Machine Learning for Cyber-security Final Project

Jianfei Zhao(jz3766) Hailin Tian(ht1244) Ruiwen Zhao(rz1601) Hongmiao Yu(hy1709)

The following words give a detailed description of our code. We divide this into two parts. In the first part, we describe the methodology of our code, in the second part we show the output of our model.

Methodology

(1)repairing model method

The main idea of our repairing method is Pruning. First, we set the validate dataset as our input and calculate the average activation of each neuron. Second, we sort those neurons with increasing order of average activation. Then, we prune the low average activation neuron in each layer to repair the BadNet. In practice, we found that the performance is not good when we just prune the low average activation neurons, so we use the validate data to refit our network after pruning. Our result shows that refitting will increase the accuracy of validate data and decrease the accuracy of poisoned data in any percentage of neurons pruning. Figure1 shows the accuracy of validate and poisoned data with refitting and without refitting. The x-axis indicates the percentage of neurons we prune in each layer, and the y-axis indicates the accuracy percent.

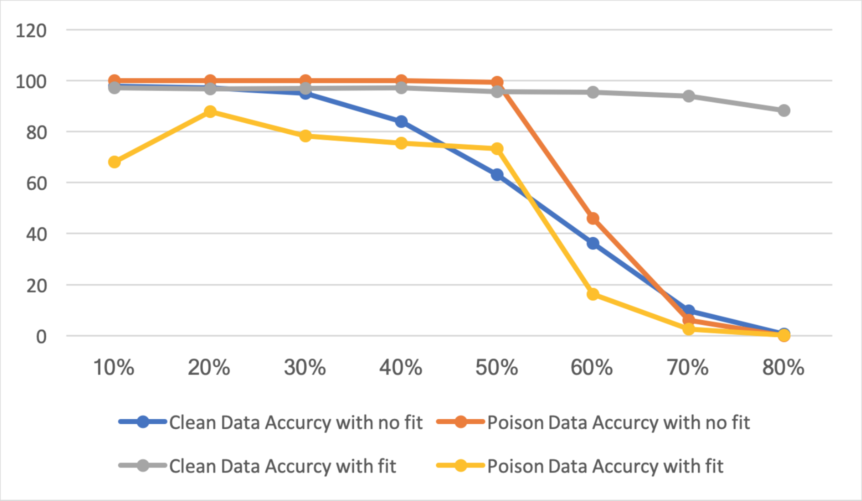


figure1. accuracy of clean and poison data in prune percentage with and without fitting

In real life, the victim may not be able to access the whole validate data for refitting, so we split the validate date and use part of the validate data for refitting. Our results show that even if we use part of the data for refitting, this will still enhance the repairing network. Figure2 shows the accuracy when we use 20% of validate data for refitting. Figure3 shows the accuracy when we use 50% of validate data for refitting.

Based on the finding above, we finally set our model repairing method as below: (1) set the validate dataset as our input and calculate the average activation of each neuron. (2)sort those neurons with increasing order of average activation. (3) prune the low average activation neuron in each layer to repair the BadNet (4) use the validate data to refit the repairing model.

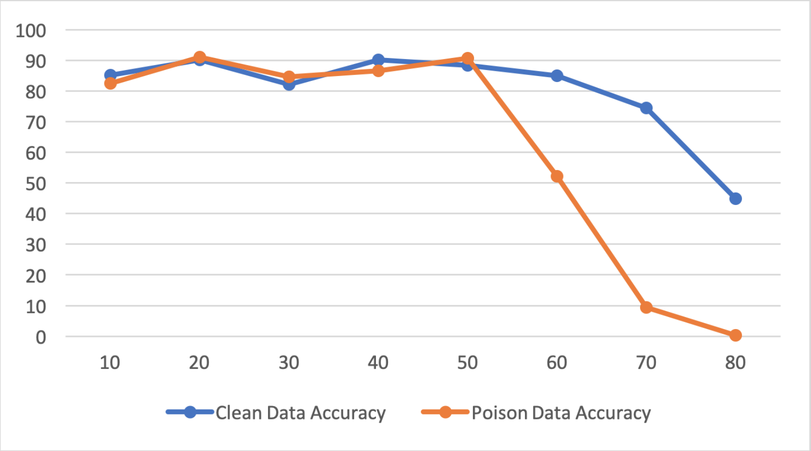


figure2. accuracy of clean and poison data in prune percentage with 20% validate refitting

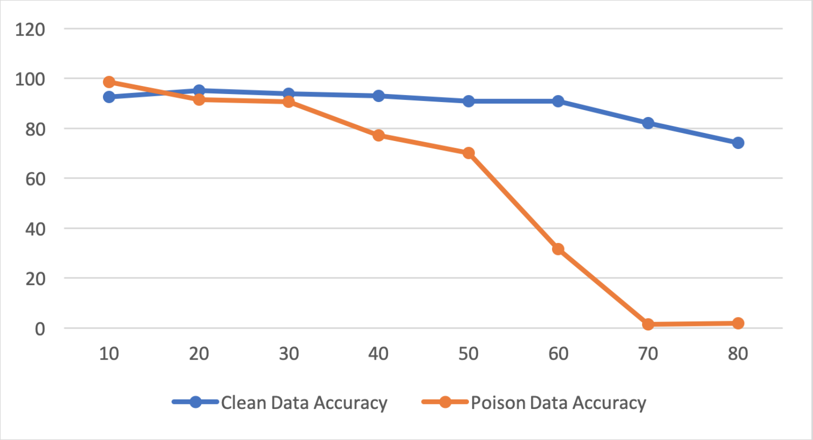


figure3. accuracy of clean and poison data in prune percentage with 50% validate refitting

(2)poison detecting method

Our poison detecting method is to use both the repairing model and BadNet model to predict the text data. If two models output two different labels, we assume this image is a poisoned image. Also, we want to decrease the misclassification of some images that have different output labels in the repairing model and BadNet model but are not poisoned data. Our result shows that even some clean data's output labels are different in the repairing model and the BadNet model, most of them have little difference in the predicted value of two models. The figure3 is based on the validate data and shows the minus between the predicted value of the repairing model and BadNet model on the index of repairing model output label. We can find that the difference of most data is below 0.5. Based on the finding above, we set our two rules on poison detecting: (1)

output label on repairing model and BadNet model are different. (2) BadNet model predict value on the index of repairing model output label minus repairing model predict value on the index of repairing model output label is below 0.5.

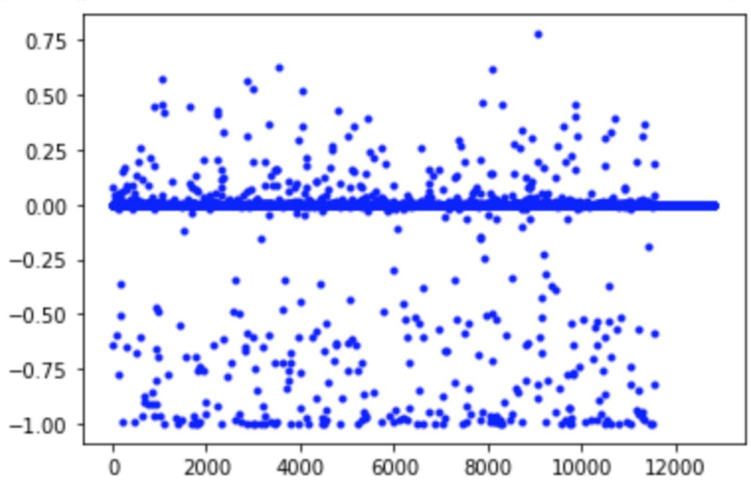
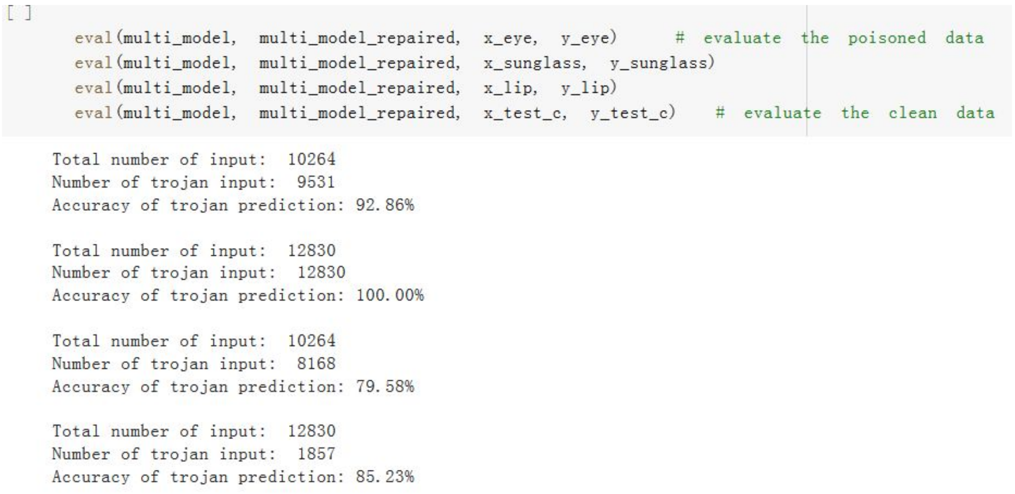


figure3. the value of BadNet model predict value on the index of repairing model output label minus repairing model predict value on the index of repairing model output label on the clean data

Result

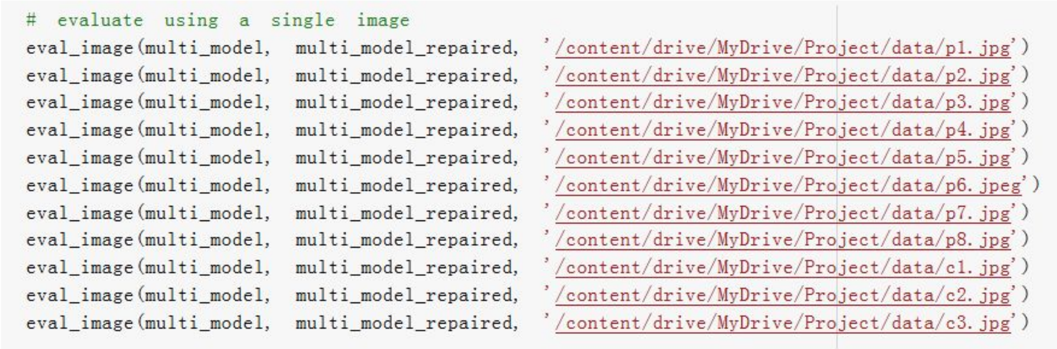
The following part shows our output result on our multi-target repairing model. First, we use the whole data set to test our model. We use eyebrows poisoned data, lipstick poisoned data, sunglasses poisoned data, and clean test data as our input. Figure4 shows the output result of our repairing model. We can see that both clean and poisoned data has a high accuracy of prediction.



```
[ ]
      eval(multi_model,  multi_model_repaired,  x_eye,  y_eye)     #  evaluate  the  poisoned  data
      eval(multi_model,  multi_model_repaired,  x_sunglass,  y_sunglass)
      eval(multi_model,  multi_model_repaired,  x_lip,  y_lip)
      eval(multi_model,  multi_model_repaired,  x_test_c,  y_test_c)    #  evaluate  the  clean  data
```

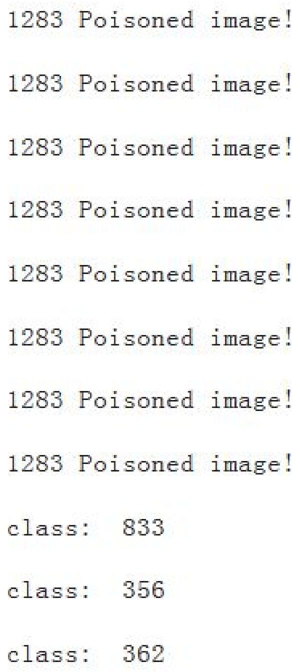
```
Total number of input:  10264
Number of trojan input:  9531
Accuracy of trojan prediction: 92.86%

Total number of input:  12830
Number of trojan input:  12830
Accuracy of trojan prediction: 100.00%

Total number of input:  10264
Number of trojan input:  8168
Accuracy of trojan prediction: 79.58%

Total number of input:  12830
Number of trojan input:  1857
Accuracy of trojan prediction: 85.23%
```

figure4. multi-target repairing model output on whole data set



```
# evaluate using a single image
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p1.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p2.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p3.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p4.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p5.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p6.jpeg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p7.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/p8.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/c1.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/c2.jpg')
eval_image(multi_model, multi_model_repaired, '/content/drive/MyDrive/Project/data/c3.jpg')
```

figure5. The input of Single images testing



```
1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

1283 Poisoned image!

class:  833

class:  356

class:  362
```

figure6. The output of Single images testing

Then, we use single images we created from the dataset to test our model. Figure5 shows the input of our testing. The file name of images starts with 'p' indicates poisoned images and starts with 'c' indicates clean images. Figure6 shows the result of the output of our model on single images. The result shows that our repairing model can classify single images precisely.