

# Machine Learning for Cyber Security ECE-GY 9163

## Lab 2 Report

- an3333\_lab2.ipynb file contains the code.
- Model info (.h5 file) stored in models folder.
- Due to a problem with keras sub-classes, repaired nets could not be saved. So, we can run the script and use the initiated models - repaired\_net\_X\_2, repaired\_net\_X\_4, and repaired\_net\_X\_10 - against the evaluation script. But a new execution of the code is required.

### Test Data Visualization (Clean):



### Sunglasses Poisoned Data Visualization:



## Pruning:

```
## To-do ##
prune_index = []
clean_accuracy_final = [] #classification accuracy
asrate = [] #attack success rate
saved_model = np.zeros(3,dtype=bool)

# Redefine model to output right after the last pooling layer ("pool_3")
intermediate_model = Model(inputs=B.inputs, outputs=B.get_layer('pool_3').output)

# Get feature map for last pooling layer ("pool_3") using the clean validation
#data and intermediate_model
feature_maps_cl = intermediate_model.predict(cl_x_valid)

# Get average activation value of each channel in last pooling layer ("pool_3")
averageActivationsCl = np.mean(feature_maps_cl,axis=(0,1,2))

# Store the indices of average activation values (averageActivationsCl)
#in increasing order
idxToPrune = np.argsort(averageActivationsCl)

# Get the conv_3 layer weights and biases from the original network
# that will be used for pruning
# Hint: Use the get_weights() method (https://stackoverflow.com/questions/4371504)
lastConvLayerWeights = B_clone.layers[5].get_weights()[0]
lastConvLayerBiases = B_clone.layers[5].get_weights()[1]

for chIdx in idxToPrune:

    # Prune one channel at a time
    # Hint: Replace all values in channel 'chIdx' of lastConvLayerWeights
    #and lastConvLayerBiases with 0
    lastConvLayerWeights[:, :, :, chIdx] = 0
    lastConvLayerBiases[chIdx] = 0

    # Update weights and biases of B_clone
    # Hint: Use the set_weights() method
    B_clone.layers[5].set_weights([lastConvLayerWeights, lastConvLayerBiases])

    # Evaluate the updated model's (B_clone) clean validation accuracy
    cl_label_p_valid = np.argmax(B_clone(cl_x_valid), axis=1)
    clean_accuracy_valid = np.mean(np.equal(cl_label_p_valid, cl_y_valid)) * 100

    # If drop in clean_accuracy_valid is just greater (or equal to)
    # than the desired threshold
    #compared to clean_accuracy, then save B_clone as B_prime and break
    if (clean_accuracy-clean_accuracy_valid >=2 and not saved_model[0]):
        print("The accuracy drops at least 2%, saved the model")
        B_clone.save('model_X_2.h5')
        saved_model[0] = 1
    if (clean_accuracy-clean_accuracy_valid >=4 and not saved_model[1]):
        print("The accuracy drops at least 4%, saved the model")
        B_clone.save('model_X_4.h5')
        saved_model[1] = 1
    if (clean_accuracy-clean_accuracy_valid >=10 and not saved_model[2]):
        print("The accuracy drops at least 10%, saved the model")
        B_clone.save('model_X_10.h5')
        saved_model[2] = 1
    # Save B_clone as B_prime and break
    break
```

361/361 [=====] - 11s 30ms/step

The accuracy drops at least 2%, saved the model

The accuracy drops at least 4%, saved the model

The accuracy drops at least 10%, saved the model

## Performance of the Repaired model on the Test Data:

```
401/401 [=====] - 1s 3ms/step
Clean Classification accuracy for B_prime_X_2: 95.90023382696803
401/401 [=====] - 1s 2ms/step
Clean Classification accuracy for B_prime_X_4: 92.29150428682775
401/401 [=====] - 1s 2ms/step
Clean Classification accuracy for B_prime_X_10: 84.54403741231489
401/401 [=====] - 1s 2ms/step
Attack Success Rate for B_prime_X_2: 100.0
401/401 [=====] - 1s 3ms/step
Attack Success Rate for B_prime_X_4: 99.98441153546376
401/401 [=====] - 1s 2ms/step
Attack Success Rate for B_prime_X_10: 77.20966484801247
```

## Performance of the Original model on the Test Data:

```
☞ 401/401 [=====] - 1s 2ms/step
Clean Classification accuracy for B: 98.62042088854248
401/401 [=====] - 1s 2ms/step
Attack Success Rate for B: 100.0
```

## Performance of the Repaired Net on the Test Data for $X=\{2,4,10\}\%$ :

```
Clean Classification accuracy for repaired net_X_2: 95.74434918160561
Clean Classification accuracy for repaired net_X_4: 92.1278254091972
Clean Classification accuracy for repaired net_X_10: 84.3335931410756
Attack Success Rate for repaired net_X_2: 100.0
Attack Success Rate for repaired net_X_4: 99.98441153546376
Attack Success Rate for repaired net_X_10: 77.20966484801247
```

## Observations:

- Even while the performance is greater, it is not noticeably better, and this may be changed by adversarial retraining.
- Pruning is done for one channel at a time.
- Additionally, the repaired networks for each X are kept in the models section of the github repository and may be tested against the eval.py script there.