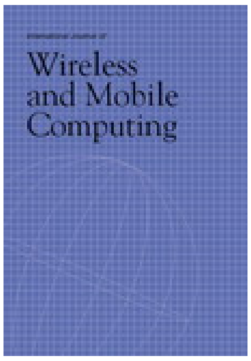


International Journal of Wireless and Mobile Computing

ISSN online: 1741-1092 - ISSN print: 1741-1084
https://www.inderscience.com/ijwmc

Hybrid compression for LSTM-based encrypted traffic classification model

Qiaoxu Mu, Meng Zhang

DOI: 10.1504/IJWMC.2023.10061557

Article History:

Received:	07 November 2022
Last revised:	30 January 2023
Accepted:	22 February 2023
Published online:	07 February 2024

Hybrid compression for LSTM-based encrypted traffic classification model

Qiaoxu Mu and Meng Zhang*

College of Computer Science and Technology,
Jilin University,
Changchun, Jilin, China
Email: 632114930@qq.com
Email: zhangmeng@jlu.edu.cn
*Corresponding author

Abstract: Traditional techniques for network traffic classification are no longer effective in handling the complexities of dynamic network environments. Moreover, deep learning methods, while powerful, demand substantial spatial and computational resources, resulting in increased latency and instability. In this paper, we propose an innovative approach to network traffic classification utilising an LSTM structure. This approach incorporates network pruning, knowledge refinement, and Generative Adversarial Networks (GAN) to reduce model size, accelerate training speed without compromising accuracy, and address challenges associated with unbalanced datasets in classification problems. Our methodology involves the pruning of unimportant filters from the teacher model, followed by retraining and knowledge distillation to generate the student model. Experimental show that the size of the pruned teacher model is only 25.69% of the original, resulting in a noteworthy 28.16% improvement in training speed. Additionally, the classification performance of various unbalanced traffic categories, such as VoIP and streaming, shows significant enhancement.

Keywords: encrypted traffic classification; filter pruning; knowledge distillation; deep learning.

Reference to this paper should be made as follows: Mu, Q. and Zhang, M. (2024) 'Hybrid compression for LSTM-based encrypted traffic classification model', *Int. J. Wireless and Mobile Computing*, Vol. 26, No. 1, pp.61–73.

Biographical notes: Qiaoxu Mu received his BS degree in Computer Science from Jilin University in 2020. Currently, he is a Third-Year Graduate student in the College of Computer Science and Technology at Jilin University. His research interests include deep learning and network security.

Meng Zhang received his PhD degree in Computer Science from Jilin University in 2003. Currently, he is a Professor at the College of Computer Science and Technology, Jilin University. His main research interests include stringology, network security and computational biology.

1 Introduction

Traffic classification is the division of traffic into multiple priority or service classes based on policies. There are three classical approaches to traffic classification: port-based approach, load-based approach, and traffic-statistics-based approach (Yamansavascilar et al., 2017). Owing to the network development, the amount of encrypted traffic has been increasing dramatically in recent years, and the classification of encrypted traffic has become increasingly important. The traditional methods all have their drawbacks. As new applications mostly use well-known port numbers or do not use standard registered port numbers, they seriously affect the accuracy of the port-based method (Rezaei and Liu, 2019). And payload-based methods, also known as deep packet parsing, are difficult to achieve encrypted traffic classification because the traffic is encrypted and the data in the payload does not have stable characteristics (Zou et al., 2018). The traffic-statistics-based methods mostly use machine learning algorithms such as random forest and KNN methods to classify traffic, but the disadvantage is that they are highly dependent on manual design hence they are unsuitable for the current complex and changing network environment.

In recent years, deep learning has had great success in directions such as image recognition, so researchers have applied Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), etc., to traffic classification. Compared with traditional methods, deep learning methods can not only maintain high accuracy but also adapt to complex and variable network environments without relying on the manual design. However, for highly redundant data sets, existing deep learning methods need to occupy massive storage and computational resources to obtain high accuracy (Hinton et al., 2015). However, when the models are deployed online, it is found that the relationship between model size and accuracy is non-linear, and the larger the number of model parameters, the slower the improvement of knowledge. This indicates that there are many redundant parameters in large models, and we can consider compressing the models to make them smaller and guarantee the accuracy of small models through ingenious training strategies.

In recent years, rapidly developing model compression techniques have provided powerful tools for reducing the space occupied by models and increasing their speed. Among them, model pruning and knowledge distillation are the most representatives. Model pruning is to design a strategy to evaluate the network parameters and remove redundant parameters on the trained model (Li et al., 2017). Pruning can be divided into unstructured pruning and structured pruning. Unstructured pruning first judges the importance of weights according to the set strategy and sets a certain percentage of weights to 0. This will produce an unstructured sparse filter, which is not much help for speedup and model reduction. Therefore, more consideration is given to using structured pruning to compress the model. Structured pruning, which removes whole rows and columns of weights, makes the number of parameters decrease and makes the model smaller while speeding up. The most used structured pruning methods are filter pruning, channel pruning, etc. Another approach is knowledge distillation, which is divided into two phases. First, we train a complex design model with high accuracy as the teacher model. Then, the softmax layer output of the teacher model is used as the soft target along with the softmax layer output of the student model as the total loss to train the student model (Hinton et al., 2015). This allows the knowledge of the teacher model to be transferred to the student model so that the more compact student model can achieve accuracy and performance similar to the teacher model.

In this paper, we propose a lightweight network traffic classification model using pruning and knowledge distillation techniques. The knowledge distillation is divided into two parts, the teacher model and the student model. Since LSTM networks can learn the spatiotemporal characteristics of network traffic well (Zou et al., 2018), the teacher model consists of three layers of LSTM and the student model consists of one layer of LSTM. To reduce the model as much as possible, this paper prunes the teacher model based on knowledge distillation. The teacher model is trained first, and the model is pruned after the training is finished to remove the weights and smaller filters. To compensate for the performance degradation caused by pruning, we use a one-time pruning and retraining strategy to recover the accuracy of the model as much as possible while reducing the time wasted in retraining. We use the pruned teacher model to generate logits to train the student model, where the total loss function is weighted by the extraction loss and the ground loss (i.e., the cross-entropy of the hard target and the student model probability). In the early stages of knowledge distillation, the training process of the student model requires the use of higher temperatures to help learn information between categories, and the presence of noise due to correlated information between categories can hinder accuracy improvement. To address this issue, we use an adaptive temperature function where the temperature changes with accuracy, making the value of temperature at different stages more reasonable.

In addition to this, this paper also designed a solution to the problem of imbalance in the number of samples in the data set. For the experimental results of the traffic classification model based on hybrid compression, it can be concluded that the classes with an extremely small number of samples in the data set. To enhance the learning ability of the model for the minority class, this paper used Generative Adversarial Network (GAN) to learn the features of the minority class samples and then generate data that can be 'faked' by the generator of the generative adversarial network. After that, the minority class data generated by the generative adversarial network is added to the original data set to generate a new data set while maintaining the ratio of the training set and test set in the original data set. Finally, the new data set is used to train a hybrid compression-based cryptographic network traffic classification model.

This paper is organised as follows. Section 2 describes the work related to network traffic classification. Section 3 describes the framework of the model and the complete training process in detail. Section 4 gives the experimental evaluation of our model. We conclude our work in Section 5.

2 Related work

2.1 *Encrypted traffic classification*

Rezaei and Liu (2019) presented a general framework for deep learning-based traffic classification, including data collection, data cleaning, feature selection and deep learning model selection. Shen et al. (2020) guided future research in the field of traffic classification. The authors focus their attention on the optimisation method of feature selection, summarise the cryptographic traffic features and introduce the feature selection framework in detail. The feature selection framework consists of three parts: feature pre-processing, feature evaluation and feature combination. Feature selection reduces the number and dimensionality of features, which in turn reduces overfitting and makes the model much more versatile.

Rachmawati et al. (2021) presented new research directions in deep learning-based traffic classification as well as a general framework. In addition to describing the application of common deep learning methods to traffic classification tasks, the authors also presented the work involved and the problems that exist. Recent work was reviewed in terms of the general framework, data preparation and preprocessing.

Ma et al. (2021) designed an encrypted traffic classification method based on traffic reconstruction, which differs from other methods in the processing of payloads. The method extracts the first 500 bytes of the payload and inserts a length threshold identifier to obtain the reconstructed traffic by traffic splitting, cleaning and reconstructing, and then classifies the traffic using a one-dimensional convolutional neural network. The reconstructed traffic has more key features, which in turn reduces the computational cost and speeds up the classification of encrypted traffic with high accuracy.

Akbari et al. (2022) designed a traffic classification method for encrypted Web protocols with a main neural network architecture based on stacked long short-term memory layers and convolutional neural network composition. The authors focused on new encrypted Web protocols, namely HTTP/2 and QUIC. In the method, the authors used standard traffic statistics, traffic shapes related to packet size, raw bytes from TLS handshake packets and arrival time and direction, which differ from common traffic classification methods, and the authors showed that the proposed feature set is more suitable for encrypted traffic classification. With the help of the neural network architecture designed by the authors, the number of trainable parameters is smaller and the possibility of overfitting is less.

2.2 *Pruning*

A sparse network is obtained after pruning is completed. Then retraining is performed and pruning and retraining can be iterated many times to reduce the network complexity as much as possible. Zhuang et al. (2019) proposed a Differentiated sensing Channel Pruning (DCP) scheme. DCP allows better results in the pruning and updating model phases by making perceptive use of discrimination-aware loss and the final loss. The authors propose a greedy algorithm for channel selection and parameter optimisation in an iterative manner.

Zheng et al. (2022) proposed a micro-network channel pruning method, which searched for the most available sub-structure that satisfies the resource constraint by gradient descent. First optimised the network parameters to make the search space continuous, then calculated the probability of the channel being retained based on the learnable probability, then pruned it, and finally restarted the training to obtain the pruned model.

Salehinejad and Valaee (2022) inspired by the dropout concept and used dropout as an energy-based framework for pruning neural networks. Unlike most methods, the energy-based model evolved stochastically to find states with lower energy loss, and then the best pruning state was selected and applied, cyclically. In this process, iterations were constantly switched between managing pruning states and updating the retained weights. This way the method allowed the pruning of the neural network without modifying the network architecture code.

Rong et al. (2020) designed a method that can correct incorrect pruning and does not rely on fine-tuning. This method is a gradient-based approach that dynamically reduces the complexity. During the pruning process, the evaluation criteria are determined based on the gradient and will classify the filters into strong and weak filters. The strong filter uses the gradients associated with the decay of the objective function and weights to update the parameters to maintain the model performance. The gradient flow of the weak filter will be blocked. As training proceeds, the set of blocking filters will continue to shrink and eventually converge to a relatively stable set. When pruning errors occur, the model can be corrected for pruning errors by reactive pruning filters in the next training cycle. That is, the accuracy of the model is guaranteed while making it as small as possible.

Guo and Li (2021) proposed a pruning method that combines convolutional kernel pruning and filter pruning. This significantly reduced floating point operations (FLOP), reduced the number of parameters and maximises the compression of the model so that it can be deployed on mobile devices.

Chen et al. (2021) proposed a hybrid pruning method to compress the CNN model. Hybrid pruning consists of filter pruning and 2:4 pruning. The authors applied filter pruning to remove the redundant filters in the convolutional layer to make the model smaller. Next, the authors used 2:4 pruning to prune the model according to the 2:4 pattern to utilise the sparse tensor core hardware for acceleration. In addition to this, the authors proposed a hybrid ranking metric that retained the filters that were important for both pruning steps, which allowed the model to obtain higher accuracy.

2.3 *Knowledge distillation*

In their paper, Li et al. (2022) proposed a new online knowledge distillation method named Feature Fusion and Self-Distillation (FFSD) for online knowledge distillation. The method divided the student part of knowledge distillation into leading and normal students, and the authors designed a module with a diversity enhancement strategy to perform feature fusion and extract knowledge to leading students to improve the generality of the model. In addition to this, a self-distillation module was proposed which converts deep feature maps into shallow feature maps, which helped students to learn the knowledge better.

Gu et al. (2021) proposed a structured attention distillation method for lightweight networks. When there was a large structural difference between the teacher network and the student network, the method enhanced the refinement of the spatial attention map by grouping features in the model according to channels, which in turn improved the feature extraction ability of students.

A new method called Relational Knowledge Distillation (RKD) is proposed to convey the structural relationships of the output (Park et al., 2019). The authors proposed two types of RKD losses: distance direction distillation loss and angular direction distillation loss. RKD calculates the relational potential of each data example as a way to transfer information from the teacher to the student. RKD is based on traditional Knowledge Distillation (KD) and can be combined with other methods to improve performance. RKD can be seen as a generalisation of the traditional KD and can also be combined with other methods to improve performance due to its complementary nature to the traditional KD.

2.4 Generating adversarial networks

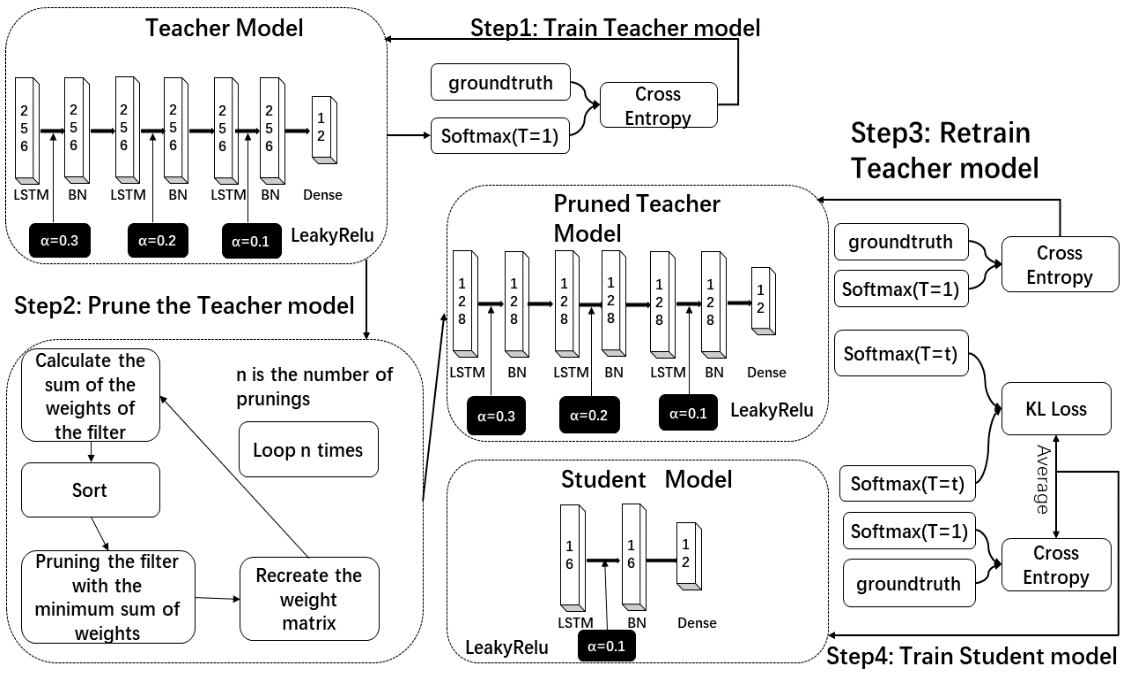
Wang et al. (2019) proposed the FlowGAN method based on GAN to solve the traffic classification imbalance problem. Taking advantage of rate GAN data augmentation, flow data is generated for classes with fewer samples, and then the generated data is combined with the original data to form a new flow data set to be used for training. The authors used the deep learning classification method MLP and conducted experiments on imbalanced data sets, oversampled data sets,and generative adversarial network generated data sets, which showed that the FlowGAN synthesised data sets have better performance. Guo et al. (2021) used GAN to overcome the problem that the oversampling technique tends to overfit and introduce noise. And to ensure the quality of the generated samples, the authors designed an end-to-end framework that integrates the generation of a few flow samples and the training of the target classifier. The authors designed a feedback mechanism that can better guide the direction of sample generation while indicating the quality of the generated samples. Wang et al. (2020) proposed a traffic data enhancement method PacketCGAN based on a Conditional Generative Adversarial Network (CGAN), a type of GAN that controls the pattern of generated data. As a generative model, PacketCGAN takes advantage of CGAN to generate synthetic traffic samples by learning the features of the original traffic data. The synthetic data is then combined with the original data (i.e., real data) to construct a new traffic data set, thus maintaining a balance between the primary and secondary classes of the data set.

3 Methodology

The main body of work in this paper is divided into two parts, the first part is the design and training of a hybrid compression-based network traffic classification model, and the second part is to generate sample images using GAN, put the sample images into the data set, and retrain the model in the first part using the new data set.

Figure 1 shows the working framework of the first part. The first step of the workflow is to train the teacher model, the body of the model consists of LSTM and the loss function for training is cross entropy. The second step is to prune the trained teacher model. The pruning method is filter pruning, which determines whether to keep the filter based on the weight and size of the convolutional kernel. The third step is to retrain the pruned teacher model. The main purpose of retraining is to compensate for the loss of model accuracy during the pruning process as much as possible. The fourth step is to train the student model with knowledge distillation. The main body of the student model consists of LSTM, and the loss function of the training consists of the average of both cross-entropy ce_loss and KL_loss derived from KL divergence.

Figure 1 Hybrid compression work frame



Teacher Model
Step1: Train Teacher model
256 256 256 256 256 256 12
LSTM BN LSTM BN LSTM BN Dense
α=0.3 α=0.2 α=0.1 LeakyRelu
groundtruth
Softmax(T=1)
Cross Entropy
Step3: Retrain Teacher model
Step2: Prune the Teacher model
Calculate the sum of the weights of the filter
Sort
Pruning the filter with the minimum sum of weights
Recreate the weight matrix
n is the number of prunings
Loop n times
Pruned Teacher Model
128 128 128 128 128 128 12
LSTM BN LSTM BN LSTM BN Dense
α=0.3 α=0.2 α=0.1 LeakyRelu
groundtruth
Softmax(T=1)
Cross Entropy
Softmax(T=t)
KL Loss
Softmax(T=t)
Average
Softmax(T=1)
groundtruth
Cross Entropy
Student Model
16 16 12
LSTM BN Dense
LeakyRelu α=0.1
Step4: Train Student model

The second part is the workflow of the generative adversarial network part. First, the data set needs to be restored from *idx* format to *png* format, and then the categories to be data enhanced are selected according to the quantitative relationship between the categories, after which the images in the data set are sampled and input to the discriminator, then the images are generated using random noise and the generator, and the generated images are added to the discriminator for discrimination and iterated several times until the two reach Nash equilibrium. Then, the data augmentation is completed by adding the newly generated images to the data set according to their categories. Finally, the data-enhanced data set is used to train the model generated in the first part.

3.1 Model structure

3.1.1 Teacher model

Recurrent Neural Networks (RNN) use internal memory to process arbitrary temporal sequences of inputs and are called RNN because the current output of a sequence is correlated with the previous output. The biggest problem with recurrent neural networks is that they are susceptible to short-term memory. If a sequence is long enough, it is difficult to pass earlier time steps to later time steps, so important information may be missed. In addition, RNN also faces the problem of gradient disappearance. LSTM has an internal mechanism called 'gate' that regulates the flow of information and has the function of choosing to keep information and forget it, which solves the problem in RNN. The flow is time-dependent within the packet, so we consider the LSTM as the teacher model and the student model. Specifically, for the teacher model, we use a three-layer LSTM, with each layer consisting of 256 cells. We added BatchNormalisation (BN) after each LSTM layer, and the purpose of adding the BN layer is to speed up the training and convergence of the network. In addition, the BN layer controls the gradient explosion, prevents gradient disappearance and prevents overfitting. At the end of the model, there is a fully connected layer, consisting of 12 units.

Between each LSTM layer and BN layer, we added LeakyReLU activation functions with α of 0.3, 0.2 and 0.1 for the functions in the teacher model.

3.1.2 Student model

To ensure that the student model can learn better from the teacher model, the structure of the teacher model and the student model should be as similar as possible. Therefore, in this experiment, the student model is also based on LSTM, but unlike the teacher model, the student model uses one layer of LSTM, consisting of 16 units and does not use L2 regularisation because the model is small and the number of parameters is not large. BatchNormalisation is also added after the LSTM layer, and a LeakyReLU activation function is added between the LSTM and BN layers with an α of 0.1.

3.2 Filter pruning

In the previous section, we trained a large teacher model with high accuracy and complex structure. In this section, we use filter pruning to compress the teacher model. The pruning criterion is the sum of the weights of the convolutional kernels, using the L1 paradigm. We consider that the generation of convolutional kernels with smaller weights represents that it generates weaker feature maps, which have little impact on the model because the information they transmit is negligible, and removing the corresponding filters does not have a significant impact on the network. After removing enough filters, the network is retrained to recover the accuracy. The process of pruning n filters from the *i*-th convolutional layer is as follows.

- For each filter F in that layer, calculate the sum of the absolute values of the convolutional kernel weights

$$\text{S}=\sum\left|W(i)\right| \tag{1}$$

- Sorting the calculated S
- Pruning weights and minimum filters and their corresponding feature maps using the Keras-surgeon library
- Create a new matrix, rearrange the remaining kernel weights and cycle through the above steps until the pruning number is reached

After pruning the filter, we need to retrain the network to compensate for the performance drop. There are two strategies.

1) *Re-training after one-time pruning*: Prune the filter of the desired layer at one time, and retrain it after the pruning is over to restore accuracy.
2) *Pruning and retraining iteration*: after pruning a filter or pruning a layer of filters, retrain once. The pruning and retraining process is iterated until the desired number of pruning's is reached.

The strategy we use is to retrain after a one-time pruning. This restores the model accuracy as much as possible. Iterating for pruning and retraining may yield better results (Li et al., 2022), but the iterative process takes more time, especially since our teacher model is more complex. It is more than worthwhile compared to the reduced training time we compress. Therefore, we use a one-time pruning and then retrain and fine-tune after the pruning is over.

3.3 Knowledge distillation

The temperature value in knowledge distillation is one of the key points, and the level of temperature changes how much attention is paid to the negative labels during the training of the student model. When the temperature is lower, less attention is paid to the negative labels; while when the temperature is higher, the negative labels receive more attention. That is, the temperature should be turned up if you want to learn more from the negative labels, and slightly lower when you want to reduce the interference from noise. So, to get better performance, the temperature T is allowed to increase first, and then gradually decrease the temperature. For this purpose in this paper, an automatically varying temperature function is designed, where α is the initial temperature value:

$$T = \alpha + (2 - 4 \times accuracy)^3 \tag{2}$$

Another key in knowledge distillation is the loss function of the student model. Unlike the teacher model, the loss function student_loss of the student model consists of KL_loss and ce_loss, with the following equation:

$$\text{student\_loss}=0.5 \times (\text{KL\_loss} + \text{ce\_loss}) \tag{3}$$

Before giving the definitions of KL_loss and ce_loss, the following definitions are stated:

v_i : The output logits of the teacher model at the *i*-th sample.

z_i : The output logits of the student model at the *i*-th sample

c_i : The truth label of the *i*-th sample

N: Total number of labels

p_i^T : The softmax output value of the teacher's model for the *i*-th sample at temperature *T*, with the following equation:

$$p_i^T = \frac{\exp\left(\frac{v_i}{T}\right)}{\sum_j^N \exp(v_j)} \tag{4}$$

q_i^T : The softmax output value of the *i*-th sample of the student model at temperature *T*, with the following equation:

$$q_i^T = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j^N \exp(z_j)} \tag{5}$$

KL_loss is the loss function calculated using the KL divergence:

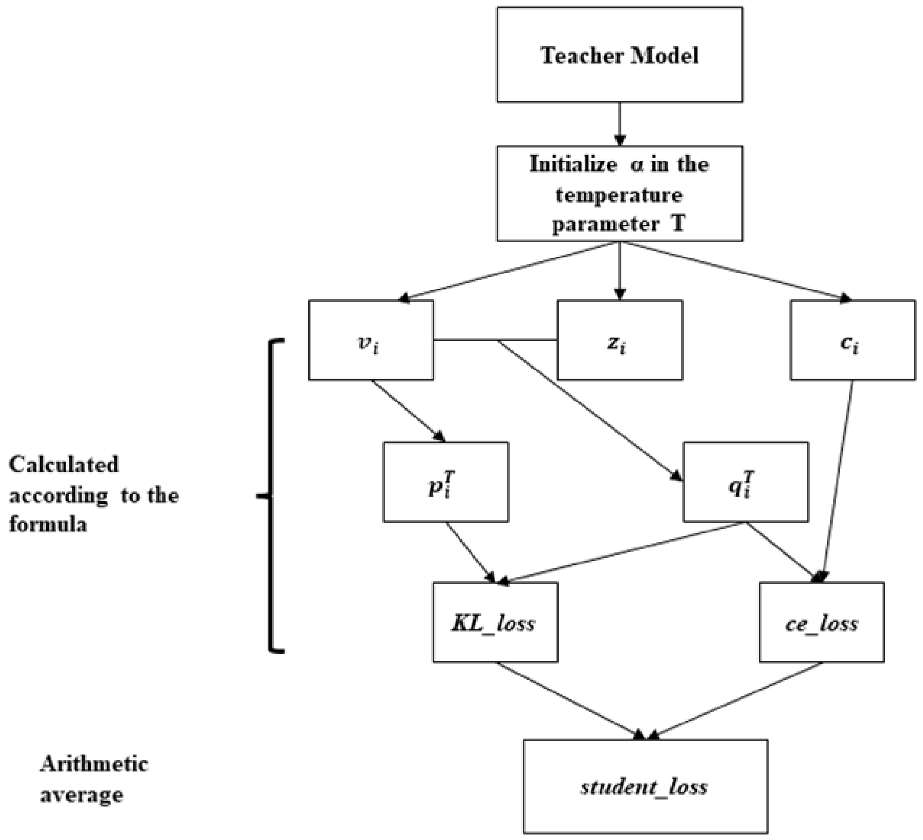
$$KL_{loss} = -\sum_i^N p_i^T \cdot \left(\log(p_i^T) - \log(q_i^T)\right) \tag{6}$$

ce_loss is the cross-entropy loss function calculated from the truth labels and the logits of the student model output:

$$ce_{loss} = -\sum_i^N c_i \log(q_i^1) \tag{7}$$

As shown in Figure 2, first, the accuracy is initialised to 0 and the temperature T value is α . Then, the model starts training by calculating the logits of the teacher model and the student model separately, then calculating KL_loss and ce_loss respectively, and then the arithmetic average of the two is used to find student_loss. To prevent model overfitting, the student model is also trained using the early stop method and the completed model is the desired target network model.

Figure 2 Knowledge distillation process



3.4 Data augmentation based on generative adversarial networks

3.4.1 The framework of GAN

GAN consists of two models: a generator and a discriminator. The task of the generator is to input random noise and create an image using a deconvolutional network. The discriminator's task is to input an image and determine whether it is a 'real' image from the data set or a 'fake' image generated by the generator.

The training process of GAN consists of the following steps:

1 The encrypted traffic data set is sampled to obtain $P_{data}(x)$ and $P_{data}(x)$ is fed to the discriminator D to obtain the discriminant result.

2 Generate the random noise $P_z(z)$, feed the random noise into the generator G model and then use G to generate the picture $G(z)$.

3 The generated picture of G is fed into the discriminator to judge the result.

4 Calculate the loss of the generator and discriminator using the loss function.

5 Update the parameters

6 Repeat the above steps until G and D reach Nash equilibrium.

Discriminator Since the work task is a binary classification problem, this paper uses a binary cross-entropy loss function to train the discriminator with the following formula:

$$D_{loss} = -\frac{1}{n} \cdot \sum_{i}^{n} \log\left(D\left(x_i\right)\right) + \log\left(1 - D\left(G\left(z\right)\right)\right) \tag{8}$$

Since the discriminator always wants to minimise the loss, it is desirable to minimise the above equation:

$$\min_{G} \max_{D} V\left(G, D\right) E_{x \sim P_{data}(x)}\left[\log\left(D\left(x\right)\right)\right] + E_{z \sim P_z(z)}\left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right] \tag{9}$$

The generator is only involved in the second half of the expression, so the generator loss function is as follows:

$$D_{loss} = \frac{1}{n} \cdot \sum_{i}^{n} \log\left(1 - D\left(G\left(z\right)\right)\right) \tag{10}$$

In the early stage of training, the discriminator can easily determine whether the image is a real image or not, precisely because $D\left(G\left(z\right)\right) \sim 0$, in which case $\log\left(1 - D\left(G\left(z\right)\right)\right)$ is saturated, so $\log\left(D\left(G\left(z\right)\right)\right)$, which is maximised by training, is chosen. The loss function is as follows:

$$D_{loss} = \frac{1}{n} \cdot \sum_{i}^{n} \log D\left(G\left(z\right)\right) \tag{11}$$

3.4.2 Training algorithms

The training of the model generally consists of two parts: the training of the discriminator and the training of the generator, in which the weights and bias vectors are updated using the Adam optimiser. The first step is to train the discriminator, which consists of an input layer, two Dense layers and an output layer. The data in the input layer is derived from the real data set, and the input multidimensional data is expanded into one-dimensional data with the distribution $P_{data}\left(x\right)$ in the input layer. The hidden layer contains two Dense layers noted as D_{h1} and D_{h2}, and the following results can be obtained according to the operations implemented by Dense:

$$D_{h1} = LeakyReLU\left(input_D \cdot W_1^D + b_1^D\right) \tag{12}$$

$$D_{h2} = LeakyReLU\left(D_{h1} \cdot W_2^D + b_2^D\right) \tag{13}$$

where $input_D$ is the output result of the input layer, W_1^D and W_2^D are the weight matrices and, b_1^D and b_2^D are the bias vectors. The activation function used is LeakyReLU, which is very similar to the ReLU function. However, there is a big difference between the two when the input is less than 0. The ReLU input is less than 0, while the LeakyReLU value is negative and the gradient is smaller. In the backpropagation process, the gradient can be calculated for the part of the LeakyReLU activation function less than zero, which avoids the possible neuron death problem of ReLU, so LeakyReLU is used as the activation function in this paper. The output layer aims to output a one-dimensional determination result as follows:

$$D_{out} = D_{h2} \cdot W_3^D + b_3^D \tag{14}$$

The second step is to train the generator, which has an input layer with a random vector of dimension 100, followed by three Dense layers with the number of neurons 256, 512 and 1024. the output of each layer is as follows:

$$G_{h1} = LeakyReLU\left(input_G \cdot W_1^G + b_1^G\right) \tag{15}$$

$$G_{h2} = LeakyReLU\left(G_{h1} \cdot W_2^G + b_2^G\right) \tag{16}$$

$$G_{h3} = LeakyReLU\left(G_{h2} \cdot W_3^G + b_3^G\right) \tag{17}$$

The input of the output layer is the result of the random noise $P_z\left(z\right)$ processed by the three Dense layers. The computed result will be activated by the *tanh* function, and a (28,28,1)-dimensional image will be output after the processing:

$$G_{out} = \tanh\left(G_{h3} \cdot W_4^G + b_4^G\right) \tag{18}$$

4 Evaluation and experimental results

4.1 Data set

4.1.1 Selection of data set

In this paper, we use the public data set ISCX-VPN-non-VPN. The ISCX data set, published by Gil et al. (2016) is rich in variety and quantity, containing seven types of regular encrypted traffic (e.g., P2P, VoIP) and seven types of encrypted traffic over VPNs. Moreover, the public data set does not compromise the credibility of the results compared to self-collected traffic or private traffic. Therefore, it is most feasible to use the ISCX data set, where the raw traffic does not have labels, so we need to tag traffic types for these files, where browser traffic from media may be tagged as browser traffic, a problem reported by He and Li (2020) and Wang et al. (2017). Therefore, we decided not to label browser and VPN browser traffic, and the detailed information is listed in Table 1.

Table 1 ISCX VPN-non-VPN data set

Categories	*Applications*
Chat	ICQ, AIM, Skype, Facebook, Hangouts
Email	SMPT, POP3, IMAP
File transfer	Skype, FTPS, SFTP
VoIP	Facebook, Skype, Hangouts, VoIP buster
P2P	Torrent
Streaming	Vimeo, YouTube, Netflix, Spotify
VPN-Chat	ICQ, AIM, Skype, Facebook, Hangouts
VPN-Email	SMPT, POP3, IMAP
VPN-File transfer	Skype, FTPS, SFTP
VPN-VoIP	Facebook, Skype, Hangouts, VoIP buster
VPN-P2P	BitTorrent

4.1.2 Data pre-processing

First, session extraction is performed. Our work is based on sessions and all layers, so we need to extract the sessions of each pcap file. Session extraction is followed by traffic clearing, which anonymises the information in the data frame headers since data frame header information (e.g., MAC address and IP address) may cause the classifier to overmatch. Then the IP addresses, TCP\UDP port numbers and data link layer addresses are randomised. Afterward, duplicate packets are removed as well as packets with empty payloads.

After performing the above operations, the packets are unified into the specified length. In this paper, we specify the packet length to 784 bytes, and the packets with insufficient length are to be truncated by filling 0 to reach the specified length, while the packets with lengths beyond the specified length are to be truncated.

Next, since IDX files are a common data storage file format in the machine learning field, The training set and the test set are stored as IDX files and we set the ratio of the number of training data to the number of test data to 9:1.

4.2 Experimental environment

The experimental platform is Windows 10 with a 2.40 GHz Intel I5-9300H CPU, an external GPU (Nvidia GeForce RTX2060) and 8GB of RAM. The software used for training and pre-processing includes Tensorflow 2.6, CUDA 11.3 and Jupyter Notebook 6.4. For convenience, the artificial neural network library Keras is chosen. pruning tools are used with the third-party open-source tool Keras-surgeon.

The number of training iterations is 1000, except for the early stop mechanism, which stops the training when the accuracy of the model no longer improves, with a patience of 100 for the teacher model and 50 for the student model.

4.3 Evaluation indicators

Four evaluation metrics were used: Accuracy, Precision, Recall and F1-Score. Accuracy is the number of samples predicted to be right as a percentage of the total number of samples. Precision is a prediction-specific metric that is how many of the samples predicted to be positive are positive samples. Recall is sample-specific and indicates how many positive cases in the sample were predicted correctly. Accuracy is used to evaluate the overall performance of the classifier, while Precision and Recall are used to evaluate the performance of each traffic class. TP: the number of positive classes predicted as positive; FN: the number of positive classes predicted as negative; FP: the number of negative classes predicted as positive; TN: the number of negative classes predicted as negative. the F1-Score takes into account both the precision and recall of the model, which better balances the two-evaluation metrics. The formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision \times Recall} \tag{22}$$

In this paper, Macro Average (MA) and Weighted Average (WA) are used to calculate the mean, respectively. The formulas for MA and WA are as follows:

$$MA_i = \frac{1}{Q} \sum_{q}^{Q} I_i^q \tag{23}$$

$$WA_i = \frac{1}{Q} \sum_{q}^{Q} \left(N_q \times I_i^q \right) \tag{24}$$

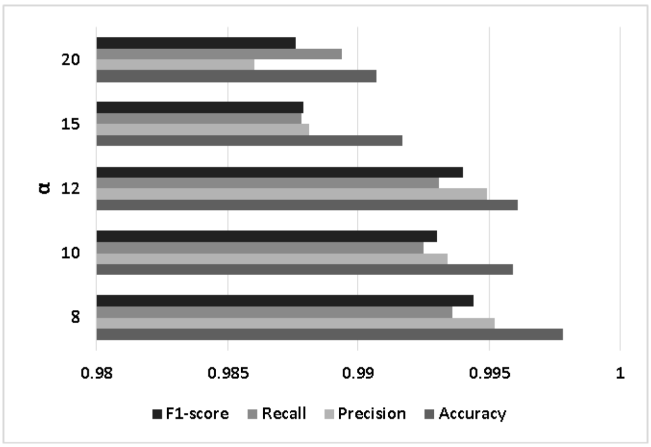
where I = {Precision, Recall, F1-score}, Q is the total number of sample categories, q is the samples in Q, and the total number of samples is N. The MA metric is calculated using the arithmetic mean method, which considers the contribution of all categories, so the categories with more samples dominate the categories with fewer samples, and then the value of using MA is severely diminished when there is a severe category imbalance in the data set. The WA takes into account the imbalance of the multi-categorical problem by assigning different weights to each category according to the sample size and then calculating the average.

In addition to this, we focus on the compression effect of the model, which can be known by the model size (Space) and the number of model parameters (TotalParams). In addition, we also focus on the training time and prediction time of the model.

4.4 Experimental results

To have a better performance of the traffic classification model based on hybrid compression, an automatically varying temperature function is designed in this paper in the knowledge distillation method. First, experiments will be conducted to investigate α in the temperature function. The experimental results are shown in Figure 3. It can be observed that the model has the best performance when $\alpha = 8$. The model has the worst performance when $\alpha = 20$, which is due to the initial temperature is too high, resulting in the temperature not being able to drop to the right range later in the training period.

Figure 3 Initial temperature parameters versus model performance



When $\alpha = 8$, the temperature function curve is shown in Figure 4. According to the multifaceted work (Hinton et al., 2018; Wu and Zhang, 2021), the range of temperature in this paper is set between [2, 20]. With the increasing training time, the accuracy of the model is also rising, when the automatically changing temperature function will gradually decrease, reducing the model can be affected by noise in the late training period, making the model can learn more knowledge from difficult features and difficult samples, improving the learning ability of the model and ensuring that a model with high performance can be obtained.

To measure the performance of the models in terms of compression, the storage and computational resources of the models need to be considered, and the experimental results are shown in Table 2 below. Among the models, T_Model and S_Model are using only the knowledge distillation method, and models PT_Model, PS_Model, GAN_T and GAN_S are derived from the hybrid compression method in this paper. Models GAN_T and GAN_S are the models derived from training using data-enhanced data sets. It can be observed that the volume of model PT_Model and GAN_T is 4079 KB, which is 25.8% of model T_Model, and the number of model parameters is 25.3% of model HCETC_T. The volume of model PS_Model and model GAN_S is only 33KB compared to model HCETC_PT, and the number of model parameters is only 0.9% of model HCETC_PT. In terms of computational resources of the model, the training time of model PT_Model decreases from 31.03 to 21.16 ms per step, and the inference time decreases by 58.3%. The training time of model GAN_T decreased to 22.29 ms, and the inference time decreased by 13.8%. Model PS_Model training time decreased by 21% and prediction time decreased by 27.7%. Model GAN_S training time decreased by 35.4% and inference time decreased by 47.2%. The PS_Model and GAN_S models that undergo filter pruning and knowledge distillation largely reduce the demand for storage and computational resources. In addition to this, the experiments include student models noKD_Model and PT_noKD derived from direct training without knowledge distillation, and models KD_Model and PT_KD are trained using the fixed temperature knowledge distillation method. It can be seen from the table that the models noKD_Model and PT_noKD are much faster than the other models in terms of training speed, which is due to the need to calculate the loss values during the training of the other models, including soft labels and student model logits, but this time consumption is acceptable compared to the reduction of the model's demand for storage and computational resources.

Figure 4 Temperature curve

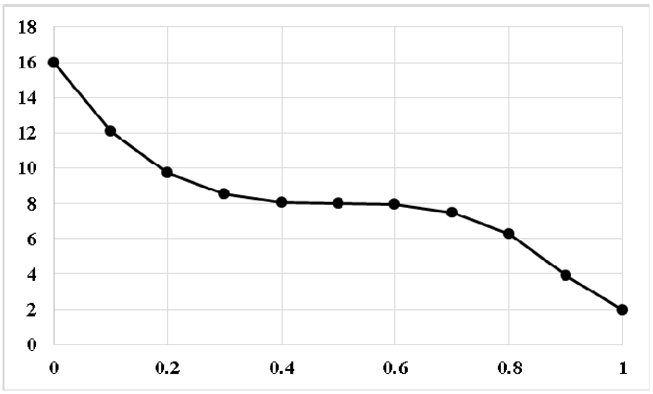


Table 2 Comparison of storage and computational resources between models

	Space (KB)	*TotalParams*	*TT (ms/step)*	*PT(s)*
T_Model	15786	1348620	31.03	2.16
noKD_Model	33	3148	7.63	0.58
KD_Model	33	3148	15.28	1.04
S_Model	33	3148	18.27	0.64
PT_Model	**4079**	**341388**	**21.16**	**0.9**
PT_noKD	33	3148	7.86	0.54
PT_KD	33	3148	14.6	1.09
PS_Model	**33**	**3148**	**16.71**	**0.65**
GAN_T	**4079**	**341388**	**22.29**	**1.86**
GAN_S	**33**	**3148**	**11.8**	**1.14**

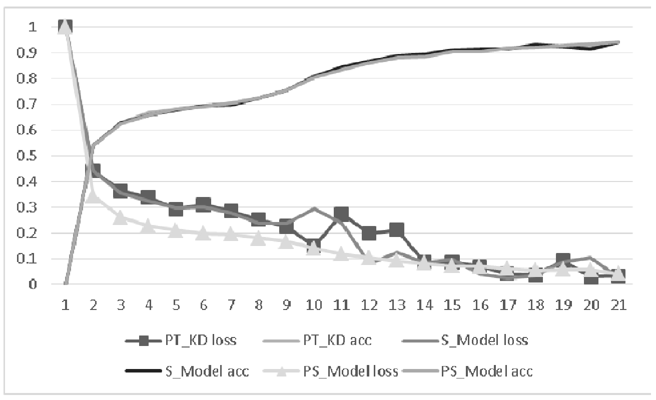
It is worth mentioning that although the model PT_Model and model GAN_T are compressed, the performance is very similar to T_Model, if not better. The experimental results are shown in the following Table 3, in general, the performance difference between the student model PS_Model and GAN_S and the teacher model is not large, and the accuracy of model PS_Model and teacher model PT_Model is only 0.38% lower than the teacher model, the most serious decrease is $Recall_{MA}$, which is 0.97% lower. The most serious drop is $Recall_{MA}$, which decreases by 0.56%. The performance of model noKD_Model is the worst. This model has not undergone knowledge distillation and has poor learning ability for teacher model T_Model. Although model KD_Model used the knowledge distillation technique, the noise problem seriously affected the feature learning ability of the model in the later stage of training due to the fixed value of the temperature parameter, which in turn seriously reduced the performance of the model. The models S_Model, PS_Model, and GAN_S with adaptive temperature functions have better performance by adaptively adjusting the temperature parameters, using a lower temperature in the early stage so that the model can learn knowledge quickly, and appropriately increasing the temperature in the later stage in the face of the noise problem, focusing on the processing of hard labels and thus improving the learning ability for difficult features and samples. Compared with model PS_Model and GAN_S and model S_Model, although they both use the adaptive temperature function designed in this paper, the teacher models of the former two models are pruned by filters to remove some of the redundant parameters, making the models have better performance and therefore the student models also have better performance. This proves the effectiveness of the hybrid compression method in this paper.

Table 3 Experimental results of the eight methods

	Accuracy	*Precision*		*Recall*		*F1-Score*	
		MA	*WA*	*MA*	*WA*	*MA*	*WA*
T_Model	0.9987	0.9921	0.9988	0.9957	0.9987	0.9937	0.9987
noKD_Model	0.9693	0.9721	0.9704	0.9703	0.9693	0.9709	0.9693
KD_Model	0.9896	0.9780	0.9897	0.9800	0.9896	0.9790	0.9896
S_Model	0.9926	0.9807	0.9927	0.9824	0.9926	0.9813	0.9927
PT_Model	**0.9997**	**0.9983**	**0.9998**	**0.9998**	**0.9997**	**0.9990**	**0.9997**
PT_noKD	0.9744	0.9778	0.9765	0.9653	0.9744	0.9707	0.9743
PT_KD	0.9909	0.9826	0.9910	0.9844	0.9909	0.9833	0.9909
PS_Model	**0.9959**	**0.9940**	**0.9960**	**0.9901**	**0.9959**	**0.9919**	**0.9959**
GAN_T	**0.9998**	**0.9998**	**0.9998**	**0.9992**	**0.9998**	**0.9995**	**0.9998**
GAN_S	**0.9978**	**0.9952**	**0.9978**	**0.9936**	**0.9978**	**0.9944**	**0.9978**

In the experiments of this paper, to obtain the best-performing model, the early stop method is used to end the training in this paper, when the performance of the model does not improve for a long time, which also leads to different training epochs for each model. At the same time, the early stop method can prevent the model from overfitting. In addition, since the loss functions of different models are not the same, resulting in the obtained loss values not having the same criteria, they cannot be compared directly. To compare the learning ability of different models, the loss values of the models were first normalised so that the loss values ranged within [0, 1], which solved the proposition that the loss functions of the models were different. After that, the loss values of each model were compared at 1 to N*10, which solved the problem of the different number of training iterations for different models. The experimental results are shown in Figure 5. In terms of model loss values, it can be observed that model PT_KD has significant fluctuations in loss value drop in the later stages of training because it uses a fixed knowledge distillation temperature, while models S_Model and PS_Model both use an adaptive temperature function so there is less fluctuation in the later stages. In addition, because the model PS_Model uses the filter pruning method to remove some redundant parameters, the model PS_Model has the fastest decline in the loss value and smoother fluctuations in the later stages of the experiment. In terms of the accuracy of the model, each model converges very well. Therefore, the model PS_Model proposed in this paper has the highest accuracy and the fastest learning speed among the student models.

Figure 5 Comparison of PT_KD, HCETC_S and HCETC_PS models



As shown in Figure 6, the ablation experiment of adaptive temperature was performed. It can be observed that the metrics of PT_Model are much better than KD_Model and PT_KD, which can prove that the adaptive temperature function is significant. The metrics of PT_KD are similar to S_Model and even better in accuracy, which also proves that our pruning algorithm which ensures the model accuracy and reduces the model size.

Figures 7 and 8 show the confusion matrices of the classification performance of model GAN_S and model PS_Model for different sample classifications. The confusion matrix focuses on the Precision of the model classification, and it can be found that model PS_Model has more categories where classification errors occur, while model GAN_S has classification errors in categories 7 and 8. This indicates that the data enhancement method designed in this paper plays an excellent role in solving the classification difficulty problem caused by data imbalance. The specific classification performance is shown in Figure 7.

Figure 6 The ablation experiment of adaptive temperature

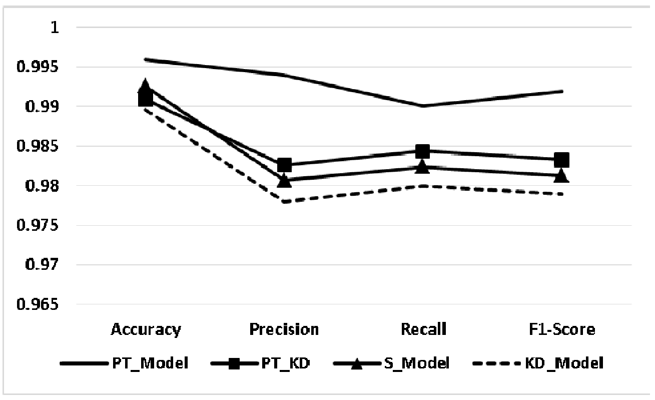


Figure 7 Confusion matrix of the model PS_Model

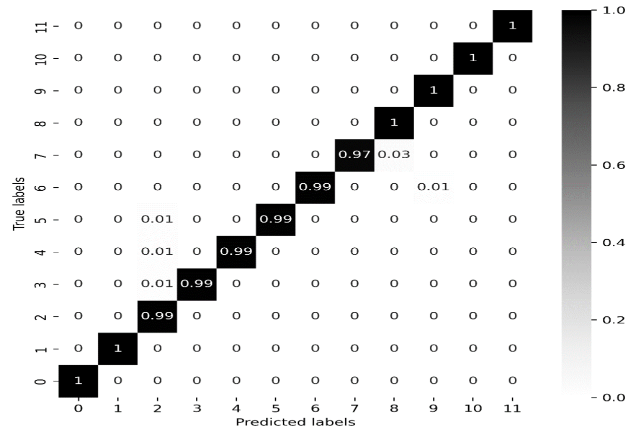


Figure 8 Confusion matrix of the model GAN_S

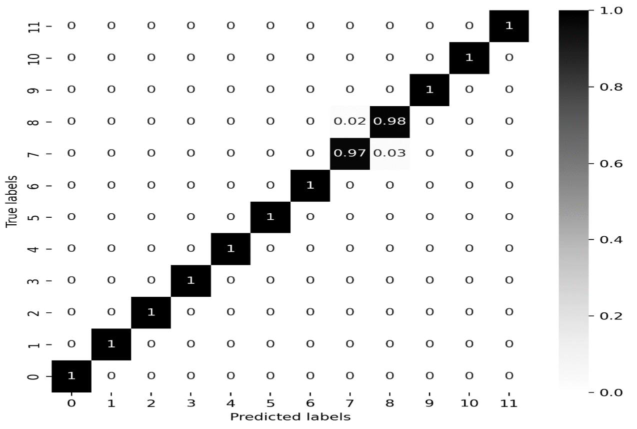
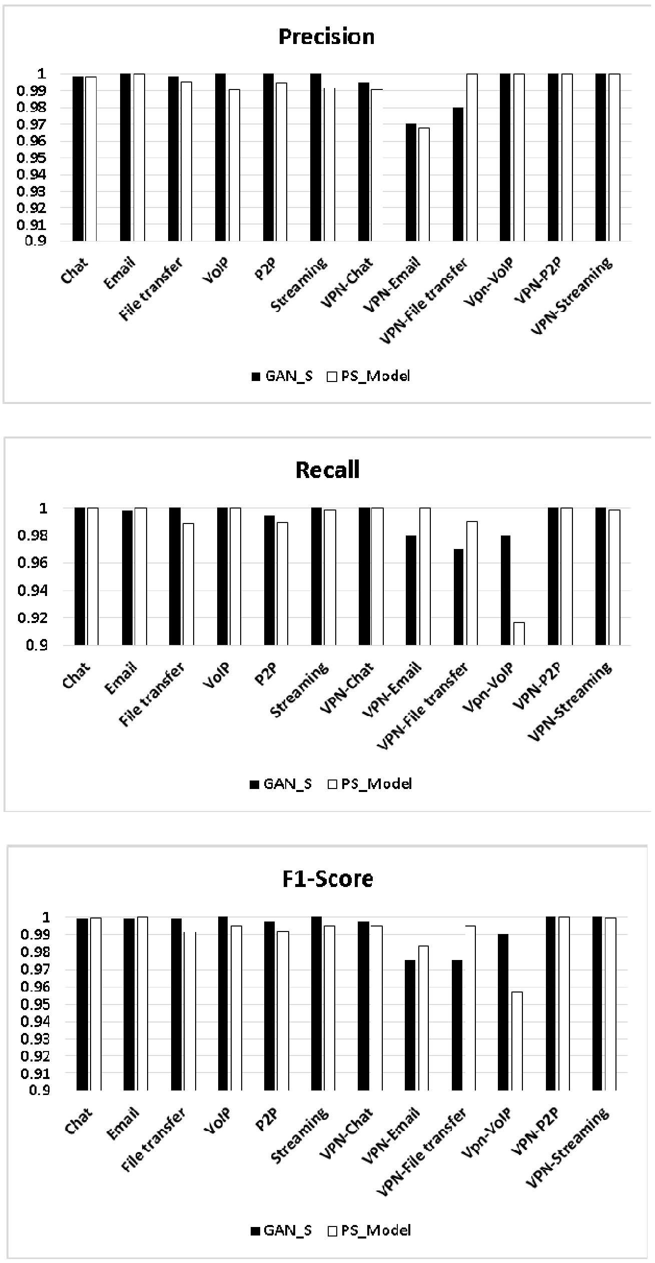


Figure 9 shows the classification performance of model GAN_S and model PS_Model for different samples, and it can be found that the GAN-S model is better than the HCETC_PS model in classifying most of the categories. Firstly, we focus on the data categories VPN-Email, Vpn-VoIP, and VPN-P2P that have been enhanced using generative adversarial networks. the Vpn-VoIP category is the most difficult category to classify by model HCETC-PS, and it can be observed from the figure that both evaluation metrics, Recall and F1-Score, have been improved by different degrees through data enhancement, by 6.33% and 3.34%, respectively. In addition, there are improvements in the categories of file transfer, VoIP and P2P. This indicates that the method designed in this paper to enhance the data set based on generating adversarial networks is effective. Finally, for the VPN-P2P category, the classification performance of both models is high, with 100% for all three-evaluation metrics, and it can be found that the performance remains unchanged after data enhancement, which indicates that the data generated by the generative adversarial network does not bring adverse effects on the data set. In addition to the above category, the VPN-Chat category is also one of the more difficult categories for the model HCETC_PS to classify. Although this paper did not perform data augmentation on this category, the classification performance of this category also has a considerable improvement, which is because some difficult categories are mistaken as VPN-Chat categories in the classification process, resulting in the degradation of the classification performance of this category.

Figure 9 Sample classification performance comparison



Our models are compared with more established models Wang et al. (2018). Owing to the different configurations of the experimental equipment, we do not compare the prediction time of the models, we compare the three aspects of size, number of parameters and F1 score. The results are shown in Table 4, where we can observe that our method not only has the smallest model size but also has the highest accuracy. Even the size of the teacher model is in the middle of the range. It can be said that our method outperforms the state-of-the-art methods in all measurements.

Table 4 Comparison of the model in this paper with the mature model

	Space	*TotalParams*	*F1-Score*
MLP	178 KB	12943	0.9653
SAE	12681 KB	1359463	0.9882
CNN	1467 KB	182927	0.9843
GAN_T	**4079 KB**	**341388**	**0.9995**
GAN_S	**33 KB**	**3148**	**0.9944**

5 Conclusion and future work

In this paper, we propose a hybrid compression-based network traffic classification model, PS_Model, which applies filter pruning based on knowledge distillation and uses an adaptive temperature function for better classification of network traffic. Compared with the traditional model, our model maintains accuracy and greatly reduces the size of the model, and speeds up the training. Filter pruning makes the teacher model required for knowledge distillation reduce the occupied storage and computational resources. In addition, this paper also designs a data augmentation method using generative adversarial networks. Experiments show that the model GAN_S obtained by training with the data-enhanced data set has the best classification performance and solves the classification difficulty caused by sample imbalance in the classification process.

In the future, we will continue to experiment with the use of other compression methods in combination with knowledge distillation, considering further compression of the model size. In addition, we will pay more attention to the problem of an unbalanced sample size and try to propose a better solution.

References

Akbari, I., Salahuddin, M.A., Ven, L., Limam, N., Boutaba, R., Mathieu, B. and Moteau, S. et al. (2022) 'Traffic classification in an increasingly encrypted web', *Communications of the ACM*, Vol. 65, No. 10, pp.75–83.

Chen, A-T., Liu, P., Hong, D-Y. and Wu, J-J. (2021) 'Accelerate CNN models via filter pruning and sparse tensor core', *Proceedings of the 10th International Symposium on Computing and Networking (CANDAR)*, IEEE, Matsue, Japan, pp.1–9.

Gil, D., Lashkari, A.H., Mamun, M. and Ghorbani, A.A. (2016) 'Characterization of encrypted and vpn traffic using time-related', *Proceedings of the 2nd international conference on information systems security and privacy* (*ICISSP*), pp.407–414.

Gu, X., Tian, H. and Dai, Z. (2021) 'Structured attention knowledge distillation for lightweight networks', *Proceedings of the 33rd Chinese Control and Decision Conference (CCDC)*, IEEE, Kunming, China, pp.1726–1730.

Guo, C.Y. and Li, P. (2021) 'Hybrid pruning for convolutional neural network convolution kernel', *Proceedings of the 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, IEEE, Changsha, China, pp.432–438.

Guo, Y., Xiong, G., Li, Z., Shi, J., Cui, M., Gou, G. (2021) 'TA-GAN: GAN based traffic augmentation for imbalanced network traffic classification', *International Joint Conference on Neural Networks (IJCNN)*, IEEE, Shenzhen, China, pp.1–8.

He, Y.J. and Li, W. (2020) 'Image-based encrypted traffic classification with convolution neural networks', *IEEE Fifth International Conference on Data Science in Cyberspace* (*DSC*), IEEE, Hong Kong, pp.271–278.

Hinton, G., Vinyals, O. and Dean, J. (2015) 'Distilling the knowledge in a neural *Network*', *arXiv:1503.02531 [stat.ML]*. Available online at: http://arxiv.org/abs/1503.02531 (accessed 29 October 2022).

Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P. (2017) *Pruning Filters for Efficient ConvNets*, arXiv:1608.08710 [cs.CV]. Available online at: http://arxiv.org/abs/1608.08710 (accessed on 29 October 2022).

Li, S., Lin, M., Wang, Y., Wu, Y., Tian, Y., Shao, L. and Ji, R. (2022) 'Distilling a powerful student model via online knowledge distillation', *IEEE Transactions on Neural Networks and Learning Systems*, pp.1–10.

Ma, Q., Huang, W., Jin, Y. and Mao, J. (2021) 'Encrypted traffic classification based on traffic reconstruction', *Proceedings of the 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, IEEE, Chengdu, China, pp.572–576.

Park, W., Kim, D., Lu, Y. and Cho, M. (2019) *Relational knowledge distillation*, *arXiv:1904.05068 [cs.CV]*. Available online at: http://arxiv.org/abs/1904.05068 (accessed on 29 October 2022).

Rachmawati, S.M., Kim, D-S. and Lee, J-M. (2021) 'Machine learning algorithm in network traffic classification', *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, Jeju Island, Korea, pp.1010–1013.

Rezaei, S. and Liu, X. (2019) 'Deep learning for encrypted traffic classification: an overview', *IEEE Communications Magazine*, Vol. 57, No. 5, pp.76–81.

Rong, J., Yu, X., Zhang, M. and Ou, L. (2020) 'Soft taylor pruning for accelerating deep convolutional neural networks', *Proceedings of the 46th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Singapore, Singapore, pp.5343–5349.

Salehinejad, H. and Valaee, S. (2022) 'EDropout: energy-based dropout and pruning of deep neural networks', *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 33, No. 10, pp.5279–5292.

Shen, M., Liu, Y., Zhu, L., Xu, K., Du, X. and Guizani, N. (2020) 'Optimizing feature selection for efficient encrypted traffic classification: a systematic approach', *IEEE Network*, Vol. 34, No. 4, pp.20–27.

Wang, P., Li, S., Ye, F., Wang, Z., Zhang, M. (2020) 'PacketCGAN: exploratory study of class imbalance for encrypted traffic classification using CGAN', *Proceedings of the IEEE International Conference on Communications (ICC)*, IEEE, Dublin, Ireland, pp.1–7.

Wang, P., Ye, F., Chen, X. and Qian, Y. (2018) 'Datanet: deep learning based encrypted network traffic classification in SDN home gateway', *IEEE Access*, Vol. 6, pp.55380–55391.

Wang, W., Zhu, M., Wang, J., Zeng, X. and Yang, Z. (2017) 'End-to-end encrypted traffic classification with one-dimensional convolution neural networks', *Proceedings of the International Conference on Intelligence and Security Informatics (ISI)*, IEEE, Beijing, China, pp.43–48.

Wang, Z., Wang, P., Zhou, X., Li, S., Zhang, M. (2019) 'FLOWGAN: unbalanced network encrypted traffic identification method based on GAN', *Proceedings of* the IEEE *International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, IEEE, Xiamen, China, pp.975–983.

Wu, Y. and Zhang, M. (2021) 'Lightweight network traffic classification model based on knowledge distillation', *Web Information Systems Engineering – WISE*, Springer, Cham, pp.107–121.

Yamansavascilar, B., Guvensan, M.A., Yavuz, A.G. and Karsligil, M.E. (2017) 'Application identification via network traffic classification', *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, pp.843–848.

Zheng, Y-J., Chen, S-B., Ding, C.H.Q. and Luo, B. (2022) 'Model compression based on differentiable network channel pruning', *IEEE Transactions on Neural Networks and Learning Systems*, pp.1–10.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J. et al. (2019) 'Discrimination-aware channel pruning for deep neural networks', *arXiv:1810.11809 [cs.CV]*. Available online at: http://arxiv.org/abs/1810.11809 (accessed 29 October 2022).

Zou, Z., Ge, J., Zheng, H., Wu, Y., Han, C. and Yao, Z. (2018) 'Encrypted traffic classification with a convolutional long short-term memory neural network', *Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp.329–334.