APPENDIX

RFG-HELAD: A ROBUST FINE-GRAINED NETWORK TRAFFIC ANOMALY DETECTION MODEL BASED ON HETEROGENEOUS ENSEMBLE LEARNING

Part 1: The algorithmic part of the model.

Our algorithmic model consists of three parts. The first is the training of the RFG-HELAD model. Then there is online anomaly detection. The last part is the retraining of the model.

Algorithm 1: Training of RFG-HELAD model

Input: Training data set $Data_{tr}$, Contrastive learning parameters suploss=0.7, Learning rate of DNN lr1=1e-2, Fourier parameters n_fft=52, hop_len=64, Noise vector dimension of GAN z_dim=100, Weights of the two generators lamda=0.1, Learning rate of GAN lr2= 0.0001, Hidden layer vector dimension (consistent for DNN and GAN) hid_dim=64, Nearest neighbor parameter k=1;

Output: Trained RFG-HELAD model, unknown attack threshold ξ, Hid_{list}.

Step 1: Train DNN+CL and GAN model.
for P_i *in* $Data_{tr}$ **do**
  $\triangleright$ P_i *is the packet in the training set.*
  $x_1 \leftarrow$ *Feature extraction*(P_i); ∇ *Here we can use Kitsune's feature extraction method.*
  $x_2 \leftarrow Fit\_transform(x_1)$;
  $x_3 \leftarrow$ *Fourier transform*(x_2); ∇ *The sixth and seventh rows only run under the adversarial defense.*
  $x_2 \leftarrow x_2 \oplus x_3$; ∇ $\oplus$ *is defined as the splicing operation, [0,1]⊕[2,3] = [[0,2],[1,3]].*
  Save x_2 to the Tr_{x_2} list
  Train DNN+CL(x_2) model
  Train GAN(x_2) model
end
Step 2: DNN+CL's adversarial training. ∇ *This step is only run in the presence of an adversarial defense.*
for x_i *in* Tr_{x_2} **do**
  $x_4 \leftarrow$ *FGSM/PGD*(x_i);
  Train DNN+CL(x_4) model
end
Step 3: Construction of distance matrix.
for x_i *in* Tr_{x_2} **do**
  $Hid_1 \leftarrow (DNN + CL)_h(x_i)$; ∇ $(DNN + CL)_h$ *represents the operation of extracting the hidden layer feature vectors of the DNN+CL model.*
  $Hid_2 \leftarrow$ *(GAN-*$D_1)_h(x_i)$; ∇ *extraction of hidden layer features using* D_1 *of GAN.*
  $Hid_{temp} \leftarrow Hid_1 \oplus Hid_2$;
  Save Hid_{temp} to the Hid_{list} list
end
Construct the distance matrix D_{TV} based on original training set Hid_{list} and Faiss (See Figure 6). Select 20% of the training set {TR} as the validation set {VA}. {VA} and {TR-VA} form a distance matrix D_{TV}, and the k-th smallest distance is selected by row to form a list {score}.
Step 4: Threshold ξ selection.
The {score} is sorted in ascending order to obtain the {score'}, and the threshold value is taken with reference to the distance value of the 95%*len({score'}) in {score'}.
return RFG-HELAD model, unknown attack threshold ξ, Hid_{list}

Algorithm 2: Online network anomaly detection

Input: Testing data set $Data_{te}$, trained RFG-HELAD model, Fourier parameters n_fft=52, hop_len=64, Hidden layer vector dimension (consistent for DNN and GAN) hid_dim=64, Nearest neighbor parameter k=1, Hid_{list}, unknown attack threshold ξ;

Output: Alarm messages for network attacks.

1 Begin test.
2 **for** P_i *in* $Data_{te}$ **do**
3 $\triangleright$ P_i *is the packet in the testing set.*
4 $x_1 \leftarrow$ *Feature extraction*(P_i);
5 $x_2 \leftarrow Fit\_transform(x_1)$;
6 $x_3 \leftarrow$ *Fourier transform*(x_2); ∇ *The sixth and seventh rows only run under the adversarial defense.*
7 $x_2 \leftarrow x_2 \oplus x_3$; ∇ $\oplus$ *is defined as the splicing operation, [0,1]⊕[2,3] = [[0,2],[1,3]].*
8 $Hid_1 \leftarrow (DNN + CL)_h(x_i)$;
9 $Hid_2 \leftarrow$ *(GAN-*$D_1)_h(x_i)$;
10 $Hid_{temptest} \leftarrow Hid_1 \oplus Hid_2$;
11 Calculate the distance between $Hid_{temptest}$ and each element in list Hid_{list}. Select the k-th smallest distance value as scoretest.
12 **if** $scoretest > \xi$ **then** Alert $(K+1)$-th unknown attacks **else** Alert the known attacks $(DNN + CL)(x_2)$ ∇ $(DNN + CL)(x_2)$ *represents the result of K classification.*
13 **end**
14 **return** Alarm messages for network attacks

Algorithm 3: Retraining of our model

Input: Training data set $Data_{tr}$, Newly discovered attack data set $Data_{trnew}$, number of newly discovered attacks ND;

Output: Trained RFG-HELAD model, unknown attack threshold ξ, Hid_{list}.

1 Change the output layer class of the DNN+CL from K to $K + ND$
2 Add $Data_{trnew}$ to the $Data_{tr}$ data set
3 Call Algorithm 1 to retrain
4 **return** RFG-HELAD model, unknown attack threshold ξ, Hid_{list}

Algorithm 1 is the training of the RFG-HELAD model. The first step is to specify the various training parameters for the model. For example, we set the contrastive learning parameters to 0.7. The specific steps are:

Step 1: Train DNN and GAN model.

Step 2: DNN's adversarial training.

Step 3: Construction of distance matrix.

Step 4: Threshold selection.

where lines 6-7, and 12-15 are only run when there is an adversarial defence requirement. When the model is trained, we obtain the trained RFG-HELAD model, the unknown attack threshold, and the set of hidden features of the training set.

Algorithm 2 is online anomaly detection. We use the output obtained during training as input for the online detection and keep the necessary parameters constant. Specifically, the extracted features are fed directly into the discriminator D_1 and the DNN+CL, which generates a hidden layer feature of the current sample. This hidden layer feature is then compared to the training set to calculate the distance vector. A threshold is used to determine whether the attack is unknown. This judgement is used to correct the K classification result of the DNN and to obtain the final $K + 1$ classification result. Finally, an alert message is issued based on the type of anomaly.

Algorithm 3 is the retraining of the model. We know that the retraining of the model can be of great help to the performance of the model. We first adjust the output class of the DNN, and then add the relabelled data to the original training set. Finally, Algorithm 1 is called again for training, which in turn the new model is obtained.

Part 2: Toy example of our RFG-HELAD model.

To better understand our model, we explain it through the toy example. Fig. 2 shows data flow diagram of RFG-HELAD and RFG-HELAD* models (toy example). We assume that the feature dimension is 3 after feature extraction and the feature dimension of the hidden layer retention vector is 2. The content of Fig. 1 can be analysed in three parts. The first part Ⓐ presents examples of the training and test sets. The second part Ⓑ is the generation of the final feature vector. In order to distinguish whether there is an adversarial attack or not, this part is discussed in two cases. The third part Ⓒ goes from the hidden features to the final detection. We present the scenario without adversarial attacks, and the scenario with adversarial attacks can be analysed similarly.

The packet Tr2 in the training set goes through the second part of the features to finally get the vector [0.16, 0.84, 0.40, 0.60]. [0.16, 0.84] is generated by DNN+CL model, and [0.40, 0.60] is generated by our GAN model. For the packet Te3 in the test set, this is an unknown attack that do not appear in the training set. Its final feature form [0.22, 0.30, 0.14, 0.16] can be obtained. For K classification, its label is assigned as [1]. This is misclassified.

The final feature is entered into the deep k-nearest neighbour and is labelled as [T]. This indicates an unknown attack. Te3 is automatically labelled as class $K+1$ by our model, which means the label is [2]. After the network security administrator checks for unknown attacks, its final label is [3], which is convenient for retraining later. As can be seen, there are a total of three packets in the test set, and the accuracy of the K classification at this time is 33.3%. The accuracy of the $K+1$ classification after the unknown attack correction is 100%.

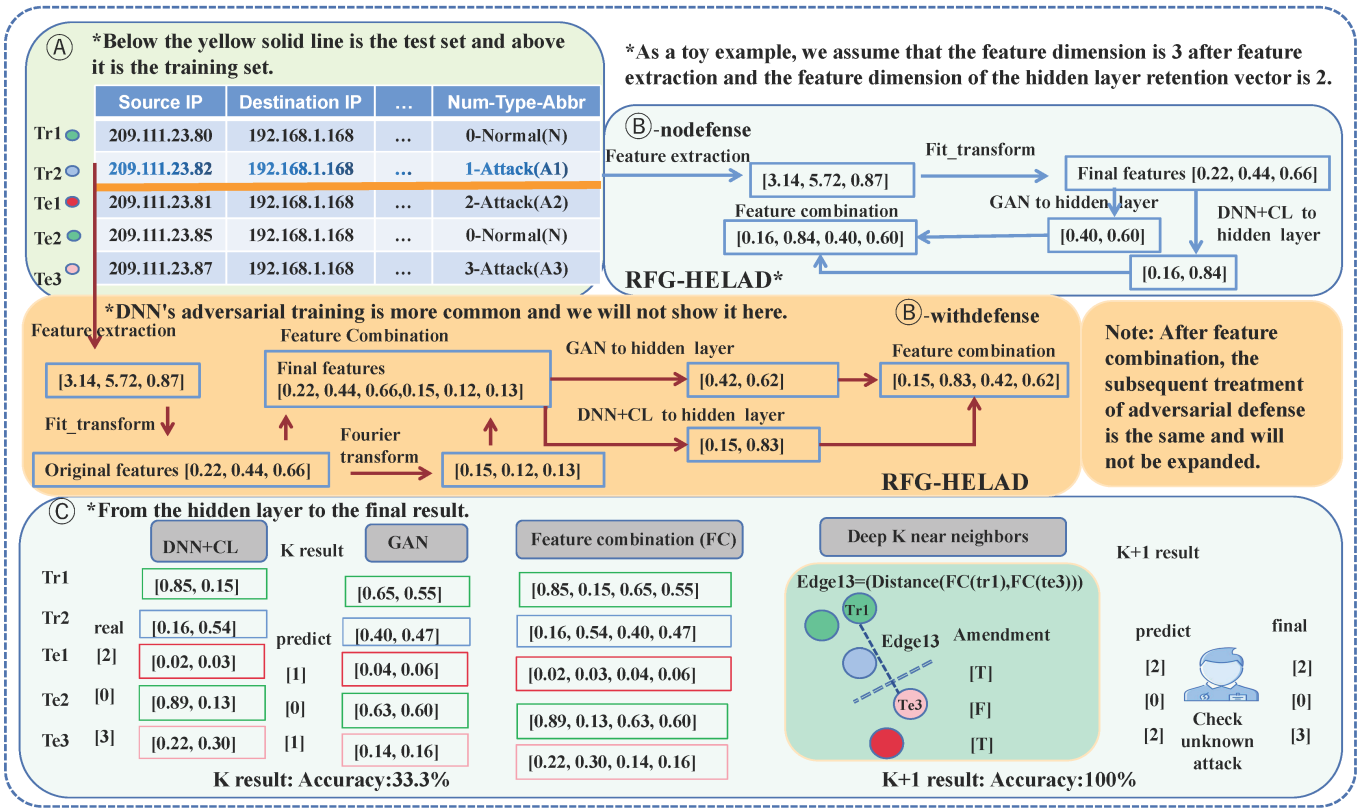


Fig. 1. Data flow diagram of RFG-HELAD and RFG-HELAD* models (toy example).

Part 3: The hardware and software configurations.

We use a server with Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz and 128G of RAM as the device. The type of GPU we use is GTX 1080Ti. The hardware and software configurations are shown in TABLE I.

TABLE I
THE SOFTWARE AND HARDWARE CONFIGURATIONS.

Resource Type	Configuration
Software environment	Ubuntu 16.04, Conda 4.10.1, jupyter 1.0.0 Python 3.5.6, Numpy 1.14.2, faiss 1.5.3 Pytorch 0.4.1, Scikit-learn 0.20.0, librosa 0.9.2
CPU	Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
GPU	GTX 1080Ti 12G
Physical cpu/Logical cpu	2/64
Cores(physical) / Memory	16/128 G

Part 4: Details of the two types of data sets.

Regarding the core experiment, we have two types of data sets. The first category is the K classification experimental data sets (without unknown attack): UKM (training set: 8923, test set: 2226, 5 categories), NSL (training set: 123059, test set: 16179, 8 categories), Kitsune (training set: 810000, test set: 90000, 10 categories). The second category is the $K+1$ classification experimental data sets (with unknown attack): UKM (training set: 8923, test set: 2578, training set 5 categories, test set 8 categories), NSL (training set: 6000, test set: 3000, training set 3 categories, test set 6 categories), Kitsune (training set: 30000, test set: 10000,, training set 6 categories, test set 10 categories). Details can be found in the TABLE II and TABLE III. The data set in TABLE II was only used for K classification experiments without adversarial attacks. The experiments for the evaluation of $K+1$ classification are based on the data set in TABLE III, except in cases where the data set is specifically stated.

Comparative analysis of TABLE II and TABLE III shows that both NSL and Kistune have become smaller and more balanced. This is because the complexity of the comparison algorithms varies. In order to have a more fair comparison, we reduce the amount of data. At the same time, in order to minimize the impact of imbalance on each algorithm (focusing on the core problem), we extract relatively balanced data sets. In order to more realistically reflect the performance of our model in real network scenarios, we have also designed corresponding experiments (original paper). First, UKM uses the complete data set, which is also unbalanced. Second, the Kitsune data set is also experimented with big data in the discussion section. In both scenarios our model performs better, which shows the effectiveness of our model. The highly imbalanced problem in traffic attack detection is the focus of our subsequent work, which is also reflected in the limitation section.

TABLE II
K CLASSIFICATION EXPERIMENTAL DATA SETS (WITHOUT UNKNOWN ATTACK)

Data set	Train set	Test set
UKM	Total: 8923 category[0-4]:[476 404 461 441 7141]	Total: 2226 category[0-4]:[116 96 140 106 1768]
	Real category: ['ARP poisining', 'BeEF HTTP exploits','Mass HTTP requests','Metasploit exploits','Normal']	
NSL	Total: 123059 category[0-7]: [3599 41214 1493 67342 201 2931 3633 2646]	Total: 16179 category[0-7]:[141 4656 73 9711 41 157 735 665]
	Real category:['ipsweep','neptune', 'nmap', 'normal','pod','portsweep','satan', 'smurf']	
Kitsune	Total: 810000 category[0-9]:[601349 20001 18147 27856 28513 37716 10595 6951 29709 29163]	Total: 90000 category[0-9]:[38585 10000 4647 9849 9492 9541 10 87 4447 3342]
	Real category:[Normal, Mirai,Fuzzing,SSDP_Flood,ARP_MitM, Active_Wiretap,SSL_Renegotiation,SYN_DoS,OS_Scan,Video_Injection]	

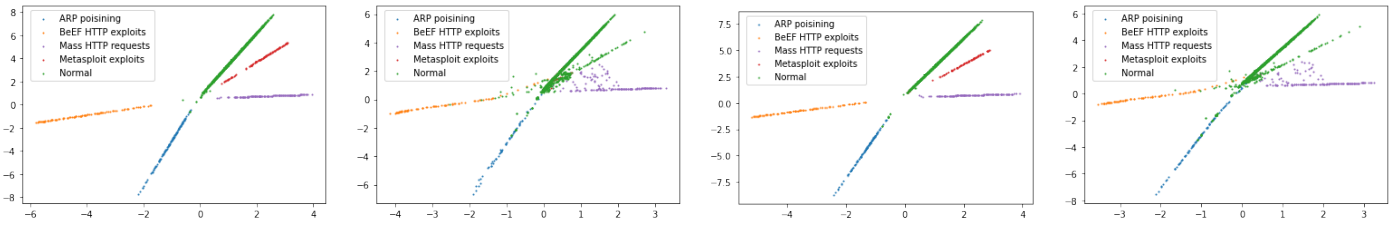
TABLE III
$K+1$ CLASSIFICATION EXPERIMENTAL DATA SETS (WITH UNKNOWN ATTACK)

Data set	Train set	Test set
UKM(5:3)	Total: 8923 category[0-4]:[476 404 461 441 7141]	Total: 2578 category[0-7]:[116 96 140 106 1768 123 121 108]
	Real category:['ARP poisining', 'BeEF HTTP exploits','Mass HTTP requests','Metasploit exploits', 'Normal','Port scanning','TCP flood','UDP data flood']	
NSL(3:3)	Total: 6000 category[0-2]: [2000 2000 2000]	Total: 3000 category[0-5]:[500 500 500 500 500 500]
	Real category:['ipsweep','neptune', 'normal','portsweep','satan', 'smurf']	
Kitsune(6:4)	Total: 30000 category[0-5]:[5000 5000 5000 5000 5000 5000]	Total: 10000 category[0-9]:[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]
	Real category:[Normal, Mirai,Fuzzing,SSDP_Flood,ARP_MitM, Active_Wiretap,SSL_Renegotiation,SYN_DoS,OS_Scan,Video_Injection]	

Part 5: Results of the PGD experiment and detailed analysis of the results of the FGSM adversarial attack.

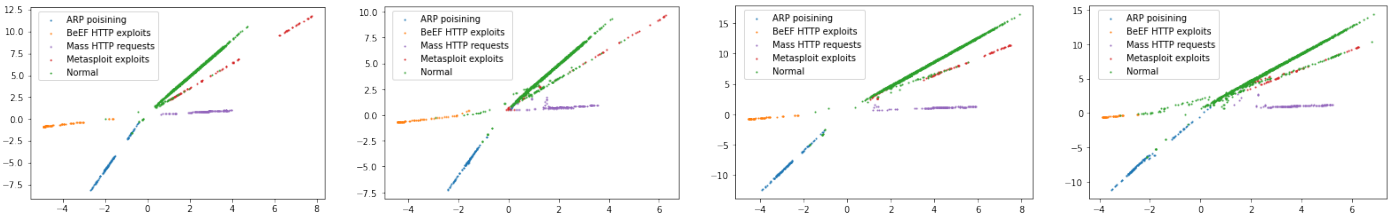
Our model has relatively good robustness in both FGSM and PGD attack environments. In the main text of our paper, we analyze the impact of FGSM on K classification models in detail. Next, we specifically discuss the impact of PGD attacks on K classification models. As shown in TABLE IV, when the test set is all perturbed data (i.e. adv_test), the accuracy of DNN alone is 0.628, while the accuracy of DNN+CL is 0.740. This shows that contrastive learning can improve robustness by compressing the low-dimensional space. The accuracy of our DNN+CL model improves to 0.840 after adding STFT, which is a significant improvement and shows that the STFT plays a important role. Finally, adding adversarial training to the DNN+CL+STFT model improves the accuracy to 0.929, which indicates that adversarial training can also improve robustness, but not as significant as STFT. Our robustness is improved in terms of sample spatial density, feature properties, and model input, respectively. Looking at the DNN+CL and DNN+CL+advT+STFT models of clean_test, it can be seen that the accuracy of RFG-HELAD-K (with adversarial defence) decreases a little in the absence of adversarial attacks. This is because using adversarial defence in the absence of an adversarial attack will degrade detection performance. Overall, the perturbation of PGD attacks affects the model more than FGSM. Even so, the defense effect of our model is still significant.

Also, we visualize the effect of the adversarial attack and the adversarial training as well as the Fourier transform on UKM data set. This is shown in Fig. 2 and Fig. 3. Here the FGSM is used as adversarial attack method. It can be known that when there is a adversarial attack, there are scattered points around. When there is adversarial training, these points are gathered again. Fig. 3 shows the effect with the Fourier transform. It is clear that the distribution is elongated, which is more beneficial for classification.



(a) noadv (b) adv (c) adversarial train-noadv (d) adversarial train-adv

Fig. 2. Relationship between our K-class classification model on adversarial attack and adversarial training in the case of no unknown attack (raw features, UKM data set, FGSM). noadv represents a scenario without adversarial attack, and adv represents a situation with adversarial attack.



(a) noadv (b) adv (c) adversarial train-noadv (d) adversarial train-adv

Fig. 3. Relationship between our K-class classification model on adversarial attack and adversarial training in the case of no unknown attack (raw features + STFT, UKM data set, FGSM).

TABLE IV

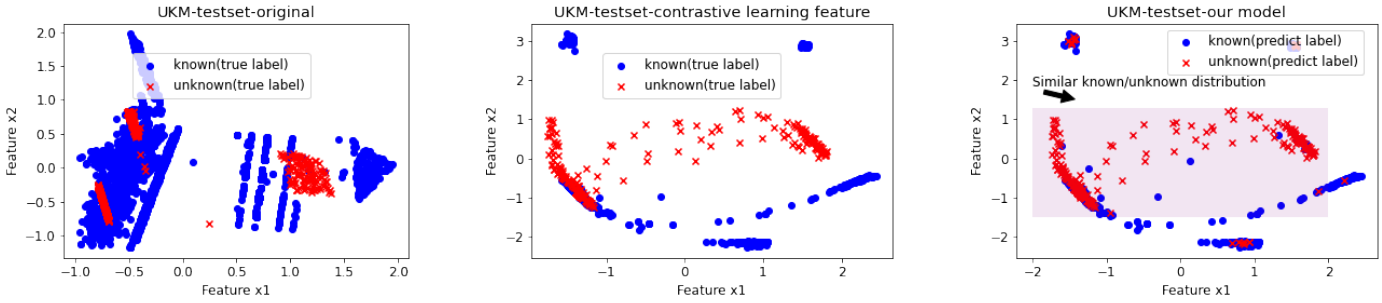
ABLATION EXPERIMENTS OF OUR DEFENSE MODEL IN THE CASE OF PGD ADVERSARIAL ATTACK AND UKM DATASET (ABS_STEPSIZE=1.1, STEPS=7, ACC-K). ADVT STANDS FOR ADVERSARIAL TRAINING. CLEAN_TEST STANDS FOR ACCURACY IN RFG-HELAD WITHOUT ADVERSARIAL ATTACKS. ADV_TEST REPRESENTS THE ACCURACY OF RFG-HELAD IN THE PRESENCE OF ADVERSARIAL ATTACKS.

model	clean_test	adv_test	AVG
DNN	0.996	0.628	0.812
DNN+advT	0.996	0.630	0.813
DNN+STFT	0.988	0.846	0.917
DNN+advT+STFT	0.988	0.848	0.918
DNN+CL	1.000	0.740	0.870
DNN+CL+advT	0.991	0.789	0.890
DNN+CL+STFT	0.987	0.840	0.913
DNN+CL+advT+STFT(ours)•	0.958	0.929	**0.943**

• *DNN+CL+advT+STFT(ours) means RFG-HELAD-K;*

Part 6: Low dimensional projection with different cases on UKM test set.

For a more visual presentation, based on the UKM data set, we visualize the true label distribution of the original data set, the true label distribution of DNN+CL, and the predicted label distribution of our model. Fig. 4 shows low dimensional projection with different cases. We are primarily interested in analysing trends, and quantitative analysis has to be done in the original feature space. Fig. 4 (a) shows that the original data is haphazard. Fig. 4 (b) reveals that contrastive learning gives some regularity to the original distribution. Fig. 4 (c) indicates that our model learns the true distribution well after the contrastive learning process.



(a) Original data distribution (b) True distribution of DNN+CL (c) Predict distribution of our model

Fig. 4. Low dimensional projection with different cases on UKM test set. In order to compare trends in the same feature space, we present the real labels and the predicted results of our model on the CL features (as used in (b) and (c)). We are primarily interested in analysing trends, and quantitative analysis has to be done in the original feature space.

Part 7: How does the proportion of different unknown attacks affect the performance of our model?

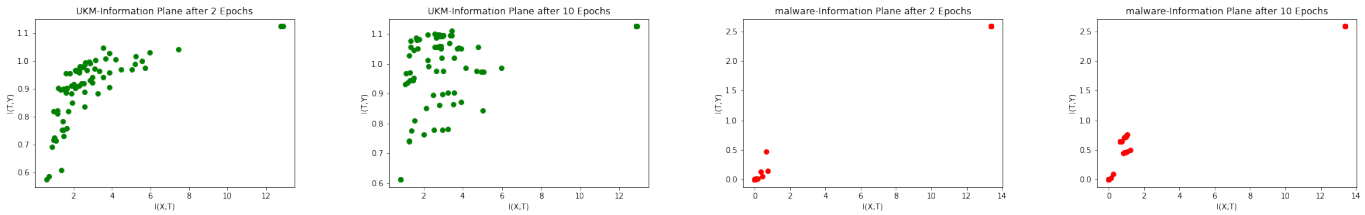
Clearly, for the same data set (e.g. UKM), the fewer unknown attacks, the better the detection. For different data sets (e.g. UKM and NSL), as you can see from the TABLE V, with the same classification complexity, the fewer the unknown attacks, the better the detection. The complexity of the data set itself can also greatly affect the detection of unknown attacks. For example, Kitsune does not have the highest percentage of unknown attacks, but the lowest accuracy. We design the experiment using the appropriate percentage of unknown attacks. Because the reality is that if all the attacks are unknown, all of them need to be checked manually. This is no different from the pure manual check.

TABLE V
RELATIONSHIP BETWEEN THE PROPORTION OF UNKNOWN ATTACKS AND MODEL PERFORMANCE.

percentage	partition	ACC-(K+1)
0.5	NSL(3:3)	0.928
0.4	Kitsune(6:4)	0.918
0.375	UKM (5:3)	0.948
0.000	UKM (8:0)	0.998

Part 8: How do we understand the superiority of DNN on network data?

We disprove by information plane that not all datasets are suitable for DNN models. [1] proposed to analyze DNNs on the information plane (IB). The information plane refers to the plane formed by the mutual information retained by each layer on the input and output variables. [2] argued that the goal of DNN networks is to optimize the information bottleneck tradeoff between compression and prediction. And it argues that the DNN layer is ultimately very close to the theoretical bounds of IB. We justify our choice of DNN by visualizing the information plane to aid in the justification.



(a) ukm2 (b) ukm10 (c) malware2 (d) malware10

Fig. 5. Mutual information presentation of UKM dataset and Malware dataset on the information plane using DNN as a classification model. ukm2 represents the case of the ukm data set trained with 2 epochs. The fewer epochs, and the more samples trained in the upper left corner, the more suitable the DNN model.

We select a training set of 8400 samples for the malware data [3], containing six categories, as in the FARE configuration, with 1400 samples per category. For the test set, it is also similar in size to the UKM data set, containing six categories with 350 samples per category. We maintain the other configurations as for the UKM data set, as for a DNN model.

As shown in Fig. 5, the DNN model can reach high mutual information very quickly for traffic data, but not for the malware data set. where the horizontal axis represents the compression rate of the hidden layer representation T to the input X. The higher the compression rate the less information the hidden layer T contains about the input X (the smaller the I(X;T)), i.e., the deeper the hidden layer. The vertical axis represents the relevant information about the prediction Y in the hidden layer representation T. The larger this value is, the smaller the distortion is. The previous part of the training, I(X;T) and I(T;Y) are increased. The network remembers the input and label information as much as possible. In the later part of the training, I(T;Y) is still increasing and I(X;T) is decreasing. At this time, the DNN tries to forget the unimportant features in X to enhance the generalization ability. In summary, DNN and network traffic data are well adapted.

Part 9: Specific presentation of assessment indicators.

We use the accuracy of multiclass classification as the main evaluation metric. If $\hat{z}_i$ is the predicted value of the i-th sample and z_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\text{accuracy}(z, \hat{z}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \mathbf{1}\left(\hat{z}_i = z_i\right)$$

where $\mathbf{1}(x)$ is the indicator function and n_{samples} is the number of test samples. We define the accuracy of K classification as **ACC-K**, and the accuracy of $K+1$ classification as **ACC-(K+1)**.

To illustrate that the use of accuracy is representative, we also evaluate other metric [4]: Area Under Curve (AUC), $avg\_Precision$ which means weighted average Precision as **P**, avg_Recall as **R**, and avg_F_1 as **F1**. Weighted average means that calculate metrics for each label, and find their average, weighted by support (the number of instances for each label). The specific details are as follows.

$$avg\_Precision = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| \, P\left(y_l, \hat{y}_l\right) \tag{1}$$

$$avg\_Recall = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| \, R\left(y_l, \hat{y}_l\right) \tag{2}$$

$$avg\_F_\beta = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| \, F_\beta\left(y_l, \hat{y}_l\right) \tag{3}$$

Among, y denotes the set of true (sample, label) pairs. $\hat{y}$ denotes the set of predicted (sample, label) pairs. L denotes the set of labels. y_l denotes the subset of y with label l, and $\hat{y}_l$ is subset of $\hat{y}$. $P(A, B) = \frac{|A \cap B|}{|B|}$ for some sets A and B. $R(A, B) = \frac{|A \cap B|}{|A|}$ (Conventions vary on handling $A = \emptyset$; this implementation uses $R(A, B) = 0$, and similar for P.) $F_\beta(A, B) = \left(1 + \beta^2\right) \frac{P(A,B) \times R(A,B)}{\beta^2 P(A,B) + R(A,B)}$ and β=1.

Area Under Curve (AUC) is the area enclosed by the receiver operating characteristic (ROC) curve and the coordinate axis. AUC is essentially a binary classification evaluation metric. There are various ways to calculate AUC. We present methods based on Wilcoxon test of ranks from a probabilistic perspective [5, 6]. |positive_example| denotes the number of positive samples. |negative_example| denotes the number of negative samples. Ranked according to the model's predicted scores, $rank_p$ denotes the sequential number of the p-th sample.

$$AUC = \frac{\sum_{p \in positive\_example} \text{rank}_p - \frac{|positive\_example| * (|positive\_example|+1)}{2}}{|positive\_example| * |negative\_example|} \tag{4}$$

We treat unknown attacks as positive examples, and known attacks and normal traffic as negative examples. To better assess generalizability, some of the experiments calculate the average of individual result, denoted as AVG. **For all metrics, the higher the value, the better**.

Part 10: Detailed format of the three data sets.

TABLE VI and VII present the format of the three data sets and their sample examples, respectively.

TABLE VI
FORMAT OF INDIVIDUAL DATA SET.

Data set	Format
UKM	dur,trnspt,srvs,flag_n,flag_arst,flag_uc,flag_sign,flag_synrst,flag_a,flag_othr,src_pkts,dst_pkts,urg_bits, push_pkts,no_lnkd,arp,src_ttl,dst_ttl,pkts_dirctn,src_byts,dst_byts,src_avg_byts,dst_avg_byts,strt_t,end_t, dst_host_count,host_dst _count,rtt_first_ack,rtt_avg,avg_t_sent,avg_t_got,repeated,fst_src_sqc,fst_dst_sqc, src_re,dst_re,src_fast_re,dst_fast_re,ovrlp_count,long_frag_count,dns_ratio,avg_rr,http_rqsts_count, http_redirct_count,http_clnt_error_count,http_srv_error_count,Class name
NSL	duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, hot, num_failed_logins, logged_in, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_host_login, is_guest_login, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate, attack_type
Kitsune	frame.time_epochframe.leneth.src, eth.dst, ip.src, ip.dst, tcp.srcport, tcp.dstport, udp.srcport, udp.dstport, icmp.type, icmp.code, arp.opcode, arp.src.hw_mac, arp.src.proto_ipv4, arp.dst.hw_mac, arp.dst.proto_ipv4, ipv6.src, ipv6.dst, attack_type

TABLE VII
EXAMPLES OF EACH DATA SET.

Data set	Example data
UKM	89.814777,6,80,0,0,0,0,1,0,0,3,2,0,0,5005,0,128,127,2,174,120,58,60,2743,2832.814777, 0,5006,0.000569,0.0039798,1410.387432,1425.363018,0, 3023308442,312736223,0,0,0,0,0,0,0,0,0,0,0,TCP flood
NSL	0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,229,10,0.00,0.00,1.00,1.00, 0.04,0.06,0.00,255,10,0.04,0.06,0.00,0.00, 0.00,0.00,1.00,1.00,neptune
Kitsune	1488018358.617213 136 00:0c:29:da:09:84 c8:00:84:2a:39:51 192.168.3.11 192.168.100.5 10696.0 1900.0 ..., SSDP_Flood

*Regarding Kitsune, we only show the part used for feature extraction.

Part 11: How does our model perform in other data sets?

In the preceding experiments, we have sufficiently demonstrated the effectiveness of our model's adversarial defense. To understand how well our model detects on other data sets, we add K and $K+1$ experiments and compare our model to the top3 comparison algorithms. We add CICIDS2018 [7] and TON2019 [8] data sets for experiments. The training set of CICIDS2018 is [Benign,SSH-Bruteforce, DoS-Hulk] [12000, 4000, 10000] and the test set is [Benign, SSH-Bruteforce, DoS-Hulk, Infiltration] [3249, 346, 1697, 4238]. What follows is the number of attacks. The training set of TON2019 is [backdoor, ddos, dos, injection, normal] [800 800 800 800 5600] and the test set is [backdoor, ddos, dos, injection, normal, password, scanning, xss] [200 200 200 200 1400 200 200 200]. As shown in TABLE VIII and TABLE IX, our model has excellent performance, which suggests that our model has good detection performance in other data sets as well.

Since the CICIDS2018 dataset is more specific in terms of the collection time of each type of attack, we further evaluate the detection speed of our model. We test it using the full attack data of SSH-Bruteforce, DoS-Hulk, and Infiltration in

the CICIDS2018 data set. As shown in TABLE X, the detection time of our model is less than the data collection time, which further indicates the superiority of our model's detection speed.

TABLE VIII
K CLASSIFICATION EXPERIMENTS ON OTHER DATA SETS (ACC-K, DROPPING UNKNOWN ATTACKS).

model	CICIDS2018(3:1)	TON2019(5:3)
DNN	1.000	1.000
SCADA	1.000	1.000
ours (RFG-HELAD*-K)	1.000	1.000

TABLE IX
$K+1$ CLASSIFICATION EXPERIMENTS ON OTHER DATA SETS (ACC-(K+1)).

model	CICIDS2018(3:1)	TON2019(5:3)
RFG-HELAD*-K	0.555	0.787
CVAE-EVT	0.555	0.783
openmax(DNN)	0.555	0.880
ours(RFG-HELAD*-K+1)	0.994	0.924

TABLE X
COMPARISON OF DATA COLLECTION TIME WITH THE DETECTION TIME OF OUR MODEL IN CICIDS2018 DATA SET.

Attacks	Number	Collection time	Detection time
SSH-Bruteforce	187,589	90 min	76s
DoS-Hulk	461,912	34 min	189s
Infiltration	68,871	155 min	32s

REFERENCES

[1] Naftali Tishby, Noga Zaslavsky, "Deep learning and the information bottleneck principle" in *ITW*, pp. 1-5, 2015.
[2] Ravid Shwartz-Ziv, Naftali Tishby, "Opening the Black Box of Deep Neural Networks via Information" in *CoRR abs/1703.00810*, 2017.
[3] Junjie Liang, Wenbo Guo, Tongbo Luo, Vasant G. Honavar, Gang Wang, Xinyu Xing, "FARE: Enabling Fine-grained Attack Categorization under Low-quality Labeled Data" in *NDSS*, 2021.
[4] Zhi-Hua Zhou, "Machine Learning," *Springer*, pp. 1-459, 2021.
[5] Tom Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861-874, 2006.
[6] James A. Hanley and Barbara J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, pp. 29-36, 1982.
[7] Iman Sharafaldin et al., "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *ICISSP*, Portugal, 2018.
[8] Nour Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society*, vol. 72, 2021.