First, we import the relevant packages

In [2]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import keras
import sys
import h5py
import warnings
```

```
Using TensorFlow backend.
```

Data processing

setting up the data path,data loading function

In [3]:

```
filePath = "E:/360MoveData/Users/11813/Desktop/NYU2021Fall/MLforCyber/labs/lab3/"
```

In [4]:

```
clean_validation_data = filePath + 'valid.h5'
poisoned_validation_data = filePath + 'bd_valid.h5'
modelName = filePath +  'bd_net.h5'
```

In [5]:

```
def data_load(filepath):
    data = h5py.File(filepath, 'r')
    x = np.array(data['data'])
    y = np.array(data['label'])
    x = x.transpose((0,2,3,1))
    return x, y
```

data display

In [6]:

```
x, y = data_load(clean_validation_data)
```

In [8]:

```
print(x.shape[0])
```

```
11547
```

In [9]:

```
# Plot clean data picture
figure = plt.figure(figsize=(12,10))
n, m = 4,4
for i in range(1, n*m+1):
  index = np.random.randint(x.shape[0], size=1)
  img, label = (x[index], y[index])
  figure.add_subplot(n, m, i)
  plt.title("Label: {}".format(label))
  plt.axis("off")
  plt.imshow(img[0]/255)
plt.show()
```

In [10]:

```
x, y = data_load(poisoned_validation_data)
```

In [11]:

```
# Plot poisoned data picture
figure = plt.figure(figsize=(12,10))
n, m = 4,4
for i in range(1, n*m+1):
  index = np.random.randint(x.shape[0], size=1)
  img, label = (x[index], y[index])
  figure.add_subplot(n, m, i)
  plt.title("Label: {}".format(label))
  plt.axis("off")
  plt.imshow(img[0]/255)
plt.show()
```

Model loading and process

Show model details

In [50]:

```
model = keras.models.load_model(modelName)

print(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: 601,643
Trainable params: 601,643
Non-trainable params: 0
__________________________________________________________________________________
_______________
None
```

In [31]:

```
#Model test
if __name__ == '__main__':
    bd_model = keras.models.load_model(modelName)

    clean_x_test, clean_y_test = data_load(clean_validation_data)
    poison_x_test, poison_y_test = data_load(poisoned_validation_data)


    clean_predict = np.argmax(bd_model.predict(clean_x_test), axis=1)
    clean_accuracy = np.mean(np.equal(clean_predict, clean_y_test))*100
    print('Clean Date Classification Accuracy is :', clean_accuracy)

    poison_predict = np.argmax(bd_model.predict(poison_x_test), axis=1)
    asr = np.mean(np.equal(poison_predict, poison_y_test))*100
    print('Attack Success Rate is :', asr)
```

```
Clean Date Classification Accuracy is : 98.64899974019225
Attack Success Rate is : 100.0
```

In [32]:

```
print(clean_x_test[0][0][0])
```

```
[61. 50. 44.]
```

Prune defense

In [33]:

```
clean_accuracy  # original accuracy, get it from the begining
```

Out[33]:

```
98.64899974019225
```

In [34]:

```
temp_model = keras.models.clone_model(model)
temp_model.set_weights(model.get_weights())
prune_index = [0] * 60
clean_acc = [0] * 60
asr_rates = [0] * 60
models =[0,0,0]
```

In [35]:

```
print(models)
```

```
[0, 0, 0]
```

In [36]:

```
warnings.filterwarnings("ignore")
```

In [37]:

```
# get  activation from pool_3
pool_3_output=temp_model.get_layer('pool_3').output
intermediate_model=keras.models.Model(inputs=temp_model.input,outputs=pool_3_output)
intermediate_prediction=intermediate_model.predict(clean_x_test)
res = np.mean(intermediate_prediction,axis=(0,1,2))
seq = np.argsort(res)
```

In [38]:

```
print(res)
```

```
[0.0000000e+00 8.5787815e-01 0.0000000e+00 5.3079766e-01 5.1451392e+00
 2.0289590e+00 6.2408652e-03 5.3690357e+00 2.1106057e+00 0.0000000e+00
 4.1488924e+00 2.1980374e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 1.5654891e+00 0.0000000e+00 5.0868411e+00 2.4381575e-01
 1.8378228e-01 8.3539173e-02 4.3979675e-02 3.0290759e-03 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 4.8440862e+00 1.0589778e+00
 0.0000000e+00 0.0000000e+00 1.5006667e-02 0.0000000e+00 0.0000000e+00
 4.8648086e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 0.0000000e+00 5.7658583e-01 4.2763177e-01 0.0000000e+00
 0.0000000e+00 1.8540379e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
 0.0000000e+00 1.3321567e-02 6.2038708e+00 0.0000000e+00 3.6191154e+00
 0.0000000e+00 1.6352931e+00 8.2229824e+00 5.0732863e-01 0.0000000e+00]
```

In [39]:

```
print(seq)
```

```
[ 0 26 27 30 31 33 34 36 37 38 25 39 41 44 45 47 48 49 50 53 55 40 24 59
  9  2 12 13 17 14 15 23  6 51 32 22 21 20 19 43 58  3 42  1 29 16 56 46
  5  8 11 54 10 28 35 18  4  7 52 57]
```

In [40]:

```
weight01 = temp_model.layers[5].get_weights()[0]
bias01 = temp_model.layers[5].get_weights()[1]
```

In [41]:

```
print(weight01)
```

```
[[[[ 7.62559753e-03 -3.68404202e-02 -4.71983105e-02 ...  9.82918069e-02
     7.46472403e-02  3.26913968e-02]
   [-1.65268257e-02 -3.71079892e-02  2.75818575e-02 ... -3.52755040e-02
    -3.04741785e-02 -1.87791381e-02]
   [ 7.64945894e-03 -3.60482603e-01  1.84942987e-02 ... -1.03047878e-01
    -1.75690532e-01 -1.17439650e-01]
   ...
   [-6.11336948e-03 -4.35380377e-02 -2.25859624e-03 ...  2.94640921e-02
     7.40277916e-02 -9.08699557e-02]
   [-2.14333478e-02  6.40141079e-03  5.64159080e-03 ...  7.65938163e-02
    -1.45234494e-03 -1.85639057e-02]
   [ 1.80702758e-04 -2.38118559e-01 -5.97669184e-03 ...  2.60631949e-01
    -1.79552302e-01  9.17865112e-02]]

  [[-9.35920328e-03 -2.96055470e-02 -4.06530499e-02 ...  4.67284732e-02
    -1.11967251e-02 -1.78794451e-02]
   [-4.66791354e-03 -3.49188745e-02 -4.25367691e-02 ... -5.64412866e-03
    -1.78415515e-02  4.47795121e-03]
   [-7.10388692e-03 -4.52145696e-01  3.88977975e-02 ... -8.04904103e-02
     6.01278581e-02 -7.57048801e-02]
   ...
   [ 1.73634775e-02 -2.13751197e-03 -1.68338567e-02 ... -8.90995041e-02
    -3.04076970e-02 -6.72771707e-02]
   [-1.92595087e-02 -6.52428775e-04  1.47904474e-02 ...  5.27229719e-02
    -3.50014563e-03 -3.05251172e-03]
   [-9.02825315e-03 -9.11225155e-02 -4.10709754e-02 ... -1.24276973e-01
    -1.14086606e-01 -4.09073718e-02]]

  [[-2.08336357e-02  3.18581276e-02 -6.26099855e-02 ... -1.04352571e-02
     2.95839254e-02  5.70460483e-02]
   [-1.04390867e-02  1.49184116e-03 -2.03672238e-02 ... -4.64526080e-02
    -1.01533830e-02  1.67442095e-02]
   [ 8.55093356e-03 -4.66640830e-01 -1.69603247e-03 ... -5.45055717e-02
     4.50788178e-02 -1.46124959e-01]
   ...
   [-1.60355214e-02 -7.31097162e-02 -4.57787188e-03 ...  4.30466682e-02
    -1.33530140e-01 -2.54486967e-02]
   [-1.76216550e-02  7.74278631e-03 -3.35166976e-02 ... -5.98099828e-03
    -4.89102444e-03  6.00110914e-04]
   [ 9.22490284e-03  1.80637360e-01 -1.07755221e-01 ... -1.70106009e-01
    -1.93025976e-01 -6.75048679e-02]]]


 [[[-1.01656897e-03 -7.11699575e-03 -4.26849239e-02 ... -6.16201349e-02
    -5.95680736e-02  9.93160605e-02]
   [-1.71678681e-02  9.75632400e-04  1.23781003e-02 ...  3.13481167e-02
     4.40175831e-02  3.60739194e-02]
   [ 1.43734915e-02 -4.93392460e-02  2.60744710e-02 ...  1.34386914e-03
    -4.39550728e-02 -4.96904254e-02]
   ...
   [ 5.95660741e-03 -4.58140150e-02 -3.99116501e-02 ...  3.26735079e-02
    -3.83364633e-02 -6.62529394e-02]
   [-3.14377323e-02 -1.01549001e-02 -1.84226893e-02 ...  1.06081538e-01
     8.58443044e-03  2.99895462e-03]
   [-6.80291653e-03  1.32156923e-01 -3.49593423e-02 ...  2.38788277e-02
    -9.00338367e-02  7.31495991e-02]]

  [[ 1.38541460e-02 -2.14290414e-02 -2.79791057e-02 ... -4.07886282e-02
    -6.67729825e-02  1.03126220e-01]
   [ 8.77903251e-04  3.78983095e-02  2.17933394e-02 ... -1.84473898e-02
    -3.38987000e-02  3.26485522e-02]
```

```
       [ 6.90411311e-03 -3.21772963e-01  1.93700497e-03 ...  1.88827246e-01
         1.93833008e-01 -1.40742734e-01]
       ...
       [-3.64642101e-03 -2.15889856e-01  6.18841080e-03 ... -2.35550888e-02
         1.75482575e-02  2.34112311e-02]
       [-1.21241622e-02  2.73878407e-02  1.00498470e-02 ...  4.52770442e-02
         2.88443826e-03  2.54335138e-03]
       [-8.97649303e-03 -8.52410719e-02 -2.13488122e-03 ...  3.55662741e-02
        -8.23865924e-03  6.17419332e-02]]

      [[-1.19305123e-02 -7.92782009e-02 -2.87245736e-02 ... -5.96597558e-03
         6.65202923e-03  1.09568864e-01]
       [ 3.53931426e-03  4.48940098e-02  2.06772182e-02 ... -1.47806900e-02
         4.09217142e-02  3.74381430e-02]
       [-3.43905180e-03 -1.34037837e-01 -3.90694328e-02 ...  4.92113270e-02
         2.50721991e-01 -1.02391236e-01]
       ...
       [-3.43104638e-03  6.36535883e-02  3.18213329e-02 ...  4.09666933e-02
        -7.96925351e-02 -2.25118529e-02]
       [-2.14784257e-02  1.02878865e-02 -3.36464890e-03 ... -1.47368126e-02
        -8.42981972e-04  1.24435881e-02]
       [ 5.12373110e-04  5.81633002e-02 -1.27388507e-01 ...  3.07366252e-02
        -1.12634279e-01 -2.29526870e-02]]]


    [[[-7.60890590e-03 -1.16570219e-01 -4.15994674e-02 ... -1.65507391e-01
       -2.05650821e-01  5.91975637e-02]
      [ 4.59528994e-03  2.67499336e-03 -3.20339575e-02 ... -2.69118026e-02
       -2.87360419e-02  3.65566052e-02]
      [ 3.04805115e-03 -2.13815346e-01  2.80980710e-02 ...  2.49219202e-02
         1.81046538e-02  5.34638762e-02]
      ...
      [-7.19634956e-03 -7.79411495e-02 -5.08415140e-02 ... -3.76618989e-02
         6.62643164e-02  2.99169365e-02]
      [ 1.04545348e-03 -2.25813258e-02 -1.34984842e-02 ...  6.16673492e-02
        -6.98504178e-03 -1.71177983e-02]
      [-1.83508988e-03 -3.90282840e-01  8.39547906e-03 ... -9.97870192e-02
        -4.08367766e-03  9.84790325e-02]]

     [[-1.14568546e-02 -3.00941896e-02 -8.32678471e-03 ... -1.44381985e-01
       -8.41605738e-02  5.66302575e-02]
      [ 6.52369717e-03  2.09826585e-02  5.32375881e-03 ...  2.80707907e-02
       -2.40302775e-02 -3.71785723e-02]
      [ 1.07278982e-02 -4.74469393e-01  4.20640735e-03 ...  3.01050879e-02
         1.07457656e-02 -1.11609094e-01]
      ...
      [-1.19471727e-02 -1.21928444e-02  2.44828966e-02 ... -1.45154390e-02
         9.20219813e-03 -3.39598581e-02]
      [-1.76136792e-02  1.19567215e-02  6.37314923e-04 ...  1.83562227e-02
         8.04304797e-03  5.57167968e-03]
      [-1.32015236e-02 -4.22072768e-01  1.24622183e-02 ...  6.25287220e-02
        -1.56711712e-02 -4.45652716e-02]]

     [[-1.19478479e-02 -1.70988008e-01 -1.53128663e-02 ...  3.88071090e-02
       -1.29359681e-02  9.91505012e-02]
      [ 3.26195336e-03  2.39628348e-02  1.40318274e-02 ...  1.89766719e-03
       -1.40088256e-02 -3.21108545e-03]
      [ 9.10226721e-03 -1.87653482e-01 -4.97735813e-02 ... -7.07690194e-02
         7.47106448e-02 -1.06910996e-01]
      ...
      [-4.12981259e-03  5.43628633e-03  5.38416766e-02 ...  2.09199004e-02
```

```
 -4.36137691e-02 -4.04315330e-02]
[-8.83697998e-03  1.66951343e-02  4.63473890e-03 ... -1.82218552e-02
  8.15996248e-03  1.96164176e-02]
[ 2.34161876e-03  3.84562984e-02 -1.48163691e-01 ... -8.54821280e-02
 -2.75227904e-01 -1.55074988e-02]]]]
```

In [42]:

```
print(len(clean_acc))
```

60

In [25]:

```
for i in range(len(seq)):
    #prune
    channel_index = seq[i]
    weight01[:, :, :, channel_index] = 0
    bias01[channel_index] = 0
    temp_model.layers[5].set_weights([weight01, bias01])

    #predict for clean data
    clean_label_predict = np.argmax(temp_model.predict(clean_x_test), axis=1)
    new_clean_accuracy = np.mean(np.equal(clean_label_predict, clean_y_test)) * 100
    clean_acc[i] = new_clean_accuracy

    #predict for poison data
    poison_predict = np.argmax(temp_model.predict(poison_x_test), axis=1)
    asr = np.mean(np.equal(poison_predict, poison_y_test)) * 100
    asr_rates[i] = asr

    #print
    print("now the new clean accuracy is: ", new_clean_accuracy)
    print("now the new attack success rate is: ", asr)
    print("The pruned channel index is: ", channel_index)


    #save model when meet requires
    if models[0] == 0 and clean_accuracy - new_clean_accuracy >= 2:
        temp_model.save('model_X=2.h5')
        models[0] = 1
        print("!Attention, The validation accuracy drops atleast 2% below the original accuracy,
 so stoped and saved the model")

    if models[1] == 0 and clean_accuracy - new_clean_accuracy >= 4:
        temp_model.save('model_X=4.h5')
        models[1] = 1
        print("!Attention, The validation accuracy drops atleast 4% below the original accuracy,
 so stoped and saved the model")

    if models[2] == 0 and clean_accuracy - new_clean_accuracy >= 10:
        temp_model.save('model_X=10.h5')
        models[2] = 1
        print("!Attention, The validation accuracy drops atleast 10% below the original accurac
y, so stoped and saved the model")

    print("#########################################################")
```

```
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  0
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  26
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  27
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  30
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  31
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  33
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  34
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  36
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  37
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  38
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  25
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  39
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  41
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  44
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  45
########################################################
now the new clean accuracy is:  98.64899974019225
```

```
now the new attack success rate is:  100.0
The pruned channel index is:  47
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  48
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  49
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  50
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  53
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  55
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  40
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  24
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  59
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  9
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  2
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  12
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  13
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  17
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  14
############################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
```

```
The pruned channel index is:  15
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  23
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  6
########################################################
now the new clean accuracy is:  98.64033948211657
now the new attack success rate is:  100.0
The pruned channel index is:  51
########################################################
now the new clean accuracy is:  98.64033948211657
now the new attack success rate is:  100.0
The pruned channel index is:  32
########################################################
now the new clean accuracy is:  98.63167922404088
now the new attack success rate is:  100.0
The pruned channel index is:  22
########################################################
now the new clean accuracy is:  98.65765999826795
now the new attack success rate is:  100.0
The pruned channel index is:  21
########################################################
now the new clean accuracy is:  98.64899974019225
now the new attack success rate is:  100.0
The pruned channel index is:  20
########################################################
now the new clean accuracy is:  98.6056984498138
now the new attack success rate is:  100.0
The pruned channel index is:  19
########################################################
now the new clean accuracy is:  98.57105741751104
now the new attack success rate is:  100.0
The pruned channel index is:  43
########################################################
now the new clean accuracy is:  98.53641638520828
now the new attack success rate is:  100.0
The pruned channel index is:  58
########################################################
now the new clean accuracy is:  98.19000606218066
now the new attack success rate is:  100.0
The pruned channel index is:  3
########################################################
now the new clean accuracy is:  97.65307006148784
now the new attack success rate is:  100.0
The pruned channel index is:  42
########################################################
now the new clean accuracy is:  97.50584567420108
now the new attack success rate is:  100.0
The pruned channel index is:  1
########################################################
now the new clean accuracy is:  95.75647354291158
now the new attack success rate is:  100.0
The pruned channel index is:  29
!Attention, The validation accuracy drops atleast 2% below the original accuracy,
so stoped and saved the model
########################################################
now the new clean accuracy is:  95.20221702606739
```

```
now the new attack success rate is:  99.9913397419243
The pruned channel index is:  16
#########################################################
now the new clean accuracy is:  94.7172425738287
now the new attack success rate is:  99.9913397419243
The pruned channel index is:  56
#########################################################
now the new clean accuracy is:  92.09318437689443
now the new attack success rate is:  99.9913397419243
The pruned channel index is:  46
!Attention, The validation accuracy drops atleast 4% below the original accuracy,
so stoped and saved the model
#########################################################
now the new clean accuracy is:  91.49562656967177
now the new attack success rate is:  99.9913397419243
The pruned channel index is:  5
#########################################################
now the new clean accuracy is:  91.01931237550879
now the new attack success rate is:  99.98267948384861
The pruned channel index is:  8
#########################################################
now the new clean accuracy is:  89.17467740538669
now the new attack success rate is:  80.73958603966398
The pruned channel index is:  11
#########################################################
now the new clean accuracy is:  84.43751623798389
now the new attack success rate is:  77.015675067117
The pruned channel index is:  54
!Attention, The validation accuracy drops atleast 10% below the original accuracy,
so stoped and saved the model
#########################################################
now the new clean accuracy is:  76.48739932449988
now the new attack success rate is:  35.71490430414826
The pruned channel index is:  10
#########################################################
now the new clean accuracy is:  54.8627349095003
now the new attack success rate is:  6.954187234779596
The pruned channel index is:  28
#########################################################
now the new clean accuracy is:  27.08928726076037
now the new attack success rate is:  0.4243526457088421
The pruned channel index is:  35
#########################################################
now the new clean accuracy is:  13.87373343725643
now the new attack success rate is:  0.0
The pruned channel index is:  18
#########################################################
now the new clean accuracy is:  7.101411622066338
now the new attack success rate is:  0.0
The pruned channel index is:  4
#########################################################
now the new clean accuracy is:  1.5501861955486274
now the new attack success rate is:  0.0
The pruned channel index is:  7
#########################################################
now the new clean accuracy is:  0.7188014202823244
now the new attack success rate is:  0.0
The pruned channel index is:  52
#########################################################
now the new clean accuracy is:  0.0779423226812159
now the new attack success rate is:  0.0
```

```
The pruned channel index is:  57
########################################################
```

In [28]:

```
print("clean_accuracy: ",clean_acc)
print("attack success rate: ",asr_rates)
```

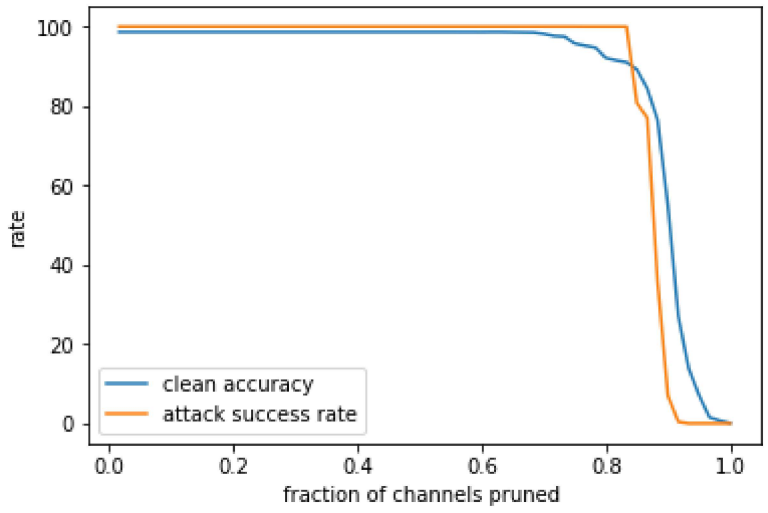
```
clean_accuracy:  [98.64899974019225, 98.64899974019225, 98.64899974019225, 98.6489
9974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.6489997401
9225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225,
98.64899974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.648
99974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.648999740
19225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225,
98.64899974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.648
99974019225, 98.64899974019225, 98.64899974019225, 98.64899974019225, 98.640339482
11657, 98.64033948211657, 98.63167922404088, 98.65765999826795, 98.64899974019225,
98.6056984498138, 98.57105741751104, 98.53641638520828, 98.19000606218066, 97.6530
7006148784, 97.50584567420108, 95.75647354291158, 95.20221702606739, 94.7172425738
287, 92.09318437689443, 91.49562656967177, 91.01931237550879, 89.17467740538669, 8
4.43751623798389, 76.48739932449988, 54.8627349095003, 27.08928726076037, 13.87373
343725643, 7.101411622066338, 1.5501861955486274, 0.7188014202823244, 0.0779423226
812159]
attack success rate:  [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 10
0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0,
100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.
0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 10
0.0, 100.0, 99.9913397419243, 99.9913397419243, 99.9913397419243, 99.991339741924
3, 99.98267948384861, 80.73958603966398, 77.015675067117, 35.71490430414826, 6.954
187234779596, 0.4243526457088421, 0.0, 0.0, 0.0, 0.0, 0.0]
```

In [29]:

```
x_axis = np.arange(1,61)/60
plt.plot(x_axis,clean_acc)
plt.plot(x_axis,asr_rates)
plt.legend(['clean accuracy','attack success rate'])
plt.xlabel("fraction of channels pruned ")
plt.ylabel("rate")
```

Out[29]:

```
Text(0, 0.5, 'rate')
```



G - repaired network

In [52]:

```
#new G
class G(keras.Model):
    def __init__(self, B, B_new):
        super(G, self).__init__()
        self.B = B
        self.B_new = B_new

    def predict(self,data):
        predict_B = np.argmax(self.B.predict(data), axis=1)
        predict_B_new = np.argmax(self.B_new.predict(data), axis=1)

        predicts = np.zeros(data.shape[0])

        for i in range(data.shape[0]):
            if predict_B[i]==predict_B_new[i]:
                predicts[i] = predict_B[i]

            else:
                predicts[i] = 1283

        return predicts
```

use data to evaluate the B' model

In [48]:

```
#test data
clean_test_data = filePath + 'test.h5'
poisoned_test_data = filePath + 'bd_test.h5'

x_test_data, y_test_data = data_load(clean_test_data)
x_poisoned_test_data, y_poisoned_test_data = data_load(poisoned_test_data)
```

In [51]:

```
#model
model_X_2_Name  =  filePath + 'model_X=2.h5'
model_X_4_Name   = filePath +  'model_X=4.h5'
model_X_10_Name   = filePath +  'model_X=10.h5'

model_X_2 = keras.models.load_model(model_X_2_Name)
model_X_4 = keras.models.load_model(model_X_4_Name)
model_X_10 = keras.models.load_model(model_X_10_Name)
```

In [17]:

```
print("x_test_data shape: ",x_test_data.shape)
print("x_test_poisoned data shape: ",x_poisoned_test_data.shape)
```

```
x_test_data shape:  (12830, 55, 47, 3)
x_test_poisoned data shape:  (12830, 55, 47, 3)
```

In [35]:

```
#clean data in B' model
clean_test_2_label_predict = np.argmax(model_X_2.predict(clean_x_test), axis=1)
clean_test_2_accuracy = np.mean(np.equal(clean_test_2_label_predict, clean_y_test))*100
poisoned_test_2_label_predict = np.argmax(model_X_2.predict(poison_x_test), axis=1)
asr_2 = np.mean(np.equal(poisoned_test_2_label_predict, poison_y_test))*100

print('2% drops model, the clean test data accuracy is :', clean_test_2_accuracy)
print('2% drops model, Attack Success Rate is:', asr_2)


clean_test_10_label_predict = np.argmax(model_X_10.predict(clean_x_test), axis=1)
clean_test_10_accuracy = np.mean(np.equal(clean_test_10_label_predict, clean_y_test))*100
poisoned_test_10_label_predict = np.argmax(model_X_10.predict(poison_x_test), axis=1)
asr_10 = np.mean(np.equal(poisoned_test_10_label_predict, poison_y_test))*100

print('10% drops model, the clean test data accuracy is :', clean_test_10_accuracy)
print('10% drops model, Attack Success Rate is:', asr_10)
```

```
2% drops model, the clean test data accuracy is : 95.75647354291158
2% drops model, Attack Success Rate is: 100.0
10% drops model, the clean test data accuracy is : 84.43751623798389
10% drops model, Attack Success Rate is: 77.015675067117
```

In [62]:

```
#test data in B' model
clean_test_2_label_predict = np.argmax(model_X_2.predict(x_test_data), axis=1)
clean_test_2_accuracy = np.mean(np.equal(clean_test_2_label_predict, y_test_data))*100
poisoned_test_2_label_predict = np.argmax(model_X_2.predict(x_poisoned_test_data), axis=1)
asr_2 = np.mean(np.equal(poisoned_test_2_label_predict, y_poisoned_test_data))*100

print('2% drops model, the clean test data accuracy is :', clean_test_2_accuracy)
print('2% drops model, Attack Success Rate is:', asr_2)


clean_test_10_label_predict = np.argmax(model_X_10.predict(x_test_data), axis=1)
clean_test_10_accuracy = np.mean(np.equal(clean_test_10_label_predict, y_test_data))*100
poisoned_test_10_label_predict = np.argmax(model_X_10.predict(x_poisoned_test_data), axis=1)
asr_10 = np.mean(np.equal(poisoned_test_10_label_predict, y_poisoned_test_data))*100

print('10% drops model, the clean test data accuracy is :', clean_test_10_accuracy)
print('10% drops model, Attack Success Rate is:', asr_10)
```

```
2% drops model, the clean test data accuracy is : 95.90023382696803
2% drops model, Attack Success Rate is: 100.0
10% drops model, the clean test data accuracy is : 84.54403741231489
10% drops model, Attack Success Rate is: 77.20966484801247
```

use test data to evaluate the repaired network -G

In [53]:

```
G_model_X_2_temp = G(model, model_X_2)
G_model_X_4_temp = G(model, model_X_4)
G_model_X_10_temp = G(model, model_X_10)
```

In [54]:

```
G_model_X_2_temp._is_graph_network = True
```

combined model

In [55]:

```
G_model_X_2 = G_model_X_2_temp
G_model_X_4 = G_model_X_4_temp
G_model_X_10 = G_model_X_10_temp
```

test data in G-model

In [60]:

```
cl_label_p_2 = G_model_X_2.predict(x_test_data)
clean_accuracy_2 = np.mean(np.equal(cl_label_p_2, y_test_data))*100
print('Clean Classification accuracy-G_model_X_2 :', clean_accuracy_2)

bd_label_p_2 = G_model_X_2.predict(x_poisoned_test_data)
asr_2 = np.mean(np.equal(bd_label_p_2, y_poisoned_test_data))*100
print('Attack Success Rate-G_model_X_2:', asr_2)

cl_label_p_4 =G_model_X_4.predict(x_test_data)
clean_accuracy_4 = np.mean(np.equal(cl_label_p_4, y_test_data)) * 100
print('Clean Classification accuracy-G_model_X_4 :', clean_accuracy_4)

bd_label_p_4 = G_model_X_4.predict(x_poisoned_test_data)
asr_4 = np.mean(np.equal(bd_label_p_4, y_poisoned_test_data)) * 100
print('Attack Success Rate-G_model_X_4:', asr_4)


cl_label_p_10 = G_model_X_10.predict(x_test_data)
clean_accuracy_10 = np.mean(np.equal(cl_label_p_10, y_test_data)) * 100
print('Clean Classification accuracy-G_model_X_10 :', clean_accuracy_10)

bd_label_p_10 =G_model_X_10.predict(x_poisoned_test_data)
asr_10 = np.mean(np.equal(bd_label_p_10, y_poisoned_test_data)) * 100
print('Attack Success Rate-G_model_X_10:', asr_10)
```

```
Clean Classification accuracy-G_model_X_2 : 95.74434918160561
Attack Success Rate-G_model_X_2: 100.0
Clean Classification accuracy-G_model_X_4 : 92.1278254091972
Attack Success Rate-G_model_X_4: 99.98441153546376
Clean Classification accuracy-G_model_X_10 : 84.3335931410756
Attack Success Rate-G_model_X_10: 77.20966484801247
```

Plot the accuracy on clean test data and the attack success rate (on backdoored test data) as a function of the fraction of channels pruned.

In [56]:

```
clean_acc01 = [0] * 60
asrate01= [0] * 60
clean_acc02 = [0] * 60
asrate02= [0] * 60
clean_acc03 = [0] * 60
asrate03= [0] * 60
```

In [57]:

```
pool_3_output=G_model_X_2.B_new.get_layer('pool_3').output
intermediate_model=keras.models.Model(inputs=G_model_X_2.B_new.input,outputs=pool_3_output)
intermediate_prediction=intermediate_model.predict(x_test_data)
res = np.mean(intermediate_prediction,axis=(0,1,2))
seq01 = np.argsort(res)
weight_2 = G_model_X_2.B_new.layers[5].get_weights()[0]
bias_2 = G_model_X_2.B_new.layers[5].get_weights()[1]
```

In [58]:

```
pool_3_output=G_model_X_4.B_new.get_layer('pool_3').output
intermediate_model=keras.models.Model(inputs=G_model_X_4.B_new.input,outputs=pool_3_output)
intermediate_prediction=intermediate_model.predict(x_test_data)
res = np.mean(intermediate_prediction,axis=(0,1,2))
seq02 = np.argsort(res)
weight_4 = G_model_X_4.B_new.layers[5].get_weights()[0]
bias_4 = G_model_X_4.B_new.layers[5].get_weights()[1]
```

In [59]:

```
pool_3_output=G_model_X_10.B_new.get_layer('pool_3').output
intermediate_model=keras.models.Model(inputs=G_model_X_10.B_new.input,outputs=pool_3_output)
intermediate_prediction=intermediate_model.predict(x_test_data)
res = np.mean(intermediate_prediction,axis=(0,1,2))
seq03 = np.argsort(res)
weight_10 = G_model_X_10.B_new.layers[5].get_weights()[0]
bias_10 = G_model_X_10.B_new.layers[5].get_weights()[1]
```

In [ ]:

```
for i in range(60):
    channel_index = seq[i]

    weight_4[:, :, :, channel_index] = 0
    bias_4[channel_index] = 0
    G_model_X_4.B_new.layers[5].set_weights([weight_4, bias_4])

    weight_10[:, :, :, channel_index] = 0
    bias_10[channel_index] = 0
    G_model_X_10.B_new.layers[5].set_weights([weight_10, bias_10])


    cl_label_p_4 = G_model_X_4.predict(x_test_data)
    clean_accuracy_4 = np.mean(np.equal(cl_label_p_4, y_test_data)) * 100
    print('Clean Classification accuracy-G_model_X_4 :', clean_accuracy_4)

    bd_label_p_4 =G_model_X_4.predict(x_poisoned_test_data)
    asr_4 = np.mean(np.equal(bd_label_p_4, y_poisoned_test_data)) * 100
    print('Attack Success Rate-G_model_X_4:', asr_4)

    cl_label_p_10 = np.argmax(G_model_X_10.B_new.predict(x_test_data), axis=1)
    clean_accuracy_10 = np.mean(np.equal(cl_label_p_10 , y_test_data)) * 100
    print('Clean Classification accuracy-G_model_X_10 :', clean_accuracy_10)

    bd_label_p_10 = np.argmax(G_model_X_10.B_new.predict(x_poisoned_test_data), axis=1)
    asr_10 = np.mean(np.equal(bd_label_p_10, y_poisoned_test_data)) * 100
    print('Attack Success Rate-G_model_X_10: ', asr_10)
    clean_acc02[i]  = clean_accuracy_4
    asrate02[i] = asr_4
    clean_acc03[i]  = clean_accuracy_10
    asrate03[i] = asr_10

    print("#########################################################")
```

In [71]:

```
for i in range(60):
    channel_index = seq[i]

    weight_2[:, :, :, channel_index] = 0
    bias_2[channel_index] = 0

    G_model_X_2.B_new.layers[5].set_weights([weight_2, bias_2])

    cl_label_p_2 =  np.argmax(G_model_X_2.B_new.predict(x_test_data), axis=1)
    clean_accuracy_2 = np.mean(np.equal(cl_label_p_2, y_test_data))*100
    print('Clean Classification accuracy-G_model_X_2 :', clean_accuracy_2)

    bd_label_p_2 = np.argmax(G_model_X_2.B_new.predict(x_poisoned_test_data), axis=1)
    asr_2 = np.mean(np.equal(bd_label_p_2, y_poisoned_test_data))*100
    print('Attack Success Rate-G_model_X_2:', asr_2)

    clean_acc01[i] = clean_accuracy_2
    asrate01[i]= asr_2

    print("#########################################################")
```

```
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
```

```
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
######################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
```

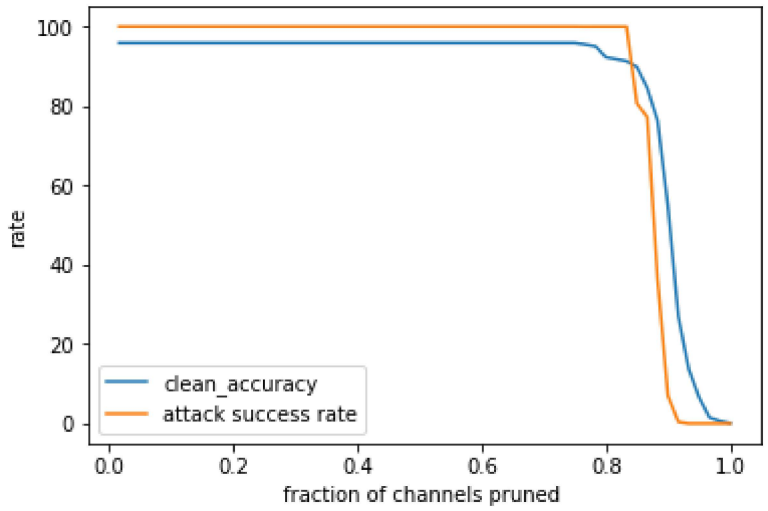
```
############################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
############################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
############################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
############################################################
Clean Classification accuracy-G_model_X_2 : 95.90023382696803
Attack Success Rate-G_model_X_2: 100.0
############################################################
Clean Classification accuracy-G_model_X_2 : 95.5261106780982
Attack Success Rate-G_model_X_2: 99.97661730319564
############################################################
Clean Classification accuracy-G_model_X_2 : 95.0584567420109
Attack Success Rate-G_model_X_2: 99.98441153546376
############################################################
Clean Classification accuracy-G_model_X_2 : 92.29150428682775
Attack Success Rate-G_model_X_2: 99.98441153546376
############################################################
Clean Classification accuracy-G_model_X_2 : 91.8082618862042
Attack Success Rate-G_model_X_2: 99.98441153546376
############################################################
Clean Classification accuracy-G_model_X_2 : 91.30943102104443
Attack Success Rate-G_model_X_2: 99.97661730319564
############################################################
Clean Classification accuracy-G_model_X_2 : 89.84411535463757
Attack Success Rate-G_model_X_2: 80.6469212782541
############################################################
Clean Classification accuracy-G_model_X_2 : 84.54403741231489
Attack Success Rate-G_model_X_2: 77.20966484801247
############################################################
Clean Classification accuracy-G_model_X_2 : 76.30553390491036
Attack Success Rate-G_model_X_2: 36.26656274356976
############################################################
Clean Classification accuracy-G_model_X_2 : 54.762275915822286
Attack Success Rate-G_model_X_2: 6.96024941543258
############################################################
Clean Classification accuracy-G_model_X_2 : 27.10054559625877
Attack Success Rate-G_model_X_2: 0.4208885424785659
############################################################
Clean Classification accuracy-G_model_X_2 : 13.756819953234606
Attack Success Rate-G_model_X_2: 0.0
############################################################
Clean Classification accuracy-G_model_X_2 : 6.570537802026501
Attack Success Rate-G_model_X_2: 0.0
############################################################
Clean Classification accuracy-G_model_X_2 : 1.5198752922837102
Attack Success Rate-G_model_X_2: 0.0
############################################################
Clean Classification accuracy-G_model_X_2 : 0.646921278254092
Attack Success Rate-G_model_X_2: 0.0
############################################################
Clean Classification accuracy-G_model_X_2 : 0.0779423226812159
Attack Success Rate-G_model_X_2: 0.0
############################################################
```

In [72]:

```
#plot for G_2
x_axis = np.arange(1,61)/60
plt.plot(x_axis,clean_acc01)
plt.plot(x_axis,asrate01)
plt.legend(['clean_accuracy','attack success rate'])
plt.xlabel("fraction of channels pruned ")
plt.ylabel("rate")
```

Out[72]:

```
Text(0, 0.5, 'rate')
```

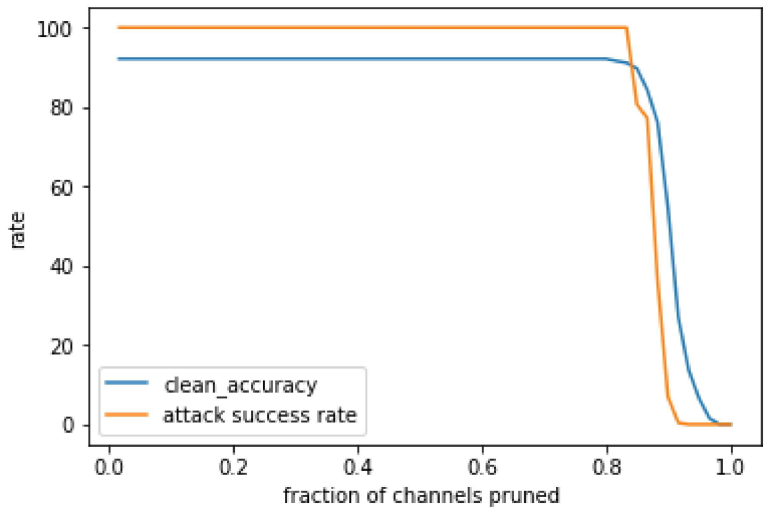


In [73]:

```
#plot for G_4
x_axis = np.arange(1,61)/60
plt.plot(x_axis,clean_acc02)
plt.plot(x_axis,asrate02)
plt.legend(['clean_accuracy','attack success rate'])
plt.xlabel("fraction of channels pruned ")
plt.ylabel("rate")
```

Out[73]:

```
Text(0, 0.5, 'rate')
```

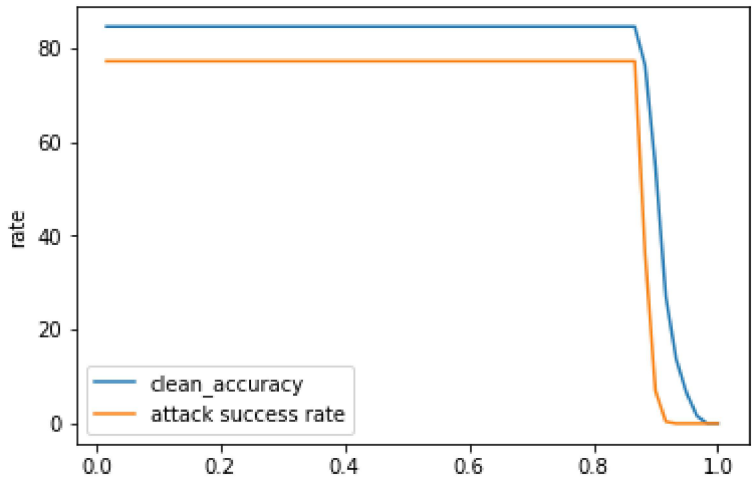


In [74]:

```
#plot for G_10
x_axis = np.arange(1,61)/60
plt.plot(x_axis,clean_acc03)
plt.plot(x_axis,asrate03)
plt.legend(['clean_acc01','asrate01'])
plt.legend(['clean_accuracy','attack success rate'])
plt.ylabel("rate")
```

Out[74]:

```
Text(0, 0.5, 'rate')
```



In [ ]: