ML_Lab_3 (1)

December 3, 2023

[1]:
```
import h5py
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Add,␣
↪Activation
from sklearn.metrics import accuracy_score
import keras
```

```
/home/as16494/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:518:
UserWarning: The value of the smallest subnormal for <class 'numpy.float32'>
type is zero.
  setattr(self, word, getattr(machar, word).flat[0])
/home/as16494/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:89:
UserWarning: The value of the smallest subnormal for <class 'numpy.float32'>
type is zero.
  return self._float_to_str(self.smallest_subnormal)
/home/as16494/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:518:
UserWarning: The value of the smallest subnormal for <class 'numpy.float64'>
type is zero.
  setattr(self, word, getattr(machar, word).flat[0])
/home/as16494/.local/lib/python3.8/site-packages/numpy/core/getlimits.py:89:
UserWarning: The value of the smallest subnormal for <class 'numpy.float64'>
type is zero.
  return self._float_to_str(self.smallest_subnormal)
```

[2]:
```
import tensorflow as tf
badnet_model = tf.keras.models.load_model('bd_net.h5')
```

[3]:
```
badnet_model.summary()
```

```
Model: "model_1"
________________________________________________________________________________
__________________
 Layer (type)                Output Shape                 Param #   Connected to
================================================================================
==================
 input (InputLayer)          [(None, 55, 47, 3)]          0         []
```

```
 conv_1 (Conv2D)             (None, 52, 44, 20)           980
['input[0][0]']

 pool_1 (MaxPooling2D)       (None, 26, 22, 20)           0
['conv_1[0][0]']

 conv_2 (Conv2D)             (None, 24, 20, 40)           7240
['pool_1[0][0]']

 pool_2 (MaxPooling2D)       (None, 12, 10, 40)           0
['conv_2[0][0]']

 conv_3 (Conv2D)             (None, 10, 8, 60)            21660
['pool_2[0][0]']

 pool_3 (MaxPooling2D)       (None, 5, 4, 60)             0
['conv_3[0][0]']

 conv_4 (Conv2D)             (None, 4, 3, 80)             19280
['pool_3[0][0]']

 flatten_1 (Flatten)         (None, 1200)                 0
['pool_3[0][0]']

 flatten_2 (Flatten)         (None, 960)                  0
['conv_4[0][0]']

 fc_1 (Dense)                (None, 160)                  192160
['flatten_1[0][0]']

 fc_2 (Dense)                (None, 160)                  153760
['flatten_2[0][0]']

 add_1 (Add)                 (None, 160)                  0
['fc_1[0][0]',
'fc_2[0][0]']

 activation_1 (Activation)   (None, 160)                  0
['add_1[0][0]']

 output (Dense)              (None, 1283)                 206563
['activation_1[0][0]']

================================================================================
==================
Total params: 601643 (2.30 MB)
Trainable params: 601643 (2.30 MB)
```

```
Non-trainable params: 0 (0.00 Byte)
________________________________________________________________________________
__________________
```

[4]:
```python
def load_dataset(h5_file_path):
    with h5py.File(h5_file_path, 'r') as file:
        features = np.array(file['data'])
        labels = np.array(file['label'])
        features = np.transpose(features, (0, 2, 3, 1))

    return features, labels
```

[5]:
```python
cl_x_valid, cl_y_valid = load_dataset('valid.h5')
cl_x_test, cl_y_test = load_dataset('test.h5')
bd_x_valid, bd_y_valid = load_dataset('bd_valid.h5')
bd_x_test, bd_y_test = load_dataset('bd_test.h5')
N = int(cl_y_test.max())
```

[6]:
```python
backdoor_model = keras.models.load_model('bd_net.h5')

def evaluate_model_performance(model, clean_data, clean_labels, backdoor_data,␣
 ↪backdoor_labels):
    clean_predictions = np.argmax(model.predict(clean_data), axis=1)
    clean_accuracy = accuracy_score(clean_labels, clean_predictions) * 100
    backdoor_predictions = np.argmax(model.predict(backdoor_data), axis=1)
    attack_success_rate = accuracy_score(backdoor_labels, backdoor_predictions)␣
 ↪* 100
    return clean_accuracy, attack_success_rate


# Evaluate the model
clean_accuracy, attack_success_rate = evaluate_model_performance(
    backdoor_model, cl_x_test, cl_y_test, bd_x_test, bd_y_test
)

print('Clean Classification Accuracy:', clean_accuracy)
print('Attack Success Rate:', attack_success_rate)
```

```
401/401 [==============================] - 2s 3ms/step
401/401 [==============================] - 1s 3ms/step
Clean Classification Accuracy: 98.62042088854248
Attack Success Rate: 100.0
```

[7]:
```python
def createPrunedModel(original_model, X_threshold):
    # Copy the model structure and weights
    modified_model = keras.models.clone_model(original_model)
    modified_model.set_weights(original_model.get_weights())
```

```
    initial_predictions = np.argmax(original_model.predict(cl_x_valid), axis=1)
    initial_accuracy = np.mean(initial_predictions == cl_y_valid) * 100

    # Target the specific layer for pruning
    target_layer = modified_model.get_layer('conv_3')
    activation_model = keras.Model(inputs=modified_model.input,␣
↪outputs=target_layer.output)
    channel_activations = activation_model.predict(cl_x_valid).sum(axis=(0, 1,␣
↪2))

    channels_sorted_by_activation = np.argsort(channel_activations)

    for channel_index in channels_sorted_by_activation:
        # Modify the weights of the target layer to "remove" a channel
        layer_weights = target_layer.get_weights()
        layer_weights[0][:, :, :, channel_index] = 0  # Set the weights of the␣
↪channel to zero
        target_layer.set_weights(layer_weights)

        modified_predictions = np.argmax(modified_model.predict(cl_x_valid),␣
↪axis=1)
        modified_accuracy = np.mean(modified_predictions == cl_y_valid) * 100

        if initial_accuracy - modified_accuracy > X_threshold:
            target_layer.set_weights(target_layer.get_weights())
            break

    return modified_model
```

[8]:
```
def evaluateModelPerformance(pruned_model, original_model):
    # Accuracy evaluation on clean test data
    pred_pruned_clean = np.argmax(pruned_model.predict(cl_x_test), axis=1)
    pred_original_clean = np.argmax(original_model.predict(cl_x_test), axis=1)

    clean_predictions = [pred if pred == pred_original else N + 1 for pred,␣
↪pred_original in zip(pred_pruned_clean, pred_original_clean)]
    accuracy = np.mean(np.array(clean_predictions) == cl_y_test) * 100

    # ASR evaluation on backdoored test data
    pred_pruned_bd = np.argmax(pruned_model.predict(bd_x_test), axis=1)
    pred_original_bd = np.argmax(original_model.predict(bd_x_test), axis=1)

    bd_predictions = [pred if pred == pred_original else N + 1 for pred,␣
↪pred_original in zip(pred_pruned_bd, pred_original_bd)]
    asr = np.mean(np.array(bd_predictions) == bd_y_test) * 100

    return accuracy, asr
```

[9]:
```python
Xs = [2, 4, 10]
for X in Xs:

    modified_model = createPrunedModel(badnet_model, X)
    # Assess the performance of the pruned model
    acc, asr = evaluateModelPerformance(modified_model, badnet_model)
    print(f"Drop = {X}%\n\tAccuracy of Pruned Model = {acc}%\n\tAttack Success Rate = {asr}%\n")
    modified_model.save(f"models/bd_prime_{X}.h5")
```

```
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 2ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
```

```
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
Drop = 2%
        Accuracy of Pruned Model = 95.8846453624318%
        Attack Success Rate = 100.0%

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

/home/as16494/.local/lib/python3.8/site-
packages/keras/src/engine/training.py:3000: UserWarning: You are saving your
model as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(

361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 2ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
```

```
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
Drop = 4%
        Accuracy of Pruned Model = 94.61418550272798%
        Attack Success Rate = 99.97661730319564%

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 2ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
```

```
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
361/361 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
```

```
401/401 [==============================] - 1s 3ms/step
401/401 [==============================] - 1s 3ms/step
Drop = 10%
        Accuracy of Pruned Model = 84.45830085736556%
        Attack Success Rate = 76.1730319563523%

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

[ ]: