Computer Networks 236 (2023) 109991

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Robust open-set classification for encrypted traffic fingerprinting

Thilini Dahanayaka [a,*], Yasod Ginige [b], Yi Huang [c], Guillaume Jourjon [d], Suranga Seneviratne [a]

[a] School of Computer Science, The University of Sydney, Australia
[b] Department Electronics and Telecommunication Engineering, University of Moratuwa, Sri Lanka
[c] University of Technology Sydney, Australia
[d] CSIRO-Space & Astronomy, Australia

ARTICLE INFO

Keywords:
Traffic analysis
Traffic fingerprinting
Open-set classification
Deep neural network quantization

ABSTRACT

Encrypted network traffic has been known to leak information about their underlying content through side-channel information leaks. Traffic fingerprinting attacks exploit this by using machine learning techniques to threaten user privacy by identifying user activities such as website visits, videos streamed, and messenger app activities. Although state-of-the-art traffic fingerprinting attacks have high performances, even undermining the latest defenses, most of them are developed under the *closed-set* assumption. To deploy them in practical situations, it is important to adapt them to the *open-set* scenario, which allows the attacker to identify its target content while rejecting other background traffic. At the same time, in practice, these models need to be deployed on in-networking devices such as programmable switches, which have limited memory and computation power. Model weight quantization can reduce the memory footprint of deep learning models while at the same time, allowing inference to be done as integer operations as opposed to floating point operations. Open-set classification in the domain of traffic fingerprinting has not been explored well in prior work and none of them explored the effect of quantization on the open-set performance of such models. In this work, we propose a framework for robust open-set classification of encrypted traffic based on three key ideas. First, we show that a well-regularized deep learning model improves the open-set classification and then we propose a novel open-set classification method with three variants that perform consistently over multiple datasets. Next, we show that traffic fingerprinting models can be quantized without a significant drop in both closed-set and open-set accuracy and therefore, they can be readily deployed on in-network computing devices. Finally, we show that when the above three components are combined, the resulting open-set classifier outperforms all other open-set classification methods evaluated across five datasets with a minimum and maximum increase in F1_Score of 8.9% and 77.3% respectively.

1. Introduction

Despite being end-to-end encrypted, network traffic flows can leak information through side-channels [1]. For example, previous works have demonstrated that visited websites [2,3], watched videos [4,5], messenger app activities [6], or even spoken words in VoIP calls [7] can be inferred by leveraging the statistical properties of encrypted traffic flows. The key idea was to use the features such as packet sizes, packet direction, or inter-packet times and train various machine learning models. Recent advances in deep learning further increased the accuracy of such inferences [2,3].

Using machine learning-based traffic fingerprinting in practice requires addressing the *open-set problem*. That is, the traffic classifier must be able to separate the traffic flows it can classify *(known–knowns)* from all other traffic flows *(unknown–unknowns)*. In Fig. 1, we illustrate a typical traffic fingerprinting use case that highlights the necessity of having an open-set classification component.

Assume the law enforcement is trying to prevent illegal trade over the anonymous network Tor and has the ability to passively observe encrypted Tor traffic in transit (e.g., at a vantage point of an ISP). They have identified a list of URLs for illegal online stores (i.e., the target list) and they want either to identify users who are browsing those URLs or simply to terminate those connections. Other than that, law enforcement does not have any interest in connections to all the other URLs (i.e., URLs that are not in the target list) happening over Tor.

To achieve this, law enforcement could first collect a dataset of traffic flows by visiting the target URLs themselves (i.e., the known classes or known–knowns) and train a classification model. However,

* Corresponding author.
E-mail addresses: tdah5330@uni.sydney.edu.au (T. Dahanayaka), yasodginige98@gmail.com (Y. Ginige), yi.huang-3@student.uts.edu.au (Y. Huang), guillaume.jourjon@data61.csiro.au (G. Jourjon), suranga.seneviratne@sydney.edu.au (S. Seneviratne).

https://doi.org/10.1016/j.comnet.2023.109991
Received 21 March 2023; Received in revised form 31 May 2023; Accepted 20 August 2023
Available online 23 August 2023

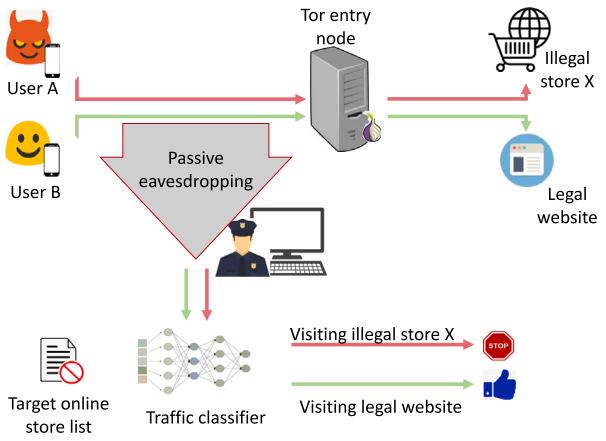


Fig. 1. Open-set traffic classification.

this naive closed-set model will have the limitation of predicting any traffic flow given to it as one of the URLs in the target list. For example, a benign user such as *User B* in Fig. 1 will also be classified as visiting an illegal URL. Existing work in encrypted traffic fingerprinting either *(i) focused only on the closed-set problem [4,5], (ii) addressed the problem by adding a background class or a binary classifier to separate knowns from unknowns [2,7,8]*, or *(iii) used thresholding on classifier confidence to filter unknowns [3]*. Accordingly, law enforcement can collect samples of visiting other URLs that are not in the target set (i.e., known–unknowns) to be used with options (ii) or (iii) from above.and add all of these samples as a single class during the training process (i.e., the background class) or use them to decide a rejection threshold (i.e., binary classification). However, these methods have their own limitations. First, they require samples from background traffic *(known-unknowns)* during training time and there is no guarantee that such available samples at training time will cover all *unknown-unknowns* the classifier may come across in the future. Thresholding on the classifier's confidence is based on the idea that the classifier must have high confidence for known classes. However, current deep learning models are known to have high confidence even if they are making a mistake [9]. As such, encrypted traffic fingerprinting models must be able to handle the *open-set* classification more realistically, where the model can correctly identify samples from known classes (i.e., known traffic flows) while effectively rejecting any samples from classes not seen during model training (i.e., background traffic flows).

At the same time, another key challenge in using deep learning-based traffic classifiers in real-world problems is that these models often need to be deployed on resource-constrained devices. More specifically, deploying traffic classification models on in-networking devices can provide the maximum use of their output in real-time compared to deploying the models on centralized servers. This is because, if traffic captured at network devices needs to be transferred to centralized servers for classification, it can only facilitate post-event-completion analysis. While post-event-completion analysis is useful in long-term network management tasks, real-time traffic classification is more beneficial as it can be used for real-time network management tasks. For example, rather than identifying a connection to an illegal store long after the store was accessed, real-time traffic classification can identify it while the illegal store is still being loaded and hence the connection can be terminated before the being fully loaded.

We identified two main resource restrictions on in-network devices that limit the deployment of deep-learning models on them. First, switches like Programming Protocol-independent Packet Processors (P4) [10] or Juniper Trio [11] that can be used as programmable switches for traffic classification, do not facilitate floating-point operations, but, trained parameters of deep learning models are often floating point numbers and require floating point operations. Second, the memory available on these devices for storing deep learning models is very limited [11], but deep learning models generally have high memory footprints due to their large number of parameters (floating point). For example, while the maximum capacity of SRAM and cache available on the Juniper Trio [11] is 8 MB and 24 MB, a typical website fingerprinting CNN can have a footprint of 8.5 MB (cf. Table 8) which would consume a significant portion of the available memory that needs to be shared with core routing functions of the switches as well as the above-mentioned prediction models. However, previous work on traffic fingerprinting has not placed enough emphasis on addressing these issues, and hence, a solution that adapts deep learning models to work with integers only and minimize the memory footprint of deep learning models needs to be developed.

To this end, we propose a framework that simultaneously addresses the open-set setting and the resource constraints of practical environments. First, we show that many of the existing traffic classifiers are not well-regularized and therefore result in poor performance in the open-set. Next, we propose a novel open-set classification method, *k-LND* based on *k-logit neighbor distances* with three variants, that perform reliably across all datasets considered. Finally, we show that traffic fingerprinting models can be quantized without a significant drop in accuracy and that the novel k-LND method has the least effect on performance due to model quantization. To the best of our understanding, this work is the first work to explore open-set classification in a resource-constrained setting that closely resembles real-world environments.

More specifically, we make the following contributions.

- We observe that many of the previous works that attempted open-set methods in traffic fingerprinting either used naive methods that resulted in poor performance or used closed-set classifiers that are not well-regularized, again resulting in poor open-set performance. Using five publicly available datasets, we show that having a well-regularized underlying closed-set classifier improves open-set results irrespective of the specific open-set classification method used, with an increase of F_Score in the range 2.99%–35.48%.
- We propose a novel open-set classification method based on *k-logit neighbor distances* with three variants that perform consistently across multiple datasets when compared to other methods. We show that k-LND methods always maintain consistent performance with $kLND_1$, $kLND_2$ and $kLND_3$ methods maintaining >85%, >85% and >95% closed-set and >65%, >75% and >75% open-set accuracies respectively.
- We evaluate the performance of open-set classification methods when the underlying deep learning model has been quantized. We show that traffic fingerprinting models can be quantized without a significant drop in accuracy while reducing the memory footprint of classifiers by at least 60%. We highlight that the novel k-LND methods always record the highest F_Scores when using quantized models across all datasets (with an increase in F1_Score in the range 8.9%–77.3%) while the best-performing variant $k\text{-}LND_3$ maintains closed-set and open-set accuracy of >90% and 70% respectively. Overall, we show that the combined contribution from a well-regularized closed-set classifier, novel k-LND open-set classification method, and model weight quantization outperforms all other methods compared across all datasets for open-set classification.

Overall, our framework supports open-set classification and model weight quantization which reduces model footprint and ensures compatibility with in-network computing devices that support only integer arithmetic.

The rest of the paper is organized as follows. Section 2 explains the traffic fingerprinting scenario considered in this work and introduces

the datasets, baseline model architectures, and existing open-set classification methods used in this work as comparisons. Section 3 presents our framework for robust open-set classification and the underlying key ideas while Section 4 presents the results. Section 5 discusses related work and Section 6 concludes the paper.

2. Background

In this section, we first provide a brief overview of the existing open-set classification methods that are most commonly used in encrypted traffic fingerprinting. Next, we introduce five publicly available datasets and the corresponding deep CNN classifier architectures that we use in our experiments.

2.1. Open-set classification methods

Next, we describe existing open-set classification methods that are commonly used by prior work in traffic fingerprinting and are used as baselines in this work.

2.1.1. Background class

As mentioned before, the background class method is the most naive way of handling open-set classification. It assumes that a subset of the open-set (i.e., samples from known-unknown classes) are available during model training, and they are combined to form a single *known-unknown* or background class. Given an input, the classifier learns either to put it into one of the known classes or into the background class, essentially making an n class classification problem into an $n+1$ class classification problem. Works in traffic classification such as [2,8] used this method to tackle the open-set problem.

The background class method is based on the strong assumption that all the samples from unknown classes will have similar characteristics as the samples from known–unknown classes. This may not necessarily be true all the time as there might be samples of unknown–unknown classes that are closer to a known class than the combined representation of known–unknown classes. Therefore, while the background class method performs well in some cases like the DF dataset [2], it performs poorly with others such as the DC dataset as we have demonstrated with an example in Appendix A.1.

2.1.2. Softmax thresholding

The majority of current deep neural networks use softmax activation in the last layer to obtain a probability vector representing the confidence of a given sample belonging to each known class. Softmax thresholding-based open-set classification is built on the intuition that any model trained on a set of classes will output a very low softmax score (confidence) for samples from the open-set. More specifically, this method simply uses a threshold over the softmax probability score of the predicted class and rejects samples with a probability lower than the threshold as open-set samples. Additionally, the attacker may use a small dataset from known unknowns to decide the threshold value to obtain a preferred balance between closed-set accuracy and open-set accuracy. Rimmer et al. [3] is one example of work that used softmax thresholding in traffic fingerprinting.

Nonetheless, since the softmax activation skews the output probabilities to favor the class with the highest probability and the training procedure does not explicitly push the model to output low confidence for open-set samples, this method cannot be expected to work always. For example, it is known that neural networks have very high confidence even if they make a wrong prediction [9]. As a result, softmax thresholding works well with some datasets but, performs poorly with others such as SETA as illustrated with an example in Appendix A.2.

2.1.3. OpenMax

OpenMax was originally proposed by Bendale and Boult [9] for open-set image classification. It is built on the intuition that the values from the penultimate layer (i.e., the layer before the softmax activation layer) of a deep neural network provide a distribution of how classes are related to each other as opposed to being independent per-class score estimates. Specifically, OpenMax uses Extreme Value Theory (EVT) modeling on the distance between a given sample and the mean of its predicted class to identify open-set samples. OpenMax has the added advantage that it does not require samples from the open-set during model training at all. Webb [12] and Yang et al. [13] explored the possibility of using OpenMax for traffic fingerprinting tasks.

When using OpenMax in our experiments, we make two main alterations to the original method proposed for computer vision applications to get better performance on network traffic data. First, we observed that for traffic fingerprinting networks, the Activation Vector sometimes contains negative logit values and when used for rescaling with EVT modeling, causes the score for being from the open-set to be increased. Therefore, we do min–max normalization on the Activation Vector (AV) before calculating the revised AV to minimize the effect of negative values in the AV on rescaling. Second, instead of using a fixed threshold for OpenMax probabilities, we used class-wise thresholds to reject open-set samples. The reasoning behind this decision is that the distribution of OpenMax probabilities differs between classes and using a common threshold may cause classes with generally lower OpenMax probabilities as the highest OpenMax score to have more false negatives (more closed-set samples rejected as open-set).

2.1.4. Ensemble learning

Wang et al. [14] proposed ensemble learning as a way of handling open-set traffic classification. More specifically, the authors assume that combining the outputs from multiple model instances based on a simple base learner would learn different sets of features and hence can help the overall model generalize better towards unknown data as opposed to a single model. Accordingly, the authors use a threshold on the averaged output from N different model instances to build an open-set website fingerprinting attack. During model training, a dynamic learning rate is used to separate model locations in the loss function by immediately increasing the learning rate at fresh starts, so that the learning point displaces by a large distance resulting in a new model having a different set of parameters and learning a different set of features. Additionally, the authors use squeeze and excitation layers as an attention technique to weigh features according to their effectiveness in the final result and increase the robustness of the model and separable convolution layers to reduce the computational cost. In our work, we implement the full ensemble model as in the original work but do not use the Denoising Auto Encoder to pre-process data and extract features since we want to make the data feed the same between all the methods we compare.

2.2. Datasets

We use five publicly available datasets corresponding to three types of encrypted network traffic as described below.

1. **Website fingerprinting over Tor:** *Automated website fingerprinting (AWF) [3]* and *Deep fingerprinting (DF) [2]* datasets contain network traffic traces for visiting homepages of top-200, and top-95 Alexa websites over `Tor`, respectively. In these datasets, each site visit is represented by the first 5000 (DF)/3000 (AWF) `Tor` packets in either direction. That is, in these datasets, a data sample is a sequence of +1 s (uploads) and −1 s (downloads). If a particular homepage visit did not generate a total respective number of packets in either direction, the remainder of the sequence was padded with zeros.

Table 1
Summary of datasets.

Dataset	Type	Details	Open-set
AWF [3]	Website	200 target websites (2500 traces/class)	400,000 classes
DF [2]	Website	95 target websites (1000 traces/class)	40,716 classes
DC [4]	Video	10 target YouTube videos (320 traces/class)	N/A[a]
SETA [5]	Video	20 target Netflix videos (100 traces/class)	N/A[a]
IoT [7]	Voice	98 target Google Home comms. (1500 traces/class)	N/A[a]

[a]We split the classes so that 40% of classes are in the closed-set.

2. **Video fingerprinting:** *Deep content dataset (DC)* [4] contains traffic traces for streaming the first three minutes of selected YouTube videos, while *SETA [5] dataset* contains the same for selected Netflix videos. In both the datasets, the three-minute interval is binned into 500 time slots (0.36 s each) and each time slot is represented by summary statistics of the packets observed during that time. While the original datasets comprised 24 features per trace, the authors of DC observed that the number of uplink packets of video streaming produced the most accurate model and hence we only use that feature in our work. Therefore, these datasets represent a traffic trace as a sequence of 500 integers.
3. **Voice command fingerprinting:** The *IoT [7] dataset* contains the traffic traces generated by Google Home devices for 98 specific voice commands. It represents a trace by the first 475 packets in either direction and for each packet, a −1 would denote a download and a +1 would denote an upload. If a particular voice command did not generate a total of 475 packets in either direction, the remainder of the sequence was padded with zeros.

We provide a summary of our datasets in Table 1. Here, *target classes* refers to the number of known/closed-set classes. For datasets that provide a separate open-set, we denote the number of classes in the open set under the last column and split datasets without a separate open-set (DC, SETA, and IoT) to create open-sets as discussed in Section 2.2.1 below.

2.2.1. Data preparation

The original works of AWF and DF considered the open-set scenario and hence already have a separate open-set. Therefore we did not have to manually split the dataset. In contrast, the original versions of DC, SETA, and IoT datasets did not have open-set samples. Therefore, to use these in our open-set experiments, we split the original datasets into two parts so that 40% of the original number of classes is used as the closed-set while the rest is considered as the open-set. To negate any effect on results from specific splits, we use multiple random splits for each dataset and report the average performance. The number of splits depends on the datasets. For DC and SETA datasets which have a smaller number of classes, we created five splits and for the IoT dataset, we created 10 splits. Later, when we report the results for these datasets, we report the average and standard deviation values of each metric across all splits of a given dataset.

Unless otherwise specified, for all the methods, the training, testing, and validation splits from the closed-set contain 200, 200, and 100 traces per class, respectively. While more data samples were available in the datasets, the original work, as well as subsequent work [15], showed that 200 training samples per class are sufficient to train a model with high test accuracy.

For the background class method, which is the only method that requires open-set samples during training (i.e., known-unknowns), we separate out 20% of classes as known-unknowns and use 200 and 100 traces from each known-unknown class in the training and validation sets, respectively. We ensured there is no overlap between the classes used for open-set during training and testing procedures. The test set comprised all the known and unknown class samples from the original test set.

2.3. Deep learning models

All of the open-set methods we use require an underlying deep neural network. For each dataset except AWF and SETA, the original work proposed the most suitable model architectures and hyperparameters, and hence we use those models as it is in our work. For the AWF dataset, we use the model proposed for DF [2] since Sirinam et al. [2] showed that the DF model is more effective for website fingerprinting. The original work for SETA did not use deep learning models and hence we use a deep CNN similar to that of DC. The model architectures used for DC and SETA datasets are shown in Figs. 3(a) and 3(b) respectively while the model architecture used with IoT is presented in Appendix B (Fig. 14).

We note that we increase dropout rates of original model architectures (for DC, SETA, and IoT only) to regularize models and the appropriate dropout rates are decided through tuning the model to have a high validation accuracy. Accordingly, for DC, SETA, and IoT models, the increased dropout rates are indicated in green in the respective figures. We further discuss this later in Section 3.1 Additionally, the ensemble model architectures used for the ensemble method of open-set classification (**cf.** Section 2.1.4) corresponding to all five datasets are explained in Appendix B.1.

3. Framework for robust open-set traffic fingerprinting

Our proposed framework for robust open-set classification targeted towards encrypted traffic fingerprinting in resource-restricted network devices is based on three key ideas. *(1) Regularized models*, *(2) k-Logit Neighbor Distance for open-set classification* and *(3) Model Quantization*.

3.1. Regularized model

All open-set classification methods we explore in our work use underlying closed-set classifiers (a deep learning model). We hypothesize that the performance of an open-set classifier depends on how well the underlying deep-learning model can identify class boundaries for the known classes. For instance, if the underlying closed-set classifier identifies a sub-optimal class boundary or is overfitted to a given dataset and class boundaries even accommodate for noise and irregularities, it would cause the subsequent open-set classifier to identify even open-set samples as samples from a known class. We illustrate such a scenario in Fig. 2.

Assume two closed-set classifiers trained on two classes with *Feature 1* and *Feature 2* referring to the features learned by the classifier (logit layer). Fig. 2 shows the distribution of Class A in this logit space. Although both Classifier 1 and 2 each have learned a boundary that encompasses all samples from Class A and would give 100% accuracy for Class A in the closed-set setting, we see that Classifier 1 has learned a sub-optimal boundary that covers a region that includes only noise samples and no actual samples from Class A. Hence, when used for open-set classification, Classifier 1 will label the open-set samples that are within the sub-optimal boundary but away from the actual Class A samples as Class A. In contrast, Classifier 2 has learned an optimal boundary that covers only the correct samples from Class A and when used for open-set classification can correctly reject all open-set samples outside the boundary of Class A. Due to the simplicity of network traffic data that allows a simple model to be easily trained to achieve high accuracies, if proper hyperparameter tuning is not done, there is a likelihood that the resulting model simply learns sub-optimal class

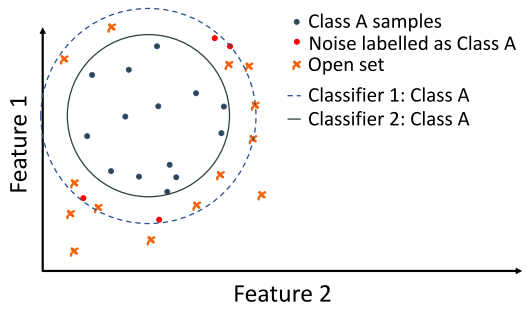
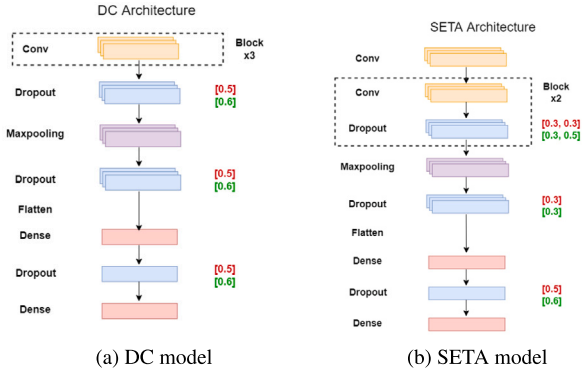


Fig. 2. Optimal class boundary.



(a) DC model (b) SETA model

Fig. 3. Model architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

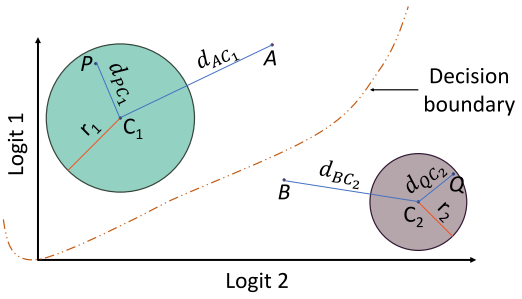


Fig. 4. k-LND method.

boundaries good enough to improve accuracy. For instance, in [4], the authors directly use the CNN model architecture proposed in [16] for traffic captured at the network level, on their traffic captured at data-link layer (WiFi) and still achieve high training and testing accuracies. As we show later, although the DC model performs well, it does not identify optimal class boundaries, and therefore tuning the model further improves open-set results.

Another possible source of such over-fitting is the number of closed-set classes. Out of the five datasets we explore, DC, SETA, and IoT did not have a separate open-set and therefore we had to split the original dataset to create an open-set. By doing so, we reduce the total number of classes (closed-set) the original model architecture was tuned for, and using the exact same model architecture as in the original work on the smaller dataset would result in overfitting.

Accordingly, we hypothesize that if a more robust model (i.e., a more generalized model that identifies optimal class boundaries without overfitting) is used as the underlying classifier, it will improve the results of any open-set classification method. To ensure that the underlying closed-set classifier has learned better class boundaries and is not overfitted to a particular dataset, we propose to properly regularize a closed-set classifier before using them for open-set classification. We use DC, SETA, and IoT datasets to test our hypothesis and regularize the baseline models (models from corresponding original work) by increasing the dropout rates. Then, we compare the open-set classification results for using the baseline model vs. the regularized model. We note that the original models used with AWF and DF datasets have undergone thorough hyperparameter tuning to ensure optimal performance and the original datasets are used in our work without splitting as they already contain open-sets. Therefore, we do not use those datasets in this experiment.

In Fig. 3, we have illustrated the deep CNN model architectures used for the DC dataset (Fig. 3(a)) proposed by [4], and SETA dataset (Fig. 3(b)). As explained before, we regularize the baseline model by increasing the dropout rates used in the baseline model. In Fig. 3, we have denoted the dropout rates to the right of each dropout layer, with the value in red referring to the baseline model parameter and the value in green referring to the value in the regularized model. We follow the same approach for the IoT dataset, and the corresponding figure is given in Appendix B.

3.2. k-Logit Neighbor Distance-based open-set classification

The second element of our framework is a novel distance-based open-set classification method named k-Logit Neighbor Distance (k-LND) Method with three variants. It is built on the intuition that the output of the logit layer (the layer before softmax activation) of a deep neural network represents how classes are related to each other as opposed to being independent per-class score estimates.

More specifically, if N is the number of closed-set classes, the output of the logit layer is a vector of length N that represents an N dimensional space where samples from the same class would be clustered together around its class center. Here, the class center is the average over the logit layer outputs of correctly classified training samples from the corresponding class, referred to as the Mean Activation Vector (MAV). Hence, if a sample is from the closed-set, we expect it to be closer to the MAV of its predicted class while being as far away from the MAVs of the k neighbor classes of the predicted class. In contrast, if a sample is from the open-set, we expect it to be distant from the MAV of its predicted class and be relatively closer to the MAVs of the k neighbor classes of the predicted class. Based on this intuition, k-LND methods use the distance between a new sample and the MAVs of known classes to identify open-set samples.

We further explain this idea using an example in Fig. 4. Consider a two-class closed-set classifier. The output from the logit layer of the closed-set classifier would be a vector of length two, with *logit 1* and *logit 2*. Fig. 4 demonstrates the space spanned by *logit 1* and *logit 2* where the reddish dotted line shows the decision boundary learned by the classifier, and C_1 and C_2 represent the MAVs of *Class 1* and *Class 2*, respectively. The circle around each MAV represents the cluster around it where a majority of samples (>90%) from its class fall into. If we define r_i as the radius of the cluster around C_i, k-LND$_1$ (the first k-LND variant) assumes that if a sample is predicted as class i by the closed-set classifier and the distance between that sample and C_i is greater than r_i, that sample is from the open-set. For instance, consider that point P and point A in Fig. 4 are both classified as class C_1 by the classifier. If d_{XC_i} defines the distance between a point X and C_i, we see that d_{PC_1} is less than r_1 while d_{AC_1} is greater than r_1. Hence, k-LND$_1$ will label sample P as *Class 1* and sample A as an open-set sample. Similarly if both samples Q and B are classified by the closed-set classifier as *Class 2*, k-LND$_1$ will label sample Q as *Class 2* since d_{QC_2} is less than r_2 and reject sample B as d_{BC_2} is greater than r_2.

k-LND$_1$ only considers the distance to the MAV of the predicted class to identify open-set samples. k-LND$_2$ and k-LND$_3$ improve on the intuition of k-LND$_1$ such that they additionally assume that a sample from a known class would be distant from the MAVs of the neighbors of

its class, in addition to being closer to its own MAV. Accordingly, the three variants of k-LND differ in the way they calculate the distance between a sample and MAVs of closed-set classes. In Eqs. (1), (2), and (3) we show how distances are calculated in k-LND$_1$, k-LND$_2$, and k-LND$_3$ respectively. Here, d_A denotes the Euclidean distance between the sample and the MAV of class A, p denotes the class predicted for the sample by the closed-set classifier and k denotes the number of neighboring closed-set classes considered. The only difference between k-LND$_2$ and k-LND$_3$ is that they use different methods to incorporate the distances with neighboring MAVs into the final distance metric so that to get a lower distance value, a sample needs to be closer to its own MAV and far away from the MAVs of its neighboring classes.

It should be noted that for datasets where the number of closed-set classes is small, k is equal to the number of closed-set classes, and otherwise, k will be less than the total number of closed-set classes and would be considered as a hyperparameter of the method. The reason for this decision is that if the logit layer output is of longer length, euclidean length calculations become less effective [17] and the computation times also increase.

$$D_1 = d_p \tag{1}$$

$$D_2 = \sum_{i=1}^{k} (\, d_i - d_p) \;\; ; i \neq p \tag{2}$$

$$D_3 = \frac{d_p}{\sum_{i=1}^{k} d_i} \; ; i \neq p \tag{3}$$

In Algorithm 1, we describe the common procedure to follow for all three methods prior to inference to calculate the Mean Class Vectors and corresponding threshold for each closed-set class.

Algorithm 1: Before Inference for k-LND

Require: Closed-set classifier without Softmax: θ, Number of closed-set samples N
Input: Set $[X_i^{setA}, Y_i^{setA}]$, with $setA$ as subset A of dataset
Define: $X_{c_i}^{setA}$: the set of correctly classified samples of class c_i from set A by closed-set classifier
Define: $n(X)$: no. of samples in X
Define: $sort(List_A)$: $List_A$ sorted in ascending order
for $c_i = 1.....N$ **do**
  Calculate $MAV_{c_i} = Mean(\theta(X_{c_i}^{train}))$
  $distance_{c_i}$=[]
  for $j = 1.....n(X_{c_i}^{val})$ **do**
    $distance_{c_i}$.append($D_K(X_{c_{ij}}^{val})$)
  $threshold_{c_i} = 90^{th}\ percentile(distance_{c_i})$
return MAV^{c_i} *and* $threshold_{c_i}$

At inference time, a sample will first be fed to the closed-set classifier to get its predicted class and logit layer output. Next, depending on which method is used, the distance values (D_1 or D_2 or D_3) are calculated using its predicted class and MAVs calculated in Algorithm 1. Finally, the calculated distance will be compared with the threshold value of the predicted class from $threshold_{c_i}$, and if the value is less than the threshold value, the predicted label will be accepted and otherwise, it will be rejected as an open-set sample.

3.3. Quantization

The third key component in our framework is model quantization. Quantization in general refers to mapping continuous values to a smaller set of discrete finite values. Usual neural network weights are real values (continuous infinite and represented as 32-bit or 64-bit floating point numbers) and quantization of neural networks maps these model parameters to 8-bit integer values within the range (−128 to 127).

As a result, deep learning models can be adapted to work with integers only (instead of floating point numbers) and the model footprint in terms of storage and memory is decreased. Additionally, inference becomes faster due to integer computations, making such models ideal to be deployed in in-network computing devices.

Since there is some information loss when the real-valued neural network weights are converted to integers, model quantization can cause a drop in performance. In our application, in addition to the drop in closed-set performance, we also need to consider the effect on open-set performance and ensure that the performance drop due to quantization is not significant. While there are many options for model quantization (e.g., quantization-aware training, post-training dynamic range quantization), here we used post-training integer quantization techniques provided by *Tensorflow*[1] to map all values from floating point numbers to `int8` format reducing the model size and perform open-set classification using the quantized model as the underlying classification method.

4. Results

In this section, we explore the efficiency of the proposed framework. We start by introducing the evaluation metrics used and then analyze the open-set performance of the framework when each one of its three key components is added. First, we show how a well-regularized model gives better open-set accuracy and then use the regularized method to evaluate the performance of four existing open-set methods against the three variants of the novel kLND method proposed. Finally, we evaluate the performance of all seven open-set methods (four baseline methods softmax thresholding, background class, OpenMax, and ensemble learning, and the newly proposed methods k-LND$_1$, k-LND$_2$ and k-LND$_3$) with a quantized classifier to show how the proposed framework that combines a well-regularized model with its model weights quantized, with the proposed k-LND method, gives the best open-set performance.

4.1. Evaluation metrics

For performance evaluation, we use *closed-set accuracy* which represents the percentage of closed-set samples correctly classified into relevant classes, and *open-set accuracy* which denotes the percentage of open-set samples correctly identified. Since we need a single metric that can be used to identify the better-performing model, we also calculate the F1_Score. More specifically, we use *Micro F1* score since it is more suited to handling class imbalance caused by the significantly larger size of the open-set compared to the closed-set that reflects the real-world scenario.

When calculating *Micro F1*, we consider only the correctly classified closed-set samples as True Positives (TP) since the classifier is trained only on closed-set data. For True Negatives (TN) and False Positives (FP), the open-set is considered as another class because samples misclassified from or to the open-set class reflect the performance of the classifier. Accordingly, *Micro F1* score is calculated as the given in Eq. (6) using precision and recall calculated according to Eq. (4) and Eq. (5), respectively, where N is the number of known classes.

However, it should be noted that as the open-set is significantly larger compared to a closed-set class, a small drop in the open-set accuracy will have a significantly large effect on the precision compared to that from the closed-set accuracy, thereby reducing the overall F1_Score. As a result, the F1_Score calculated as described above will be biased towards a better open-set accuracy than a closed-set accuracy. A more detailed explanation of the behavior of F1_Score is provided in Appendix C. From here onwards, *Micro F1 score* is referred to as F_Score.

[1] https://www.tensorflow.org/lite/performance/post_training_integer_quant.

Table 2
Closed-set classifier performance.

Model	Accuracy (%)		
	DC	SETA	IoT
Baseline	99.08 ± 0.88	98.17 ± 1.70	97.49 ± 0.48
Regularized	99.82 ± 0.85	98.87 ± 1.77	97.33 ± 0.51

and when comparing two open set classifiers, we consider the classifier with the highest F_Score as the better classifier.

$$Precision_{Micro} = \frac{\sum_{n=1}^{N} TP_i}{\sum_{n=1}^{N} TP_i + FP_i} \tag{4}$$

$$Recall_{Micro} = \frac{\sum_{n=1}^{N} TP_i}{\sum_{n=1}^{N} TP_i + FN_i} \tag{5}$$

$$F\_Score_{Micro} = 2 \times \frac{Precision_{Micro} \times Recall_{Micro}}{Precision_{Micro} + Recall_{Micro}} \tag{6}$$

4.2. Regularized models

In Section 3.1 we explained that for a better open-set classification, the closed-set classifier needs to be well regularized. This is majorly required if the closed-set classifier is trained with a subset of the dataset used by the initial work to propose the original model. To show the effect of regularization, we trained a baseline model (taken from the original work) and a regularized model with high dropout rates for each dataset. Here, note that since AWF and DF datasets have separate open-sets, we do not have to consider data splits and hence the models proposed in the original work are considered the optimized models.

First, we compare the performance of each closed-set classifier where we do not consider the open-set and report the results in Table 2. According to Table 2, we see that for any of the three datasets, the performance of the two models is almost the same in the closed-set setting.

Next, we evaluate the performance of the two models in the open-set setting. Fig. 5 shows the percentage increase in F_score due to regularized model and in Table 3 we report the closed-set and open-set accuracy values for open-set classification when using the two models. Observing Fig. 5, we see that across all three datasets and four open-set methods, the regularized model achieves a higher F_score compared to the baseline model where the gap between the baseline and regularized models lies in the range of 2.99%–35.48%. For the DC dataset, the highest increase in F_score of 20.69% is observed for the ensemble method while the lowest increase of 2.99% is seen for the OpenMax method. Similarly, for the SETA dataset, the highest increase in F_score of 35.48% corresponds to the ensemble method while the lowest increase of 15.38% is observed with OpenMax. For the IoT dataset, the maximum improvement of F_score of 8.33% is observed for OpenMax while the lowest increase of 3.23% is for the ensemble learning.

According to Table 3, in most cases, the regularized model increases the open-set accuracy with a less than 1% decrease in the closed-set accuracy. For the DC dataset, increasing the dropout rates improved the open-set accuracy by a maximum of 21.24% (background class) and a minimum of 4.65% (ensemble). However, OpenMax for DC dataset deviates from the above trend where the closed-set increases by 5.19% while the open-set decreases by 1.14% for the regularized model. Even then, the F_score of OpenMax for the regularized model is still higher. With the SETA dataset, the open-set accuracy increases by a minimum of 4.02% (OpenMax) and a maximum of 38.36% (background class) with less than a 1% drop in the closed-set when the underlying model is regularized. Similarly for the IoT dataset, for a less than 1% drop in the closed-set, the open-set increases by a maximum of 32.7% (background) and a minimum of 4.63% (ensemble) for the regularized model compared to the baseline model. Based on the results across all datasets and methods, we conclude that our hypothesis that suggests regularizing the underlying deep learning model improves the results of open-set classification is accurate.

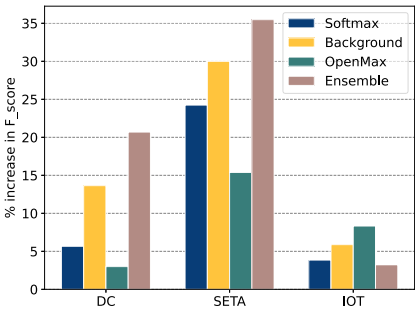


Fig. 5. Effect of regularized model.

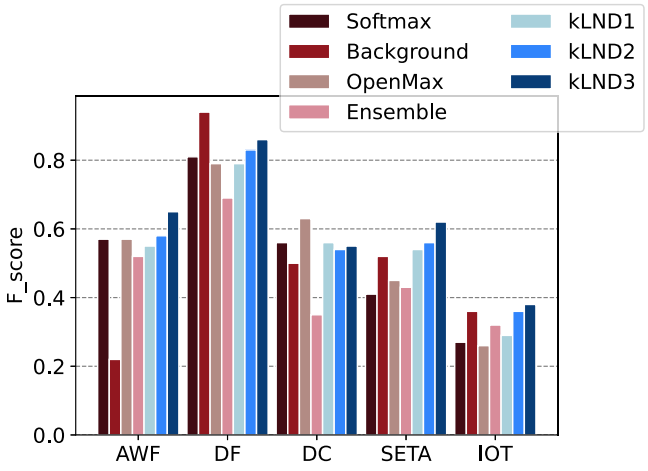


Fig. 6. Default model F_scores.

4.3. k-Logit Neighbor Distance method

We next present the results of evaluating the performance of the three novel open-set methods (k-LND$_1$, k-LND$_2$, k-LND$_3$) we propose in Section 3.2 against four existing methods and report the closed-set and open-set accuracies of each method on all five datasets in Tables 4 and 5. We have also shown the corresponding F_score values in Fig. 6. According to F_score values in Fig. 6, we see that k-LND$_3$ outperforms all other methods for AWF, SETA, and IoT datasets while background class and OpenMax give the best results for DF and DC datasets respectively.

Furthermore, we observe that k-LND$_1$, k-LND$_2$, and k-LND$_3$ methods show consistent results across all the datasets. k-LND$_1$ and k-LND$_2$ maintain >85% closed-set and >65% open-set accuracy while k-LND$_3$ performs the best and maintains >90% closed-set and >70% open-set accuracy regardless of the dataset. In contrast, the performance of other methods fluctuates between datasets. For example, softmax thresholding and OpenMax both have relatively lower open-set accuracy for IoT compared to other datasets. While the background class method performs very well on the DF dataset with >95% in both closed and open-sets, it performs very poorly in the open-set of all other datasets.

In k-LND$_2$ and k-LND$_3$ we calculate the single parameter considering distances to all the class centers (MAV) which embeds the complete information about the output vector placement in the logit space. In other words, they calculate a comparative value from the penultimate layer output and use a threshold to do open-set classification. We attribute the consistently better behavior of novel methods to the fact that in novel methods, the open-set samples are identified by being compared with all closed-set classes in the logit space as opposed to just using a threshold for a single element (maximum value) or comparing with just the predicted class in the logit space.

Table 3
Effect of regularized model.

Dataset	Model	Softmax Thresh.		Background class		OpenMax		Ensemble learning	
		Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc
DC	Base.	90.85 ± 1.9	87.7 ± 3.8	98.70 ± 1.0	36.87 ± 15.9	89.22 ± 2.7	92.16 ± 2.7	84.80 ± 4.0	63.88 ± 22.2
	Reg.	**89.41 ± 2.1**	**89.4 ± 4.1**	**98.69 ± 0.8**	**44.7 ± 11.0**	**93.85 ± 2.1**	**91.11 ± 5.5**	**84.25 ± 4.1**	**66.85 ± 13.3**
SETA	Base.	86.25 ± 4.6	71.59 ± 7.2	96.45 ± 2.4	33.71 ± 11.4	83.75 ± 5.4	79.51 ± 6.1	79.51 ± 1.6	69.16 ± 17.5
	Reg.	**85.41 ± 3.6**	**79.21 ± 9.3**	**95.62 ± 2.7**	**46.64 ± 13.0**	**83.33 ± 4.0**	**82.71 ± 9.8**	**78.8 ± 2.4**	**77.05 ± 19.9**
IOT	Base.	87.55 ± 0.7	59.23 ± 2.8	96.91 ± 0.4	26.21 ± 8.3	87.28 ± 0.7	55.22 ± 6.8	84.78 ± 2.9	68.95 ± 7.3
	Reg.	**87.04 ± 0.6**	**62.35 ± 4.6**	**96.21 ± 0.5**	**34.78 ± 8.6**	**86.86 ± 0.5**	**58.92 ± 8.3**	**84.16 ± 1.5**	**72.14 ± 9.6**

Notes: 'Base.' and 'Reg.' refer to the baseline and regularized models respectively.

Table 4
Open-set method performance — Existing methods.

Dataset	Softmax Thresh.		Background class		OpenMax		Ensemble learning	
	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc
AWF	87.35	88.16	81.60	25.11	87.32	87.93	83.30	85.90
DF	90.69	84.66	**95.20**	**97.40**	88.56	83.99	87.80	69.90
DC	89.41 ± 2.1	89.40 ± 4.1	98.69 ± 0.8	44.70 ± 11.0	**92.15 ± 2.5**	**90.87 ± 5.6**	84.25 ± 4.1	66.85 ± 13.3
SETA	85.41 ± 3.6	79.21 ± 9.3	95.62 ± 2.7	46.64 ± 13.0	83.33 ± 4.0	82.71 ± 9.8	78.80 ± 2.4	77.05 ± 19.9
IOT	87.04 ± 0.6	62.35 ± 4.6	96.21 ± 0.5	34.78 ± 8.1	86.86 ± 0.5	58.92 ± 8.3	84.16 ± 1.5	72.14 ± 9.6

Table 5
Open-set method performance — kLND methods.

Dataset	k-LND$_1$		k-LND$_2$		k-LND$_3$	
	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc
AWF	89.37	85.43	89.88	88.12	**97.98**	**89.23**
DF	88.45	83.99	88.29	88.04	97.84	87.21
DC	91.63 ± 1.7	87.78 ± 6.9	94.24 ± 1.4	86.26 ± 7.1	94.51 ± 1.8	86.92 ± 7.3
SETA	85.41 ± 1.6	84.69 ± 9.8	85.21 ± 1.1	85.16 ± 6.7	**95.42 ± 1.7**	**87.84 ± 9.1**
IOT	85.62 ± 0.6	65.92 ± 5.1	85.49 ± 0.9	76.19 ± 2.4	**97.33 ± 0.5**	**74.47 ± 3.5**

To summarize, all three variants of k-LND method perform consistently across all datasets with the best-performing variant k-LND$_3$ maintaining >95% closed-set and >75% open-set accuracy. It should be noted that all three variants of k-LND are lightweight compared to the background class method that requires open-set samples for training and OpenMax which needs additional EVT modeling. Hence we highlight that k-LND performs well consistently across all datasets while consuming the least resources.

4.4. Quantization

Next, we investigate the effect of quantizing the underlying deep learning model on open-set classification. Fig. 7 shows the F_score values for using the above-mentioned open-set classification methods with an underlying quantized model. Accordingly, we see that across all datasets, the kLND methods perform the best when the underlying model is quantized, with an increase of F_Score compared to the best-performing baseline in the range of 8.9%–77.3%.

In Tables 6 and 7, we present the closed-set and open-set accuracies of the four existing methods and the three novel kLND methods when the weights of the underlying models are quantized. For each dataset, we have indicated the method with the highest F_Score in bold. When using the quantized models for the AWF dataset, only softmax thresholding and the k-LND methods achieve >85% in both closed and open-set accuracies. Out of those four methods, the k-LND$_3$ method records the best performance with 97.98% and 84.02% as closed and open-set accuracy respectively. With the DF dataset, only the novel three methods obtain >85% in both closed-set and open-set accuracy with k-LND$_3$ method achieving the best performance with closed and open-set accuracies of 97.9% and 87.2% respectively, when using the quantized models. For the DC dataset when using the quantized models, softmax thresholding, OpenMax and all three novel methods obtain >85% in both closed-set and open-set accuracy. However, the k-LND

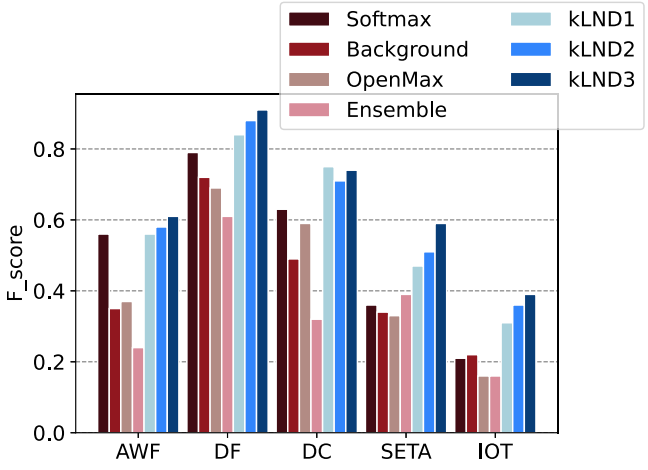


Fig. 7. Quantized model F_scores.

methods record the best F_scores with the k-LND$_1$ reporting the highest F_score of 0.75. After quantizing the models for the SETA dataset, k-LND$_3$ performs the best with 89.6% closed-set and 70.7% open-set accuracy while none of the other methods achieve >85% closed-set accuracy with >70% open-set accuracy. Similarly for the IoT dataset, k-LND$_3$ performs the best with 95.9% closed-set and 76.2% open-set accuracy while none of the other methods achieve >85% closed-set accuracy with >70% open-set accuracy. If we consider the overall result, we observe that k-LND methods perform best on quantized models achieving the highest F_score for all datasets. More specifically, k-LND$_2$ performs best for DC while for all other datasets, k-LND$_3$ performs the best.

Table 6
Quantized model performance — Existing methods.

Dataset	Softmax Thresh.		Background class		OpenMax		Ensemble learning	
	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc
AWF	89.10	86.90	80.20	73.50	81.90	73.60	85.2	31.8
DF	94.40	78.22	92.50	67.45	94.88	61.75	90.20	58.90
DC	93.72 ± 1.9	90.89 ± 5.0	96.20 ± 2.1	44.10 ± 15.3	89.80 ± 4.7	89.84 ± 6.4	77.22 ± 8.8	59.54 ± 9.4
SETA	81.54 ± 11.7	52.17 ± 20.2	78.75 ± 8.9	34.88 ± 12.2	51.55 ± 14.7	77.62 ± 16.3	69.30 ± 14.26	68.48 ± 21.19
IOT	89.18 ± 5.6	46.70 ± 11.2	91.31 ± 3.4	42.50 ± 8.9	78.09 ± 6.2	18.87 ± 9.8	72.53 ± 5.9	21.30 ± 10.1

Table 7
Quantized model performance — kLND methods.

Dataset	k-LND$_1$		k-LND$_2$		k-LND$_3$	
	Closed Acc	Open Acc	Closed Acc	Open Acc	Closed Acc	Open Acc
AWF	89.08	86.89	89.88	88.22	**97.98**	**84.02**
DF	87.72	84.20	87.92	87.80	**97.98**	**87.22**
DC	**87.94 ± 3.9**	**93.71 ± 2.4**	93.18 ± 0.9	88.22 ± 5.9	94.21 ± 1.2	87.92 ± 7.3
SETA	70.62 ± 9.1	68.29 ± 14.4	73.33 ± 8.0	73.77 ± 9.1	**89.58 ± 7.1**	**70.74 ± 11.5**
IOT	83.59 ± 1.2	68.86 ± 6.9	83.56 ± 0.9	77.22 ± 1.9	**95.98 ± 0.5**	**76.17 ± 4.8**

Table 8
Quantized models comparison.

Dataset	Model size		Close set accuracy	
	Original model	Quantized model	Original model	Quantized model
AWF	8.51 MB	2.14 MB	98.09	97.32
DF	8.30 MB	2.08 MB	97.86	97.02
DC	755 kB	189 kB	99.82 ± 0.9	98.69 ± 0.6
SETA	3.41 MB	1.36 MB	98.87 ± 1.8	98.12 ± 1.2
IoT	1.02 MB	261 kB	97.33 ± 0.5	96.87 ± 0.5

Finally, in Table 8, we compare the model sizes before and after being quantized in the closed-set setting. Here *Original model* refers to the default model without model weight quantization. Accordingly, we observe that the quantization of model weights reduces the storage requirement of a model by a minimum of 60% (SETA) and a maximum of 75.01% (IoT).

4.5. Result analysis

In Section 4.4, we showed how the k-LND methods work best when quantizing the underlying deep learning model and we next explore possible reasons for this observation.

As discussed in Section 2.1, the background class method treats the open-set as just another class and uses a subset of open-set samples as a single class during training. Because of this, when quantizing the background class, the samples from the open-set are also considered in the discrete mapping process. Since the training set from the open-set consists of a relatively larger number of samples coming from a large number of different classes, the range of continuous values for elements of input samples increases making the discrete mapping process harder for the quantizer as it now has to map a larger range of continuous values to a fixed smaller range. (This mapping procedure is further discussed in [18,19].). We attribute the degradation of the performance of the background class method with quantizing, to the large error caused by the mapping function as discussed above.

When comparing softmax thresholding, OpenMax, and the three novel methods, the error they encounter at the start from the model output at the logit layer is the same. Softmax thresholding uses this output and performs softmax activation which maps the output vector to another space where the initial error can further propagate. Similarly, the OpenMax method maps the penultimate layer output to another space first using a Weibull distribution before generating the final probability vector which would cause the error to increase further. Ensemble learning methods that aggregate the result from multiple models in a method similar to softmax thresholding can be expected to face the drawback of softmax thresholding. In contrast, in all three k-LND methods, a single variable is calculated based on euclidean distances between penultimate layer outputs. Additionally, k-LND$_2$ and k-LND$_3$ methods consider the relative distance between a sample and its predicted class center vs. centers of multiple other classes, which can have a negation effect on the initial error resulting in a lower effect on the final results.

We further demonstrate this concept with the SETA dataset by calculating the percentage between *error_before* and *error_after* as calculated by Eqs. (7) and (7) respectively, for each method.

Here assume a sample X, deep learning model Θ_O and quantized version of the model Θ_Q, and d is euclidean distance. Also, P and $\overline{P}$ refer to prenultimate layer output of the final probability vector from the open-set classifier respectively. The results are illustrated in Fig. 8.

$$error\_before = d(\Theta_{QP}(X), \Theta_{OP}(X)) \tag{7}$$

$$error\_after = d(\Theta_{Q\overline{P}}(X), \Theta_{O\overline{P}}(X)) \tag{8}$$

According to Fig. 8, the change of error in OpenMax is 105% which shows that the initial error has increased by 105.89% due to Weibull conversion resulting in the high F_score drop in Table 4 for SETA after quantization. Similarly, softmax thresholding also has a positive change of error of 52.6% which can be seen as the reason for the relatively high F_score drop after model quantization. In contrast, the change of error for k-LND$_1$, k-LND$_2$, and k-LND$_3$ are less than zero which shows that it reduces the error in the mapping which results in lesser Micro F1 drops. Accordingly, we can conclude that the three novel open-set classification methods we propose are more compatible with model quantization than other existing methods.

5. Related work

Under related works, we discuss previous work under three topics, traffic fingerprinting, open-set classification, and quantization.

5.1. Traffic fingerprinting

Side-channel information leaks of end-to-end encrypted internet traffic were known for a while. Early work by Chen et al. [1] demonstrated that passively capturing encrypted WiFi traffic from a house allows inferring health conditions and personal income of the occupants. The authors attributed this information leakage to the stateful nature of web communications and low entropies of user inputs. Subsequent work extended these attacks to other application domains. For example, several works showed the possibility of identifying websites

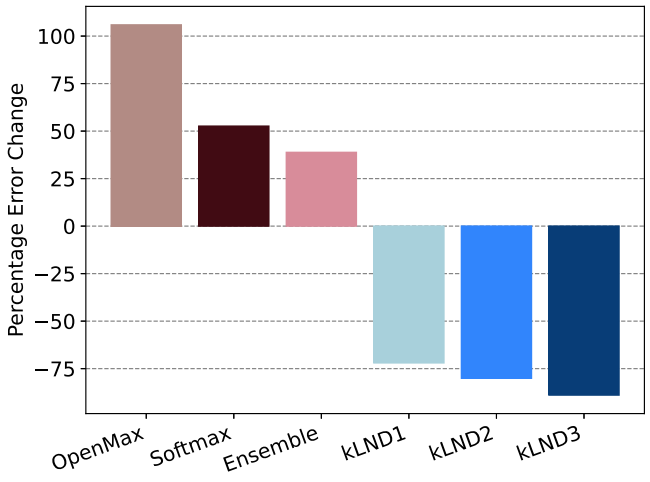


Fig. 8. Percentage of change in error.

visited over Tor [20,21] and HTTPS [22]. Similarly, multiple other works demonstrated the possibility of identifying user activities on messenger apps such as Apple iMessage, Whatsapp, and Telegram that use XMPP over HTTPS [6,23].

Several works investigated the side-channel information leaks in audio and voice of IP applications [24–27]. For instance, Wright et al. [24] showed that due to the variable bit rate codecs used in Skype, the lengths of encrypted VoIP packets can be used to identify the phrases spoken within a call. Zhu and Fu [27] extended the work and showed that the distribution of packet lengths can also be used to identify exact speakers in a VOIP call. It was also shown that it is possible to identify videos from DASH streaming traffic [5,28]. The common approach followed by many of these works is to create a feature vector from the statistical properties of the traffic flows containing features such as packet length, packet direction, and inter-packet times, and then train machine learning models to identify the targeted scenario.

More recent work in traffic fingerprinting [2–4,8,16] used deep learning models, understandably due to their success in other application domains such as computer vision and speech processing. For instance, Payap et al. [2] showed that a basic 1D Convolutional Neural Network (CNN) could identify websites over Tor traffic with over 98% accuracy undermining state-of-the-art defenses. Schuster et al. [16] and Li et al. [4] showed that 1D CNNs can be used to identify exact videos from DASH video streaming traffic. *However, the majority of these works that used either classical machine learning or deep learning, did not have an in-depth look into the more realistic open-set traffic classification problem.*

For instance works such as [4,5] did not consider the open-set classification problem at all. Another set of works considered the open-set problem, but addressed it using naive methods; i.e., *using a background class or separate binary classifier to pre-filter unknowns [2,7,8,16]* or *used classifier confidence [3]. To the best of our knowledge, none of the existing work in traffic fingerprinting has neither explored beyond naive methods for open-set classification nor proposed novel open-set methods for traffic fingerprinting.*

Using side-channel information leaks to infer applications corresponding to encrypted traffic is another related area of research. Similar to traffic fingerprinting, initial app classification attacks were based on traditional machine learning models [29,30] but later moved towards deep learning models [31,32]. Most works in this area did not address the open-set problem while some used the *one-vs-all* approach [30,33]. One key work that specifically addressed the open-set problem is Yang et al. [13]. The authors describe four main approaches, using k-means clustering in the input space, using k-means clustering on the last fully connected layers' output, methods using information from the last layer of deep learning models (softmax thresholding or OpenMax), and gradient-based rejection. While using k-means clustering on the last fully connected layer output follows a similar approach as our proposed method, our method directly calculates class means using correctly classified samples from the training set instead of using k-means clustering. Furthermore, the distance metric used by our proposed method favors shorter distances to a sample's predicted class mean and longer distances to other class means which helps our method identify clusters better and therefore give better performance. In contrast, [13] considers the distance to the predicted class mean only and therefore does not perform as well as our method.

5.2. Open-set classification

As deep learning started making breakthroughs in various fields of machine learning, open-set classification became an interesting yet challenging problem that require attention. The intuitive idea of open-set classification methods is to teach neural networks to know when they do not know [9].

While the open-set classification problem has been explored in the context of traditional machine learning methods such as Support Vector Machines [34,35], much of the recent effort has been on deep neural networks [36]. One of the key works in open-set deep neural networks is OpenMax [9] as described in Section 2. The key idea behind OpenMax and its variants [37,38] is to leverage the fact that the penultimate layer output of deep neural networks is a representation of relationships between classes in the closed-set and open-set samples will have anomalous behaviors. Another body of work [39,40] explored the idea of replacing the traditional cross-entropy loss of deep neural networks with novel losses to move open-set samples to specific areas with respect to the decision boundaries. Another approach is to use the reconstruction-based models [41,42] with the intuition that the model will do an accurate reconstruction of closed-set samples while it will not do a better job in reconstructing open-set samples.

Recently, generative models [36] have been explored in the context of open-set classification. The key idea is to generate known-unknown samples and use them during training to improve the differentiation between the known and unknown classes. For example, [38,43,44] utilized GANs (Generative Adversarial Networks) to generate synthetic samples and then leveraged them to explicitly or implicitly calibrate the learning models. Geng et al. [45] proposed a Hierarchical Dirichlet process (HDP) generative model-based collective decision framework for open-set recognition. This approach used HDP in the inference process and does not depend on threshold seeking because HDP can automatically discover novel classes.

Few recent works tried to address the open-set setting on encrypted traffic fingerprinting. Wang et al. [14] proposed to use ensemble learning as a way of handling open-set traffic classification based on the intuition that combining the outputs from multiple model instances based on a simple base learner would learn different sets of features and hence can help the overall model generalize better towards unknown data as opposed to a single model. Li et al. [46] proposed a framework that combines novel data augmentation and feature extraction methods with self-supervised learning for few-shot open-set traffic classification.

Open-set classification is still an evolving area in machine learning. To the best of our knowledge, most state-of-the-art open-set classification methods have not been tried in traffic classification. Our work is the first to conduct methodical and extensive experiments to compare their performance with more commonly used methods for open classification by the networking community such as using a background class or softmax thresholding.

5.3. Quantization

Quantization is well-known to improve the efficiency of neural networks by reducing the effect of over-parameterization and thereby reducing memory footprint and inference times. While the concept of quantizing neural networks has been discussed since the early 90's [47–49] can be considered as the first key work that directly contributed towards quantization of deep neural networks used today. In [49], Han et al. proposed a three-stage pipeline that first prunes and quantizes a neural network before encoding the model weights using Huffman Coding.

Deep neural network quantization can be broadly categorized into two categories as *Quantization-Aware Training* and *Post-Training Quantization*. Quantization-Aware Training (QAT) retrains a model with quantized weights in order to correct quantization bias. More specifically, forward and backward passes are performed on the quantized model in floating point, but the model parameters are quantized after each gradient update [50–52]. While QAT results in a model with good accuracy, it requires longer training times and the computational cost of re-training the model.

Post-Training Quantization (PTQ) quantizes the model weights without any re-training or fine-tuning of the original model. Therefore, PTQ is very fast and the overhead related to quantization is very low. However, the accuracy of models directly quantized with PQT is lower compared to QAT. Hence, multiple works proposed various approaches to mitigate the accuracy loss of PTO such as bias correction methods [53,54] and equalizing weight ranges [55,56]. Another key work in PTQ, AdaRound [57] proposed an adaptive rounding method to reduce accuracy loss. Hao Wu et al. [58] proposed partial quantization as a solution to minimize the accuracy loss of PTQ where the most sensitive layers are left unquantized.

Although model quantization and its effects have been explored in other domains, to the best of our knowledge, we are the first to apply quantization techniques on traffic fingerprinting DNNs in order to deploy them on in-network computing devices.

6. Conclusion

In this paper, we first hypothesize that using a robust classifier as the underlying deep learning model improves the performance of open-set classifiers. We use five publicly available datasets with model architectures proposed in the original work and show that when the number of classes in the dataset is changed, adjusting the model architecture to avoid overfitting improves the performance of open-set classifiers that use such models. Next, we propose three novel open-set classification methods and compare their performance with four existing methods using five publicly available encrypted traffic fingerprinting datasets. We also show how two most commonly used open-set traffic fingerprinting methods; background class and softmax thresholding do not work in all the datasets. While they perform really well in some datasets they also perform really poorly in other datasets. In contrast, our proposed methods consistently perform well across all datasets. We also show that the proposed methods outperform OpenMax, a popular open-set classifier proposed for computer vision and ensemble learning which was proposed for traffic fingerprinting. Finally, we compare the performance of the proposed method against existing methods when underlying deep learning models are quantized. Overall, we show that our framework which combines a well-regularized closed-set classifier, novel kLND open-set classification method, and model weight quantization, outperforms all other open-set classification methods evaluated across five datasets with a minimum and maximum increase in F_Score of 8.9% and 77.3% respectively. Accordingly, we can conclude that the proposed framework is suited for traffic fingerprinting tasks that need to be carried out in resource-constrained network devices like P4 switches, smart NICs or FPGAs. We note that our results, together with the codes and other artifacts we release,[2] can act as a baseline framework for future work in open-set traffic fingerprinting.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work was conducted in partnership with the Defence Science & Technology Group and Data61/CSIRO, under the Next Generation Technologies Program (C010938).

Appendix A. Naive open-set classification methods

A.1. Background class method

In this section, we explain why background class does not always perform as expected. We use split 5 of the DC dataset to demonstrate this. We use the background class model trained in Section 4.2 and get the output vector from the Softmax layer for the corresponding test set. Next, we plot the 2-dimensional t-SNE graph [59] for the output values as shown in Fig. 9. We note that in this specific split, the classes [0, 1, 3, 5] make up the closed-set while classes [6, 7, 8, 9] make up the open-set. Classes 2 and 4 are used as the known-unknowns.

As can be seen from Fig. 9, the samples from closed-set classes (brownish shades) form four distinct clusters while the samples from the two classes used as known unknowns (greenish shades) are also clustered close together. However, the samples from the open-set classes (blueish shades) which ideally should be clustered closer to known-unknowns (i.e, greenish clusters), can be seen clustered either with a specific closed-set class (i.e, class 9 clustered close to class 3) or as a separate cluster further away from the known-unknown classes. Notice how none of the open-set classes cluster close to the two known-unknown classes (greenish). This confirms how for some datasets, open-set samples would have more similarities with closed-set classes as opposed to known-unknowns. In such cases, the background class method fails.

A.2. Softmax threshold method

Next, we demonstrate why the softmax thresholding method does not always work, using the same split 2 of the SETA dataset. We feed both closed and open-set test sets to the closed-set classifier trained in Section 4.2 and extract the softmax score for the predicted class for each sample. In Fig. 10 we show the histogram of these softmax scores drawn to a log scale. Note that the sub-figure shows the histogram for the entire range of softmax score (0.0–1.0) while the main figure shows the zoomed-in version to clearly emphasize the section of the figure corresponding to the score in the range [0.995–1.0]. The figure shows how almost all closed-set samples have softmax scores greater than 0.999 as expected. However, over 50% of the open-set samples also have softmax scores in the same range. This makes rejecting open-set samples by thresholding on softmax ineffective, especially when sacrificing closed-set accuracy is not acceptable.

[2] https://github.com/ThiliniDahanayaka/Open-Set-Traffic-Classification.

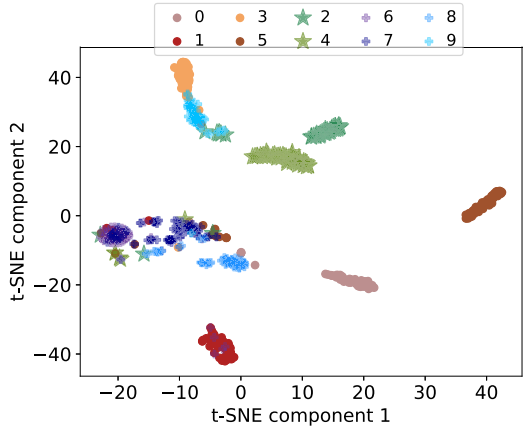


Fig. 9. DC: t-SNE plot for background class method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

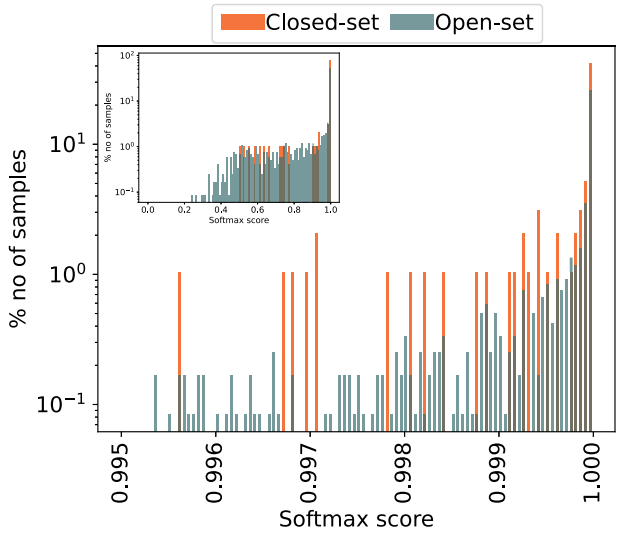


Fig. 10. SETA: Histogram for Softmax scores.

Appendix B. Model architectures

Fig. 11 illustrates the deep CNN model architecture used with the IoT dataset. As discussed in Section 3, dropout rates of the model are increased to regularize a model and in the figure, corresponding dropout layers are shown to the right of the dropout layer. Here, values of the baseline model are mentioned in red while those of the regularized model are mentioned in green. All the other hyperparameter values are the same as in the original work.

B.1. Ensemble model architectures

As proposed in [14], the ensemble model of the AWF dataset is implemented using TensorFlow framework (Fig. 12). Since the DF dataset is also a website fingerprinting dataset much similar to AWF and has a relatively large number of closed-set classes, we used the same architecture with some hyperparameter changes. For DC, SETA, and IoT, the complexity of the model is reduced by reducing the number of repetitive blocks. Fig. 13 shows the ensemble model architecture for the DC dataset and the ensemble model for SETA is similar to that except for a few hyperparameters. The ensemble model for IoT is shown in Fig. 14. When checking the validity of the regularization of DC, SETA, and IoT models, we add an additional dropout layer before the final dense layer in each model. Results are given in Table 3.

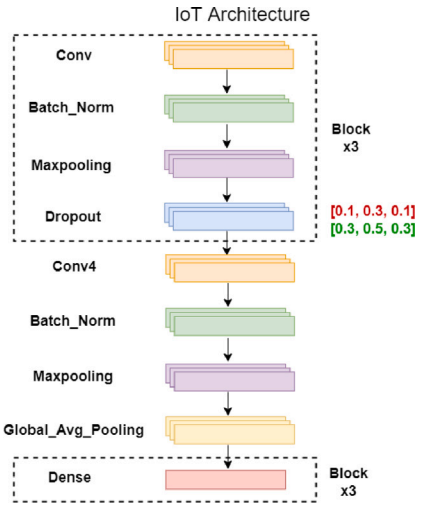


Fig. 11. IoT model architecture. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

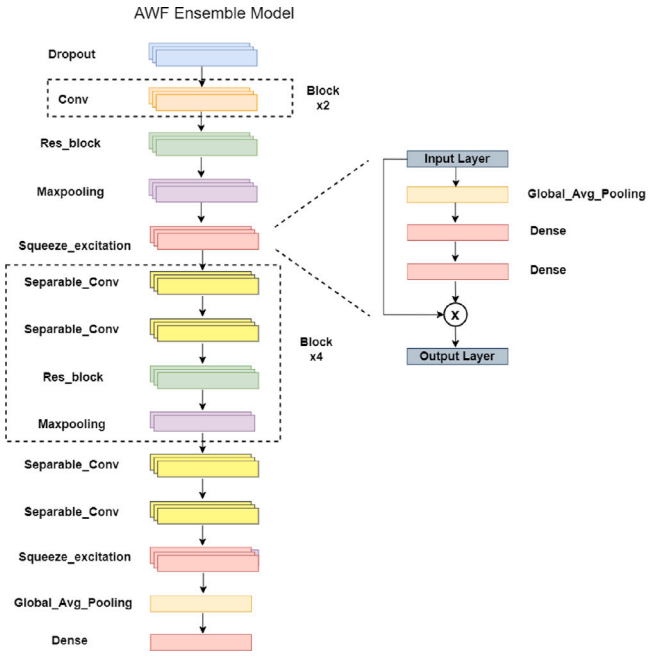


Fig. 12. AWF Ensemble model architecture.

Appendix C. F_Score

In this section, we briefly explain the behavior of F1_Score used as an evaluation metric in our work. Remember that in Section 4.1, we discussed how precision (Eq. (4)) and recall (Eq. (5)) are calculated which are then used to calculate F_Score as given in Eq. (6). Also note that according to Eq. (6), F_Score is the harmonic mean of the precision and recall, and therefore, a drop in either precision or recall values causes a drop in the F_Score.

$$Precision_{Micro} = \frac{\sum_{n=1}^{N} TP_i}{\sum_{n=1}^{N} TP_i + FP_i} \tag{4}$$

$$Recall_{Micro} = \frac{\sum_{n=1}^{N} TP_i}{\sum_{n=1}^{N} TP_i + FN_i} \tag{5}$$

$$F\_Score_{Micro} = 2 \times \frac{Precision_{Micro} \times Recall_{Micro}}{Precision_{Micro} + Recall_{Micro}} \tag{6}$$

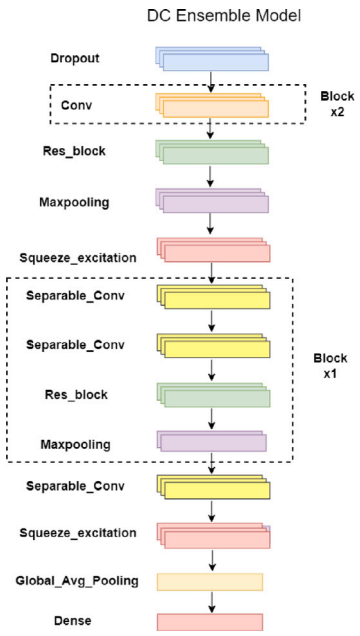


Fig. 13. DC Ensemble model architecture.

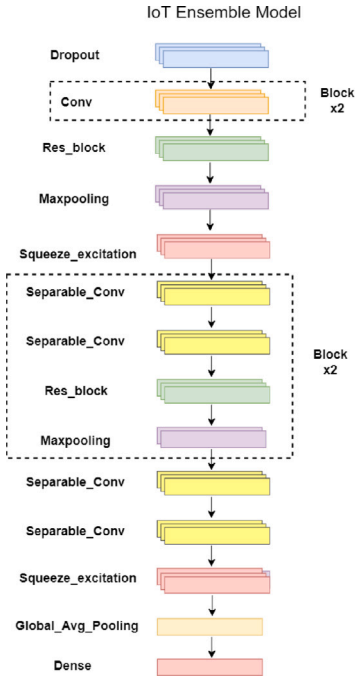


Fig. 14. IoT Ensemble model architecture.

We also discussed in Section 4.1 that we consider the open-set when calculating the False Positives (FP) and True Negatives (TN) only (not considered for false negatives or true positives as the classifier is trained only on closed-set data). Therefore, the open-set accuracy will only affect the precision value as recall which is calculated based on TP and FN has no effect from open-set samples. At the same time, the open-set in our experiments is comparatively larger than the closed-set in accordance with the real world, and therefore, a small drop in the open-set accuracy will significantly increase the number of FPs (open-set samples incorrectly classified as a known class are FPs) thereby lowering the precision and F_Score. We further demonstrate this using a toy example in Table 9.

Table 9
Sample F_Score calculation.

Closed-set acc	Open-set acc	TP	FP	FN	Precision	Recall	F_Score
100.00	100.0	200	0	0	1	1	1
95.00	97.0	190	28	10	0.870	0.950	0.90
97.50	95.0	195	43	5	0.820	0.975	0.89
70.00	90.0	140	113	30	0.550	0.820	0.66
90.00	68.0	180	262	20	0.410	0.900	0.56

In Table 9, we present five example scenarios with varying TP, FP, FN values and corresponding closed-set and open-set accuracy values, with the F_Score calculated as explained above.

According to Table 9

- Row 1: All closed-set and open-set samples are correctly classified and hence F_Score is 1.
- Row 2: >95% As the open-set accuracy is high, FPs are very low, resulting in >0.85 precision and 0.9 F_Score.
- Row 3: Open-set accuracy is 2% lower than the row above while the closed-set accuracy is 2% higher than the row above.
  - The 2% drop in open-set accuracy compared to the row above resulted in a 53.57% increase in the FPs (As the open-set is large, a large number of samples needs to be misclassified to change the open-set accuracy even by a small amount.).
  - The 2% increase in closed-set accuracy compared to the row above resulted in only a 2.63% increase in TPs (As the closed-set is relatively smaller, only a small number of samples needs to be correctly classified to change the open-set accuracy by a small amount.)

  Even though the increase in closed-set accuracy was the same as the decrease in open-set accuracy, overall F_Score dropped by 1.12%.
- Row 4 and 5: Open-set accuracy of row 5 is 22% lower than that of row 4 while the closed-set accuracy is 20% higher than row 4.
  - The 22% drop in open-set accuracy resulted in a 131.85% increase in the FPs.
  - The 20% increase in closed-set accuracy resulted in only a 28.57% increase in TPs.

  Even though the increase in closed-set accuracy was similar to the decrease in open-set accuracy, overall F_Score dropped by 15.15%.

Accordingly, we can see that a higher open-set accuracy always results in a higher F1_Score due to the class imbalance between the closed-set and the open-set. It should be noted that this class imbalance reflects the real-world scenario where the unknown open-set is much larger compared to the known closed-set.

References

[1] S. Chen, R. Wang, X. Wang, K. Zhang, Side-channel leaks in web applications: A reality today, a challenge tomorrow, in: 2010 IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 191–206.

[2] P. Sirinam, M. Imani, M. Juarez, 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.

[3] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, W. Joosen, Automated website fingerprinting through deep learning, in: 25th NDSS, 2018.

[4] Y. Li, Y. Huang, R. Xu, S. Seneviratne, K. Thilakarathna, A. Cheng, D. Webb, 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.

[5] K.N. Choi, A. Wijesinghe, C.M.M. Kattadige, K. Thilakarathna, S. Seneviratne, 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.

[6] S.E. Coull, K.P. Dyer, Traffic analysis of encrypted messaging services: Apple imessage and beyond, ACM SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 5–11.

[7] C. Wang, S. Kennedy, H. Li, K. Hudson, G. Atluri, X. Wei, W. Sun, B. Wang, Fingerprinting encrypted voice traffic on smart speakers with deep learning, in: Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2020, pp. 254–265.

[8] P. Sirinam, N. Mathews, M.S. Rahman, M. Wright, Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1131–1148.

[9] A. Bendale, T.E. Boult, Towards open set deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1563–1572.

[10] Open Networking Foundation, P4 open source programming language, 2023, URL https://p4.org/.

[11] M. Yang, A. Baban, V. Kugel, J. Libby, S. Mackie, S.S.R. Kananda, C.-H. Wu, M. Ghobadi, Using trio: Juniper networks' programmable chipset - for emerging in-network applications, in: Proceedings of the ACM SIGCOMM 2022 Conference, SIGCOMM '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 633–648, http://dx.doi.org/10.1145/3544216.3544262.

[12] D. Webb, Applying softmax classifiers to open set, in: T.D. Le, K.-L. Ong, Y. Zhao, W.H. Jin, S. Wong, L. Liu, G. Williams (Eds.), Data Mining, Springer Singapore, Singapore, 2019, pp. 104–115.

[13] L. Yang, A. Finamore, F. Jun, D. Rossi, Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4103–4118.

[14] Y. Wang, H. Xu, Z. Guo, Z. Qin, K. Ren, SnWF: Website fingerprinting attack by ensembling the snapshot of deep learning, IEEE Trans. Inf. Forensics Secur. 17 (2022) 1214–1226.

[15] T. Dahanayaka, G. Jourjon, S. Seneviratne, Understanding traffic fingerprinting CNNs, in: 2020 IEEE 45th Conference on Local Computer Networks (LCN), IEEE, 2020, pp. 65–76.

[16] R. Schuster, V. Shmatikov, E. Tromer, Beauty and the burst: Remote identification of encrypted video streams, in: 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 1357–1374.

[17] C.C. Aggarwal, A. Hinneburg, D.A. Keim, On the surprising behavior of distance metrics in high dimensional space, in: J. Van den Bussche, V. Vianu (Eds.), Database Theory — ICDT 2001, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 420–434.

[18] K. Onishi, M. Hashimoto, et al., Memory efficient training using lookup-table-based quantization for neural network, in: 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), IEEE, 2020, pp. 251–255.

[19] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2704–2713.

[20] A. Panchenko, L. Niessen, A. Zinnen, T. Engel, Website fingerprinting in onion routing based anonymization networks, in: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, 2011, pp. 103–114.

[21] T. Wang, I. Goldberg, Improved website fingerprinting on tor, in: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, 2013, pp. 201–212.

[22] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, K. Wehrle, Website fingerprinting at internet scale, in: NDSS, 2016.

[23] K. Park, H. Kim, Encryption is not enough: Inferring user activities on KakaoTalk with traffic analysis, in: International Workshop on Information Security Applications, Springer, 2015, pp. 254–265.

[24] C. Wright, et al., Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations, in: 2008 IEEE S&P, 2008.

[25] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, G.M. Masson, Uncovering spoken phrases in encrypted voice over IP conversations, ACM Trans. Inf. Syst. Secur. 13 (4) (2010) 35.

[26] A.M. White, A.R. Matthews, K.Z. Snow, F. Monrose, Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks, in: 2011 IEEE Symposium on Security and Privacy, IEEE, 2011, pp. 3–18.

[27] Y. Zhu, H. Fu, Traffic analysis attacks on skype VoIP calls, Comput. Commun. 34 (10) (2011) 1202–1212.

[28] A. Reed, 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, 2017, pp. 361–368.

[29] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407–414.

[30] V.F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2016, pp. 439–454.

[31] W. Wang, M. Zhu, J. Wang, X. Zeng, 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.

[32] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, M. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, Soft Comput. 24 (3) (2020) 1999–2012.

[33] T. Shapira, Y. Shavitt, Flowpic: Encrypted internet traffic classification is as easy as image recognition, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019, pp. 680–687.

[34] W.J. Scheirer, A. de Rezende Rocha, A. Sapkota, T.E. Boult, Toward open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (7) (2012) 1757–1772.

[35] M.D. Scherreik, B.D. Rigling, Open set recognition for automatic target classification with rejection, IEEE Trans. Aerosp. Electron. Syst. 52 (2) (2016) 632–642.

[36] C. Geng, S.-j. Huang, S. Chen, Recent advances in open set recognition: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (10) (2020) 3614–3631.

[37] S. Prakhya, V. Venkataram, J. Kalita, Open set text classification using CNNs, in: Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017), NLP Association of India, Kolkata, India, 2017, pp. 466–475, URL https://aclanthology.org/W17-7557.

[38] Z. Ge, S. Demyanov, Z. Chen, R. Garnavi, Generative OpenMax for multi-class open set classification, in: British Machine Vision Conference 2017, British Machine Vision Association and Society for Pattern Recognition, 2017.

[39] D. Miller, N. Sunderhauf, M. Milford, F. Dayoub, Class anchor clustering: A loss for distance-based open set recognition, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 3570–3578.

[40] A.R. Dhamija, M. Günther, T. Boult, Reducing network agnostophobia, Adv. Neural Inf. Process. Syst. 31 (2018).

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

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

[43] I. Jo, J. Kim, H. Kang, Y.-D. Kim, S. Choi, Open set recognition by regularising classifier with fake data generated by generative adversarial networks, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 2686–2690.

[44] K. Lee, H. Lee, K. Lee, J. Shin, Training confidence-calibrated classifiers for detecting out-of-distribution samples, in: International Conference on Learning Representations, 2018.

[45] C. Geng, S. Chen, Collective decision for open set recognition, IEEE Trans. Knowl. Data Eng. 34 (1) (2020) 192–204.

[46] J. Li, C. Gu, L. Luan, F. Wei, W. Liu, Few-shot open-set traffic classification based on self-supervised learning, in: 2022 IEEE 47th Conference on Local Computer Networks (LCN), IEEE, 2022, pp. 371–374.

[47] E. Fiesler, A. Choudry, H.J. Caulfield, Weight discretization paradigm for optical neural networks, in: Optical Interconnections and Networks, Vol. 1281, SPIE, 1990, pp. 164–173.

[48] W. Balzer, M. Takahashi, J. Ohta, K. Kyuma, Weight quantization in Boltzmann machines, Neural Netw. 4 (3) (1991) 405–409.

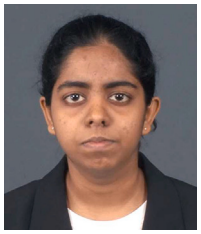
[49] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016, URL http://arxiv.org/abs/1510.00149.

[50] J. Choi, Z. Wang, S. Venkataramani, P.I.-J. Chuang, V. Srinivasan, K. Gopalakrishnan, Pact: Parameterized clipping activation for quantized neural networks, 2018, arXiv preprint arXiv:1805.06085.

[51] B. Zhuang, C. Shen, M. Tan, L. Liu, I. Reid, Towards effective low-bitwidth convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7920–7928.

[52] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, A. Joulin, Training with quantization noise for extreme model compression, 2020, arXiv preprint arXiv:2004.07320.

[53] B. Ron, N. Yury, H. Elad, et al., Post training 4-bit quantization of convolution networks for rapid-deployment, in: Advances in Neural Information Processing Systems, Vancouver, Canada, 2019, pp. 7948–7956.
[54] A. Finkelstein, U. Almog, M. Grobman, Fighting quantization bias with bias, 2019, arXiv preprint arXiv:1906.03193.
[55] E. Meller, A. Finkelstein, U. Almog, M. Grobman, Same, same but different: Recovering neural network quantization error through weight factorization, in: International Conference on Machine Learning, PMLR, 2019, pp. 4486–4495.
[56] M. Nagel, M.v. Baalen, T. Blankevoort, M. Welling, Data-free quantization through weight equalization and bias correction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1325–1334.
[57] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, D. Soudry, Improving post training neural quantization: Layer-wise calibration and integer programming, 2020, arXiv preprint arXiv:2006.10518.
[58] H. Wu, P. Judd, X. Zhang, M. Isaev, P. Micikevicius, Integer quantization for deep learning inference: Principles and empirical evaluation. arxiv 2020, 2004, arXiv preprint arXiv:2004.09602.
[59] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).



Thilini Dahanayaka is currently working as a research associate at the School of Computer Science, University of Sydney where she completed her Ph.D degree in early 2023. She received her B.Sc Eng degree specialized in Computer Engineering from University of Peradeniya, Sri Lanka in 2017. Her major research interest is in Cyber Security with a main focus on machine learning related threats on encrypted traffic flows.

Yasod Ginige is currently following his bachelors, specialized in Electronics and Telecommunication engineering at the University of Moratuwa. Also, he works as a visiting instructor at the department of Electronics and Telecommunication Engineering, University of Moratuwa. His major research interests are applications of machine learning in Cyber security and IoT related areas.

Yi Huang is currently a PhD student in the School of Computer Science at University of Technology Sydney (UTS). She received her B.S. degree in Communication Engineering from University of Electronic Science and Technology of China (UESTC) in 2005. She received her M.S. degree in Communication and Information System from Xi'an Jiaotong University (XJTU) in 2008. Her research interests include classification, anomaly detection and few-shot learning techniques and the application of deep learning to network traffic analysis.

Guillaume Jourjon is senior researcher at Data61-CSIRO. He received his Ph.D. from the University of New South Wales and the Toulouse University of Science in 2008. Prior to his Ph.D., he received a Engineer Degree from the ENSICA. He also received a DEUG in Physics and Chemistry (Major) and Mathematic (Minor) from the University of Toulouse III. His research areas of interest are related to Distributed Computing, Software Defined Network, in-Network Computing, and Security and Privacy of Networked Systems.

Suranga Seneviratne is a Lecturer in Security at the School of Computer Science, The University of Sydney. He received his Ph.D. from University of New South Wales, Australia in 2015. His current research interests include privacy and security in mobile systems, AI applications in security, and behavior biometrics. Before moving into research, he worked nearly six years in the telecommunications industry in core network planning and operations. He received his bachelor degree from University of Moratuwa, Sri Lanka in 2005.