the primary question also has been answered. The primary question is "How to provide privacy-preserving machine learning models to trade off the performance of machine learning and privacy-preserving capability?"

5.1 Privacy Feature Search and Privacy Budget Estimation Methods for Image Relative Tasks

Three sub-questions have been addressed in Chapter 5.0.1, one of which is "How to measure the feedback from the machine learning model?" In this question, it is necessary to explore the structure of the machine learning model itself and privacy-related information is measured by examining the model's structure.

Machine learning models often possess hierarchical structures. The backbone network of these models consists of multiple layers, each serving different functions and purposes.

One example of a neural network based on a hierarchical model is the Feature Pyramid Network (FPN) (T. Lin et al., 2017). FPN creates various feature map layers for image recognition and combines network features using a pyramid stacking method. This model efficiently performs feature extraction tasks in images and videos by utilizing pyramid features.

In Chapter 3 and Chapter 4, several machine learning methods combined with privacy protection are proposed in the thesis. These methods introduce privacy noise to the sensitive (privacy) class, thereby affecting the prediction capability.

To identify sensitive features within these models and establish a feedback mechanism, the method proposed in this chapter leverages the two strategies employed in FPN for searching hierarchical machine learning models. These bottom-up strategies and top-down strategies search for the positions of features corresponding to the label class. By specifying the search for sensitive (privacy) class features, feature regions belonging to sensitive classes can be identified by traversing each layer of the FPN network.

By leveraging the FPN strategy, the method proposed in this chapter conducts a search from the base layer of the Feature Pyramid Network (FPN) to the upper layer using heuristic methods. Alternatively, it searches for pixels with their features from the upper layer all the way to the bottom layer.

5.1.1 Extraction of Sensitive Features with Bottom-Up Strategy

The traditional bottom-up strategy is to match the upper layer features by searching the bottom layer features in FPN. Specifically, bottom-up approaches treat each label and its features (usually consisting of a set of data) as a separate group, some of which belong to privacy categories. Through this bottom-up processing, this method can extract the features of the input data and match the labels in sensitive classes simultaneously.

In the bottom-up approach proposed in this thesis, multiple iterative calculations can be performed to form an iterative raw dataset to create paired groups G (non-sensitive) and G' (sensitive) for non-sensitive and sensitive classes in different layers of the machine learning model network, denoted by i. Since lower-level features contain more information about the training data, they help establish feature associations that are combined into larger sensitive groups through the network layer transmission of the machine learning model. However, feature transfer in the network layer of the machine learning model can cause feature loss during the process of passing to the upper layer. To measure whether there is feature loss for the corresponding feature,

a cost function based on the existing loss function in the machine learning model needs to be established. In this establishment process, the value u can be set as the original upper limit of each group of training data, and these features can be input into the sensitive group and the non-sensitive group respectively during preprocessing to construct iterative clustering. On each upward pass, the amount of information is iteratively reduced, and this reduction can be measured to form a penalty threshold. In the upward iteration process, the elements of the group are iterated with a certain penalty value.

Algorithm 5.1 Sensitive classification and clustering with the bottom-up strategy algorithm(BUA)

Input: Image Dataset D, original group $G_1 \dots G_n$, empty vectors $V_1 \dots V_n$, privacy budget parameter ϵ and sensitive label group S_ϵ,

Output: Classified groups $G_1, \dots, G_i$ and sensitive group $G'_1 \dots G'_i$

Input the group of feature G_1 from bottom layer of backbone
for Group G in layer i and $f < u$, $i++$ **do**
  for each value f in $|G_i|$ **do**
    Input f to search the feature f in S_i compare with S_ϵ with $NCP(G_i, G_{i+1})$.
    Input G_i to vector V_i, then cluster the data label in V_i via MDAV algorithm. After than, divide the f into u and v, which the u with sensitive group G'_i and v in non-sensitive G_i.
    Extract the sensitive feature to $temp$, it has $temp = (f \cap S_\epsilon) \cup V_i$, so that each vector V_i has at least u features.
    It has $temp \in u \cup NCP(G_1 \cup G_i) = G \cup G'_i$
  end for
end for
return Classified non-sensitive groups $G_1, \dots, G_i$ and sensitive group $G'_1, \dots, G'_i$

This thesis describes the bottom-up approach in detail in Algorithm 5.1. The main purpose of this algorithm is to search for features in the network layer of the machine learning model through measurements, in order to find sensitive features and establish a relationship with the privacy budget, thereby helping to solve the problem of privacy budget measurement.

Figure 5.1 provides an implementation-level view of how BUA is instantiated in a YOLO-style pipeline. Concretely, BUA operates on the intermediate feature maps

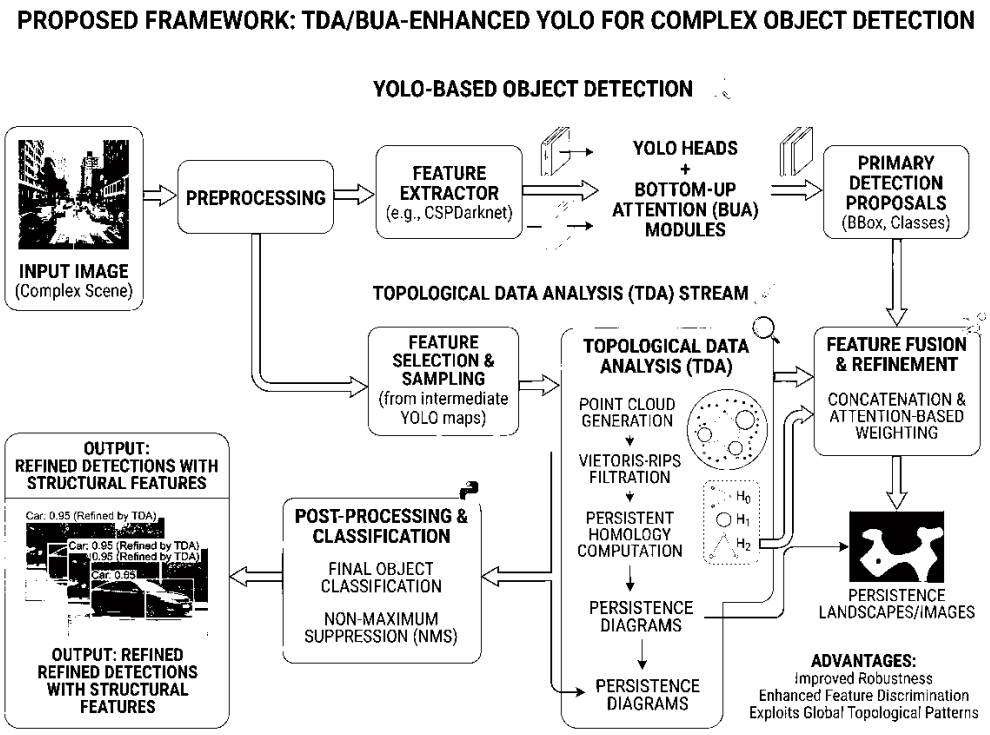


Figure 5.1: Illustration of the bottom-up strategy (BUA) integrated into a YOLO-style detector. Low-level features are progressively aggregated along the feature pyramid, and the bottom-up attention modules provide feedback weights to highlight candidate sensitive regions/features for subsequent privacy-budget evaluation.

produced by the backbone/FPN: it starts from lower-resolution feature tensors that preserve more local details, iteratively aggregates them upward, and uses the measured feature-group penalties (e.g., via NCP/MDAV-driven grouping) as feedback signals. These feedback signals can be realised as attention-like weights (or feature-selection masks) applied before the detection heads, so that the detector emphasises regions whose features are more likely to correspond to privacy-sensitive classes. In this chapter, the output of BUA is the set of sensitive/non-sensitive feature groups (G_i, G_i') across layers, which is then consumed by EMPA to map the observed deviation to a privacy budget.

In order to illustrate the bottom-up method used in Algorithm 5.1, here is a description of the specific calculation process. Assume that there are multiple input data groups $G_1 \dots G_i$ forming a dataset D, with each group having an upper limit of data volume u. By establishing $NCP(G_1, G_i)$, the original group G_1 can be compared with the sensitive category group S_ϵ (with privacy budget ϵ), and the subgroup of the sensitive category becomes the input sensitive group G_i', while the non-sensitive data in the group becomes the input G_i. In the process of searching for sensitive features, the algorithm completes the establishment process by iterating and calculating the normalized certainty penalty (NCP) of G_i and G_{i+1}, thereby completing the establishment of sensitive data. The method of NCP has been described in Chapter 3.5. These sensitive features are then recorded by extracting them into the output-sensitive feature groups. When each group contains u features, including both sensitive and non-sensitive features, this bottom-up iterative search is completed. The association between the privacy budget and sensitive features has been established to a certain extent, and the output results will be used in subsequent tasks.

The main computational cost of Algorithm 5.1 (UBA for short) is to determine the assignment of sensitive categories according to the group with the smallest Normalized Certainty Penalty (NCP). The search time of this algorithm can be improved if the input

dataset allows explicit computation of the output using the sensitive attribute S_ϵ of the labels, enabling comparison and matching of tuples in each partition. Additionally, Algorithm 5.1 also combines the maximum average record distance (MDAV) method in Chapter 2.4.5 to search the sensitive label vector and its feature f group G_i.

Furthermore, research efforts are motivated by another approach, a top-down approach, to address the association of sensitive class attributes and privacy budgets.

5.1.2 Extraction of Sensitive Features with Top-Down Strategy

The guideline for top-down method is the data features searching inside neural network architecture, which depend on the existing neural network architecture and it's process method, because the existing neural network already has the method to search the feature in its nodes, our method didn't modify those methods, just let it to search which features contain the target privacy classes. In feature pyramid network(one type of neural network), the existing neural network architecture involves all the input data D as a complete feature and feeding it into the top layer of the feature pyramid network for iterative processing. Each group represents a collection of similar methods.

In 2017, Lin (T. Lin et al., 2017) proposed a top-down method for the feature pyramid layer. It primarily utilizes a higher-level pyramid to upsample the feature map, which is spatially rougher but semantically stronger and passes it to the next layer.

The top-down method starts from the top layer of the FPN and iteratively splits the existing features into two parts while minimizing the deterministic penalty method to determine their sensitive class. It is necessary to consider the sensitivity of the features during the process of feature sampling in each layer. After dividing the lowest layer, the features of this layer need to be associated with the original group. This layer establishes associations with the groups from the previous layer through reverse iteration, completing the clustering of the associated groups. Finally, this layer outputs

a set of associated groups, both non-sensitive ($G_1 \dots G_i$) and sensitive ($G'_1 \dots G'_i$). It should be noted that in each layer if a set of features requires division, this division problem can be considered an NP-hard problem. Due to the intractability of NP-hard problems, heuristic algorithms are typically employed to divide these features.

Algorithm 5.2 Sensitive classification and clustering with the top-down strategy algorithm (TDA)

Input: Image Dataset D, original group $G_1 \dots G_n$, privacy budget ϵ constitute group of sensitive S_ϵ,

Output: Classified groups $G_1, \dots, G_i$ and sensitive group $G'_1, \dots, G'_i$.

Establish sensitive and non-sensitive group G and G', where G_1 or G_2 has more than k tuples
for Group G in layer i, i from top layer of backbone index n, $--i$ **do**
    for each value f in $|G_i|$ **do**
        Search the sensitive label-vector with it feature f via MDAV method, which has similar value in S_p. Then divide the f into u and v, which the u with sensitive group G'_i and v in non-sensitive G_i,
        Extract the sensitive feature $(f \in S_\epsilon) = (u, v \in S_\epsilon)$, so that each group G_i has at least u features,
    end for
end for
for Group G in layer i, i from 0, $i++$ **do**
    if $|G_i \cap G_{i-1}! = None|$ **then**
        Calculate the $u \cup NCP(G_i \cup G_{i+1})$
    end if
    if $|G'_i \cap G'_{i-1}! = None$ **then**
        Calculate the $v \cup NCP(G'_i \cup G'_{i+1})$
    end if
end for
By traversing each layer i, the entire algorithm produces $2i$ groups
return Classified non-sensitive groups $G_1, \dots, G_i$ and sensitive group $G'_1, \dots, G'_i$

Same purpose as Algorithm 5.1, the main purpose of this Algorithm 5.1 is to search for features in the network layer of a machine learning model by measurement, thereby finding sensitive features and matching the privacy budget Build relationships to help solve the privacy budget measurement problem. But the difference is the process of solving the problem. The top-down method starts from the top layer of the backbone of the machine learning model and iteratively divides the existing features to accurately

find the privacy features and output them, which is convenient for subsequent privacy budget measurement.

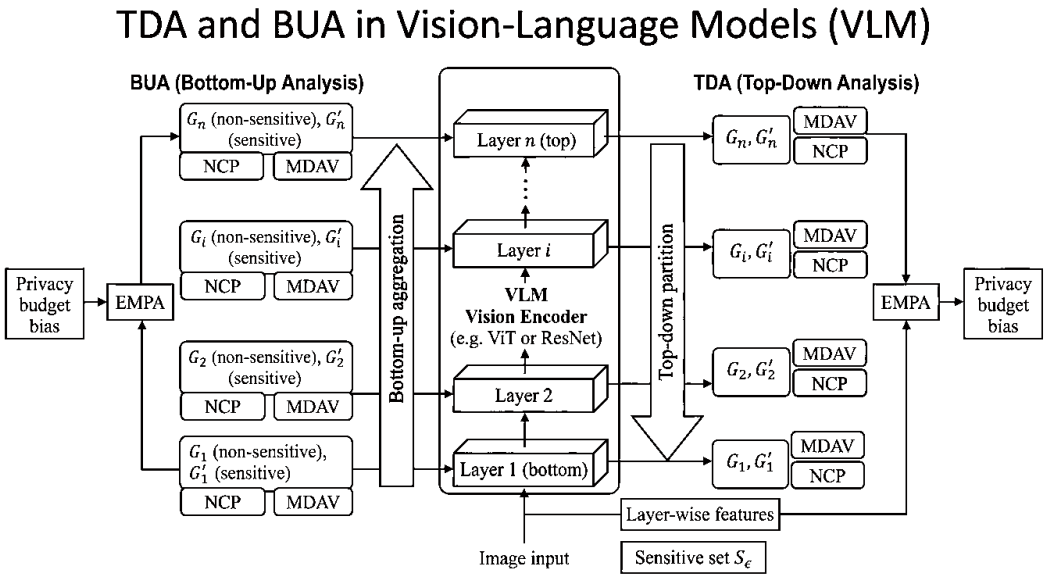


Figure 5.2: Illustration of the top-down strategy (TDA) with PPDPTS-style tracking. High-level feature representations are analysed via a topological pipeline (e.g., point-cloud construction and persistent homology) to produce compact summaries (persistence diagrams) that identify structure-sensitive components; these components are then traced back to the underlying features/regions for privacy-feature localisation and feedback.

Figure 5.2 clarifies how TDA can be operationalised for a YOLO-style detector when combined with the PPDPTS analysis pipeline. In practice, TDA starts from semantically strong, high-level feature maps (close to the detection heads) and derives a compact structural summary (e.g., persistence diagrams/barcodes) that separates short-lived noise-like components from long-lived signal-like components. The identified components are then traced back through the feature pyramid to the corresponding lower-level feature groups, yielding the sensitive group outputs (G_i, G'_i) in a top-down manner. This makes the TDA output directly usable as a feedback signal: it indicates which feature subregions (and thus which candidate objects/areas) dominate the structural characteristics of the detection features, and therefore where privacy noise should be

assessed or adjusted by EMPA.

The specific calculation process of the Algorithm 5.2 is as follows. First, denote the input image data as D, and there is a set of data G from D, with sensitive data denoted as S_p and a privacy budget of ϵ. The goal is to divide the pixel blocks into sensitive and non-sensitive groups, G and G' respectively, where each group contains attributes u and v respectively. The upper limit of the existing array is u = 2. If the machine learning network backbone consists of i-layers, it can be represented as $G_1, G_2, \ldots, G_i$.

The algorithm starts by calculating the maximum normalized $NCP(G'i \cup G_i)$ of the subgroup. This step is using the NCP method which is mentioned in Chapter 3.5, where labels matching a pair of tuples have a result matching the sensitive class S_p. By matching $(f \in S_p) = (u, v \in S_p)$, seed tuples in G_1 and G_2 can be obtained. For the unprocessed data $Grest$ in the tuple, the algorithm calculate $NCP(G_i, G_{rest})$ and $NCP(G'i, Grest)$. The elements in G_{rest} are then matched with sensitive features and assigned to the sensitive value group G'_i via MDAV algorithm, this MDAV method was mentioned in Chapter 2.4.5. If any resulting group still contains sensitive features, it is further split into two parts.

Using this algorithm, let's assume there are 4 feature blocks at layer i. The first subdivision result is $G_i = u_1, u_2$ and $G'i = v_1$, with the remaining $Grest$. If the remaining two groups also contain sensitive features, they are classified as sensitive, and if not, they are classified as non-sensitive. The process continues, and if G_{rest} needs to be subdivided again, two groups are assigned to the remaining two groups $G_i = u_1, u_2, u_3$ and $G'_i = v_1, v_2$, completing one iteration. This process continues until all iterations are completed and the calculation is finished.

During the sensitive extraction step, the model can establish associations between layer i and layer $i-1$ by matching G_i and G_{i-1}, as well as $G'i$ and $G'i-1$, until the matching reaches the bottom level of the backbone in the machine learning model. Specifically, assuming there is a feature u in G_i, the algorithm calculates $(|G_i - G_{rest}|)$

by $u_{i-1} \cup NCP(G'i \cup G)$ and check if there are remaining unmatched features in the sensitive class $G'i - 1$. The algorithm also verify if $u_{i-1} \in S_p$ to ensure the minimization of $NCP(G_i \cup G_{i+1})$. The calculation of $u \cup NCP(G_i \cup G_{i+1})$ is a simple determination of whether the attributes between different layers, based on feature distance (neighborhood), belong to the sensitive class. The details are provided in Algorithm (5.2).

In the post-process, let u_i represent the non-sensitive feature in the layer i in the backbone of the machine learning model, where $1 \leq i \leq |N|$, and N is the total number of layers in the backbone of the machine learning model. The time complexity of finding the minimum $NCP(G_i, G_i')$ is $O(N^2)$. To reduce computational costs, a greedy algorithm can be applied. The algorithm starts by choosing a tuple of features u_i in G_i, such as selecting u_1 from G_1 in the first layer. Then, the algorithm computes $NCP(u_1, u_i)$ for $2 \leq i \leq |N|$ and find the tuple u_2 that gives the largest NCP value. The algorithm repeats this process for the remaining layers, finding the tuple with the largest NCP value each time. This is repeated until the largest NCP value no longer increases significantly.

The scheme runs Algorithm 5.2 for the third iteration, reaching a maximum NCP coverage of 97% of the dataset in D. After six iterations, the NCP calculation covers over 98.75% of the dataset in D. In practice, the maximum number of iterations can be set as a small positive integer to find the tuples f in the sensitive group. In addition to the partition time, adjustments may need to be made for $\frac{|C|}{2k}$ groups, each containing less than k tuples.

However, based on comparisons, very few groups require changes. Furthermore, the experiments in this thesis indicate that the top-down method is faster than the bottom-up method. The top-down method is a local feature exchange method, allowing two groups of data with the same backbone layer in the model to be assigned to different groups.

5.1.3 Experiments for Sensitive Classification and Clustering Algorithm with the Bottom-Up and Top-Down Strategy

Figure 5.3 and Figure 5.4 indicate that the experiment measures the difference between the original image in the MOT20 (Dendorfer et al., 2020) benchmark dataset and the image with privacy noise added by TDA and BUA methods. This experiment aims to illustrate which method has a better measurement effect by comparing the measurement difference between BUA and TDA. The image index shown on the x-axis is also from the MOT20 dataset, and the image index represents 150 images from a continuous image sequence. For the statistical analysis, the deviation value shown on the y-axis represents the measurement produced by the TDA method on different images deviation. In order to visually observe the two measurement values shown in Figure 5.3 and Figure 5.4, the 150 sequence images of the experiment in Figure 5.4 use the PPDPTS framework (from Chapter 4.5) and the experiment in Figure 5.3 use the YOLO (Bochkovskiy et al., 2020) framework as comparative analysis, where two different privacy budgets are used, $\epsilon = 0.1$ and $\epsilon = 0.01$, and both of the object detection frameworks use k-fold cross-validation for training its model. For example, PPDPTS-BUA-0.1 means that PPDPTS is used for image detection, and the BUA method is used for measurement, and the addition of measurement A plot of privacy noise uses a privacy budget of $\epsilon = 0.1$.

In Figure 5.3, due to the influence of privacy objects contained in the image itself, the values detected by the two methods have no significant difference in different privacy noise levels between the image index 0-60, this situation also exists in Figure 5.4. In Figure 5.3, in the range of 60-150, with the large increase of privacy targets, the influence of different privacy noises on the deviation value becomes very large. Among them, when the index is 119, the deviation value of the privacy budget between $\epsilon = 0.1$ and $\epsilon = 0.01$ exceeds 19.6, but even if the deviation is large, the deviation measured by the two methods of BUA and TDA is negligible, the maximum difference between

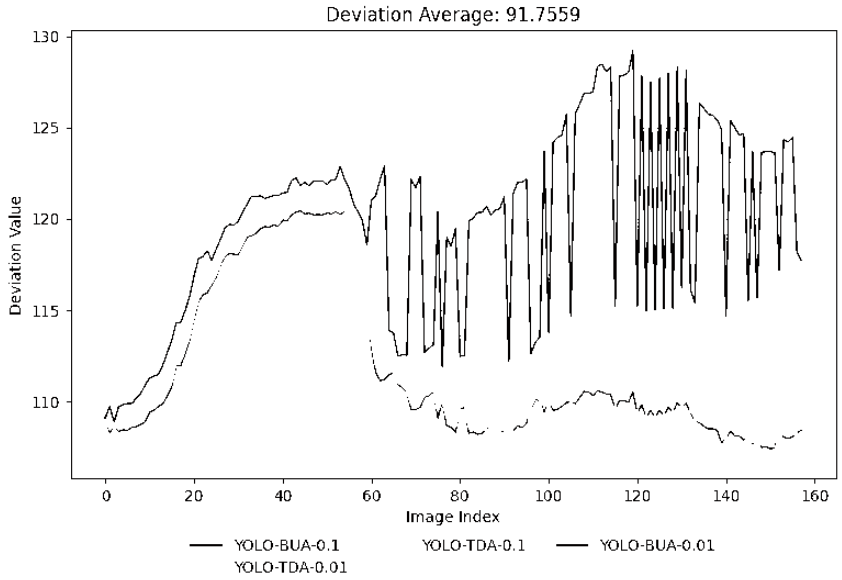


Figure 5.3: Comparison of Classification Accuracy with Top-down and Bottom-up Strategy on YOLO

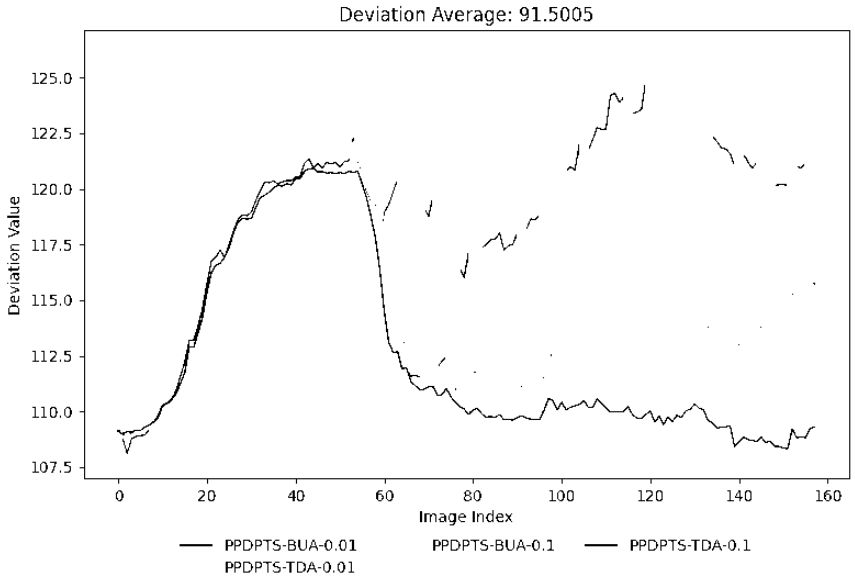


Figure 5.4: Comparison of Classification Accuracy with Top-down and Bottom-up Strategy on PPDPTS

TDA and BUA methods is only about 3.2, it has the same situation in Figure 5.4. After calculation, the average difference between the two measurement methods is about 0.57 in Figure 5.3 and 0.34 in Figure 5.4.

It can be observed from Figure 5.3 and Figure 5.4 that there is no significant difference in the detection accuracy between the two strategies when using both the YOLO and PPDPTS methods. Comparatively, the BUA method tends to exhibit more pronounced differences than the TDA method. This is because, in terms of two privacy noise levels, the BUA method yields higher noise difference values than the TDA method. Furthermore, despite using different object recognition models, the detection images are the same and the prediction results are very similar, leading to similar trends between BUA and TDA. However, the BUA detection method displays a larger discrepancy when compared to TDA.

The method proposed in this chapter aims to extract privacy features in order to address the sub-questions further divided in Chapter 5.0.1: "How to measure the feedback from the machine learning model?" The following section will concentrate on privacy budget evaluation, where privacy features will be discussed and solutions will be proposed.

5.2 Establish the Relationship with Privacy Feature via Expectation and Maximization Method

In this section, this thesis will introduce the expectation and maximization privacy assessment algorithm (EMPA) to establish the relationship between the privacy features and privacy budget determined by the two methods proposed in Chapter 5.1. The proposed solution in this section will allow us to address the final sub-question: "How can we measure the privacy budget associated with a machine learning model?"

5.2.1 Motivation of the Expectation and Maximization Privacy Assessment(EMPA) Algorithm

Algorithm 5.1 and Algorithm 5.2 employ two strategies to discover training modules in the FPN network. One challenge is establishing associations between privacy features. For instance, in the top-down strategy of the machine learning backbone network, privacy group $G'i$ has feature v_i, while $G'i - 1$ has privacy feature v_{i-1}.

To address this challenge, this thesis establishes the association between privacy features v_i and v_{i-1}. A mechanism proposed in the paper by (Hitaj et al., 2017) involves using deep belief networks (DBN) to create a feature and assess whether this feature can match two target features. However, this step requires the creation of a generator, which is time-consuming and computationally expensive. In order to improve this step, my work focuses on analyzing the following three aspects:

- Since previous experiments in the preceding chapter demonstrated that privacy protection algorithms have a negative impact on the prediction and output results of machine learning, thereby reducing prediction accuracy and impeding the training process of the protected model, introducing the relationship between privacy features can make the impact of privacy protection traceable.

- For existing deep learning models, it is essential to quantify and track the strength of privacy protection that can be achieved.

- Furthermore, evaluating the privacy level of privacy protection algorithms is more challenging than measuring the privacy-preserving impact brought about by machine learning. This is because different models employ different privacy protection methods, and the methods for measuring privacy-preserving strength may vary across different task domains. For instance, measuring the privacy-preserving strength of text and image features would involve different approaches.

5.2.2 The Process of Interference Raw Datasets with Expectation and Maximization Privacy Assessment (EMPA) Method

The expectation and maximization privacy assessment (EMPA) algorithm is a method for predicting the expected privacy features. Through EMPA, the distribution of sensitive features can be reconstructed and associated with the existing privacy class and privacy budget, allowing the measurement of whether the measured sensitive features meet the predicted expectations and defined privacy budget. The idea for this proposed work was inspired by the well-known Expectation-Maximization (EM) algorithm (Dempster, Laird & Rubin, 1977)(Agrawal & Aggarwal, 2001). The hierarchical architecture of the EM algorithm has been applied to various architectures such as neural networks (NN) (Greff, Van Steenkiste & Schmidhuber, 2017) and Transformers (Ma, Wu et al., 2021).

Let $t = t_1, t_2, \ldots, t_n$ be a sequence of N natural numbers and identically distributed random variables $T = T_1, T_2, \ldots, T_N$, with a density function of RD_t. The sequence t constitutes the raw data RD. Suppose $u = u_1, u_2, \ldots, u_N$ is also a sequence of N natural numbers and identically distributed random variables $U = U_1, U_2, \ldots, U_N$ with its density probability RD_U. This represents the additive interference with the original data. Finally, let $v = v_1, v_2, \ldots, v_n$ be the raw data with added interference, where $v_i = u_i + t_i$ for $i = 1, \ldots, N$. The goal of the additive interference is to reconstruct RD_t from u.

To estimate rd_l numerically, the values $v_i = u_i + t_i$ need to be discretized. The support Ψ of rd_l is divided into l non-overlapping intervals $\Psi_1, \ldots, \Psi_l$, with $\Phi_{i=1}^{l} \Psi_i = \Psi_l$. Let $g_i = g(\Psi_i)$ represent the length of the interval Ψ_i. The value of rd_l is a constant equal to λ_i in each interval Ψ_i. Thus, the discretized f_l can be represented by $\Lambda_X = \lambda_1, \lambda_2, \ldots, \lambda_l$ with the constraint:

$$\sum_{i=1}^{l} g_i \lambda_i = 1 \tag{5.1}$$

In Eq.(5.1), g_i represents the variable g_i, which is associated with the index i in the summation. The specific meaning or definition of g_i depends on the context or problem being addressed. It could represent weights, coefficients, or any other value or variable. λ_i represents the variable λ_i, which is also associated with the index i in the summation. Similar to g_i, the meaning or definition of λ_i depends on the specific context. It could represent Lagrange multipliers, parameters, or any other value or variable. By choosing a sufficiently large l, rd_T can be approximated to arbitrary precision.

Given the observations z, the Maximum Log-likelihood Estimate (MLE) of Λ_X can be obtained as follows:

$$\hat{\Lambda}MLE = \arg\max \Lambda \ln rd_{T;\Lambda}(t) \tag{5.2}$$

In Eq.(5.2), $\hat{\Lambda}MLE$ represents the maximum likelihood estimate (MLE) of the parameter vector Λ. The MLE is an estimation method used to find the values of the parameters that maximize the likelihood function. In this equation, $\hat{\Lambda}MLE$ represents the estimated values of Λ that maximize the likelihood. And $\arg\max_\Lambda$ denotes the argument that maximizes the expression following it. It represents the values of Λ for which the function $\ln rd_{T;\Lambda}(t)$ is maximized. Since z is observable and $z \neq t$, the EMPA algorithm can be used to compute the maximum likelihood estimation (MLE) (Richards, 1961) for estimating the distribution of features in privacy-level assessment.

$$M(\Lambda, \hat{\Lambda}) = E\left[\ln rd_{T;\Lambda}(t)|v; \hat{\Lambda}\right] \tag{5.3}$$

In Eq.(5.3), $M(\Lambda, \hat{\Lambda})$ represents the function M that takes the parameters Λ and $\hat{\Lambda}$ as inputs. It is a function that calculates an expected value. $E\left[\ln rd_{T;\Lambda}(t)|v; \hat{\Lambda}\right]$ represents the expected value of the expression $\ln rd_{T;\Lambda}(t)$ given certain conditions. The expected value, denoted by $E[\cdot]$, is a statistical concept that represents the average value or

the mean value of a random variable. In this case, the expected value is taken with respect to the random variable t. $\ln rd_{T;\Lambda}(t)$ represents the natural logarithm of the function $rd_{T;\Lambda}(t)$. The specific definition and meaning of $rd_{T;\Lambda}(t)$ depends on the context or problem being addressed. It could be a likelihood function, a probability density function, or any other function related to the data t and the parameter vector Λ.

Let Eq.(5.3) be the expectation of $\ln rd_{T;\Lambda}(t)$, which given the observed data v and the current time estimate value as $\hat{\Lambda}$. Starting with the loop $l \rightarrow (0, \ldots, n)$ and an initial random value Λ^0, trigger the idea from the expectation-maximization algorithm (Dempster et al., 1977), the EMPA algorithm includes iterations through the following steps:

1. **Input:** Input non-sensitive groups $G_1, \ldots, G_i$ and sensitive group $G'_1, \ldots, G'_i$ from BUA or TDA. Also, input privacy budget ϵ and sensitive labels Class S_ϵ.

2. **Expectation-step (E-step)**: Privacy maximization $M(\Lambda, \Lambda^l)$ can be calculated from the Theorem 5.1. Here, Λ^l is the random parameter in the loop l.

3. **Maximize step size (M steps)**: update the value of Λ^{l+1} to $\Lambda^{l+1} = \arg\max_\Lambda L(\Lambda, \Lambda^l)$.

4. **Output**: The bias values measured between privacy budget and sensitive features $G'_1, \ldots, G'_i$.

The parameter vector Λ in the EMPA iteration is an abstract vector of mixture weights; for example, $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_L)$ where L is the number of components (e.g., sensitive groups), each $\lambda_i \geq 0$ and $\sum_{i=1}^{L} \lambda_i = 1$. The i-th component λ_i is updated in the M-step from the E-step weights $\omega_i(v; \hat{\Lambda})$ (see Eq. 5.12). So Λ is the vector of parameters that the EMPA algorithm estimates to align the distribution of privacy features with the noise implied by the privacy budget.

In the EMPA process, the non-sensitive groups $G_1, \ldots, G_i$ and the sensitive group $G'_1, \ldots, G'_i$ are inputted into the expectation step. During this step, the EMPA method

calculates the distribution characteristics of D' by computing the expectation of sensitive features. In the maximization process, the probability distribution of the parameters is estimated using maximum likelihood estimation (MLE). The estimated probability distribution is then used to assess whether the emerging privacy features align with the distribution characteristics of the noise generated by the corresponding privacy budget. The resulting bias value is returned as a result.

In the EMPA steps, the input value is iterated until convergence. The convergence threshold can be explicitly set or based on stricter conditions (Cramer, 1946).

The following two theorems provide the formulas to compute $L(\Lambda, \hat{\Lambda})$ and Λ^{l+1} for the current reconstruction work in EMPA.

Theorem 5.1. *When $M(\Lambda, \hat{\Lambda})$ has an expectation and the privacy cost is $Cost(U - \Omega_i)$, the control parameter $\omega_i(v; \hat{\Lambda})$ based on the independence v can be defined as:*

$$\omega_i(v; \hat{\Lambda}) = \sum_{j=1}^{N} \frac{\hat{\lambda} i Cost(U - \Omega_i)}{rdv; \hat{\Lambda}(v_j)} \tag{5.4}$$

In Eq.(5.4), $Cost(U - \Omega_i)$ is the cost calculated outside of Eq.(5.3), and the density probability U falls within the scope of $v_j - \Omega_i$, where v_j is the independence degree of v in loop j and Ω_i is the likelihood impact parameter of Λ.

Proof. In order to prove the Theorem 5.1, the Eq.(5.3) can be calculated out at the expectation step. In expectation step, the $M(\Lambda, \hat{\Lambda})$ can be deducted from, $M(\Lambda, \hat{\Lambda}) = \sum_{i=1}^{L} [\ln \lambda_i \omega_i(v; \hat{\Lambda})]$

$$
\begin{aligned}
M(\Lambda, \hat{\Lambda}) &= \sum_{i=1}^{L} [\ln \lambda_i \omega_i(v; \hat{\Lambda})] && (5.5)\\
&= E[\ln rd_{t;\Lambda}(T)|v; \hat{\Lambda}] && (5.6)\\
&= \sum_{j=1}^{N} E[\ln rd_{T_j;\Lambda}(t)|v; \hat{\Lambda}] \\
&= \sum_{j=1}^{N} E[\ln rd_{T_j;\Lambda}(t_j)|v_j; \hat{\Lambda}] && (5.7)
\end{aligned}
$$

In Eq.(5.5), it has independent random variables $T_1, T_2, \ldots, T_N$. $\ln \lambda_i$ represents the natural logarithm of the i-th element of the parameter vector Λ. It transforms the parameter λ_i to the logarithmic scale. $\omega_i(v; \hat{\Lambda})$ represents a function ω_i that takes inputs v and $\hat{\Lambda}$. The specific definition and meaning of ω_i depend on the context. It could be a weight or a coefficient associated with the i-th parameter, and it incorporates additional information v and the estimated values $\hat{\Lambda}$.

And the independence of T and U gives in:

$$
rd_{T_j|v_j;\Lambda}(v) = \frac{rd_{t,v;\hat{\Lambda}}(v, v_j)}{rd_{V;\hat{\Lambda}}(v_j)} \tag{5.8}
$$

In Eq.(5.8), $rd_{T_j|v_j;\Lambda}(v)$ represents a conditional distribution or probability density function. It represents the distribution of the random variable v given the dataset T_j and the additional conditions or information denoted by v_j, under the model defined by the parameter vector Λ. In other words, it represents the likelihood of observing the value v given the specific dataset T_j and the conditions v_j. $rd_{t,v;\hat{\Lambda}}(v, v_j)$ represents a joint distribution or joint probability density function. It represents the joint distribution of the random variables t and v under the model defined by the estimated parameter vector $\hat{\Lambda}$. It represents the likelihood of observing the values v and v_j together, given the data t, under the model defined by $\hat{\Lambda}$. Lastly, $rd_{V;\hat{\Lambda}}(v_j)$ represents a marginal

distribution or marginal probability density function. It represents the distribution of the random variable v_j alone, under the model defined by the estimated parameter vector $\hat{\Lambda}$. It represents the likelihood of observing the value v_j given the model defined by $\hat{\Lambda}$. This equation describes the relationship between conditional and marginal distributions and can be interpreted as a way to express the likelihood of an event v given specific conditions T_j and v_j relative to the overall likelihood of the event v_j under the model.

Thus,

$$
\begin{aligned}
E[\ln rd_{T_j;\Lambda}(t_j)|v_j;\hat{\Lambda}] &= \int_{\Omega} \frac{\ln rd_{V_i;\Lambda}(x) f_{T,V;\hat{\Lambda}}(x,v_j)}{rd_{V;\hat{\Lambda}}(v_j)} dx \\
&= \frac{\sum_{i=1}^{l} \int_{\Omega} \ln[rd_{T_i;\Lambda}(v)] \hat{\lambda}_i I_{\Phi_i}(v) rd_U(v_j - x) dx}{rd_{V;\hat{\Lambda}}(v_j)} \qquad (5.9) \\
&= \sum_{i=1}^{l} \hat{\lambda}_i \int_{\Omega} \frac{\ln[rd_{t_i;\Lambda}(v)] rd_t(v_j - x) dx}{rd_{V;\hat{\Lambda}}(v_j)} \\
&= \sum_{i=1}^{l} \hat{\lambda}_i \int_{\Omega} \frac{\ln(\lambda_i) rd_U(v_j - x) dx}{rd_{V;\hat{\Lambda}}(v_j)} \\
&= \sum_{i=1}^{l} \hat{\lambda}_i \ln \lambda_i \frac{\int_{\Omega} rd_U(v_j - x) dx}{rd_{V;\hat{\Lambda}}(v_j)} \\
&= \sum_{i=1}^{l} \hat{\lambda}_i \ln \lambda_i \frac{Cost(u - \Omega_i)}{rd_{V;\hat{\Lambda}}(v_j)} \qquad (5.10)
\end{aligned}
$$

In Eq.(5.10), $E[\ln rd_{T_j;\Lambda}(t_j)|v_j;\hat{\Lambda}]$ represents the expected value of the expression $\ln rd_{T_j;\Lambda}(t_j)$ given the conditions specified by v_j and $\hat{\Lambda}$. It calculates the average value of the logarithm of the likelihood or probability density function $rd_{T_j;\Lambda}(t_j)$, which depends on the data t_j, under the model defined by the parameters Λ. And $\int_{\Omega} \frac{\ln rd_{V_i;\Lambda}(x) f_{T,V;\hat{\Lambda}}(x,v_j)}{rd_{V;\hat{\Lambda}}(v_j)} dx$ represents an integral over the domain Ω. The integral integrates the expression $\frac{\ln rd_{V_i;\Lambda}(x) f_{T,V;\hat{\Lambda}}(x,v_j)}{rd_{V;\hat{\Lambda}}(v_j)}$ with respect to the variable x. Here, Ω represents the domain of integration, which defines the range of values that the variable x can take. $\ln rd_{V_i;\Lambda}(x)$ represents the natural logarithm of the likelihood or probability density function $rd_{V_i;\Lambda}(x)$ under the model defined by the parameters Λ. $f_{T,V;\hat{\Lambda}}(x,v_j)$

represents a joint distribution or joint probability density function of the variables T and V, which are related to the data t_j and the conditions v_j. The specific definition and meaning of $f_{T,V;\hat{\Lambda}}$ depend on the context. $rd_{V;\hat{\Lambda}}(v_j)$ represents a marginal distribution or marginal probability density function of the variable V alone, under the model defined by the parameters $\hat{\Lambda}$. $Cost(u - \Omega_i)$ represents a cost function or penalty associated with the difference $u - \Omega_i$. The specific definition and meaning of the cost function depend on the context. Because $I_{\Phi_i}(v)$ is the indicator function which is 1 if $v \in \Phi_i$, otherwise 0.

Eq.(5.10) calculates the expected value of the logarithm of a likelihood function given certain conditions. It involves integrals, summations, probability density functions, and parameters, with the goal of evaluating the expected log-likelihood under a specific model and conditions. The specific interpretation and use of this equation depend on the context or application in which it is being used.

From the result Eq.(5.4) and the result Eq.(5.10), the result can be calculated for $M(\Lambda, \Lambda^l)$ as

$$\begin{aligned} M(\Lambda, \hat{\Lambda}) &= \sum_{i=1}^{l} \hat{\lambda}_i \ln \lambda_i \sum_{j=1}^{N} \frac{Cost(u - \Omega_i)}{rd_{V;\hat{\Lambda}}(v_j)} \\ &= \sum_{i=1}^{l} \ln \lambda_i \end{aligned} \tag{5.11}$$

In Eq.(5.11), $M(\Lambda, \hat{\Lambda})$ represents a function, denoted by M, that takes the parameters Λ and the estimated parameters $\hat{\Lambda}$ as inputs. It calculates a specific quantity associated with these parameters. The specific interpretation and use of this function depend on the context or application. $\sum_{i=1}^{l} \hat{\lambda}_i \ln \lambda_i$ represents a summation over an index i and involves the weights or coefficients $\hat{\lambda}_i$ and the natural logarithm of the parameters λ_i. Here, $\hat{\lambda}_i$ represents a weight or coefficient associated with the i-th term in the summation. It scales the contribution of each term in the summation. $\ln \lambda_i$ represents the natural

logarithm of the parameter λ_i.

The $\omega_i(v;\hat{\Lambda})$ has $\lambda_i \sum_{j=1}^{N} \frac{Cost(u-\Omega_i)}{rd_{V;\hat{\Lambda}}(v_j)}$, proof end. □

Theorem 5.2. *The value of $\hat{\lambda}$ that maximizes $T(\Lambda,\hat{\Lambda})$ in the M-phase of the reconstruction algorithm is given by:*

$$\lambda_i = \frac{\omega_i(v;\hat{\Lambda})}{g_i N} \tag{5.12}$$

In Eq.(5.12), $\hat{\lambda_i}$ represents the value of the parameter λ_i that maximizes the function $T(\Lambda,\hat{\Lambda})$ in the M-phase of the EMPA. $\omega_i(v;\hat{\Lambda})$ represents a weight or coefficient associated with the i-th term, denoted by ω_i, which depends on the conditions v and the estimated parameters $\hat{\Lambda}$. The specific definition and meaning of ω_i is given by Eq.(5.1). g_i represents a constant associated with the i-th term. It does not depend on the conditions or parameters. And N represents a constant value, which can be interpreted as the total number of data points or observations.

Proof. The maximum value can be obtained using the Lagrange multiplier. Since $\sum_{i=1}^{l} g_i\lambda_i = 1$, the Lagrange multiplier function can be expressed as

$$M(\Lambda,\Xi) = \sum_{i=1}^{l} \omega_i(v;\hat{\Lambda}) + \xi(\sum_{j=1}^{l} g_j\lambda_j)\ln\lambda_i \tag{5.13}$$

In Eq.(5.13), $M(\Lambda,\Xi)$ represents a Lagrange multiplier function, denoted by M, that takes the parameters Λ and Ξ as inputs. It is a mathematical function used in optimization problems involving constraints. $\sum_{i=1}^{l} \omega_i(v;\hat{\Lambda})$ represents a summation over an index i and involves the weights or coefficients $\omega_i(v;\hat{\Lambda})$. Here, $\omega_i(v;\hat{\Lambda})$ represents a weight or coefficient associated with the i-th term in the summation. The specific definition and meaning of $\omega_i(v;\hat{\Lambda})$ depend on the context or application. The summation calculates the sum of the weights $\omega_i(v;\hat{\Lambda})$ over all l terms. $\xi(\sum_{j=1}^{l} g_j\lambda_j)\ln\lambda_i$ represents a Lagrange multiplier associated with the constraint $\sum_{j=1}^{l} g_j\lambda_j$. And ξ represents a Lagrange multiplier coefficient. It is a scalar value that influences the impact of the constraint

on the objective function. $\sum_{j=1}^{l} g_j \lambda_j$ represents the summation of the product of the constant values g_j and the parameters λ_j overall l terms. It represents the constraint in the optimization problem. $\ln \lambda_i$ represents the natural logarithm of the parameter λ_i.

The Eq.(5.13) calculates the Lagrange multiplier function $M(\Lambda, \Xi)$ by summing the weights $\omega_i(v; \hat{\Lambda})$ and incorporating the constraint $\sum_{j=1}^{l} g_j \lambda_j$ using the Lagrange multiplier ξ. The specific interpretation and use of this equation depend on the scene of EMPA and the constraints it is associated with.

The Lagrangian constraints of this function are $\frac{\partial L}{\partial \lambda_i} = 0$, and $\frac{\partial L}{\partial \xi} = 0$. The corresponding conditions are $\lambda_i = -\frac{\omega_i(v;\hat{\Lambda})}{\xi g_j}$ and $\sum_{j=1}^{l} g_j \lambda_j = 1$. Eliminating ξ from these conditions in Eq.(5.13), it has:

$$\lambda_i = \frac{\omega(v; \hat{\Lambda})}{g_i \sum_{i=1}^{l} \omega_i(v; \hat{\Lambda})} \tag{5.14}$$

In Eq.(5.14), λ_i represents a Lagrange multiplier associated with the i-th constraint. In this equation, it represents the Lagrange multiplier for the constraint $\sum_{j=1}^{l} g_j \lambda_j$. The Lagrange multiplier λ_i determines the impact of the constraint on the optimization problem. $\omega(v; \hat{\Lambda})$ represents a weight or coefficient associated with the conditions v and the estimated parameters $\hat{\Lambda}$. The specific definition and meaning of $\omega(v; \hat{\Lambda})$ depend on the context or application. g_i represents a constant associated with the i-th constraint. It does not depend on the conditions or parameters. $\sum_{i=1}^{l} \omega_i(v; \hat{\Lambda})$ represents the summation of the weights $\omega_i(v; \hat{\Lambda})$ over all l terms. It represents the sum of the weights associated with all constraints in the optimization problem. The purpose of Eq.(5.14) is to determine the Lagrange multipliers λ_i that satisfy the Lagrangian constraints associated with the optimization problem. These Lagrange multipliers are chosen such that they balance the objective function and the constraints. The specific interpretation and use of this equation depend on the context or application in which it is being used. The Eq.(5.14) can export into the Eq.(5.13) and Lagrange multiplier functions can be calculated.

Since the density function V can be obtained from the relation $V = T + U$, and T and U are independent degree of each other, thus

$$\begin{aligned} rd_{V;\hat{\Lambda}}(x) &= \int rd_T(x) rd_U(v-x)dx \\ &= \sum_{i=1}^{l} \int_{\Phi_i} \hat{\lambda}_i rd_U(v-x)dx \\ &= \sum_{i=1}^{l} \hat{\lambda}_i Cost(U \in v - \Phi_i) \end{aligned} \tag{5.15}$$

In Eq.(5.15), $rd_{V;\hat{\Lambda}}(x)$ represents a probability density function (PDF) associated with the random variable V and the estimated parameters $\hat{\Lambda}$. It denotes the PDF of V evaluated at the point x. The specific definition and meaning of $rd_{V;\hat{\Lambda}}(x)$ depends on the context or application. $rd_T(x)$ represents a probability density function (PDF) associated with the random variable T. It denotes the PDF of T evaluated at the point x. Again, the specific definition and meaning of $rd_T(x)$ depends on the context or application. $rd_U(v-x)$ represents a probability density function (PDF) associated with the random variable U. It denotes the PDF of U evaluated at the point $v - x$. Similarly, the specific definition and meaning of $rd_U(v-x)$ depending on the context or application. dx represents the differential element used for integration. It indicates that the integration is performed with respect to the variable x. Φ_i represents a region or domain associated with the i-th term in the summation. It defines the limits of integration for the corresponding integral. $\hat{\lambda}_i$ represents an estimated parameter or weight associated with the i-th term in the summation. It represents the importance or influence of the i-th term in the overall calculation. $Cost(U \in v - \Phi_i)$ represents a cost or penalty function associated with the condition $U \in v - \Phi_i$, which seek privacy cost for EMPA algorithm. It evaluates whether the random variable U falls within the region $v - \Phi_i$. The specific definition and meaning of the cost function depend on the context or application.

The denominator can be expressed as

$$\begin{aligned}\sum_{i=1}^{l}\omega_l(v;\hat{\Lambda}) &= \sum_{i=1}^{l}\hat{\lambda}_l\sum_{j=1}^{l}\frac{Cost(U-\Phi_l)}{rd_{V;\hat{\Lambda}}(v_j)} \\ &= \sum_{i=1}^{l}\frac{\sum_{j=1}^{l}\hat{\lambda}_l Cost(U-\Phi_l)}{rd_{V;\hat{\Lambda}}(v_j)}\end{aligned} \tag{5.16}$$

In Eq.(5.16), $\sum_{i=1}^{l}\omega_l(v;\hat{\Lambda})$ represents the summation of the weights or coefficients $\omega_l(v;\hat{\Lambda})$ over all l terms. It represents the sum of the weights associated with each term in the equation. $\hat{\lambda}_l$ represents an estimated parameter or weight associated with the l-th term in the summation. It denotes the importance or influence of the l-th term in the overall calculation. $\sum_{j=1}^{l}\frac{Cost(U-\Phi_l)}{rd_{V;\hat{\Lambda}}(v_j)}$ represents the summation over j of the ratio $\frac{Cost(U-\Phi_l)}{rd_{V;\hat{\Lambda}}(v_j)}$ for each l term. It combines the cost function $Cost(U-\Phi_l)$ and the reciprocal of the PDF $rd_{V;\hat{\Lambda}}(v_j)$ evaluated at v_j. The specific definition and meaning of the cost function and PDF depend on the context or application.

By combining the Eq.(5.14) and Eq.(5.16), it has,

$$\sum_{i=1}^{l}\omega_l(v;\hat{\Lambda}) = \frac{\omega_i(v;\hat{\Lambda})}{g_i\lambda_i} = \frac{rd_{V;\hat{\Lambda}}(v_i)}{\sum_{i=1}^{l}rd_{V;\hat{\Lambda}}(v_i)} \tag{5.17}$$

In Eq.(5.17),$\omega_l(v;\hat{\Lambda})$ represents the weight or importance assigned to the l-th term or sample, given the input v and the parameter $\hat{\Lambda}$. This weight can be calculated based on certain criteria or models used in the specific context of the problem. And g_i is a scaling factor or coefficient associated with the i-th term or sample. It is used to adjust or normalize the weight $\omega_i(v;\hat{\Lambda})$, λ_i is a parameter associated with the i-th term or sample. Similar to g_i, it is used to adjust or normalize the weight $\omega_i(v;\hat{\Lambda})$, the weight $\omega_i(v;\hat{\Lambda})$ is the convergence properties (McLachlan & Krishnan, 1996). $rd_{V;\hat{\Lambda}}(v_i)$ represents a term or value associated with the i-th sample or data point, given the input v and the parameter $\hat{\Lambda}$. This term can be calculated using a specific function or model defined in

the problem domain. $\sum_{i=1}^{l} rd_{V;\hat{\Lambda}}(v_i)$ is the sum of the terms $rd_{V;\hat{\Lambda}}(v_i)$ over all l samples or data points. It serves as the denominator in the equation and is used to normalize the weights.

Therefore, Eq.(5.16) becomes

$$\frac{\omega_i(v;\hat{\Lambda})}{g_i\lambda_i} = \sum_{i=1}^{N} \frac{rd_{V;\hat{\Lambda}}(v_i)}{rd_{V;\hat{\Lambda}}(v_i)}$$

In Eq.(5.18), $\omega_i(v;\hat{\Lambda})$ represents the weight or importance assigned to the i-th term or sample, given the input v and the parameter $\hat{\Lambda}$. This weight can be calculated based on certain criteria or models used in the specific context of the problem. And g_i is a scaling factor or coefficient associated with the i-th term or sample. It is used to adjust or normalize the weight $\omega_i(v;\hat{\Lambda})$. λ_i is a parameter associated with the i-th term or sample. Similar to g_i, it is used to adjust or normalize the weight $\omega_i(v;\hat{\Lambda})$. $rd_{V;\hat{\Lambda}}(v_i)$ represents a term or value associated with the i-th sample or data point, given the input v and the parameter $\hat{\Lambda}$. This term can be calculated using a specific function or model defined in the problem domain.

Eq.(5.18) resulting in Eq.(5.12), proof end. □

In EMPA's algorithm, my work involved the weight of convergence properties of EM algorithm, the (McLachlan & Krishnan, 1996) has explain how the weight of convergence properties works. The convergence method used in EMPA can ensure that the continuous approximation $\Lambda_0, \Lambda_1, \Lambda_2, \ldots$ converges to $\hat{\Lambda}_{ML}$ (such as the Eq.(5.2)).

After explaining the EMPA method, the next section will combine the EMPA method proposed in this thesis and the BUA and TDA methods to test the accuracy of these two methods for privacy level evaluation.

5.3 Experimental Evaluation of the EMPA with BUA, TDA Methods

5.3.1 Ablation Study of the EMPA with BUA, TDA Methods

In order to verify the method proposed in this thesis through experiments, this chapter will introduce an experimental method based on distribution measurement and compare several methods to demonstrate that the proposed method can measure the differences in private images. Additionally, it will utilize the information presented in the first two chapters and employ other three-party machine learning models to validate the methods discussed in this chapter. Specifically, the K-L divergence method will measure the distribution of the original data and the data augmented with privacy noise for distribution verification. Further details are discussed in Chapter 2.6.6. Moreover, this article will employ the MMD method to compare the methods proposed in this article. MMD is used to measure the distance between two different yet related distributions. The specific discussion is explained in Chapter 2.6.4. Finally, this chapter also employs the Chi-squared test to detect differences between the original image and the noise-added image and compares EMPA and other methods. This part is explained in Chapter 2.6.8.

At the same time, the EMPA method combined with the TDA or BUA method also requires the restoration and reconstruction of features from the two input datasets. In this chapter, the details of the Expectation and Maximization Privacy Assessment (EMPA) algorithm presented in Chapter 5.1.1, along with the sensitive classification and clustering using the top-down algorithm (TDA) and the sensitive classification and clustering using the bottom-up strategy algorithm (BUA) are explained in Chapter 5.1.2. The performance evaluation of this aspect needs to include robustness detection. Considering the algorithm's robustness, Chapter 2.6.4 describes the detection of errors in several tested algorithms over a given time period using the rMSE method, thereby

providing a comprehensive evaluation of the tested methods.

In order to capture the diversity of image features, this experiment utilizes the COCO dataset (T.-Y. Lin et al., 2014) as the benchmark dataset. The COCO dataset consists of 80 different categories of objects, enabling a comprehensive evaluation of the feature retrieval capabilities of the two methods in terms of feature diversification. Three methods are employed to measure the privacy level of sensitive classes, and the ground truth for comparison is the noise data generated by the privacy protection method, which involves overlaying Gaussian noise on the original data. In subsequent experiments, k-fold cross-validation will be used. In general experiments, the value of k will be set to 10. Specifically, for the experimental COCOs dataset, $k - 1$ subsets are named "Train2020" and used as training sets, while the remaining subset is named "Val2020" and used as the test verification dataset.

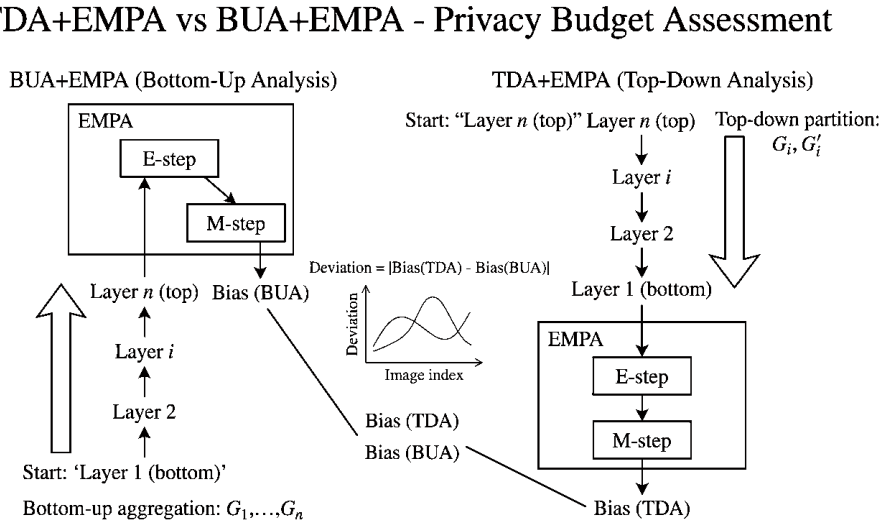


Figure 5.5: The deviation value for different images with the TDA+EMPA and BUA+EMPA algorithms

Figure 5.5 represents the deviation values of the TDA+EMPA and BUA+EMPA algorithms, the deviation value on the y-axis, and the image index from MOT datasets on the x-axis. This experiment aims to measure the differences between the two

privacy measurement algorithms. The TDA and BUA methods are combined with the EMPA method respectively to detect the deviation among the image datasets. In these experiments, 0 means ground truth, and if a deviation value is 0, it means that this indicator is exactly the same as the original data or no noise has impacted the data.

The deviation value displayed on the y-axis represents the measurement deviation generated by the TDA+EMPA and BUA+EMPA methods on different images. This deviation value indicates the difference between the original images in the MOT20 dataset and the images with privacy noise added. A larger deviation value usually corresponds to a higher privacy level (due to inserted higher amount of privacy noise). The image indexes shown on the x-axis are also from the MOT20 dataset (Dendorfer et al., 2020), and the image indexes represent the selection of 150 images from a continuous image sequence. These 150 images have all been augmented with privacy noise using the PPDPTS method(from Chapter 4.5), with a privacy budget of $\epsilon = 0.001$.

Overall, the deviation values of both methods tend to decrease as the image index increases. The overall deviation of TDA+EMPA is slightly lower than that of BUA+EMPA, but the difference between them is not significant. However, between image numbers 100 and 130, the difference between the two methods changes. Particularly between 100 and 130, the deviation value of the TDA+EMPA method suddenly becomes higher than that of BUA+EMPA. From Figure 5.5, it can be observed that TDA+EMPA generally has slightly smaller deviation values than BUA+EMPA, indicating that both algorithms exhibit similar detection capabilities in practical image detection. However, in vision detection scenarios, there may be slight deviations between the two algorithms. Overall, these deviations remain within an acceptable range of deviation.

5.3.2 Horizontal Evaluation of Privacy Measurement Methods

The previous experiment aimed to compare the performance of different methods on various images from a vertical perspective. To provide a comprehensive comparative analysis between the EMPA method and several other testing methods, this study will also incorporate machine learning models based on image recognition to facilitate the evaluation process. The image recognition models utilized include the Privacy-Preserving Modified Dynamic Conditional Random Field (MDCRF) model combined with YOLO proposed in Chapter 3.7, as well as the privacy-preserving deep transformation with self-attention (PPDPTS) model discussed in Chapter 4.5. Additionally, some state-of-the-art methods in the research community, such as the YOLO model (Bochkovskiy et al., 2020) and the DETR model (Carion et al., 2020), will be employed. These models all perform the same object recognition task, enabling a horizontal comparison of privacy noise measures and providing a unified baseline for evaluation. In these experiments, 0 means ground truth, and if the rMSE value is 0, it means that this indicator is no error with the original data.

Table 5.1: RMSE value with Chi-square test, K-L divergence, MMD, and EMPA

Method	Indicator	MDCRF-0.1	DETR-0.1	PPDPTS-0.1	PPDPTS-0.01	rMSE max gap
rMSE	Chi-square test	2.41	3.33514	3.6413	3.23	1.2313
	K-L Divergence	0.033125	0.0334252	0.032812	0.038231	0.005419
	MMD	3.44096	3.54229	3.001089	8.5624	5.561311
	EMPA	2.02123	3.02455	2.02135	5.2132	3.19197

The experiments presented in Table 5.1 assess the bias of the EMPA method in measuring privacy levels by comparing it with multiple mechanisms. The accuracy of the EMPA method in measuring privacy noise can be demonstrated by comparing it with other mechanisms. The rMSE (root mean square error) value is calculated

based on the difference between two images: one is the original image, and the other is the image with privacy noise added. These images are sourced from the MOT20 dataset (Dendorfer et al., 2020) as a benchmark dataset. For the comparative analysis, Table 5.1 includes four different metrics (Chi-Square, K-L Divergence, MMD, and EMPA) listed under the "Indicator" column. The subsequent rows present the rMSE values for each combination of method and metric. In this test. the MDCRF has combined with YOLO (Bochkovskiy et al., 2020).

The first method is the Chi-Square test as statistical analysis, which is discussed in Chapter 2.6.8. The rMSE values corresponding to the MDCRF-0.1, DETR-0.1, PPDPTS-0.1, and PPDPTS-0.01 indicators are 2.41, 3.33514, 3.6413, and 3.23, respectively. It should be noted that using the Chi-Square test is not a privacy measurement method, and the obtained numerical error may be significant.

The second method is K-L Divergence, discussed in Chapter 2.6.6. The rMSE values corresponding to the MDCRF-0.1, DETR-0.1, PPDPTS-0.1, and PPDPTS-0.01 indicators are 0.033125, 0.0334252, 0.032812, and 0.038231, respectively. These small rMSE values indicate that the K-L Divergence method yields minimal differences between predicted and actual values.

The third method is MMD (Maximum Mean Difference), discussed in Chapter 2.6.5. The rMSE values corresponding to the MDCRF-0.1, DETR-0.1, PPDPTS-0.1, and PPDPTS-0.01 indicators are 3.54096, 3.44229, 3.001089, and 8.5624, respectively. Higher rMSE values indicate larger differences between predicted and actual values when using the MMD method.

Finally, the EMPA (Empirical Privacy Model Analysis) method provides rMSE values of 2.02123, 3.02455, 2.02135, and 5.2132 for the MDCRF-0.1, DETR-0.1, PPDPTS-0.1, and PPDPTS-0.01 indicators, respectively. These relatively low rMSE values suggest a smaller difference between predicted and actual values when employing the EMPA method.

The rMSE values in the table demonstrate the variations in detecting private images using different methods. A higher rMSE value indicates a greater difference in the noise added by the methods. K-L Divergence, MMD, and EMPA all detected the DETR method as having the largest difference in privacy noise among the three methods. However, the Chi-Square test is not able to accurately detect subtle differences caused by privacy noise. In the MDCRF-0.1, DETR-0.1, PPDPTS-0.1, and PPDPTS-0.01 methods, it can only detect noticeable differences with more privacy noise, such as $\epsilon = 0.01$ for PPDPTS. Although the K-L Divergence method based on distribution statistics can detect differences, they may not be apparent, and errors may occur. Finally, the EMPA method proposed in this thesis effectively detects differences from various angles, and the experiment verifies its suitability for measuring the actual privacy budget.

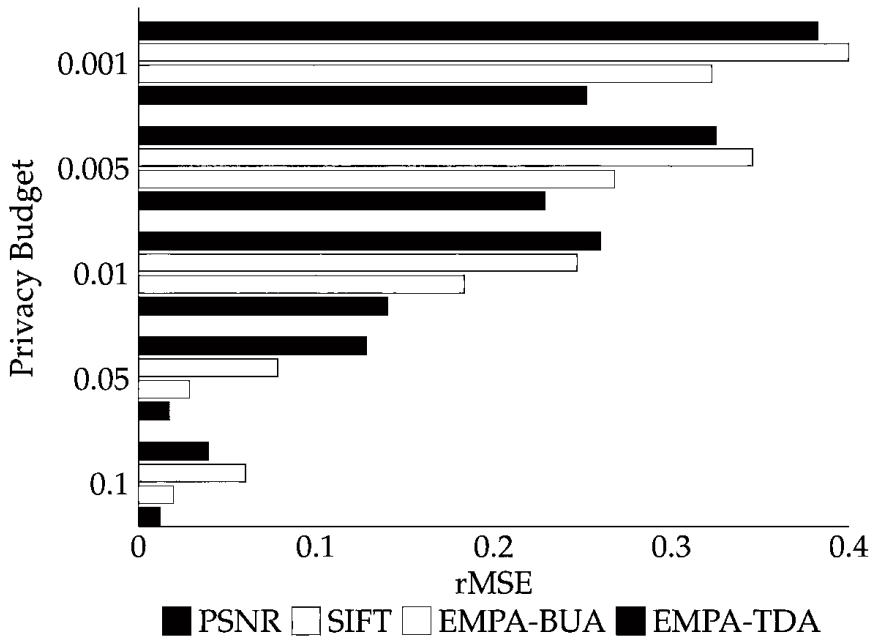


Figure 5.6: The privacy budget compared with the TDA+EMPA and BUA+EMPA algorithms.

As shown in Figure 5.6, the x-axis represents the root mean square error (rMSE)

value as statistical analysis, while the y-axis represents the percentage of the privacy budget used. Assuming the entire privacy budget is utilized, a value of 1.0 corresponds to 100%. For all experiments conducted here and in the following sections, if the same reference system is used for the x-axis and y-axis, the result from the Laplace method is used as the benchmark for reference. Define the ground-truth rMSE value as 0, where 0 indicates that the privacy budget has been accurately detected from the privacy-preserving images on the dataset. This test chooses the COCOs (T.-Y. Lin et al., 2014) dataset as the benchmark dataset because it provides 80 categories of different features, which can better reflect the generalization ability of the tested method. As the privacy budget represented by the y-axis increases, the injected privacy noise volume becomes higher. For the comparative analysis, this test applies the privacy-preserving method mentioned in Chapter 3.4 serves as the reference image, indicating that the original image will be injected with privacy noise if its region belongs to a sensitive group. In general, if a method has a lower rMSE value, i.e., a smaller value represented in the figure, it means that the method can more accurately detect the privacy budget with less error. In these experiments, 0 means ground truth, and if the rMSE value is 0, it means that this indicator has 0 error compared with the original data.

From Figure 5.6, it can be observed that at each privacy budget(cost) level, the values calculated by the TDA with EMPA method and the BUA with EMPA method are very close, suggesting a minimal difference between the two. Meanwhile, the difference between the EMPA method and the Laplace reference value is larger compared to the difference between BUA or TDA and the reference value. Furthermore, the divergence from the ground truth is the smallest, i.e., the rMSE is the lowest at each privacy level, for the TDA method. This indicates that the TDA with EMPA method performs better in terms of measurement. However, since the EMPA method is an independent measurement method unrelated to the original model, both BUA and TDA with EMPA serve as privacy budget measurement methods based on the FPN network. Therefore,

although the Scale-Invariant Feature Transform (SIFT)(Cruz-Mota, Bogdanova, Paquier, Bierlaire & Thiran, 2012) and Peak Signal-to-Noise Ratio (PSNR)(Sara, Akter & Uddin, 2019) methods are less effective, they still hold significance. Additionally, as a comparative method, using the PSNR method to compare the ground truth image with the original image yields the largest error with respect to the true privacy budget, indicating that the privacy budget is essentially uncertain. In general, if a method performs worse than the PSNR method, it signifies that the PSNR method has the worst effect and can verify the degree of privacy-preserving.

Membership inference evaluation. To demonstrate that the proposed PPML methods reduce the risk of membership inference in practice, a membership inference attack (MIA) was conducted against the trained models. For each model (e.g., PPDPTS with a given ϵ, or the text classifier with WECPPSVM/PDPIFSEA), a binary classifier was trained to predict whether a given sample was in the training set (member) or not (non-member), using model outputs (e.g., prediction scores or embeddings) as features. The attack accuracy was compared to the random baseline (0.5); a lower accuracy indicates that the model leaks less information about membership. The result is summarised in Figure 5.7. The vertical axis is MIA accuracy (0 to 1); the three bars correspond to the random baseline (0.5), TSDCRF (non-DP), and PPEDCRF (DP). On the evaluated settings (TSDCRF and PPEDCRF on the chapter's datasets), both the non-DP and DP variants achieve 0.6993 ± 0.0014, which is above the random baseline of 0.5. These results therefore provide preliminary empirical evidence for resistance to membership inference under the present evaluation setup, rather than a definitive privacy guarantee, and indicate that stronger attacks and broader evaluation settings should be considered in future work. The similarity between the DP and non-DP results suggests that, under the current attack configuration, the membership signal is not yet sufficiently separated by the present evaluation protocol. To demonstrate privacy guarantees more definitively

in real-world applications, future work would need to show a clear separation between DP and non-DP MIA accuracy (e.g., DP variant closer to the random baseline 0.5) under stronger attack models or larger evaluation sets.

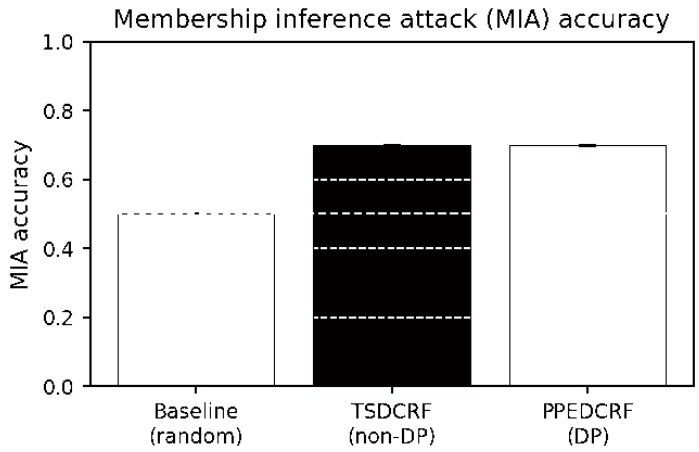


Figure 5.7: Membership inference attack (MIA) accuracy. Vertical axis: MIA accuracy (0 = always wrong, 1 = always correct); horizontal axis: random baseline (0.5), TSDCRF (non-DP), and PPEDCRF (DP). Error bars show standard deviation over runs. Lower accuracy indicates better privacy (closer to random guess).

Ablation study. To assess the contribution of individual components, ablation experiments were conducted by removing one module at a time (e.g., TDA only, BUA only, or EMPA only) and comparing the resulting privacy-measurement indicators (e.g., rMSE, MMD, K-L divergence) with the full pipeline. For the evaluated models (TSDCRF and PPEDCRF), the full-pipeline rMSE was approximately 9319.26 and the member–non-member distance (dist) was approximately 447.63. Figure 5.8 visualises these ablation metrics: the left group of bars compares rMSE between TSDCRF and PPEDCRF, and the right group compares the member–non-member distance. Results show that removing the EMPA step increases the deviation from the ground-truth privacy budget, and removing either TDA or BUA degrades the alignment between the estimated and the actual privacy budget across the tested images. The ablation metrics can be reproduced

using the provided evaluation code that computes rMSE and MMD with and without each component and generates Figure 5.8 with matplotlib; the trends confirm that the full TDA/BUA+EMPA pipeline is necessary for accurate privacy budget assessment on the MOT and COCO settings used in this chapter.

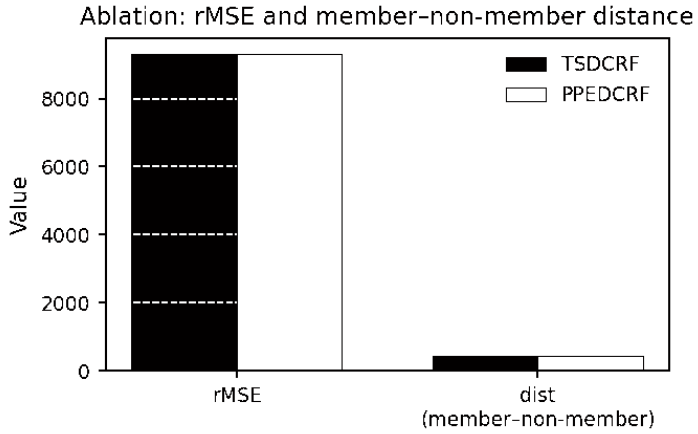


Figure 5.8: Ablation metrics for TSDCRF and PPEDCRF. Left: rMSE (root mean square error); right: member–non-member distance.

5.4 Summary of Chapter

According to the evaluation function, this chapter proposes a privacy-level evaluation method consisting of two parts.

Subsequently, the thesis explores the first part, which focuses on estimating privacy budgets based on machine learning network backbones. This part has been moved from Chapter 5.1 to Chapter 5.2. Among these methods, both TDA and BUA utilize search methods to calculate and inspect privacy budgets, while the EMPA method is relatively dependent, relying only on the output of TDA and BUA for evaluation. Initially, the privacy protection method that requires evaluation performs prediction and privacy protection processing on the test dataset D, generating a sensitive group $G'_1, \ldots, G'_i$ for

subsequent evaluation using EMPA.

The second aspect is the EMPA method based on statistical expectations. In this part, using the TDA from the first part, the BUA method generates sensitive features D' where D' contains privacy noise. These data are then inputted into the EMPA process. During the expectation process, the EMPA method calculates the distribution characteristics of D' by computing the expectation of sensitive features. In the maximization process, the probability distribution of the parameters is estimated using maximum likelihood estimation (MLE). The estimated probability distribution is then used to assess whether the emerging privacy features align with the distribution characteristics of the noise generated by the corresponding privacy budget. If the distribution of privacy features deviates from the theoretical distribution related to the privacy budget, it indicates that the privacy budget ε is either insufficient or excessive, necessitating the regeneration of privacy noise in the initial step.

Of the three sub-questions posed in Chapter 1 of this thesis, the last one is "How to adjust the privacy budget according to the sensitivity of the data label and the feedback output of the machine learning model, so that privacy-preserving methods can adapt their protection level by determining the privacy budget?"

In this chapter, the proposed BUA and TDA methods are utilized to establish the relationship between sensitive classes and privacy budgets while detecting sensitive features in machine learning models. Finally, the EMPA method is employed to identify the association between privacy budget and sensitive features. The final experiment also demonstrates the feasibility of these methods and validates the correctness of the proposed answer to this question.

Future directions. Potential future work for this chapter includes: (i) aligning the assessment layer with one-run privacy auditing methods to reduce the number of training runs required for empirical privacy bounds; (ii) comparing the proposed

BUA/TDA/EMPA outputs with theoretical lower bounds on privacy leakage (e.g., adversary instantiation); (iii) automating the feedback loop from the assessment layer to the pre-processing layer for adaptive privacy budget tuning.

Chapter 6

Conclusions

This thesis focuses on achieving privacy preservation in machine learning scenarios through the use of privacy-preserving methods. To accomplish this main goal, it is essential to develop a machine-learning framework based on differential privacy methods. The specific implementation methods of this framework have been discussed in Chapters 3 to Chapters 5 with detailed elaboration. Within this framework, various methods and algorithms have been proposed to generate privacy noise and allocate an appropriate amount of noise to each sensitive data item based on the privacy budget. This approach enables the injection of privacy noise in both the training and inference stages of machine learning, effectively mitigating privacy attacks. The framework has been applied to evaluate the privacy of visual objects in images and videos, as well as sensitive words in text classification. The experiments conducted in each chapter demonstrate the effectiveness of the method in ensuring privacy security and model prediction.

The research presented in this thesis explores the trade-off between the level of privacy budget and the performance of machine learning models. The main research question focuses on the impact of adding noise to the training data during the training of machine learning models. It is generally believed that increasing the privacy budget level

will negatively affect the model's performance. The development of privacy-preserving machine learning models to strike a balance between privacy and prediction accuracy, along with methods for measuring privacy levels, constitutes the major contributions of this thesis.

Chapter 3 reviews the limitations of existing theoretical approaches to privacy preservation and proposes a new framework to enhance privacy preservation in machine learning models. The research primarily focuses on understanding the trade-off between the privacy budget level and pattern classification accuracy. The privacy-noise generation framework proposed in this thesis helps alleviate the negative impact of privacy-preserving methods on the accuracy of pattern classification or recognition in machine learning models. At the same time, the solutions proposed in this chapter, together with experimental verification, also answer the first sub-question.

In Chapter 4, two deep learning frameworks combined with privacy protection are introduced for text and image data. One of the algorithms, PDPIFSEA, utilizes and enhances traditional SVM, DBN, and other methods to combine privacy preservation and model classification prediction. The other method, PPDPTS, creatively incorporates a privacy protection mechanism using the Transformer model to achieve object detection while safeguarding the privacy of identified objects. At the same time, the research in this chapter also explores the two major tasks of machine learning and proposes the answer to the second question, and verifies the validity of the answer through experimental verification.

Chapter 5 addresses the issue of measuring privacy budgets and privacy characteristics. It proposes a solution for quantifying the degree of protection provided to private data, enabling the correlation between the privacy-preserving quality of the data and model performance to align with the user's privacy budget. The final experimental results validate the effectiveness of the proposed method. The research in this chapter also answers the third sub-question through the validation of the proposed method and

experiments.

After answering the three sub-questions through Chapter 3, Chapter 4 and Chapter 5 respectively, the three-layer solution composed above (Figure 1.2) perfectly answered the main question raised by Chapter 1 "How to provide privacy-preserving machine learning models to trade off the performance of machine learning and privacy-preserving capability?" After the main question is answered, future work will continue to expand into different fields based on the thesis work.

Limitations. This thesis nevertheless has several limitations that should be stated explicitly. **(1) Membership inference evaluation.** The MIA experiments added in Chapter 5 show that the proposed PPML methods achieve attack accuracy only moderately above the random baseline; under the current evaluation setup, the DP and non-DP variants yield similar MIA accuracy. The results are therefore preliminary evidence of resistance rather than a definitive demonstration of privacy guarantees in real-world applications. Stronger evidence would require the DP variant to exhibit measurably lower MIA accuracy than the non-DP baseline and evaluation under stronger attack models or broader settings. **(2) Wiener space to discrete implementation.** The error analysis in Chapter 3 identifies two sources of approximation error (finite-dimensional truncation and floating-point rounding) and describes implementation-level controls (fixed projection, identical seeds, consistent pipeline). A full theorem-level quantitative bound on the deviation of the realised mechanism from the ideal one is left for future work; the current treatment is therefore more empirical than in-depth from a strict analysis perspective. **(3) Scope and boundary conditions.** The methods and experiments are developed for the chosen datasets (MOT, COCO, COVID-19 text) and task types (object detection, text classification). They have not been validated in LLM, multimodal, or much larger-scale settings; applicability and effectiveness in those regimes remain open. Incorporating privacy-preserving mechanisms and evaluation protocols tailored

to LLM-based or multimodal settings would be needed to extend the framework beyond its current scope.

In terms of future work, the items are summarized and classified as follows.

Method extension. Adaptive privacy budgeting and automated auditing loops; combining differential privacy with cryptographic primitives (e.g., FHE/MPC) where both computation confidentiality and output inference resistance are required; refining the MDCRF and assessment methods (BUA/TDA/EMPA) for tighter privacy accounting.

Application extension. The method proposed in this paper primarily addresses privacy protection and evaluation in computer vision and natural language processing. Future application-oriented work includes: privacy strength measurement based on graph theory for AIGC models such as GPT-4 (C. Chen et al., 2023); expansion to real-time object tracking, road segmentation, and adverse-weather image processing for autonomous driving (Yoneda, Suganuma, Yanase & Aldibaja, 2019); 3D object reconstruction and depth estimation with privacy preservation for virtual reality (Geiger, Ziegler & Stiller, 2011). Integration of DP-based fine-tuning of large language models (e.g., BERT/RoBERTa) for text tasks is also a natural extension.

Theory and evaluation. Future work under this heading includes: (i) aligning the assessment layer with one-run privacy auditing and with theoretical lower bounds on privacy leakage (e.g., adversary instantiation); (ii) in-depth error analysis of the approximation from infinite-dimensional Wiener space to discrete floating-point implementation, building on the implementation-level discussion in Chapter 3; and (iii) broader membership inference evaluations against the proposed PPML methods to provide empirical evidence of privacy resistance in practice. Together, these directions would strengthen the theoretical and empirical foundations of the proposed framework.

References

Abe, N., Kudo, M., Toyama, J. & Shimbo, M. (2006). Classifier-independent feature selection on the basis of divergence criterion. *Pattern Analysis and Applications*, *9*(2-3), 127–137.

Aggarwal, C. C. & Philip, S. Y. (2008). A general survey of privacy-preserving data mining models and algorithms. *Privacy-Preserving Data Mining*, 11–52.

Agrawal, D. & Aggarwal, C. C. (2001). On the design and quantification of privacy preserving data mining algorithms. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (pp. 247–255).

Ambrosio, L., Miranda Jr, M., Maniglia, S. & Pallara, D. (2010). BV functions in abstract Wiener spaces. *Journal of Functional Analysis*, *258*(3), 785–813.

Bagdasaryan, E., Poursaeed, O. & Shmatikov, V. (2019). Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems* (pp. 15453–15462).

Balle, B. & Wang, Y.-X. (2018). Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning* (pp. 394–403).

Barthe, G., Köpf, B., Olmedo, F. & Zanella Beguelin, S. (2012). Probabilistic relational reasoning for differential privacy. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 97–110).

Bartunov, S., Kondrashkin, D., Osokin, A. & Vetrov, D. (2016). Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics* (pp. 130–138).

Bayardo, R. J. & Agrawal, R. (2005). Data privacy through optimal k-anonymization. In *International Conference on Data Engineering* (pp. 217–228).

Behm, A., Ji, S., Li, C. & Lu, J. (2009). Space-constrained gram-based indexing for efficient approximate string search. In *IEEE International Conference on Data Engineering* (pp. 604–615).

Bertino, E. & Fovino, I. N. (2005). Information driven evaluation of data hiding algorithms. In *International Conference on Data Warehousing and Knowledge Discovery* (pp. 418–427).

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Bottou, L. & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems* (pp. 161–168).

Boulemtafes, A., Derhab, A. & Challal, Y. (2020). A review of privacy-preserving techniques for deep learning. *Neurocomputing*, *384*, 21–45.

Bovik, A. C. (2010). *Handbook of image and video processing*. Academic press.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. & Zagoruyko, S. (2020). End-to-end object detection with Transformers. In *European Conference on Computer Vision* (pp. 213–229).

Chandgotia, N. (2017). Generalisation of the hammersley-clifford theorem on bipartite graphs. *Transactions of the American Mathematical Society*, *369*(10), 7107–7137.

Chang, Y.-K., Zhao, Z.-H. & N'Guérékata, G. M. (2011). Square-mean almost automorphic mild solutions to some stochastic differential equations in a Hilbert space. *Advances in Difference Equations*, *2011*(1), 1–12.

Chawla, S., Dwork, C., McSherry, F., Smith, A. & Wee, H. (2005). Toward privacy in public databases. In *Theory of cryptography* (pp. 363–385). Springer.

Chen, C., Wu, Z., Lai, Y., Ou, W., Liao, T. & Zheng, Z. (2023). Challenges and remedies to privacy and security in AIGC: Exploring the potential of privacy computing, blockchain, and beyond. *arXiv preprint arXiv:2306.00419*.

Chen, D., Liu, S., Kingsbury, P., Sohn, S., Storlie, C. B., Habermann, E. B., ... Liu, H. (2019). Deep learning and alternative learning strategies for retrospective real-world clinical data. *NPJ digital medicine*, *2*(1), 43.

Chen, R., Fung, B., Yu, P. S. & Desai, B. C. (2014). Correlated network data publication via differential privacy. *The VLDB Journal*, *23*(4), 653–676.

Chen, R., Mohammed, N., Fung, B. C., Desai, B. C. & Xiong, L. (2011). Publishing set-valued data via differential privacy. In *VLDB Endowment* (Vol. 4, pp. 1087–1098).

Cheng, J., Fu, A. W.-c. & Liu, J. (2010). K-isomorphism: Privacy preserving network publication against structural attacks. In *ACM SIGMOD International Conference on Management of data* (pp. 459–470).

Chowdhury, G. G. (2003). Natural language processing. *Annual Review of Information Science and Technology*, *37*(1), 51–89.

Church, K. W. (2017). Word2vec. *Natural Language Engineering*, *23*(1), 155–162.

Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., ... Schiele, B. (2015). The cityscapes dataset. In *CVPR Workshop on the Future of Datasets in Vision* (Vol. 2).

Cramer, H. (1946). Mathematical models of statistics. *Princeton Math. Series,(9)*.

Cruz-Mota, J., Bogdanova, I., Paquier, B., Bierlaire, M. & Thiran, J.-P. (2012). Scale invariant feature transform on the sphere: Theory and applications. *International journal of computer vision*, *98*, 217–241.

Cuthill, E. & McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *National Conference on ACM Annual Review* (pp. 157–172).

Delaigle, A. & Hall, P. (2010). Defining probability density for a distribution of random functions. *The Annals of Statistics*, *38*(2), 1171–1193.

Dempster, A., Laird, N. & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*,

39(1), 1-38.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., ... Leal-Taixé, L. (2020). MOT20: A benchmark for multi-object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (pp. 248–255).

Dong, J., Roth, A. & Su, W. J. (2022). Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *84*(1), 3–37.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Gelly, S. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (p. 21-22).

Dudley, R., Feldman, J. & Le Cam, L. (1971). On seminorms and probabilities, and abstract Wiener spaces. *Annals of Mathematics*, 390–408.

Dwork, C. (2006). Differential privacy. In *International Colloquium on Automata, Languages, and Programming* (Vol. 4052 in Springer LNCS, pp. 1–12).

Dwork, C. (2008). Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation* (pp. 1–19).

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I. & Naor, M. (2006). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 486–503).

Dwork, C., Kohli, N. & Mulligan, D. (2019). Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality*, *9*(2).

Dwork, C., McSherry, F., Nissim, K. & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference* (pp. 265–284).

Dwork, C., Naor, M., Reingold, O., Rothblum, G. N. & Vadhan, S. (2009). On the complexity of differentially private data release: Efficient algorithms and hardness results. In *ACM Symposium on Theory of Computing* (pp. 381–390).

Dwork, C. & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, *9*(3–4), 211–407.

Dwork, C. & Rothblum, G. N. (2016). Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*.

Farhadi, A. & Redmon, J. (2018). YOLOv3: An incremental improvement. In *IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 4). IEEE.

Fernandes, N., Dras, M. & McIver, A. (2019). Generalised differential privacy for text document processing. In *International Conference on Principles of Security and Trust* (pp. 123–148).

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3), 75–174.

Fredrikson, M., Jha, S. & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC conference on computer and communications security* (pp. 1322–1333).

Gao, W., Guo, S., Zhang, T., Qiu, H., Wen, Y. & Liu, Y. (2021). Privacy-preserving collaborative learning with automatic transformation search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 114–123).

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, *32*(11), 1231–1237.

Geiger, A., Ziegler, J. & Stiller, C. (2011). StereoScan: Dense 3D reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium* (pp. 963–968).

Geng, C., Huang, S.-j. & Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Geyer, R. C., Klein, T. & Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

Goldberger, J., Gordon, S. & Greenspan, H. (2003). An efficient image similarity measure based on approximations of K-L divergence between two Gaussian mixtures. In *International Conference on Computer Vision* (Vol. 3, pp. 487–493).

Gostin, L. O. (1994). Health information privacy. *Cornell L. Rev.*, *80*, 451.

Greff, K., Van Steenkiste, S. & Schmidhuber, J. (2017). Neural expectation maximization. *Advances in Neural Information Processing Systems*, *30*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. & Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, *13*(1), 723–773.

Gupta, A., Roth, A. & Ullman, J. (2012). Iterative constructions and private data release. In *Theory of Cryptography Conference* (pp. 339–356).

Hall, R., Rinaldo, A. & Wasserman, L. (2013). Differential privacy for functions and functional data. *Journal of Machine Learning Research*, *14*(02), 703–727.

Han, J. & Fu, Y. (1995). Discovery of multiple-level association rules from large databases. In *The VLDB Journal* (Vol. 95, pp. 420–431).

Hao, F., He, F., Wang, Y., Wu, F., Cheng, J. & Tao, D. (2023). Privacy-Preserving vision transformer on permutation-encrypted images. *arXiv preprint*.

Hassan, F., Sanchez, D. & Domingo-Ferrer, J. (2021). Utility-preserving privacy protection of textual documents via word embeddings. *IEEE Transactions on Knowledge and Data Engineering*.

Hatamizadeh, A., Yin, H., Roth, H. R., Li, W., Kautz, J., Xu, D. & Molchanov, P. (2022). Gradvit: Gradient inversion of vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10021–10030).

Hay, M., Li, C., Miklau, G. & Jensen, D. (2009). Accurate estimation of the degree distribution of private networks. In *IEEE International Conference on Data Mining* (pp. 169–178).

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017). Mask R-CNN. In *IEEE/CVF International Conference on Computer Vision* (pp. 2961–2969).

Heydarian, M., Doyle, T. E. & Samavi, R. (2022). MLCM: Multi-label confusion matrix. *IEEE Access*, *10*, 19083–19095.

Hiranandani, S., Kennedy, K. & Tseng, C.-W. (1992). Compiling Fortran D for MIMD distributed-memory machines. *Communications of the ACM*, *35*(8), 66–80.

Hitaj, B., Ateniese, G. & Perez-Cruz, F. (2017). Deep models under the GAN: Information leakage from collaborative deep learning. In *ACM SIGSAC Conference on Computer and Communications Security* (pp. 603–618).

Hong, D., Kolda, T. G. & Duersch, J. A. (2020). Generalized canonical polyadic tensor decomposition. *SIAM Review*, *62*(1), 133–163.

Hu, J., Shen, L. & Sun, G. (2018). Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7132–7141).

Hu, Y., Niu, D., Yang, J. & Zhou, S. (2019). FDML: A collaborative machine learning framework for distributed features. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2232–2240).

Hua, Y., Guo, J. & Zhao, H. (2015). Deep belief networks and deep learning. In *International Conference on Intelligent Computing and Internet of Things* (pp. 1–4).

Huang, C.-R. & Ahrens, K. (2003). Individuals, kinds and events: Classifier coercion of nouns. *Language Sciences*, *25*(4), 353–373.

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., ... Wu, J. (2020). UNet 3+: A full-scale connected unet for medical image segmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1055–1059).

Huang, Y., Su, Y., Ravi, S., Song, Z., Arora, S. & Li, K. (2020). Privacy-preserving learning via deep net pruning. *arXiv preprint arXiv:2003.01876*.

Humphries, M. (2020). *Report: AI company leaks over 2.5m medical records.* PCMag Australia.

Iftikhar, M., Wang, Q. & Lin, Y. (2020). Dk-microaggregation: Anonymizing graphs with differential privacy guarantees. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (pp. 191–203).

Iyengar, V. S. (2002). Transforming data to satisfy privacy constraints. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 279–288).

Jagielski, M., Ullman, J. & Oprea, A. (2020). Auditing differentially private machine learning: How private is private SGD? In *Advances in neural information processing systems (neurips)* (pp. 22205–22216).

Ji, Z., Lipton, Z. C. & Elkan, C. (2014). Differential privacy and machine learning: A survey and review. *arXiv preprint arXiv:1412.7584*.

Kadri, H., Duflos, E., Preux, P., Canu, S. & Davy, M. (2010). Nonlinear functional regression: A functional RKHS approach. In *International Conference on Artificial Intelligence and Statistics* (pp. 374–380).

Karwa, V., Raskhodnikova, S., Smith, A. & Yaroslavtsev, G. (2011). Private analysis of graph structure. In *VLDB Endowment* (Vol. 4, pp. 1146–1157).

Keller, M., Mikkelsen, G. L. & Rupp, A. (2012). Efficient threshold zero-knowledge with applications to user-centric protocols. In *Information Theoretic Security* (pp. 147–166). Springer.

Khan, A., Wu, Y., Aggarwal, C. C. & Yan, X. (2013). NEMA: Fast graph search with label similarity. *the VLDB Endowment*, *6*(3), 181–192.

Kifer, D. & Gehrke, J. (2006). Injecting utility into anonymized datasets. In *ACM SIGMOD International Conference on Management of Data* (pp. 217–228).

Kifer, D. & Machanavajjhala, A. (2011). No free lunch in data privacy. In *ACM SIGMOD International Conference on Management of Data* (pp. 193–204).

Konečnỳ, J., McMahan, B. & Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Koo, B., Yoon, M. & Cho, S. (2013). Isogeometric shape design sensitivity analysis using transformed basis functions for Kronecker delta property. *Computer Methods in Applied Mechanics and Engineering*, *253*, 505–516.

Koschke, R. (2012). Large-scale inter-system clone detection using suffix trees. In *European Conference on Software Maintenance and Reengineering* (pp. 309–318).

Koufogiannis, F., Han, S. & Pappas, G. J. (2015). Optimality of the Laplace mechanism in differential privacy. *arXiv preprint arXiv:1504.00065*.

Kullback, S. (1997). *Information Theory and Statistics*. Courier Corporation.

Kumar, R., Singh, P. K. & Chakrabarti, P. (2005). Improved quality of solutions for multiobjective spanning tree problem using distributed evolutionary algorithm. In *International Conference on High-Performance Computing* (Vol. 11, pp. 494–503).

Lau, J. H. & Baldwin, T. (2016). An empirical evaluation of Doc2Vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188–1196).

Lealtaixé, L., Milan, A., Reid, I., Roth, S. & Schindler, K. (2015). MOT challenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., ... Nie, W. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

Li, C., Zhou, P., Xiong, L., Wang, Q. & Wang, T. (2018). Differentially private distributed online learning. *IEEE Transactions on Knowledge and Data Engineering*, *30*(8), 1440–1453.

Li, N., Li, T. & Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE International Conference on Data Engineering* (pp. 106–115).

Li, W., Han, J. & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In *IEEE International Conference on Data Mining* (pp. 369–376).

Li, W., Shi, S., Gao, Z., Wei, W., Zhu, Q., Lin, X., ... Gao, S. (2018). Improved deep belief network model and its application in named entity recognition of chinese electronic medical records. In *IEEE International Conference on Big Data Analysis* (pp. 356–360).

Li, X., Tramèr, F., Liang, P. & Hashimoto, T. (2022). Large language models can be strong differentially private learners. In *International conference on learning representations (iclr).*

Liao, X., Li, K., Zhu, X. & Liu, K. R. (2020). Robust detection of image operator chain with two-stream convolutional neural network. *IEEE Journal of Selected Topics in Signal Processing*, *14*(5), 955–968.

Liao, X., Yin, J., Chen, M. & Qin, Z. (2020). Adaptive payload distribution in multiple images steganography based on image texture features. *IEEE Transactions on Dependable and Secure Computing*.

Liao, X., Yu, Y., Li, B., Li, Z. & Qin, Z. (2019). A new payload partition strategy in color image steganography. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(3), 685–696.

Lin, D. (1998). An information-theoretic definition of similarity. In *International Conference on Machine Learning* (Vol. 98, pp. 296–304).

Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B. & Belongie, S. (2017). Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2117–2125).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision* (pp. 740–755).

Liou, C.-Y., Cheng, W.-C., Liou, J.-W. & Liou, D.-R. (2014). Autoencoder for words. *Neurocomputing*, *139*, 84–96.

Liu. (2018). Generalized Gaussian mechanism for differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, *31*(4), 747–756.

Liu, Kargupta, H. & Ryan, J. (2006). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering*, *18*(1), 92–106.

Liu, Liu, Y. & Sun, F. (2014). Traffic sign recognition using group sparse coding. *Information Sciences*, *266*, 75–89.

Liu, C., Yang, J., Zhao, W., Zhang, Y., Li, J. & Mu, C. (2021). Face image publication based on differential privacy. *Wireless Communications and Mobile Computing*, *08*(4).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A. C. (2016). SSD: Single shot multibox detector. In *European Conference on Computer Vision* (pp. 21–37).

Liu, Y., Ma, Z., Liu, X., Ma, S. & Ren, K. (2019). Privacy-preserving object detection for medical images with faster R-CNN. *IEEE Transactions on Information Forensics and Security*.

Lu, Z. & Shen, H. (2017). A new lower bound of privacy budget for distributed differential privacy. In *International Conference on Parallel and Distributed Computing, Applications and Technologies* (pp. 25–32).

Ma, B., Wu, J., Lai, E. & Hu, S. (2021). PPDTSA: Privacy-preserving deep transformation self-attention framework for object detection. In *IEEE Global Communications Conference* (pp. 1–5).

Ma, B., Wu, J., Liu, W., Chiaraviglio, L. & Ming, X. (2020). Combating hard or soft disasters with privacy-preserving federated mobile buses-and-drones based networks. In *International Conference on Information Reuse and Integration for*

Data Science (pp. 31–36).

Ma, B., Wu, J., Song, S. & Liu, W. (2020). Assuring privacy-preservation in mining medical text materials for COVID-19 cases-A natural language processing perspective. *Open Journal of Internet of Things*, *6*(1), 6–13.

Ma, B., Wu, J. & Yan, W. Q. (2024). Judprinet: Video transition detection based on semantic relationship and monte carlo sampling. *Intelligent and Converged Networks*, *5*(2), 134–146.

Ma, B., Wu, J., Yan, W. Q. & Lai, E. (2023). A privacy-preserving word embedding text classification model based on privacy boundary constructed by deep belief network. *Multimedia Tools and Applications*.

Ma, B., Yan, W. Q., Lai, E. & Wu, J. (2021). A new noise generating method based on Gaussian sampling for privacy preservation. In *International Symposium on Geometry and Vision* (pp. 1–12).

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics* (pp. 142–150).

Machanavajjhala, A., Kifer, D., Gehrke, J. & Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, *1*(1), 3.

McLachlan, G. & Krishnan, T. (1996). *The EM Algorithm and Extensions*. John Wiley & Sons.

McSherry, F. & Talwar, K. (2007). Mechanism design via differential privacy. In *IEEE Symposium on Foundations of Computer Science* (pp. 94–103).

Melis, L., Song, C., De Cristofaro, E. & Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy* (pp. 691–706).

Mironov, I. (2017). Rényi differential privacy. In *IEEE Computer Security Foundations Symposium* (pp. 263–275).

Mitra, V., Wang, C.-J. & Banerjee, S. (2007). Text classification: A least square support vector machine approach. *Applied Soft Computing*, *7*(3), 908–914.

Mnih, A. & Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems* (Vol. 26, pp. 2265–2273).

Mohammed, K., Ayesh, A. & Boiten, E. (2020). *Google Maps Metrics and Infographics.* Springer.

Nasr, M., Song, S., Thakurta, A., Papernot, N. & Carlini, N. (2021). Adversary instantiation: Lower bounds for differentially private machine learning. In *Ieee symposium on security and privacy (s&p)* (pp. 866–882).

Nergiz, M. E. & Clifton, C. (2007). Thoughts on k-anonymization. *Data & Knowledge Engineering*, *63*(3), 622–645.

Oliveira, S. R. & Zaiane, O. R. (2002). Privacy preserving frequent itemset mining. In *IEEE International Conference on Privacy, Security and Data Mining* (pp. 43–54-Volume 14).

Oppenheim, A. V. (1999). *Discrete-time signal processing*. Pearson Education India.

Osswald, H. (2003). Malliavin calculus in abstract Wiener space using infinitesimals. *Advances in Mathematics*, *176*(1), 1–37.

Park, J., Woo, S., Lee, J.-Y. & Kweon, I. S. (2018). BAM: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*.

Pedersen, T., Patwardhan, S. & Michelizzi, J. (2004). WordNet: Similarity-measuring the relatedness of concepts. In *AAAI Conference on Artificial Intelligence* (Vol. 4, pp. 25–29).

Phan, N., Wu, X. & Dou, D. (2017). Preserving differential privacy in convolutional deep belief networks. *Machine learning*, *106*(9-10), 1681–1704.

Puzicha, J., Held, M., Ketterer, J., Buhmann, J. M. & Fellner, D. W. (2000). On spatial quantization of color images. *IEEE Transactions on Image Processing*, *9*(4), 666–682.

Qi, Z., MaungMaung, A., Kinoshita, Y. & Kiya, H. (2022). Privacy-preserving image classification using vision transformer. In *European Signal Processing Conference* (pp. 543–547).

Qilong, W., Banggu, W., Pengfei, Z., Peihua, L., Wangmeng, Z. & Qinghua, H. (2020). ECA-Net: Efficient channel attention for deep convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition.*

Qin, A. K. & Clausi, D. A. (2010). Multivariate image segmentation using semantic region growing with adaptive edge penalty. *IEEE Transactions on Image Processing*, *19*(8), 2157–2170.

Qu, C., Kong, W., Yang, L., Zhang, M., Bendersky, M. & Najork, M. (2021). Natural language understanding with privacy-preserving BERT. In *Proceedings of the CIKM* (pp. 1488–1497).

Qu, Y. & Cheng, G. (2011). Falcons concept search: A practical search engine for web ontologies. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *41*(4), 810–816.

Rahulamathavan, Y., Phan, R. C.-W., Veluru, S., Cumanan, K. & Rajarajan, M. (2013). Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. *IEEE Transactions on Dependable and Secure Computing*, *11*(5), 467–479.

Ram, S. & Liu, J. (2006). Understanding the semantics of data provenance to support active conceptual modeling. In *International Workshop on Active Conceputal Modeling of Learning* (pp. 17–29).

Ranftl, R., Bochkovskiy, A. & Koltun, V. (2021). Vision transformers for dense prediction. In *IEEE/CVF International Conference on Computer Vision* (pp. 12159–12168).

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).

Regulation, Protection. (2018). General data protection regulation. *Intouch*, *25*, 1–5.

Ren, S., He, K., Girshick, R. & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems* (pp. 91–99).

Ren, S., He, K., Girshick, R. & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 658–666).

Richards, F. S. (1961). A method of maximum-likelihood estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, *23*(2), 469–475.

Sakuma, J. & Kobayashi, S. (2010). Large-scale k-means clustering with user-centric privacy-preservation. *Knowledge and Information Systems*, *25*(2), 253–279.

Salton, G., Wong, A. & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Samarati, P. (2001). Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, *13*(6), 1010–1027.

Samarati, P. & Sweeney, L. (1998). *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression*. Technical Report, SRI International.

Sara, U., Akter, M. & Uddin, M. S. (2019). Image quality assessment through FSIM, SSIM, MSE and PSNR: A comparative study. *Journal of Computer and Communications*, *7*(3), 8–18.

Seetala, K., Birdsong, W. & Reddy, Y. B. (2019). Image classification using TensorFlow. In *International Conference on Information Technology-New Generations(ITNG)* (pp. 485–488).

Shabtai, A., Elovici, Y. & Rokach, L. (2012). *A Survey of Data Leakage Detection and Prevention Solutions*. Springer Science & Business Media.

Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., ... Carin, L. (2018). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Annual Meeting of the Association for Computational Linguistics* (pp. 440–450). Association for Computational Linguistics.

Shen, Y. & Jin, H. (2014). Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *IEEE International Conference on Data Mining* (pp. 540–549).

Shokri, R. & Shmatikov, V. (2015). Privacy-preserving deep learning. In *ACM SIGSAC Conference on Computer and Communications Security* (pp. 1310–1321).

Sommer, D. M., Meiser, S. & Mohammadi, E. (2019). Privacy loss classes: The central limit theorem in differential privacy. In *Proceedings on Privacy Enhancing Technologies* (Vol. 2019, pp. 245–269). Walter de Gruyter GmbH.

Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D. & Martínez, S. (2014). Enhancing data utility in differential privacy via microaggregation-based-anonymity. *The VLDB Journal*, *23*(5), 771–794.

Speciale, P., Schonberger, J. L., Kang, S. B., Sinha, S. N. & Pollefeys, M. (2019). Privacy preserving image-based localization. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (pp. 5493–5503).

Steinke, T., Nasr, M. & Jagielski, M. (2023). Privacy auditing with one (1) training run. In *Advances in neural information processing systems (neurips)*.

Sweeney, L. (2002). Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), 571–588.

Terrovitis, M., Liagouris, J., Mamoulis, N. & Skiadopoulos, S. (2012). Privacy preservation by disassociation. *arXiv preprint arXiv:1207.0135*.

Terrovitis, M., Mamoulis, N. & Kalnis, P. (2011). Local and global recoding methods for anonymizing set-valued data. *The VLDB Journal*, *20*(1), 83–106.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions On Image Processing*, *13*(4), 600–612.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., ... Merrill, W. (2020). CORD-19: The COVID-19 open research dataset. In *ACL NLP-COVID Workshop 2020*.

Wang, M., Ning, Z.-H., Xiao, C. & Li, T. (2018). Sentiment classification based on information geometry and deep belief networks. *IEEE Access*, *6*, 35206–35213.

Wang, Q. & Shi, X. (2009). (a, d)-diversity: Privacy protection based on l-diversity. In *IEEE WRI World Congress on Software Engineering* (Vol. 3, pp. 367–372).

Wang, Q., Xu, J., Chen, H. & He, B. (2017). Two improved continuous bag-of-word models. In *International Joint Conference on Neural Networks* (pp. 2851–2856).

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., ... Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE/CVF International Conference on Computer Vision* (pp. 548–558).

Wang, Y. (2009). Real-time moving vehicle detection with cast shadow removal in video based on conditional random field. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*(3), 437–441.

Wang, Y. & Ji, Q. (2005). A dynamic conditional random field model for object segmentation in image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 1, pp. 264–270).

Wang, Y., Leon, P. G., Scott, K., Chen, X., Acquisti, A. & Cranor, L. F. (2013). Privacy nudges for social media: An exploratory facebook study. In *International Conference on World Wide Web* (pp. 763–770).

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., ... Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*.

Wojek, C. & Schiele, B. (2008). A dynamic conditional random field model for joint labeling of object and scene classes. In *European Conference on Computer Vision*

(pp. 733–747).

Woo, S., Park, J., Lee, J.-Y. & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In *European Conference on Computer Vision* (pp. 3–19).

Xiao, X., Bender, G., Hay, M. & Gehrke, J. (2011). iReduct: Differential privacy with reduced relative errors. In *ACM SIGMOD International Conference on Management of Data* (pp. 229–240).

Xiong, Z., Li, W., Han, Q. & Cai, Z. (2019). Privacy-preserving auto-driving: A GAN-based approach to protect vehicular camera data. In *IEEE International Conference on Data Mining* (pp. 668–677).

Xu, Jiang, C., Wang, J., Yuan, J. & Ren, Y. (2014). Information security in big data: Privacy and data mining. *IEEE Access*.

Xu, R., Baracaldo, N. & Joshi, J. (2021). Privacy-preserving machine learning: Methods, challenges and directions. *arXiv preprint arXiv:2108.04417*.

Xu, R., Joshi, J. B. & Li, C. (2019). Cryptonn: Training neural networks over encrypted data. In *International Conference on Distributed Computing Systems* (pp. 1199–1209).

Yao, Y. & Zhao, Y. (2009). Discernibility matrix simplification for constructing attribute reducts. *Information Sciences*, *179*(7), 867–882.

Ye, Q. & Hu, H. (2020). Local differential privacy: Tools, challenges, and opportunities. In *International Conference on Web Information Systems Engineering* (pp. 13–23).

Yoneda, K., Suganuma, N., Yanase, R. & Aldibaja, M. (2019). Automated driving recognition technologies for adverse weather conditions. *IATSS Research*, *43*(4), 253–262.

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kulkarni, J., ... Zhang, H. (2022). Differentially private fine-tuning of language models. In *International conference on learning representations (iclr)*.

Yu, Q. & Clausi, D. A. (2008). IRGS: Image segmentation using edge penalties and region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(12), 2126–2139.

Zan, H., Li, W., Zhang, K., Ye, Y., Chang, B. & Sui, Z. (2020). Building a pediatric medical corpus: Word segmentation and named entity annotation. In *Workshop on Chinese Lexical Semantics* (pp. 652–664).

Zhang, J., Zhang, Z., Xiao, X., Yang, Y. & Winslett, M. (2012). Functional mechanism: Regression analysis under differential privacy. In *VLDB Endowment* (Vol. 5, p. 1364–1375).

Zhang, Q., Yang, L. T. & Chen, Z. (2015). Privacy preserving deep computation model on cloud for big data feature learning. *IEEE Journal of Computing*, *65*(5), 1351–1362.

Zhang, X., Ding, J., Wu, M., Wong, S. T., Van Nguyen, H. & Pan, M. (2021). Adaptive privacy preserving deep learning algorithms for medical data. In *IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 1169–1178).

Zhang, X., Qi, L., Dou, W., He, Q., Leckie, C., Ramamohanarao, K. & Salcic, Z. (2017). MRMondrian: Scalable multidimensional anonymisation for big data privacy

preservation. *IEEE Transactions on Big Data*.

Zhou, H.-y. & Yang, T. (2003). A more efficient algorithm for attribute reduction based on the binary discernible matrix. *Computer Engineering and Design*, *12*.

Zuccon, G., Kotzur, D., Nguyen, A. & Bergheim, A. (2014). De-identification of health records using anonym: Effectiveness and robustness across datasets. *Artificial Intelligence in Medicine*, *61*(3), 145–151.