Encrypted Mobile Traffic Classification with a Few-shot Incremental Learning Approach

Yongming Chen*[1], Yuzhou Tong*[1], Gwee Bah Hwee[1], Qi Cao[2], Sirajudeen Gulam Razul[3], Zhiping Lin[1]
[1]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*
[2]*School of Computing Science, University of Glasgow Singapore, Singapore*
[3]*Temasek Laboratories, Nanyang Technological University, Singapore*
{yongming001, ytong004}@e.ntu.edu.sg, {ebhgwee, esirajudeen, ezplin}@ntu.edu.sg, qi.cao@glasgow.ac.uk

***Abstract*—Mobile traffic classification is an essential task for network security and management. Even though some progress has been made, the existing methods have limitations regarding plasticity and the requirement for large amounts of labeled data for training. In real-world wireless networks, new applications are constantly emerging. The lack of plasticity means the model must be retrained entirely whenever a larger dataset with new classes is obtained, which is time-consuming. Furthermore, obtaining large amounts of labeled data is often complicated and expensive. To overcome these limitations, we proposed a novel approach for classifying encrypted mobile traffic using the few-shot incremental learning with a Long Short-Term Memory (LSTM) model. We pre-train an LSTM model with a base dataset, then incrementally add classes and update the model with few-shot datasets. We leverage the exemplar selection and knowledge distillation to keep the stability and plasticity of the model. We validate our method by collecting Downlink Control Information (DCI) of twenty different mobile applications from commercial Long Term Evolution (LTE) networks. Our experimental results demonstrate the effectiveness of the proposed method.**

***Index Terms*—Encrypted traffic classification, Incremental learning, Few-shot dataset, LTE networks traffic.**

I. Introduction

Mobile traffic classification is an essential task in modern network management and security. With the proliferation of mobile devices and the ever-increasing volume of traffic generated by them, accurately classifying mobile traffic has become an essential challenge for network operators and security analysts [1]. In recent years, many techniques have been reported for mobile traffic classification leveraging UDP/TCP port analysis or packet-level information [2]–[4]. However, due to the encryption in Long Term Evolution (LTE) and 5G networks, there is no passive method to acquire the traffic data from the network layer or transport layer. This encryption mechanism makes traffic classification in LTE and 5G networks a challenge.

With the assistance of the LTE downlink decoding software (e.g., SRS Airscope [5] or OWL [6]), analyzing the Downlink Control Information (DCI) becomes possible. In LTE, the Physical Downlink Control Channel (PDCCH) is used to transmit the DCI message from the base station to the user equipment (UE). The DCI message includes various control information elements, such as the modulation and coding scheme, resource allocation, power control commands, and other parameters required for efficient data transmission. With the user-specific traffic information from DCI, it enables traffic classification by a passive sniffer. The literature on this topic has explored various methods and techniques for using DCI to classify network traffic. A model based on recurrent neural network (RNN) was presented in [7] to recognize traffic service types by decoding DCI. Trinh *et al.* introduced the method to recognize not only traffic services but also the usage type of applications (apps) in both supervised and unsupervised ways based on machine learning [8]. In [9], it presents a method utilizing data link layer information to accurately recognize 20 different types of apps on both iOS and Android mobile phone platforms. However, the effectiveness of these methods depends on both training and evaluation samples coming from a fixed dataset with a large number of samples.

As the evolving of mobile networks, new applications are constantly emerging. To correctly recognize these new applications, the multi-class classifier must be entirely retrained on the expanded dataset. Retraining the entire multi-class classifier is time-consuming, especially for large datasets. Therefore, a key issue in encrypted traffic classification is the ability to incrementally add new applications to the classification system without retraining the entire system from scratch. Unfortunately, fine-tuning the classifier is not an option, as it leads to the catastrophic forgetting problem [10]. With incremental learning approaches growing mature gradually, some progress has been made on this topic. In [11], an incremental Support Vector Machines approach is deployed to traffic classification. An incremental learning framework is reported that employs neural network classifiers [12]. This work is based on a dataset [13] collected by monitoring Domain Name System (DNS) traces from the application layer. A method proposed in [14] employs incremental learning based on generative replay for encrypted traffic classification. It uses Generative Adversarial Networks to select representative samples and generate high-quality samples. Although all these works demonstrate good performance, they train and evaluate the models on a large labeled dataset. However, getting a large dataset is complicated and expensive. Besides, they are all based on network layer information that can not be acquired passively in LTE/4G and 5G networks.

In this paper, we present a novel framework for mobile

*These authors contributed equally to this work.

encrypted traffic classification based on a few-shot incremental learning approach to address the constraints mentioned above. We briefly summarize our contributions as follows:

- We propose an incremental learning framework that utilizes rehearsal and knowledge distillation techniques in combination with the Long Short-Term Memory (LSTM) model [15] for mobile encrypted traffic classification.
- We adopt an exemplar selection algorithm to identify representative samples from the training data of the current training session. This allows us to effectively control the dataset's size after adding new applications to the system. Meanwhile, this mechanism can avoid the catastrophic forgetting problem effectively.
- We collect a real-world LTE traffic dataset consisting of 20 different mobile applications to evaluate our proposed model. After learning new applications, the experimental results show that our approach maintains ideal classification accuracy.

The rest of this paper is organized as follows. Section II introduces the details of the incremental learning framework. Section III introduces the evaluation and presents the experimental results. Section IV concludes this paper.

II. Methodology

A. Problem Setting

The few-shot class-incremental learning (FSCIL) setting can be defined as follows. The entire learning process is divided into *t* training sessions, and we consider a sequence of labeled training sets $\{D(1), D(2), ..., D(t)\}$, where $D(t) = \{\mathbf{x}_i, y_i\}_{i=1}^{|D(t)|}$. $L(t)$ denotes the classes included in $D(t)$, then $\forall i, j,\ L(i) \cap L(j) = \varnothing$. $D(1)$ is a large-scale training set of base classes, and $D(t{>}1)$ is the few-shot training set of new classes. The model Θ is trained sequentially on $\{D(1), D(2), ..., D(t)\}$ with adaptable feature extractor and classifier, while only $D(t)$ is available at training session *t*. Once trained on $D(t)$, the model Θ is tested to recognize all the encountered classes up to session *t*. The main challenge in FSCIL is to avoid catastrophic forgetting of old classes. According to [16], there are mainly two strategies for addressing catastrophic forgetting: 1) by freezing parts of the network weights, 2) by rehearsal, i.e., continuously stimulating the network with the earlier data.

For our model, we adopt the rehearsal mechanism to update the model parameters. We use the training data from the currently available classes and the stored exemplars from earlier classes. Additionally, our model utilizes knowledge distillation [17] to prevent the deterioration of information in the network over time.

B. Model

In this paper, $D(1)$ is the traffic series of base application classes, and $D(t{>}1)$ is the traffic series of newly added application classes. Since we aim to adapt the traffic classification system efficiently and effectively, the new applications only have very limited training samples. Therefore, the problem nature requires a few-shot incremental learning system. The system should assimilate new knowledge with minimum effort while applying previously learned knowledge in future updates. This enables the updated classification system to process new applications while retaining its ability to classify existing ones.

As illustrated in Figure 1, the traffic classification system consisting of an encoder for preprocessing the traffic series, a feature extractor based on the LSTM network (f^t_{LSTM}), and a fully connected layer as a classifier (f^t_{FC}). Therefore, the overall objective predictive function at training session t is defined in Eq. (1):

$$f^t(.) = f^t_{FC}(f^t_{LSTM}(.)) \tag{1}$$

For an input sequence, $S = \{s_1, s_2, s_3, ...s_T\}$, the forward process of an LSTM cell at time t is shown in Eq. (2) - (7).

$$i_t = \sigma(W_{is}s_t + W_{ih}h_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma(W_{fs}s_t + W_{fh}h_{t-1} + b_f) \tag{3}$$

$$o_t = \sigma(W_{os}s_t + W_{oh}h_{t-1} + b_o) \tag{4}$$

$$\tilde{c}_t = \tanh(W_{cs}s_t + W_{ch}h_{t-1} + b_c) \tag{5}$$

$$m_t = f_t \odot m_{t-1} + i_t \odot \tilde{c}_t \tag{6}$$

$$h_t = o_t \odot \tanh(m_t) \tag{7}$$

where i_t, f_t, and o_t represent the input gate, forget gate, and output gate, respectively. $\tilde{c}_t$ is the intermediate state; m_t is a memory cell and h_t is the hidden state. $tanh(.)$ and $\sigma(.)$ are activation functions while $\odot$ denotes point-wise multiplication.

When the system obtains a new dataset, it updates the feature extractor and the classifier. It is followed by constructing an exemplar set from the training dataset for the last session. And the overall training process is as follows.

- Before training, we use an encoder to compress the raw traffic series to a consistent length. The resulting encoded data is then fed into the deep learning model. The detail of the encoder is introduced in Section III.
- At the first training session, we train the feature extractor and the classifier using the encoded traffic series from the base training set $D(1)$ including *n* different types of applications. Each application has l_{base} training samples. Once completed, we get the $f^1(.) = f^1_{FC}(f^1_{LSTM}(.))$.
- At the *t*-th ($t > 1$) training session, only the current dataset $D(t)$ is available. $D(t)$ includes *m* different types of applications and each application has l_{inc} training samples. We combine $D(t)$ and the stored exemplars to form an augmented training set. We inherit the feature extractor from the previous session while extending the classifier by *m* cells for classifying new applications. The weights of the extended cells are randomly initialized. Then we train the model with $D(t)$ to get the $f^t(.) = f^t_{FC}(f^t_{LSTM}(.))$.

After each training session *t*, the model is tested with a dataset consisting of all the encountered applications, i.e., totally

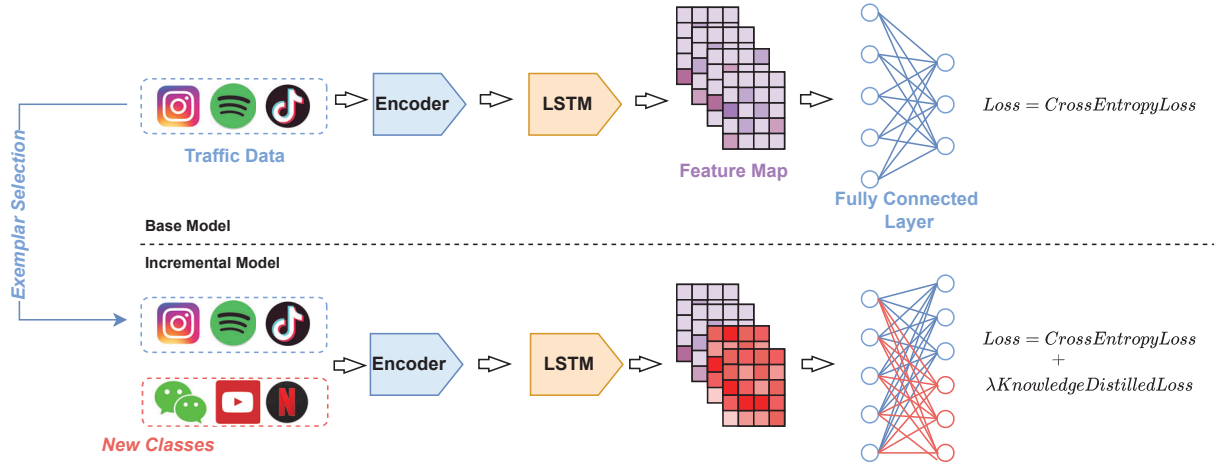


Fig. 1. Few-shot incremental learning model architecture. For the base model, we train the LSTM model with the base training dataset. After that, some exemplar samples are selected from the base dataset. The incremental model is adapted from the base model and trained with the new classes and the selected exemplar samples. Meanwhile, a new loss function considering knowledge distilled is used to train the incremental model.

$n + (t-1)m$ types of applications at the session t. And each application has l_{test} test samples. For classification, the softmax function is used to get the probability of each class i shown in Eq. (8).

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n+(t-1)m} e^{z_j}} \tag{8}$$

where $\boldsymbol{z} = (z_1, z_2, \ldots, z_{n+(t-1)m})^T$ is the output vector of the fully connected layer and $n + (t-1)m$ is the number of the classes at session t.

C. Exemplar Selection

In order to prevent catastrophic forgetting, the system stores K training samples from each class to form an exemplar set after each training session. We should only select the most representative samples to better represent the class feature. We decompose the training set $D(t)$ into $|L(t)|$ subsets $\boldsymbol{X}^{(c)}$ with l samples for each class c. Then, we use f^t_{LSTM} to extract the feature vector $f^t_{LSTM}(\mathbf{x}_i^{(c)}) = \phi(\mathbf{x}_i^{(c)})$ of each traffic series $\mathbf{x}_i^{(c)}$ from $\boldsymbol{X}^{(c)}$ and calculate the mean feature vector $\boldsymbol{\mu}^{(c)}$ of the class. To form a representative exemplar set, we aim to minimize the Euclidean distance between the mean feature vectors of the exemplar set $P^{(c)}$ and the class c. A greedy strategy is applied to achieve this goal, where the detailed procedure is described in Algorithm 1, according to [16].

Algorithm 1 Exemplar Selcetion

Inputs: Training set $D(t)$
Inputs: K exemplars per class
Require: feature extractor $\phi(.)$
Outputs: Exemplars set $\mathcal{P}$
Divide $D(t)$ into $\boldsymbol{X}^{(c)}, c = 1, \ldots, |L(t)|$
for $c = 1, ..., |L(t)|$ **do**
  $\boldsymbol{\mu}^{(c)} = \frac{1}{C} \sum_{x_i \in \boldsymbol{X}^{(c)}} \phi(\mathbf{x}_i^{(c)})$
  for $k = 1, ..., K$ **do**
    $\boldsymbol{p_k} \leftarrow \arg\min_{x_i \in \boldsymbol{X}^{(c)}} \|\boldsymbol{\mu}^{(c)} - \frac{1}{k}(\phi(\mathbf{x}_i^{(c)}) + \sum_{j=1}^{l-1} \phi(\boldsymbol{p_j}))\|$
  end for
  $P^{(c)} = \{\boldsymbol{p_k}\}_{k=1}^{K}$
  $\mathcal{P} = \mathcal{P} \cup P^{(c)}$
end for

D. Knowledge Distillation

Although we have chosen exemplars to augment the supervision information for the old class, we still suffer from catastrophic forgetting when the number of optional exemplars is very limited. Therefore, we require additional supervision information to assist the model in preserving the representative capacity of the old classes. Knowledge distillation is a widely used method that transfers key knowledge from a teacher model to a student model. In our case, the model from the previous session serves as the teacher model, while the current model is a student model. The overall loss is a combination of the cross-entropy loss $\mathcal{L}_{CE}$ with the knowledge distillation loss $\mathcal{L}_{KD}$. The overall loss is given as follows:

$$\mathcal{L}(\mathbf{x}, y) = L_{CE}(\mathbf{x}, y) + \lambda L_{KD}(\mathbf{x}, y) \tag{9}$$

where λ is a hyper-parameter governing the effect of knowledge distillation. According to [18], λ is set to $\frac{L_{old}}{L_{old}+m}$, where L_{old} is the number of classes learned in the previous session. The cross-entropy is given as:

$$\mathcal{L}_{CE}(\mathbf{x}, y) = \sum_{c=1}^{L_{old}+m} -\mathbb{1}(c = y)\log(p_c(\mathbf{x})), \tag{10}$$

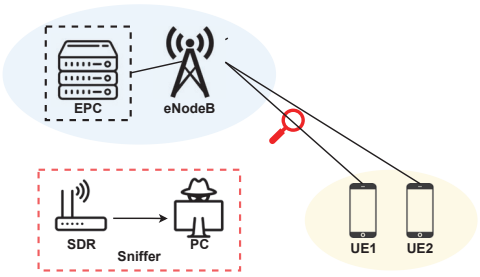


Fig. 2. LTE Network Infrastructure and the position of the deployed traffic monitor

TABLE I
NAMES AND CATEGORIES OF THE APPS FOR EXPERIMENT

Traffic Service Type	App Name
Streaming Video	Tik Tok
	YouTube
	Amazaon Prime Video
	Netflix
Streaming Music	Spotify
	YouTube Music
Social Media	Instagram
	Xiaohongshu
Text Chat	WhatsApp
	WeChat
	Telegram
Video Calls	WhatsApp
	WeChat
	Telegram
Game	Genshin
	Pubg Mobile
Shopping	Shopee
	Lazada
Travel&Local	Grab
Tools	Google

where $p_c(\mathbf{x})$ is the output probability of the sigmoid function for the c-th class. And distillation loss is defined as follows.

$$\mathcal{L}_{KD}(\mathbf{x}) \triangleq \begin{cases} \sum_{c=1}^{L_{old}} -\hat{q}_c(\mathbf{x}) \log(q_c(\mathbf{x})), & t > 1 \\ 0, & t = 1 \end{cases} \tag{11}$$

where $L_{old} = n + (t-2)m$ is the number of known classes at session t, and $\hat{q}_c(\mathbf{x}) = \frac{e^{\hat{o}_c(\mathbf{x})/T}}{\sum_{c=1}^{L_{old}} e^{\hat{o}_j(\mathbf{x})/T}}$; $\hat{o}_c(\mathbf{x})$ is an element of $\hat{\boldsymbol{o}}(\mathbf{x}) = (\hat{o}_1(\mathbf{x}), \ldots, \hat{o}_{L_{old}}(\mathbf{x}))^T$ representing the output logits of the teacher model; $o_c(\mathbf{x})$ is an element of $\boldsymbol{o}(\mathbf{x}) = (o_1(\mathbf{x}), \ldots, o_{L_{old}}(\mathbf{x}))^T$ representing the output logits of the student model. So, $q_c(\mathbf{x}) = \frac{e^{o_c(\mathbf{x})/T}}{\sum_{c=1}^{L_{old}} e^{o_j(\mathbf{x})/T}}$.

III. Experiment and Analysis

A. Data acquisition and encoder

To collect data for our experiment, we connected our mobile phones to a commercial LTE network and launched the downlink traffic monitor. This allowed us to collect broadcast information from the base station connected to user devices. To achieve this, we used the SDR device USRP X310, equipped with SRS AirScope, a downlink sniffer software that enabled us to capture all DCI messages and MAC layer packets, including the Radio Resource Control (RRC) connection setup message. The position of the traffic monitor was between the eNodeB and UEs, as shown in Fig. 2.

In order to monitor the traffic series created by the experiment phone, we need to identify the phone first. LTE networks utilize several identifiers, with the International Mobile Subscriber Identity (IMSI) being a unique identifier used globally to identify a SIM card. However, due to the high sensitivity of the IMSI, the network allocates a Temporary Mobile Subscriber Identifier (TMSI) to the subscriber when they first access the network. During the setup of the RRC connection, the eNodeB transmits the TMSI without encryption, making it possible to obtain the TMSI [19]. Although the TMSI is refreshed periodically, it has a longer lifetime than the Cell Radio Network Temporary Identifier (C-RNTI). As a result, this allows us to track a specific user for a long period.

To determine the initial RNTI of the mobile phone under experimentation, we first uploaded a large file to create a burst in uplink data rate. Then, we instructed the mobile phone to use specific apps, selecting five common traffic service types in mobile networks and setting the mobile device to use representative apps for each type. The list of selected apps is presented in TABLE I. We collected 100 traffic traces for apps at the base session and 20 traffic traces for apps at the incremental session, each lasting for 20 seconds. Afterward, we mapped multiple RNTIs with the TMSI, and the corresponding DCI messages were filtered using the RNTIs chain to obtain all DCI messages of the experimented mobile phone. This study used uplink and downlink Transport Block Size (TBS) in DCI messages as the raw traffic data.

We also encode the raw data to a consistent length for an easy training step in the deep learning model. We slice the raw data with a non-overlapping sliding window of 0.1 seconds. Then we calculate eight features of the split series: mean TBS in the uplink, mean TBS in the downlink, max TBS in the uplink, max TBS in the downlink, total TBS in the uplink, total TBS in the downlink, the total number of packets arrived in the uplink, and the total number of packets arrived in the downlink.

B. Experiment and results

In order to evaluate our proposed approach, we compare the performance of the proposed method with two baseline methods.

- Joint-LSTM: At each training session t, we discard the model obtained from the previous session and retrain the entire model with training set $\{D(1), ..., D(t)\}$. However, retraining the entire model is increasingly time-consuming as the number of classes increases.
- Finetuning-LSTM: At each training session t, we keep the model obtained from the previous session and adapt the classifier, then directly update the model with the new dataset $D(t)$.
- Proposed: It is our proposed incremental learning method.

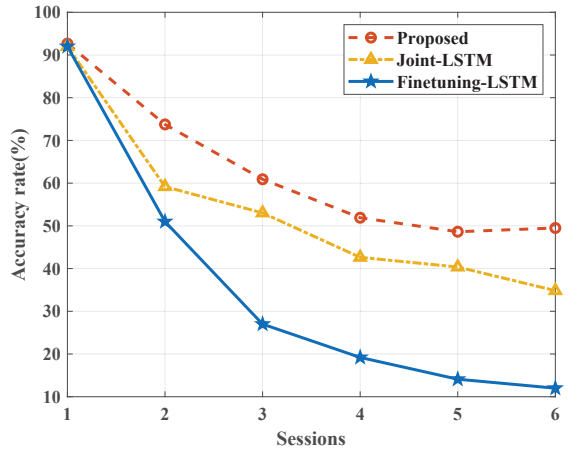
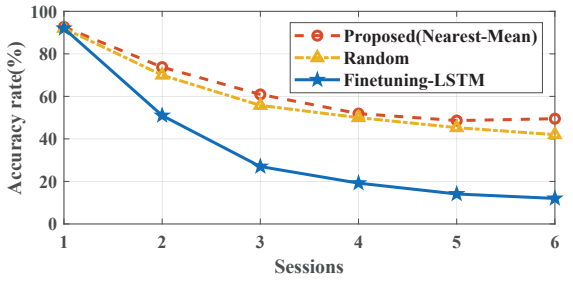
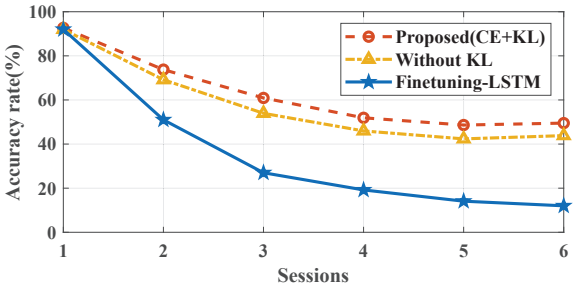


Fig. 3. Accuracy Rates *vs* Training Sessions



(a) Impact of exemplar selection



(b) Impact of knowledge distillation

Fig. 4. Incremental learning performance for ablation study

All models use the same dataset for training and evaluation on the same test dataset. The training set includes 90 samples for each of the 5 base classes and 10 samples for each of the 15 incremental classes, while the test set includes 10 samples for each class. We have 6 training and testing sessions. The first session trains the base model with the five base class data. After that, in each incremental session, we add three classes. Once completing each session, the model stores an exemplar set, including 5 representative samples per class. The accuracy rates on the test data at each session are reported as the metrics of the model's performance.

The experiment results are shown in Fig. 3, all the methods achieve 92% classification accuracy at session 1. Although the accuracy rate decreases as the number of classes increases, the proposed model always achieves a higher accuracy than the baseline models. And when there are 20 classes, the proposed model can still achieve a 50% classification accuracy, while the accuracy rates of Joint-LSTM and Finetuning-LSTM are 35% and 12%, respectively.

For Joint-LSTM, the degeneration of the accuracy is caused by class imbalance problem [20], as base classes have much more training samples than incremental classes. But for the proposed model, the exemplar set and the incremental training set have comparable sizes, so the class imbalance problem does not strongly impact our model. For Finetuning-LSTM, catastrophic forgetting leads to severe degeneration of the classification accuracy. In our model, the loss of previous knowledge is relieved by introducing rehearsal and knowledge distillation.

In order to analyze the impact of exemplar selection and knowledge distillation, we perform ablation study experiments. For exemplar selection, we compare the performance of our method with a random selection method, in which the exemplar set consists of randomly selected K samples from each class. Figure. 4(a) shows that the nearest-to-mean exemplar selection strategy increase the classification accuracy by 7.4% on average. The enhancement demonstrates the superiority of selecting the most representative samples by the nearest-to-mean method. For knowledge distillation, our method is compared with a model using cross-entropy loss solely. Figure. 4(b) shows that adding knowledge distillation loss helps the model to achieve an average of 6% higher accuracy rates. By adding knowledge distillation loss, we restrict the deviation of the current model from the previous model, therefore reducing the loss of previous knowledge. And the experiment result shows the enhancement by introducing knowledge distillation loss.

IV. Conclusion

This paper introduces a few-shot incremental learning framework that utilizes an LSTM network with an adaptive expanding classifier. The system allows the acquisition of new knowledge by expanding the classifier and updating the network. To deal with catastrophic forgetting, we introduce rehearsal and knowledge distillation techniques. To ensure efficient learning, we use a sample selection algorithm based on herding to identify representative examples from the dataset for rehearsal. To evaluate the effectiveness of our proposed framework, we collected a real-world traffic dataset. Our experimental results demonstrate that the proposed framework can incrementally learn new applications without retraining from scratch while maintaining high levels of classification accuracy.

References

[1] W. Wu, L. Jiang, C. He, D. He, and J. Zhang, "Ravenflow: Congestion-aware load balancing in 5g base station network," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[2] H. F. Alan and J. Kaur, "Can Android applications be identified using only TCP/IP headers of their launch time traffic?" in *Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks*, 2016, pp. 61–66.

[3] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic," in *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015, pp. 433–441.

[4] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Can't you hear me knocking: Identification of user actions on android apps via traffic analysis," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 297–304. [Online]. Available: https://doi.org/10.1145/2699026.2699119

[5] S. R. Systems, "SRS Airscope," 2022. [Online]. Available: https://srscom.com/airscope/

[6] N. Bui and J. Widmer, "OWL: A reliable online watcher for LTE control channel measurements," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, 2016, pp. 25–30.

[7] J.-W. Son, S. Lee, and M.-h. Han, "Supervised Service Classification using Downlink Control Indicator in LTE Physical Downlink Control Channel," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 1533–1536.

[8] H. D. Trinh, A. F. Gambin, L. Giupponi, M. Rossi, and P. Dini, "Mobile traffic classification through physical control channel fingerprinting: a deep learning approach," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1946–1961, 2020.

[9] L. Zhai, Z. Qiao, Z. Wang, and D. Wei, "Identify What You are Doing: Smartphone Apps Fingerprinting on Cellular Network Traffic," in *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2021, pp. 1–7.

[10] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed. Academic Press, Jan. 1989, vol. 24, pp. 109–165.

[11] G. Sun, T. Chen, Y. Su, and C. Li, "Internet traffic classification based on incremental support vector machines," *Mobile Networks and Applications*, vol. 23, pp. 789–796, 2018.

[12] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, L. Ouyang, and P. Yang, "Incremental Learning for Mobile Encrypted Traffic Classification," in *ICC 2021 - IEEE International Conference on Communications*, Jun. 2021, pp. 1–6.

[13] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, and Y. Wang, "Rethinking Encrypted Traffic Classification: A Multi-Attribute Associated Fingerprint Approach," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Oct. 2019, pp. 1–11.

[14] W. Zhu, X. Ma, Y. Jin, and R. Wang, "ILETC: Incremental learning for encrypted traffic classification using generative replay and exemplar," *Computer Networks*, vol. 224, p. 109602, Apr. 2023.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[18] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.

[19] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking lte on layer two," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1121–1136.

[20] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.