Automatic Multi-task Learning System for Abnormal Network Traffic Detection

https://doi.org/10.3991/ijet.v13i04.8466

He Huang, Haojiang Deng, Jun Chen(✉) and Luchao Han
University of Chinese Academy of Science, Beijing, China
chenj@dsp.ac.cn

Wei Wang
University of Science and Technology of China, Hefei, China

Abstract—Since the last decade of the 20th century, the Internet had become flourishing, which drew great interest in the detection of abnormal network traffic. Particularly, it's impossible to manually detect the abnormal patterns from enormous traffic flow in real time. Therefore, multiple machine learning methods are adopted to solve this learning problem. Those methods differ in mathematical models, knowledge models, application scenarios and target flows. In recent years, as a consequence of the technological breakthrough of Web 3.0, the traditional types of traffic classifiers are getting outdated and people start to focus on deep learning methods. Deep learning provides the potential for end-to-end learning systems to automatically learn the abnormal patterns without massive feature engineering, saving plenty of detecting time. In this study, to further save both memory and times of learning systems, we propose a novel multi-task learning system based on convolutional neural network, which can simultaneously solve the tasks of malware detection, VPN-capsulation recognition and Trojan classification. To the best of our knowledge, it's the first time to apply an end-to-end multi-task learning system in traffic classification. In order to validate this method, we establish experiments on public malware dataset CTU-13 and VPN traffic dataset ISCX. Our system found a synergy among all these tasks and managed to achieve the state-of-the-art output for most of the experiments.

Keywords—Machine learning, Automatic learning systems, Multi-task learning, End-to-end learning, Network anomaly detection

1 Introductions

In recent years, automatic learning systems have been widely adopted to filter and classify network traffic, especially for virus, intrusions, malwares, VPN-capsulation traffics or other abnormal traffics [1]. The systems help our network automatically learn the intricate patterns of malicious flows so as to protect our network security. To cope with the growing network security demand, researchers applied deep learning-based methods to enhance the learning efficiency. However, though the hotspot tech-

nology has largely improved the performance of few traffic detection tasks such as malware detection, other tasks such as VPN recognition, Trojan classification and sophisticated classification for different application flows are still challenging as a result of complicated network environment. Furthermore, all those tasks are usually dealt with in separate systems, making it difficult to integrate them into one end-to-end system. For instance, a practical learning system for Trojan classification needs to ensure that its input is malicious before it can start classification. Lastly, integration of different learning and detecting tasks can save much learning time for reduplicative contents, so that we are capable to detect more traffics in real time and prevent further potential damage to the network. Hence, we should integrate the learning of different tasks and at the same time guarantee each task's detection rate.

In this paper, we propose a novel learning system which simultaneously deals with the tasks of malware detection, VPN-capsulation recognition and Trojan classification. We select these three tasks because they cover a mass of network security problems. It is proven that joining correlated learning tasks together is capable to improve the performance of each tasks [2][3]. Consequently, with adequate design of the proposed learning system, we can expect a better learning performance. To solve those learning problems in one time, we train a jointly CNN in a multi-task learning framework shown in Fig. 1[4], so that lower layers' parameters could be shared by all three tasks. In this way, the lower common layers learn the public knowledge, while the upper individual layers learn their specific knowledge. So, our learning system is capable to cut down over-fitting value, and get robustness on each learning task. This model can simultaneously deal with the tasks and only demand the memory for an individual CNN model rather than all CNN for each task, thus greatly saving both time and memory for further learning and detecting operations.

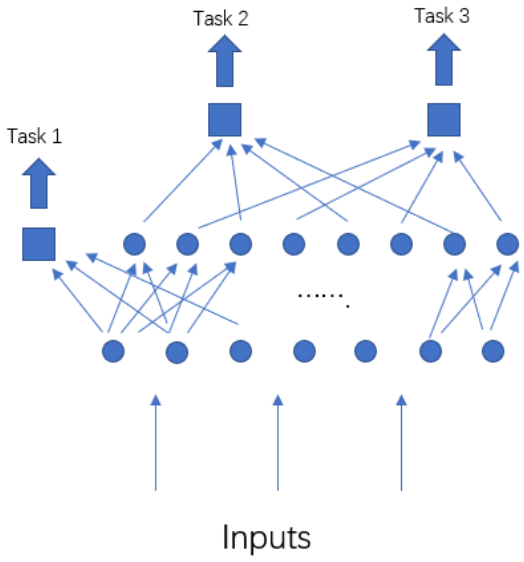


Fig. 1. A general multi-task learning framework based on CNN architecture

Our work mainly makes contributions as follows:

1. We first applied a multi-task learning system in the founding of raw traffic detector to save training time and memory size. Our approach does not need prior knowledge to function, ensuring the stability of learning.
2. We implement a novel CNN architecture that is capable for simultaneous learning of different tasks, and prove its feasibility.
3. We reach state-of-the-art performances on most experiments for all three tasks.

The rest part of our paper is arranged as follows. Section II puts forward a summary of recent related work. Section III displays the multi-task learning system and the theoretical proof of its feasibility. Section IV presents the experimental results for all tasks on the given public datasets. Finally, Section V puts forward the conclusions and future work for this study.

2 Related Work

The concept of multi-task learning (MTL) was first proposed and analyzed in detail by Caruana [4]. Issues relevant to both machine learning and MTL have been researched upon in last several years, and most of them are focused on problems of Computer Vision. In earlier stage, Zhu et al. [7] proposed a joint learning project in face detection, pose estimation and landmark localization in wild environment. They successfully achieved an epoch-making performance on saving learning time and detecting rate for dynamic picture processing. They used a model consisting of multiple learning trees with shared pool for each task. In the next few years, Levi [8]and Ehrlich [9] extended MTL to age estimation, facial attributes, motion prediction and other advanced tasks. However, MTL's application is still very few in many other fields, including network anomaly detection.

Remarkable researches have been established for individual tasks on network anomaly detection, but they still have disadvantages as following. Buczak [1]made a large-scale survey on the application of traditional machine learning approaches in anomaly detection, stimulating many researchers to work on this field, but few approaches were implemented on artificial neural network and MTL was not applied yet. Zhang [10] 's learning system for all kinds of anomaly traffic is memory inefficient. As for the most commonly used machine learning approaches, Naïve Bayes and SVM, Xie [11] and Feng [12] provided state-of-the-art performance using them. However, the two-aforementioned abnormal traffic learning systems are not efficient enough for practical use in many scenes. To avoid the above disadvantages of traditional machine learning approaches, the present study will focus on the application of deep learning architectures.

As for malware detection, Kolosnjaji [13] provided a prototype of deep learning system using raw data, and Wang [14] designed an adversarial network-based method against adversarial malware samples in real network environment. Their work indicates that CNN architecture is efficient and practical in malware detection.

The task VPN-capsulation recognition has only attracted a handful of interests, due to lack of validate training data and researchers' general anti-authority awareness. Studies [15] made use of flow and packet features respectively. They all applied C4.5 decision tree as the learning approaches and achieved detecting precision of nearly 90%.

Trojan (or other types of malware) classification is a multi-class problem that few studies' accuracy was high enough for real time use until deep learning approaches have been put into use recently [18].

We aim to integrate the three aforementioned learning tasks in one CNN architecture, so as to get at least the same learning ability as each task works individually.

3 Methodology

3.1 Datasets

In order to deal with the aforementioned three tasks in raw data, we choose CTU-13[5] and ISCX[6] dataset for our study. CTU-13 is a dataset established in 2011 by network security researchers of the Czech Technical University, containing various kinds of famous raw malware traffic since then. As displayed in Table 1. , eight typical types of malware in CTU-13 are chosen to evaluate our learning system. CTU-13 provides all the data labeled in pcap format, which is easy to set up supervised learning.

Table 1. Data Selected From CTU-13

CTU Num	Malware Name	Description
43	Neris	famous malware, captured in Aug 2011
114-3	Emotet	banking trojan malware, captured from Apr to Jun 2015
116-1	Kazy	widely spread trojan, captured in May 2012
119-2	Geodo	banking trojan malware, captured from Apr to Jun 2015
127-1	Miuref	Windows targeted trojan, captured in Jun 2015
142-1	Shifu	Japanese banking trojan, captured in Sep 2015
147-1	Avzhan	DDoS bot, captured in Sep 2015
158-1	Tinba	Windows targeted banking trojan, captured in Apr 2016

ISCX was created by Draper-Gil et al in 2015, containing 7 kinds of commonly used traffic and their VPN-encapsulated forms. So it has 14 class of traffic in total. We abandon the raw traffic with no labels and choose eight distinct kinds of traffic for our study (Some kinds of traffic can be classified into different classes) as shown in Table 2.

In the following step, we treat each piece of data as an array and convert them into cubic grey-scale map, which is easy for CNN's learning. According to Wang [18], traffic representation of session + all is the most efficient for the neural network learning, in which a "session" includes both directions of flows and All implies the consid-

eration of data of all layers. We then set the size of each piece of data unified to 32*32. We found the mean length of all the piece of data is less than 600 bytes and the standard deviation of their length is 78. Thus, it is safe to conclude that if we take the first 32*32 bytes of each piece, we are sure to get full knowledge of over 99% of traffic data. If the piece's size is shorter than 32*32, then 0x00s will be added in the end to complement it to 32*32 bytes. Next, we transformed the image-like 32*32 bytes traffic into IDX format files, as shown in Fig. 2 and Fig. 3. Furthermore, Fig. 4 implies that the same class of data will have inherent relationship and share the same pattern, making the image-based learning more reasonable.

Table 2. Data Selected from ISCX

Traffic Style	Content
VPN-Chat	AIM, Facebook, ICQ and Skype
Chat	
VPN-Email	Gmail
Email	
VPN-Stream	Netflix, Spotify, Sopcast and Skype
Stream	
VPN-P2P	Bittorrent, and μTorrent
P2P	

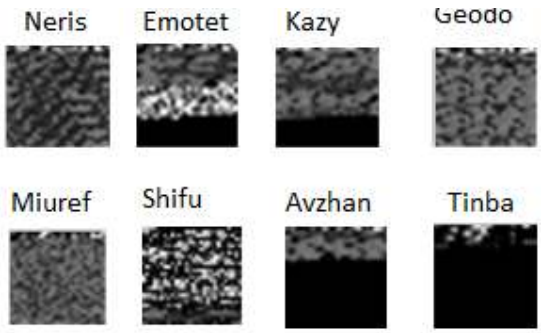


Fig. 2. Transformed Traffic Data from CTU-13

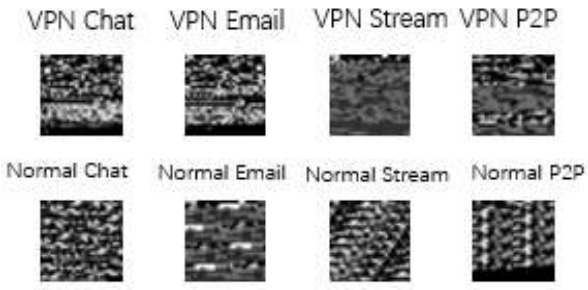


Fig. 3. Transformed Traffic Data from ISCX

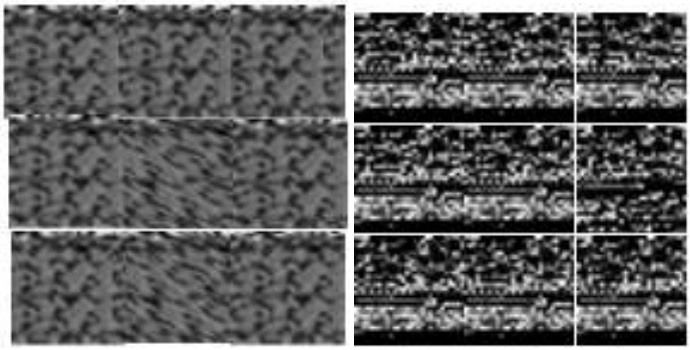


Fig. 4. Consistency in the same class of "Geodo" and "VPN-Email"

3.2 Deep Learning Architecture

In this section, we set up a Multi-Task Learning architecture based on CNN. Before designing the whole architecture, we establish a small experiment to see how many convolutional layers can reach the highest learning efficiency. We build up five simple CNNs with 1 to 5 convolutional layers respectively, and set 20,000 pieces of CTU-13 32*32 traffic data as training input. We only test the simplest 2-class task, namely malware detection, on it. As shown in Fig. 5 we compare two output forms, one is fussing each layers' output as the BIG arrow below, whereas the other only takes advantage of the last layer as the small arrow on the right.

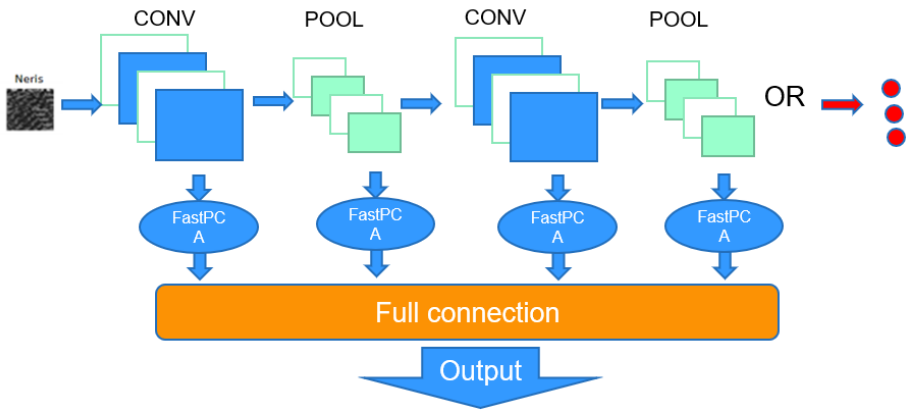


Fig. 5. Simple CNNs for Testing and 2 Convolutional Layers for Example

Fig. 6 demonstrates that when the number of layers comes to two, the learning efficiency reaches the peak. At that point, the detecting rate starts to convergent to the highest limitation of CNN, nevertheless, the time consumption still scales linearly as the layer number grows. So, we can design our architecture with three convolutional layers, two for the 2-class tasks and the extra layer for the Trojan classification.

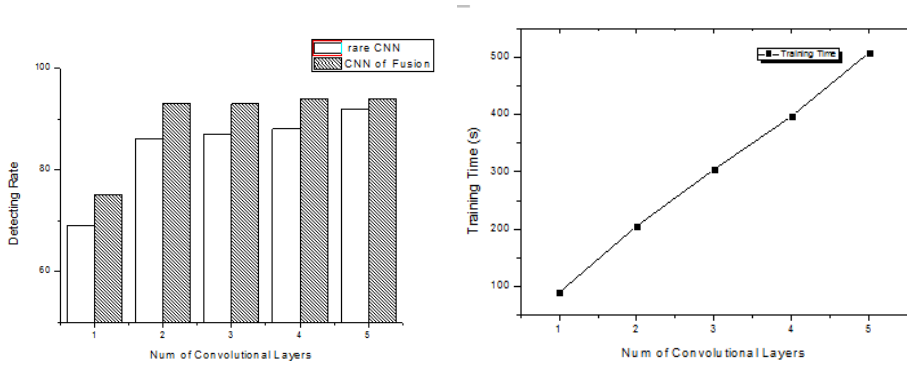


Fig. 6. Learning Ability Analysis and Learning Time Efficiency Analysis

Our architecture consists of four convolutional layers and 2 max pooling layers, as shown in detail in Table 3. It has three outputs, each for one learning task. Fig. 7 displays the whole architecture, and marks all the six layers mentioned in Table 3. . As we can see in Fig. 7, lower convolutional layers, namely layer 1 and 3, have their parameters shared by all tasks. All the final pattern maps are as large as 8*8 bytes, and serve as the inputs of 1024-dimensional Full Connection (FC) layers.

We merge the output of layers 1 and 3 for the learning of malware detection and VPN recognition, since they are 2-class learning tasks that depend more on the shallow image information. Two pooling layers, namely layer 2 and layer 4, are added to maintain the consistence of the image patterns learned by layers 1 and 3. Layer 6 is set up to acquire a fixed pattern map of 8*8 bytes. Next, we set a 1024-dimensional FC layer to produce a general representation for the 2 tasks. Finally, the two specific tasks flow to different output FC separately.

The remainder of these learning tasks goes along the backbone into layer 5. Then the data will input into a 1024-dimensional FC layer. Finally, it is fed to a 9-class output layer, classifying its Trojan type (for CTU-13) or application type (for ISCX).

Table 3. Parameters of MTL Model

Layer Number	Layers	Input Size	Kernel Size	Padding	Stride	Output Size
1	Conv+ReLU	32*32	32*5*5	2	1	32*32*32
2	MAX pool	32*32*32	2*2	0	2	32*16*16
3	Conv+ReLU	32*16*16	64*5*5	2	1	64*16*16
4	MAX pool	64*16*16	2*2	1	2	64*8*8
5	Conv+ReLU	64*8*8	128*3*3	1	1	64*8*8
6	Conv+ReLU	32*16*16	64*3*3	1	1	64*8*8

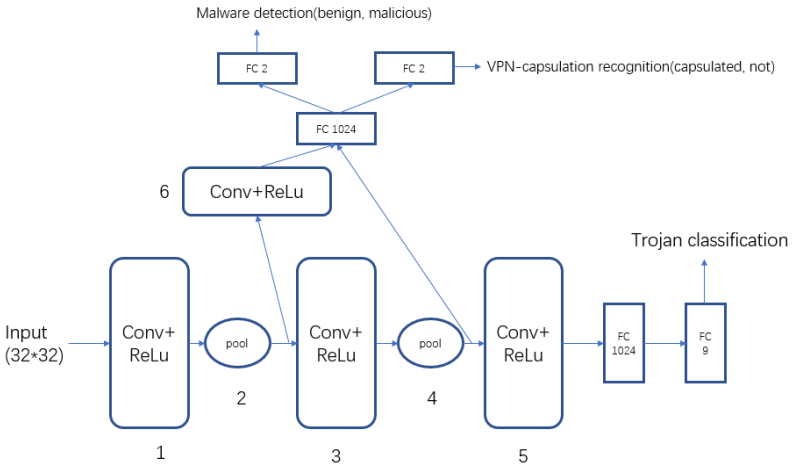


Fig. 7. MTL Architecture which can deal with 3 tasks simultaneously

3.3 Multi-task Learning's Feasibility

Due to the insufficiency of real time testing data, the overfitting problem is difficult to directly deal with in our experiments. Overfitting means the trouble that our learning system can exactly learn all the information on training dataset but cannot work on test datasets that are randomly given. The more complicated a system is, the more serious overfitting it may be faced with. The most common and valid method to cope with overfitting is regularization, which adds a regularizing function to the loss function to simplify its high-dimensional structure.

The feasibility of our MTL approach relies on whether it will be more overfitting than individual learning tasks. Set the input as $\mathrm{I} = (\vec{\mathrm{x}}, \mathrm{y})$, where x serves as patterns and y is the labels. θ_s is the set of shared parameters with the optimal value of Θ_s while θ_i is the set of distinct parameters for each task i with the optimal value of Θ_i.

As for individual learning, the cost function J can be described as equation (1):

$$J(\theta_s, \theta_i) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\theta_s}(\vec{x}) + h_{\theta_i}(\vec{x}) - y\right) = f(I; \theta_s, \theta_i) \tag{1}$$

Thus, the optimal value can be expressed as equation (2):

$$(\Theta_s, \Theta_i) = \mathrm{argmin}_{\theta_s,\theta_i} f(I; \theta_s, \theta_i) \tag{2}$$

As for MTL, through minimizing the weighted sum of J we can acquire Θ_i. and Θ_s. Weight for θ_i is α_i

$$(\Theta_s, \Theta_i) = \sum_{j=1}^{n} \alpha_j f(I; \theta_s, \theta_j) \tag{3}$$

For i, we only need to focus on the contribution of θ_i and ignore other tasks, since other tasks don't use the same convolutional parameters. So we can regard other tasks' loss function as a regularizing function $R_i$as equation (4). Therefore, optimal

shared parameters are restricted in a smaller solution domain, making the learned Θ_s equally valid for other tasks. As a result, we can safely apply MTL in our learning system, decreasing overfitting and capable to achieve higher learning performance. Thus, specific loss functions of each task can work individually during the learning process.

$$(\Theta_s, \Theta_i) = \text{argmin}_{\theta_s,\theta_i}[\alpha_i f(I;\theta_s,\theta_i)] + \sum_{j\neq i}^{n} \alpha_j f\left(I;\theta_s,\theta_j\right)$$
$$= \text{argmin}_{\theta_s,\theta_i}[\alpha_i f(I;\theta_s,\theta_i)] + \lambda R_i(I;\theta_s) \quad (4)$$

3.4 Learning Process

Our automatic learning system consists of three specific sub-networks with different input data but the same lower layers. Malware detection and VPN recognition are treated in a common FC layer because they are relatively simple 2-class problems. Trojan classifying is a bit more complicated due to its variety of traffic, thus its data will be fed to an extra convolutional layer (layer 5). All the learning tasks are implemented end-to-end. Next, we will discuss their different loss functions.

Malware Detection. This is a comparatively simple binary classification since its data size is smaller, contains less information and is easier for learning. Cross-entropy based loss function is suitable for this task, which is shown in equation (5):

$$f_1 = \text{mlog}(\frac{1}{p_m}) + (1-m)\text{log}(1-\frac{1}{p_m}) \quad (5)$$

Where, m=1 for malicious and m=0 for benign. *Pm* is a dynamic predicted probability that the piece of data is malicious.

VPN Recognition. This task aims to recognize VPN-capsulation traffic in the real network environment but in some scenarios the boundary is not explicit. In this situation network administrators only need to determine the degree of suspiciousness. Therefore, it can be treated as a binary classification problem or a regression problem. The former one is shown in equation (6) and the latter in equation (7):

$$f_{21} = \text{vlog}(\frac{1}{p_v}) + (1-v)\text{log}(1-\frac{1}{p_v}) \quad (6)$$

Where, v=1 for VPN-capsulation and v=0 for benign. p_v is a dynamic predicted probability that the piece of data is VPN-capsulated.

$$f_{22} = \lambda\frac{1}{2}(y-s)^2 + (1-\lambda)[1-\exp\left(\frac{-(y-s)^2}{2\sigma^2}\right)] \quad (7)$$

Where, y is the predict degree of suspiciousness, s is the actual degree of suspicious and σ^2 is the variance of the degree. $\frac{1}{2}(y-s)^2$ is Euclidean loss, while $1-\exp(\frac{-(y-s)^2}{2\sigma^2})$ is Gaussian loss. Euclidean loss works better when initial prediction is far away from the actual value, while Gaussian loss drops more rapidly around it as σ^2 was given. Network administrators need to alter the value of λ to get better learning performance.

Trojan Classifying. We apply a multi-class cross-entropy loss function for this task, which is shown in equation (8):

$$f_3 = \sum_{i=0}^{9} t_i \log\left(\frac{1}{p_i}\right) \tag{8}$$

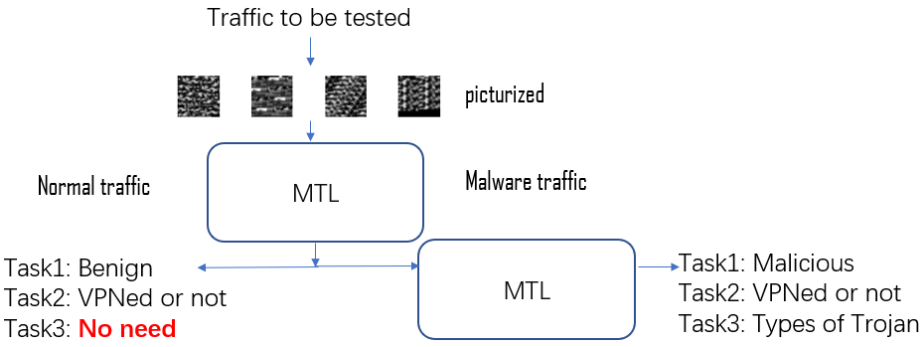
We set the total loss F as shown in equation (9):

$$F = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3 \tag{9}$$

Where, α_i is the corresponding weight for task i, and f_2 is chosen between f_{21} and f_{22} according to specific application scenarios. Since regression tasks tend to get smaller loss gradient than classification tasks during the learning, we will set α_2 higher than other α_1 once $f_2 = f_{22}$.

3.5 Validating Process

We design a 2-step end-to-end validating scheme for validating process as shown in 0, which can simplify the validating for Trojan classification because this kind of classification is of insignificance if the target data is benign.



Flow Chart Describing the End-to-end Validating Process of the Proposed MTL

4 Experiments

4.1 Experimental Setup

We used TensorFlow1.4 (established in November 2017, GPU version), and Ubuntu 16.04 64-bit OS as the framework for experiments. The framework was implemented on a quad-core Intel-CORE i7-7700HQ CPU with memory of 8 GB. Additionally, a GPU of Nvidia GTX1070 was applied to accelerate the learning process. We adopted the one-tenth cross validation method to evaluate the learning system. That's to say, we operate the learning-validating process for ten turns, and at each turn we chose one tenth of the data for validating, while the remainder was used for learning. Further, we applied adaptive moment estimation (ADAM) as the optimizer.

We set the batch size as 200, and the total size of input training dataset was 96,000, with 6,000 pieces of data for each type of traffic (8 for Trojan, 4 for VPNed benign traffic, and 4 for non-VPNed benign traffic). We select another set of 64,000 pieces of data as the test dataset, with 4,000 pieces of data for each type of traffic.

4.2 Learning for Malware Detection

To evaluate our MTL architecture's learning ability for task 1 in the elemental level, we take the Single-task Learning (STL) part into study. According to Fig. 7, the STL part for task 1 can be displayed in Fig. 8 which has only one output layers of 2 classes. The data from ISCX is set as benign input, while the data from CTU-13 is set as malicious input. In consideration of the input style, we divide the whole experiment into two sub-experiments, "Separated" and "Integrated". The "Separated" sub-experiment is conducted by respectively learning eight malwares in turn, whereas the "Integrated" one is conducted by taking all kinds of malwares to learning process in one time.

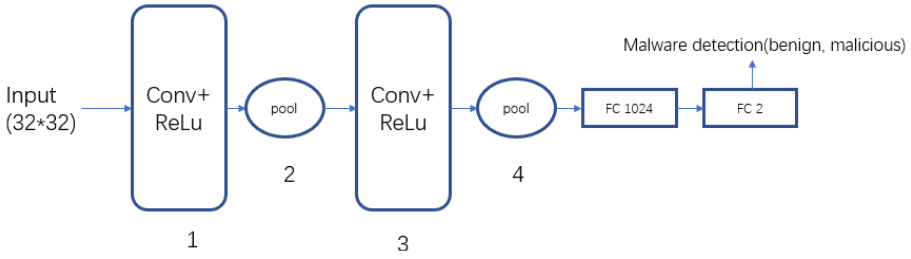


Fig. 8. Single Task Learning for Task 1

In the "Separated" sub-experiment, each type of flow is learnt by three models, HDCN, STL and the proposed MTL. Therefore, we get 24 learning processes and the same number of precision-recall pairs, as shown in Fig. 10 and **Error! Reference source not found.**. Fig. 10 shows that our STL and MTL have achieved observably high precision than HDCN. MTL gains nearly the same score as STL, which means that our MTL structure can work as efficient as individual detectors. As for the types with sharper patterns, like Neris and Emotet, MTL gains the greatest advantage. The dark and dim Avzhan makes it difficult for MTL to show its strength. Tinba is even a darker one, but its pattern is obviously sharp, so STL and MTL still gain great advantages. In **Error! Reference source not found.**, MTL maintains its great advantage while STL fails to do so. The reason is that our MTL takes full advantage of the information of all convolutional layers.

In the "Integrated" sub-experiment, all eight kinds of malwares are required to be detected at the same time, hindering the models' precision. In this case, the model that suffers from the least performance loss gets the highest robustness. In Table 4. , "Average" implies the mean value of each model's performance in Fig. 10 and Fig. 11. From the table we can see that our MTL's performance is the most stable. Its fall on precision is almost negligible and obtains the largest rise on recall

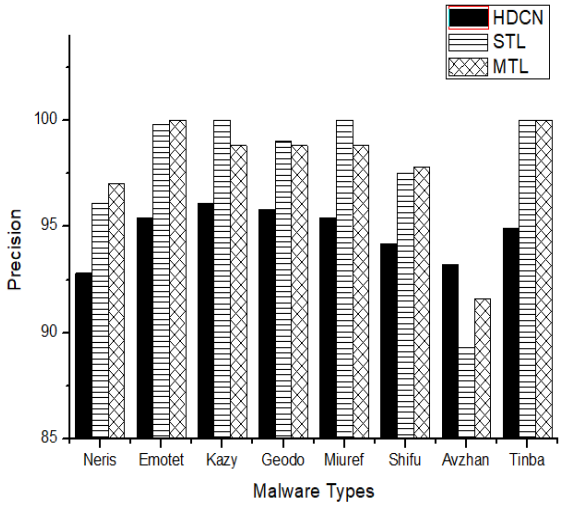


Fig. 9. Precision for Each Individual Malware in Different Scenes

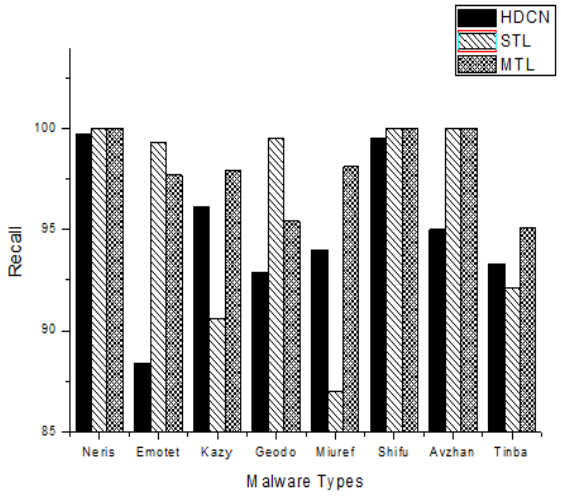


Fig. 10. Recall for Each Individual Malware in Different Scenes

Table 4. Performance Comparison between Separated Sub-experiment and Integrated Sub-experiment

Traffic	Precision			Recall		
	HDCN	***STL***	***MTL***	***HDCN***	***STL***	***MTL***
Average	94.7	97.7	97.9	94.9	96.3	98.0
All	93.8	96.1	97.8	95.2	96.8	99.0
Diff.	-1.1	-1.6	-0.1	+0.3	+0.5	+1.0

4.3 Learning for VPN-capsulation Recognition

In this subsection, we only compare our MTL with the state-of-the-art method on ISCX dataset. Since our MTL is deep structure based, we unsurprisingly achieve much better performance than the decision tree-based model. Our MTL's precision on VPN and Non-VPN traffic are 10.9% and 9.3% higher, whereas its recall on VPN and Non-VPN traffic is 6.8% and 10.0% higher respectively. The average improvement is 9.25%. The state-of-the-art also provides some experiments of multi-class learning, which is an advantage of C4.5 decision tree. However, their accuracy, precision and recall values still cannot exceed the performance of our MTL on task 3.

Table 5. Performance Comparison between C4.5 and Our MTL on Task 2

Method	VPN		Non-VPN	
	precision	***recall***	***precision***	***recall***
C4.5	89.0	92	90.6	88.8
MTL	99.9	99.8	99.8	98.8
Diff.	+10.9	+6.8	+9.3	+10.0

We select relevant structure of MTL to validate our opinion, as shown in Fig. 11. The input contains ISCX data only, and this sub-architecture aims to learn the classification information implied in Fig. 3 (8 known types and 1 unknown). The result shown in Table 6. demonstrates that MTL even obtains bigger advantage over the decision tree-based model with an average improvement of 11.6%. Thus, we are able to conclude that our MTL achieves the state-of-the-art performance in the field of VPN data classification.

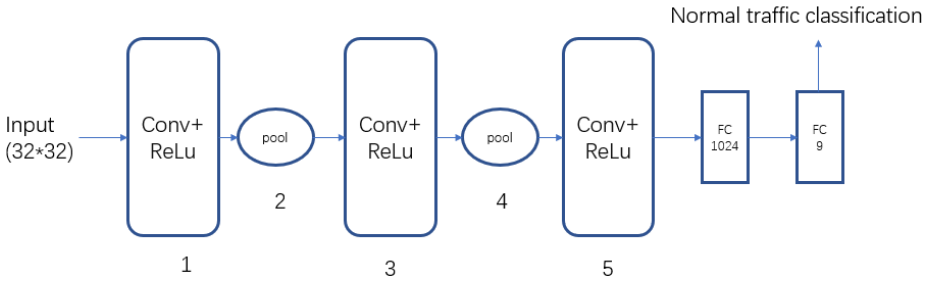


Fig. 11. Multi-class Learning for ISCX Data Only

Table 6. Performance Comparison between C4.5 and Our MTL on Task 2

Method	VPN		Non-VPN	
	precision	***recall***	***precision***	***recall***
C4.5	78.2	81.3	84.3	79.3
MTL	90.9	92.8	95.2	90.6
Diff.	+12.7	+11.5	+10.9	+11.3

4.4 Learning for Trojan Classification

In our last experiment, we take the performance of WCM, HDCM and our proposed MTL into comparison. All the three learning models are neural network based and designed to be multi-classifiers. WCM is lighter CNN model that only has two convolutional layers and deals with a smaller size of 24*24 picturized data. HDCM is a hybrid of FNN (Feedforward Neural Networks, a simply connected network similar to FC layer) and CNN, in which FNN is used to deal with formatted data (such as 5-tuples), thus decreasing the overall complexity of learning. All these models are able to get accuracy, precision and recall of higher than 90%. In order to differentiate their performance, we record the classification result of every type in detail, which is shown in Table 7. Further, we record the training time for each learning model. We list the mean performance data in.

Table 7. Comparison of Our MTL and State-of-the-art Models

(A) By WCM

Actual	*Nor*	*Ner*	*Emo*	*Kaz*	*Geo*	*Miu*	*Shi*	*Avz*	*Tin*
Normal	**.994**	0	0	.002	.004	0	0	0	0
Neris	.012	**.963**	0	.014	0.011	0	0	0	0
Emotet	0	0	**.998**	0	0	0	0	.002	0
Kazy	.018	0	0	**.927**	0	.043	0	.012	0
Geodo	0	0	0	0	**1**	0	0	0	0
Miuref	0	0	0	0	0	**1**	0	0	0
Shifu	0	0	.001	0	0	0	**.999**	0	0
Avzhan	.002	0	0	.012	0	0	0	**.896**	.090
Tinba	0	0	0	0	0	0	0	0	**1**

(B) By HDCM

Actual	*Nor*	*Ner*	*Emo*	*Kaz*	*Geo*	*Miu*	*Shi*	*Avz*	*Tin*
Normal	**.946**	0	0	.006	.044	.004	0	0	0
Neris	0	**.928**	.052	.010	.008	.002	0	0	0
Emotet	0	.002	**.954**	.044	0	0	0	0	0
Kazy	0	0	.025	**.961**	.004	.010	0	0	0
Geodo	0	0	0	.012	**.958**	.030	0	0	0
Miuref	0	0	0	0	.042	**.954**	.004	0	0
Shifu	0	0	.038	0	.006	.014	**.942**	0	0
Avzhan	0	0	0	0	0	0	0	**.932**	.068
Tinba	0	0	0	.002	0	0	0	.049	**.949**

(C) By MTL(Our Learning Model)

Actual	*Nor*	*Ner*	*Emo*	*Kaz*	*Geo*	*Miu*	*Shi*	*Avz*	*Tin*
Normal	**.986**	0	0	0	.013	.001	0	0	0
Neris	.002	**.970**	0	.018	.002	.008	0	0	0
Emotet	0	0	**1**	0	0	0	0	0	0
Kazy	0	0	0	**.988**	.012	0	0	0	0
Geodo	.006	0	0	0	**.988**	.006	0	0	0
Miuref	0	0	.012	0	.010	**.978**	0	0	0
Shifu	0	0	.012	0	.006	.004	**.978**	0	0
Avzhan	.003	0	0	.003	0	0	0	**.916**	.052
Tinba	0	0	0	0	0	0	0	0	**1**

Table 8. Overall Performance Comparison for Task 3

Measures	WCM	HDCM	Our MTL
Detection Rate (Precision %)	97.5	94.8	99.4
Recall (%)	94.7	94.7	98.0
Training time(s)	25.42	21.75	9.19

Table 9. Confidence Intervals of t-Tests

Measures	μ_1-μ_3	μ_2-μ_3
Detection Rate (Precision %)	(-3.561,1.316)	(-6.711, -1.985)
Recall (%)	(-6.671, -0.482)	(-6.735, -0.281)

Results in Table 8. indicate that our learning model surpasses the remainder ones in both precision and recall. The training time of our MTL is unsurprisingly much shorter than other ones, since it can learn 3 tasks in one time.

In order to theoretically validate our model, we introduced the method of pair-sampled *t-test* with 90% confidence level. This method reveals the inherent relationship of 2 counter samples and tells whether they are essentially different. Since the number of class is 9 and there are 3 learning models, the degree of t-test freedom is $\mathrm{df} = 3 * (9 - 1) = 24$. Let μ be each measure's mean value, and footnotes 1, 2, 3 refer to the WCM, HDCM, and the proposed MTL respectively, we calculated the confidence intervals described in Table 9. , demonstrating that our MTL surpasses both models in recall and outperforms HDCM in precision. What's more, our MTL obtains at least the same performance with WCM in precision. Compared with the latest recently researched models, our MTL is capable to deal with 3 different learning tasks simultaneously without losing learning performance.

5 Conclusions and Future Work

In this paper, we presented a novel Multi-Task Learning system based on CNN for malware detection, VPN-capsulation recognition and Trojan classification. CNN was chosen for the reason that it gets high detection rate when managing instances of thousands of bytes and it's convenient for integrating different tasks on one backbone. In the designing stage of this model, we made parallel experiments to confirm that two convolutional layers are suitable for traffic detection and add one more convolutional layers for multi-class learning. Furthermore, we put forward the mathematical proof for the feasibility of this MTL architecture, proving that it won't get extra overfitting degree by integrating multiple modules. Moreover, we defined appropriate loss functions for each learning tasks respectively, making them efficiently studied by CNN structure. Extensive experiments on several public, large-scaled traffic datasets imply that we have obtained state-of-the-art performances for each learning tasks. The results validate that each individual task can benefit from the shared parameters, which is promising for future study. Further, our MTL can save plenty of learning

time by learning tasks from different domains in one time. The crucial time reduction is significant for the model's application in real-time learning.

Several exciting questions can be raised by our work and we project to continue research in relevant fields. For example, cyber-security and other security problems are calling for efficient intrusion learning methods, our MTL can be extended to these fields and thus ensure public safety. Other questions include how to make the best of the internal structures of different kinds of neural networks so as to optimize learning systems in different scenarios, and how to implement it in various real network environments

6 Acknowledgments

The authors would like to thank the reviewers for their helpful comments and advice. This work has been supported by the First-Action Program of Institute of Acoustics, Chinese Academy of Sciences (SXJH201609).

7 References

[1] Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153-1176. https://doi.org/10.1109/COMST.2015.2494502

[2] Chen, D., Ren, S., Wei, Y., Cao, X., & Sun, J. (2014, September). Joint Cascade Face Detection and Alignment. In *European Conference on Computer Vision*, 109-122. Springer, Cham.

[3] Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. M. and Makhoul, J. (2014). Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *ACL (1)* 1370-1380. https://doi.org/10.3115/v1/P14-1129

[4] Caruana, R. (1998). Multitask Learning. In *Learning to learn*, 95-133. Springer US. https://doi.org/10.1007/978-1-4615-5529-2_5

[5] CTU13 2015. CodeBlue: Stratosphere IPS Project. (2015).

[6] ISCX VPN-nonVPN Encrypted Network Traffic Dataset.

[7] Zhu, X. and Ramanan, D. (2012). Face Detection, Pose Estimation, and Landmark Localization in the Wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2879-2886.

[8] Levi, G. and Hassner, T. (2015). Age and Gender Classification Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* 34-42.

[9] Ehrlich, M., Shields, T. J., Almaev, T. and Amer, M. R. (2016). Facial Attributes Classification Using Multi-Task Representation Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* 47-55.

[10] Zhang, B. Y., Wang, L. and Wang, H. Q. (2017). Information Classification and Applications Research in ICN. *Journal of Network New Media,* 6(1), 19–23.

[11] Xie, Y. and Yu, S. Z. (2009). A Large-scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors. *IEEE/ACM Transactions on Networking (TON)*, *17*(1), 54-65. https://doi.org/10.1109/TNET.2008.923716

[12] Feng, W., Zhang, Q., Hu, G. and Huang, J. X. (2014). Mining Network Data for Intrusion Detection Through Combining Svms with Ant Colony Networks. *Future Generation Computer Systems*, *37*, 127-140. https://doi.org/10.1016/j.future.2013.06.027

[13] Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016). Deep Learning for Classification of Malware System Call Sequences. In *Australasian Joint Conference on Artificial Intelligence*, 137-149. Springer International Publishing.

[14] Wang, Q., Guo, W., Zhang, K., Ororbia II, A. G., Xing, X., Liu, X. and Giles, C. L. (2017). Adversary Resistant Deep Neural Networks with an Application to Malware Detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1145-1153. ACM.

[15] Wang, D., Zhang, L., Yuan, Z., Xue, Y. and Dong, Y. (2014). Characterizing Application Behaviors for Classifying P2p Traffic. In *Computing, Networking and Communications (ICNC), 2014 International Conference on* 21-25. IEEE. https://doi.org/10.1109/ICCNC.2014.6785298

[16] Coull, S. E. and Dyer, K. P. (2014). Traffic Analysis of Encrypted Messaging Services: Apple Imessage and Beyond. *ACM SIGCOMM Computer Communication Review*, *44*(5), 5-11. https://doi.org/10.1145/2677046.2677048

[17] Aghaei-Foroushani, V. and Zincir-Heywood, A. N. (2015). A Proxy Identifier Based on Patterns in Traffic Flows. In *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, 118-125. https://doi.org/10.1109/HASE.2015.26

[18] Wang, W., Zhu, M., Zeng, X., Ye, X. and Sheng, Y. (2017). Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In *Information Networking (ICOIN), 2017 International Conference on*, 712-717. https://doi.org/10.1109/ICOIN.2017.7899588

8 Authors

He Huang, a Ph.D. candidate in signal and information processing from Institute of Acoustics, Chinese Academy of Sciences (IACAS). His research interests include Network Security and AI (Artificial Intelligence). (huangh@dsp.ac.cn).

Haojiang Deng, the vice director of National Network New Media Engineering Research Center, IACAS. His research interests include network communications and new media technology (denghj@dsp.ac.cn).

Jun Chen, a professor in IACAS. Her research activities have been concerned with information system security, network protocol analysis, and streaming media processing. (chenj@dsp.ac.cn).

Wei Wang, a Ph.D. candidate at the University of Science and Technology of China, Hefei, China. His research interests include machine learning and cyberspace security. (ww8137@mail.ustc.edu.cn)

Luchao Han, a Ph.D. candidate in signal and information processing from IACAS. His research interests include computer network and deep learning. (Email: hanlc@dsp.ac.cn).

Article submitted 20 February 2018. Resubmitted 09 March 2018. Final acceptance 25 March 2018. Final version published as submitted by the authors.