UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Classification Modelling for Encrypted Network Traffic Captured in Air

by

Yi Huang

A Thesis Submitted
in Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2023

Certificate of Original Authorship

I, Yi Huang, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and IT at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 17/11/2023

ABSTRACT

Classification Modelling for Encrypted Network Traffic Captured in Air

by

Yi Huang

End-to-end encrypted traffic predominates the network traffic on the Internet to protect personal privacy and information security. However, unveiling encrypted harmful content (e.g., fake news and propaganda videos) and detecting malicious activities such as distributing copyrighted material become difficult to achieve through traditional network surveillance.

To address network surveillance problems triggered by traffic encryption, this thesis researches on identifying (i.e., classifying) the content of encrypted network traffic captured in air from three aspects, namely classification in the closed world, novelty detection in the open world, and few-shot learning for dynamically changing targets in the open world. Recent research has verified that machine learning and deep learning methods contribute to extracting underlying patterns from encrypted network traffic, and deep learning can perform well without elaborate feature engineering. Therefore, this thesis focuses on developing deep learning based models and strategies to tackle corresponding issues.

First, for closed world classification, it has been evaluated that it is feasible to identify the content of the encrypted network traffic captured through the Network Layer. Nonetheless, it is still challenging to identify the content of encrypted network traffic passively sniffed in air (Data Link Layer) due to the lack of packet header information in upper protocol layers. In this thesis, the encrypted network traffic is captured in air without connecting to the WiFi network. I evaluate that it is possible to classify the content of these traffic samples with the raw data and frame-level features by leveraging a lightweight deep learning model.

Second, to use classification models in practice, the proposed model should be able to identify test samples as the target traffic (i.e., inliers) or background traffic (i.e., outliers). Novelty detection is a promising technique to detect outliers located at any location, such as abnormalities (i.e., far distance outliers) and novel patterns (i.e., close distance outliers). Many different novelty detection approaches have been proposed in the literature, but they generally focus on detecting one specific type of outlier, e.g., far or close distance outliers. However, in the real world, it is difficult to measure in advance whether the distance between outliers and inliers is far or close. In this thesis, a new unified model, named Calibrated Reconstruction Based Adversarial AutoEncoder (CRAAE) model, is proposed to implement location agnostic outlier detection. The key idea is to integrate implicit and explicit confidence calibration strategies into a reconstruction based model. I leverage the category information disentangled from feature space to calibrate the decision metric (i.e., reconstruction error) constructed in the original data space for building a more accurate decision boundary. CRAAE also adds Uniform or Dirichlet noise into the artificial outlier generation process to represent various outliers. Experimental results show that CRAAE can outperform state-of-the-art unified models and achieve similar performance to other methods that only address close or far distance outlier detection.

Finally, to meet higher practical requirements, several approaches are proposed to address open-set recognition problems on dynamically changing tasks (e.g., changes in the target website or video list). While few-shot learning and open-set recognition methods have been proposed for domains such as computer vision, *few-shot open-set recognition* for encrypted network traffic remains an unexplored area. I propose a task adaptive Siamese Neural Network (SNN) for open-set recognition. My contributions are three-fold: First, introducing generated positive and negative pairs into the SNN training process to shape a more precise similarity boundary through bidirectional dropout data augmentation. Second, utilising Dirichlet Process Gaussian Mixture Model (DPGMM) distribution to fit the similarity scores of the negative pairs constructed by the support set of each query task, and creating

a new open-set recognition metric. Third, constructing a hierarchical cross entropy loss by leveraging the extracted features at coarse and fine granular levels to improve the confidence of the similarity score. Extensive experiments on a network traffic dataset and the Omniglot dataset demonstrate the superiority and generalizability of my proposed approach.

This thesis started with application research and drilled down to meet more comprehensive scenarios with better performance. To address the specific application scenarios, I developed deep learning techniques in novelty detection and few-shot learning areas, which are demonstrated not only to solve the problems in the network traffic domain but also to be transferred to other fields such as image processing.

Dissertation directed by Dr. Christy Jie Liang & Prof. Richard Yi Da Xu
School of Computer Science

Acknowledgements

When I finally stood at the other side of my PhD career, all kinds of feelings well up in my heart. Although there were failures and pains along the way, every bit of success was so precious when looking back. 2018 was my second year immigrating to Australia, and that year I decided to apply for my PhD. I still clearly remember how I felt sitting in the lounge area outside my supervisor's office, waiting for the interview. Looking at the people around me, I yearned to be one of them, but I was not so confident about the interview. I worried that my supervisor might not give me the opportunity. Before that, I never imagined that I would have the chance to study for a PhD, and I never imagined that I could get a doctorate at the age of forty.

First of all, I would like to express my heartfelt thanks to my supervisors, Richard Xu and Christy Liang, for giving me this opportunity, allowing me to realize my dream, and adding rich color to my life journey. I sincerely thank you for your continuous guidance and encouragement during my PhD research. Academically, you have guided me and developed my creative mind; you have also given me a lot of sincere advice on how to communicate, which has given me a lot of valuable experience for my future career. In addition, you also considered my financial situation and tried your best to get me various scholarships.

I would also like to thank my project partners. Thanks to the industry experts Guillaume Jourjon, Adriel Cheng, and Darren Webb from Data 61 and Defense Science and Technology (DST) for funding the project and providing advice from the industry perspective. Thanks to the academics, Dr. Suranga Seneviratne and Dr. Kanchana Thilakarathnathe from the University of Sydney for helping me polish my papers. I also thank them for their patience and kindness, which enabled me to express my ideas comfortably in the meeting when I was not confident with the language in the early stage.

Thanks to my research partner Ying Li. During the first two years of my PhD study, we were together almost every day, discussing and researching in academics and sharing emotions of joy and disappointment in life. We cooperated and published some articles together. Some of the content of these articles contributed to chapters 3 of this thesis. Due to the pandemic in 2019, we cannot continue to communicate every day. Still, we often discuss issues through the Internet, inspiring each other to move forward and maintaining close cooperation.

Thanks to Dr. Minqi Li for helping me understand non-parametric Bayesian more deeply, and thanks to Tim for reproducing two baseline methods in the Deep Bypass project.

Thanks to the other colleagues in our lab (Haodong Chang, Ziyue Zhang, Wanming Huang, Wei Huang, Chen Deng, Caoyuan Li, Xuan Liang, Jason Traish, etc.) for not alienating me because of the age gap and for comforting, helping, and encouraging me when I encountered various problems.

Finally, I would like to thank my parents (Yunlin Huang and Chongjun Kang), parents-in-law (Jishu Cheng and Fengling Bai), husband (Peng Cheng), and daughters (Shiran Cheng and Shiwei Cheng). During my PhD study, my parents helped me take care of my children and reduce my burden of housework as much as possible. My husband took care of the family's financial burden alone and paid attention to my feelings, helping me relieve my destructive emotions when I was frustrated. My daughters could not understand why I did not spend much time with them at first. Gradually, they accepted my working status and began to be interested in my research. They tried to learn programming, hoping that they could do research like me one day. Thank you to my closest family for your unconditional support and understanding, which enabled me to overcome all difficulties and finally complete my studies successfully.

Yi Huang

Sydney, Australia, 2023.

List of Publications

Journal Papers

J-1. **Yi Huang**, Ying Li, Guillaume Jourjon, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, Richard Yi Da Xu. Calibrated Reconstruction Based Adversarial AutoEncoder Model for Novelty Detection. *Pattern Recognition Letters*, vol. 169, pp. 50-57, 2023.

J-2. **Yi Huang**, Ying Li, Timothy Heyes, Guillaume Jourjon, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, Richard Yi Da Xu. Task Adaptive Siamese Neural Networks for Open-Set Recognition of Encrypted Network Traffic With Bidirectional Dropout. *Pattern Recognition Letters*, vol. 159, pp. 132-139, 2022.

J-3. Ying Li, **Yi Huang**, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Guillaume Jourjon, Darren Webb, David B. Smith, Richard Yi Da Xu. From Traffic Classes to Content: A Hierarchical Approach for Encrypted Traffic Classification. *Computer Networks*, vol. 212, 109017, 2022.

J-4. Thilini Dahanayaka, Yasod Ginige, **Yi Huang**, Guillaume Jourjon, Suranga Seneviratne. Robust Open Set Classification for Encrypted Traffic Fingerprinting. *Computer Networks (2023)*, 109991.

Conference Papers

C-1. Ying Li, **Yi Huang**, Richard Yi Da Xu, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, Guillaume Jourjon. Deep content: Unveiling video streaming content from encrypted wifi traffic. *In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, IEEE, 2018, pp. 1–8.

Contents

List of Figures

List of Tables

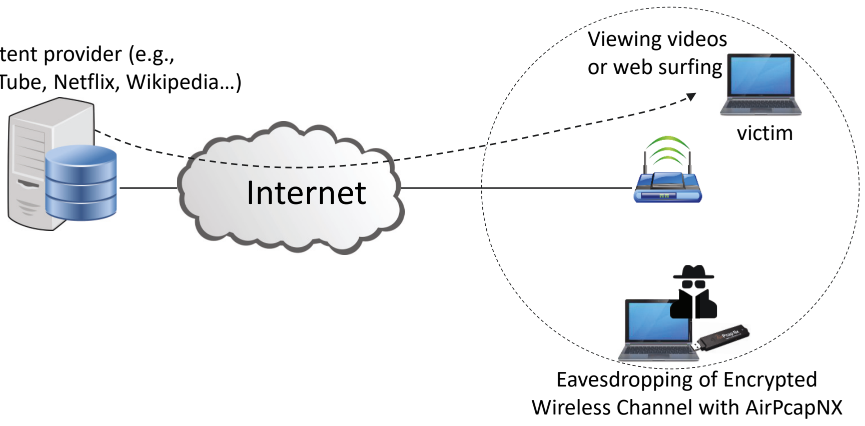
Chapter 1

Introduction

1.1 Background and Motivation

By 2023, approximately 80% of web pages on Firefox and Chrome are loaded over Hyper Text Transfer Protocol Secure (HTTPS) protocol [68]. HTTPS ensures the connections on the network avoid eavesdroppers and hijackers by using encryption technology, Security Socket Layer (SSL) or Transport Layer Security (TLS). Namely, encryption protects one's send-receive information from interception and ensures its integrity [42]. However, the wide use of end-to-end encryption brings some challenges for network management. For instance, in the core network, end-to-end encryption can have a negative effect on traffic analysis for intrusion detection, content filtering and network optimisations implemented by telecommunication technicians. Furthermore, the fine-grained network surveillance like traffic content classification becomes nearly impossible with end-to-end encryption and national security activities [84] are also interfered by encryption.

Traditional network traffic analysis approaches include port number mapping, protocol parsing, and payload-based signature. All these methods depend on details encapsulated in packet layer by layer. When these layers are encrypted, the listed packet information is not available. For instance, if network traffic is captured in air, shown as Figure 1.1. The captured network traffic will be encrypted on Application/Transport Layers (TLS) and Data Link Layers (WPA2), as per the TCP/IP five-layer model. In this condition, all traditional network traffic analysis methods are disabled. Therefore, how to make inferences and predictions from



Content provider (e.g., YouTube, Netflix, Wikipedia...)
Internet
Viewing videos or web surfing
victim
Eavesdropping of Encrypted Wireless Channel with AirPcapNX

Figure 1.1 : Network Traffic Captured in Air.

deeply encrypted network traffic becomes an appealing topic. The motivation of this thesis is based on the following three problems.

1.1.1 Is it feasible to identify the content of deeply encrypted network traffic?

Before further discussion about encrypted network traffic, I clarify that when I say network traffic is encrypted on a specific communication protocol layer, this traffic is encrypted on protocol layers, including current and upper layers. For example, the traffic encrypted on the Transport Layer means it is encrypted on the Application and Transport Layers.

Even though encryption limits the application of traditional network traffic analysis approaches, Azab et al. [10] proposed to characterise network traffic through analysing network traffic behaviours in different communication statuses (e.g., user login, call establishment and answering) and Ceesay et al. [17] deciphered the protocols of tunnelling technologies by investigating how a secure communication channel is established and looking for metadata related to encryption. However, these methods rely on domain knowledge, which is a barrier for cross-domain researchers. In addition, Azab et al. [8] investigated recent research which verified that machine learning and deep learning methods can contribute to encrypted network traffic analysis. For example, [9] explored lenient and strict classifiers using machine learning methods like Support Vector Machine (SVM), Naïve Bayes, Decision Tree (C4.5), and Random Forest (RF) and network statistical features to classify encrypted traffic flows. [109, 4, 23] extracted hidden patterns from network traffic with different levels of encryption. They conducted their traffic analysis by investigating Internet Protocol (IP) packets, Transmission Control Protocol (TCP) flows, and HTTPS flows of the encrypted network traffic on the Transport and Application Layers.

Previous research demonstrated the feasibility of inferences based on encrypted traffic flows (for the TCP Layer) or packets (for the Network Layer). One of the reasons for this conclusion is that the metadata and statistical properties of the traffic still exist, though the message content is encrypted. It is feasible to utilise metadata and statistical properties to extract the underlying traffic pattern implicitly. For example, metadata such as destination and source IP addresses, port numbers, protocols, and other encapsulated header information can be treated as feature filters to extract clean flows from captured raw data. Then statistical properties of traffic flows, like average packet lengths, packet direction of filtered flows, and inter-packet times can be used as features in feature engineering.

This thesis focuses on making fine-grained inferences from passively observed WiFi traffic encrypted on the Data Link Layer. The gaps between the research question of this thesis and the existing research include i) the fine-grained traffic classification task and ii) deficient metadata due to the deepest level of encryption on the Data Link Layer.

About the first gap, as surveyed in [8], most existing research focused on protocol, application, and service group granularity of traffic classification tasks. A few studies [99, 100] researched finer-grained traffic classification, i.e., identifying encrypted video streaming using video titles. They verified that the encrypted video streaming contains an information leak because the Dynamic Adaptive Streaming over HTTP (DASH) based segmentation standard creates content dependent packet bursts. DASH is a streaming technique with an adaptive bitrate that ensures the media content delivery between HTTP web servers and consumers on the Internet can implement high-efficiency streaming. Therefore, different video streams are distinct in their particular burst patterns.

Nevertheless, the deficient metadata due to encryption on the Data Link Layer (second gap) limits feature engineering, which is crucial to machine learning approaches. In addition, Deep Fingerprinting (DF) [116] and Automatic Website Fingerprinting (AWF) [102] showed that deep learning methods outperform classical machine learning methods like SVM, k-Nearest Neighbors, and RF. For this reason, this thesis focuses on developing deep learning models that can implement various classification tasks in the deeply encrypted scenario without complex handcrafted features.

In 2017, [126, 125] and [109] concurrently started to leverage deep learning models like Convolutional Neural Networks (CNNs) to address the encrypted network traffic classification tasks. Specifically, Schuster et al. [109] proposed a CNN model for fine-grained traffic classification (identifying encrypted video streams) with very coarse network measurements and obtained satisfactory performance. In this thesis, I aim to explore a weaker but more realistic hacking scenario, where the hacker only needs to be within the coverage of the target WiFi network rather than needing to access the IP level traffic (i.e., a malicious ISP or government entity). Compared to the encrypted traffic on the Transport Layer in previous research, the encrypted traffic on the Data Link Layer lacks metadata such as IP addresses. In this scenario, I captured 802.11 frames on a certain channel, and the MAC address is the only piece of metadata that can be used to filter frames and trace the flow between two nodes. Ensuring that the filtered frames belong to a specific flow is challenging, particularly because the two nodes might communicate on different channels due to WiFi's Frequency Hopping Spread Spectrum (FHSS) mechanism. Conversely, the Network Layer traffic can be filtered by protocols and the IP addresses, as mentioned in [109], making the raw data for feature engineering more accurate and cleaner. Therefore, making inferences from traffic encrypted on the Data Link Layer presents more challenges.

In summary, although this research question has been explored in specific scenarios by previous studies, validating the feasibility of finer-grained classification tasks in a more rigorously encrypted environment remains valuable.

1.1.2 How to handle classification for previously unseen classes in the open world?

As stated in the Section 1.1.1, deep learning is a promising way for the fine-grained classification task and deeply encrypted scenario in this thesis. However, despite deep learning models have achieved unprecedented success in classification tasks, these models have limitations when handling previously unseen classes. For example, while a well-trained closed-set deep learning model can classify inliers (i.e., known classes) with high accuracy, it is bound to make errors if the input is from an unknown class. In this case, the classifier will still classify this data point as one of the known classes.

Handling unseen classes can be looked at from three different approaches: anomaly detection, outlier detection, and novelty detection. These three approaches are tightly related, and most of the time, they can be used interchangeably. Specifically, if outliers (unseen samples) are treated as abnormalities, then using anomaly detection is more suitable. Meanwhile, if outliers are treated as a new class, novelty detection is more appropriate. Outlier detection typically implies that the training data contains outliers, and outliers cannot form a dense cluster. In contrast, novelty detection usually does not contain outliers in the training data and outliers (i.e., novelties) can form a dense cluster. The techniques leveraged by anomaly detection, novelty detection and outlier detection are similar. To avoid ambiguity, throughout this thesis, I will use the terminology 'novelty detection'.

Novelty detection research involves correctly identifying outliers that are unseen during training. Outliers can be located at any location, for example, abnormalities are usually far from inliers, but novel/unobserved patterns are near inliers. Various approaches addressing novelty detection include both Frequentist and Bayesian approaches, extreme value statistics, information theory, machine learning (SVM), and deep learning (neural networks) ([95]). The objective of these methods is to construct a decision boundary to distinguish inliers from outliers. Recently, neural network based supervised learning methods have achieved favorable performance ([18]). The research in [55] compared a traditional method with a deep learning model in a network intrusion detection scenario and demonstrated that deep learning methods become increasingly effective compared to traditional approaches as the sample size grows. In addition, the study in [26] compared traditional machine learning methods (clustering, local outlier factor (LOF), principal component analysis (PCA), and SVM) with deep learning models (MLP, LSTM, and CNNs) and the results also endorsed the growing trend of utilizing deep learning techniques in the context of novelty detection.

In general, deep learning methods for novelty detection can be categorized into two main classes. *One-class Novelty Detection* (OCND) methods utilise strategies like reconstruction ([91]) and density estimation ([101]) to construct a decision boundary. In addition, *Multi-Class Open Set Recognition* (MCOSR) approaches that aim to distinguish outliers from inliers and classify inliers usually construct the decision boundary through some strategies to improve their boundary accuracy.

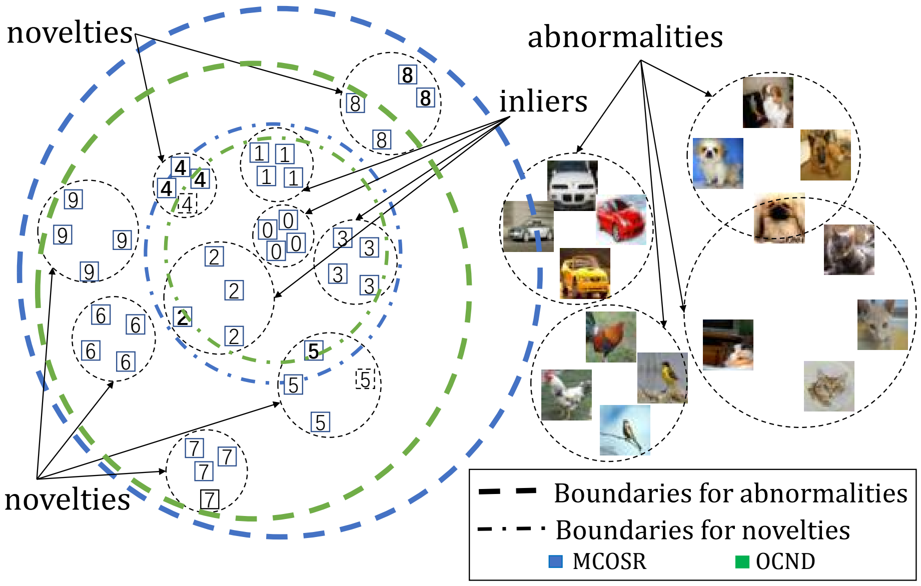


Figure 1.2 : A schematic view of novelty detection decision boundaries with MCOSR and OCND approaches for different location outliers, such as abnormalities and novelties.

For example, including artificial outliers for training ([67]) and temperature scaling ([45]) to implicitly and explicitly calibrate the softmax confidence. Figure 1.2 shows a schematic view of novelty detection decision boundaries with different approaches including MCOSR and OCND.

MCOSR approaches are based on the softmax score of an elaborate classifier. Therefore, their decision boundaries are prone to envelop each inlier class boundary because classifying inliers is one of their targets. On the other hand, the decision boundaries of OCND approaches are tighter in contrast to MCOSR approaches. Tight boundaries outperform loose boundaries when outliers are located near inliers (see thin dash-dot lines in Figure 1.2). However, it becomes a drawback when outliers are far from inliers because it is likely to misclassify more inliers as outliers (see bold dash lines in Figure 1.2).

While MCOSR and OCND approaches have achieved state-of-the-art performance in their corresponding outlier location scenarios, they usually underperform in detecting an accurate decision boundary for different outlier location scenarios. The Generative Probabilistic Novelty Detection (GPND) model proposed by [94] attempted to handle this problem. However, the performance of GPND in the close distance outlier detection scenario is still limited, especially when there are multiple inlier classes.

1.1.3 How to tackle dynamically changing classification tasks with few samples in the open world?

While traditional network traffic analysis methods, including port number mapping, protocol parsing, and payload-based signatures, struggle to handle encrypted network traffic, deep learning approaches are proven successful in this field. In [109], a CNN was utilised to identify which video was streamed over HTTPS from multiple video providers based on side-channel features only. Similar results have been shown using deep learning methods for website fingerprinting over encrypted Domain Name System (DNS) traffic [25], over the anonymous network Tor [116, 102] and VPN [22, 60].

Nonetheless, previous work on encrypted traffic classification is unsuitable for real-world applications due to two limitations. First, the models do not generalize and cannot handle dynamically changing tasks. Deep learning methods are known for their high training data requirements [122]. Traffic fingerprinting and its applications such as parental control, surveillance, and censorship are highly dynamic. The target list of content or websites to monitor can change frequently, and in some cases, there may not be enough training data to train deep models.

The second limitation of existing work is that they mainly consider closed-set classification only. In contrast, a real-world traffic fingerprinting system requires the ability to separate background traffic (unknown classes) from target traffic (known classes), i.e., there is little focus on open-set recognition (OSR).

Few-shot learning is a promising way to address the dynamically changing task challenge. Many models [62, 6, 72] with few-shot learning have been applied to computer vision and natural language processing. Recent research demonstrated that few-shot learning also can achieve high performance in intrusion detection [131, 33, 129], website fingerprinting attack [19] and communication jamming recognition [74]. However, its application in fine-grained encrypted network traffic identification has not been extensively explored. The more challenging scenario is that only one training sample per content or website is available. In addition, classic few-shot learning models usually handle closed world tasks and do not support unseen classes in testing.

1.2 Contributions

In Section 1.1, I stated the necessity and feasibility of the encrypted network traffic analysis. I introduced three problems to be solved and explained the current research approaches and their limitation. This thesis will address the problems as mentioned above in encrypted network traffic classification, with specific corresponding contributions stated as below:

To solve the first research question "Is it feasible to identify the content of deeply encrypted network traffic?", I configured a data collection environment to capture network traffic in air and demonstrated the possibility of identifying video streaming content from encrypted WiFi traffic traces utilizing a deep learning model architecture. In addition, I constructed a bunch of features and evaluated their importance by utilizing them one by one to see the experiment performance. More and more people enjoy various videos on the Internet through mobile phones or computers, conveniently benefiting from the high bandwidth communication technology (e.g., the fourth generation of mobile communication technology (4G) and WiFi). Meanwhile, the popularity of video consumption breeds misuse and even crime. For example, malicious parties utilise video to spread fake news, propaganda, and crime related content. Therefore, it is necessary to identify whether certain target videos are streaming in a certain area or being viewed by particular individuals. Due to this, I focus on identifying video streams for closed world scenario.

The contributions of this part include:

- First, I collected a new dataset of network traffic captured through a WiFi sniffer.
- Second, I evaluated the feasibility using a model based on a lightweight deep learning architecture CNNs to classifying newly collected encrypted video streams.
- Third, I evaluated the performance of each constructed feature based on my feature engineering work and noted the best performing ones.

The second research question is "How to handle classification for previously unseen classes in the open world?". I went into the novelty detection field to pursue solutions for classification in the open set scenario. I built a unified Calibrated Reconstruction Based Adversarial Auto-Encoder (CRAAE) model for location agnostic outlier detection. My CRAAE model embraces reconstruction strategies of OCND and the confidence calibration mechanism of MCOSR. The proposed model divides the extracted latent representation vector (feature space) from the encoder into category and style components, which are assumed to carry class label information (e.g. number '0' or '1' in the MNIST dataset) and background style information (e.g. upright or tilted writing style in the MNIST dataset) of the input data as explained in [77, 101]. I leverage the category information from feature space to explicitly calibrate the reconstruction error constructed in the original data space for creating a more accurate decision boundary. I introduce decoder generated data as *known outliers* during the training process for implicit calibration and to calculate; (i) the cross entropy loss for inliers between the softmax output of the category component and the ground truth, and (ii) the Kullback-Leibler (KL) loss for known outliers between the softmax output of category component and the Uniform distribution.

My main contributions to this problem can be summarized as follows:

- First, I propose a novel unified model framework CRAAE for location agnostic outlier detection. CRAAE integrates the explicit and implicit confidence calibration strategies into a reconstruction-based model. I leverage the category information from the feature space to explicitly calibrate the reconstruction error constructed in the original data space for building a more accurate decision boundary. I add the decoder generated data into the training process for implicit calibration.

- Second, I propose using Uniform and Dirichlet distributions to replace the Gaussian distribution when generating known outliers to represent more generalized and precise outliers. This approach improves performance in the far distance outlier detection scenario.

- Third, I validate my model on both the network traffic datasets and image datasets, which achieves state-of-the-art performance for any outlier location scenarios as a unified model when compared to GPND, as well as performs competitively in comparison to OCND and MCOSR methods in the close and far distance outlier detection, respectively.

To solve the third research question "How to tackle dynamically changing classification tasks with few samples in the open world?", I extend the large dataset based classification to few-shot learning based models. In this thesis, I propose a **bidirectional dropout data augmentation** method that works in conjunction with Siamese Neural Networks (SNN)-based few-shot learning ([62]). More specifically, I treat dropout as a resampling tool to generate new samples that are similar to given samples (i.e., positive pairs) and new samples that are different (i.e., negative pairs).

To improve open-set recognition performance, I leverage two key ideas. The first is a **task adaptive metric**. Usually, to classify a query sample under the SNN architecture in a closed-set, I need to pair the query sample with each sample in the support set and classify the query sample to the class of the pair with the maximum similarity score. To extend this to the open-set, I can use a threshold on the maximum similarity score. However, similarity scores of pairs for SNN vary according to the class diversity of support sets. As such, I need a method to identify the value range of the similarity scores for closed-set and open-set settings for each support set.

Though there is only one sample for each support class in one-shot learning, I can construct negative pairs that can be used as supplementary information to estimate the confidence of similarity scores for open-set recognition. I feed these negative pairs into the trained SNN to obtain task dependent negative similarity scores. Similarity scores will form multiple clusters according to the different levels of similarity. I assume these negative similarity scores follow a Gaussian Mixture Model distribution. However, the number of clusters with negative similarity scores is unknown. Therefore, I utilise the **Dirichlet Process Gaussian Mixture Model (DPGMM)**, which can automatically generate new clusters according to the data to fit the distribution of these negative pairs. Then, I leverage the probability of each query sample following the trained DPGMM distribution to build the new OSR metric.

The second key idea to improve the open-set performance is to use **hierarchical cross entropy loss**. SNN consists of CNN blocks that can extract features of the input, and different depths of the CNN can extract features from a coarse to a fine level. When coarse and fine level labels (e.g., alphabets and characters in the Omniglot dataset or video platforms and exact videos in the encrypted network traffic dataset) are available, I can use their corresponding features to construct hierarchical cross entropy loss at different levels to improve the network performance.

Finally, the training pairs of SNN are irrelevant to the number of support classes of a query task. A query task consists of a N-way one-shot support set and one or more query samples. I can calculate the accuracy and AUROC of any N-way one-shot validation/test query task for the same trained model. The best performing model varies with 'N', the number of support classes. Thus, to improve the robustness of model selection, I propose a multi-model ensemble strategy, i.e., I select several best performing models based on multiple support classes scenarios, and these selected models work together to identify the query samples.

In summary, I make the following contributions to this problem:

- First, I propose a bidirectional dropout data augmentation method to enrich training samples and create highly distinguishable similarity scores in the training process.
- Second, I utilise the DPGMM distribution to fit the similarity scores of the trained model fed by task dependent negative pairs and create a new open-set recognition metric, the probability of each sample under the trained DPGMM distribution.
- Third, I construct the hierarchical cross entropy loss to improve the confidence of similarity scores and apply the multi-model ensemble method to the test process to ensure the robustness of model selection.

- Finally, the experiments demonstrate that the proposed approaches obtain a significant performance gain of 4.5% and 4.0% in terms of accuracy and AUROC on the encrypted network traffic dataset as well as 1.2% and 1.8% on the Omniglot dataset.

1.3 Thesis Organisation

This thesis is organized into five primary sections to provide a comprehensive overview of my research. These sections include the Abstract, Acknowledgments, Lists of Publications, Figures and Tables, the Main Body, and the Bibliography.

Within the Main Body, the content is further divided into seven distinct but interrelated chapters, as outlined below:

- **Chapter 1: Introduction** - This opening chapter provides the background and clarifies the motivation for the study, as well as outlining the contributions made by this thesis.
- **Chapter 2: Literature Survey** - This chapter offers a comprehensive survey of existing research in areas pertinent to this study, such as network traffic analysis, feature engineering, novelty detection, and few-shot learning.
- **Chapter 3: Classification for Encrypted Network Traffic in the Closed World** - Addressing the first research question, this chapter presents the methodologies employed for data collection and processing of encrypted network traffic. It validates the feasibility of using CNNs to classify deeply encrypted network traffic in a closed-world scenario.
- **Chapter 4: Classification Models for Encrypted Network Traffic in the Open World** - In response to the second research question, this chapter presents my proposed models and approaches for novelty detection, i.e.,

CRAAE model framework and Dirichlet Mixture Model-based softmax calibration method.

- **Chapter 5: Few-Shot Learning for Encrypted Network Traffic in the Open World** - Tackling the third research question, this chapter unveils my innovative task-adaptive Siamese open-set recognition model for encrypted network traffic. It presents each strategy of this model, including bidirectional dropout data augmentation, DPGMM based task adaptive OSR metric, hierarchical SNN, and multi-model ensemble.
- **Chapter 6: Conclusion and Future Work** - This chapter summarises the key findings and contributions of my thesis while also discussing the potential avenues for future research.
- **Chapter 7: Appendix** - This chapter serves as a repository for additional experimental details and outcomes, augmenting the core chapters and enabling a deeper understanding of the research.

Chapter 2

Literature Survey

2.1 Introduction

This thesis started with the application requirement unveiling the underlying pattern of encrypted network traffic captured through WiFi sniffing. Furthermore, to meet real world scenarios, I aimed to classify encrypted network traffic and identify the traffic from unseen classes with either a large number or a small number of training samples. To achieve the target, I investigated previous research achievements about network traffic analysis, feature engineering, novelty detection and few-shot learning. I knew about the traditional techniques and cutting edge technology that had been applied to solve the similar problems. Though there are different limitations in existing approaches, they laid the foundation and inspired my idea. I summarize my literature review into four parts in this chapter.

2.2 Network Traffic Analysis

Network Traffic analysis can be divided into packet-based and flow-based categories. Packet-based traffic analysis collects packet information, including IP addresses (source and destination), port numbers (source and destination), packet sizes, and specific payload data. Flow-based traffic is aggregated by filter rules. A flow is a series of packets filtered by addresses, ports, and protocols. The collected flow information is the bit rate, size and duration of the flow and the number of flows per time bin. There are several packet capture tools, namely TCP-dump, Wireshark, Snort, and common flow capture tools (e.g., NetFlow and JFlow) [16].

In my research, my analysis object is frame-based network traffic captured by Wireshark, and my analysis task is network traffic classification in both the close-set and open-set situations. Frame-based network traffic is similar to packet-based network traffic, but it only collects information that includes source and destination Media Access Control (MAC) addresses, frame sizes, frame direction, frame types, and radiotap headers. Next, in this section, I will introduce various network traffic analysis methods, including traditional methods, classic machine learning and deep learning methods.

2.2.1 Traditional Packet Analysis Methods

Traditional analysis methods for traffic packets include port number mapping, protocol parsing, and payload-based signature.

Port number mapping associates port number [53] with the corresponding traffic type. For example, TCP ports 80, 53, and 25 correspond to web traffic, Domain Name System (DNS), and e-mail. This method is simple and fast because it only depends on packet headers. However, dynamically allocated port numbers will be the future trend because more and more applications prefer to conceal their traffic for privacy reasons or other targets. This situation results in the limitation of port number based traffic classification methods.

Complete Protocol parsing analyzes packets from various protocol headers. It is an accurate but not realistic solution because of its defects such as high computational complexity for implementing each current protocol parser, lack of authority for proprietary protocols, and secured protocols [16]. As a result, it is usually used in combination with other methods, e.g., signature.

Signature is a kind of payload-based method which aims at packet payloads rather than packet headers. This method utilises predefined byte sequences to classify traffic types. e.g., the string '\xe3 \x38' in eDonkey P2P traffic and '\GET' in web traffic [16].

Signature makes traffic classification less computationally complex but exists several problems needed to be solved. First, the selection of signature needs adequate experience; otherwise, the classification will not be accurate. Second, for offline signature analysis, the signature may be in the truncated and threw part by chance, and it will result in a low hit ratio. Finally, this method cannot deal with encrypted content, which is a critical defect in the future.

In [111], the five common P2P traffic types can be accurately classified based on elaborately selected application level signatures from available packet-level traces according to previous knowledge. However, It consumes plenty of time to extract signatures manually. Chhabra et al. [21] proposed the Packet Imprint in Security Attacks algorithm (PISA), which clusters similar flows based on packet information and extracts signatures from intensive clusters which consume a large proportion of the bandwidth. PISA provided an aggregation view of the traffic and defined the significance of a signature in terms of dimensionality, intensity, persistence, and distribution. As stated above, most network connection related application classification methods or tools are based on various signatures. However, encryption is an apparent bug that can be used to bypass these mechanisms. When encryption mechanisms such as Secure Sockets Layer (SSL), Secure Shell (SSH), and Internet Protocol Security (IPsec) are readily available and widely used to encrypt any type of traffic, the problem becomes increasingly prominent. In paper [12], they proposed a method to solve a part of the encryption problem. This method only leverages the size of the first few packets to identify the underlying application in SSL encrypted connections with about 85% accuracy.

Much related work contributes to traditional packet analysis. Even though most of these methods are no longer practical with wide encryption in internet network communication, their methodology and thinking are still helpful to inspire future researchers.

In the following two sections, I will introduce classic machine learning and deep learning methods which can handle more complex and rigorous situations.

2.2.2 Classic Machine Learning Methods

As I stated before, the limitation of previous techniques is usually caused by their dependence on common port numbers or interpreting payload contents. As a result, new approaches turn to analyze traffic's statistical characteristics. The Network Layer traffic has features like bytes, duration, and arrival periodicity, which are unique for distinguishing traffic. Based on previous work, machine learning techniques have emerged as a promising tool to handle traffic patterns, flow attributes and packet characteristics in multi-dimensional spaces with large scale datasets [88].

Mitchell [80] defined that "Machine Learning is the study of computer algorithms that improve automatically through experience." I notice from the definition that algorithms and experience corresponding to mathematical models and training datasets are the core elements of machine learning. Machine learning learned from data and automatically improved their algorithm parameters to achieve high performance of their tasks. What machine learning learns can not be explicitly programmed in advance. In the real world, there are many challenging tasks for a human to create the programmable or interpretable algorithms manually and specify every necessary step[3]. For these tasks, machine learning techniques are feasible approaches to solve the problem if there are enough training data.

Machine learning approaches are usually divided into three broad categories: supervised, unsupervised, and reinforcement learning. For supervised learning, each data sample has a label that plays the "teacher" role in telling the machine to adjust its parameters to achieve higher performance. In contrast, unsupervised learning is not given labels to estimate how the algorithm works during training. The task of unsupervised learning is usually to discover hidden patterns of given data. For reinforcement learning, it pursues a trade-off between exploring unknown areas and exploiting existing knowledge. In other words, it enables models (agents) to take actions in the environment that can maximize the cumulative reward. Reinforcement learning is widely used in game theory, information theory, simulation-based optimization, and multi-agent systems. Recently, some work [118, 37] applied reinforcement learning to address network traffic analysis, especially for network defense scenarios. In this thesis, the data is labelled while it is being collected. For example, the video title is the label for the network traffic sample streaming this video. Therefore, here I focus on investigating supervised learning.

In the supervised learning scope, there are regression and classification approaches that are different in the inference process. Classification is to map data instances to a series of discrete values, which are named as classes. However, regression is to map data instances to a series of continuous values. There is a simple example to understand the difference between classification and regression. Predicting whether an account has the credit risk belongs to classification, and predicting the probability of the credit risk for an account belongs to regression. Next, I will summarise popular supervised learning techniques in the network traffic classification field because classification techniques are the core of this thesis.

In 2004, Roughan et al. [103] proposed Class-of-Service (CoS) Mapping for Quality of Service (QoS), which utilised machine learning algorithms, NN, LDA, and QDA, to match various network applications with corresponding QoS traffic classes. Then Moore et al. [82] applied Naïve Bayes estimator to implement application based network traffic classification. This work fed the manually labelled traffic samples into the Naïve Bayes estimator for supervised learning. It was proved that this approach could achieve a high level of accuracy. In [87], a classification sliding window method was proposed to address timely and continuous classification, which is a significant constraint for a traffic classifier in the real world. The classification sliding window method only used little packets to keep the timeliness of classification and reduce the packet's storage space. This method is different from other approaches, which need to capture the beginning of a traffic flow. Therefore, it provided the feasibility of monitoring traffic. In 2013, [36] proposed a discriminative restricted Boltzmann machine (DRBM) based approach for anomaly detection on the KDD cup dataset [71]. DRBM belongs to the class of statistical machine learning and is capable to integrate generative power to capture the underlying characteristics of the normal traffic class. This paper used 28 of 41 features of KDD cup dataset for classification.

2.2.3 Deep Learning Methods

Deep learning is indeed a subarea of broader machine learning. Deep learning models usually consist of at least three layers of artificial neural networks. Deep learning models benefit from current fast computers and big data, achieving very high performance with vast parameters. In the network traffic domain, deep learning models, for example, CNNs and Recurrent Neural Networks (RNNs) have been widely applied for intrusion detection.

In 2015, [89] applied a Self-taught Learning model combined with softmax regression [97] to detect abnormal connections in the NSL-KDD cup dataset [121]. In 2016, [76, 61, 120, 59] proposed various deep learning approaches for intrusion detection. Ma et al. [76] proposed a spectral clustering based ensemble model that consists of multiple parallel sub-models. Each sub-model is independent and corresponds to a cluster. Then Kim et al. [61] proposed a Long Short-Term Memory (LSTM) recurrent neural networks model to identify the network traffic. This paper only utilised 300 entries of each class for intrusion detection. This model can achieve high true positive rates. However, false positive rates of this model are also very high (up to 80%). In [120], the authors focused on feature engineering. Their one-class model only leveraged 6 features out of the 41 features. Previous work focused on Internet network connections that use the TCP/IP protocol. Kang et al.[59] extend the application scenario to a vehicular network that uses the Controller Area Network (CAN) protocol [34]. This paper proposed a Multilayer Perceptron (MLP) model as a one-class classifier and only leveraged the "DATA" payload of each packet consisting of a 64 bits vector.

In 2017, [78] surveyed the performance comparison of various deep learning and traditional Bayesian models. They noticed that these approaches could achieve similar performances. In the same year, [109] firstly proposed a CNNs model to identify the content of encrypted traffic rather than only traffic types. The model could identify videos streaming over HTTPS with very high accuracy through features extracted from temporal network traffic. Then [75] also employed a CNN model to classify the traffic flows according to the patterns of the packet sequence in each flow.

2.3 Feature Engineering

Features are shared attributes or characteristics of the data samples which are utilised to analyze or predict. Feature engineering may affect the performance of machine learning algorithms because features represent various information hidden in data samples. There are feature extraction and feature selection two main parts of feature engineering. Feature extraction is to extract or construct features from raw data through data process techniques based on domain knowledge. Feature selection is to check how the extracted features work and choose the most miniature necessary set of features for the target model.

For network traffic, Moore et al. [81] provided 249 features, including simple statistics of packet length, packet number, and inter-packet timings, and features derived from the protocol header (e.g., SYN and ACK counts) for TCP flows on up, down and combined up and down direction. In 2017, Moore et al. [83] extended their feature extraction work to feature selection. This paper utilised a wrapper method to select features that combined classification and feature selection. Here, network traffic data was from the 2003–2007 and 2009 Department of Defense Cyber Defense Exercises (CDXs). This paper combined the artificial neural network (ANN) model with a feature signal-to-noise ratio (SNR) to classify threat traffic flows and select features. The result shows that there are 18 salient features of 248 features from the CDXs data.

For malware written in Portable Executable (PE) format, Raman et al. [98] identified seven key features to assist machine learning algorithms for novelty detection. They selected 100 features based on their domain experience at first. Then, they utilised the Random Forest algorithm on these experience based 100 features one by one to filter 13 features with the highest individual accuracy. In addition, they divided these features into seven buckets according to where the features originated in the PE file and assumed that the most essential, less-correlated features would be the features with the highest individual accuracy from each bucket. Finally, they found a minimum feature set by selecting features one by one from these buckets according to the descending order of individual accuracy. This order was determined by utilizing machine learning algorithms, e.g., J48, J48 Graft, and Random Forest, to see the performance of each feature.

For image data, Jiang [58] introduced four category feature extraction techniques based on human expert knowledge, image local structure, image global structure, and machine learning. In recent years, the CNN has been replacing traditional feature extractors since it can extract complex features hidden in images and filter more efficient task-based features. Zhao et al. [140] proposed a framework combining dimension reduction for spectral extraction and CNN for spatial feature extraction. Then, the integrated features, through stacking spectral and spatial features together, were applied for image classification and achieved good performance.

In summary, even though feature extraction is very domain specific, there are some general methods for feature extraction [46], such as the dimensionality reduction through space embedding or projection (e.g., PCA), and the extraction of local features using convolutional methods. When original features are not enough compared with the complex problem, dimensionality increase is also a reasonable method consisting of statistics of the original features.

'Not too many, not too few' is crucial but challenging for feature engineering. Too few features cannot adequately represent the characteristics of the data samples. To better represent the data samples, it is necessary to extract enough features. However, a high number of features may result in the 'Curse of Dimensionality' problem. As the number of features increases, the feature space expands exponentially, making the data points within it sparser. Then, models based on data in a sparse feature space may lack generalization ability because they tend to overfit to the unique data points. For this reason, feature extraction needs to consider feature dimensionality. In addition, feature selection can help to eliminate unnecessary features, thereby further reducing dimensionality.

For feature selection, Liu et al. [73] divided the approaches into three main categories: filter, wrapper, and embedded methods. Filter methods select the best feature subset according to specific metrics before the model training based on the general characteristics of data [30]. For example, Pearson's correlation can be used to measure the linear dependence between two features, after which redundant features can be identified. Filter methods are not dependent on the classification algorithm, which means once the feature subset is selected, it can be applied to different classifiers. However, filter methods ignore the feature effect on a classifier, which may result in a decline in classification performance. Wrapper methods iteratively train a model using a subset of features and adjust the number of features in the subset based on the model's performance in the previous iteration. For example, 'Forward Selection' initiates with zero feature in a model and continuously adds the feature that most improves the model to the feature subset, until adding an extra feature no longer brings any performance improvement. Wrapper methods are computationally intensive because feature subset search and model selection are conducted interactively. Embedded methods implement feature selection simultaneously with model training, requiring less computation than the wrapper method. For example, random forest methods use the random extraction of the features and samples to train hundreds of trees. Each tree only sees a part of features and samples, ensuring the trees are less apt to over-fitting.

2.4 Novelty Detection

There is a plethora of work in novel detection [95] and anomaly detection [18]. Such work can be categorized from various perspectives such as MCOSR [107, 108, 67] and OCND [92, 94, 13] or Model-closed assumption [13, 101] and Model-open assumption [85]. This section discusses deep learning based related work in MCOSR and OCND categories. MCOSR approaches for novelty detection usually coordinate with the softmax score of an elaborate classifier through strategies such as explicit and implicit confidence calibration to distinguish outliers from inliers. On the other hand, OCND approaches utilise strategies in the likes of reconstruction and density estimation.

2.4.1 MCOSR Approaches

Scheirer et al. [107] presented the problem of open set recognition and asserted in [108] that real-world data is comprised of three basic categories, that are, known-known, known-unknown, and unknown-unknown data. I introduce MCOSR related work from explicit and implicit confidence calibration aspects.

Explicit confidence calibration mainly includes probability-based and temperature scaling strategies. Bendale et al. [11] proposed a probability based method OpenMax which influenced many other subsequent research works in this field. The authors adapted deep neural networks for open set recognition and utilised the penultimate layer of the classifier model to fit a Weibull distribution, then estimated the probability of an input being from an unknown class. Webb [130] introduced softmax output equalization, which improved the performance of OpenMax, and Ge et al. [38] extended OpenMax by employing a GAN for novel category image synthesis. Furthermore, Yoshihashi et al. [134] utilised multiple layers of the classification model to replace the penultimate layer of OpenMax in order to keep hidden information, that otherwise vanishes in the middle of the inference chains. Temperature scaling [45] is a scalar calibration of the logit vector. Liang et al. [70] combined temperature scaling confidence calibration with slightly perturbed input to separate the softmax score distributions between inlier and outlier and proposed Out-of-Distribution Image Detection (ODIN) that does not require any change to the pre-trained neural network.

Besides, Devries et al. [28] proposed a method of learning confidence estimates by a new neural network layer. Here, the function of confidence estimates and temperature scaling is similar, but confidence estimates are learned during the training process in contrast to temperature scaling, which is learned during the test process. Hassen et al. [49] treated the softmax score as the projection of the input instance in another space and combined a distance loss function with cross entropy. The model was trained to maximize the distance between different classes (inter class separation) and minimize cross entropy and distance of an instance from its class mean (intra class spread). Finally, the outlier score is calculated as the closest Euclidean distance of an instance from each class mean. [44] proposed a framework for joint modelling of labels and data by treating the discriminative classifier as an energy based model for the joint distribution.

The above work provides explicit confidence calibration for the softmax scores. Meanwhile, some MCOSR methods propose implicit confidence calibration by introducing known unknown data (i.e., known outlier) into the training process to calibrate the softmax score. For instance, [29, 51, 14] added some existing datasets as negative samples into the training process. Furthermore, Lee et al. [67] developed a training method that introducing data generated by a GAN [41] as negative samples and jointly trained the classifier and the GAN model with an integrated loss that includes cross entropy loss, original GAN loss, and KL loss between softmax output and the Uniform distribution.

2.4.2 OCND Approaches

OCND approaches can be broadly categorized into reconstruction-based methods and density estimation-based methods.

Reconstruction-based methods [5, 92, 138, 104, 91, 94, 133], demonstrated that the difference between inliers outliers can be detected through reconstruction metrics such as reconstruction error and reconstruction probability.

Sabokrou et al. [104] leveraged reconstruction error directly as the decision score, while Zenati et al. [138] used adversarially learned features through a bi-directional GAN to adjust reconstruction errors and to determine if a data sample is anomalous. On other hand, multiple work [5, 94, 91, 92, 133] utilised reconstruction probability, which is a probabilistic measure. For instance, Pidhorsky et al. [94] discretized the reconstruction error of the training set using a histogram method and then applied the histogram parameters to discretize the reconstruction error of the test set. More recently, In [91], the latent representation vectors were conditioned by Feature-wise Linear Modulation (FiLM) [93] to make the decoder perfectly and poorly reconstruct the inliers and the outliers, respectively.

Under the density estimation-based methods, Bishop et al. [13] proposed Uniform distribution to represent outliers which were effective in low dimension data. Recently, based on Bishop's idea, Ren et al. [101] assumed that each object to detect has a background component and semantic component. The authors trained two models with the same architecture but different parameters to identify inliers and outliers. In contrast, the typical set algorithm proposed by Nalisnic et al. [85] focuses on sculping the distribution of inliers accurately by leveraging a deep generative model and typical set theory which can address the curse of dimensionality.

There is also some work on novelty detection based on unsupervised and semi-supervised learning. Basically, these methods share some similarities with supervised learning methods but usually combine clustering techniques to construct decision metrics. For example, [1, 115] leveraged reconstruction error and latent vectors similarity based on GAN and Variational Autoencoder as decision metrics. [119, 110] included more samples into the training process based on a contrastive learning scheme to discriminate in- and out-of-distribution samples.

To summarize, some approaches (e.g., [67, 70]) are suitable for the far distance outlier detection scenario. On the other hand, some approaches (e.g., [92, 104]) perform better in the close distance outlier detection scenario. Few methods ([94, 110]) attempt to address both far and close distance outlier detection scenarios. However, they have limitations in the close distance outlier detection scenario, especially when there are multiple inlier classes or depend on complicated deep learning models with massive hyperparameter tuning. This thesis will propose a lightweight unified model to address the location agnostic outlier detection problem.

2.5 Few-Shot Learning

To describe few-shot learning in a formal way, I leverage the definition of FSL in [127]. They firstly re-defined machine learning as "a computer program is said to learn from experience E with respect to some classes of task T and performance measure P if its performance can improve with E on T measured by P." FSL was then defined as "a type of machine learning problems (specified by E, T and P), where E contains only a limited number of examples with supervised information for the target T."

From the definition, FSL looks opposite to classic machine learning [54] which tackles problems with a fixed task on heavy training. As we know, numerous supervised training samples bring significant performance improvement in classic machine learning, and enormous training samples are one of the key factors for classic machine learning achievement. Therefore, limited supervised training with few samples and dynamic tasks is a big challenge. Few-shot learning is a promising way to address the problem. There are different genres of approaches based on such as distance metrics, model architectures, optimization algorithms, and data augmentation. Next, I first introduce the classic few-shot learning approaches and then explore recent work and few-shot learning applications. Finally, I investigate the few-shot learning approaches in open-set recognition.

2.5.1 Classic Approaches

For distance metric based approaches, in 2015, [62] proposed SNN, which aims to distinguish classes from the similarity of input pairs. The core network of an SNN framework consists of a series of convolutional layers which are used to extract the features of inputs. Each sample of input pairs is fed to the same neural networks and projected to the same feature space. Then the L1 distance of each pair is calculated in this feature space, and the normalized distance is sent to the binary cross entropy module to train the network. Once the network has been tuned, it can adapt new classes from unknown distributions and classify them based on the similarity generated by the trained powerful feature extractor. In 2016, Vinyals et al. [124] proposed an attention mechanism based nearest-neighbor classification approaches, Matching Network, to implement multi-classification through cosine distance on feature space. This approach utilised the meta-learning training and test setting, training a feature extraction network by a set of labeled examples (i.e., the support set) and predicting classes for the novel samples (the query set). This paper defined an episode term as the combination of a support set and a query set which is widely used in subsequent research. Each episode represents a few-shot task. Training and testing on the same type of episodes make the classification power transfer between training and testing more reliable. In 2017, [117] proposed Prototypical Networks (PNs), which are inspired by k-Means clustering. PNs learn a metric space and build prototype representations for each class. Then PNs can classify the test sample by searching the minimum distance to prototype representations.PNs handled both few-shot learning and zero-shot learning. At training time, a base network is trained to learn the cluster centers, using a subset of classes and a subset of samples (e.g., 5-way, the support set, and 1-shot the query set). At test time, a similar setup is used for unseen classes.

For the model architecture based approach, Santoro et al [106] proposed a Memory-Augmented Neural Network (MANN), which is the foundation of model-based approaches. MANN first introduced external memory into few-shot learning to work like the LSTM mechanism. MANN utilised a controller to implement long-term weights storage through slow updating and short-term weights storage utilizing the external memory. This method gives the model the ability to rapidly adapt to new data and use this data to make accurate predictions given only a tiny sample of data.

For the optimization algorithm based approach, Model Agnostic Meta Learning (MAML) was proposed by Finn e al. [35]. MAML learned how to initialize with the best possible model parameters to achieve fast learning on a new task with fewer gradient steps. This algorithm is model-agnostic; that is, it can work with any model trained with gradient descent no matter model application target is classification, regression, or reinforcement learning. However, it has some issues, for example, complex hyper-parameter searches for stable training and high computational costs during both training and inference.

The lack of training samples may result in over-fitting, a difficulty that few-shot learning needs to overcome. Collecting more samples for dynamically changing classification tasks are usually challenging. Therefore, it is better to generate more samples based on existing data using data augmentation techniques. In the data augmentation sub area, Antoniou et al. [7] utilised an image condition GAN referred to as DAGAN to generate new samples to improve the few-shot learning performance. Meanwhile, Hariharan et al. [48] proposed a sample reconstructor that projects a sample to another sample of the same category.

2.5.2 Recent Exploration

Since 2018, more research studies have contributed to few-shot learning. Recent ideas include extensions to classical few-shot learning methods and extending few-shot learning approaches from supervised learning to semi-supervised learning.

For the distance metric-based sub area, Ye et al. and Puch et al. [132, 96] extended previous approaches (based on L1 or L2 loss) by constructing triplet loss. In addition, Gidaris et al. [39] proposed a classification weight generator. In detail, Gidaris et al. utilised a cosine-similarity based recognition model and attention-based weight generator to improve few-shot classification performance for both novel and base categories. In [40], they leveraged a Graph Neural Network (GNN) architecture based Denoising Autoencoders (DAE) to reconstruct target-discriminative classification weights. Furthermore, in 2019, Allen et al. [2] proposed an Infinite mixture prototypes (IMP) approach that can represent both simple and complex data distributions for few-shot learning. IMP integrated the advantage of deep representation learning and the Bayesian nonparametric by representing each class with scalable clusters rather than a single cluster which is widely used in previous prototypical methods. IMP approach pursued a balance point between nearest neighbor and prototypical representation approaches. In 2020, Ji et al. [57] utilised intra-class distribution information to improve current Prototypical Networks.

For the model-based sub area, Doveh et al. [32] proposed a Meta-learned task-adaptive (MetAdapt) architecture using Network Architecture Search (NAS) cell to optimize the model architecture and to adaptively change itself given novel few-shot tasks. MetAdapt assists in alleviating over-fitting issues caused by very small samples of various few-shot tasks. Though it is a model-based method, it only applies NAS to the last block of the network architecture rather than the full model.

For the optimization based sub area, Antoniou et al. [6] identified problems of the original MAML, which resulted in unstable training, high dependence on hyper-parameter, and high computation and time cost on training and inference. These problems also limit the flexibility and generalization ability of the framework. Antoniou et al. proposed corresponding methods to address each issue in MAML and demonstrated that learning rates, batch normalization parameters, and target losses, when learned and optimized on a per-step basis, can significantly improve the original MAML.

For the data augmentation sub area, Zhang et al. [139] proposed a MetaGAN approach similar to DAGAN. MetaGAN utilised GAN to generate fake samples rather than real ones and trained the generator and classifier simultaneously to learn sharper decision boundaries. MetaGAN can address both supervised and semi-supervised few-shot learning. Meanwhile, Wang et al. [128] combined meta-learner with a "hallucinator" similar to MetaGAN and end-to-end optimized both models. The hallucinator creates an additional supervised training set, taking existing samples with added noise as input. These approaches are extension studies of data augmentation approaches.

Few-shot learning approaches are developing at high speed. These methods may belong to different sub areas or have been implemented in different application fields. Comparing the performance of various few-shot learning approaches under the same scale is important to estimate their impact and benefit later researchers. However, the different experiment setting details of these algorithms make effective comparison challenging. For this reason, Chen et al. [20] presented a consistent comparative analysis of several classic few-shot classification algorithms and showed that deeper backbones could reduce the performance variation among different approaches when base data and novel data are from a similar domain. It is possible to achieve the state-of-the-art performance using only a modified baseline method. This result guides later researchers, especially those in cross-domain studies, to initially attempt basic methods and focus on model fine-tuning.

2.5.3 Few-shot Learning Application

Multiple application domains have benefited from few-shot learning techniques. The most popular application of few-shot learning is on computer vision, such as object recognition and tracking ([31]), talking head image generation ([136]) and face reenactment ([47]).

Dong et al. [31] implemented fast visual object tracking with improvement in the robustness of representations by replacing the twin branches and pair-loss of the Siamese network with four branches and a combination loss of pair-loss and triplet-loss. Zakharov et al. [136] proposed a framework utilizing adversarial learning (generator and discriminator) to create a talking head image through several, even one photograph. The generated talking head images can be conditioned on the landmarks extracted from the same or different persons, but the quality of generated images based on landmarks of another person is mixed. In 2020, to address this problem, that is, the facial characteristics mismatch between target and driver caused by using another person's landmarks, Ha et al. [47] proposed a few-shot learning framework that combines landmark transformer and image attention modules with autoencoder based target feature alignment mechanism. The landmark transformer can disentangle the expression geometry from the landmarks and significantly mitigate the identity mismatch issue.

In addition, more studies for application on network traffic analysis have been proposed since 2020. For example, in the network intrusion detection area, Xu et al. [131] proposed a deep neural network based meta-learning framework to distinguish a normal sample from a malicious one. In 2021, Wang et al. [129] built a Siamese capsule network to handle imbalanced data challenges in intrusion detection because it can dynamically extract the relationship among traffic features. Based on previous research, Duan et al. [33] surveyed few-shot learning methods applied in intrusion detection. They introduced approaches like embedded learning, multi-task learning, and generative models, indicating that insufficient data is the most important challenge. Furthermore, Chen et al. [19] explored new data augmentation methods for few-shot website fingerprinting attacks, including intra-sample transformation (i.e., random rotation and masking) and inter-sample transformation (i.e., randomly mixing two traffic samples), to address the data insufficiency issue. Liu et al. [74] combined a classic few-shot learning model with three representations of communication signals to identify the jamming. It is demonstrated that the model architecture based on MAML can automatically extract the features from signals and address the shortcomings of handcrafted feature extraction.

2.5.4 Few-Shot Open-set Recognition

Few-shot learning approaches have addressed the problem of classification on a new task. However, most approaches still cannot handle unseen categories for the support set of new tasks. Recently, this problem has gained much attention.

There are a few SNN based open-set few-shot learning approaches. Siamese Autoencoder network (SiA) ([123]), an anomaly detection approach, utilised Mahalanobis distance on feature space for anomaly detection and preserved the original data structure during feature extraction. In [105], they proposed an open-set face recognition approach for small galleries based on SNN. This method replaced a CNN based SNN with a fully connected layers based SNN, and the OSR method was based on the softmax threshold for the output of the SNN.

Further open-set few-shot learning research is based on the prototype network. An oPen sEt mEta LEaRning (PEELER) algorithm proposed in [72] introduced unseen categories into the training process and modified the original loss function by adding open loss. Open deep network based on the prototype (P-ODN) ([114]) introduced manual labeling into training, so it gave open-set recognition scalability. Previous identified unseen samples will be manually labeled as a new category and added into later training. The number of categories is also changed to include the new category. The P-ODN trained prototypes and prototype radii obtained more distinguishable features and proposed a multi-class triplet thresholding method for OSR. In [56], they utilised a reconstruction mechanism on the feature prototypes. This method trains a transformer function(reconstruction network). It applies this function to both the original and modified prototype sets (replace the predicted category prototype with the query feature). If the distance between these two transformed prototype sets is more significant than a threshold, the query sample will be identified as an unseen category. In [52], they proposed a method to modify the feature prototype and generate a negative prototype. By leveraging an attention mechanism according to the base category memory, support samples of each task were used to generate the negative prototype.

Compared to previous work, I extend the few-shot learning application to fine-grained network traffic classification (i.e., content level) and propose a similarity controllable data augmentation method. Also, the proposed hierarchical cross entropy loss and DPGMM based open-set metric can adapt to dynamically changing tasks with highly diverse support classes in training and test processes, respectively.

2.6 Summary

To summarise, this chapter provides a comprehensive overview of existing literature pertinent to the research questions of this thesis. It covers traditional methods as well as machine learning and deep learning methods for network traffic analysis, feature engineering, novelty detection, and few-shot learning. While substantial progress has been made in these areas, gaps and challenges still exist. One example is the difficulty of content-level classification for deeply encrypted network traffic (i.e., encrypted on the Data Link Layer) in an open-world setting where sample sizes are small and classification tasks are dynamically changing. These gaps not only serve as the motivation for this research but also as the direction for the specific research aims outlined in Section 1.1. In the subsequent chapters, I will systematically present the methodologies and findings aimed at addressing these research questions.

Chapter 3

Classification for Encrypted Network Traffic in the Closed World

This chapter is to address the first research question, "Is it feasible to identify the content of deeply encrypted network traffic?" This chapter is the foundation of this thesis because it includes the details of the data collection and data processing of the new encrypted network traffic dataset captured in air. I demonstrate that using a deep learning model to address the fine-grained (content level) encrypted network classification in the closed world is feasible even with limited metadata and can achieve good performance. Then the subsequent work can utilise the dataset and feature engineering results from this chapter to address more challenging classification problems in the open world.

3.1 Encrypted Network Traffic Data Collection and Processing

This section describes encrypted network traffic data collection, data preprocessing, including data filtering and frame type identification, and feature engineering procedures.

3.1.1 Data Collection

For verifying the feasibility of eavesdropping on encrypted wireless network traffic, data is collected in an environment that consists of two laptops, a WiFi wireless network of a university, an Airpcap sniffer, and the Wireshark software (see Figure 1.1).

One laptop, connected to a WiFi Access Point (AP) using a certain channel of the 2.4 GHz spectrum, like channel 1 or channel 6, is treated as the victim. Sometimes, the laptop is connected to a channel first and then re-connected to another channel because of internet interruption or insufficient channel capacity on the original channel. One Airpcap USB device plugged into another laptop treated as an eavesdropper can only monitor one channel. According to the observation, the victim laptop is mostly connected to channel 6 in the data collection environment. Therefore, Airpcap is set to monitor channel 6 in our data collection. The WiFi network of this data collection environment follows the 802.11n standard with WPA2 encryption. The victim laptop is manipulated by a script to mimic streaming traffic by repeatedly playing the target contents (video, audio, and web surfing).

I consider the time consumption and the necessity of adequate data to avoid model over-fitting. I first only select 10 videos on the YouTube platform (the most popular video platform) to evaluate the feasibility of identifying the video through the encrypted traffic captured in air. I name these video samples as the classification dataset in this thesis. Then we extend the content type from video to audio and web surfing to assist in solving more challenges, e.g., novelty detection. I select 10 videos on Netflix; 10 songs on YouTube Music; and 10 web page lists from Wikipedia. Here, I still set the number of videos, songs, and web page lists as 10 to maintain consistency with the number of YouTube videos. I name these samples as the novelty detection dataset. Finally, more classes of video samples are collected for few-shot learning. Few-shot learning usually evaluates the performance for 5-way, 10-way, and even 20-way n-shot query samples. Therefore, more than 10 classes are needed to represent the variety of the training samples. Data collection increases the classes of videos and expands the platforms, doubling the initial number of classes (10) on three popular video platforms (Netflix, Stan, and YouTube). These 60 classes (i.e., $2 \times 10 \times 3$) of samples are utilised to evaluate the new idea in few-shot learning. I name these samples as the few-shot learning dataset.

Through a web browser, one laptop (victim) accesses the selected content repeatedly (about 300 times for YouTube videos and about 100 times for other content). Another laptop (eavesdropper) captures each stream's first 180 seconds of traffic. On this eavesdropper laptop, the AirPcapNx* is used as a sniffer to collect frames streaming on a channel. At first, collecting about 300 runs of video streaming for each video on YouTube ensures the samples are enough to overcome over-fitting issues caused by limited samples. After the feasibility evaluation of using deep learning techniques to classify the video streaming contents, the sample volume (300 runs per class) is proved more than enough. Considering the time cost, only about 100 runs are collected for each class in the subsequent data collection. Regarding the traffic length, I make a trade-off between the selection of Reed et al. [99] (about 2 minutes) and Schuster et al. [109] (about 5 minutes), capturing 3 minutes of traffic.

When collecting the traffic from YouTube and YouTube Music, the advertisements at the beginning are removed by using a premium account to avoid their interference. In addition, I set up the system to automatically open 18 URLs one by one at 10-second intervals to obtain a traffic sample of 180 seconds. I have observed and estimated that the average time for opening and browsing a news website is several seconds. Therefore, I select an integer of 10 seconds as the time interval to simulate simplified web surfing operations.

Overall, we collected 11,529 samples, 3,198 (YouTube video) samples for the classification dataset, 1,749 (Netflix video, YouTube Music song, and Wikipedia web page list) samples for the novelty detection dataset, and 6,000 samples (Netflix, Stan, and YouTube video) for the few-shot learning dataset.

It is worth noting that, during the data collection, some factors may cause the

*`https://www.riverbed.com`

failure of the traffic capture, for example, internet interruption, Wireshark fault, and channel hopping. Therefore, to improve the sample quality, it is necessary to check the captured traffic and ensure it contains streaming data through the size of the pcap files. The normal pcap file size is about tens of Megabytes, but the invalid pcap file size is about several Megabytes or less. The invalid samples are discarded. Then each class usually does not have N valid samples after N runs. For example, when collecting YouTube video traffic samples, a video needs more than 300 runs to contain at least 300 valid samples. At last, we collected 3,198 rather than just 3,000 samples. For Netflix video, YouTube Music song, and Wikipedia web surfing data collection, we captured 3,000 samples, but only 2,331 of them are valid, which are then split into training (1,749) and test (582) set (utilised by other tasks of the project). The training set (1,749) samples are used in this thesis for novelty detection experiments as outliers. Then we collected 20 class samples on Netflix, Stan, and YouTube, respectively, using the same method as stated in YouTube video data collection, but we only kept 100 valid samples per class to make the number easy to remember.

3.1.2 Data Preprocessing

Data Filter

The Airpcap sniffer captures all frames available on specified channel in its vicinity. According to IEEE 802.11 protocol, WiFi uses WPA-2 for frame encryption. Therefore, I cannot extract information about the Network Layer and higher protocol layers, such as IP addresses and port numbers. I only can access a few basic information from the Data Link Layer, such as the size, type, duration time of a frame, the source MAC addresses, the destination MAC addresses, and the radio information (i.e., signal strength and noise level). I also collect these parameters for the traffic flowing through (in and out) the target victim in each direction. I consider the MAC address of the target victim as a unique identifier to filter the captured traffic. Then I analyze all of the frames from and to this target victim.

Frame Type Identification

There are three types of 802.11 frames, namely, management frames, control frames, and data frames. Among them, data frames contain the payload including information in Network, Transport and Application Layers. Though the payload is encrypted, the temporary size of the data frames (during a time period) still may reflect the streaming data profile. Meanwhile, the management frames are used to manage the Basic Service Set (BSS) like probing, roaming, and disconnecting clients from the BSS. Control frames are related to medium access and frame acknowledgement. The time-sequential number of management and control frames during a time period may reflect the communication pattern between source and destination. To obtain the target traffic streaming between the victim and the Access Point (AP), I first filter all types of frames by the MAC address of the target victim. Then I use the frame size parameter (greater than 64 bytes) to select the data frame. Other control and management frames (less than 64 bytes) are referred to as non-data frames.

3.1.3 Feature Engineering

The first three minutes (180 seconds) of each stream are captured and then grouped according to the traffic direction (i.e., up-link, down-link, and combination (up-link and down-link)). I investigated the I/O graphs through Wireshark and found that a 1-second time interval allows for a clear visual observation of the traffic pattern, although some details may be lost. In contrast, a 10-millisecond time interval tends to obscure the patterns due to excessive detail. Therefore, I selected a time interval between 10 milliseconds and one second for binning the packets. Meanwhile, the number of bins should be about 512 or 256, which are commonly

Table 3.1 : Constructed feature list on three traffic directions: uplink (U), downlink (D), and combination of up and down (C).

Feature ID	**Feature Meaning**
F1 (U, D, C)	The number of data frames in a time bin
F2 (U, D, C)	The number of bytes for data frames in a time bin
F3 (U, D, C)	The number of non-data frames
F4 (U, D, C)	The number of bytes for non-data frames in a time bin
F5 (U, D, C)	The minimum frame size in a time bin
F6 (U, D, C)	The maximum frame size in a time bin
F7 (U, D, C)	The average frame size in a time bin
F8 (U, D, C)	The variance frame size in a time bin

used as the number of hidden neurons in neural networks, to keep a similar number of parameters when utilizing the classic deep learning models. Finally, I set the bin size as 0.36, which is similar to the setting in [109], and obtain 500 bins for each stream. For each direction, I compute features based on certain characteristics within each bin. The constructed four primary features include the number of data frames, the number of bytes for data frames, the number of non-data frames (i.e., management and control frames), and the number of bytes for non-data frames. Furthermore, to represent the traffic from more perspectives, I construct four additional features: the minimum frame size, the maximum frame size, the average frame size, and the variance frame size. These features are summarised in Table 3.1. In this thesis, the concept of 'features' differs from traditional machine learning, where the number of handcrafted features usually matches the dimension of a processed sample. This implies that the number of features is 12,000 (500×24) according to the definition of traditional machine learning methods. However, the 500 features are the bins for the entire time sequence traffic, and each bin does not have an independent meaning.

Therefore, I treat the eight binning methods across the three different directions as features, resulting in 24 features in total.

3.2 Model Architecture

In this section, I implemented a CNN model to evaluate the classification performance of each constructed feature on the collected network traffic dataset in the closed world.

CNN model has been employed to identify the content of the traffic in [109] and achieved excellent performance. The task goal of [109] is similar to mine in this work. In addition, the studies in [69, 24] also demonstrated, in traffic fingerprinting, CNNs outperform RNNs, which are generally suited for time series data. Furthermore, works like WaveNet [90] showed that noisy time series data is better modelled by 1D CNNs. Therefore, I utilise CNN model to evaluate the feasibility of unveiling the video stream captured in air. It is worth mentioning that, in my experiments, I apply the CNN model to the traffic captured in air through sniffing rather than traffic captured through a wired port. The architecture of the CNN model is shown in Figure 3.1. This model consists of three convolution layers, two fully connected layers and a max-pooling layer. I select the Adam optimizer and train this model with batch size 64. To simplify the feasibility evaluation experiment, I keep the parameters (e.g., model network architecture, optimizer and batch size) the same as the experimental settings in [109].

3.3 Experiments and Results

3.3.1 Experimental Setup

I use the YouTube video traffic captured by the method stated in Section 3.1.1 which consists of 3,198 video streaming samples of size 500×24 in 10 videos, with about 300 video streaming samples per video. The dataset is randomly permu-

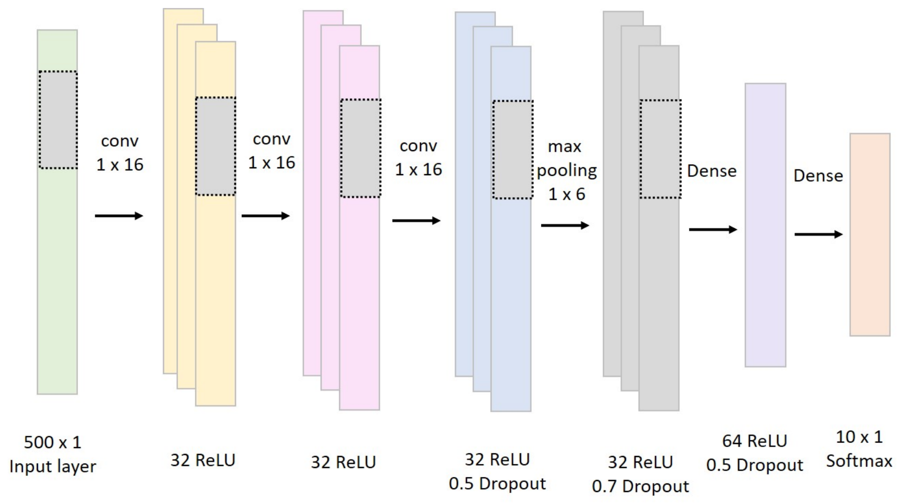


Figure 3.1 : CNN Model Architecture.

tated and split into a training set (80%) and a testing set (20%). Even though, the training-test split ratios, like 8:2, 7:3 and 6:4 are common in practice, the 8:2 ratio that often referred to as the Pareto principle is the most popular. Hence, I use this split ratio. The other experimental parameters are listed in the Appendix 7.1. I use the same Batch size, activation, optimizer as in the settings in [109]. Regarding the learning rate, batch normalisation decay and batch normalisation epsilon, I selected the specific values based on experiment evaluation. For example, I attempted learning rate from 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-1}, and found 10^{-4} can quickly converge and achieve nearly the same accuracy. Therefore, I set the learning rate as 10^{-4}.

3.3.2 Classification Performance Analysis

I apply the CNN model architectures on video streaming traffic classification. I train the 1D CNN model with each feature (i.e., each 500×1 vector) listed in Table 3.1 individually. The performance for each feature is shown in Figure 3.2.

I notice that the features obtained from the captured traffic profiles directly, such as F1 (U, D, C), F2 (U, D, C), F3 (U, D, C), and F4 (U, D, C), can achieve excellent performance (with accuracy about 97%). On the other hand, the performance of features generated from the statistic of the captured data, i.e., F5 (U, D, C), F6 (U, D, C), F7 (U, D, C), and F8 (U, D, C), is mixed. For example, the F6 (D, C), F7 (D, C), and F8 (C) perform competitively with the F1, F2, F3, and F4. However, the accuracy with F5 (D), F8 (D), F5 (C), and the F5, F6, F7 and F8 on the up-link is between 92% and 95%.

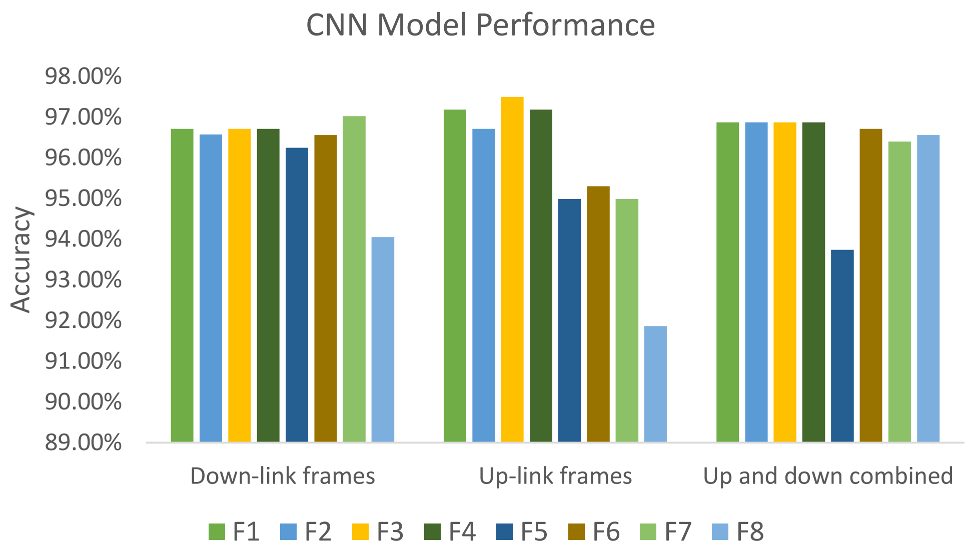


Figure 3.2 : CNN Model Performance.

Intuitively, I predict that the number of data frames would perform best because the video content data is encapsulated in the payload of a data frame. This assumption can be explained by the network traffic patterns of different videos as shown in Figure 3.3. This figure demonstrates that the traffic profile (i.e., the fluctuation of “Frame count” along with time) is a crucial characteristic for distinguishing between videos. However, the experiments show that video streaming also can be accurately classified using non-data frames. As stated before, non-data frames include the management and control frames related to communication patterns between two nodes. The experiment results indicate that both the bursts of video content streaming and the interaction process between the server and the client carry useful information for identifying streaming videos. Compared to the findings in [109], this thesis shows that non-data frames are still meaningful for content-level encrypted network traffic classification tasks. It is reasonable to combine data and non-data features to achieve higher performance.

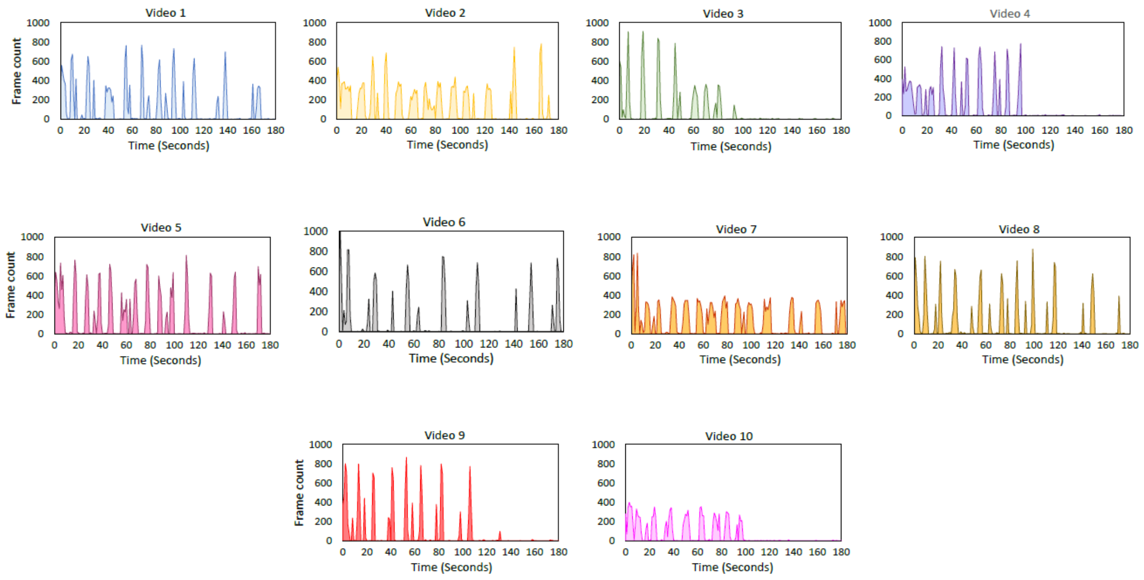


Figure 3.3 : Network Traffic Patterns for 10 Different Videos.

3.4 Summary

To summarise, this chapter i) explains data collection and data preprocessing details of the new dataset, ii) conducts feature engineering, which guides the future feature selection of the subsequent research (combining the data and non-data features), iii) evaluates the feasibility of using a deep learning model to classify encrypted network traffic on the content level in the closed world. Additional research about the closed world classification for encrypted network traffic captured in air is discussed in my collaborative paper [69]. This thesis focuses on the following open world classification, including novelty detection and few-shot open-set recognition.

Chapter 4

Classification Models for Encrypted Network Traffic in the Open World

I have answered the first research question, “Is it feasible to identify the content of deeply encrypted network traffic?” in Chapter 3. I have determined that deep learning techniques are feasible for fine-grained (content level) network traffic classification in a deeply encrypted scenario. Deep learning techniques do not depend on elaborately designed features that need experienced domain knowledge and enough metadata. However, I only discussed classification tasks in the closed world, which is too idealized to adapt to the real world, which is an open world. Ensuring all the test samples belong to the trained classes is unrealistic. Therefore, to adapt to the real world, I raise the second research question: “How to handle classification for previously unseen classes in the open world?” I will continue exploring deep learning techniques to address this question in this chapter. After investigating existing studies and analyzing their advantages and limitations, as stated in Sections 2.4 and 1.1.2, I propose two methods to address this question. One is a new model based on framework innovation, and the other is based on algorithm innovation. In this chapter, I will describe the two models in detail, covering model framework, model algorithm, experimental settings, and result analysis.

4.1 Calibrated Reconstruction Based Adversarial AutoEncoder (CRAAE) Model

I will use novelty detection techniques to detect previously unseen classes (i.e., outliers). Outliers can be located far from or near the known classes (i.e., far distance outlier and close distance outlier). Existing novelty detection approaches usually conduct detection only for one type of outlier. However, in the real world, whether the outliers are far from or close to the inliers is unknown. Taking advantage of two mainstream approaches (MCOSR and OCND), I propose the CRAAE Model, which integrates implicit and explicit confidence calibration strategies into a reconstruction-based model to implement agnostic location novelty detection. The proposed CRAAE is expected to generate a more accurate decision boundary shown in Figure 4.1.

4.1.1 Methodology for CRAAE

In this section, I first present an overview of the proposed CRAAE model framework and then present the training and inference process of the model. Finally, I describe the specific model optimizations for each module.

CRAAE Model Framework

CRAAE is a multi-function, lightweight integrated framework based on AAE. The first contribution, as shown in Figure 4.2, is using the category component to classify inliers. This category component also serves as the explicit calibrator for the novelty detection score, improving the accuracy of the decision boundary. The second contribution, shown in Figure 4.3, involves generating samples via the decoder using random inputs from statistical distributions like Uniform and Dirichlet to represent more generalized and precise outliers. Finally, as shown in Figure 4.4, I feed these generated samples into the encoder as fake inputs for the training process.

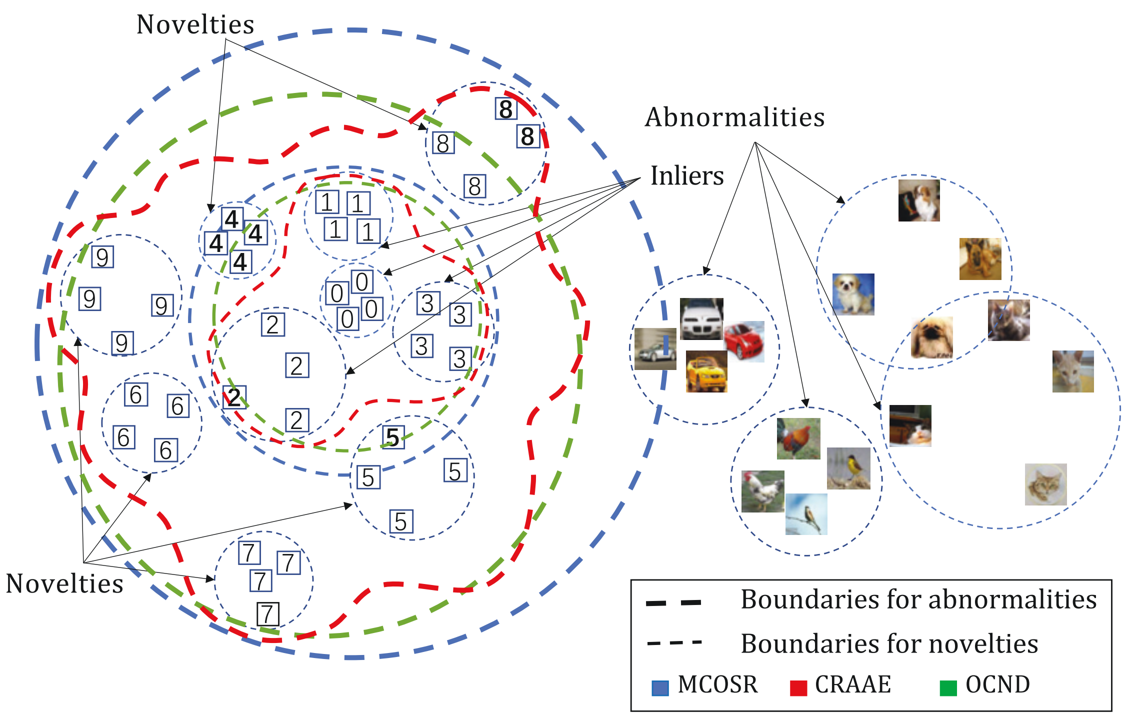


Figure 4.1 : A schematic view of novelty detection decision boundaries with CRAAE, MCOSR and OCND approaches for different location outliers, such as abnormalities and novelties.

I compute KL loss for fake inputs and cross-entropy loss for true inputs, introducing the third contribution: implicit calibration of the decision metric.

As shown in Figure 4.4, the complete CRAAE framework includes one encoder **E**, one decoder **G**, and two discriminators $\mathbf{D_x}$ and $\mathbf{D_z}$. Both modules **E** and **G** play multiple roles. **E** is the latent representation extractor as well as the classifier. In addition, it also acts as a calibrator and provides both explicit and implicit calibration to the decision scores of novelty detection. **G** is the original input reconstructor and the known outlier generator.

E consists of four CNN layers that extract category and style features from the input. **G** also consists of four deconvolutional layers corresponding to **E**. Two

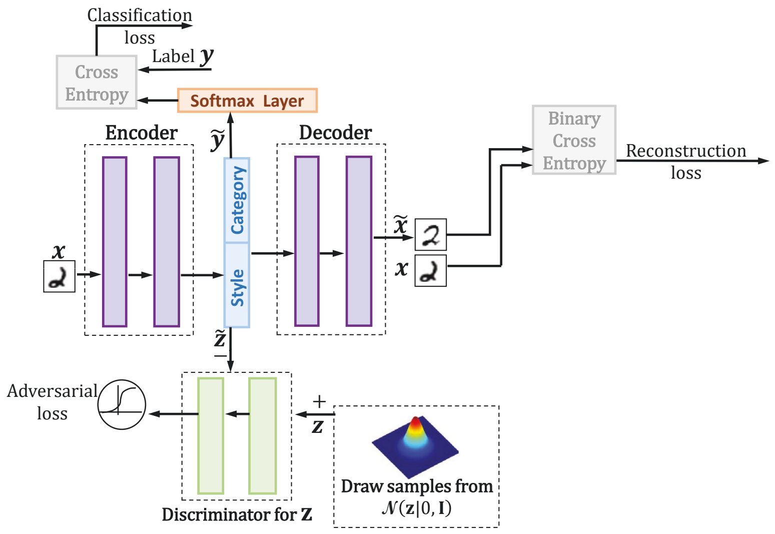


Figure 4.2 : Calibrated Reconstruction Based Adversarial Autoencoder (CRAAE) Model Framework - Contribution 1.

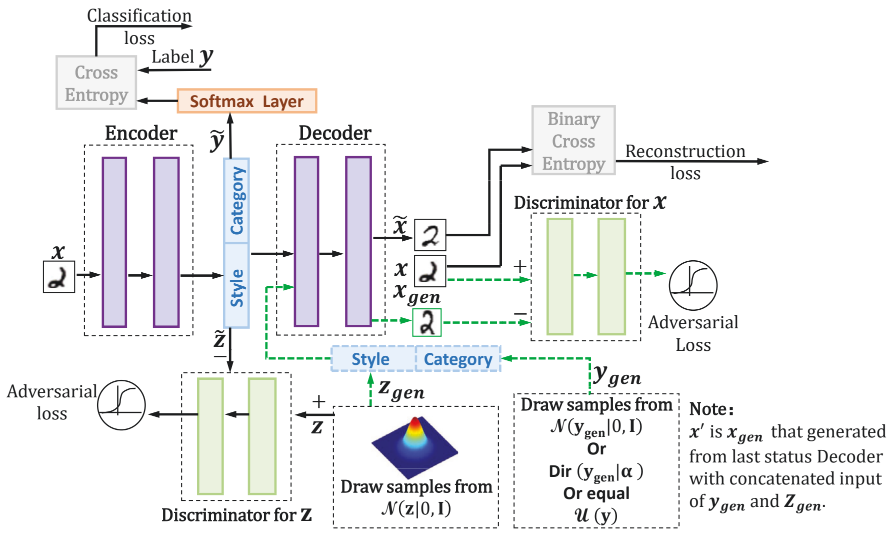


Figure 4.3 : Calibrated Reconstruction Based Adversarial Autoencoder (CRAAE) Model Framework - Contribution 2.

discriminators assist **E** and **G** respectively for adversarial learning. $\mathbf{D_z}$ forces the encoded style component to follow standard Gaussian distribution while $\mathbf{D_x}$ distinguishes between original and generated inputs. I describe the dataset-specific changes to the model architecture in Section 4.1.2.

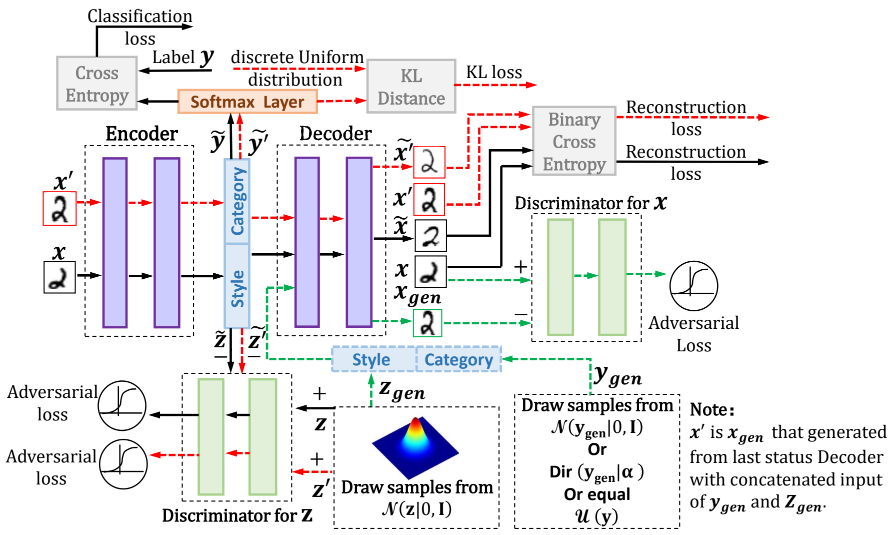

Classification loss
Label y
Cross Entropy
discrete Uniform distribution
Softmax Layer
KL Distance
KL loss
Binary Cross Entropy
Reconstruction loss
Reconstruction loss
Encoder
Decoder
Category
Style
Discriminator for x
Adversarial Loss
y_{gen}
z_{gen}
Adversarial loss
Adversarial loss
Discriminator for Z
Draw samples from $\mathcal{N}(z|0,I)$
Draw samples from $\mathcal{N}(y_{gen}|0,I)$ Or Dir $(y_{gen}|\alpha)$ Or equal $\mathcal{U}(y)$
Note: x' is x_{gen} that generated from last status Decoder with concatenated input of y_{gen} and z_{gen}.

Figure 4.4 : Calibrated Reconstruction Based Adversarial Autoencoder (CRAAE) Model Framework - Contribution 3.

Training and Inference Processes

Before detailing the training and inference processes, I first explain the symbols and variables in Figure 4.4. $\boldsymbol{x}$ is the true input (inlier) and $\boldsymbol{x'}$ is the fake input (known outlier). $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}'$ are the reconstructions corresponding to $\boldsymbol{x}$ and $\boldsymbol{x'}$. $\boldsymbol{x}_{gen}$ is the generated fake input from synthetic latent representation. $\boldsymbol{y}$ is the ground truth label corresponding to $\boldsymbol{x}$. $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}}'$ represent the category component of the latent representation extracted from $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. $\boldsymbol{y}_{gen}$ is the synthetic category component. Similarly, $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{z}}'$ represent the style component of the latent representation from $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. $\boldsymbol{z}$ and $\boldsymbol{z'}$ are drawn from the standard Gaussian distribution as the adversarial learning targets of $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{z}}'$. $\boldsymbol{z_{gen}}$ is the synthetic style component. Next, I describe the training and inference process.

The black solid lines and the red dotted lines indicate the training procedure of the true input $\boldsymbol{x}$ and the fake input $\boldsymbol{x'}$, respectively. The green dotted lines denote the generation procedure of $\boldsymbol{x}_{gen}$ which is used as $\boldsymbol{x'}$. The difference between the training procedures of $\boldsymbol{x}$ and $\boldsymbol{x'}$ is the processing of category components. $\tilde{\boldsymbol{y}}$ and

$\tilde{\boldsymbol{y}'}$ are fed into the softmax layer, then $\mathbf{s}(\tilde{\mathbf{y}})$ and $\mathbf{s}(\tilde{\mathbf{y}'})$, the output of the softmax layer from $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}'}$, are separately used to compute the cross entropy loss with the ground truth $\boldsymbol{y}$ and the KL loss with a discrete Uniform distribution $\mathcal{U}(\boldsymbol{y})$.

It is worth noting that the reason why I use KL loss rather than cross entropy loss for minimizing the distance between $\mathbf{s}(\tilde{\mathbf{y}'})$ and $\mathcal{U}(\boldsymbol{y})$. In general, the KL distance of distribution $\mathcal{P}$ and $\mathcal{Q}$ (see the Equation 4.1) equals the cross entropy of $\mathcal{P}$ and $\mathcal{Q}$ (first term in Equation 4.1) subtracts the entropy of $\mathcal{P}$ (second term in Equation 4.1), where $\mathcal{P}$ is the distribution of the actual data while $\mathcal{Q}$ is the hypothetical distribution.

$$\mathcal{D}_{KL}(\mathcal{P}\|\mathcal{Q}) = -\sum_i \mathcal{P}_i \log \mathcal{Q}_i - (-\sum_i \mathcal{P}_i \log \mathcal{P}_i). \tag{4.1}$$

In my model, $\mathcal{P}$ is $\mathbf{s}(\tilde{\mathbf{y}'})$ and $\mathcal{Q}$ is $\mathcal{U}(\boldsymbol{y})$. If $\mathbf{s}(\tilde{\mathbf{y}'})$ is for samples from a given dataset, the entropy of $\mathbf{s}(\tilde{\mathbf{y}'})$ can be regarded as a constant. However, $\mathbf{s}(\tilde{\mathbf{y}'})$ is for generated samples in my model and the generated samples are continuously changed during the training. Therefore, the second term of KL distance is not a fixed value which can be ignored.

During the generation procedure, $\boldsymbol{y}_{gen}$ can be a discrete Uniform distribution or a distribution sampled from a Dirichlet distribution other than being drawn randomly from a Gaussian distribution. My motivation is to describe $\boldsymbol{y}_{gen}$ with a distribution that is similar to the softmax output, not only a randomly sampled vector. I treat each softmax output vector as a discrete distribution because the sum of each softmax output entry equals 1, and the value of each entry is between 0 and 1.

In the inference process, first, I feed the training set X_{train} into the trained $\mathbf{E}$ and $\mathbf{G}$ to obtain $\tilde{\boldsymbol{Z}}_{train}$ and $\tilde{\boldsymbol{X}}_{\boldsymbol{train}}$ which are used for computing the two terms of the novelty detection decision score, $\log P_{\tilde{\boldsymbol{Z}}_{train}}(\tilde{\boldsymbol{z}})$ and $\log P_{\boldsymbol{X}_{train}}(\boldsymbol{x})$. I fit $\tilde{\boldsymbol{Z}}_{\boldsymbol{train}}$ to a generalized Gaussian distribution and compute the probability density function of

X_{train} based on the histogram of the reconstruction error $||\boldsymbol{X}_{train} - \tilde{\boldsymbol{X}}_{\boldsymbol{train}}||$. The algorithm details for the two items are stated in [94]. Next, the test samples are fed into $\mathbf{E}$ and then softmax layer to obtain $\tilde{\boldsymbol{y}}_{\boldsymbol{test}}$ and the softmax vectors. Finally, the maximum value of the softmax vector is utilised as the explicit calibration factor to weight the novelty detection decision score, which is given by

$$(\log P_{\tilde{\boldsymbol{Z}}_{train}}(\tilde{\boldsymbol{z}}_{test}) + \log P_{\mathbf{X}_{train}}(\boldsymbol{x}_{test}))/\max_{i=1\cdots K}(\frac{e^{\tilde{\boldsymbol{y}}_{test,i}}}{\sum_{j=1}^{K} e^{\tilde{\boldsymbol{y}}_{test,j}}}), \tag{4.2}$$

where K is the class number.

CRAAE Model Optimization Update

I optimize each module of the CRAAE model through the following objective function, which consists of five terms:

$$\begin{aligned}&\mathcal{L}_{\mathbf{GE}}(\boldsymbol{x}, \mathbf{E}, \mathbf{G}, \mathbf{D}_{\boldsymbol{z}}) + \mathcal{L}_{\mathbf{E}}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{E}, \mathbf{G}) + \mathcal{L}_{\mathbf{G}}(\mathbf{G}, \mathbf{D}_{\boldsymbol{x}}) \\ &+\mathcal{L}_{\mathbf{D}_{\boldsymbol{x}}}(\boldsymbol{x}, \mathbf{G}, \mathbf{D}_{\boldsymbol{x}}) + \mathcal{L}_{\mathbf{D}_{\boldsymbol{z}}}(\boldsymbol{x}, \mathbf{E}, \boldsymbol{G}, \mathbf{D}_{\boldsymbol{z}}).\end{aligned} \tag{4.3}$$

I update $\mathbf{D}_{\boldsymbol{x}}$ by maximizing adversarial loss $\mathcal{L}_{\mathbf{D}_{\boldsymbol{x}}}(\boldsymbol{x}, \mathbf{G}, \mathbf{D}_{\boldsymbol{x}})$ which is expressed by

$$\mathcal{L}_{\mathbf{D}_{\boldsymbol{x}}}(\boldsymbol{x}, \mathbf{G}, \mathbf{D}_{\boldsymbol{x}}) = \mathbb{E}[\log \mathbf{D}_{\boldsymbol{x}}(\boldsymbol{x})] + \mathbb{E}[\log(1 - \mathbf{D}_{\boldsymbol{x}}(\boldsymbol{x}_{gen}))], \tag{4.4}$$

where $\boldsymbol{x}_{gen} = \mathbf{G}([\boldsymbol{y}_{gen}, \boldsymbol{z}_{gen}])$, $\boldsymbol{z}_{gen}$ is drawn from the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$), and $\boldsymbol{y}_{gen}$ can be a discrete Uniform distribution, a distribution drawn from a Dirichlet distribution or a vector randomly drawn from a Gaussian distribution.

I update $\mathbf{G}$ through minimizing adversarial loss $\mathcal{L}_{\mathbf{G}}(\mathbf{G}, \mathbf{D}_{\boldsymbol{x}})$ shown as

$$\mathcal{L}_{\mathbf{G}}(\mathbf{G}, \mathbf{D}_{\boldsymbol{x}}) = -\mathbb{E}[\log \mathbf{D}_{\boldsymbol{x}}(\mathbf{G}([\boldsymbol{y}_{gen}, \boldsymbol{z}_{gen}]))]. \tag{4.5}$$

I update $\mathbf{D}_{\boldsymbol{z}}$ by maximizing adversarial loss $\mathcal{L}_{\mathbf{D}_{\boldsymbol{z}}}(\boldsymbol{x}, \mathbf{E}, \boldsymbol{G}, \mathbf{D}_{\boldsymbol{z}})$ given by

$$\begin{aligned}\mathcal{L}_{\mathbf{D}_{\boldsymbol{z}}}(\boldsymbol{x}, \mathbf{E}, \boldsymbol{G}, \mathbf{D}_{\boldsymbol{z}}) =&\mathbb{E}[\log \mathbf{D}_{\boldsymbol{z}}(\boldsymbol{z})] \times 2.0 + \mathbb{E}[\log(1 - \mathbf{D}_{\boldsymbol{z}}(\tilde{\boldsymbol{z}}))] + \\ &\mathbb{E}[\log(1 - \mathbf{D}_{\boldsymbol{z}}(\tilde{\boldsymbol{z}'}))],\end{aligned} \tag{4.6}$$

where $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{z}'}$ is the output of $\mathbf{E}(\boldsymbol{x})$ and $\mathbf{E}(\boldsymbol{x}')$, respectively, and $\boldsymbol{z}$ is drawn from the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$). Here, $\boldsymbol{x}' = \boldsymbol{x}_{gen}$, and $\boldsymbol{x}_{gen}$ is generated by $\mathbf{G}$.

I update $\mathbf{E}$ through minimizing $\mathcal{L}_{\mathbf{E}}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{G}, \mathbf{E})$ which includes the KL loss and the cross entropy loss

$$\mathcal{L}_{\mathbf{E}}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{G}, \mathbf{E}) = \mathbb{E}[\mathcal{D}_{\mathbf{KL}}(\mathbf{s}(\tilde{\mathbf{y}'})) \parallel \mathcal{U}(\mathbf{y})] \times \beta - \mathbb{E}[\log \mathbf{P}(\mathbf{y} = \mathbf{s}(\tilde{\mathbf{y}})|\mathbf{x})], \tag{4.7}$$

where $\mathbf{s}(\cdot)$ is the softmax function, $\tilde{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}'}$ is the output of $\mathbf{E}(\boldsymbol{x})$ and $\mathbf{E}(\boldsymbol{x}')$, respectively, and β is the loss balance parameter for $\mathbf{E}$ (I set $\beta = 0.03$ here.).

I jointly update $\mathbf{E}$ and $\mathbf{G}$ through minimizing $\mathcal{L}_{\mathbf{GE}}(\boldsymbol{x}, \mathbf{E}, \mathbf{G}, \mathbf{D}_{\boldsymbol{z}})$ which includes following three parts: 1) the reconstruction loss $\mathcal{L}_{rec}(\boldsymbol{x}, \mathbf{E}, \mathbf{G})$ can be written as

$$\begin{aligned}\mathcal{L}_{rec}(\boldsymbol{x}, \mathbf{E}, \mathbf{G}) = &- \{\mathbb{E}[\boldsymbol{x} \log \mathbf{G}(\mathbf{E}(\boldsymbol{x}))] + \mathbb{E}[(1 - \boldsymbol{x}) \log(1 - \\ &\mathbf{G}(\mathbf{E}(\boldsymbol{x})))]\} \times \gamma,\end{aligned} \tag{4.8}$$

where γ is the loss balance parameter for $\mathbf{E}$-$\mathbf{G}$ (I set $\gamma = 2.0$); 2) the difference loss between the reconstruction loss for inliers and known outliers $\mathcal{L}_{rec_{diff}}(\boldsymbol{x}, \mathbf{E}, \mathbf{G})$ is shown as

$$\mathcal{L}_{rec_{diff}}(\boldsymbol{x}, \mathbf{E}, \mathbf{G}) = \max\left(\mathbf{0}, (\mathcal{L}_{\mathbf{rec}}(\mathrm{x}, \mathbf{E}, \mathbf{G}) - \mathcal{L}_{\mathbf{rec}}(\mathrm{x}', \mathbf{E}, \mathbf{G}))\right), \tag{4.9}$$

which restrains $\mathcal{L}_{rec}(\boldsymbol{x}, \mathbf{E}, \mathbf{G}) \geq \mathcal{L}_{\mathbf{rec}}(\mathrm{x}', \mathbf{E}, \mathbf{G})$; 3) the adversarial loss $\mathcal{L}_{\mathbf{E}}(\boldsymbol{x}, \mathbf{G}, \mathbf{E}, \mathbf{D}_{\boldsymbol{z}})$ is given by

$$\mathcal{L}_{\mathbf{E}}(\boldsymbol{x}, \mathbf{G}, \mathbf{E}, \mathbf{D}_{\boldsymbol{z}}) = -\{\mathbb{E}[\log \mathbf{D}_{\boldsymbol{z}}(\tilde{\boldsymbol{z}'})] + \mathbb{E}[\log \mathbf{D}_{\boldsymbol{z}}(\tilde{\boldsymbol{z}})]\}. \tag{4.10}$$

4.1.2 Experimental Setup

I evaluate the performance of CRAAE using the network traffic dataset and several standard image datasets. I select additional image datasets to demonstrate the universality of CRAAE model because novelty detection is critical in many real-world domains. I also compare the performance of my model with several state-of-the-art open-set recognition approaches. In the evaluation, I use several evaluation metrics such as the F1 score, the area under the receiver operating characteristic curve (AUROC), the false positive rate (FPR) at 95% true positive rate (TPR), the detection error, and the area under the precision-recall curve (AUPR).

Network Traffic Dataset

We collected the network traffic dataset for novelty detection in Section 3.1.1. The novelty detection dataset contains four subsets of HTTP video, audio, and web surfing network traffic captured over encrypted wireless networks.

For video traffic, this dataset has two subsets, corresponding to the YouTube and Netflix platforms. Audio traffic is captured while streaming music from YouTube Music, and web surfing traffic is captured when visiting Wikipedia web pages. More specifically, the YouTube video subset consists of 3,198 video streams of 10 videos, with approximately 300 video streams per video. The Netflix video, YouTube music, and Wikipedia subsets consist of 540 video streams, 603 audio streams, and 606 web surfing streams, respectively. Further detail of the network data capturing procedure and underlying network conditions can be found in Section 3.1.1. I represent each network trace as a vector of 2×500, where 500 is the number of bins over the trace length, and 2 represents the number of selected features. The number of non-data frames on the down-link (F3 (D)), and the number of data frames on both up-link and down-link (F1 (C)) were identified as the two most important features in [69].

My experiments are designed to evaluate the model performance in different outlier location scenarios. I simulate the close distance outlier detection scenario by using several classes of a dataset as inliers and others as outliers, which is named as *intra dataset novelty detection.* Meanwhile, the far distance outlier detection scenario is simulated by using outliers and inliers from different datasets, respectively. This scenario is named as *inter dataset novelty detection.* For intra dataset novelty detection, the YouTube video subset is sliced into five folds; each fold contains 20%

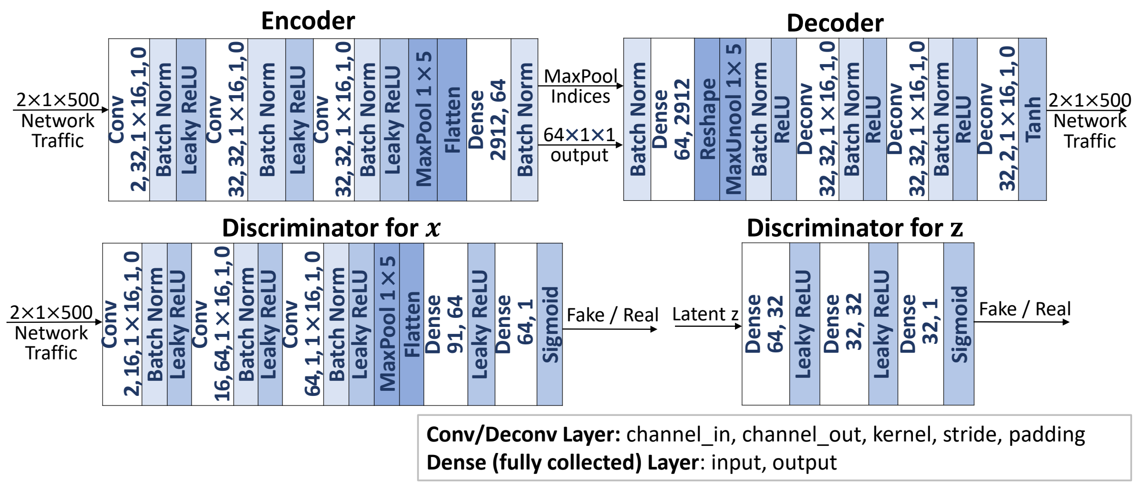


Figure 4.5 : CRAAE Module Architecture for Network Traffic Dataset.

randomly selected samples of each class. The training set, validation set, and test set occupy 3 folds, 1 fold, and 1 fold, respectively. In addition, I select the different number of classes in order, such as the first 1 class, first 3 classes and first 5 classes from the training set as inlier classes to train my proposed model. Then the remaining classes are used as outliers. For inter dataset novelty detection, I treat YouTube video subset as the inliers and other three subsets as the outliers. I use 80% of the YouTube video subset as the training set and 10% each as the test set and the validation set.

Since this dataset is different in configuration compared to the image datasets, I propose network traffic compatible modules in Figure 4.5 based on the architecture prototype in [94] and the neural network parameter setting in [69].

Image Datasets

I use four image datasets in the evaluation. MNIST ([66]) contains 60,000 training and 10,000 test handwritten digits from 0 to 9 with size 28×28. CIFAR10 ([64]) contains 50,000 training and 10,000 test images of size 32×32 with 10 classes. I use the test set of LSUN ([135]), which consists of 10,000 images of 10 scenes, as the outliers in far distance outlier detection. For TinyImageNet ([27]), I only use its test set, which consists of 10,000 images with 200 classes, again as the outliers in far distance outlier detection. All images stated here are resized to a size of 32×32.

I use the MNIST dataset for intra dataset novelty detection. The split ratio for the training, validation and test subset and the selection method for outliers and inliers are the same as the network traffic dataset. For inter dataset novelty detection, I treat CIFAR10 as the inliers and other datasets as the outliers. In order to ensure fair comparison under similar experimental conditions, the network architecture for image datasets follows the protocol stated in [94]

To demonstrate the performance of my CRAAE model on imbalanced data, I construct multiple test and validation sets varying the ratio of outliers to inliers from 10% to 50% by randomly down-sampling or up-sampling outliers following the method in [94].

In my experiments, the activation function is LeakyReLU with negative slope as 0.2, and the optimizer is Adam. The learning rate, learning rate decay, batch size, and training epoch are 0.0001, 0.25 every 30 epochs, 64, and 100 for network traffic datasets; 0.002, 0.25 every 30 epochs, 128, and 80 for image datasets; The latent size for the network traffic dataset, MNIST dataset, and other image datasets is 64, 16 and 32, respectively (see Table 7.2 in Appendix 7.1).

4.1.3 Results

This section presents the experimental results for different outlier detection scenarios (i.e., intra dataset and inter dataset novelty detection).

Intra Dataset Novelty Detection

I compare the performance of my CRAAE model with GPND ([94]) in intra dataset novelty detection. GPND provides F1 score results on a dataset using only

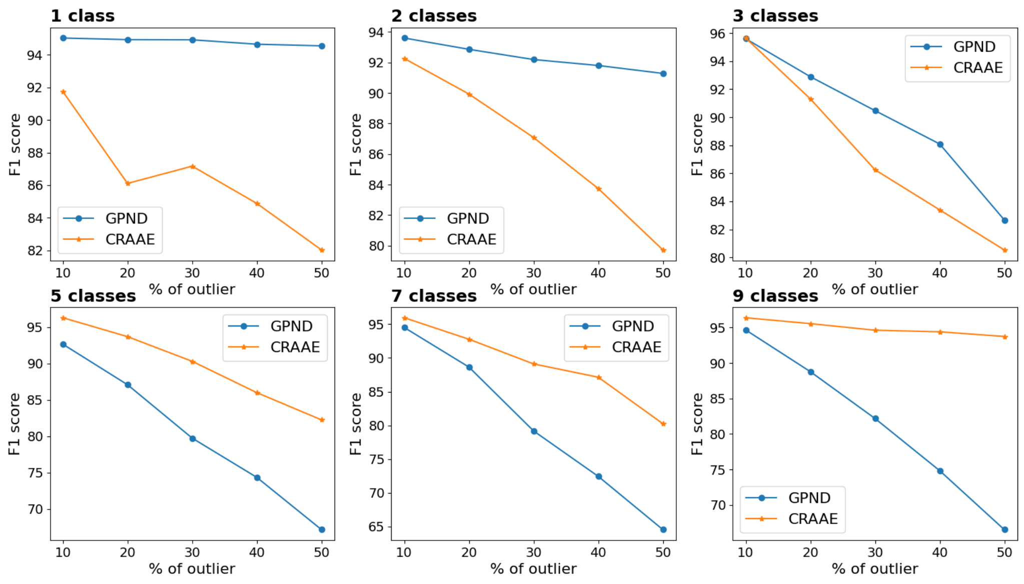


Figure 4.6 : F1 Score on the YouTube Video Traffic Dataset.

one of the ten classes as the inlier class. I extend the experiments to include more inlier classes. As illustrated in Figure 4.1, the variety and complexity of the inliers (i.e., the various distances between inlier samples) may influence the detection performance, which is determined by the decision boundary. I propose to integrate the category information into the decision metric to build a more accurate decision boundary (red line). It is expected to perform better than other methods (blue and green lines). To evaluate this, I set six groups of inlier classes, that is, video/digit 0, video/digit 0 and 1, video/digit 0, 1 and 2, video/digit 0 to 4, video/digit 0 to 6 and video/digit 0 to 8. The remaining classes of each group are treated as outliers. About the module architecture (c.f. Figure 4.5), I design it based on the setting in GPND to fit the characteristics of the network traffic dataset.

To evaluate the generalization of my proposed CRAAE framework, I extend my experiments to image datasets. To ensure a fair comparison, I utilise the same module architecture as GPND for image datasets.

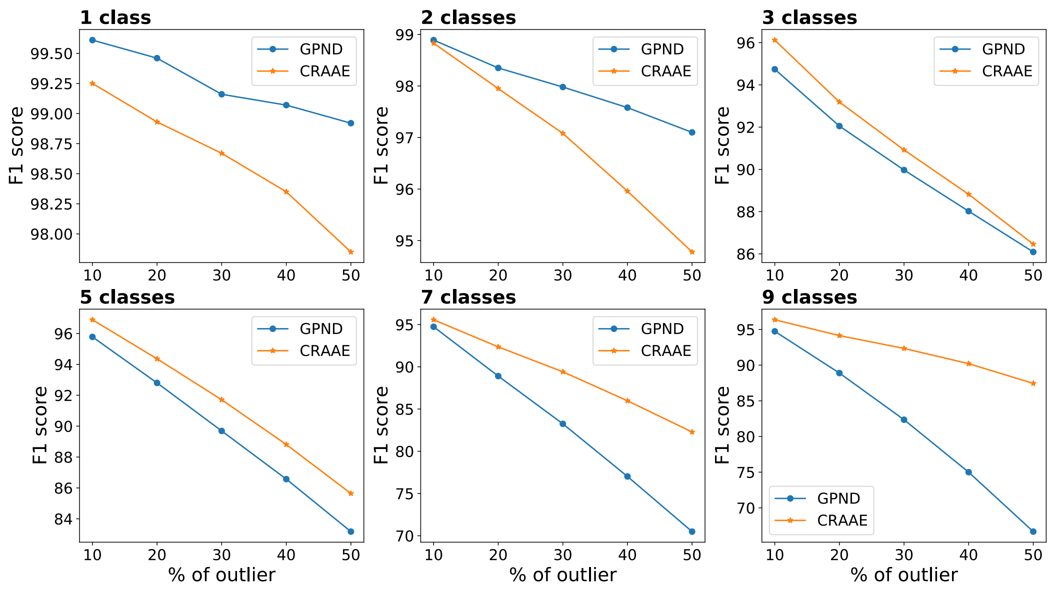


Figure 4.7 : F1 Score on the MNIST Dataset.

Table 4.1 : Performance of intra dataset novelty detection on YouTube video traffic dataset.

Inlier	AUROC ↑	FPR at 95% TPR ↓	Detection Error ↓	AUPR-In ↑	AUPR-Out ↑
Classes	GPND/CRAAE				
1	**96.80**/77.00	**3.14**/97.91	**3.67**/16.90	**98.93**/81.13	**81.42**/49.01
2	**94.41**/77.06	**65.37**/82.10	**7.77**/22.65	**97.67**/83.42	**80.52**/53.31
3	**93.49**/80.55	**47.62**/51.86	**11.10**/22.11	**97.31**/86.92	**73.02**/72.06
5	65.95/**87.68**	91.07/**37.86**	36.39/**19.15**	78.12/**93.26**	40.66/**72.26**
7	62.29/**88.41**	98.94/**37.18**	36.95/**20.81**	81.04/**94.17**	33.16/**79.06**
9	63.22/**96.79**	92.25/**15.90**	35.77/**4.76**	82.04/**98.84**	34.80/**77.88**

Figure 4.6 and 4.7 shows that my CRAAE framework performs at least 1% better than GPND when the number of the inlier classes is greater than two, and the performance advantage grows when the proportion of outliers increases. Further-

Table 4.2 : Performance of intra dataset novelty detection on MNIST dataset.

Inlier	AUROC ↑	FPR at 95% TPR ↓	Detection Error ↓	AUPR-In ↑	AUPR-Out ↑
Classes	GPND/CRAAE				
1	**99.92**/99.76	**0.06**/0.42	**1.01**/1.94	**99.96**/99.89	**99.70**/99.09
2	**99.50**/98.65	**1.47**/6.14	**2.85**/5.24	**99.78**/99.29	**98.23**/96.91
3	92.57/**93.50**	48.65/**33.82**	13.46/**13.43**	**96.84**/96.76	75.49/**83.03**
5	91.14/**91.91**	38.90/**28.59**	17.08/**15.12**	**95.57**/95.22	79.56/**84.08**
7	77.63/**88.24**	77.63/**42.21**	29.61/**18.59**	89.48/**92.93**	53.90/**74.72**
9	68.98/**93.18**	93.10/**27.10**	34.47/**12.22**	84.83/**95.94**	40.33/**83.68**

more, Table 4.1 and 4.2 shows that other performance measurements are consistent with the F1 score, which confirms that my proposed model outperforms GPND in intra dataset novelty detection with two or more inlier classes.

In addition, I compare my intra dataset novelty detection performance with the state of the art provided by One-Class novelty detection using GANs (OCGAN) ([92]). OCGAN previously reported the novelty detection performance with AUROC metric for each class of the MNIST dataset (one class as inliers and other classes as outliers), but it lacked experimental results for multiple inlier classes. Therefore, I compare the performance of CRAAE with OCGAN for the experimental setup where inliers include only class '0'. The AUROC of my CRAAE is 99.76% which is close to the 99.8% AUROC of OCGAN.

In summary, the results show that my CRAAE model outperforms GPND on various datasets in intra dataset novelty detection when the number of the inlier classes is greater than two or three. As shown in Figure 4.1, for a tighter boundary (OCND approaches like GPND), having more inlier classes tends to shape a more complex sample distribution that may lead to misclassification (i.e., classifying inliers as outliers). CRAAE builds an accurate decision boundary that matches the complex sample distribution and then reduces this misclassification. Therefore, the CRAAE performance gain in intra dataset novelty detection is more significant when the number of inlier classes is larger. For example, for 7 class inliers, CRAAE outperforms GPND by 26.21% for AUROC, 61.76% for FPR at 95% TPR, and 16.14% lower for detection error on the YouTube video traffic dataset. Similarly, the performance gain on the MNIST dataset is 10.61% AUROC, 35.42% FPR at 95% TPR, and 11.02% reduction in detection error.

Inter Dataset Novelty Detection

For inter dataset novelty detection, based on the CRAAE framework, I propose another approach to improve the decision boundary accuracy by adjusting the known outlier generator mechanism. I simulate category components by sampling from the Dirichlet distribution or directly using the discrete Uniform distribution. The concentration parameters of Dirichlet distribution are represented as a $1 \times class\_number$ vector; every entry is 1 except the entry corresponding to the ground truth label, which is 100. For example, if there are five classes, the concentration parameters of Dirichlet distribution for the third class is $[1, 1, 100, 1, 1]$.

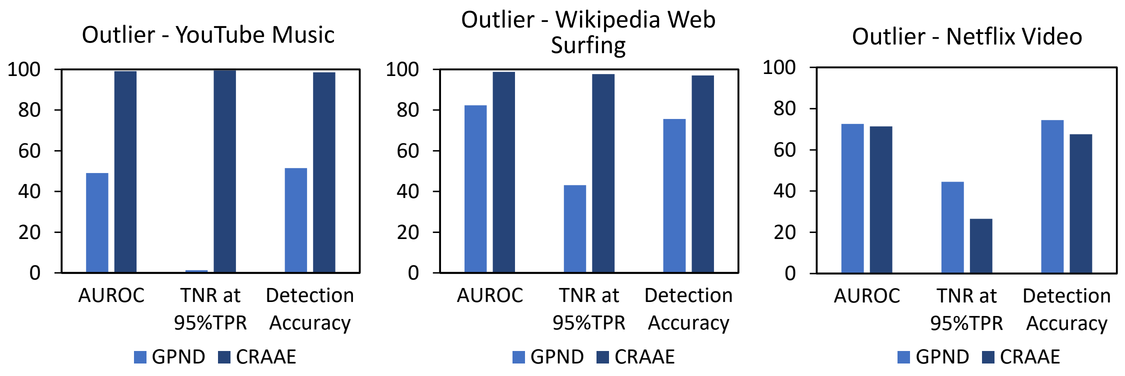


Figure 4.8 : Inter Dataset Novelty Detection (Inlier:YouTube Video).

I evaluate my CRAAE model on the network traffic dataset, treating YouTube videos as the inlier dataset. Other network traffic subsets, including YouTube music audio traffic, Wikipedia web surfing traffic, and Netflix video traffic, are utilised as outliers. According to the hierarchical characteristics of the network traffic datasets, network traffic differs in terms of HTTP type (e.g., video, audio, and web surfing), content platform (e.g., YouTube and Netflix), and content (e.g., different videos or songs). Therefore, I divide the far distance outlier detection scenario into finer scenarios. I assume that the distance between the YouTube video traffic inlier and the web surfing traffic outlier is further than between the YouTube video traffic inlier and the YouTube music (same platform) or Netflix video (same HTTP type) traffic outlier. The experiment evaluates CRAAE model performance in both the medium distance and far distance outlier detection scenarios. Instead of FPR at 95% TPR and detection error metrics, I measure TNR at 95% TPR and detection accuracy so that the performance comparison from the histograms can be made more clearly. I have conducted experiments under various conditions, such as with both explicit and implicit calibration, with only implicit calibration, and with Dirichlet distribution or uniform distribution category component. The performance obtained with uniform and Dirichlet distribution category components are similar. I present the best performance obtained with explicit calibration and the uniform distribution category component in Figure 4.8. My CRAAE model advantages are significant for both web surfing and YouTube audio traffic outliers (far and medium distance) when compared with GPND. For Netflix video traffic outliers (medium distance), both GPND and CRAAE models do not perform well, whereby their detection accuracy and AUROC are about 70% and TNR at 95% TPR is below 50%.

To verify the universality of my CRAAE model, I applied it on image datasets. Figure 4.9 shows the inter dataset novelty detection performance with CIFAR as the inlier dataset and TinyImageNet and LSUN as outlier datasets. The results in Figure 4.9 are obtained without explicit calibration and with the Dirichlet distribution category component. I compare the results with the state-of-the-art performance of

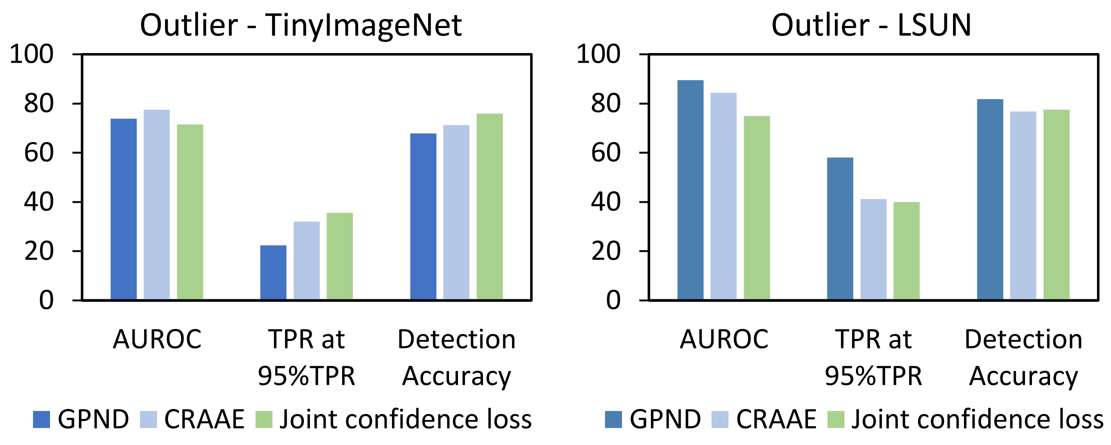


Figure 4.9 : Inter Dataset Novelty Detection (Inlier: CIFAR).

the GPND (an OCND approach) and joint confidence loss (an MCOSR approach) proposed in [67]. From Figure 4.9, I can see that CRAAE performs competitively with the other two methods. The performance of CRAAE is better than GPND on TinyImageNet and is better than joint confidence loss on LSUN. I note that my CRAAE model only requires 40% of the number of parameters used in the joint confidence loss model.

To summarize, as a unified model, CRAAE performs better than baseline GPND in location agnostic (far, close, and certain medium distance) outlier detection and is comparable to OCND and MCOSR methods in close and far distance outlier detection, respectively.

Ablation Study

My proposal includes three enhancements, namely implicit calibration, explicit calibration, and category component simulation. In *"Intra Dataset Novelty Detection"* and *"Inter Dataset Novelty Detection"*, I have demonstrated the combination effect of the proposed approaches on different novelty detection scenarios, respectively. I evaluate their contribution independently in this section. It is worth noting that the explicit calibration correlates to $\mathbf{E}$ and $\mathbf{D_z}$ modules, and the second and fifth items of the objective function; the category component simulation correlates to $\mathbf{G}$ and $\mathbf{D_x}$ modules, and the third and fourth items of the objective function; the implicit calibration correlates to $\mathbf{E}$, $\mathbf{G}$, $\mathbf{D_x}$ and $\mathbf{D_z}$ modules and the whole objective function. The modules in the proposed model framework are not isolated but rather interact with each other.

In Figure 4.10, I present the performance comparison through applying the three enhancements incrementally with the YouTube video subset as inliners and the other three network traffic subsets as outliers. I notice that the implicit calibration can

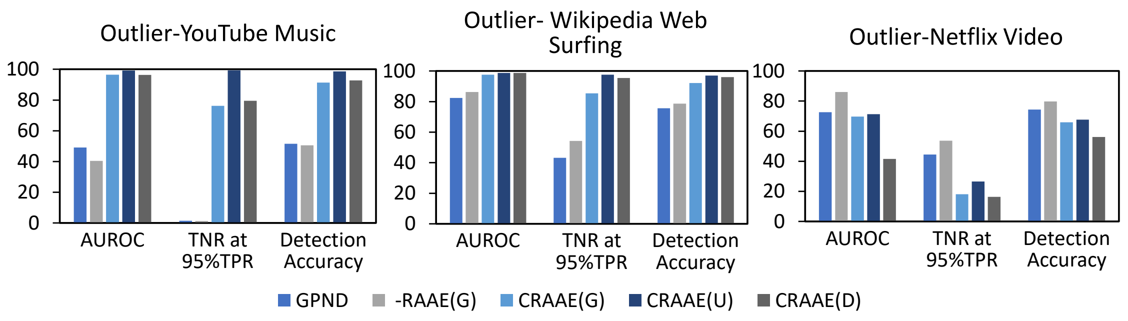


Figure 4.10 : Performance gained by adding implicit calibration (-RAAE), explicit calibration (CRAAE) and category component simulation (**G**aussian, **U**niform or **D**irichlet distribution) incrementally. (Inlier: YouTube Video).

obtain about 3% to 10% performance gain compared to GPND on various metrics in the far and medium distance (same HTTP type) scenarios. The explicit calibration provides 10% to 50% performance gain in far, medium distance (same platform) and close distance (demonstrated in Section 4.1.3) scenarios. In addition, the performance gain with uniform distribution category component is 1% to 3%, 8% to 23%, and 2% to 7% on AUROC, TNR, and detection accuracy metrics in different scenarios. Dirichlet distribution category component performs similar to the Uniform distribution in the far and medium distance (same platform) scenarios, but does not work on another medium distance (same HTTP type) scenario. In general, the combination of implicit, explicit calibration and the Uniform distribution category component simulation can achieve high novelty detection performance in most of the scenarios.

Furthermore, the proposed CRAAE performs well not only in novelty detection but also can achieve good multi-classification performance by leveraging the classification function of the encoder. I use a new metric "Multi-Accuracy (Multi-Acc)" for multi-classification with the unknown class. The Multi-Acc computes the percentage of correctly classified samples, including known and unknown data in the test process. If there are ten known classes, the unknown class will be labeled as the eleventh class. The experimental setup and process are similar to the novelty detection ablation study. The ratio of the outliers (the eleventh class) to inliers (the ten known classes) is set as $1 : 10$ to avoid data imbalance for multi-classification. According to Table 4.3, category component simulation improves the performance of the Multi-Acc metric by 1% to 5%. I also calculated the closed-set classification accuracy for the YouTube video dataset (without outliers) using Gaussian, Uniform, or Dirichlet distribution category components. The results are 96.01%, 97.27%, and 96.08%, respectively. The results approximate the 97.5% performance stated in [69], indicating that my proposed approaches can keep a high performance level on both novelty detection and multi-classification tasks.

4.1.4 Conclusion

In this section, I proposed CRAAE – a unified framework for accurate location agnostic outlier detection. To build a more accurate decision boundary, I proposed implicit and explicit calibration on the decision metric by introducing known outliers into the training process. I also proposed to leverage uniform and Dirichlet noise instead of Gaussian noise in generating known outliers. I evaluated my proposal on

Table 4.3 : Multi-Accuracy performance gained by adding implicit calibration (-RAAE), explicit calibration (CRAAE) and category component simulation (**G**aussian, **U**niform or **D**irichlet distribution) incrementally (Inlier: YouTube Video).

Outlier Dataset	Multi-Accuracy↑			
	-RAAE (G)	CRAAE (G)	CRAAE (U)	CRAAE (D)
YouTube Music	86.41	90.53	87.54	**91.76**
Web Surfing	90.07	91.65	88.12	**93.63**
Netflix Video	**88.63**	87.48	87.52	87.37

network traffic datasets and image datasets for different outlier location scenarios. The results demonstrate that CRAAE can achieve state-of-the-art performance for any outlier location scenarios as a unified model on both datasets. CRAAE can also achieve similar performance compared to OCND and MCOSR methods in close and far distance outlier detection, respectively. Furthermore, CRAAE can be used for multi-classification tasks and can achieve high performance.

4.2 Flexible Dirichlet Mixture Model (FDMM) Based Softmax Calibration

For the novelty detection topic, except for proposing the CRAAE model, I also attempt a Flexible Dirichlet Mixture Model (FDMM) based softmax calibration approach to improve novelty detection performance. The characteristic of softmax output vectors includes the sum of each vector entry equals one. The vector corresponding to a category usually has a peak value at the corresponding entry. Therefore, I assume the softmax output of multi-classification following a Dirichlet Mixture Model distribution. I attempt to fit the mixture distribution through the softmax output vectors of training samples. I then compute the likelihood of the softmax output for each test sample under the fitted distribution. I utilise this likelihood to calibrate the original softmax output to decrease the over-fitting confidence in deep learning models.

4.2.1 FDMM Introduction

The flexible Dirichlet mixture model [79] with K components can be written as:

$$f_{FD}\left(\boldsymbol{s} \mid \Theta_1, \ldots, \Theta_K\right) = \sum_{j=1}^{K} \boldsymbol{\pi}_j f_D\left(\boldsymbol{s} \mid \boldsymbol{\alpha_j}\right), \tag{4.11}$$

where $\Theta_j = \{\boldsymbol{\alpha_j}, \boldsymbol{\pi_j}, \tau\}$ is the group of parameters for component j, $\boldsymbol{\alpha}_j = \boldsymbol{\alpha} + \tau\mathbf{e}_j$ is the series of concentration parameter for jth component, where $\mathbf{e}_j$ is a canonical vector. Each entry of $\mathbf{e}_j$ is 0 except for the jth one which is 1. $\boldsymbol{\pi_j}$ is the mixing proportion for jth component and it must be positive. The sum of each $\boldsymbol{\pi}$ equals one. $f_D(\boldsymbol{s} \mid \boldsymbol{\alpha})$ is expressed as:

$$f_D(\boldsymbol{s} \mid \boldsymbol{\alpha}) = \frac{\Gamma\left(\alpha^{+}\right)}{\prod_{h=1}^{K} \Gamma\left(\alpha_h\right)} \left(\prod_{h=1}^{K} s_h^{\alpha_h - 1}\right), \tag{4.12}$$

where $\alpha^{+} = \sum_{j=1}^{K} \alpha_j$.

$$\begin{aligned}\pi_j\left(\boldsymbol{s} \mid \boldsymbol{\Theta}\right) &= \frac{\pi_j f_D\left(\boldsymbol{s} \mid \boldsymbol{\alpha}_j\right)}{\sum_{h=1}^{K} \pi_h f_D\left(\boldsymbol{s} \mid \boldsymbol{\alpha}_h\right)} \\ &= \pi_j \frac{\frac{\Gamma(\alpha_j)}{\Gamma(\alpha_j+\tau)} s_j^{\tau}}{\sum_{h=1}^{K} \pi_h \frac{\Gamma(\alpha_h)}{\Gamma(\alpha_h+\tau)} s_h^{\tau}}, j = 1, \ldots, K.\end{aligned} \tag{4.13}$$

Given n independent samples $\boldsymbol{s}_i, \quad i = 1, \ldots, N$, the completed sample vector $\boldsymbol{S_c}$ is given by:

$$\boldsymbol{S_c} = (\boldsymbol{S}, \boldsymbol{Y}) = \left(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\right), \tag{4.14}$$

where the K-dimensional label vector $\boldsymbol{y}_i = \left(y_{i1}, \ldots, y_{iK}\right)(i = 1, \ldots, N)$ denotes the missing data. When the ith sample is from the jth component of the mixture model, y_{ij} equals to 1 and the other entries of $\boldsymbol{y}_i$ equal to 0. The complete data log-likelihood can be written as:

$$\log L_c(\boldsymbol{\Theta}) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \left\{ \log \pi_j + \log f_D \left(\boldsymbol{s}_i \mid \boldsymbol{\alpha}_j \right) \right\}. \tag{4.15}$$

Here I use the EM algorithm to estimate parameters of a flexible Dirichlet mixture model [79]. I introduce the EM algorithm as follows.

E-step: I have the observed samples and estimated parameters for the current step k, $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)\boldsymbol{\Theta}^{(k)} = \left(\boldsymbol{\alpha}^{(k)}, \boldsymbol{\pi}^{(k)}, \tau^{(k)} \right)$. Then I can calculate the conditional expectation of the complete-data log-likelihood

$$\begin{aligned} \log L_c(\boldsymbol{\Theta}^{(k)}) = \textstyle\sum_{j=1}^{K} \sum_{i=1}^{N} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right) \\ \times \left\{ \log \pi_j^{(k)} + \log f_D \left(\boldsymbol{s}_i \mid \boldsymbol{\alpha}^{(k)} + \tau^{(k)} \boldsymbol{e}_j \right) \right\}, \end{aligned} \tag{4.16}$$

where $\pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right)$ represents the likelihood that $\boldsymbol{s}_i$ belongs to the jth component of the mixture given $\boldsymbol{\Theta}^{(k)}$.

M-step:

$$\begin{aligned} \boldsymbol{\Theta}^{(k+1)} = \underset{\Theta}{\operatorname{argmax}} \textstyle\sum_{j=1}^{K} \sum_{i=1}^{N} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right) \\ \times \left\{ \log \pi_j + \log f_D \left(\boldsymbol{s}_i \mid \boldsymbol{\alpha} + \tau \boldsymbol{e}_j \right) \right\} \end{aligned}. \tag{4.17}$$

Maximize Equation 4.16 to obtain the maximum likelihood estimates of the parameters (see Equation 4.17). In particular, I have

$$\hat{\pi}_j^{(k+1)} = \frac{1}{n} \sum_{i=1}^{N} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right), (j = 1, \ldots, K-1), \tag{4.18}$$

Here, I apply a Newton- Raphson method [86] to calculate $\hat{\boldsymbol{\alpha}}^{(k+1)}$ and $\hat{\tau}^{(k+1)}$, see Equation 4.19, 4.20 and Equation 4.23, 4.24.

$$g_\alpha = \nabla \ell \left(\boldsymbol{\alpha} \mid \boldsymbol{S} \right) = \begin{pmatrix} \sum_{i=1}^{N} \sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right) \left(\Psi \left(\alpha_+ + \tau \right) - \Psi \left(\alpha_1 + \tau e_{j1} \right) + \log s_{i1} \right) \\ \vdots \\ \sum_{i=1}^{N} \sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)} \right) \left(\Psi \left(\alpha_+ + \tau \right) - \Psi \left(\alpha_K + \tau e_{jK} \right) + \log s_{iK} \right) \end{pmatrix}, \tag{4.19}$$

$$H_\alpha = \nabla^2 \ell\left(\boldsymbol{\alpha} \mid \boldsymbol{S}\right) = \begin{pmatrix} H_{\alpha_{1,1}} & \dots & H_{\alpha_{1,K}} \\ \vdots & \vdots & \vdots \\ H_{\alpha_{K,1}} & \dots & H_{\alpha_{K,K}} \end{pmatrix}, \tag{4.20}$$

for $h = 1, \dots, K$, I can deduce:

$$H_{\alpha_{h,h}} = \sum_{i=1}^{N}\sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)}\right)\left(\Psi'\left(\alpha_+ + \tau\right) - \Psi'\left(\alpha_h + \tau e_{jh}\right)\right), \tag{4.21}$$

$$H_{\alpha_{h,-h}} = \sum_{i=1}^{N}\sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)}\right)\Psi'\left(\alpha_+ + \tau\right), \tag{4.22}$$

$$g_\tau = \frac{\partial \ell\left(\tau \mid \boldsymbol{S}\right)}{\partial \tau} = \sum_{i=1}^{N}\sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)}\right)\left(\Psi\left(\alpha_+ + \tau\right) - \Psi\left(\alpha_j + \tau\right) + \log s_{ij}\right), \tag{4.23}$$

$$H_\tau = \frac{\partial^2 \ell\left(\tau \mid \boldsymbol{S}\right)}{\partial \tau^2} = \sum_{i=1}^{N}\sum_{j=1}^{K} \pi_j \left(\boldsymbol{s}_i \mid \boldsymbol{\Theta}^{(k)}\right)\left(\Psi'\left(\alpha_+ + \tau\right) - \Psi'\left(\alpha_j + \tau\right)\right), \tag{4.24}$$

where

$$\Psi(x) = \frac{\mathrm{d}\log\Gamma(x)}{\mathrm{d}x} \quad \text{and} \quad \Psi'(x) = \frac{\mathrm{d}\Psi(x)}{\mathrm{d}x}.$$

Then I can obtain

$$\hat{\boldsymbol{\alpha}}^{(k+1)} = \boldsymbol{\alpha}^{(k)} - H_\alpha^{-1} g_\alpha, \tag{4.25}$$

$$\hat{\tau}^{(k+1)} = \tau^{(k)} - g_\tau / H_\tau. \tag{4.26}$$

For component parameters, I set a joint prior distribution G_0 and introduce indicator variables $z_i, i = 1, \dots, n$, then the sample generation process can be written as:

$$\boldsymbol{s}_i \mid z_i, \Theta \sim f_D\left(\boldsymbol{\alpha}_{\boldsymbol{z_i}}\right)$$

$$z_i \mid \boldsymbol{\pi} \sim \text{ Mult}\left(\pi_1, \dots, \pi_K\right)$$

$$\boldsymbol{\pi} \mid \beta \sim \mathrm{Dir}(\beta/K, \ldots, \beta/K)$$

$\boldsymbol{\alpha_j} \sim G_i$ drawn from a Dirichilet Process, $G_i \sim DP\,(\gamma, G_0)$, where G0 is a Gamma distribution.

4.2.2 FDMM Initialization Algorithm

Algorithm 1 Pseudocode for FDMM Initialization

```
    Required: $\mathcal{C}$ = trained classification neural network model
    Required: $K$ = number of class
    Required: $\mathcal{D}_{mle}$ = maximum likelihood estimation function of Dirichlet distribution
    Required: $(\boldsymbol{X}, \boldsymbol{y}) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$, labeled known data
 1: procedure PRETRAIN($\boldsymbol{X}, \boldsymbol{y}$)
 2:     Softmax layer output $\boldsymbol{S} = \mathcal{C}\,(\boldsymbol{X})$
 3:     Gather $\boldsymbol{S}$ into $K$ groups $\{\boldsymbol{S_1}, \ldots, \boldsymbol{S_K}\}$ according to labels $\boldsymbol{y}$
 4:     for i = 1 to $K$ do
 5:         $\boldsymbol{\alpha}_i = \mathcal{D}_{mle}\,(\boldsymbol{S_i})$
 6:         $\pi_i^{(0)} = \mathbf{len}\,(\boldsymbol{S_i})\,/\mathbf{len}\,(\boldsymbol{S})$
 7:     end for
 8:     $\boldsymbol{\alpha}^{(0)} = \mathbf{mean}(\mathbf{sum}(\boldsymbol{\alpha}, \mathbf{axis} = 0) - \mathbf{diag}(\boldsymbol{\alpha}))$
 9:     $\tau^{(0)} = \mathbf{diag}(\boldsymbol{\alpha}) - \boldsymbol{\alpha}^{(0)}$
10:     Return $\boldsymbol{\alpha}^{(0)}, \tau^{(0)},$ and $\boldsymbol{\pi}^{(0)}$
```

4.2.3 FDMM Inference Algorithm

Algorithm 2 Pseudocode for FDMM EM procedure

```
Required: $\mathbf{\Theta}^{(0)} = \left(\boldsymbol{\alpha}^{(0)}, \tau, \boldsymbol{\pi}^{(0)}\right)$, initial parameters for this FDMM model from Pretrain procedure.
1:  procedure EM-FDMM($\boldsymbol{S}, \mathbf{\Theta}$)                ▷ Training fixed components FDMM
2:      Calculate $k = 0$ step $\log L_c(\mathbf{\Theta^{(0)}})$, using Eq. 4.16
3:      while $\log L_c(\mathbf{\Theta}^{(k)}) \geq \log L_c(\mathbf{\Theta}^{(k-1)})$ do
4:          Update mixing proportions $\hat{\boldsymbol{\pi}}^{(k+1)}$, using Eq. 4.18
5:          Calculate k+1 step $\log L_c(\mathbf{\Theta^{(k+1)}})$, using Eq. 4.16
6:          Compute $g_\tau$ using Eq. 4.23                          ▷ First derivative
7:          Compute $H_\tau$ using Eq. 4.24                         ▷ Second derivative
8:          Update $\hat{\tau}^{(k+1)}$ using Eq. 4.26
9:          for i = 1 to $K$ do
10:             Compute $g_{\alpha_i}$ using Eq. 4.19              ▷ $i$th element of gradient
11:             Compute $H_{\alpha_{i,i}}$ using Eq. 4.21          ▷ Diagonal elements of $H_\alpha$
12:             Compute $H_{\alpha_{i,-i}}$ using Eq. 4.22         ▷ Other elements of $H_\alpha$
13:         end for
14:         Update $\hat{\boldsymbol{\alpha}}^{(k+1)}$ using Eq. 4.25
15:     Return: $\hat{\boldsymbol{\alpha}}, \hat{\tau},$ and $\hat{\boldsymbol{\pi}}$
```

4.2.4 Experimental Setup and Results

I treat softmax output as samples drew from Dirichlet distribution. The softmax output of the multi-class sample set should follow a Dirichlet Mixture Model distribution. I fit a group of synthetic samples to a fixed component DMM.

The DMM setting is inspired by FDMM, where I can use fewer parameters to describe the mixture distribution. For normal DMM, each component has a set of parameters $\Theta_j = \{\boldsymbol{\alpha_j}, \boldsymbol{\pi_j}\}$. For FDMM, the parameters for component j is $\Theta_j = \{\boldsymbol{\alpha_j}, \boldsymbol{\pi_j}, \tau\}$. $\boldsymbol{\alpha}_j = \boldsymbol{\alpha} + \tau \mathbf{e}_j$, is the series of concentration parameter for jth component. Based on the characteristic of softmax output, it is intuitive to only use the same $\boldsymbol{\alpha}$ for each component and use the $\boldsymbol{\tau}$ to adjust the softmax peak value position according to its corresponding class.

I use the parameter inference EM algorithm stated in Section 4.2.3 to infer the parameters of FDMM. Figure 4.11 shows the inference performance based on this algorithm simply. The inference for parameter π is very accurate. For this reason, this information is not illustrated in this Figure. It is clear that my algorithm works well on a synthetic dataset. For different α and τ and initialization parameters, the training convergence speed is different. When $\boldsymbol{\alpha} = (5, 7, 9)$ and $\boldsymbol{\tau} = 50$, the training process can quickly achieve convergence in about 20 epochs. However, the training process is highly dependent on the choice of initialization parameters. If initialization parameters are not suitable, the training process can not achieve convergence. Therefore, I propose an initialization algorithm stated in Section 4.2.2 to improve the model inference.

Based on this EM algorithm model, I then simulate the novelty detection process. I simulate inlier softmax output through Dirichlet distribution sampling. For example, $\boldsymbol{\alpha} = (5, 5, 5, 5, 5), \boldsymbol{\tau} = 50, \boldsymbol{\pi} = (0.2, 0.2, 0.2, 0.2, 0.2)$. For class ‘0’, its $\boldsymbol{\alpha_j = (55, 5, 5, 5, 5)}$. I create 100 samples for each class and split them into training and test set with a ratio $8 : 2$. I also slice 10 samples of training set for pre-training the initialization parameters. Other training samples are used to inference the parameters $\boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{\pi}$.

For outlier test samples, I still simulate them through Dirichlet distribution sampling with $\boldsymbol{\alpha} = (1, 1, 1, 1, 1), \boldsymbol{\tau} = 0, \boldsymbol{\pi} = (0.2, 0.2, 0.2, 0.2, 0.2)$. The sample number of the outlier is set the same as the total test sample number, that is, 100

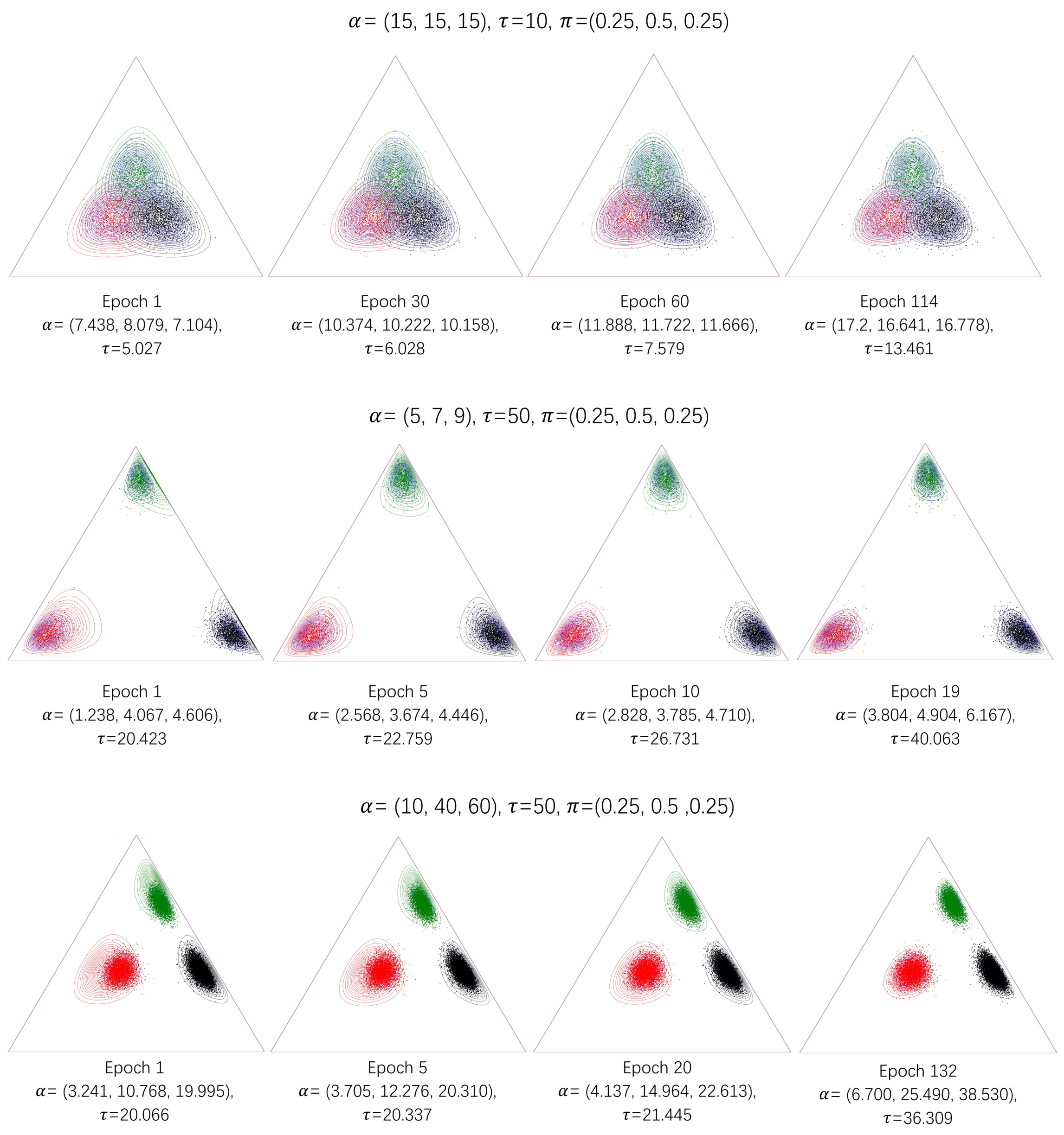


Figure 4.11 : Dirichlet Mixture Model EM Inference Performance on Synthetic Dataset.

samples, according to the above example setting. The outlier and inlier test samples are concatenated to test the open set recognition performance of my model. After the parameter inference process, I calculate the likelihood of a test sample belonging to this DMM distribution and leverage it as the calibration factor of the softmax output.

My calibrated softmax threshold (based on DMM) novelty detection experiment setting is as below.

$\boldsymbol{\alpha_{in}}$ is a vector with same entry value. In my experiment, this entry value is 5. τ is 50. The component number is 10, and the mixing proportion π for each component is the same. There are 100 synthetic samples for each inlier class. The proportion of samples for pre-training, training, and test is 10:70:20. The amount of outlier test samples is the same as the inlier test samples. The entry value of $\boldsymbol{\alpha_{out}}$ is 1.0.

First, I fix the above settings, only changing the component number. The result is listed in Table 4.4. The performance of both approaches is nearly the same even though it looks strange because classifiers usually perform better for fewer classes. It is reasonable in my experiment setting because the softmax output is generated perfectly and independent of the number of inlier classes.

Then I fix the component number as 10 and kept other settings except for τ, see the result in Table 4.4. I note that my approach can outperform the softmax threshold method slightly when τ is equal to or smaller than 30. When the original softmax output is more distinguishable, my approach impacts the result less. It is worth mentioning that my approach does not reduce the performance of the original approach. My approach keeps the same performance as the original approach in the worst condition. For this reason, my approach can be applied to any exiting softmax output based novelty detection and can be used in conjunction with other novelty detection strategies.

Table 4.4 : Performance of DMM probability calibrated softmax threshold novelty detection on synthetic dataset.

τ	AUROC ↑	FPR at 95% TPR ↓	Detection Error ↓	AUPR-In ↑	AUPR- Out ↑
(Inlier Classes = 50)	Softmax/Softmax+DMM				
10	**98.96**/98.96	**8.5**/8.5	**3.5**/3.5	**98.29**/98.28	**99.25**/99.25
20	**99.46**/99.46	4.75/**3.75**	**2.25**/2.25	**99.34**/99.34	**99.58**/99.58
30	**99.67**/99.67	**3.67**/3.67	**1.92**/1.92	**99.64**/99.64	**99.72**/99.72
τ	AUROC ↑	FPR at 95% TPR ↓	Detection Error ↓	AUPR-In ↑	AUPR- Out ↑
(Inlier classes = 10)	Softmax/Softmax+DMM				
50	**98.96**/98.96	**8.5**/8.5	**3.5**/3.5	**98.29**/98.28	**99.25**/99.25
40	**97.05**/97.05	**19.5**/19.5	**6.75**/6.75	**95.29**/95.28	**97.79**/97.79
30	90.95/**90.98**	42.0/**41.0**	14.5/**14.5**	**86.02**/85.99	93.19/**93.23**
20	73.14/**73.9**	82.5/**78.5**	30.25/**28.75**	64.21/**64.54**	77.69**79.19**
10	32.36/ **39.79**	**99.5**/1.0	50.0/**49.75**	38.08/**40.89**	38.81**44.28**

4.2.5 Conclusion

To the best of my knowledge, my proposed FDMM based softmax calibration approach is the first one that treats the softmax output of multi-classification as a Dirichlet Mixture Model distribution. This idea is novel, and its novelty detection performance can be better than the basic approach based on the softmax threshold when τ is equal to or smaller than 30. However, the performance improvement is not significant. Therefore, I did not implement more experiments for this approach.

4.3 Summary

In this chapter, to answer the research question, "How to handle classification for previously unseen classes in the open world?" I conduct my work from two aspects, model innovation and algorithm innovation. I propose a new model framework

CRAAE and a novel FDMM-based novelty detection method. CRAAE, as a unified model for different outlier location scenarios, can achieve state-of-the-art performance. Moreover, CRAAE can maintain good multi-classification performance by leveraging its encoder as the classifier. In addition, I also explore novelty detection techniques through the FDMM-based method, which for the first time proposes to assume the softmax output of multi-classification as a Dirichlet Mixture Model distribution. It can outperform the basic approach based on the softmax threshold, though its performance improvement is limited compared to the improvement gained through the CRAAE model.

In summary, driven by addressing the application research question for encrypted network traffic identification in the open world, I have made contributions to the field of novelty detection through an effective model framework and the introduction of a novel idea for further exploration.

Chapter 5

Few-Shot Learning for Encrypted Network Traffic in the Open World

In Chapter 4, I explored new methods for novelty detection to address the classification of previously unseen classes in the open world. However, the best-performing model is only trained for a specific classification task (e.g., 10 videos or 10 digits). The trained model does not generalize when the identification task changes (e.g., to another 10 videos or 10 characters). It is necessary to collect new samples for these new tasks to train the model again, which is time-consuming. To this end, in this chapter, I address this research question "How to tackle dynamically changing classification tasks with few samples in the open world?"

Few-shot learning is a promising method to adapt an already trained model to new classification tasks using only a limited, in some extreme cases, only one sample from each new class. Major few-shot learning approaches include distance metric based methods, model architecture based methods, optimization based, and data augmentation based methods. Though few-shot learning recently has been applied to network analysis, its application in fine-grained encrypted network traffic identification has yet to be explored much. In addition, classic few-shot learning models usually focus more on closed-world than open-world tasks. When open-world tasks (i.e., novelty detection or open-set recognition) and few-shot classification are considered together, distance metric based few-shot learning models can easily adapt open world by using similarity to classify a sample into a known class or identify it as an unknown class. Therefore, I select SNN, a classic and widely used distance metric based model, as the basic model architecture for my research for open-set few-shot learning. My subsequent experiments also validate that SNN outperforms other few-shot learning models.

To address the limitation of traditional novelty detection and few-shot learning models, I propose an SNN based framework that contains four key ideas in data augmentation, model architecture, decision metric, and model selection aspects.

i) To adapt dynamically changing classification tasks, more classes in the training set will improve the model's generalization ability. Hence, data augmentation is a promising way to improve model performance without time-consuming data collection. In addition, inspired by the previous work for novelty detection in Chapter 4, it is obvious that adding generated samples into the training process can achieve better novelty detection performance. However, the training samples of SNN are special. It is a sample pair consisting of two samples. If the two samples are from the same class, the sample pair is labeled as positive, otherwise labeled as negative (see Figure 5.1. Therefore, I propose a data augmentation method to obtain synthetic samples for both negative and positive sample pairing.

ii) In the test process of SNN, each query sample needs to pair with each sample of support set (classification task), and the sample pair is fed into the trained model to obtain a similarity score for classification. For open-set recognition, there should be a reasonable similarity score threshold to identify the query sample as an unknown class. Considering that the support set can be used to generate negative pairs, I utilise them to help the model understand the negative similarity distribution. Based on the exploration of the Dirichlet Process Mixture Model in Chapter 4, I use DPGMM in this chapter to represent the negative similarity distribution.

iii) Based on the feasibility evaluation of the CNNs model in Chapter 3, I used CNNs as the shared network of SNN. Utilizing the characteristic of CNNs that different layers of CNNs can produce features with different degrees of discrimination, I use the model to map the input to multiple feature spaces and obtain different similarity scores to work together.

iv) Compared to classic few-shot learning approaches, SNN constructs training batches without dependence on the number of support classes (N-way) of each query task. It is an advantage for training because training once is enough to query samples with any number of support classes. However, the characteristic brings the problem for testing about selecting the best model/models for different N-way one-shot validation. I combine multiple N-way validation results using an ensemble learning mechanism to choose the best model.

These four ideas are referred to as: bidirectional dropout data augmentation, DPGMM based task adaptive open-set recognition metric, hierarchical cross-entropy loss, and multi-model ensemble. Next, I will explain them in detail. My proposed model pipeline is shown in Figure 5.1. This figure explains the training, validation, and test process and the construction of the training pair and test batch. It would be more clear about how and where to apply my proposed ideas with this figure. Next, I will present the details of these ideas, including methodology, experimental setup, and results.

5.1 Methodology

5.1.1 Bidirectional Dropout Data Augmentation

Data augmentation increases the variety of training data which is critical for deep learning, especially for few-shot learning. Data augmentation usually expands the dataset in two ways; i) extend the class diversity by generating new items as novel classes [62, 139], ii) enrich the existing class samples by using the new generated items as extra samples of the same class [7, 128, 141]. However, to the best of my knowledge, existing data augmentation approaches, for example, deep learning

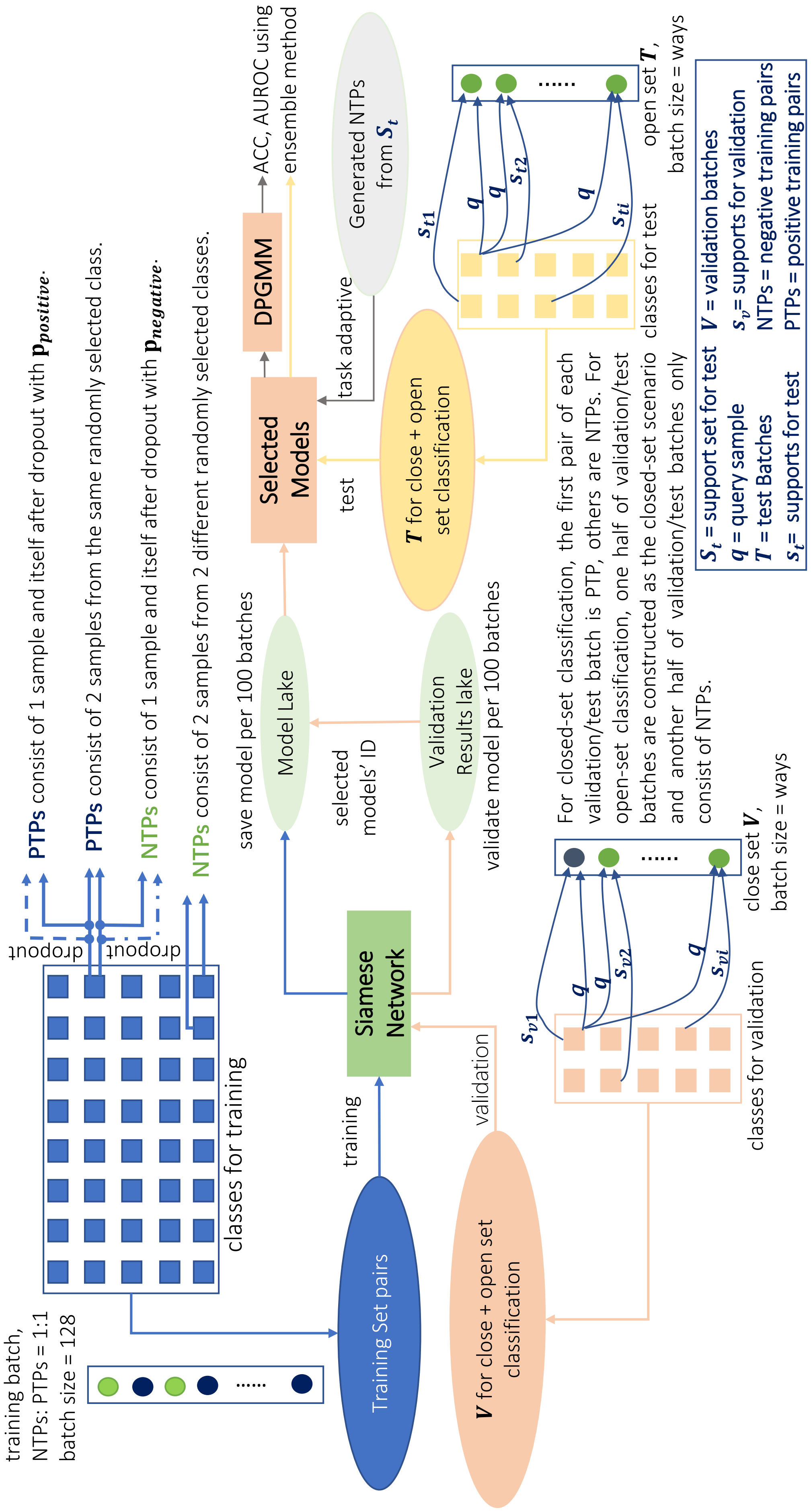

training batch,
NTPs: PTPs = 1:1
batch size = 128
......
classes for training
dropout
dropout
PTPs consist of 1 sample and itself after dropout with $\mathbf{p}_{positive}$.
PTPs consist of 2 samples from the same randomly selected class.
NTPs consist of 1 sample and itself after dropout with $\mathbf{p}_{negative}$.
NTPs consist of 2 samples from 2 different randomly selected classes.
Training Set pairs
training
Siamese Network
validation
save model per 100 batches
Model Lake
selected models' ID
Validation Results lake
validate model per 100 batches
Selected Models
DPGMM
ACC, AUROC using ensemble method
task adaptive
Generated NTPs from S_t
test
T for close + open set classification
V for close + open set classification
classes for validation
s_{v1}
q
q
s_{v2}
q
s_{vi}
......
close set V,
batch size = ways
classes for test
s_{t1}
q
q
s_{t2}
q
s_{ti}
......
open set T,
batch size = ways
For closed-set classification, the first pair of each validation/test batch is PTP, others are NTPs. For open-set classification, one half of validation/test batches are constructed as the closed-set scenario and another half of validation/test batches only consist of NTPs.
S_t = support set for test
q = query sample
T = test Batches
s_t = supports for test
V = validation batches
s_v = supports for validation
NTPs = negative training pairs
PTPs = positive training pairs

Figure 5.1 : Model Pipeline.

based methods (e.g., feature space augmentation and GAN based augmentation) as well as geometric and photometric transformation based approaches [113], do not directly provide a similarity metric to measure the similarity between the generated sample and the given original sample during data generation. For example, it is difficult to distinguish which is more similar to the original image between the top-to-bottom flipped image and the horizontally mirrored image. During the GAN-based data augmentation, the discriminator output can be utilised as a similarity metric. However, each generated sample has a unique similarity score, which cannot be controlled or set before sample generation. Hence these techniques are not directly applicable for generating both within-class and novel-class samples concurrently. In light of this, using dropout provides advantages over other data augmentation methods.

In deep learning, dropout refers to randomly ignoring units (e.g., nodes or pixels) during the training phase with a certain probability p. Dropout is often applied to the feature space of a neural network to prevent overfitting. In [63], dropout was used on the input space as a data augmentation method. Encrypted network traffic for each video is represented as temporal bins of a specific time interval and the total number of packets or other measurable features in each bin [69]. Therefore, dropout for network traffic is to randomly select some bins and set them as zero. If I treat the network traffic of a video as a time sequence signal, dropout for this network traffic can be regarded as random packet losses caused by channel interruptions. The dropout probability p controls the ratio of lost packets. The range of p is from 0.0 to 1.0, and the larger the value of p, the more packets will be lost. Setting $p = 0.5$ means 50% of information is discarded. Therefore, 0.5 is used as the threshold to distinguish whether a dropout sample is more similar to or different from the original. The profiles of the network traffic after dropout with different probabilities are shown in Figure 5.2, which also validates that a threshold of 0.5 is reasonable.

I notice that the essential characteristics (e.g., spike location) and shape pattern of the network traffic after dropout could be almost the same as the original one when p is small enough (below 0.5), but it also could be significantly different from the original one when p is large (over 0.5). That is, I can adjust the similarity of the generated profiles and their original ones by changing the dropout probability p. I set positive dropout probability p_{pos} and negative dropout probability p_{neg} to respectively generate within-class samples (i.e., positive sample) and samples of a novel class (i.e., negative sample).

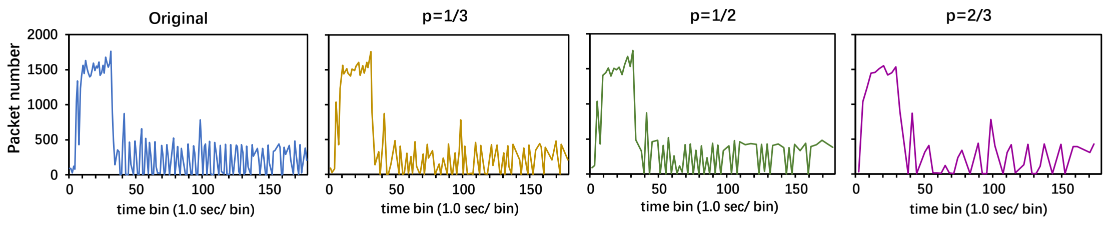


Figure 5.2 : Network Traffic Patterns Based on Different Dropout Probabilities p.

I tune hyperparameters p_{pos} and p_{neg} through a two-step traversing method. First, I only focus on tuning p_{pos} by varying p_{pos} from 0.1 to 0.4 (below the threshold 0.5) with a step interval of 0.1. In this situation, I set p_{neg} as 0.0, which means no dropout for negative sampling to avoid performance interference. Next, p_{pos} is fixed as the probability value that performs best, and I adjust p_{neg} from 0.5 to 0.9 (not small than the threshold 0.5) with a step interval of 0.1 to seek the best-performing p_{neg}.

5.1.2 DPGMM Based Task Adaptive OSR Metric

SNN measures the degree of similarity of two inputs (i.e., sample pair). The two inputs are fed into two identical neural networks (with the same architecture and weights). The neural networks map the two inputs into a feature space and form new representations. Then the distance between the new representations of the two inputs is calculated. Finally, the distance is fed to a fully collected layer to obtain a scalar, i.e., the similarity score.

After training based on the binary cross-entropy loss, the SNN's normalized similarity scores (applying the sigmoid function on similarity scores) of positive pairs (two inputs are from the same class, labelled '1') and negative pairs (two inputs are from different classes, labelled '0') should approach 1.0 and 0.0, respectively. Therefore, when not normalized, the similarity scores of positive and negative pairs can be viewed as clusters centered around the mean of the similarity scores, which are near large positive and negative values, respectively. Additionally, different tasks may yield varying similarity scores that correspond to different cluster centers. For example, the similarity score between two different handwritten digits '1' is likely to be higher than the similarity score between two different breeds of cats. I utilise the support set of a query task (only one sample for each support class in one-shot learning) to find the negative similarity distribution. If the maximum similarity score of a query task has a high likelihood under the negative similarity distribution, the query sample is prone to be identified as an unseen class.

For each query task, I construct negative pairs based on the N-way one-shot support set following the $\binom{N}{k}$[*]. Here, N denotes the number of support classes, k is set to 2. For example, 5-way and 10-way one-shot support sets can construct 10 pairs and 45 pairs, respectively. I also can integrate the dropout data augmentation method easily into the negative pair construction process. I create novel class samples according to the tuned negative dropout probability and leverage the generated and original samples of the support set to construct negative pairs.

I then feed these negative pairs into the trained SNN model to obtain similarity scores. The similarity scores of negative pairs can be grouped into many clusters

[*]The combination operator with k items selected from a collection of size N.

according to various dissimilarity levels. For example, the samples in the support set may come from not only diverse fine classes but also various coarse classes. Specifically, the similarity score whose negative pair consists of samples from the same coarse class (e.g., YouTube video traffic/cat) but different fine classes (e.g., (YouTube traffic of video 1 and video 2)/(Persian and Siamese cat)) should fall into a different cluster from the one whose negative pair consists of samples from different coarse classes (e.g., (YouTube and Netflix video traffic)/(cat and bird)). The similarity scores of a cluster are real-valued random variables distributed near a center (the mean of similarity scores of the cluster). Therefore, it is reasonable to fit the similarity scores of a cluster to a Gaussian distribution and to fit the similarity scores of several clusters to the Gaussian Mixture Model (GMM) distribution, which is widely used for clustering. For training the GMM, I need to preset the hyperparameter K, which represents the number of mixture components (clusters). However, the levels of dissimilarity would vary with the dynamically changing task, making it challenging to directly infer the number of clusters K. Therefore, I utilise a non-parametric Bayesian-based infinite mixture model DPGMM to identify the effective number of components automatically.

As a generative model, a sample point $\boldsymbol{x}$ can be obtained through the following steps. First, I represent the Dirichlet Process as below ([15, 43], the random measure G is distributed with concentration parameter α and base distribution H:

$$G \mid \alpha, H \sim DP(\alpha, H). \tag{5.1}$$

Then, I can draw mean vector $\boldsymbol{\mu_i}$ and covariance matrix Σ_i of Gaussian distribution for the ith sample point from G:

$$(\boldsymbol{\mu_i}, \Sigma_i) \mid G \sim G. \tag{5.2}$$

Finally, I generate a sample point by drawing them from a Gaussian distribution with parameters $\boldsymbol{\mu_i}$ and Σ_i:

$$\boldsymbol{x} \mid (\boldsymbol{\mu_i}, \Sigma_i) \sim \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu_i}, \Sigma_i\right). \tag{5.3}$$

Here, I utilise the similarity scores $\boldsymbol{s_{neg}}$ obtained from the constructed negative pairs of each new task to fit a DPGMM distribution using the variational inference method:

$$p\left(s \mid \Theta_1, \ldots, \Theta_K\right) = \sum_{j=1}^{K} \pi_j \mathcal{N}\left(s \mid \boldsymbol{\mu_j}, \Sigma_j\right), \tag{5.4}$$

where K is the number of fitted clusters, s denotes any similarity score, $\Theta_j = \{\pi_j, \boldsymbol{\mu_j}, \Sigma_j\}$ is the set of parameters for jth cluster. π_j is the mixing proportion. Then I compute the log-likelihood of each $\boldsymbol{s_{neg,l}}$ under the current fitted distribution, where $l \in \{1, \ldots, N_{neg}\}$, N_{neg} is the number of constructed negative pairs, and then calculate their mean μ_{LL} and standard deviation σ_{LL}:

$$\mu_{LL} = \frac{1}{N_{neg}} \sum_{l=1}^{N_{neg}} \log p(s = s_{neg,l} | \Theta_1, \ldots, \Theta_K), \tag{5.5}$$

$$\sigma_{LL}^2 = \frac{1}{N_{neg}} \sum_{l=1}^{N_{neg}} (\log p(s = s_{neg,l} | \Theta_1, \ldots, \Theta_K) - \mu_{LL})^2. \tag{5.6}$$

Next, I compute the log-likelihood for the maximum similarity score s_{Tmax} of a test pair batch $\boldsymbol{T}$ constructed by paring each query sample with the samples of the support set. Then I standardize this log-likelihood with μ_{LL} and σ_{LL} to reduce bias. Finally, I leverage the maximum similarity score and the standardization log-likelihood to build a new OSR metric:

$$s_{Tmax} - \beta[(\log p(s = s_{Tmax} | \Theta_1, \ldots, \Theta_K) - \mu_{LL})/\sigma_{LL}]. \tag{5.7}$$

Here, I select Z-score as the normalization method because it can handle outliers well compared with another common normalization method, Min-max scaling.

Min-max scaling can obtain data with the same scale but can not handle outliers well. The log-likelihood for s_{Tmax} can be out of the range of the log-likelihood for $s_{neg,l}$. Therefore, I select the Z-score method, which can handle outliers more effectively. I treat the second term of Equation 5.7 as a calibration term for balancing the confidence of the maximum similarity score. The hyperparameter β serves as the calibration factor and its optimal value is determined through experiments, as detailed in Section 5.3.2.

5.1.3 Hierarchical SNN

It is commonly known that different layers of CNNs can produce features with varying degrees of discrimination ([137]). For example, the extracted features of a facial image from shallow to deep layers can be the vertical edge, face outline, and other more detailed facial features, such as the eyes, nose, and mouth. Features extracted from coarse to fine level with different depths of CNNs have been used for hierarchical classification ([112]).

The traditional SNN model maps the inputs to a single feature space, where the distance between two inputs represents their similarity. Considering the characteristics of CNNs, I map the inputs to multiple feature spaces to obtain a range of similarity scores with varying degrees of discrimination. Then I utilise the coarse and fine level labels of each sample, along with the corresponding features (similarity scores), to construct a hierarchical cross entropy loss function, thereby achieving higher model robustness. In the encrypted network traffic dataset, video platforms and specific videos correspond to coarse and fine labels; similarly, alphabets and characters in the Omniglot dataset.

The difference in the Hierarchical SNN frame (see Figure 5.3) from the classical SNN is that there are multiple level features extracted from the network $\mathbf{G}(\boldsymbol{x})$. The same level features of $\boldsymbol{x1}$ and $\boldsymbol{x2}$ are used to compute the Manhattan distances of this level. Then the different level distances are fed to a fully collected layer to obtain the similarity scores, which are utilised to calculate the cross entropy with their corresponding level labels.

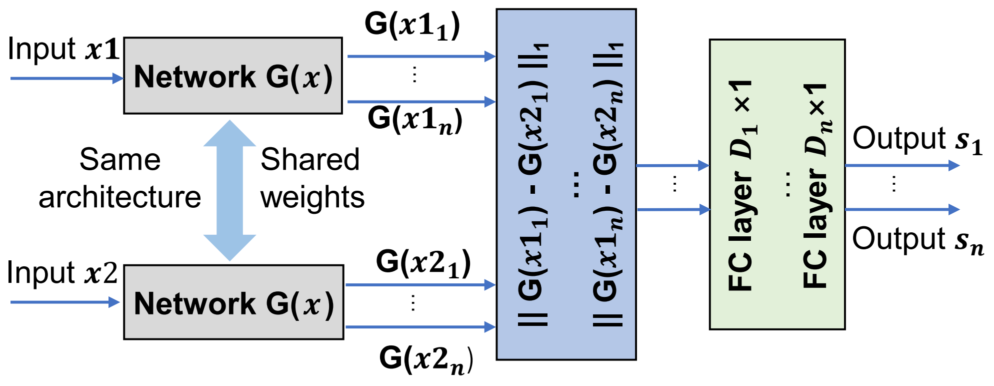


Figure 5.3 : Hierarchical Siamese Neural Network Framework.

The network architecture of $\mathbf{G}(\boldsymbol{x})$ is shown in Figure 5.4. This architecture is based on a four layer convolutional backbone. The filter settings of each convolutional layer and max pooling kernel size are designed for the encrypted network traffic dataset. Here, I only gave a specific setting for the last dense layer because I set other dense layers to have the same output dimension as the last one. The input dimension of these layers is decided by the output of their corresponding flatten layers. For example, the dense layer kernel for $\mathbf{G}(\boldsymbol{x_3})$ is 6528×4096. On the basis of this framework, I propose a hierarchical cross entropy loss function to maintain the consistency of loss functions on multiple levels:

$$\mathcal{L}_H = \mathbb{E}\left\{ \sum_n \gamma_n \left[y_n \log s_n + (1 - y_n) \log (1 - s_n) \right] \right\}, \tag{5.8}$$

where γ_n is the weighted factor for the nth level cross entropy loss, s_n is the nth similarity score output of hierarchical SNN (see Figure 5.3), y_n is the corresponding nth level label for a sample pair. I will tune hyperparameters γ_n to achieve high performance through experiment evaluation.

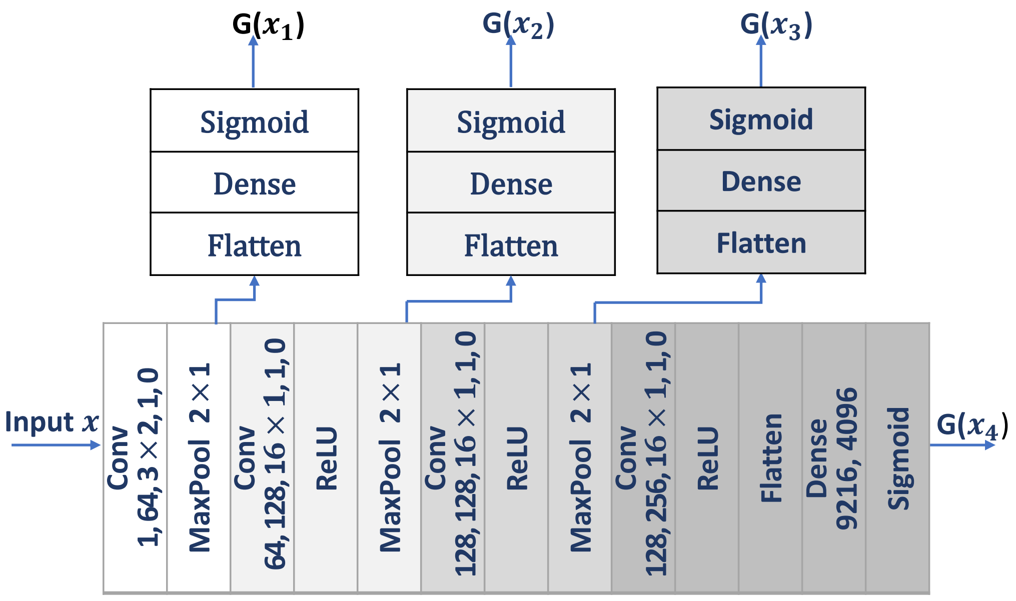


Figure 5.4 : Hierarchical Network Architecture.

5.1.4 Multi-model Ensemble

An advantage of an SNN compared to other classic few-shot learning approaches, including matching and prototypical models, is that the construction of training batches is independent of the query task. This advantage implies I only need to train once to query samples with any number of support classes. Therefore, the problem for testing is how to select the best model/models through validation.

In [62], the authors utilised validation error as the model selection indicator and selected the best-performing model through a 20-way one-shot validation set. The selected model is then applied only on the 20-way one-shot test set. I notice that the selected model is not always the same based on different N-way one-shot validation sets. Therefore, I propose a multi-model ensemble strategy to improve the model selection robustness, thus enhancing the classification accuracy.

The multi-model ensemble strategy includes two parts. First, I select the best-performing models through validation. At inference time, I then apply these models for ensemble classification and open-set recognition.

I use both the best accuracy and the best average accuracy of five successive iterations as the model selection metric. Considering storage space and the number of candidate models, I save the training model every 100 iterations and evaluate these models with different N-way one-shot support sets, such as 3-way, 5-way, and 10-way. I record the best accuracy and the best average accuracy for each kind of support set. To ensure the selected models are more likely to converge, I select the models with the best accuracy or the best average accuracy at the end of the training process. For instance, the best accuracy and the best average accuracy of 3-way, 5-way, and 10-way query tasks occurs respectively at the iterations 5000 and 6000 to 6400, 8000 and 7800 to 8200, 8800 and 9900 to 10300, then I select the models from the five iterations 9900 to 10300 as the best-performing models. If the best accuracy occurs later than the best average accuracy, I only select one model.

For ensemble classification and open-set recognition, I apply different methods. I utilise voting to classify each sample to the class with the highest share of votes. For open-set recognition, I utilise the mean of the open-set recognition decision metric across all best-performing models to identify whether a sample belongs to an unseen class.

5.2 Experimental Setup

5.2.1 Datasets and Evaluation Protocol

To demonstrate my contributions in both network analysis and computer vision domain, I use the collected network traffic dataset and the Omniglot dataset. The Omniglot dataset is a widely used public image dataset in the few-shot learning field. It is usually used as the benchmark dataset in few-shot learning to compare the performance of different models, like MNIST/CIFAR/ImageNet datasets in multi-classification.

For the encrypted network traffic dataset (named as few-shot learning dataset in Section 3.1.1), we collected 60 classes of video traffic from three different video platforms (YouTube, Netflix, and Stan, 20 classes per platform). I randomly split 60 classes into training, validation, and test sets with a ratio of 4:1:1, which is just enough to construct 10-way 1-shot validation/test while maximising the training diversity. Each class has 100 samples (i.e., 100 runs for each video). I represent each video streaming as a vector of $500 \times n$, where n is the number of the features, and 500 is the number of bins over the 3 minutes video streaming length. Based on the feature engineering result shown in Figure 3.2 and the experimental verification shown in Appendix (Table 7.4), I only select the top two important features to boost computational efficiency and save computer memory capacity. The features used are the number of non-data frames on the down-link (F3 (D)) and the number of data frames on both the up-link and down-link (F1 (C)).

The Omniglot dataset [65] contains 1,623 different handwritten characters from 50 different alphabets. Each character has 20 samples with a size 105×105 drawn by 20 different people. The Omniglot dataset is split into a background set and an evaluation set. The background set consists of 30 alphabets, and the evaluation set consists of 20 alphabets. I use the background set to construct a training set and divide the evaluation set evenly into validation and test set (i.e., each set has 10 alphabets).

I use accuracy as the evaluation metric for closed-set classification and the Area Under the Receiver Operating Characteristic Curve (AUROC) for OSR, where the ROC curve is a graph with the true positive rate (TPR) as the Y-axis and the false positive rate (FPR) as the X-axis across different decision thresholds. For binary classification (OSR) models, the accuracy metric depends on the threshold selection, but AUROC shows its ability to discriminate between inliers (positive samples) and outliers (negative examples) from a more general perspective without the need of selecting a specific threshold.

5.2.2 Implementation Details

In my experiments, I select the ratio of positive and negative training pairs as 1:1 for data balance and choose Adam as the model optimizer based on the experimental results shown in Table 7.4. According to the original SNN setting in [62], the training pair batch is 128, and the learning rate is selected from 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}. The model performs similarly with learning rates 10^{-3} and 10^{-4}. Subsequently, I fine-tune the learning rate from 2×10^{-4} to 9×10^{-4} and select the best-performing one, 6×10^{-4}. In addition, the number of training iterations in [62] for the Omniglot is about $46k$ (training samples ($30k$) divide the batch size (128) and times the epoch (200)). I round the iterations to $50k$ for the Omniglot dataset. For the encrypted network traffic dataset, I traverse the iterations from $10k$ to $50k$ with the interval of $10k$ and select the best-performing iterations of $20k$. I construct 1200/1200 query tasks for the test/validation set, which consist of samples of known and unseen classes with a ratio of 1:1. The network architecture used for the encrypted network traffic dataset is shown in Figure 5.4. For the Omniglot dataset, I also use the framework of Figure 5.4, but the network architecture details, such as the convolutional backbone, filter settings, the max pooling kernel size, activation functions, and the dense layer setting, are stated in [62]. It is worth mentioning that the different dense layer kernels in Figure 5.4 have the same output dimension (4,096) and the input dimension depends on the output of their previous flatten layer.

5.3 Results

In this section, I first present the performance comparison among several few-shot learning approaches to verify that SNN can handle both closed-set classification and OSR tasks on the encrypted network traffic dataset. Next, I introduce the hyperparameter selection process of my proposed model. Finally, I present the ablation study on both the encrypted network traffic dataset and the Omniglot dataset to evaluate the contribution of each proposed approach.

5.3.1 Performance Comparison Among Different Models

Table 5.1 presents the experiment results of SNN, MAML, and PEELER in terms of accuracy and AUROC for 3-way, 5-way, and 10-way one-shot query tasks. For MAML and PEELER, I use the same settings as [6] (4 layer convolutional backbone)[†] and [72] (ResNet-10 backbone)[‡] except for a few minor modifications according to the input data dimension. The MAML paper did not tackle the open-set problem, thus I only list its accuracy results. It is significant that SNN outperforms PEELER and MAML, especially in 10-way one-shot query tasks. Therefore, I use the SNN model as the baseline.

Table 5.1 : Performance comparison among different models on the encrypted network traffic dataset.

N way	SNN	MAML[†]	PEELER[‡]
1 shot	Accuracy/AUROC		
3	**94.7/93.6**	92.1/-	92.4/89.1
5	**90.7/90.7**	87.9/-	89.8/85.4
10	**82.7/87.0**	65.7/-	70.2/77.1

[†]`https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch`

[‡]`https://github.com/BoLiu-SVCL/meta-open`

5.3.2 Hyperparameter Selection

The hierarchical SNN model consists of $\mathbf{G}(\boldsymbol{x_3})$ and $\mathbf{G}(\boldsymbol{x_4})$ to construct a two level cross entropy loss. I set the fine loss factor as 1.0 for $\mathbf{G}(\boldsymbol{x_4})$ based on early experiments. Also, as can be seen from Table 5.1, fine cross entropy loss can operate independently to achieve good performance. Therefore, I retain this to dominate the total cross entropy, and only fine tune the coarse loss factor. I present performance results with coarse loss factors for $\mathbf{G}(\boldsymbol{x_3})$ from 0.0 (original SNN) to 1.0 in Figure 5.5. I select 0.3 as the coarse loss factor for subsequent experiments considering both classification and OSR performance.

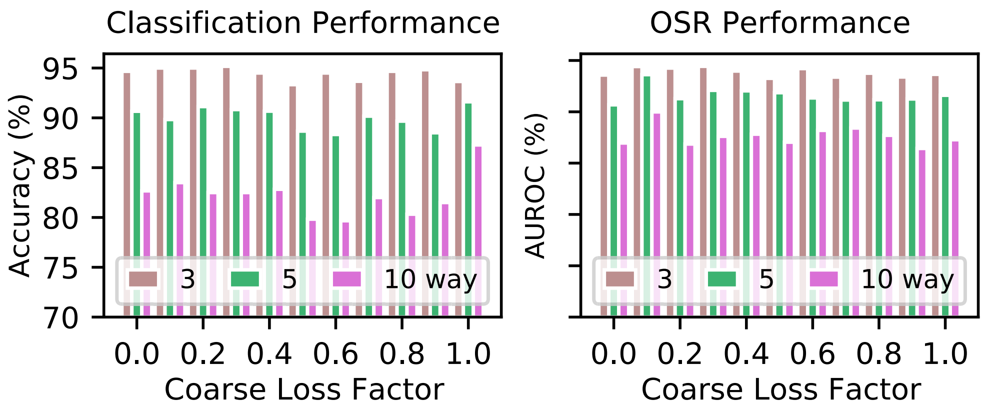


Figure 5.5 : Coarse Loss Factor Selection.

For Bidirectional Dropout Data Augmentation, I assume 0.5 as the separation point between positive and negative dropout probability (p_{pos} and p_{neg}) based on Figure 5.2. I first add samples for positive pairs generated with the p_{pos} from 0.1 to 0.4 into training phase. Figure 5.6 (left) shows that the p_{pos} 0.4 performs best. Then I fixed the p_{pos} and apply the p_{neg} from 0.5 to 0.9 for data augmentation (see Figure 5.6 (right)). Finally, I set the p_{pos} and p_{neg} as 0.4 and 0.6 for subsequent experiments.

For DPGMM based task adaptive metric, I used a normalized log-likelihood

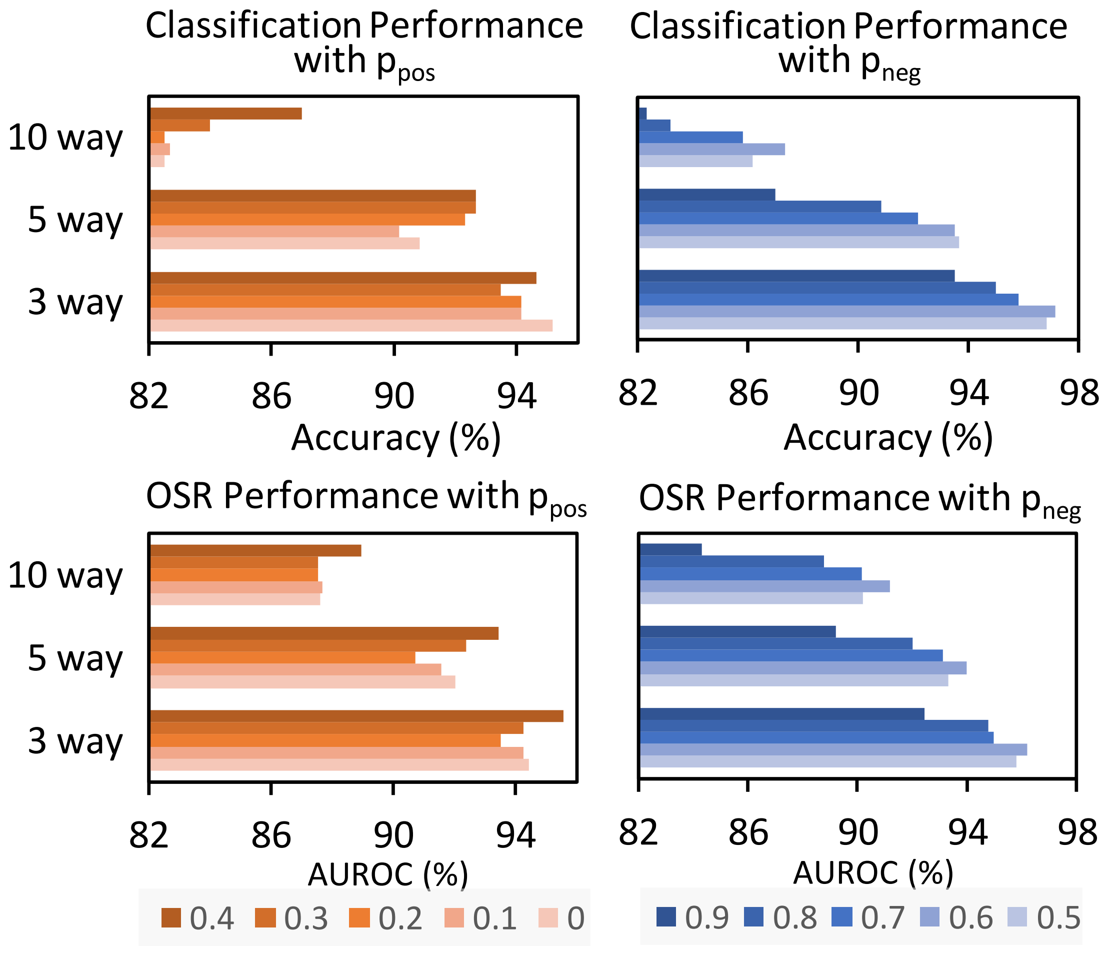


Figure 5.6 : Positive and Negative Dropout Probability (p_{pos} and p_{neg}) Selection.

to calibrate the original similarity score. I also tuned the calibration factor β with values 0.1, 0.5 and 1.0 and selected $\beta = 1.0$ which performed best (see the Table 5.2). This experiment was implemented with p_{pos} (positive dropout probability) 0.4 , p_{neg} (negative dropout probability) 0.6 and without hierarchical loss.

5.3.3 Ablation Study

In this section, I discuss the performance gained by introducing each enhancement from my proposal incrementally. Table 5.3 and Table 5.4 present performance results for the encrypted network traffic dataset and Omniglot dataset, respectively.

The first column of these tables is the baseline traditional SNN without applying

Table 5.2 : Calibration factor β selection.

N way	0.0	0.1	0.5	1.0
1 shot	AUROC			
3	95.70	95.76	95.85	**95.86**
5	92.41	92.44	92.52	**92.57**
10	90.03	90.04	90.14	**90.19**

Table 5.3 : Performance gained by introducing the **H**ierarchical cross entropy loss (H), multi-model **E**nsemble (E), bidirectional **D**ropout data augmentation (D) and DPGMM based task **A**daptive open-set recognition metric (A) incrementally on the encrypted network traffic dataset.

N way	SNN	+H	+H+E	+H+E+D	+H+E +D+A
1 shot	Accuracy/AUROC				
3	94.7/93.6	94.5/94.4	95.2/94.4	**97.2/96.2**	97.2/95.9
5	90.7/90.7	90.8/91.9	90.8/92.1	93.5/94.0	**93.5/94.1**
10	82.7/87.0	82.7/87.4	82.5/87.6	87.3/91.2	**87.3/91.2**

any proposed enhancements.

The above two tables demonstrate that the performance contribution of **H** is about 1.0% in OSR for the network traffic dataset as well as 1.0% and 0.5% in classification and OSR for the Omniglot dataset. The performance contribution of **E** is not very significant. To illustrate the performance difference clearly, I apply multi-model ensemble to the hierarchical SNN model on the encrypted network traffic dataset and the results shown in Figure 5.7 demonstrate that this strategy can improve both classification and OSR performance slightly. The performance

Table 5.4 : Performance gained by introducing each enhancement incrementally on the Omniglot dataset.

N way	SNN	+H	+H+E	+H+E+A
1 shot	Accuracy/AUROC			
5	96.1/96.0	96.8/96.3	97.2/96.5	**97.2/97.9**
10	95.1/94.2	**95.4**/94.4	95.2/94.8	95.2/**96.0**
20	90.2/91.6	89.7/91.4	91.3/91.9	**91.3/92.6**
30	85.1/86.4	85.7/87.0	85.8/87.5	**85.8/88.0**

values corresponding to the coarse loss factor 0.1 in Figure 5.7 overlay because the selected model is a single model, hence multi-model ensemble cannot work.

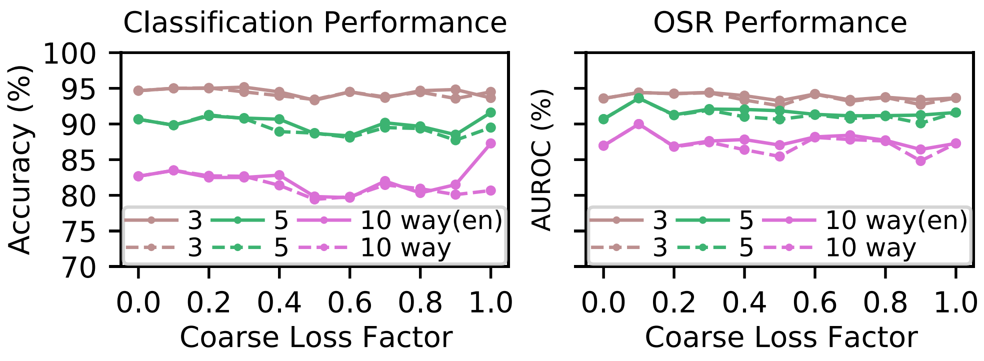


Figure 5.7 : Performance on hierarchical SNN model (dotted line) with the multi-model ensemble method (solid line).

The performance contribution of **D** is significant, from 2.0% to 4.8% in classification and from 1.8% to 3.6% in OSR. However, I do not apply **D** to the Omniglot dataset because valuable information (handwritten character) in each Omniglot image is centrally located, and is not distributed over the whole image. Therefore, it is hard to generate different similarity images by simply using the dropout strategy over all image pixels. In the future, I can attempt to apply the dropout strategy on relevant patches of the image ([50]) to generate positive and negative samples.

I notice that the contribution of **A** on the Omniglot dataset is greater than the encrypted network traffic dataset. This is due to the number and diversity of classes in the encrypted network traffic dataset (60 fine classes and 3 coarse classes) which is significantly less than the Omniglot dataset (1,623 fine classes and 50 coarse classes). I conclude the **A** enhancement boosts OSR performance, especially for dynamically changing query tasks with high diversity.

In summary, each enhancement contributes to the standard SNN model and the model with all enhancements increases classification and OSR performance significantly; demonstrating up to 4.6% and 4.2% improvement in accuracy and AUROC on the encrypted network traffic dataset, as well as 1.1% and 1.9% on the Omniglot dataset. To highlight the statistical significance of the improvements benefited from my proposed methods, I also provide t-test results in Table 5.5 and Table 5.6.

Furthermore, to strengthen the validity of my experimental results, I provide the mean and standard deviation of the accuracy and AUROC across five test sets. Each test set consists of 1,200 query tasks. Additionally, I select the t-test, a common statistical test used to compare the means of two groups, to validate the statistical significance of the enhancement improvements. I carry out t-tests with the null hypothesis that the means of each performance evaluation metric (i.e., accuracy and AUROC) from two models with and without specific enhancements are equal. I obtain the following p-values: p-value (+H) for SNN and SNN + H, p-value (+D) for SNN+H and SNN+H+D, p-value (+A) for SNN+H+D and SNN+H+D+A (in encrypted network traffic dataset) and p-value (+A) for SNN+H and SNN+H+A (in Omniglot dataset). I utilise these p-values and a significance level of 5% to evaluate whether I should reject the null hypothesis. I can conclude that an enhancement

Table 5.5 : Performance gained by introducing the **H**, **D** and **A** methods incrementally on the encrypted network traffic dataset.

N way	SNN	+H	p-value (+H)	+H+D	p-value (+D)	+H+D+A	p-value (+A)
1 shot	Accuracy/AUROC (%)						
3	93.9 $\pm$1.0/94.5 $\pm$ 0.5	94.5 $\pm$0.6/95.0 $\pm$ 0.3	.3/.2	96.9 $\pm$0.4/96.7 $\pm$ 0.2	$<$ .001 /$<$ .001	96.9 $\pm$0.4/96.6 $\pm$ 0.4	-/.9
5	88.8 $\pm$1.6/90.6 $\pm$ 0.3	89.3 $\pm$1.3/91.7 $\pm$ 0.7	.6/$<$ .01	93.0 $\pm$0.4/93.8 $\pm$ 0.4	$<$ .001 /$<$ .001	93.0 $\pm$0.4/93.8 $\pm$ 0.4	-/.9
10	80.1 $\pm$1.7/86.2 $\pm$ 0.5	81.4 $\pm$1.7/86.9 $\pm$ 0.6	.3/.1	85.7 $\pm$1.9/89.7 $\pm$ 0.8	$<$ .01 /$<$ .001	85.7 $\pm$1.9/89.7 $\pm$ 0.8	-/.9

Table 5.6 : Performance gained by introducing **H** and **A** methods incrementally on the Omniglot dataset.

N way	SNN	+H	p-value (+H)	+H+A	p-value (+A)
1 shot	Accuracy/AUROC (%)				
5	97.0 $\pm$0.7/96.8 $\pm$ 0.6	97.6 $\pm$0.4/97.3 $\pm$ 0.6	.2/.3	97.6 $\pm$0.4/98.6 $\pm$ 0.4	-/$<$ .01
10	94.0 $\pm$0.9/93.6 $\pm$ 0.9	94.5 $\pm$0.8/94.9 $\pm$ 0.4	.4/$<$ .05	94.5 $\pm$0.8/96.0 $\pm$ 0.4	-/$<$ .01
20	88.8 $\pm$0.9/90.2 $\pm$ 1.0	90.1 $\pm$0.8/91.4 $\pm$ 0.5	.06/.06	90.1 $\pm$0.8/92.2 $\pm$ 0.3	-/$<$ .05
30	84.9 $\pm$1.3/87.2 $\pm$ 1.0	86.0 $\pm$1.2/88.7 $\pm$ 0.8	.3/$<$ .05	86.0 $\pm$1.3/89.2 $\pm$ 0.8	-/.4

significantly improves performance if its p-value is less than 0.05. I did not consider multi-model ensemble method here because this method is mainly used to address the model selection problem. The method assists my model in obtaining more stable results but does not improve the performance significantly. Table 5.5 and Table 5.6 back up the performance analysis stated before. That is, **D** (bidirectional dropout data augmentation) can improve both closed-set classification and OSR performance on the encrypted network traffic dataset; **A** (DPGMM based task adaptive OSR metric) can enhance OSR performance on the Omniglot dataset, and **H** (hierarchical SNN) also can partially boost OSR performance on both datasets. Combining all enhancements, my model presents up to 5.6% and 3.5% improvement in accuracy and AUROC on the encrypted network traffic dataset, as well as 1.3% and 2.4% on the Omniglot dataset.

5.4 Summary

This chapter addressed the open-set recognition of encrypted network traffic classification under dynamically changing tasks. First, a simple and efficient data augmentation method, bidirectional dropout, was proposed to enrich the training pairs by generating both positive and negative pairs. Then a novel DPGMM based decision metric was built to implement task adaptive open-set recognition. In addition, I constructed hierarchical cross entropy loss to improve the confidence of the similarity score and proposed a multi-model ensemble method to ensure the robustness of model selection. Experiments demonstrated that my approaches could contribute to closed-set and open-set classification on both the encrypted network traffic dataset and Omniglot dataset.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis mainly solved three research questions related to content level classification for deeply encrypted network traffic in the closed and open worlds.

- "Is it feasible to identify the content of deeply encrypted network traffic?"
- "How to handle classification for previously unseen classes in the open world?"
- "How to tackle dynamically changing classification tasks with few samples in the open world?"

For the first research question, I emphasized feasibility verification and application performance. I collected data and constructed features. Then I attempted a CNN model on video streaming traffic to verify the feasibility of unveiling the deep content of the encrypted network traffic captured in air and selected the best performing features for subsequent researches.

For the second research question, I found that typical classifiers only could see a limited set of classes during the training time. Consequently, they will not be able to handle real-world data that usually contain previously unseen classes. Therefore I proposed CRAAE – a unified framework for accurate location agnostic outlier detection. In order to establish a more accurate decision boundary, I introduced known outliers into the training process through the CRAAE model's generation function to implicitly calibrate the decision metric (reconstruction error) on the original data space. I also proposed explicit calibration on reconstruction error by the category information disentangled from the feature space. In addition, I proposed to leverage Uniform and Dirichlet noise instead of Gaussian noise to generate known outliers. I apply the chosen network architecture and various evaluation metrics to evaluate the performance of my proposal on both the image datasets and network traffic datasets under different outlier location scenarios. The results show that my method can be used as a unified model for image and network traffic datasets, achieving state-of-the-art performance in any outlier location scenario. Compared to the performance of OCND and MCOSR methods in close and far outlier detection, respectively, CRAAE can also achieve a similar performance level. Furthermore, I also attempted another calibration strategy, the DMM-based softmax calibration method, which regards softmax output as a Dirichlet mixed model and used the predicted proportion of each component to calibrate the softmax. This method is the first to treat the multi-class softmax output as a Dirichlet mixture model distribution. It is novel but only can slightly outperform the most basic softmax thresholding method when τ is equal to or less than 30.

For the third research question, I extend a large dataset-based application model to a few-shot learning-based application model. Under the dynamically changing task scenario, I addressed the one-shot open-set recognition problem for encrypted network traffic classification. First, I proposed a simple and efficient data augmentation method, bidirectional dropout, to enrich the training pairs by generating both positive and negative pairs. This method results in high performance on time series data samples or samples with uniformly distributed significant information. However, data such as digit pictures with a large proportion of background (i.e., information that does not assist the decision) will not benefit from this method. In addition, I utilised the support set of each query task to construct task adaptive negative pairs and fitted a DPGMM distribution which is used to calibrate the original decision metric for open-set recognition. This DPGMM based method performs well for datasets with diverse and large number of classes but has less effect on simple datasets with fewer classes. Finally, I constructed a hierarchical cross entropy loss to improve the confidence of the similarity score. This method extended the original single loss to a combined, hierarchical loss to maintain the consistency of loss functions at multiple levels by leveraging multiple labels available for each sample. Experiments demonstrated that my proposed approaches could increase closed-set and open-set classification performance on both the encrypted network traffic dataset and Omniglot dataset.

6.2 Future work

As stated before, my research about encrypted network traffic data has solved the classification application in the closed and open world, and also tackled the changing task and few sample problems. My future work includes two aspects. One is about model improvement, and another one is about application extension. For model improvement, I should consider scalable novel class discovery and cross-domain novelty detection in few-shot learning. My proposed novelty detection approaches can identify the novel classes but still can not add the identified novel classes into known classes. My few-shot learning model focused on video streaming traffic and had not tackled the cross-domain problems. For example, the novel class is from audio or web surfing streaming traffic. I have investigated some papers in this field and found that cross-media retrieval is a promising way to address this problem. However, existing research on cross-media retrieval is usually based on a large-scale dataset. Therefore, combing the few-shot learning ideas with the cross-media retrieval approaches should be an interesting and challenging future work. For application extension, I have applied my proposed approaches to the encrypted network traffic dataset and some image datasets. I can verify my approaches on more datasets with different traffic types (the data format is not the same as our collected data), such as Website fingerprinting and voice command fingerprinting. Then I can explore the implementation of my approaches in other application scenarios, like intrusion detection.

Chapter 7

Appendix

7.1 Neural Networks Parameters

Table 7.1 lists the parameters that I used in closed world network traffic classification.

Table 7.1 : List of neural networks parameters for closed world classification.

Parameter Name	**Value**
Learning Rate	1e-4
Batch Size	64
Activation	ReLU
Optimizer	Adam
Batch Normalisation Decay	0.5
Batch Normalisation Epsilon	1e-3

For CRAAR novelty detection, the neural network parameters are presented in the Table 7.2.

7.2 Additional Experiment Results

SNN is the benchmark. I had done many experiments before using it as the basic architecture of my proposed methods. For example, feature selection, optimizer selection, data split, and best-trained model selection.

Table 7.2 : List of neural networks parameters for novelty detection.

Parameter Name	**Mnist**	**Network traffic**	**Other datasets**
Learning Rate	2e-3	1e-4	2e-3
Batch Size	128	64	128
Latent Size	16	64	32
Training Epoch	80	100	80
Learning Rate Decay	0.25 every 30 epochs		
Activation	LeakyReLU with negative slope as 0.2		
Optimizer	Adam		

I first evaluated this model on the previously collected encrypted network traffic dataset captured in air. I only collected 10 class samples (100 samples per class). Therefore, the dataset is insufficient for an SNN to train an accurate classification for a new task. Then we collected more classes, 60 in total, and each class had 100 samples. To simplify the data collection experiment environment, we used a physical port to capture network traffic. However, we only extracted Data Link Layer information such as packet number and packet length of the frames as our data features.

Next, I will introduce those experiments for building the baseline with an SNN architecture.

The new dataset consists of 60 video streaming classes from three different video platforms (YouTube, Netflix, and Stan, with 20 classes for each platform). I split 60 classes into training and test set. The training set consists of 40 classes of video traffic where 0 to 14 classes correspond to 1 to 15 classes of Netflix platform, 15 to 29 classes correspond to 1 to 15 classes of Stan platform, and 30 to 39 classes correspond to 1 to 10 classes of YouTube platform. Each class of video traffic has

100 samples (i.e., 100 runs for each video). The left 20 classes are utilised as the test set.

To make more types of N-way one-shot learning, e.g., 15-way and 20-way one-shot learning, I utilise the left 20 classes as the test set (without validation set). I test the model performance and save the model every 100 training batches where the batch size is 128. There are 1,000 test batches (test batch size is N for N-way one-shot). I evaluate the performance in terms of best average accuracy (for closed-set classification) and AUROC (for open-set recognition) measure metrics. The best average accuracy is the maximum mean of every 20 successive test epoch results. I use this metric because it can represent the more approximate convergence status performance in no validation set condition. Test batch is constructed as stated in Figure 5.1. I normalize the output of each test batch using softmax. Then I utilise the softmax score as the decision score for classification and novelty detection. The experiment results are shown in Table 7.3. I can see the AUROC is not sensitive to the ratio of inliers to outliers. Therefore, in subsequent experiments, I will fix the ratio of inliers to outliers as 1:1.

Table 7.3 : Open-Set Recognition (OSR) performance of an SNN.

N Way	Best Average	AUROC	
1 Shot	Accuracy	inliers:outliers = 1:1	inliers:outliers = 3:1
3	89.75	66.73	64.98
5	83.27	61.52	60.89
10	71.6	57.97	58.83
15	65.62	57.8	58.0
20	59.27	57.46	57.09

The experiment results confirm that an SNN can classify the encrypted network traffic dataset with reasonable accuracy when changing the classification task dynamically. However, novelty detection performance for open-set scenarios is not as good as close-set classification performance. Then I do extensive feature engineering and hyper-parameter selection experiments to improve the performance of an SNN.

First, I attempt to leverage the output of an SNN (similarity score) rather than the softmax score as the decision score directly. Second, I try different feature combinations based on the results listed in Figure 3.2. I initially use all 24 features together. However, the performance is similar to using just one feature. This suggests that low-performance features could interfere with classification. According to the performance ranking in Figure 3.2, F1, F2, F3, and F4 perform well and stably across different directions. Specifically, F1 and F2 represent the bursts of streaming the video content (data frames), and F3 and F4 represent the interaction process between the server and the client (non-data frames). Therefore, I only select the better-performing ones (i.e., F1 and F3) to represent data and non-data frames. I utilise F1 and F3 in different directions (up-link (U), down-link (D) and combining up- and down-link (C)) together (six features, F1 (U, D, C) and F3 (U, D, C)) to evaluate the performance. The results, as shown in Table 7.3 and Table 7.4, demonstrate that the open-set recognition performance using six features is significantly better than using just one feature. Moreover, the closed-set classification performance is similar for both situations. Considering computational efficiency and computer memory capacity, I apply fewer (four) features to this model. The best-performing combination is (F1 (U, C) and F3 (D, C), which outperforms using one feature but is not as effective as using six features. From these four features, I further narrow down to the best-performing two features: F1 (C) and F3 (D). Applying F1 (C) and F3 (D), the performance for both closed-set classification and open-set recognition is highly improved. Therefore, I will use these two features in subsequent experiments. Finally, I evaluate different optimizers (i.e., SGD and

Adam). The experiment results are list in Table 7.4. Regarding optimizer selection, I knew SGD usually could achieve better local convergence with enough training iteration. However, it is too slow, and I can see Adam outperforms SGD when training iteration times are the same because Adam converges more quickly. Furthermore, I evaluate the SNN performance according to the selected optimizer and features (i.e., Feature F3 and F1, that is, the number of non-data frames on the down-link (D) and the number of data frames on combination links (C) (i.e., both up-link (U) and down-link (D).). I note that the performance is improved significantly based on the new decision score, Adam optimizer, and features F1 (C) and F3 (D), see Table 7.5.

Table 7.4 : Feature engineering and hyper-parameter selection for an SNN.

Feature Selection		
Features	Best Average Accuracy (5-way 1-shot)	AUROC (inliers:outliers = 1:1)
F1 (U, D, C), F3 (U, D, C)	82.29	81.46
F1 (U, C), F3 (D, C)	82.09	79.62
F1 (C), F3 (D)	**86.55**	**83.51**

Optimizer Comparison		
N Way 1 Shot	Best Average Accuracy	AUROC
	SGD/Adam	
5	84.09/**86.55**	**87.3**/83.51
20	59.10/**63.99**	72.83/**73.08**

My previous experiments focus on new classification tasks (i.e., based on unknown classes). I also evaluate the OSR performance of an SNN for classification tasks based on known classes. For this experiment, I do a small change on the dataset split. I divide the previous training set into two sets, one for training and another for test. The new training set only has 80 samples for each class, and the new test set has another 20 samples for each class. I still keep the original test set for experiment comparison.

Table 7.6 demonstrates that an SNN can obtain high performance on both orig-

Table 7.5 : OSR performance of an SNN with F1 (C), F3 (D) and Adam optimizer.

N Way 1 Shot	Best Average Accuracy	AUROC
3	90.96	87.9
5	86.55	83.51
10	75.11	79.95
15	68.27	73.91
20	63.99	73.08

inal and new classification tasks. Here, I add another performance measure metric, Best_Acc, to see the upper bound of the performance. This accuracy is the best accuracy among the best 20 successive accuracies.

Table 7.6 : OSR performance of an SNN on original and new classification tasks.

N Way	Best_Ave_Acc	Best_Acc	AUROC	Best_Ave_Acc	Best_Acc	AUROC
1 Shot	Test set sliced from 40 training classes			Test set from 20 test classes		
5	93.78	94.93	96.12	84.67	86.4	84.95
10	89.09	89.9	95.13	73.94	75.73	78.28
15	86.17	87.15	94.16	67.22	69.02	73.53
20	84.09	85.16	94.34	62.41	63.86	71.93

My previous experiments have no validation set for constructing more kinds of test classification tasks. To adapt more common experiment setting, I slice the validation set from the current test set. The validation set consists of 10 classes of video traffic, where 1 to 5 classes correspond to 15 to 19 classes of the YouTube platform, and 6 to 10 classes correspond to 16 to 20 classes of the Stan platform. The test set consists of 10 classes of video traffic where 1 to 5 classes correspond to

11, 12, 13, 14, 20 classes of the YouTube platform, and 6 to 10 classes correspond to 16 to 20 classes of the Netflix platform.

I first evaluate on the validation set and select the five best performing models for N-way one-shot tasks. Then I test the performance of these selected models on the test set. I also exchange the validation and test set, using the test set for selecting models and the validation set for testing the performance.

Table 7.7 : OSR performance of an SNN on models selected through validation.

N Way	Average Acc	Average AUROC	Acc	AUROC
1 Shot	Validation set for model selection			
5	88.73	88.7	88.5	89.01
10	82.17	83.34	81.83	83.62
15	76.89	81.36	80.16	78.96
20	72.43	77.15	74.33	77.3
	Test set for model selection			
5	77.39	82.71	79.66	83.25
10	66.03	78.72	64.83	82.2
15	61.79	76.26	63.83	80.14
20	56.33	75.65	57.83	77.51

In Table 7.7, average accuracy and AUROC are the mean values for results on the selected five successive best performing models. Accuracy and AUROC are the mean values for five repeated running results on the best performing model. For an SNN, I only need one training process compared to prototype network approaches because the construction of training pairs for any way any shot learning is the same. Every 100 training batches, I validate the performance for 5-, 10-, 15-, and 20-way one-shot learning, respectively, to select the best performing models and then test on 5-, 10-, 15-, and 20-way one-shot learning models separately.

I need to clarify that the 15-way and 20-way one-shot learning in this experiment is not the real sense of 15-way and 20-way. I only have 10 classes of test samples; 15- and 20-way can be constructed when some of the negative query pairs are repeatedly sampled from the same class. Even though the test batch size is 15 or 20, some of the negative test pairs are very similar. For example, one negative pair is constructed by a sample 'A' from the class '0' and a sample 'B' from the class '1', and another negative pair is constructed by a sample 'A' from the class '0' and a sample 'C' from the class '1'. Despite that, to some extent, I can predict the performance of 15-way and 20-way one-shot scenarios by this kind of test batch construction.

In addition, I can infer from the results list in Table 7.7 that my model achieves convergence because the best performing model performs consistently with the best five successive models. To compare with other popular few-shot learning model frameworks, I re-arrange the dataset split randomly, which is the same as Meta-open and MAML, in Chapter 5 of this thesis.

Bibliography

[1] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian conference on computer vision.* Springer, 2018, pp. 622–637.

[2] K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum, "Infinite mixture prototypes for few-shot learning," in *International Conference on Machine Learning.* PMLR, 2019, pp. 232–241.

[3] E. Alpaydin, *Introduction to machine learning.* MIT press, 2020.

[4] R. Alshammari and A. N. Zincir-Heywood, "Investigating two different approaches for encrypted traffic classification," in *Sixth Annual Conference on Privacy, Security and Trust.* IEEE, 2008, pp. 156–166.

[5] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.

[6] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," in *Seventh International Conference on Learning Representations*, 2019.

[7] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[8] A. Azab, M. Khasawneh, S. Alrabaee, K.-K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digital Communications and Networks*, 2022.

[9] A. Azab, O. Maruatona, and P. Watters, “Avocad: Adaptive terrorist comms surveillance and interception using machine learning,” in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).* IEEE, 2019, pp. 85–94.

[10] A. Azab, P. Watters, and R. Layton, “Characterising network traffic for skype forensics,” in *2012 Third cybercrime and trustworthy computing workshop.* IEEE, 2012, pp. 19–27.

[11] A. Bendale and T. E. Boult, “Towards open set deep networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[12] L. Bernaille and R. Teixeira, “Early recognition of encrypted applications,” in *International Conference on Passive and Active Network Measurement.* Springer, 2007, pp. 165–175.

[13] C. M. Bishop, “Novelty detection and neural network validation,” *IEE Proceedings-Vision, Image and Signal processing*, vol. 141, no. 4, pp. 217–222, 1994.

[14] J. Bitterwolf, A. Meinke, and M. Hein, “Certifiably adversarially robust detection of out-of-distribution data,” *arXiv preprint arXiv:2007.08473*, 2020.

[15] D. M. Blei, M. I. Jordan *et al.*, “Variational inference for dirichlet process mixtures,” *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[16] A. Callado, C. Kamienski, G. Szabó, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, “A survey on internet traffic identification,” *IEEE communications surveys & tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[17] E. N. Ceesay, T. N. Do, and P. A. Watters, "Cyber-situational awareness in the presence of encryption," in *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2017, pp. 1621–1626.

[18] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[19] M. Chen, X. Zhu, Z. Qin, and Y. Wang, "Few-shot website fingerprinting attack," *arXiv preprint arXiv:2101.10063*, 2021.

[20] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[21] P. Chhabra, A. John, and H. Saran, "Pisa: automatic extraction of traffic signatures," in *International Conference on Research in Networking*. Springer, 2005, pp. 730–742.

[22] K. N. Choi, A. Wijesinghe, C. M. M. Kattadige, K. Thilakarathna, S. Seneviratne, and G. Jourjon, "Seta: Scalable encrypted traffic analytics in multi-gbps networks," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE, 2020, pp. 389–392.

[23] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[24] T. Dahanayaka, G. Jourjon, and S. Seneviratne, "Understanding traffic fingerprinting cnns," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE, 2020, pp. 65–76.

[25] T. Dahanayaka, Z. Wang, G. Jourjon, and S. Seneviratne, "Inline traffic analysis attacks on dns over https," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE, 2022, pp. 132–139.

[26] F. Del Buono, F. Calabrese, A. Baraldi, M. Paganelli, and A. Regattieri, "Data-driven predictive maintenance in evolving environments: A comparison between machine learning and deep learning for novelty detection," in *Sustainable Design and Manufacturing: Proceedings of the 8th International Conference on Sustainable Design and Manufacturing (KES-SDM 2021)*. Springer, 2021, pp. 109–119.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[28] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[29] A. R. Dhamija, M. Günther, and T. Boult, "Reducing network agnostophobia," in *Advances in Neural Information Processing Systems*, 2018, pp. 9157–9168.

[30] Y. Dhote, S. Agrawal, and A. J. Deen, "A survey on feature selection techniques for internet traffic classification," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2015, pp. 1375–1380.

[31] X. Dong, J. Shen, D. Wu, K. Guo, X. Jin, and F. Porikli, "Quadruplet network with one-shot learning for fast visual object tracking," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3516–3527, 2019.

[32] S. Doveh, E. Schwartz, C. Xue, R. Feris, A. Bronstein, R. Giryes, and L. Karlinsky, “Metadapt: Meta-learned task-adaptive architecture for few-shot classification,” *Pattern Recognition Letters*, vol. 149, pp. 130–136, 2021.

[33] R. Duan, D. Li, Q. Tong, T. Yang, X. Liu, and X. Liu, “A survey of few-shot learning: an effective method for intrusion detection,” *Security and Communication Networks*, vol. 2021, pp. 1–10, 2021.

[34] M. Farsi, K. Ratcliff, and M. Barbosa, “An overview of controller area network,” *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 113–120, 1999.

[35] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[36] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, “Network anomaly detection with the restricted boltzmann machine,” *Neurocomputing*, vol. 122, pp. 13–23, 2013.

[37] M. Foley, C. Hicks, K. Highnam, and V. Mavroudis, “Autonomous network defence using reinforcement learning,” in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1252–1254.

[38] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, “Generative openmax for multi-class open set classification,” *arXiv preprint arXiv:1707.07418*, 2017.

[39] S. Gidaris and N. Komodakis, “Dynamic few-shot visual learning without forgetting,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

[40] S. Gidaris and N. Komodakis, “Generating classification weights with gnn de-

noising autoencoders for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 21–30.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[42] Google, "HTTPS encryption on the web," https://transparencyreport.google.com/https/overview?hl=en&load_os_region=chrome-usage:1;series:page-load;groupby:os&lu=load_os_region, 2019, online; accessed 27-11-2019.

[43] D. Görür and C. E. Rasmussen, "Dirichlet process gaussian mixture models: Choice of the base distribution," *Journal of Computer Science and Technology*, vol. 25, no. 4, pp. 653–664, 2010.

[44] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and M. Norouzi, "Your classifier is secretly an energy based model and you should treat it like one," in *International Conference on Learning Representations*, 2020.

[45] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1321–1330.

[46] I. Guyon and A. Elisseeff, "An introduction to feature extraction," in *Feature extraction*. Springer, 2006, pp. 1–25.

[47] S. Ha, M. Kersner, B. Kim, S. Seo, and D. Kim, "Marionette: Few-shot face reenactment preserving identity of unseen targets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 893–10 900.

[48] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3018–3027.

[49] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," in *Proceedings of the 2020 SIAM International Conference on Data Mining.* SIAM, 2020, pp. 154–162.

[50] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[51] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *International Conference on Learning Representations*, 2019.

[52] S. Huang, J. Ma, G. Han, and S.-F. Chang, "Task-adaptive negative class envision for few-shot open-set recognition," *arXiv preprint arXiv:2012.13073*, 2020.

[53] IANA, "Service Name and Transport Protocol Port Number Registry," https://www.iana.org/assignments/service-names-port-numbers, 2020, online; accessed 25-06-2020.

[54] Izgi Arda Ozsubasi, "Few-Shot Learning (FSL): What it is its Applications," https://research.aimultiple.com/few-shot-learning/, 2020, online; accessed 25-03-2021.

[55] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 21–26.

[56] M. Jeong, S. Choi, and C. Kim, "Few-shot open-set recognition by transformation consistency," *arXiv preprint arXiv:2103.01537*, 2021.

[57] Z. Ji, X. Chai, Y. Yu, Y. Pang, and Z. Zhang, "Improved prototypical networks for few-shot learning," *Pattern Recognition Letters*, vol. 140, pp. 81–87, 2020.

[58] X. Jiang, "Feature extraction for image recognition and computer vision," in *2009 2nd IEEE international conference on computer science and information technology.* IEEE, 2009, pp. 1–15.

[59] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PloS one*, vol. 11, no. 6, p. e0155781, 2016.

[60] C. Kattadige, K. N. Choi, A. Wijesinghe, A. Nama, K. Thilakarathna, S. Seneviratne, and G. Jourjon, "Seta++: Real-time scalable encrypted traffic analytics in multi-gbps networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3244–3259, 2021.

[61] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Platform Technology and Service (PlatCon), 2016 International Conference on.* IEEE, 2016, pp. 1–5.

[62] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[63] K. Konda, X. Bouthillier, R. Memisevic, and P. Vincent, "Dropout as data augmentation," *stat*, vol. 1050, p. 29, 2015.

[64] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[65] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[66] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[67] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryiAv2xAZ

[68] Let's Encrypt, "Let's Encrypt Stats," https://letsencrypt.org/stats/, 2023, online; accessed 15-11-2023.

[69] Y. Li, Y. Huang, R. Xu, S. Seneviratne, K. Thilakarathna, A. Cheng, D. Webb, and G. Jourjon, "Deep content: Unveiling video streaming content from encrypted wifi traffic," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2018, pp. 1–8.

[70] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *International Conference on Learning Representations*, 2018.

[71] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham *et al.*, "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2000, pp. 12–26.

[72] B. Liu, H. Kang, H. Li, G. Hua, and N. Vasconcelos, "Few-shot open-set recognition using meta-learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8798–8807.

[73] H. Liu and H. Motoda, *Computational methods of feature selection.* CRC Press, 2007.

[74] M. Liu, Z. Liu, W. Lu, Y. Chen, X. Gao, and N. Zhao, "Distributed few-shot learning for intelligent recognition of communication jamming," *IEEE Journal*

of Selected Topics in Signal Processing, vol. 16, no. 3, pp. 395–405, 2021.

[75] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *CoRR*, vol. abs/1709.02656, 2017. [Online]. Available: http://arxiv.org/abs/1709.02656

[76] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[77] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[78] A. K. J. Michael, E. Valla, N. S. Neggatu, and A. W. Moore, "Network traffic classification via neural networks," 2017.

[79] S. Migliorati, A. Ongaro, and G. S. Monti, "A structured dirichlet mixture model for compositional data: inferential and applicative issues," *Statistics and Computing*, vol. 27, no. 4, pp. 963–983, 2017.

[80] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[81] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Tech. Rep., 2013.

[82] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 50–60.

[83] K. L. Moore, T. J. Bihl, K. W. Bauer Jr, and T. E. Dube, "Feature extraction and feature selection for classifying cyber traffic threats," *The Journal of Defense Modeling and Simulation*, vol. 14, no. 3, pp. 217–231, 2017.

[84] K. Moriarty and A. Morton, "Effects of Pervasive Encryption on Operators," Internet Requests for Comments, RFC Editor, RFC 8404, July 2018. [Online]. Available: https://tools.ietf.org/html/rfc8404

[85] E. Nalisnick, A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan, "Detecting out-of-distribution inputs to deep generative models using a test for typicality," *arXiv preprint arXiv:1906.02994*, vol. 5, 2019.

[86] K. W. Ng, G.-L. Tian, and M.-L. Tang, *Dirichlet and related distributions: Theory, methods and applications.* John Wiley & Sons, 2011, vol. 888.

[87] T. T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks.* IEEE, 2006, pp. 369–376.

[88] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[89] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS), BICT-15*, vol. 15, 2015, pp. 21–26.

[90] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[91] P. Oza and V. M. Patel, "C2ae: Class conditioned auto-encoder for open-set recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2307–2316.

[92] P. Perera, R. Nallapati, and B. Xiang, "Ocgan: One-class novelty detection using gans with constrained latent representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2898–2906.

[93] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," *arXiv preprint arXiv:1709.07871*, 2017.

[94] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," in *Advances in neural information processing systems*, 2018, pp. 6822–6833.

[95] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[96] S. Puch, I. Sánchez, and M. Rowe, "Few-shot learning with deep triplet networks for brain imaging modality recognition," in *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*. Springer, 2019, pp. 181–189.

[97] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 759–766.

[98] K. Raman *et al.*, "Selecting features to classify malware."

[99] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate dash

videos streamed over encrypted 802.11n connections," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2016, pp. 1107–1112.

[100] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy.* ACM, 2017, pp. 361–368.

[101] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 707–14 718.

[102] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *arXiv preprint arXiv:1708.06376*, 2017.

[103] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 135–148.

[104] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[105] G. Salomon, A. Britto, R. H. Vareto, W. R. Schwartz, and D. Menotti, "Open-set face recognition for small galleries using siamese networks," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP).* IEEE, 2020, pp. 161–166.

[106] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint*

arXiv:1605.06065, 2016.

[107] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, “Toward open set recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1757–1772, 2012.

[108] W. J. Scheirer, L. P. Jain, and T. E. Boult, “Probability models for open set recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[109] R. Schuster, V. Shmatikov, and E. Tromer, “Beauty and the burst: Remote identification of encrypted video streams,” in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1357–1374. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schuster

[110] V. Sehwag, M. Chiang, and P. Mittal, “Ssd: A unified framework for self-supervised outlier detection,” *arXiv preprint arXiv:2103.12051*, 2021.

[111] S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in-network identification of p2p traffic using application signatures,” in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 512–521.

[112] J. Shen, X. Tang, X. Dong, and L. Shao, “Visual object tracking by hierarchical attention siamese network,” *IEEE transactions on cybernetics*, vol. 50, no. 7, pp. 3068–3080, 2019.

[113] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[114] Y. Shu, Y. Shi, Y. Wang, T. Huang, and Y. Tian, “p-odn: prototype-based open deep network for open set recognition,” *Scientific reports*, vol. 10, no. 1, pp. 1–13, 2020.

[115] L. Sintini and L. Kunze, “Unsupervised and semi-supervised novelty detection using variational autoencoders in opportunistic science missions.” in *BMVC*, 2020.

[116] P. Sirinam, M. Imani, M. Juarez, and M. Wright, “Deep fingerprinting: Undermining website fingerprinting defenses with deep learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[117] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” *arXiv preprint arXiv:1703.05175*, 2017.

[118] E. Suwannalai and C. Polprasert, “Network intrusion detection systems using adversarial reinforcement learning with deep q-network,” in *2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE, 2020, pp. 1–7.

[119] J. Tack, S. Mo, J. Jeong, and J. Shin, “Csi: Novelty detection via contrastive learning on distributionally shifted instances,” *arXiv preprint arXiv:2007.08176*, 2020.

[120] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for network intrusion detection in software defined networking,” in *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*. IEEE, 2016, pp. 258–263.

[121] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.

[122] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *nature*, vol. 381, no. 6582, pp. 520–522, 1996.

[123] L. V. Utkin, V. S. Zaborovsky, A. A. Lukashin, S. G. Popov, and A. V. Podolskaja, "A siamese autoencoder preserving distances for anomaly detection in multi-robot systems," in *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. IEEE, 2017, pp. 39–44.

[124] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 3637–3645.

[125] L. Vu, C. T. Bui, and Q. U. Nguyen, "A deep learning based method for handling imbalanced problem in network traffic classification," in *Proceedings of the 8th international symposium on information and communication technology*, 2017, pp. 333–339.

[126] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[127] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[128] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7278–7286.

[129] Z.-M. Wang, J.-Y. Tian, J. Qin, H. Fang, and L.-M. Chen, "A few-shot learning-based siamese capsule network for intrusion detection with imbalanced training data," *Computational intelligence and neuroscience*, vol. 2021, 2021.

[130] D. Webb, "Applying softmax classifiers to open set," in *Australasian Conference on Data Mining.* Springer, 2019, pp. 104–115.

[131] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3540–3552, 2020.

[132] M. Ye and Y. Guo, "Deep triplet ranking networks for one-shot recognition," *arXiv preprint arXiv:1804.07275*, 2018.

[133] S. Yoon, Y. kyun Noh, and F. Park, "Autoencoding under normalization constraints," in *ICML*, 2021.

[134] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4016–4025.

[135] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[136] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky, "Few-shot adversarial learning of realistic neural talking head models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9459–9468.

[137] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision.* Springer, 2014, pp. 818–833.

[138] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 2018, pp. 727–736.

[139] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song, "Metagan: An adversarial approach to few-shot learning." *NeurIPS*, vol. 2, p. 8, 2018.

[140] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[141] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.