A C-IFGSM Based Adversarial Approach for Deep Learning Based Intrusion Detection

Speaker：Yingdi Wang
Email：17120482@bjtu.edu.cn

Content

01 Part one: Introduction

02 Part two: Constraint-IFGSM

03 Part three: Experiments and Results

04 Part four: Conclusion

Content

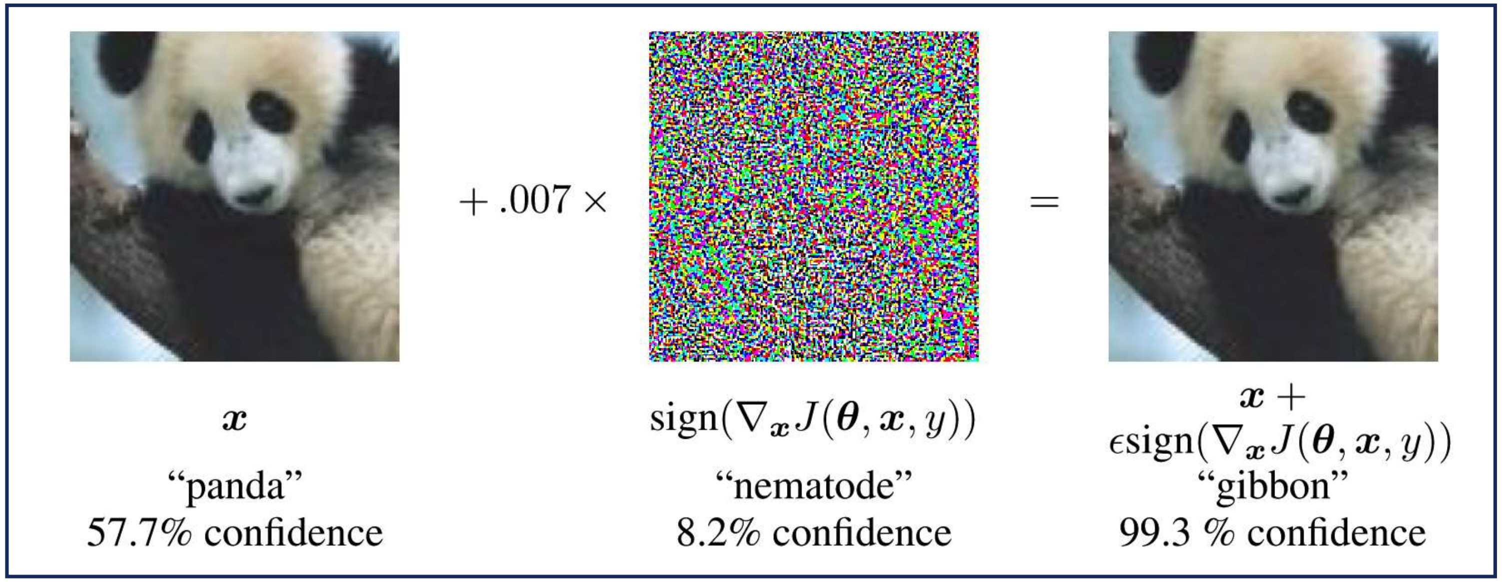
Part One

Introduction

machine learning based intrusion detection

- Machine learning algorithms have helped humans solve problems in various fields.
- Machine learning algorithms have also been increasingly applied to security fields, such as spam detection, malware detection and intrusion detection.
- There are advantages and disadvantages of intrusion detection based on machine learning. Intrusion detection classifiers can detect unknown types of attacks, but they may also be attacked by adversarial examples.

adversarial examples in machine learning algorithms

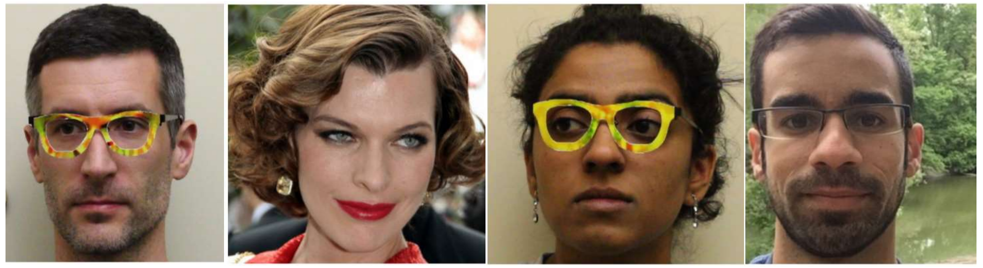


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

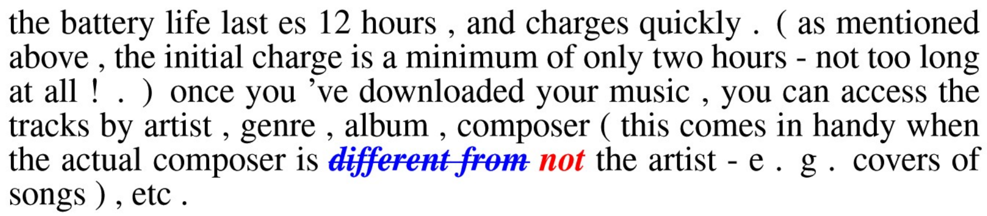
- Adversarial examples will make the machine learning model misclassified by adding a slight modification to real examples, which cannot be distinguished by humans.
- Especially for neural network-based intrusion detection, the misclassified of anomaly network traffic by intrusion detection systems may lead to malicious attacks on systems.

crafting adversarial examples in different applications

- The methods of crafting adversarial examples in different applications are restricted by the specific characteristics of dataset.



- An attacker can construct a pair of glasses to allow the wearer to evade recognition by the facial recognition system by adding noise to a specific area.



the battery life last es 12 hours , and charges quickly . (as mentioned above , the initial charge is a minimum of only two hours - not too long at all ! .) once you 've downloaded your music , you can access the tracks by artist , genre , album , composer (this comes in handy when the actual composer is ~~*different from*~~ *not* the artist - e . g . covers of songs) , etc .

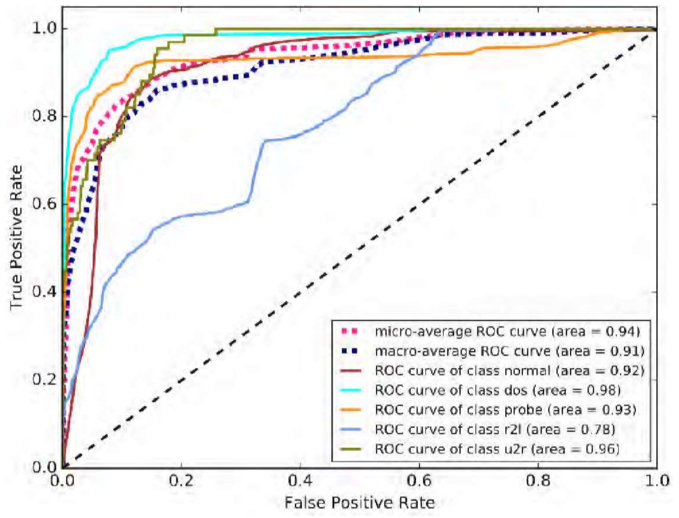
- Crafting text adversarial examples needs to consider the discrete features of words and the grammar of sentences.

$$i_{max} = \arg\max_{j \in \Gamma \cap \mathbf{I}, X_j = 0} \frac{\partial \mathbf{F}_y(\mathbf{X})}{\partial \mathbf{X}_j}$$

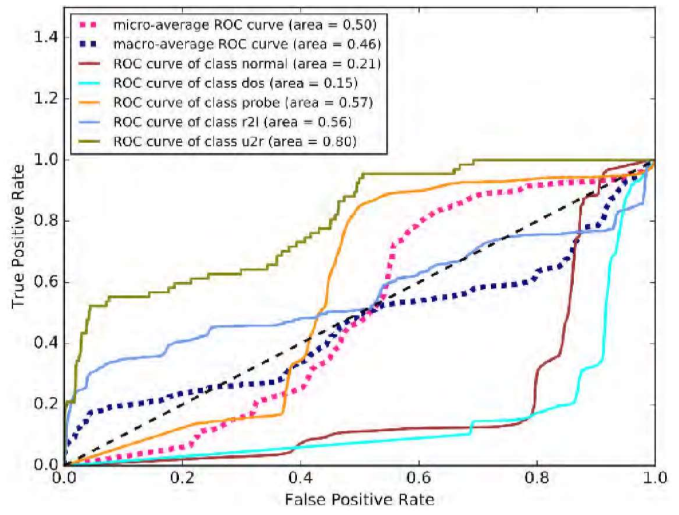
- The methods for crafting malware adversarial examples need to consider both the type of features and its functionality

crafting adversarial examples for intrusion detection classifiers

- There have been researchers using FGSM to generate adversarial examples for NSL-KDD dataset.



Adversarial Examples Attack



flag_OTH	flag_REJ	······	flag_SF	flag_SH
0	0	······	1	0
0	1	······	1	0

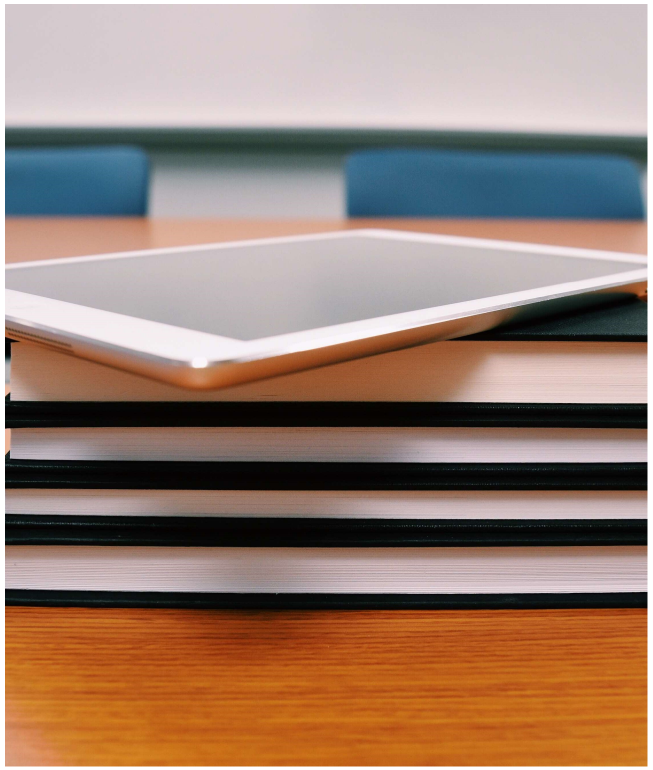
- Although FGSM can add a slight noise to the original examples to cause deep neural networks misclassification, it cannot guarantee that the types and relationship of features in adversarial examples are consistent with original examples.

Content

Part Two

Constraint-IFGSM

Research Targets



- This paper proposed a C-IFGSM based method which can adapt to complex datasets with multiple types of features and multiple relationship among features.
- The intrusion adversarial examples need to satisfy the following three parts:
  - making the intrusion detection classifiers go wrong
  - keeping the feature types unchanged
  - keeping the feature relationships unchanged

Analysis of network traffic data

01 A variety of feature types

Due to the complexity of network traffic, there may be various types of statistical features, such as character features, discrete features and continuous features.

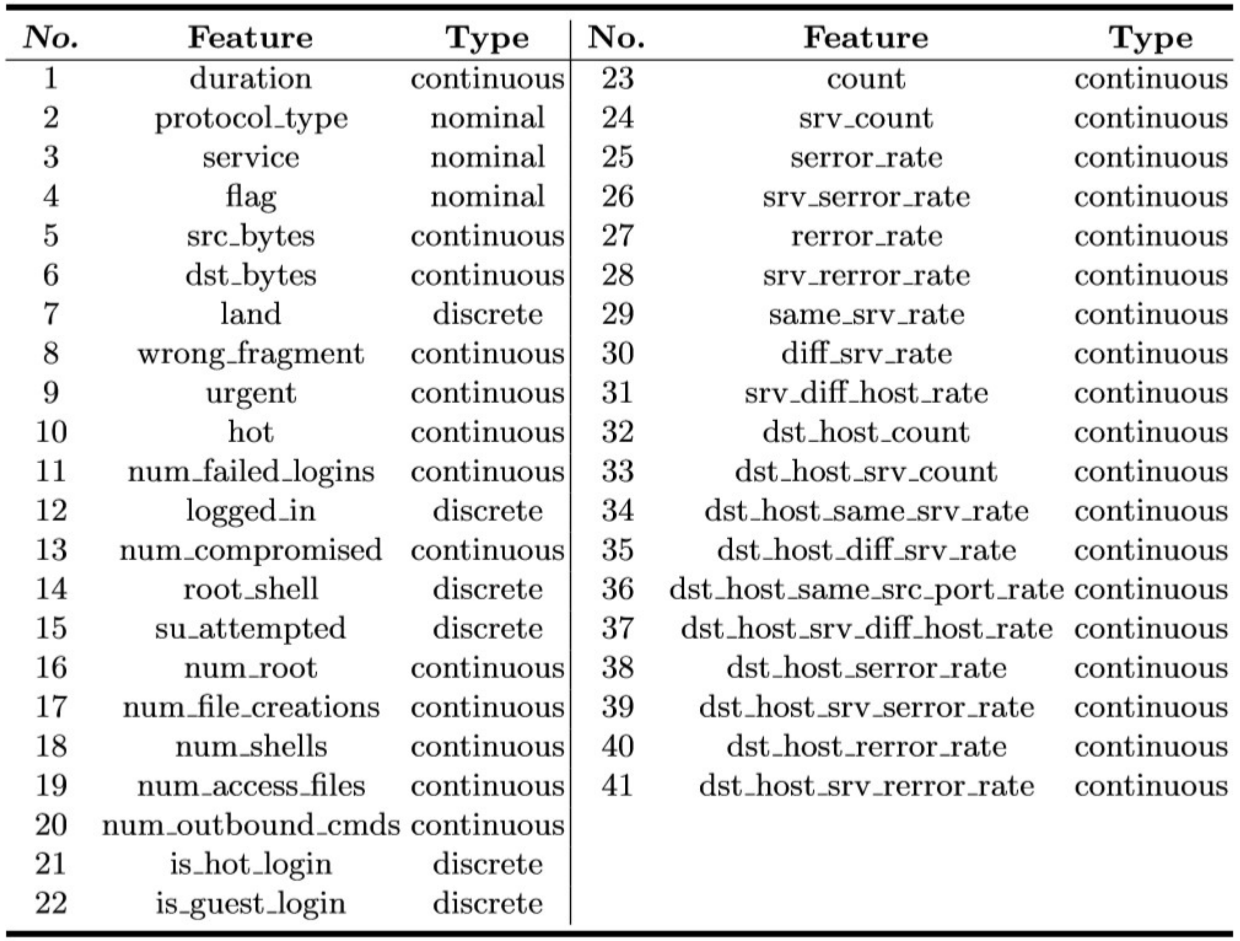
02 Specific malicious payloads

Network traffic data has specific attack effect, and its malicious payload may be reflected in some features, or fixed feature format.

03 Constraint relationships between features

There may be a certain constraint relationship between the features of network traffic data, which may be caused by some certain attack patterns or correlation between statistical methods of some features.

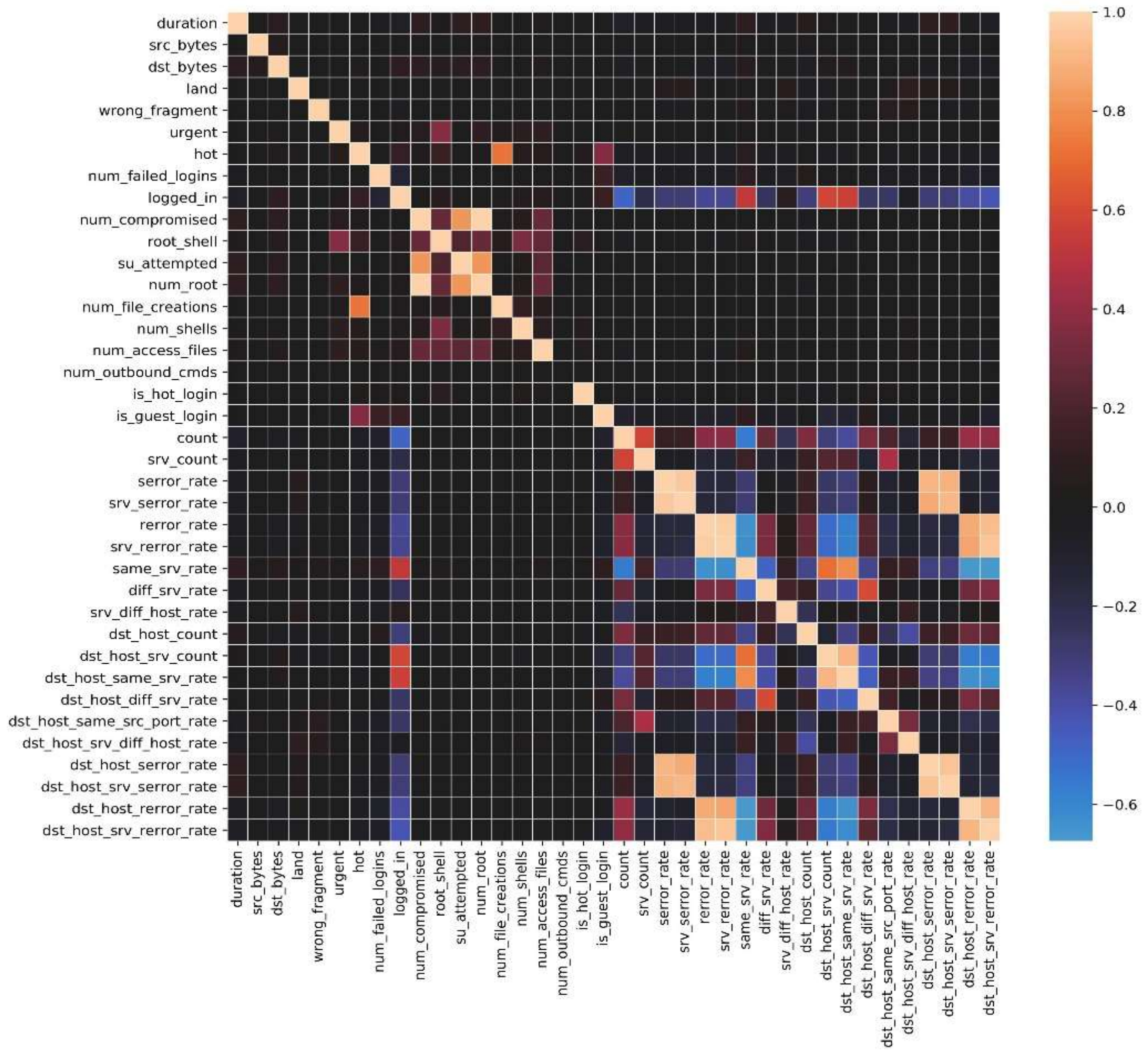
multi-type features of NSL-KDD dataset



No.	Feature	Type	No.	Feature	Type
1	duration	continuous	23	count	continuous
2	protocol_type	nominal	24	srv_count	continuous
3	service	nominal	25	serror_rate	continuous
4	flag	nominal	26	srv_serror_rate	continuous
5	src_bytes	continuous	27	rerror_rate	continuous
6	dst_bytes	continuous	28	srv_rerror_rate	continuous
7	land	discrete	29	same_srv_rate	continuous
8	wrong_fragment	continuous	30	diff_srv_rate	continuous
9	urgent	continuous	31	srv_diff_host_rate	continuous
10	hot	continuous	32	dst_host_count	continuous
11	num_failed_logins	continuous	33	dst_host_srv_count	continuous
12	logged_in	discrete	34	dst_host_same_srv_rate	continuous
13	num_compromised	continuous	35	dst_host_diff_srv_rate	continuous
14	root_shell	discrete	36	dst_host_same_src_port_rate	continuous
15	su_attempted	discrete	37	dst_host_srv_diff_host_rate	continuous
16	num_root	continuous	38	dst_host_serror_rate	continuous
17	num_file_creations	continuous	39	dst_host_srv_serror_rate	continuous
18	num_shells	continuous	40	dst_host_rerror_rate	continuous
19	num_access_files	continuous	41	dst_host_srv_rerror_rate	continuous
20	num_outbound_cmds	continuous			
21	is_hot_login	discrete			
22	is_guest_login	discrete			

- ☐ **Basic features(1-9):** contain the basic connection information of data packets.
- ☐ **Content features(10-22):** contain some useful payload in data packets.
- ☐ **Traffic features(23-41):** show the traffic information presented by the current connection and its nearby connections. Traffic features are calculated based on two statistics method, a 2-second time window(23-31) and a 100-connection window(32-41).

multi-relationship features of NSL-KDD dataset

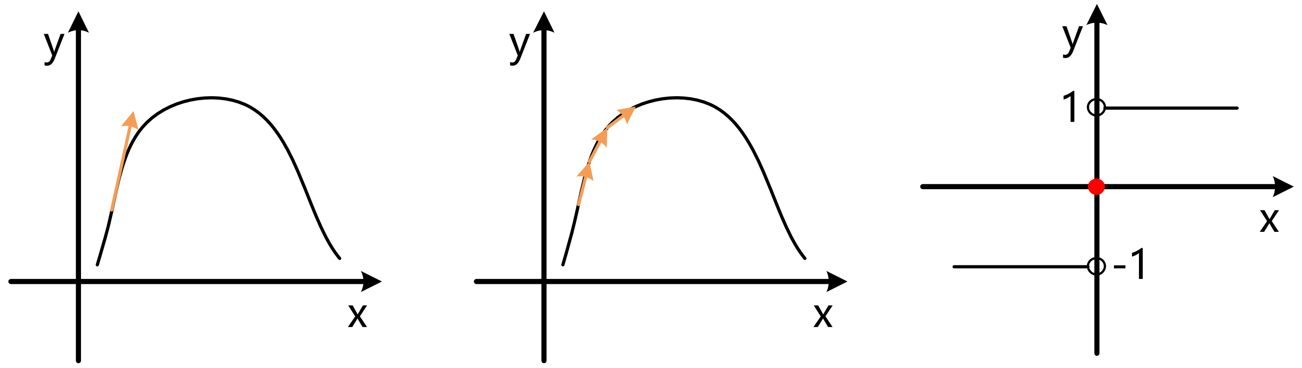


- $\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y}$
- The PCC between the two features is calculated by the ratio of the covariance to the standard deviation between the two features. The larger the absolute value, the stronger the correlation between the two features.
- The correlation between features mainly exists in the traffic features.
- The Pearson Correlation Coefficient matrix heat map of numeric and discrete features in NSL-KDD dataset.

FGSM & IFGSM

Fast Gradient Sign Method，FGSM

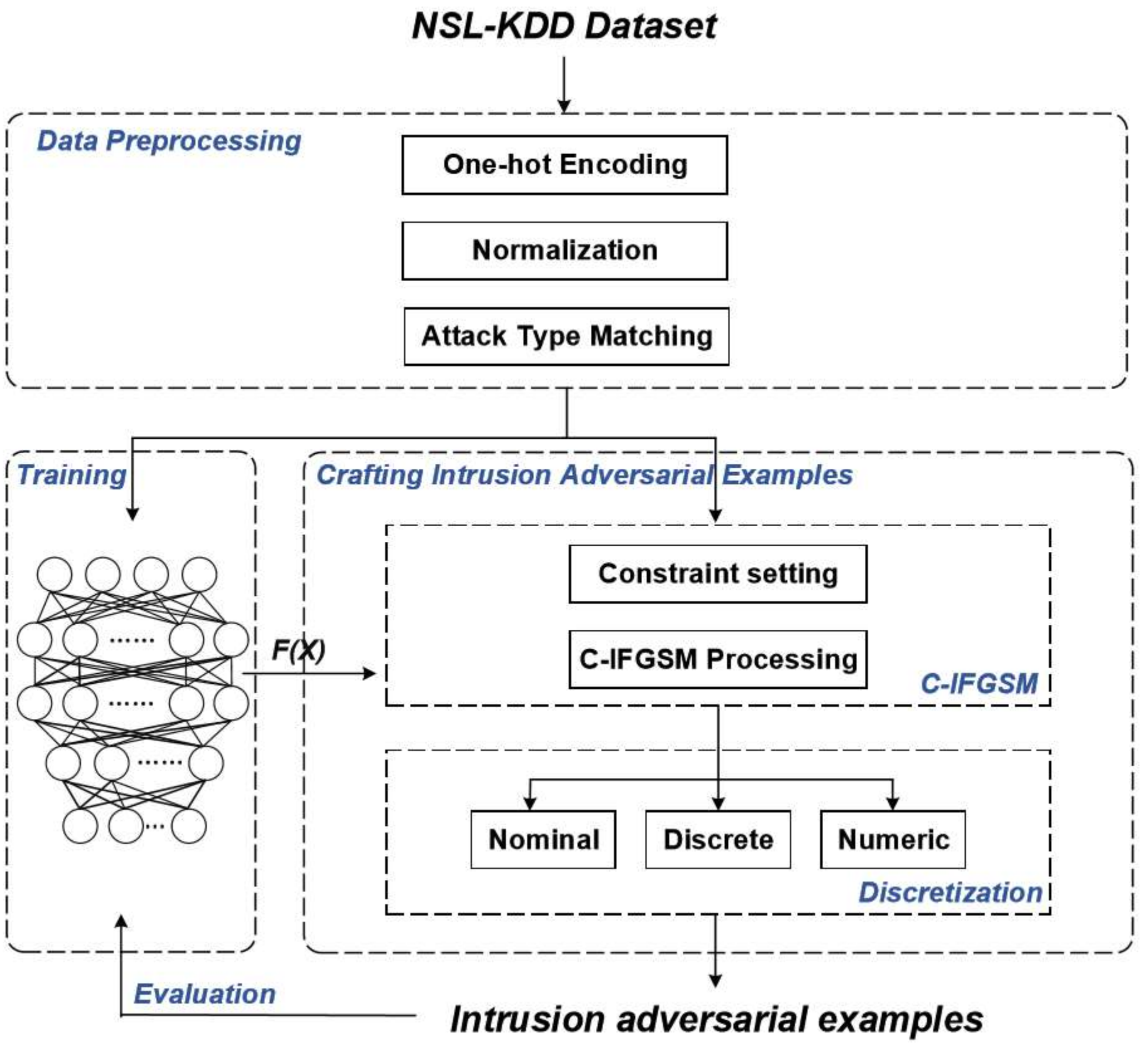
- The formula： $X^{adv} = X + \varepsilon sign(\nabla_x J(X, Y_{true}))$
- The adversarial example will be obtained by moving the original example X one step in the direction of the gradient of loss function $J(\theta, x, y)$.



Iterative Fast Gradient Sign Method，IFGSM

- IFGSM added multiple iterations with small step size to FGSM.
- The formula ： $X_0^{adv} = X$， $X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha sign(\nabla_x J(X_N^{adv}, Y_{true}))\}$
- To ensure the perturbation within the range of ε, there is a relationship between α and the number of iterations T, $\alpha = \varepsilon / T$

C-IFGSM Based Adversarial Approach



1 Data Processing

Due to the complexity of network traffic features, it is necessary to preprocess the dataset before building the classification model and crafting intrusion adversarial examples.

2 DNN-based Intrusion Detection

IFGSM is a white-box attack algorithm, which means that it needs to obtain the parameters of the model in advance. So we first need to train a deep learning model for intrusion detection.

3 Crafting Intrusion Adversarial Examples

First part is C-IFGSM algorithm, where we add a constraint matrix to IFGSM. The second part is discretization.

Data Preprocessing

- **One-Hot Encoding.** One-hot encoding can convert the nominal features to numeric features. The number of bits for a numeric feature is the same with the number of its categories. In this way, for each category of a numeric feature, there is only one bit is 1, the others are all 0.
- Protocol_type： **tcp, udp, icmp, udp**
- After one-hot encoding, they will be expressed as **[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]**
- Therefore, after one-hot encoding, and the total number of features will change from 41 to 122.
- **Normalization.** After one-hot encoding, there will be a large range difference between discrete and continuous features, which will affect the weights of features. Therefore, we normalize the features to make them all in range of [0,1].
- **Attack Types Mapping**. The NSL-KDD dataset contains 39 types of attack, which belong to 4 categories: Denial of Service(DOS), Probe, Remote to Local(R2L), and User to Root(U2R). We map 39 attack types to 4 categories of attacks and build up an intrusion detection classifier of five-classification(4 attack categories and 1 normal category).

Constraint Iterative Fast Gradient Sign Method(C-IFGSM)

- C-IFGSM adds a constraint matrix C_n to IFGSM. By setting the constraint matrix C_n, C-IFGSM can restrict the modification of features in original examples.

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = Clip_{X,\epsilon}\{X_N^{adv} + \alpha C_n \odot sign(\nabla_x J(X_N^{adv}, Y_{true}))\}$$

- Hadamard product is a matrix product operation that multiplies the elements in the same position of two matrixes without changing the dimension of the matrix.

$$\begin{aligned} X^{adv} &= X + \alpha C_n \odot sign(\nabla_x J\left(X_i^{adv}, Y_{true}\right) \\ &= X + \alpha C_n \odot S_n \\ &= X + \alpha[c_i \times s_i] \end{aligned}$$

- By setting the constraint matrix $c_i \times s_i$, we can control the noise value added to the original examples.

➢ Setting c_i to 1, the modification will completely depend on the s_i in sign matrix.
➢ Setting c_i to 0, the coefficient α will not work and feature x_i will not be modified.
➢ Setting c_i to a specific can change the scale of modification and control the value of perturbation added to feature x_i.

Discretization Process

- Because the added perturbation can be a small continuous value, so the nominal features and discrete features in the adversarial examples will change to continuous feature.
- **Discrete Features.** For discrete feature whose value can only be 0 or 1, we can discretize it with a boundary of 0.5.

$$IF\ x_i^{adv} \geq 0.5, THEN\ x_i^{adv} = 1;$$

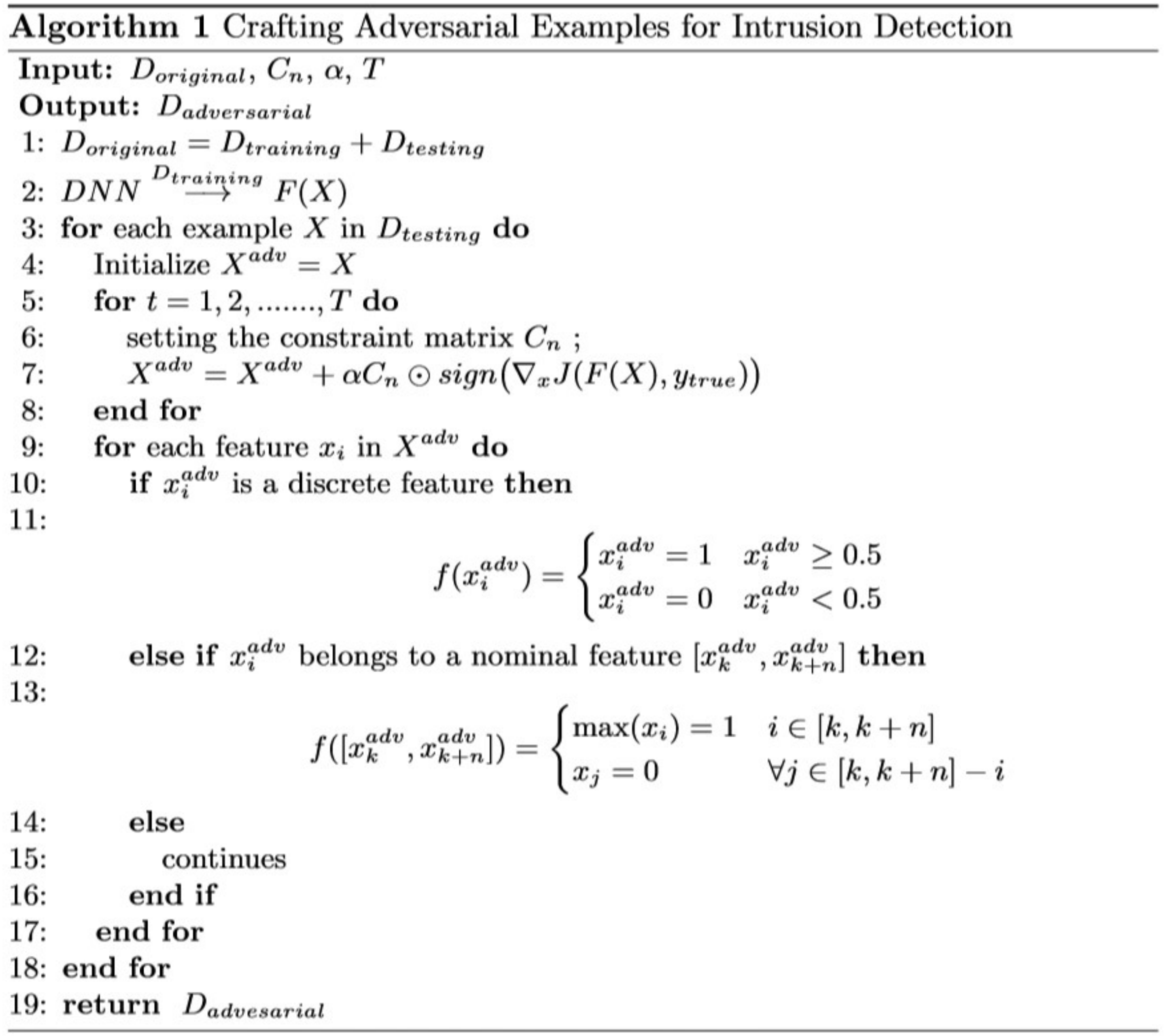
$$IF\ x_i^{adv} < 0.5, THEN\ x_i^{adv} = 0$$

- **Nominal Features.** For a nominal feature x_i, after one-hot encoding, x_i is convert to $[k, k+n]$, where n is the total number of categories. There is only one feature is assigned 1, and the others are all 0. So we setting the feature with the largest value in $[k, k+n]$ as 1, and setting the rest features as 0.

$$\max(x_i) = 1, i \in [k, k+n]$$

$$x_j = 0, \forall j \in [k, k+n] - i$$

C-FGSM based Intrusion Adversarial Example Crafting



Algorithm 1 Crafting Adversarial Examples for Intrusion Detection

```
Input: $D_{original}$, $C_n$, $\alpha$, $T$
Output: $D_{adversarial}$
1: $D_{original} = D_{training} + D_{testing}$
2: $DNN \overset{D_{training}}{\longrightarrow} F(X)$
3: for each example $X$ in $D_{testing}$ do
4:     Initialize $X^{adv} = X$
5:     for $t = 1, 2, ......., T$ do
6:         setting the constraint matrix $C_n$ ;
7:         $X^{adv} = X^{adv} + \alpha C_n \odot sign(\nabla_x J(F(X), y_{true}))$
8:     end for
9:     for each feature $x_i$ in $X^{adv}$ do
10:        if $x_i^{adv}$ is a discrete feature then
11:            $f(x_i^{adv}) = \begin{cases} x_i^{adv} = 1 & x_i^{adv} \geq 0.5 \\ x_i^{adv} = 0 & x_i^{adv} < 0.5 \end{cases}$
12:        else if $x_i^{adv}$ belongs to a nominal feature $[x_k^{adv}, x_{k+n}^{adv}]$ then
13:            $f([x_k^{adv}, x_{k+n}^{adv}]) = \begin{cases} \max(x_i) = 1 & i \in [k, k+n] \\ x_j = 0 & \forall j \in [k, k+n] - i \end{cases}$
14:        else
15:            continues
16:        end if
17:    end for
18: end for
19: return $D_{advesarial}$
```

- **Input:** The original dataset $D_{original}$ is used to initialize the adversaria examples, the single step size α and the number of iterations T specify the iterative process of generating the adversaria examples.

- **Output:** The adversarial dataset$D_{adversarial}$

- Under the limitation of constraint matrix C_n, the adversarial examples are generated iteratively. In each iteration process, the constraint matrix C_n is set and updated according to the constraint between features.

Content

Part Three

Experiments and Results

Experiment Setup

Experiment Environment

- Python 3.6 was used as the programming language, JetBrains PyCharm as IDE for adversarial examples generation experiments.
- The pytorch 0.4.1 machine learning framework was used to build the intrusion detection classification model and generate the intrusion adversarial examples.

NSL-KDD Dataset

- The distribution of each category of data in the training and testing sets of NSL-KDD.

Categories	DOS	Probe	R2L	U2R	Normal	Total
Training Set	45,927	11,656	995	52	67,373	126,003
Testing Set	7,460	2,421	2,885	67	9,711	22,544
Total	53,387	14,077	3,880	119	77,084	148,547

Building the Intrusion Detection Classifier

Models Comparing

- We compared the performance of several widely used neural network models, and selected the model with the highest accuracy as the target of adversarial attack.

Algorithms	Accuracy
Deep Neural Network (DNN)	0.79
Convolutional Neural Network（CNN)	0.76
Multilayer Perceptron（MLP)	0.74

DNN model

- The network structure of DNN is shown in the table below.
- Because the adversarial examples have transferability in neural network structures, so it is representative to select basic DNN model as the attack target.

Layers	Active Function	Input×Output
Input		Batchsize×122
Hidden Layer 1	ReLu	122×512
Hidden Layer 2	LeakyReLu	512×512
Hidden Layer 3	LeakyReLu	512×256
Hidden Layer 4	LeakyReLu	256×64
Output	softmax	64×5

Evaluation Measures

Accuracy of Intrusion Detection

- By comparing the classification accuracy of the original testing set and the adversarial example set The more the accuracy rate decreases, the higher the probability of adversarial example set successfully attacks the model.

Rank of the matrix

- By calculating the rank of the two matrixes, we can determine whether the two datasets are equivalent. If the two datasets are equivalent, we can assume that the feature relationship of original dataset have not been changed.

Euclidean Metric

- We calculate the Euclidean Metric between the two PCC matrixes of the original dataset and the adversarial dataset to compare whether the linear relationship of the two datasets is similar.

Feature Types Matching

- By checking the feature types in the adversarial example set, we can know whether they are consistent with the feature types of original examples.

Crafting Intrusion Adversarial Examples

Constraint	Parameters	Accuracy	Rank	Euclidean Metric
Only traffic features	$\alpha = 1/255, T = 4$	0.75	109	0.34
	$\alpha = 2/255, T = 4$	0.74	109	0.59
	$\alpha = 2/255, T = 8$	0.70	109	1.04
Only numerical features	$\alpha = 1/255, T = 4$	0.59	111	5.15
	$\alpha = 2/255, T = 4$	0.42	112	7.14
	$\alpha = 2/255, T = 8$	0.33	113	9.02
All features	$\alpha = 1/255, T = 4$	0.59	111	5.12
	$\alpha = 2/255, T = 4$	0.41	112	7.20
	$\alpha = 2/255, T = 8$	0.32	113	9.15

- Accuracy decreases when the noise value increases, but the relationship between features will change to a greater extent. The Accuracy decreases when the number of modified features increases, but the Rank will satisfy the original testing set.
- We chose ” Modifying all features, α = 2/255,T = 4” as the best parameter setting.

Compared with Baseline

The performance comparison with baseline(IFGSM) under parameters “Modifying all features, α = 2/255,T = 4”

Evaluation Measures	C-IFGSM	IFGSM	Original Dataset
Classification Accuracy	0.41	0.48	0.79
Feature Types Matching	1	0.69	1
Euclidean Metric	7.2	7.82	0
Rank of the Matrixes	112	111	112

- The adversarial examples generated by C-IFGSM makes the intrusion detection classifier get a lower accuracy, a higher feature types matching, a small Euclidean Metric, and a same rank of dataset.
- With the same parameter settings, C-IFGSM can better adapt to network traffic datasets and generate high-quality intrusion adversarial examples.

Verifying the Transferability

Algorithms	Original Dataset	Adversarial Dataset
Decision Tree	0.73	0.25
MLP	0.74	0.73
CNN	0.77	0.68

- We input the intrusion adversarial examples into other classification models and compared the accuracy decreases of other models to verify the transferability of the intrusion adversarial examples.
- C-IFGSM can make an apparent decline on the accuracy of decision tree, and an lesser impact on CNN and MLP models.
- the intrusion adversarial examples generated by C-IFGSM can also attack other models, but the adversarial effect is not very good in other neural network structure models.

Content

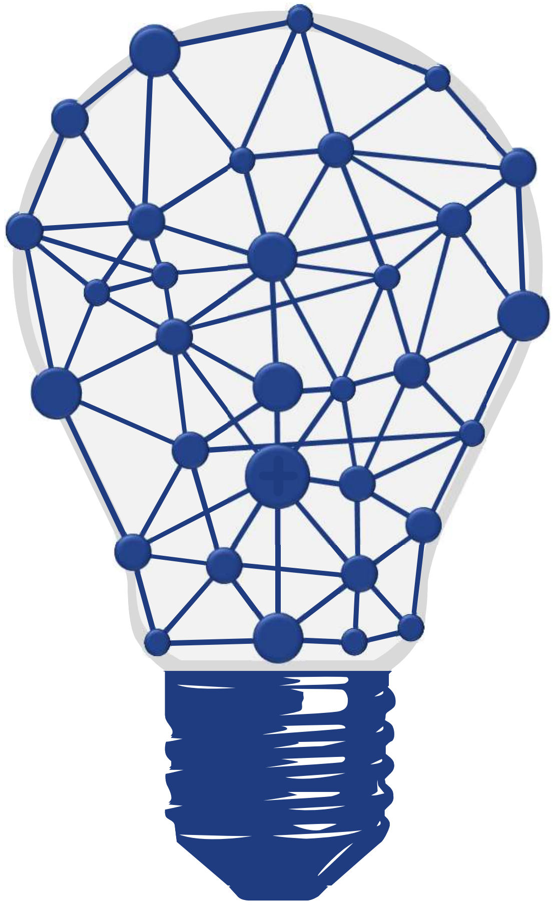
Part Four

Conclusion

Conclusion

Conclusion

- ☐ We analyze different data characteristics between images and network traffic data.
- ☐ Based on the analysis, we implement a C-IFGSM based approach to generating intrusion adversarial examples with satisfying the multiple types and relationship of features.
- ☐ The experiments prove that C-IFGSM can effectively adapt to network traffic datasets and generate high-quality intrusion adversarial examples.



Future Work

- ☐ How to automate the mining of more complex mathematical constraints between features?
- ☐ How to propose a new defense method for the intrusion adversarial examples?

Thanks

Speaker：Yingdi Wang
Email：17120482@bjtu.edu.cn