ML_Sec_Lab3_ac9025

December 16, 2021

1 Backdoor detector and pruning Defence

[ ]:
```
!git clone https://github.com/csaw-hackml/CSAW-HackML-2020.git
```

```
Cloning into 'CSAW-HackML-2020'…
remote: Enumerating objects: 220, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 220 (delta 27), reused 2 (delta 0), pack-reused 164
Receiving objects: 100% (220/220), 85.94 MiB | 30.00 MiB/s, done.
Resolving deltas: 100% (82/82), done.
```

[ ]:
```
!cp -r ./drive/MyDrive/Lab3/* ./CSAW-HackML-2020/lab3/data/
```

[ ]:
```
import keras
import keras.backend as K
from keras import models
from keras.models import Model
from keras import initializers

import tensorflow as tf
import h5py
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
def data_loader(filepath):  # from eval.py
    data = h5py.File(filepath, 'r')
    x_data = np.array(data['data'])
    y_data = np.array(data['label'])
    x_data = x_data.transpose((0, 2, 3, 1))
    return x_data, y_data
```

Paths to the data and the models

[ ]:
```
bad_model = './CSAW-HackML-2020/lab3/models/bd_net.h5'
bad_model_weights = './CSAW-HackML-2020/lab3/models/bd_weights.h5'

clean_test = './CSAW-HackML-2020/lab3/data/cl/test.h5'
```

```
clean_valid = './CSAW-HackML-2020/lab3/data/cl/valid.h5'
poison_test = './CSAW-HackML-2020/lab3/data/bd/bd_test.h5'
poison_valid = './CSAW-HackML-2020/lab3/data/bd/bd_valid.h5'

x_clean_test, y_clean_test = data_loader(clean_test)
x_clean_valid, y_clean_valid = data_loader(clean_valid)
x_poison_test, y_poison_test = data_loader(poison_test)
x_poison_valid, y_poison_valid = data_loader(poison_valid)
```

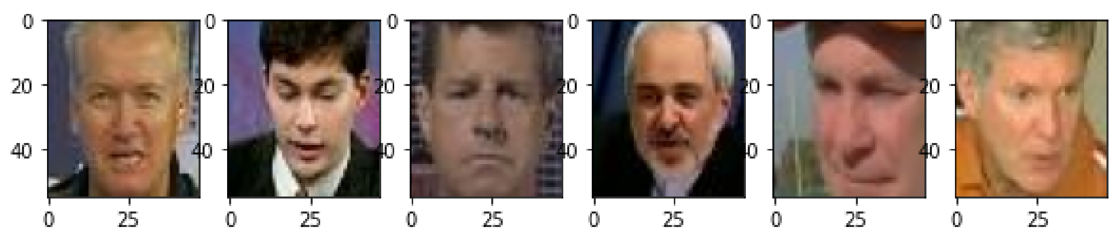
Plotting some test images.

[ ]:
```
plt.figure(figsize=(10,2))
for id in range(1,7):
    plt.subplot(1,6,id)
    plt.imshow(x_clean_test[id]/255)

plt.figure(figsize=(10,2))
for id in range(1,7):
    plt.subplot(1,6,id)
    plt.imshow(x_clean_valid[id]/255)

plt.figure(figsize=(10,3))
for id in range(1,7):
    plt.subplot(1,6,id)
    plt.imshow(x_poison_test[id]/255)

plt.figure(figsize=(10,2))
for id in range(1,7):
    plt.subplot(1,6,id)
    plt.imshow(x_poison_valid[id]/255)
```



Checking the accuracy on the clean and poison data before pruning

```
[ ]: def check_acc(model,x,y):
       label = np.argmax(model.predict(x), axis=1)
       acc = np.mean(np.equal(label,y))*100
       return acc
```

```
[ ]: badnet = keras.models.load_model(bad_model)
     print("Bad net accuracy for clean validation images:␣
      ↪",check_acc(badnet,x_clean_valid,y_clean_valid))
     print("Bad net accuracy for clean test images:␣
      ↪",check_acc(badnet,x_clean_test,y_clean_test))
     print("Bad net accuracy for poison validation images:␣
      ↪",check_acc(badnet,x_poison_valid,y_poison_valid))
     print("Bad net accuracy for poison test images:␣
      ↪",check_acc(badnet,x_poison_test,y_poison_test))
```

```
Bad net accuracy for clean validation images:  98.64899974019225
Bad net accuracy for clean test images:  98.62042088854248
Bad net accuracy for poison validation images:  100.0
Bad net accuracy for poison test images:  100.0
```

1.1 Pruning the badnet model

[ ]:
```
def pruner(model, layer_name, x_clean_valid, num_prune):
  m_input = model.input
  m_output = model.get_layer(layer_name).output
  partial_m = Model(inputs = m_input, outputs = m_output)

  conv3out = partial_m.predict(x_clean_valid)
  chann = conv3out.shape[3]
  num_sample = np.shape(conv3out)[0]
  conv3out_sum = [np.sum(conv3out[:, :, :, i]) for i in range(chann)]
  ids_sorted = np.argsort(conv3out_sum)

  og_weights, og_bias = model.get_layer(layer_name).get_weights()

  for i in range(num_prune):
    id_neuron = ids_sorted[i]
    og_weights[:, :, :, id_neuron] = np.zeros(np.shape(og_weights[:, :, :,␣
↪id_neuron]))
    og_bias[id_neuron]=0

  model.get_layer(layer_name).set_weights((og_weights,og_bias))
  return model
```

Prune the neurons in the increasing order of activation.

[ ]:
```
clean_acc = []
poison_acc = []
```

[ ]:
```
for i in range(54):
  K.clear_session()
  badnet = keras.models.load_model(bad_model)
  pruned_model = pruner(badnet,"conv_3",x_clean_valid,i)
  print("Number of pruned neurons: ",i)
  acc_c = check_acc(pruned_model,x_clean_valid,y_clean_valid)
  acc_p = check_acc(pruned_model,x_poison_valid,y_poison_valid)
  clean_acc.append(acc_c)
  poison_acc.append(acc_p)
  print("Pruned net on clean validation data acc: ",acc_c)
  print("Pruned net on poison validation data acc: ",acc_p)
```

```
Number of pruned neurons:  0
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  1
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  2
Pruned net on clean validation data acc:  98.64899974019225
```

```
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  3
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  4
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  5
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  6
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  7
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  8
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  9
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  10
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  11
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  12
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  13
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  14
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  15
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  16
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  17
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  18
Pruned net on clean validation data acc:  98.64899974019225
```

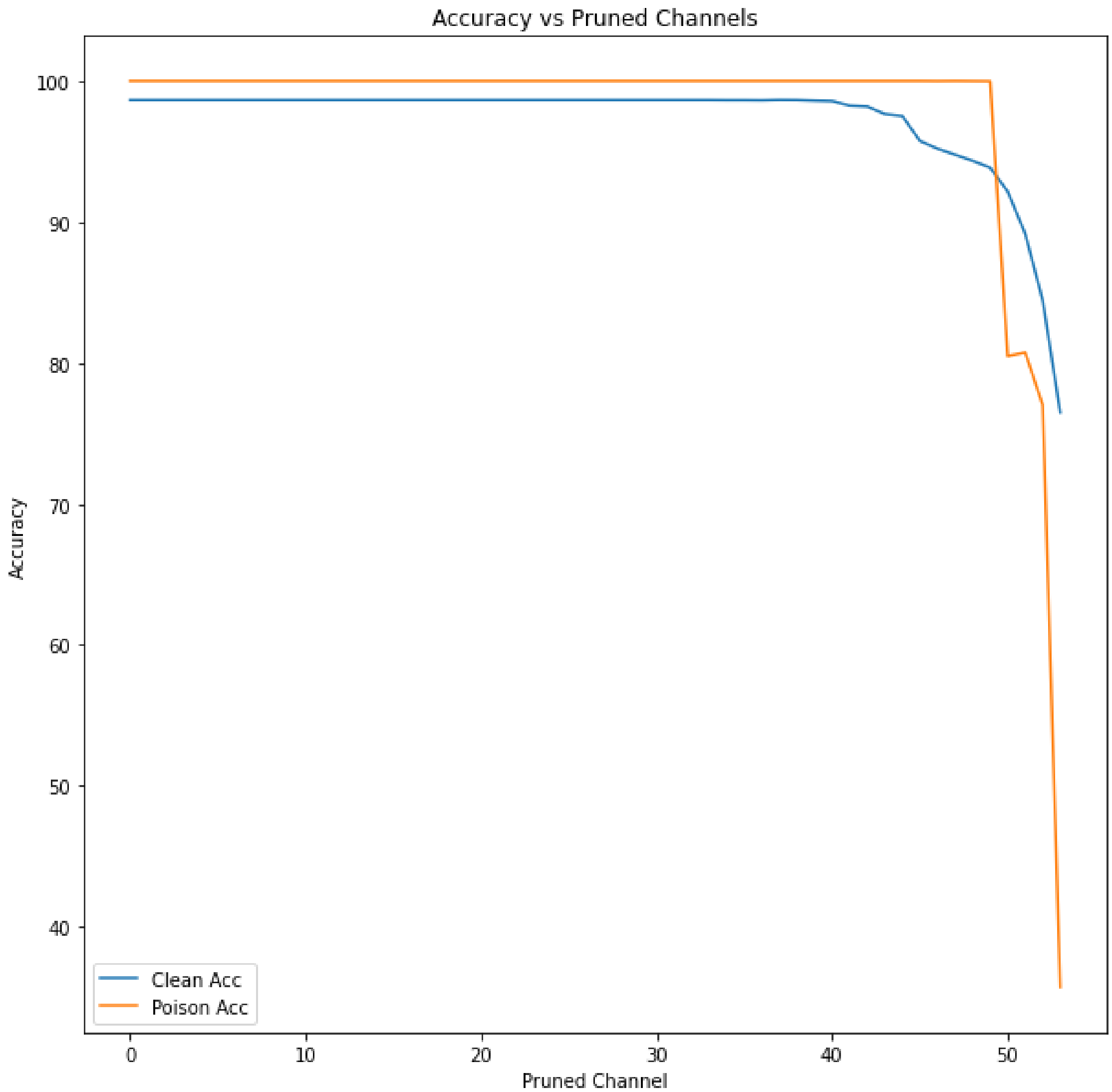
```
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  19
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  20
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  21
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  22
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  23
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  24
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  25
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  26
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  27
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  28
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  29
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  30
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  31
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  32
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  33
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  34
Pruned net on clean validation data acc:  98.64033948211657
```

```
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  35
Pruned net on clean validation data acc:  98.64033948211657
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  36
Pruned net on clean validation data acc:  98.63167922404088
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  37
Pruned net on clean validation data acc:  98.65765999826795
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  38
Pruned net on clean validation data acc:  98.64899974019225
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  39
Pruned net on clean validation data acc:  98.6056984498138
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  40
Pruned net on clean validation data acc:  98.57105741751104
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  41
Pruned net on clean validation data acc:  98.25062786871048
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  42
Pruned net on clean validation data acc:  98.19000606218066
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  43
Pruned net on clean validation data acc:  97.65307006148784
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  44
Pruned net on clean validation data acc:  97.50584567420108
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  45
Pruned net on clean validation data acc:  95.75647354291158
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  46
Pruned net on clean validation data acc:  95.20221702606739
Pruned net on poison validation data acc:  99.9913397419243
Number of pruned neurons:  47
Pruned net on clean validation data acc:  94.77786438035854
Pruned net on poison validation data acc:  100.0
Number of pruned neurons:  48
Pruned net on clean validation data acc:  94.344851476574
Pruned net on poison validation data acc:  99.9913397419243
Number of pruned neurons:  49
Pruned net on clean validation data acc:  93.85121676625964
Pruned net on poison validation data acc:  99.98267948384861
Number of pruned neurons:  50
Pruned net on clean validation data acc:  92.16246644149996
```

```
Pruned net on poison validation data acc:  80.47977829739327
Number of pruned neurons:  51
Pruned net on clean validation data acc:  89.17467740538669
Pruned net on poison validation data acc:  80.73958603966398
Number of pruned neurons:  52
Pruned net on clean validation data acc:  84.43751623798389
Pruned net on poison validation data acc:  77.015675067117
Number of pruned neurons:  53
Pruned net on clean validation data acc:  76.48739932449988
Pruned net on poison validation data acc:  35.71490430414826
```

[ ]:
```
num = np.arange(54)
plt.figure(figsize=(10,10))
plt.plot(num,clean_acc,label="Clean Acc")
plt.plot(num,poison_acc,label="Poison Acc")
plt.legend()
plt.xlabel('Pruned Channel')
plt.ylabel('Accuracy')
plt.title('Accuracy vs Pruned Channels')
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy vs Pruned Channels')
```



X=2% less accuracy than original model.

[ ]:
```
badnet = keras.models.load_model(bad_model)
pruned_model = pruner(badnet,"conv_3",x_clean_valid,45)
print("Number of pruned channels: ",45)
print("Pruned net on clean validation data acc:␣
↪",check_acc(pruned_model,x_clean_valid,y_clean_valid))
print("Pruned net on poison validation data acc:␣
↪",check_acc(pruned_model,x_poison_valid,y_poison_valid))
```

```
Number of pruned channels:  45
Pruned net on clean validation data acc:  95.75647354291158
Pruned net on poison validation data acc:  100.0
```

[ ]:
```
pruned_model.save("./drive/MyDrive/Local_Lab3/pruned_net_2.h5")
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/functional.py:1410:
CustomMaskWarning: Custom mask layers require a config and must override
get_config. When loading, the custom mask layer must be passed to the
custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

X=4% less accuracy than original model.

[ ]:
```
badnet = keras.models.load_model(bad_model)
pruned_model = pruner(badnet,"conv_3",x_clean_valid,48)
print("Number of pruned channels: ",48)
print("Pruned net on clean validation data acc:␣
 ↪",check_acc(pruned_model,x_clean_valid,y_clean_valid))
print("Pruned net on poison validation data acc:␣
 ↪",check_acc(pruned_model,x_poison_valid,y_poison_valid))
```

```
Number of pruned channels:  48
Pruned net on clean validation data acc:  94.344851476574
Pruned net on poison validation data acc:  99.9913397419243
```

[ ]:
```
pruned_model.save("./drive/MyDrive/Local_Lab3/pruned_net_4.h5")
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/functional.py:1410:
CustomMaskWarning: Custom mask layers require a config and must override
get_config. When loading, the custom mask layer must be passed to the
custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

X=10% less accuracy than the original model

[ ]:
```
badnet = keras.models.load_model(bad_model)
pruned_model = pruner(badnet,"conv_3",x_clean_valid,51)
print("Number of pruned channels: ",51)
print("Pruned net on clean validation data acc:␣
 ↪",check_acc(pruned_model,x_clean_valid,y_clean_valid))
print("Pruned net on poison validation data acc:␣
 ↪",check_acc(pruned_model,x_poison_valid,y_poison_valid))
```

```
Number of pruned channels:  51
Pruned net on clean validation data acc:  89.17467740538669
Pruned net on poison validation data acc:  80.73958603966398
```

[ ]:
```
pruned_model.save("./drive/MyDrive/Local_Lab3/pruned_net_10.h5")
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/functional.py:1410:
CustomMaskWarning: Custom mask layers require a config and must override
get_config. When loading, the custom mask layer must be passed to the
custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

X=30% less accuracy than original model

```
[ ]: badnet = keras.models.load_model(bad_model)
pruned_model = pruner(badnet,"conv_3",x_clean_valid,53)
print("Number of pruned channels: ",53)
print("Pruned net on clean validation data acc:␣
 ↪",check_acc(pruned_model,x_clean_valid,y_clean_valid))
print("Pruned net on poison validation data acc:␣
 ↪",check_acc(pruned_model,x_poison_valid,y_poison_valid))
```

```
Number of pruned channels:  53
Pruned net on clean validation data acc:  76.48739932449988
Pruned net on poison validation data acc:  35.71490430414826
```

```
[ ]: print("Pruned net on clean test data acc:␣
 ↪",check_acc(pruned_model,x_clean_test,y_clean_test))
```

```
Pruned net on clean test data acc:  76.30553390491036
```

```
[ ]: pruned_model.save("./drive/MyDrive/Local_Lab3/B1.h5")
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/functional.py:1410:
CustomMaskWarning: Custom mask layers require a config and must override
get_config. When loading, the custom mask layer must be passed to the
custom_objects argument.
  layer_config = serialize_layer_fn(layer)
```

1.2 Making the good net using Badnet B and Pruned model B'

```
[ ]: from keras.layers.merge import concatenate
from keras.utils.vis_utils import plot_model

good_model = "./drive/MyDrive/B1.h5"
B = keras.models.load_model(bad_model)
B1 = keras.models.load_model(good_model)
members = [B,B1]
```

```
[ ]: def compare_class(x):
  out1 = x[0]
  out2 = x[1]
  ans = tf.where(out1==out2, out2, 1283)
  return ans
```

The model G will be combine both the goodnet and Badnet into a single model. The argmax function and the if condition to check for backdoored input is included in the layers therefore the Evaluation script will be different than regular model.

```
[ ]: def model_G(members):
  for i in range(len(members)):
```

```
    model = members[i]
    term = ('badnet' if i==0 else 'prunednet')
    for layer in model.layers:
      layer.trainable = False
      layer._name = term+str(i+1)+'_'+layer.name
  model_inputs = [model.input for model in members]
  out1 = keras.layers.Lambda(K.argmax, arguments={'axis': 1})(members[0].output)
  out2 = keras.layers.Lambda(K.argmax, arguments={'axis': 1})(members[1].output)
  out = keras.layers.Lambda(compare_class)([out1, out2])
  model = Model(inputs=model_inputs, outputs=out)
  plot_model(model, show_shapes=True, to_file='./drive/MyDrive/Local_Lab3/
↪model_graph.png')
  return model
```

[ ]:
```
G = model_G(members)
```

[ ]:
```
G.summary()
```

```
Model: "model_4"
________________________________________________________________________________
__________________
 Layer (type)                   Output Shape          Param #     Connected to
================================================================================
==================
 badnet1_input (InputLayer)     [(None, 55, 47, 3)]   0           []

 prunednet2_input (InputLayer)  [(None, 55, 47, 3)]   0           []

 badnet1_conv_1 (Conv2D)        (None, 52, 44, 20)    980
['badnet1_input[0][0]']

 prunednet2_conv_1 (Conv2D)     (None, 52, 44, 20)    980
['prunednet2_input[0][0]']

 badnet1_pool_1 (MaxPooling2D)  (None, 26, 22, 20)    0
['badnet1_conv_1[0][0]']

 prunednet2_pool_1 (MaxPooling2  (None, 26, 22, 20)   0
['prunednet2_conv_1[0][0]']
 D)

 badnet1_conv_2 (Conv2D)        (None, 24, 20, 40)    7240
['badnet1_pool_1[0][0]']

 prunednet2_conv_2 (Conv2D)     (None, 24, 20, 40)    7240
['prunednet2_pool_1[0][0]']

 badnet1_pool_2 (MaxPooling2D)  (None, 12, 10, 40)    0
```

```
['badnet1_conv_2[0][0]']

 prunednet2_pool_2 (MaxPooling2  (None, 12, 10, 40)  0
['prunednet2_conv_2[0][0]']
 D)

 badnet1_conv_3 (Conv2D)        (None, 10, 8, 60)    21660
['badnet1_pool_2[0][0]']

 prunednet2_conv_3 (Conv2D)     (None, 10, 8, 60)    21660
['prunednet2_pool_2[0][0]']

 badnet1_pool_3 (MaxPooling2D)  (None, 5, 4, 60)     0
['badnet1_conv_3[0][0]']

 prunednet2_pool_3 (MaxPooling2  (None, 5, 4, 60)    0
['prunednet2_conv_3[0][0]']
 D)

 badnet1_conv_4 (Conv2D)        (None, 4, 3, 80)     19280
['badnet1_pool_3[0][0]']

 prunednet2_conv_4 (Conv2D)     (None, 4, 3, 80)     19280
['prunednet2_pool_3[0][0]']

 badnet1_flatten_1 (Flatten)    (None, 1200)         0
['badnet1_pool_3[0][0]']

 badnet1_flatten_2 (Flatten)    (None, 960)          0
['badnet1_conv_4[0][0]']

 prunednet2_flatten_1 (Flatten)  (None, 1200)        0
['prunednet2_pool_3[0][0]']

 prunednet2_flatten_2 (Flatten)  (None, 960)         0
['prunednet2_conv_4[0][0]']

 badnet1_fc_1 (Dense)           (None, 160)          192160
['badnet1_flatten_1[0][0]']

 badnet1_fc_2 (Dense)           (None, 160)          153760
['badnet1_flatten_2[0][0]']

 prunednet2_fc_1 (Dense)        (None, 160)          192160
['prunednet2_flatten_1[0][0]']

 prunednet2_fc_2 (Dense)        (None, 160)          153760
['prunednet2_flatten_2[0][0]']
```

```
 badnet1_add_1 (Add)            (None, 160)          0
['badnet1_fc_1[0][0]',
'badnet1_fc_2[0][0]']

 prunednet2_add_1 (Add)         (None, 160)          0
['prunednet2_fc_1[0][0]',
'prunednet2_fc_2[0][0]']

 badnet1_activation_1 (Activati  (None, 160)         0
['badnet1_add_1[0][0]']
 on)

 prunednet2_activation_1 (Activ  (None, 160)         0
['prunednet2_add_1[0][0]']
 ation)

 badnet1_output (Dense)         (None, 1283)         206563
['badnet1_activation_1[0][0]']

 prunednet2_output (Dense)      (None, 1283)         206563
['prunednet2_activation_1[0][0]']

 lambda (Lambda)                (None,)              0
['badnet1_output[0][0]']

 lambda_1 (Lambda)              (None,)              0
['prunednet2_output[0][0]']

 lambda_2 (Lambda)              (None,)              0
['lambda[0][0]',
'lambda_1[0][0]']

================================================================================
==================
Total params: 1,203,286
Trainable params: 0
Non-trainable params: 1,203,286
--------------------------------------------------------------------------------
------------------
```

[ ]:
```
#assigning poison class to 1283 the last class N+1
poison_class = np.ones(np.shape(y_poison_test))*1283

label = G.predict([x_clean_test,x_clean_test]) # As it has 2 subnets therefore␣
 ↪replicating the data.
acc_clean = np.mean(np.equal(label,y_clean_test))*100
```

```python
print("Good net for Clean test data acc: ",acc_clean)

label = G.predict([x_poison_test,x_poison_test])
acc_poison = np.mean(np.equal(label,y_poison_test))*100
print("Good net for poison test data acc: ",acc_poison)
```

```
Good net for Clean test data acc:  76.16523772408418
Good net for poison test data acc:  36.26656274356976
```