Name: Sai Surya Salalith Mantha

USC_ID: 9463553709

Email: saisurym@usc.edu

Steps:

1) Read the data form file.
2) Split the train and validation data.
3) Preprocessing and Augmenting data.
4) Built two models.
5) Train the models.
6) Test on Validation data.
7) Test on test data.
8) Analyze the results.

I have tried two different Architecture first one is given in assignment and other is modified version where I increased the number of kernels, decreased kernel size and increased number of nodes in fully connected layers, increased the batch size.

Overall, I trained the models for 35 epochs each, but I have taken only first 10 epochs into consideration.

Results:

Model-1:

Training:

Loss=2.4429 and Validation Accuracy=34.9%

Testing:

Accuracy for 100 labels= 35.13%

Accuracy for 20 labels= 48.41%

Considering Top-3 predictions:

Accuracy for 100 classes= 35.13%

Accuracy for 20 classes= 48.41%

Considering Top-5 predictions:

Accuracy for 100 classes= 65.22%

Accuracy for 20 classes= 77.59%

Model-2:

Training:

Loss=1.2183 and Validation Accuracy=38.41%

Testing:

Accuracy for 100 labels= 39.08%

Accuracy for 20 labels= 51.69%

Considering Top-3 predictions:

Accuracy for 100 classes= 39.08%

Accuracy for 20 classes= 51.69%

Considering Top-5 predictions:

Accuracy for 100 classes= 67.66%

Accuracy for 20 classes= 80.71%

From the above results we can observe that model-2 has better results compared to model-1, we can add some more layers make the model complex and get the accuracy increased some more.

In [ ]:

```
import pickle
import numpy as np
import tensorflow as tf
import matplotlib
matplotlib.use('agg')
import pylab as plt
```

In [ ]:

```
count=1
epc=1
loss_graph_list=[]


#Reading the data
def unpickle(file):

    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

train=unpickle("cifar-100-python/train")
test=unpickle("cifar-100-python/test")

print(train.keys())
print(test.keys())

#Train data
Xtrain=train[b'data'].reshape((len(train[b'data']), 3, 32, 32)).transpose(0, 2
, 3, 1)
ytrain=np.array(train[b'fine_labels'])

#Test data
Xtest=test[b'data'].reshape((len(test[b'data']), 3, 32, 32)).transpose(0, 2, 3
, 1)
ytest=np.array(test[b'fine_labels'])

Xtrain=Xtrain.astype(np.float32)
Xtest=Xtest.astype(np.float32)

Xval=Xtrain[40000:,:,:,:]
Xtrain=Xtrain[0:40000]

yval=ytrain[40000:]
ytrain=ytrain[0:40000]

# Reset graph parameters
tf.reset_default_graph()

x = tf.placeholder(tf.float32, shape=(None, 32, 32, 3), name='input_x')
y = tf.placeholder(tf.int32, shape=(None,), name='output_y')
```

In [ ]:

```
# model Architecture
def leNet(features):
    # layer 1 input
    # input_layer=tf.reshape(features['x'],[-1,32,32,3])

    # conv_layer #1
    conv1 = tf.layers.conv2d(inputs=features, filters=6,
                             kernel_size=[5, 5], strides=1,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    # pooling_layer #1
    pool1 = tf.layers.max_pooling2d(inputs=conv1,
                                    pool_size=[2, 2],
                                    strides=2)

    batch1=tf.layers.batch_normalization(inputs=pool1)




    # conv_layer #2
    conv2 = tf.layers.conv2d(inputs=batch1, filters=16,
                             kernel_size=[5, 5], strides=1,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    # pooling_layer #2
    pool2 = tf.layers.max_pooling2d(inputs=conv2,
                                    pool_size=[2, 2],
                                    strides=2)

    batch2=tf.layers.batch_normalization(inputs=pool2)



    # Dense_layer #1
    pool2_flat = tf.reshape(batch2, [-1, 400])
    dense1 = tf.layers.dense(inputs=pool2_flat, units=120,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    batch3=tf.layers.batch_normalization(inputs=dense1)


    # Dense_layer #2
    dense2 = tf.layers.dense(inputs=batch3, units=84,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    batch4=tf.layers.batch_normalization(inputs=dense2)
    # logits_final_layer
    logits = tf.layers.dense(inputs=batch4, units=100)

    return logits
```

In [ ]:

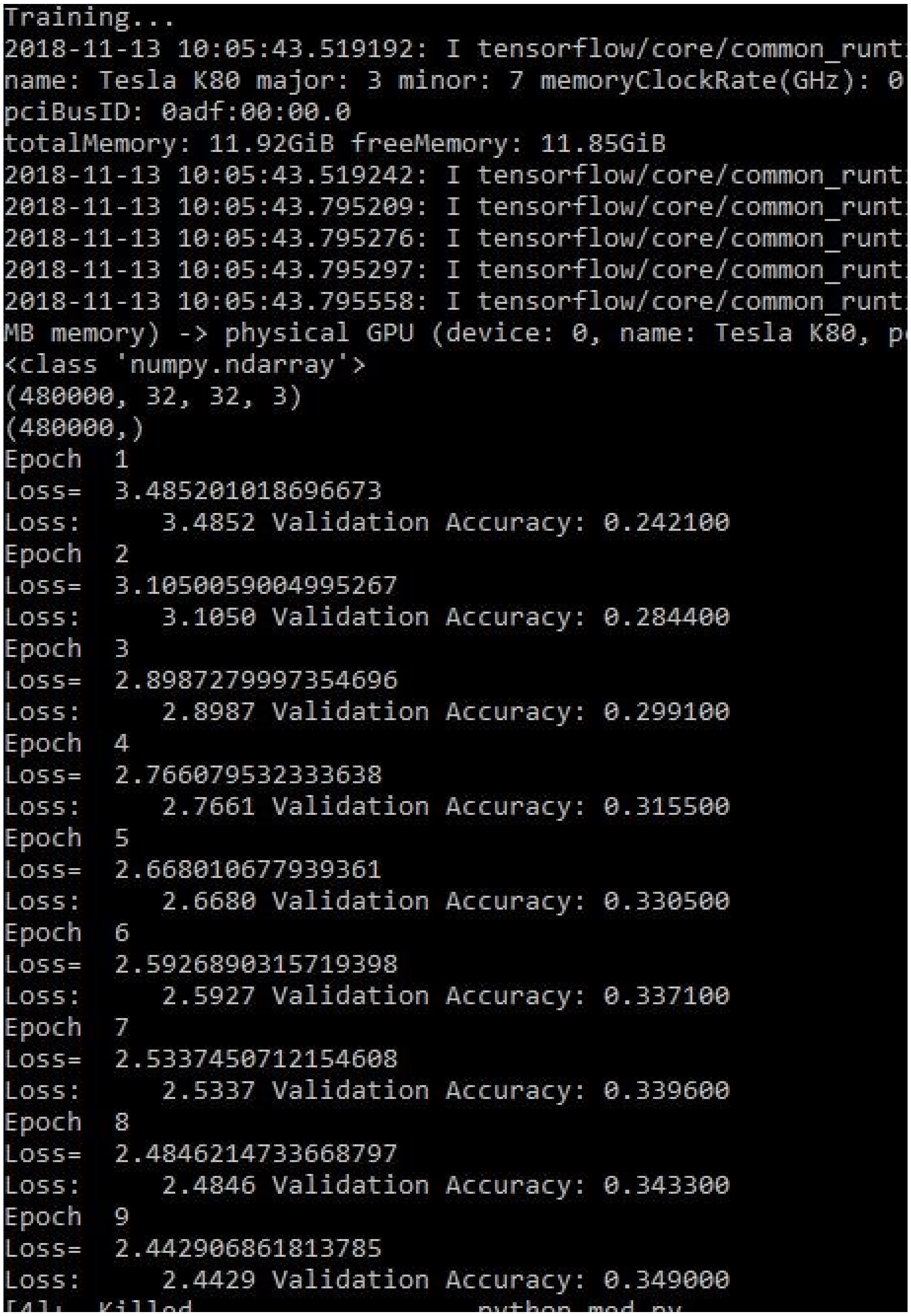
```
#epochs, batchSize,Learning rate
epochs = 2
batch_size = 64
learning_rate = 0.001

# Output of model
logits = leNet(x)
model = tf.identity(logits, name='logits')

#Loss function & Optimization Algorithm
cost = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=lo
gits, labels=y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)


#Prediction and Accuracy
correct_pred = tf.equal(tf.argmax(logits, 1),tf.cast(y,tf.int64))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32), name='accuracy')
```

In [ ]:

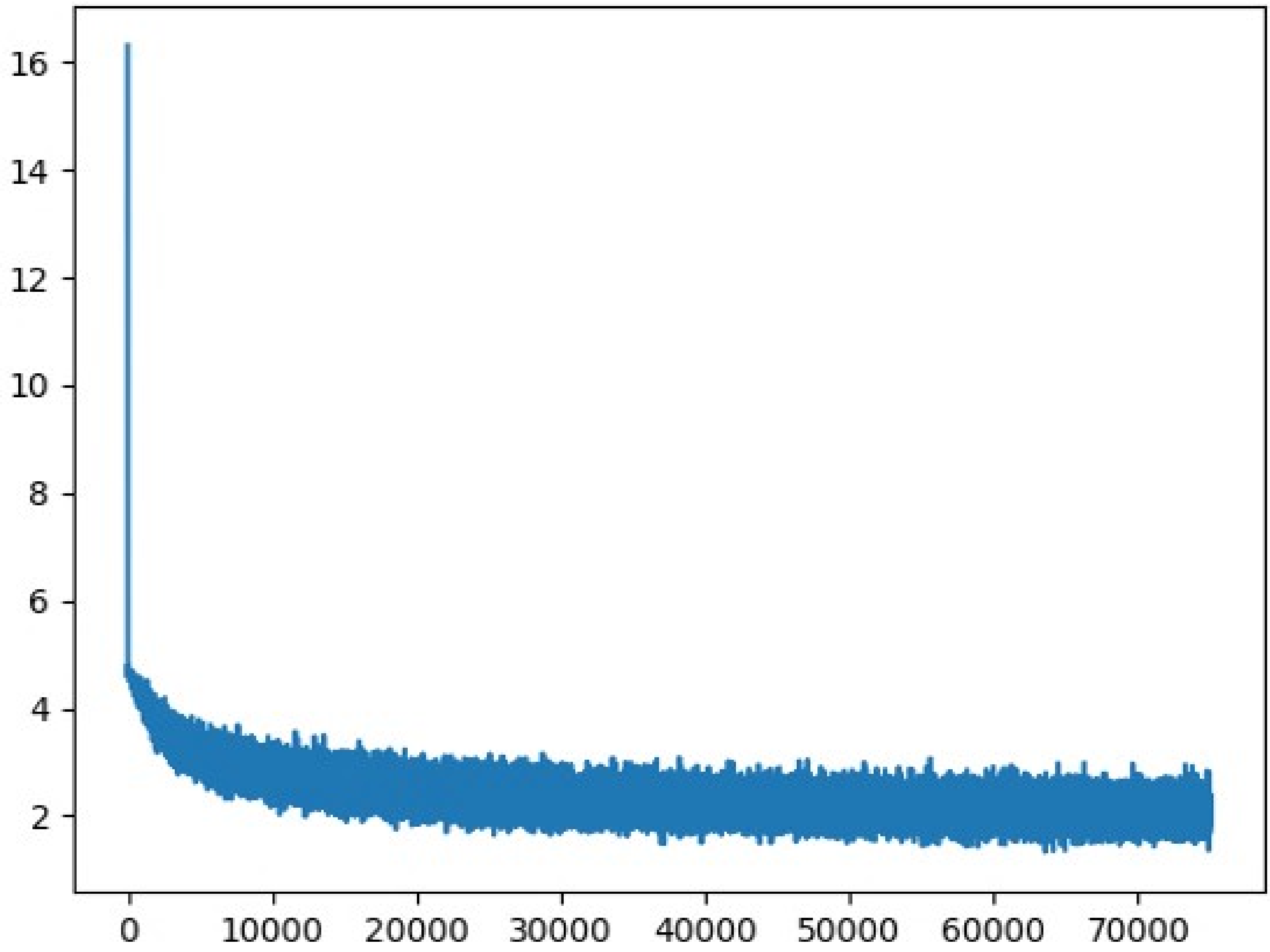
```
# Shuffling Data
def batch_features_labels(features, labels, batch_size):

    rand_index=np.random.choice(len(features),size=len(features))
    for start in range(0,len(features),batch_size):
        end=min(start+batch_size,len(features))
        tmp=np.array(rand_index[start:end])
        yield features[tmp],labels[tmp]
```

In [ ]:

```
#saving model
model_path = './image_classification'


#Creating Session
print('Training...')
with tf.Session() as sess:
    #inistalizing Global_variables
    sess.run(tf.global_variables_initializer())
    ###########

    # Data Augmentation
    data_tf=tf.convert_to_tensor(Xtrain,np.float32)
    big=tf.image.resize_images(Xtrain,(36,36))
    top_r=tf.image.crop_to_bounding_box(big,0,0,32,32)
    top_l=tf.image.crop_to_bounding_box(big,0,4,32,32)
    bot_r=tf.image.crop_to_bounding_box(big,4,0,32,32)
    bot_l=tf.image.crop_to_bounding_box(big, 4, 4, 32, 32)
    cen=tf.image.crop_to_bounding_box(big,2,2,32,32)
    flip=tf.image.flip_left_right(data_tf)

    fbig = tf.image.resize_images(flip, (36, 36))
    ftop_r = tf.image.crop_to_bounding_box(fbig, 0, 0, 32, 32)
    ftop_l = tf.image.crop_to_bounding_box(fbig, 0, 4, 32, 32)
    fbot_r = tf.image.crop_to_bounding_box(fbig, 4, 0, 32, 32)
    fbot_l = tf.image.crop_to_bounding_box(fbig, 4, 4, 32, 32)
    fcen = tf.image.crop_to_bounding_box(fbig, 2, 2, 32, 32)


    Xtrain1 = tf.concat([data_tf, top_r, top_l, bot_l, bot_r, cen, flip], axis
=0)
    Xtrain2=tf.concat([ftop_l,ftop_r,fbot_l,fbot_r,fcen],axis=0)

    # Xtrain=tf.concat([Xtrain1,Xtrain2],axis=0)

    sess.run(Xtrain1)
    sess.run(Xtrain2)
    Xtrain1=Xtrain1.eval()
    Xtrain2=Xtrain2.eval()
    Xtrain=np.concatenate((Xtrain1,Xtrain2))
    ytrain=np.concatenate((ytrain,ytrain,ytrain,ytrain,ytrain,ytrain,ytrain,yt
rain,ytrain,ytrain,ytrain,ytrain))
    # Xtrain=Xtrain.eval()
    print(type(Xtrain))
    print(Xtrain.shape)
    print(ytrain.shape)

    #Calculating mean of the dataSet
    sub = np.mean(Xtrain, axis=0)
    Xtrain = Xtrain - sub
    Xval=Xval-sub
    np.save("mean_vec",sub)

    # Traning and Validation
    for epoch in range(0,epochs):
        for batch_features, batch_labels in batch_features_labels(Xtrain, ytra
in, batch_size):
            sess.run(optimizer,
                            feed_dict={
                                x: batch_features,
                                y: batch_labels
                            })

            loss = sess.run(cost,
                            feed_dict={
                                x: batch_features,
                                y: batch_labels
                            })
            loss_graph_list.append(loss)

            if(epoch==epc):
                epc+=1
                print('Epoch {:>2}\n'.format(epoch), end='')
                print(sum(loss_graph_list)/len(loss_graph_list))

            if(epoch==count):
                count+=1
                valid_acc = sess.run(accuracy,
                                     feed_dict={
                                         x: Xval,
                                         y: yval})

                print('Loss: {:>10.4f} Validation Accuracy: {:.6f}'.format(los
s, valid_acc))


    # Saving model and plotting Loss Graph
    plt.plot(loss_graph_list)
    plt.show()
    plt.savefig("myfig")
    model_saver = tf.train.Saver()
    save_path_model = model_saver.save(sess, model_path)
```



```
Training...
2018-11-13 10:05:43.519192: I tensorflow/core/common_runt
name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0
pciBusID: 0adf:00:00.0
totalMemory: 11.92GiB freeMemory: 11.85GiB
2018-11-13 10:05:43.519242: I tensorflow/core/common_runt
2018-11-13 10:05:43.795209: I tensorflow/core/common_runt
2018-11-13 10:05:43.795276: I tensorflow/core/common_runt
2018-11-13 10:05:43.795297: I tensorflow/core/common_runt
2018-11-13 10:05:43.795558: I tensorflow/core/common_runt
MB memory) -> physical GPU (device: 0, name: Tesla K80, p
<class 'numpy.ndarray'>
(480000, 32, 32, 3)
(480000,)
Epoch  1
Loss=  3.485201018696673
Loss:      3.4852 Validation Accuracy: 0.242100
Epoch  2
Loss=  3.1050059004995267
Loss:      3.1050 Validation Accuracy: 0.284400
Epoch  3
Loss=  2.8987279997354696
Loss:      2.8987 Validation Accuracy: 0.299100
Epoch  4
Loss=  2.766079532333638
Loss:      2.7661 Validation Accuracy: 0.315500
Epoch  5
Loss=  2.668010677939361
Loss:      2.6680 Validation Accuracy: 0.330500
Epoch  6
Loss=  2.5926890315719398
Loss:      2.5927 Validation Accuracy: 0.337100
Epoch  7
Loss=  2.5337450712154608
Loss:      2.5337 Validation Accuracy: 0.339600
Epoch  8
Loss=  2.4846214733668797
Loss:      2.4846 Validation Accuracy: 0.343300
Epoch  9
Loss=  2.442906861813785
Loss:      2.4429 Validation Accuracy: 0.349000
[4]+  Killed                  python mod.py
```


16
14
12
10
8
6
4
2
0
10000
20000
30000
40000
50000
60000
70000

Name: Sai Surya Salalith Mantha

USC_ID: 9463553709

Email_ID: Saisurym@usc.edu

In [1]:

```
import tensorflow as tf
import pickle
import random
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
np.set_printoptions(threshold=np.nan)
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWa
rning: Conversion of the second argument of issubdtype from `float` to `np.fl
oating` is deprecated. In future, it will be treated as `np.float64 == np.dty
pe(float).type`.
  from ._conv import register_converters as _register_converters
```

In [2]:

```
model_path = './image_classification'
batch_size = 64
n_samples = 10
top_n_predictions = 5


def unpickle(file):

    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

train=unpickle("cifar-100-python/train")
test=unpickle("cifar-100-python/test")
meta=unpickle("cifar-100-python/meta")

Xtest=test[b'data'].reshape((len(test[b'data']), 3, 32, 32)).transpose(0, 2, 3
, 1)
ytest=np.array(test[b'fine_labels'])

Xtest=Xtest.astype(np.float32)


sub=np.load("mean_vec.npy")
Xtest=Xtest-sub

ytestC=np.array(test[b'coarse_labels'])

lab={}
for i in range(0,20):
    lab[i]=set()
for i in range(0,len(ytest)):
    lab[ytestC[i]].add(ytest[i])


name_100_labels=[str(i)[2:-1] for i in meta[b'fine_label_names']]
name_20_labels=[str(i)[2:-1] for i in meta[b'coarse_label_names']]
```

In [3]:

```
def batch_features_labels(features, labels, batch_size):
    for start in range(0, len(features), batch_size):
        end = min(start + batch_size, len(features))
        yield features[start:end], labels[start:end]
```

In [4]:

```
loaded_graph = tf.Graph()
with tf.Session(graph=loaded_graph) as sess:

    loader = tf.train.import_meta_graph(model_path + '.meta')
    loader.restore(sess, model_path)
    loaded_x = loaded_graph.get_tensor_by_name('input_x:0')
    loaded_y = loaded_graph.get_tensor_by_name('output_y:0')
    loaded_logits = loaded_graph.get_tensor_by_name('logits:0')
    loaded_acc = loaded_graph.get_tensor_by_name('accuracy:0')

    label_predictor = sess.run(
            tf.nn.top_k(tf.nn.softmax(loaded_logits),1),
            feed_dict={loaded_x: Xtest, loaded_y: ytest})
#         print(label_predictor[1])

    matFine=confusion_matrix(ytest,label_predictor[1])
    accFine=accuracy_score(ytest,label_predictor[1])

    print("---------------------------100 classes------------------")




    print("\nAccuracy and confusion matrix for 100 labels")
    print("----------------------------------Accuracy=",100*accFine,"---------
---------------")
    print("\nConfusion-Matrix:\n",matFine)




#     print("\nPredicted_labels:\n",label_predictor[1].reshape(len(label_predi
ctor[1])).tolist())
#     print("\n Correct_labels:\n",ytest)
    print("\n\n\n\n\n")



    coarse_pred=[]
    pred=np.reshape(label_predictor[1],(len(label_predictor[1]))).tolist()
    for i in range(0,len(pred)):
        for pos in lab.keys():
            if(pred[i] in lab[pos]):
                coarse_pred.append(pos)
    matCoa = confusion_matrix(ytestC,coarse_pred)
    accCoa = accuracy_score(ytestC, coarse_pred)
    print("---------------------------20  classes------------------")
    print("Accuracy and confusion matrix for 20 labels")
    print("----------------------------------Accuracy=",100*accCoa,"----------
--------------")
    print("Confusion-Matrix:\n",matCoa)



#     random_test_features, random_test_labels = tuple(zip(*random.sample(list
(zip(Xtest, ytest)), n_samples)))
```

```
    Top_3_pred= sess.run(
        tf.nn.top_k(tf.nn.softmax(loaded_logits), 1),
        feed_dict={loaded_x: Xtest, loaded_y: ytest})
    print("---------------------------------------Considering Top-3-----------
-----------------\n")
#     print("Top 5 Classes:\n"Top_5_pred[1])
    acc_top_3=0
    for i in range(0,len(Top_3_pred[1])):
        if(ytest[i] in Top_3_pred[1][i]):
            acc_top_3+=1
    print("------------------------Accuracy for 100 classes= ",100*acc_top_3/l
en(ytest))

    cacc_top_3=0
    for i in range(0,len(ytest)):
        temp=[]
        for j in lab.keys():
            if(ytest[i] in lab[j]):
                temp.append(lab[j])
                break
#         print(temp)
#         print(Top_5_pred[1][i])
        if(len(temp)!=0):
            temp2=set(temp[0]).intersection(Top_3_pred[1][i])
#             print(temp2)
            if(len(temp2)!=0):
                cacc_top_3+=1
    print("------------------------Accuracy for 20 classes= ",100*cacc_top_3/l
en(ytest))


    Top_5_pred= sess.run(
        tf.nn.top_k(tf.nn.softmax(loaded_logits), top_n_predictions),
        feed_dict={loaded_x: Xtest, loaded_y: ytest})
    print("---------------------------------------Considering Top-5-----------
-----------------\n")
#     print("Top 5 Classes:\n"Top_5_pred[1])
    acc_top_5=0
    for i in range(0,len(Top_5_pred[1])):
        if(ytest[i] in Top_5_pred[1][i]):
            acc_top_5+=1
    print("------------------------Accuracy for 100 classes= ",100*acc_top_5/l
en(ytest))

    cacc_top_5=0
    for i in range(0,len(ytest)):
        temp=[]
        for j in lab.keys():
            if(ytest[i] in lab[j]):
                temp.append(list(lab[j]))
                break
        if(len(temp)!=0):
            temp2=set(temp[0]).intersection(Top_5_pred[1][i])
            if(len(temp2)!=0):
                cacc_top_5+=1
    print("------------------------Accuracy for 20 classes= ",100*cacc_top_5/len(ytest))
```

```
INFO:tensorflow:Restoring parameters from ./image_classification
---------------------------100 classes------------------

Accuracy and confusion matrix for 100 labels
----------------------------------Accuracy= 35.13 ------------------------

Confusion-Matrix:
 [[62  3  1  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  1  0  0  0
   1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0
   0  0  0  0  0  3  0  0  0  6  0  0  0  0  1  0  0  0  0  0  0  0  1  0
   0  0  0  0  0  0  0  0  0  0  1  6  0  0  0  0  0  0  0  0  8  0  0  0
   0  0  1  0]
 [ 0 51  0  0  0  0  1  0  0  0  1  1  0  1  3  0  0  0  1  0  0  0  1  0
   0  0  2  1  0  0  0  0  0  1  1  0  0  1  1  1  0  1  1  0  1  1  0  0
   4  0  0  1  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  3  0
   0  1  1  1  0  1  1  0  0  0  2  2  0  0  0  1  0  0  1  2  5  0  0  0
   0  0  0  0]
 [ 0  0 23  0  0  1  1  0  0  0  0  2  0  0  2  0  0  0  0  1  1  0  0  0
   0  0  4  0  0  0  0  0  1  0  0 10  2  0  0  0  0  1  0  0  0  1  3  0
   2  0  0  1  0  1  2  0  1  1  0  0  0  2  0  0  0  2  1  0  0  0  0  0
   0  0  0  0  0  2  0  0  0  0  0  3  2  0  2  2  0  0  0  1  2  0  0  0
   0  1 18  1]
 [ 0  0  0 15  6  0  4  1  1  0  0  0  0  1  1  0  1  0  0  6  0  7  0  0
   0  0  1  0  1  3  2  3  0  0  0  0  0  0  0  0  0  0  4  0  0  2  0  0
   3  0  0  3  1  0  0  1  1  0  1  1  0  2  0  1  0  3  3  0  0  0  0  0
   4  0  2  4  0  1  2  1  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0  1
   0  2  1  0]
 [ 0  0  0  1 13  0  1  1  0  0  1  0  0  1  0  1  1  0  1  2  0  3  0  0
   1  0  2  1  0  1  0  0  1  4  0  1  0  0  3  0  0  0  3  4  1  1  0  1
   0  0  3  4  0  0  0  0  3  0  1  0  0  0  0  3  3  0  7  0  0  0  0  0
   2  0  7  1  0  3  1  0  0  2  0  0  0  0  0  1  1  3  1  1  0  2  0  1
   0  0  0  0]
 [ 0  0  0  0  0 29  0  0  2  0  1  0  3  4  1  0  1  0  0  1  8  0  1  2
   1  3  1  0  1  0  0  0  1  0  0  0  6  2  0  0  1  1  0  0  0  3  0  0
   0  0  0  0  0  0  1  0  0  1  1  0  0  1  0  0  0  1  0  0  0  0  0  0
   0  0  0  0  0  1  0  1  0  4  0  0  3  0  0  6  0  1  1  2  0  0  2  0
   0  0  1  0]
 [ 0  1  1  1  0  0 46  4  0  0  0  0  0  0  6  0  0  0  3  1  1  0  0  0
   2  0  0  0  0  2  0  0  0  0  2  0  0  0  0  0  0  1  0  2  0  3  1  0
   2  0  1  0  0  0  2  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0
   1  0  0  1  0  0  0  1  0  0  7  0  1  0  1  0  1  0  0  0  1  0  0  0
   0  1  1  0]
 [ 0  0  0  0  0  0  3 43  1  0  1  0  0  0  6  0  1  0  0  1  2  2  0  0
   8  0  1  0  1  1  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  1  1  1
   6  0  0  0  0  0  0  1  1  0  0  0  0  0  2  0  1  0  0  0  1  0  0  0
   0  0  0  0  0  0  4  2  0  0  0  0  0  0  2  0  1  0  0  0  0  0  0  2
   0  0  0  1]
 [ 0  1  1  1  0  0  2  2 30  0  0  0  0  0  0  0  0  0  1  2  0  0  0  0
   1  0  0  0  0  0  1  0  0  1  1  1  0  0  0  0  0  0  0  1  0  0  2  0
  24  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1  0  0  0  0  0  0  0
   1  0  0  0  0  0  1  1  0  1  0  0  3  0  2  1  2  5  1  0  0  1  0  2
   0  1  1  2]
 [ 0  0  0  1  0  1  0  2  0 48  0  0  1  0  0  0  2  1  1  1  0  1  0  0
   0  0  0  0  2  1  0  0  0  0  0  1  0  0  0  0  4  2  0  0  0  0  2  0
   1  0  0  1  0  1  0  1  0  0  0  0  0  2  0  0  1  1  1  0  0  1  0  0
   0  0  0  0  2  0  1  1  0  1  0  3  2  0  2  4  0  1  0  0  1  0  0  0
   0  0  1  0]
```

```
[ 3  1  1  0  0  0  0  1  0  0 20  1  0  1  2  0  3  0  0  1  3  0  5  0
  0  0  3  0  6  0  0  0  0  1  0  0  0  0  0  1  1  0  0  0  0  1  0  0
  0  0  0  1  0  5  0  0  1  2  0  0  0 11  2  0  0  0  1  3  0  0  1  0
  0  0  0  1  0  0  0  0  0  1  0  2  2  0  1  2  1  0  0  2  2  0  0  1
  0  1  1  1]
[ 0  2  8  0  1  0  1  0  1  0  0 18  0  2  3  0  0  0  0  0  2  5  1  1
  0  1  0  0  1  1  0  0  0  0  0 15  2  1  2  0  1  0  0  1  0  1  2  0
  0  0  0  1  0  1  0  0  0  0  4  0  0  0  0  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  1  0  0  0  0  0  1  5  1  0  1  0  1  1  0  0  0  0  0
  0  0  9  0]
[ 0  0  0  0  0  1  0  0  2  0  0  0 43  3  0  0  0  3  2  2  2  0  1  0
  0  1  1  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  1  0  0  1
  4  0  0  0  0  0  0  0  1  0  4  1  0  0  0  0  0  0  0  0  1  1  0  0
  0  0  0  0  0  0  0  0  1  2  0  0  3  0  1  2  1  2  5  0  0  0  0  3
  0  0  0  1]
[ 0  0  0  0  0  1  0  0  0  0  1  1  2 31  1  0  2  0  0  0  0  1  0  0
  0  1  0  0  0  0  0  0  0  1  0  0  0  4  0  0  0  2  0  0  0  0  0  0
  1  0  0  0  0  0  0  0  1  0 14  0  1  0  0  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  1  0  0 14  0  0  4  0  1  5  1  2  3  0  2  0  0  1
  0  0  0  0]
[ 0  2  0  1  0  0  4  3  1  1  1  0  0  0 40  0  0  0  4  2  0  1  0  0
  1  0  5  0  0  0  0  0  0  2  0  0  0  0  1  0  1  0  1  0  0  2  0  0
  4  0  0  1  0  0  2  0  1  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0
  0  0  1  2  0  1  0  1  0  0  0  1  1  0  3  0  0  1  0  1  2  0  0  0
  0  0  0  1]
[ 0  1  0  2  1  0  2  1  1  2  1  1  2  0  2 26  1  3  0  5  1  0  1  0
  2  0  0  0  0  8  0  0  0  1  0  0  2  0  2  0  0  0  1  5  1  0  0  0
  1  0  1  1  0  0  0  0  1  0  0  0  0  0  0  0  2  0  1  0  0  0  0  0
  1  0  0  4  0  0  0  2  0  0  0  0  2  0  1  1  1  1  0  0  0  2  0  0
  0  0  3  0]
[ 0  0  1  0  0  2  0  0  1  5  0  0  0  3  0  0 31  0  1  0  1  0  5  0
  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  2  1  0  0  0  0  1  0  0
  4  0  1  0  0  0  0  0  0  0  2  0  0  4  1  0  0  0  2  0  0  0  0  0
  1  0  0  0  1  0  0  0  0  3  0  1  4  0  7  5  0  0  0  0  1  0  2  3
  0  1  1  0]
[ 0  0  0  0  0  0  0  0  0  1  0  0  2  2  0  2  0 46  0  4  0  0  0  0
  0  0  0  0  0  0  0  0  0  1  0  0  0 22  0  0  0  0  1  1  0  0  0  0
  0  2  0  0  1  0  0  0  1  0  1  1  0  0  0  0  0  0  2  0  1  1  0  0
  0  0  0  0  1  0  0  0  0  1  0  0  0  1  0  0  0  0  2  1  0  0  0  0
  1  1  0  0]
[ 0  1  0  0  0  1  4  1  0  1  2  0  0  1  8  2  3  0 32  1  0  1  0  0
  0  1  0  0  0  0  0  0  0  1  0  0  0  1  1  0  0  2  1  0  3  1  0  0
  0  0  0  0  0  0  1  0  0  1  1  0  0  0  1  0  2  0  0  0  0  2  0  0
  0  0  2  0  0  3  1  1  0  0  1  2  2  0  1  0  0  0  0  4  1  0  0  1
  3  0  0  1]
[ 0  1  0  1  1  0  0  2  1  0  0  0  0  0  1 11  0  1  0 33  0  0  1  0
  0  1  1  0  0  4  0  4  0  1  3  0  0  1  1  0  0  1  1  4  0  2  1  0
  5  0  0  0  1  0  0  0  0  0  2  0  0  0  0  0  0  1  4  0  0  0  0  1
  0  0  1  0  0  0  2  1  0  0  0  0  1  0  0  0  0  1  2  0  0  0  0  0
  0  0  0  0]
[ 0  0  0  0  0  1  1  0  0  3  0  1  0  1  1  1  1  0  0  1 71  0  0  0
  0  2  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0
  3  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  2  0  2  2  0  0  0  1  0  0  0  0
  0  0  0  0]
[ 0  0  0  8  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  1  0 43  0  0
  0  0  0  0  0  1  0  0  0  1  0  1  0  0  0  2  1  0  0  1  0  0  3  0
```

```
   4  0  1  0  0  0  0  0  0  0  2  0  0  0  1  0  3  0  7  0  0  0  0  0
   0  0  1  2  0  0  2  1  0  2  0  0  0  0  1  1  1  0  0  0  0  0  0  0
   0  2  4  1]
 [ 1  1  0  0  0  0  1  0  1  0  6  1  0  3  1  0  0  0  0  0  1  0 27  0
   0  1  1  0  0  0  0  0  0  0  0  0  2  2  0  0  1  0  0  2  1  0  0  0
   5  0  0  1  0  0  0  0  0  0  2  0  0 11  1  0  1  0  1  0  0  0  1  0
   1  0  0  0  0  2  2  1  0  2  1  0  0  0  4  4  2  0  0  0  0  0  2  1
   0  0  1  1]
 [ 0  0  0  0  0  0  0  0  0  1  0  1  2  0  0  0  0  0  1  0  0  0  0 38
   0  1  0  1  4  0  2  0  2  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0
   0 10  1  0  0  0  0  0  0  0  1  0  0  1  0  0  1  0  1  4  0  1  0 13
   0  4  0  0  0  0  0  0  0  2  0  0  0  2  0  0  0  0  0  0  0  1  1  1
   0  1  0  0]
 [ 0  0  0  0  0  0  2  5  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0
  69  0  2  0  1  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  1  1  0  0
   2  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   2  0  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  2
   0  0  2  0]
 [ 0  1  0  0  1  6  0  1  4  1  0  3  5  0  0  0  3  4  0  1  4  1  0  1
   0 17  2  0  0  0  1  0  0  3  0  0  0  0  0  0  1  0  0  0  0  2  0  0
   0  0  0  0  0  0  0  0  0  0  2  0  0  0  1  0  0  2  1  0  0  0  1  0
   0  0  0  1  0  0  0  0  0  1  1  1  7  1  3  4  0  3  1  3  1  0  2  0
   0  0  2  0]
 [ 0  2  0  0  0  0  2  2  0  0  1  0  0  0  5  0  0  0  1  0  1  0  1  0
   2  0 32  0  0  2  0  0  0  0  0  0  2  0  0  1  0  1  0  1  4  9  2  0
   6  0  2  2  0  0  0  0  0  0  0  0  0  2  2  0  1  0  2  0  0  0  0  0
   0  0  4  0  0  1  1  0  0  0  0  1  1  0  1  0  0  1  0  1  1  0  0  0
   0  0  0  0]
 [ 0  1  1  0  1  1  2  3  1  0  0  1  0  0  1  1  1  0  5  0  0  0  0  1
   0  1  1 16  0  0  1  0  0  0  1  1  0  1  1  0  1  0  2  0  8  3  0  0
   2  1  0  1  0  0  0  0  0  0  3  0  0  0  0  0  0  0  3  1  0  0  0  0
   1  0  2  4  0  2  3  0  1  0  0  1  2  1  2  0  0  0  1  0  0  2  0  3
   3  2  1  1]
 [ 1  0  0  0  0  0  1  0  0  1  1  2  1  0  0  1  1  0  1  0  1  0  0  0
   0  1  0  0 48  0  0  0  1  0  0  1  0  0  0  0 13  0  0  0  0  0  3  0
   1  0  0  0  0  0  0  0  0  0  1  0  0  1  1  0  0  2  1  1  1  3  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  3  0  3  0  0  0  0  0  1  0  0  1
   0  0  1  1]
 [ 0  0  0  2  0  0  5  1  2  0  0  0  0  1  2  3  0  0  0  8  0  0  0  0
   1  0  4  0  0 32  0  3  0  0  1  0  0  2  0  0  0  0  0  0  3  2  1  0
   2  0  0  1  1  1  0  0  0  0  0  1  0  0  0  0  2  0  2  0  0  0  0  0
   0  0  1  1  0  0  0  1  1  0  0  0  1  0  1  0  0  2  1  0  0  2  0  3
   0  0  1  2]
 [ 0  0  0  0  0  0  0  0  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0 37  1  1  0  0  1  0  0  1  1  0  0  0  0  0  0  0  0
   0  3  0  1  0  0  0  0  1  0  0  1  0  0  0  0  0  0  1  1  0  4  0  0
   0  4  0  1  0  0  1  1  0  1  0  0  0  0  1  1  0  0  0  1  0  3  0 28
   0  1  0  0]
 [ 0  0  0  4  2  0  2  0  3  0  0  0  0  0  1  4  0  1  0  7  0  1  1  0
   0  1  1  0  0  8  0 33  1  2  0  0  0  2  1  0  0  0  0  0  0  0  0  1
   1  0  0  1  1  0  0  2  1  1  0  1  0  0  0  0  2  0  4  0  0  0  0  0
   0  0  1  0  1  0  2  0  1  0  0  0  2  0  1  0  0  0  0  0  0  1  0  0
   0  1  0  0]
 [ 0  1  2  0  2  0  1  0  1  0  0  3  0  4  3  0  0  1  0  2  0  0  1  0
   2  0  4  0  0  1  0  0 27  0  0  0  1  0  0  1  0  0  0  1  1  0  0  1
   4  0  0  1  0  1  0  1  0  0  1  0  0  0  0  2  0  2  2  2  0  0  0  0
   0  0  1  0  1  0  2  0  0  2  0  0  0  0  3  0  0  2  1  4  2  2  1  1
   0  0  2  0]
 [ 0  1  0  0  0  0  1  0  0  2  0  0  2  1  0  1  0  2  5  2  0  0  0  0
   0  0  0  0  0  0  0  2  0 39  0  0  0  3  1  0  0  0  1  1  0  1  0  0
   0  1  0  1  1  0  0  0  0  0  0  3  2  0  1  0  0  0  0  0  0  0  0  0
   0  0  0  1  1  2  0  2  1  1  0  1  0  1  0  0  0  2  0  0  0  0  2  0
  11  1  0  0]
 [ 0  1  0  1  2  0  6  1  1  0  0  1  0  0  4  3  1  0  1  3  0  0  0  0
   0  0  1  0  1  4  0  1  0  2 24  1  2  1  1  0  0  0  4  3  2  2  0  0
   3  1  0  2  0  0  1  0  0  0  0  0  0  0  0  0  1  0  2  0  0  1  0  0
   0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  1  4  1  0  0  3  0  0  0
   0  2  2  0]
 [ 0  0  9  1  0  0  1  2  0  1  0  8  0  0  3  1  2  0  0  0  0  2  2  0
   3  0  0  0  0  0  0  0  0  0  0 19  2  0  0  1  0  0  0  1  0  2  0  0
   1  0  0  0  0  1  3  0  0  1  1  0  0  1  0  1  1  0  0  0  0  0  0  0
   0  0  0  0  0  0  2  0  0  0  0  1  5  0  1  1  2  0  0  1  0  0  0  0
   0  4 13  0]
 [ 0  2  3  0  0  1  0  0  0  0  1  1  0  0  4  0  0  0  1  0  0  1  0  0
   0  0  2  0  0  0  0  0  2  0  3  3 33  1  0  0  0  0  0  4  0  0  0  1
   0  0  3  1  0  0  1  0  0  0  0  0  0  3  0  0  5  3  3  0  0  1  0  0
   0  1  1  0  0  0  1  0  0  1  0  0  0  0  1  0  1  2  0  0  1  1  0  0
   0  3  4  0]
 [ 0  0  0  0  0  2  0  0  0  2  0  1  5 14  0  1  0  5  0  1  0  0  0  0
   0  2  0  0  0  1  0  1  0  0  0  1  0 28  1  0  0  1  2  0  0  1  0  0
   3  0  0  0  1  0  0  0  2  0  1  1  0  0  0  0  0  0  0  0  1  0  0  0
   1  0  0  1  1  0  0  0  0  6  0  0  1  0  0  6  0  2  3  0  0  0  0  1
   0  0  0  0]
 [ 0  0  0  4  1  0  3  0  2  0  0  0  0  0  3  2  1  1  1  4  0  6  0  0
   0  0  1  1  1  1  0  2  0  0  1  0  0  2 11  0  0  0  2  4  5  0  1  0
   0  0  1  0  0  0  0  0  1  0  0  0  0  1  0  0  1  3  4  1  0  0  0  0
   0  0  3  2  0  4  1  2  3  1  0  0  0  0  0  0  0  3  1  0  0  0  0  0
   0  4  2  2]
 [ 0  1  0  0  0  1  0  0  1  1  1  0  0  3  1  0  1  0  0  0  1  1  1  1
   0  1  0  0  0  0  0  1  2  3  0  1  0  1  0 44  1  0  0  2  0  2  0  0
   1  1  0  0  0  1  1  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  1
   0  0  0  0  1  0  1  0  0  3  0  0  3  0  1  1  0  0  0  1  1  0  1  1
   0  1  1  5]
 [ 1  0  0  0  0  0  1  1  0  1  0  2  3  0  1  0  2  0  0  2  2  1  0  1
   0  0  1  0  5  2  1  0  0  1  1  2  0  0  0  0 36  1  0  0  0  0  1  0
   0  0  1  5  0  0  0  1  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0
   0  0  0  3  1  0  1  2  0  2  0  0  2  0  2  1  1  0  0  1  2  2  0  0
   0  0  2  0]
 [ 0  1  0  0  0  0  1  1  0  0  0  2  0  1  6  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1 60  0  0  0  3  0  0
   6  0  0  0  1  0  1  0  0  0  3  0  0  0  0  0  0  0  1  1  0  0  0  0
   1  0  1  0  0  0  1  0  0  0  0  0  0  0  0  3  0  1  2  0  1  0  0  0
   0  0  0  0]
 [ 0  1  0  2  2  1  7  1  0  0  0  0  0  1  2  2  0  0  0  0  0  1  0  0
   0  0  0  1  0  3  0  0  0  5  2  0  0  0  1  0  0  0 15  4  0  2  0  0
   4  0  1  1  0  0  0  0  1  0  0  0  0  1  0  1  2  1  4  0  0  0  0  0
   1  0  2  1  0  1  5  0  1  0  0  0  1  0  1  1  7  0  0  0  1  1  0  0
   1  7  0  0]
 [ 0  0  1  0  1  0  2  1  1  0  0  0  0  0  2  5  0  0  0  3  0  0  1  0
   0  0  1  0  0  0  0  0  0  0  2  0  1  0  0  0  0  0  5 46  1  2  0  1
   0  0  0  2  0  1  0  0  0  1  0  0  0  0  0  0  1  0  1  0  0  0  0  0
   0  0  2  0  0  0  1  1  2  0  2  0  0  0  1  0  3  3  0  0  0  0  0  0
   0  1  2  0]
 [ 0  0  2  1  0  2  2  1  0  0  1  1  0  0  3  2  0  0 10  0  0  0  0  0
```

```
  1  1  4  3  0  0  0  0  0  1  0  0  1  0  0  0  0  1  1  0 16  2  0  1
  5  0  1  0  0  0  0  0  1  0  0  0  0  0  1  0  0  1  1  0  0  0  0  0
  0  0  1  0  0  1  7  3  0  0  0  1  1  0  2  1  2  1  0  3  1  3  0  0
  0  2  1  3]
 [ 0  1  2  0  0  0  4  2  1  0  1  0  2  1  8  0  0  0  0  0  2  0  1  0
  1  0  6  1  0  6  1  1  0  1  1  1  0  1  0  0  0  0  0  1  4 25  0  0
  4  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  2  0
  0  0  1  0  0  0  1  0  0  2  0  3  1  0  0  0  1  1  0  2  1  1  0  0
  1  0  0  1]
 [ 0  0  3  1  1  1  0  0  0  1  2  4  0  0  1  0  1  0  0  1  0  3  2  0
  1  0  0  0  0  1  0  1  2  0  0 11  1  0  0  0  1  0  0  2  0  1  9  0
  6  0  0  2  0  0  0  0  3  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0
  0  1  0  4  0  1  0  0  0  2  0  2  0  0  3  2  0  0  2  2  1  0  1  0
  0  0 15  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
  0  0  2  0  0  2  0  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0  0 44
  0  0  0  0 22  0  0  0  3  0  0  8  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  2  0  0  0
  9  0  0  0]
 [ 0  0  0  0  0  0  0  3  2  0  0  0  0  1  1  0  1  0  0  0  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  2  0  0
 78  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  2  2  0  1  0  0  1  0  0  0
  0  0  0  0]
 [ 0  1  0  0  0  1  0  0  1  0  0  0  8  0  1  0  0  5  0  0  1  0  0  1
  0  0  0  0  0  1  1  0  0  0  0  0  1  2  0  1  0  0  0  0  0  0  0  1
  2 40  0  1  2  0  0  0  0  0  0  3  1  0  0  0  0  0  0  0  0  1  0  5
  0  0  0  0  1  0  0  0  1  2  0  0  0  1  0  0  0  0  4  0  0  1  0  7
  1  1  0  0]
 [ 0  0  2  1  1  2  2  2  2  0  2  0  0  2  0  2  0  0  1  0  0  0  0  0
  2  0  9  0  1  0  1  0  2  1  1  1  6  0  2  1  0  0  2  4  1  0  0  0
  0  0  8  4  0  0  1  0  0  0  0  0  0  0  1  0  1  1  6  0  0  1  0  0
  1  0  4  0  0  1  2  0  0  0  1  0  2  1  1  0  1  0  0  1  2  1  0  1
  1  3  1  0]
 [ 0  1  0  1  2  0  6  0  0  1  1  0  0  1  6  2  0  0  3  1  0  1  0  0
  0  1  4  1  0  3  0  0  0  2  0  1  0  1  0  0  1  0  2  1  0  0  1  0
  2  0  0 21  0  0  0  0  1  0  0  0  1  0  1  0  1  0  2  0  0  0  1  0
  0  0  0  5  0  2  1  1  0  1  0  1  3  0  0  1  3  0  1  1  4  0  0  0
  0  0  2  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0
  0  0  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0 16
  0  0  0  0 64  0  0  0  1  0  0  5  0  0  0  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  4  0  0  0]
 [ 1  0  0  0  0  0  0  1  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0 73  0  0  0  3  0  0  0  0  5  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  8  1  0  0  0  0  0  0  0  3  0  0  0
  0  0  1  0]
 [ 0  1  1  0  0  0  5  0  1  0  0  1  0  0  3  0  0  0  0  0  0  0  0  0
  0  1  1  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  1  0  5  0  0
  0  0  0  0  0  1 49  0  0  1  0  0  0  1  2  0  0  1  2  0  0  0  5  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 10  0  0  0
  0  2  1  2]
 [ 0  0  1  4  2  0  0  1  1  0  1  2  1  0  4  0  0  0  2  1  0  5  0  0
  0  0  4  2  1  3  2  3  0  1  1  1  1  0  4  0  0  0  0  0  1  1  0  0
  0  0  0  1  0  0  0  5  2  0  1  0  0  0  0  0  0  1  3  0  0  0  0  1
  2  1  3  5  1  1  3  2  0  1  0  0  1  2  0  2  0  1  1  0  0  2  0  6
  0  0  1  1]
[ 0  0  0  0  0  0  1  0  1  0  0  0  2  0  1  0  0  2  0  0  0  1  0  0
  0  0  0  0  0  7  0  1  0  3  1  0  0  0  0  0  3  0  0  0  0  0  0  2
  0  0  0  2  5  0  0  0 54  0  0  5  0  0  0  0  0  0  0  0  0  1  0  0
  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  2
  2  0  0  1]
[ 1  0  1  0  0  0  3  2  0  1  0  1  0  0  5  1  0  0  1  0  0  0  2  0
  1  0  2  0  0  2  0  0  1  0  1  1  0  0  2  0  0  0  0  0  0  2  0  0
  1  0  0  0  0  9  2  0  0 34  0  0  0  3  2  0  0  0  0  0  1  1  0  1
  0  1  0  0  0  1  0  0  1  0  2  3  0  0  0  0  0  1  0  0  1  0  0  0
  0  0  4  2]
[ 0  0  0  1  0  1  0  0  2  0  1  0  1 11  0  0  0  0  0  0  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1  0  0
 10  0  0  0  0  0  0  0  0  0 54  0  0  0  0  0  0  0  0  0  1  0  0  0
  1  0  0  1  0  0  1  0  0  5  0  0  1  0  0  0  0  3  2  0  0  0  0  0
  0  0  0  0]
[ 0  0  0  0  1  0  1  1  1  0  0  0  3  1  0  0  0  1  1  1  0  0  0  0
  0  0  0  0  0  1  0  1  0  4  0  0  1  0  0  0  0  1  1  1  0  2  0  9
  1  0  0  0 21  0  0  0 12  0  1 17  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  1  0  0  0  0  2  0  0  0  1  0  0  1  1  2  0  0  0  0  2
  6  0  0  0]
[ 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  4  0  0  0  0  0  2
  0  2  0  0  1  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0  0  0
  0  2  0  0  0  0  0  0  0  0  0  0 70  0  0  0  0  0  0  0  1  0  0  9
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  2  0  0  0  0  0
  1  0  0  0]
[ 0  1  1  0  0  0  0  0  0  1 13  1  0  0  0  0  0  0  0  0  0  0 11  0
  0  0  2  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0  0  1  0  1  0  0
  0  0  0  0  0  0  1  0  0  0  1  0  0 49  1  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  1  2  0  0  0  1  1  0  0  2  0  0  0  0  2  0  0  0  0
  1  0  2  0]
[ 0  1  1  0  0  0  1  2  0  1  2  0  0  0  1  0  0  0  0  0  0  0  0  0
  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  1  0
  1  0  0  1  0  6  3  0  0  0  1  0  0  0 47  0  0  0  0  0  0  0  5  0
  0  0  0  0  0  0  0  0  0  0  2  2  0  0  0  0  0  0  0  0 16  1  0  0
  0  0  0  0]
[ 0  1  0  7  3  0  2  1  0  0  0  0  0  0  1  0  1  0  0  0  0  5  1  0
  0  0  0  2  0  0  1  1  1  1  1  0  1  0  1  0  0  1  4  1  0  1  0  0
  1  0  0  3  0  0  0  0  2  0  0  0  0  0  0 21  4  3  5  1  0  0  0  0
  1  0 10  1  0  3  0  0  0  0  0  1  2  0  0  0  0  0  0  0  0  1  0  0
  0  3  0  0]
[ 0  0  2  2  1  0  1  1  2  0  0  0  0  0  6  2  0  0  1  1  0  1  0  0
  1  1  0  1  0  1  0  1  1  0  1  2  1  0  1  0  0  0  1  3  2  1  0  0
  2  0  1  1  0  1  0  0  0  0  0  0  0  0  0  3 11  1 15  0  0  0  0  0
  0  0  1  3  0  0  5  1  2  0  0  0  1  0  2  2  0  0  1  0  0  1  0  0
  0  5  2  1]
[ 0  0  5  3  0  1  1  1  1  0  0  0  0  0  4  0  0  0  2  2  0  2  1  0
  0  0  1  2  0  0  0  0  0  2  2  1  0  0  2  1  0  1  0  3  0  1  0  0
  0  0  2  1  0  0  2  0  2  0  2  0  0  4  0  0  1 16  4  0  0  0  0  0
  0  1  1  1  0  1  1  5  1  2  0  0  0  0  1  0  1  1  1  0  3  2  1  0
  1  1  0  2]
[ 0  1  1  3  0  0  1  0  2  0  0  0  0  1  2  0  0  0  0  2  0  0  1  0
  1  0  0  0  0  0  0  0  0  1  0  1  2  0  2  0  0  1  1  1  0  1  0  0
  5  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0 45  0  0  0  0  0
  0  0  2  8  0  0  0  1  1  0  0  0  2  0  1  0  2  0  0  0  0  0  0  0
  0  3  1  0]
```

```
[ 0  0  0  0  0  0  1  0  1  0  1  0  1  0  1  1  0  0  0  0  0  0  1  0
  0  0  0  1  1  2 12  0  3  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0
  0  1  2  1  0  0  0  0  0  1  0  0  0  1  0  0  3  0  1 18  0  0  0  0
  1 13  2  0  0  3  0  1  0  0  0  1  1  0  2  0  0  0  1  1  0  6  0 10
  0  1  0  1]
[ 0  0  1  0  0  0  0  0  0  0  0  1  5  2  0  0  1  1  0  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  3  0  0  0  5  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  0  0  0  1  0  3  0  0  0  1  1  0  0 59  0  0  5
  1  0  0  1  0  0  0  1  0  0  1  0  1  0  0  0  0  0  1  1  0  0  0  1
  0  1  0  0]
[ 0  0  0  0  0  2  1  0  0  2  0  0  4  0  0  0  1  0  1  0  1  0  0  0
  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  6  0  0  1  1  0  1  0
  1  2  0  0  0  0  1  0  1  2  1  0  0  1  0  0  0  0  0  1  0 50  0  1
  0  1  0  0  2  0  0  1  0  0  0  0  1  0  0  0  1  0  1  0  0  1  0  3
  0  0  0  4]
[ 2  4  0  0  0  0  0  0  0  0  1  0  0  0  1  0  1  0  0  0  1  0  0  0
  0  0  2  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
  2  0  0  2  0  0  9  0  0  0  2  0  0  1 11  0  0  0  0  0  0  0 31  0
  0  0  0  0  0  0  0  0  0  0  0  2  0  0  1  1  0  2  0  0 21  1  0  0
  0  0  1  0]
[ 0  1  0  0  1  0  0  0  0  0  0  0  2  1  0  0  0  0  0  0  1  0  0  5
  0  1  0  0  2  0  1  0  0  0  0  0  0  0  0  2  2  0  0  0  0  2  0  0
  0  1  0  1  0  0  0  0  1  0  0  0  4  0  0  0  0  0  0  1  4  0  0 57
  0  2  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  5
  0  0  0  0]
[ 0  0  1  4  3  0  1  3  0  0  1  1  2  0  1  0  0  0  0  0  1  4  1  0
  0  0  0  2  0  2  1  1  3  1  0  0  0  2  0  2  1  1  3  0  0  0  0  0
  4  1  0  1  0  0  0  2  2  1  1  1  0  0  1  0  1  1  3  0  0  0  0  0
  5  0  2  3  1  3  1  3  0  0  0  0  1  2  1  0  1  0  3  0  0  1  0 11
  0  0  1  0]
[ 0  2  0  0  0  0  0  1  1  0  0  0  1  1  0  0  0  0  2  0  0  1  0  0
  0  0  0  1  1  1 19  1  1  0  1  0  0  0  0  3  2  0  0  0  0  1  1  0
  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
  0 22  0  3  0  0  0  0  0  1  0  0  1  0  1  0  0  0  2  3  0  6  0 12
  0  0  0  4]
[ 0  0  0  0  5  1  2  5  1  0  1  2  0  0  3  0  0  0  1  1  0  1  1  0
  0  1  3  0  0  2  0  0  0  0  1  0  1  0  2  3  0  0  3  1  2  1  0  0
  0  0  3  0  1  0  1  0  0  0  2  0  0  0  0  1  0  0  9  0  0  0  0  0
  0  0 16  2  0  1  3  0  0  0  0  0  0  0  1  0  2  0  2  1  0  2  0  1
  0  4  2  2]
[ 0  0  0  2  0  0  1  2  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0
  2  0  1  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  6  0  0  0  0  0
  0  0  0 62  0  0  4  1  0  0  0  0  0  1  3  1  0  1  1  2  0  0  0  1
  0  0  1  0]
[ 0  0  0  1  0  0  0  1  1  3  0  0  2  0  0  0  3  7  0  1  0  0  1  0
  0  0  0  0  2  2  0  1  0  1  0  0  0  0  1  0  0  0  1  0  1  0  0  0
  0  2  0  0  2  0  0  0  2  0  0  2  0  0  0  0  0  0  0  1  0  6  0  0
  0  0  0  0 41  0  0  0  1  0  0  0  0  2  2  0  0  1  1  0  0  0  0  7
  0  0  0  1]
[ 0  3  2  0  1  1  4  6  0  0  1  1  0  0  6  1  0  0  1  2  0  1  2  0
  0  0  2  1  1  2  0  0  0  2  1  2  2  0  0  0  0  0  1  1  0  3  0  1
  0  0  0  4  0  0  2  0  1  1  1  0  0  1  0  0  1  0  1  0  0  0  1  0
  1  0  0  2  0 12  3  5  0  0  0  0  1  0  1  0  1  2  1  2  1  1  1  1
  0  0  0  1]
[ 0  0  0  0  1  0  1  2  0  0  1  0  0  0  3  2  0  0  2  0  0  3  3  0
  1  0  5  2  0  0  0  0  0  2  0  0  0  0  0  1  2  0  1  2  4  3  0  0
  7  0  1  1  0  0  0  0  0  0  0  0  0  5  0  0  2  0  4  0  0  2  1  0
  0  0  2  2  0  0 17  0  0  0  1  0  2  0  1  3  2  1  0  0  0  0  0  0
  0  0  3  2]
[ 0  0  0  3  0  0  8  8  0  0  0  0  0  0  2  1  1  0  0  1  0  0  0  0
  3  0  2  1  0  2  0  1  0  5  1  0  0  1  2  0  0  0  0  0  2  0  0  0
  1  0  0  3  0  0  0  3  3  0  0  0  0  0  0  0  1  0  1  1  0  1  0  0
  1  0  0  1  0  0  0 28  2  0  2  0  1  1  0  0  0  1  0  0  0  0  1  0
  1  2  1  0]
[ 0  1  1  1  0  0  6  3  4  1  0  0  0  0  3  0  0  1  3  3  0  0  0  0
  0  0  3  0  1  1  0  0  1  2  3  1  0  0  7  0  1  0  3  1  2  1  1  0
  1  0  1  6  0  0  1  1  2  1  0  0  0  1  0  0  1  1  5  0  0  1  0  0
  0  0  0  1  0  2  1  2  5  0  0  0  0  0  2  2  4  0  0  0  0  0  1  0
  0  2  1  0]
[ 0  0  0  1  0  2  0  0  1  0  0  0  3 14  1  1  0  0  0  0  2  2  0  0
  0  0  1  0  0  0  0  1  3  0  0  0  3  0  1  0  0  1  0  0  0  0  0  0
  1  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  0  0  0 43  0  0  2  0  0  4  0  3  7  0  0  0  0  0
  0  0  0  0]
[ 0  1  0  0  0  0  3  0  0  0  0  0  0  1  2  0  0  0  0  0  0  0  0  0
  1  0  0  0  0  2  0  0  0  1  1  0  0  0  0  0  0  0  0  1  0  1  0  0
  2  1  0  0  0  3  1  0  0  1  0  0  0  0  6  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0 67  3  0  0  0  0  0  1  0  0  1  0  0  0
  0  0  0  0]
[ 7  2  0  0  0  0  2  2  0  1  1  0  0  0  2  0  3  0  1  0  0  0  0  0
  0  0  1  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  3  0  0  0
  1  0  0  0  0 13  0  0  0  1  0  0  0  0  4  0  0  0  0  0  0  0  5  0
  0  0  0  0  0  0  0  0  0  0  1 35  0  0  0  0  2  0  1  0  8  0  1  0
  0  0  0  1]
[ 0  0  0  0  0  7  2  1  1  1  1  2  0  0  0  0  4  1  0  3  6  0  1  0
  0  1  2  0  1  0  0  0  0  1  1  0  1  0  0  2  0  0  0  1  0  0  0  0
  1  0  1  1  0  2  0  0  1  0  1  0  1  1  0  0  0  0  3  0  0  0  0  0
  0  0  0  1  0  1  0  0  0  2  0  0 28  0  4  4  2  0  3  0  0  0  1  0
  0  1  1  0]
[ 0  0  0  0  1  0  0  0  3  0  0  0  3  5  1  0  0  1  0  1  0  0  0  0
  0  0  1  1  0  0  0  0  1  1  0  0  0  2  0  1  0  3  0  0  0  0  0  0
  5  0  0  1  0  0  0  0  0  1  7  0  1  0  0  0  0  0  3  0  0  1  0  0
  0  0  0  3  0  0  0  0  0  4  0  0  1 34  1  0  0  6  6  0  0  0  0  0
  0  0  1  0]
[ 0  0  0  0  0  1  0  4  0  1  2  2  2  0  0  0  1  0  0  0  1  1  2  0
  3  2  0  0  2  0  0  0  0  0  0  0  1  1  0  0  1  1  0  0  1  0  0  0
  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  1  2  0  0  1  0  0  0
  0  0  0  2  0  0  1  1  0  1  2  0  1  0 42  7  0  0  1  2  0  0  3  0
  0  0  2  0]
[ 0  0  0  0  0  1  0  0  0  2  1  0  2  4  0  0  1  0  0  1  2  1  2  0
  0  1  0  0  0  0  0  0  0  1  0  0  1  1  0  0  1  0  0  0  0  0  1  0
  1  0  0  0  0  0  0  1  0  0  0  0  0  2  0  1  0  0  0  0  0  0  0  0
  0  0  0  2  0  0  0  0  0  2  0  0  5  0  2 51  0  0  0  0  0  0  7  0
  0  0  3  0]
[ 0  1  0  0  1  1  3  2  2  0  2  0  0  0  6  1  0  0  0  2  0  2  0  0
  0  0  3  0  1  3  0  0  0  3  2  0  1  0  0  0  0  0  4  3  0  0  1  0
  3  0  0  1  0  0  0  1  1  0  0  0  0  0  0  0  2  0  3  0  0  0  0  0
  0  0  1  0  0  1  5  1  0  1  1  0  3  0  1  1 25  1  1  0  1  0  0  0
  0  2  0  0]
[ 0  0  0  0  0  0  2  1  4  0  0  0  0  3  0  1  0  0  1  0  1  0  0  0
  0  0  1  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  0
 16  0  0  0  0  0  0  0  2  0  7  0  0  0  0  0  0  0  2  0  0  1  0  0
  0  0  1  0  0  1  0  1  0  6  2  0  0  1  0  0  0 38  3  0  1  0  0  0
  1  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  2  5  1  0  0  1  0  0  0  2  0  0
  0  0  0  0  0  1  0  1  1  2  0  0  0  3  0  0  0  1  0  0  0  1  0  0
  7  0  0  0  0  0  0  0  1  0  4  1  0  0  0  0  1  0  1  0  0  0  0  0
  0  0  0  3  0  0  1  1  0 14  0  0  0  2  0  1  0  4 37  0  1  0  0  0
  0  0  0  0]
[ 0  3  0  0  1  0  1  0  1  0  1  1  1  1  0  0  1  0  4  0  2  0  0  0
  0  0  2  3  0  2  2  0  0  0  2  0  0  0  0  0  1  0  2  0  1  0  0  0
  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1
  0  2  1  6  0  0  1  0  0  0  0  0  4  0  0  1  0  1  1 44  1  0  0  0
  0  0  0  2]
[ 1  2  0  0  0  0  1  1  0  0  0  0  0  0  3  0  0  0  1  2  0  0  2  0
  2  0  0  0  1  0  0  0  1  0  1  0  1  0  0  2  0  0  1  0  0  1  0  0
  3  0  0  1  0  4  7  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  4  0
  0  0  0  0  0  0  0  0  0  0  3  4  0  0  0  0  0  1  0  0 40  0  0  0
  0  0  0  0]
[ 0  2  0  1  0  0  0  2  2  0  0  0  1  1  1  0  0  1  0  0  0  0  0  0
  1  0  7  2  0  2  3  0  1  0  0  0  2  0  0  0  0  1  0  0  3  1  0  0
  1  0  0  1  3  0  0  1  4  1  0  1  0  0  0  1  1  2  2  1  0  0  0  0
  3  4  3  1  0  0  1  1  0  2  0  0  0  0  2  0  0  0  2  0  1 19  0  6
  0  1  0  1]
[ 0  0  0  0  0  1  0  0  0  2  0  0  0  0  0  0  1  0  0  0  0  0  0  0
  0  1  0  0  0  0  0  1  0  1  0  0  0  0  0  1  0  0  0  1  0  0  1  0
  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  4  0  4  9  0  0  0  0  0  0 68  0
  1  0  2  0]
[ 0  0  0  1  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  1
  2  1  0  0  1  0 12  0  1  0  0  0  0  0  0  3  1  0  0  0  0  0  0  0
  5  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  0
  2  5  0  2  0  0  0  0  0  0  0  0  1  0  2  1  0  0  0  1  0  1  0 48
  0  0  0  2]
[ 0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  1  0  0  0  0  0
  0  0  0  0  0  1  0  0  0 11  0  0  0  0  0  0  0  0  1  1  0  1  0 11
  0  2  0  2 17  0  0  0  4  0  0  6  1  0  0  1  1  0  0  0  0  0  0  0
  0  0  0  0  3  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0  1  1  0  0
 28  0  0  0]
[ 0  0  0  1  0  0  0  1  1  0  1  0  0  0  1  2  0  0  0  1  0  2  1  0
  0  0  1  2  0  2  1  0  0  0  0  0  0  3  2  0  0  0  2  3  1  0  0  0
  0  0  0  1  0  0  0  0  1  0  1  0  0  0  0  2  2  1 11  0  0  0  0  0
  0  1  5  2  0  0  4  0  1  2  0  0  2  0  1  1  1  0  0  2  0  1  0  2
  0 27  1  0]
[ 0  1  5  1  0  3  5  0  0  2  1  9  0  1  0  0  1  0  0  2  1  1  1  0
  1  1  6  0  0  1  0  0  0  0  0 10  0  0  0  1  0  1  0  0  0  2  3  0
  2  0  0  1  0  2  0  0  0  1  0  0  0  0  0  0  0  1  2  0  0  0  1  0
  0  0  1  2  0  0  0  0  0  1  0  0  0  0  2  1  0  0  1  0  0  0  0  0
  0  0 21  1]
[ 1  0  1  0  0  1  1  1  0  2  0  0  0  0  3  0  0  0  0  0  0  0  3  2
  1  0  2  0  1  0  1  0  0  1  0  0  1  0  1  7  2  1  0  1  2  2  0  0
  4  0  0  0  0  0  0  0  0  1  0  1  0  8  0  0  1  1  1  0  0  0  0  0
  1  1  0  2  0  0  5  0  1  0  0  0  2  0  2  1  1  0  0  3  1  0  0  3
  0  1  1 20]]
```

```
---------------------------20  classes------------------
Accuracy and confusion matrix for 20 labels
----------------------------------Accuracy= 48.41 ------------------------
Confusion-Matrix:
 [[183  21   1   6   8  18   5  19  23   8  13  29  34  24  12  28  19  11
   20  18]
 [ 61 200  15   7   9  19   9  21   5   7   4   6  22  24  10  33  11   3
   27   7]
 [  0  10 350   8  34   7   2  27   5   0   2   2   5  17   6   4   4   0
   13   4]
 [  7  11  13 251  21  70  22   9   5   7   1   4   8  12  20   5   5   2
   15  12]
 [  2   9  51  12 287   4   7  40   9   2   5   7   9  20  15   8   2   1
    7   3]
 [  5   8  11  42   9 231  39  13   8  14   9   8  13  16  19  11   7   1
   23  13]
 [  2   9   5  25   7  45 274   8   4  16   8   9   7  15  14   1  10   2
   26  13]
 [  9  10  25  13   7  10   9 295   9   4   3  13   8  29   7  13   5   7
   18   6]
 [ 23   5   6   9  10  10   8  34 180   3   8  40  46  18   6  35  24   8
   20   7]
 [ 14   3   2  12   0  13  13   3   9 283  17  16   5   5   4   5   3  17
   50  26]
 [ 18  17   1  12   4   6  13   8   5  39 301   6   3   7   1   5   4  24
   12  14]
 [  8   4   1   7   3  11   8  22  46  14   6 197  40  18  16  39  19   7
   25   9]
 [  9   8   4   3   8  13   7  34  52   1   5  26 214  14  15  24  30   3
   24   6]
 [ 13  16  13  19  18  22  13  71  14   4  11  11  18 151  13  39  16   8
   19  11]
 [  2  11   9  15  20  25  24  24  14   1   1  19  14  23 247   5  11   4
   24   7]
 [ 23  13   5   8   8  19  11  50  19   5   5  23  28  49  13 145  19  15
   31  11]
 [ 11  12  14  15  13  16  10  46  48   2   5  25  50  37  29  24 106   8
   18  11]
 [  6   0   4   0   6   4   1   7   5  17  22   6   4   6   2  14   1 375
    8  12]
 [  6   2   5   5   0  14  13  12   7  16   5   6  11   8   6   6   0   4
  319  55]
 [  8   6   6   4   4  17  12  16   4  18  10   5  10  13   4   5   2   6
   98 252]]
---------------------------------------Considering Top-3---------------------
-------

------------------------Accuracy for 100 classes=  35.13
------------------------Accuracy for 20 classes=  48.41
---------------------------------------Considering Top-5---------------------
-------

------------------------Accuracy for 100 classes=  65.22
------------------------Accuracy for 20 classes=  77.59
```

Model-2

In [ ]:

```
import pickle
import numpy as np
import tensorflow as tf
import matplotlib
matplotlib.use('agg')
import pylab as plt
```

In [ ]:

```
count=1
epc=1
loss_graph_list=[]


#Reading the data
def unpickle(file):

    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

train=unpickle("cifar-100-python/train")
test=unpickle("cifar-100-python/test")

print(train.keys())
print(test.keys())

#Train data
Xtrain=train[b'data'].reshape((len(train[b'data']), 3, 32, 32)).transpose(0, 2
, 3, 1)
ytrain=np.array(train[b'fine_labels'])

#Test data
Xtest=test[b'data'].reshape((len(test[b'data']), 3, 32, 32)).transpose(0, 2, 3
, 1)
ytest=np.array(test[b'fine_labels'])

Xtrain=Xtrain.astype(np.float32)
Xtest=Xtest.astype(np.float32)

Xval=Xtrain[40000:,:,:,:]
Xtrain=Xtrain[0:40000]

yval=ytrain[40000:]
ytrain=ytrain[0:40000]

# Reset graph parameters
tf.reset_default_graph()

x = tf.placeholder(tf.float32, shape=(None, 32, 32, 3), name='input_x')
y = tf.placeholder(tf.int32, shape=(None,), name='output_y')
```

In [ ]:

```
# model Architecture
def leNet(features):
    # layer 1 input
    # input_layer=tf.reshape(features['x'],[-1,32,32,3])

    # conv_layer #1
    conv1 = tf.layers.conv2d(inputs=features, filters=64,
                             kernel_size=[3,3], strides=1,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    # pooling_layer #1
    pool1 = tf.layers.max_pooling2d(inputs=conv1,
                                    pool_size=[2, 2],
                                    strides=2)

    batch1=tf.layers.batch_normalization(inputs=pool1)




    # conv_layer #2
    conv2 = tf.layers.conv2d(inputs=batch1, filters=128,
                             kernel_size=[3,3], strides=1,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    # pooling_layer #2
    pool2 = tf.layers.max_pooling2d(inputs=conv2,
                                    pool_size=[2, 2],
                                    strides=2)

    batch2=tf.layers.batch_normalization(inputs=pool2)



    # Dense_layer #1
    pool2_flat = tf.reshape(batch2, [-1, 4608])
    dense1 = tf.layers.dense(inputs=pool2_flat, units=256,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    batch3=tf.layers.batch_normalization(inputs=dense1)


    # Dense_layer #2
    dense2 = tf.layers.dense(inputs=batch3, units=512,
                             activation=tf.nn.relu,
                             kernel_initializer=tf.contrib.layers.xavier_initi
alizer())

    batch4=tf.layers.batch_normalization(inputs=dense2)
    # logits_final_layer
    logits = tf.layers.dense(inputs=batch4, units=100)

    return logits
```

In [ ]:

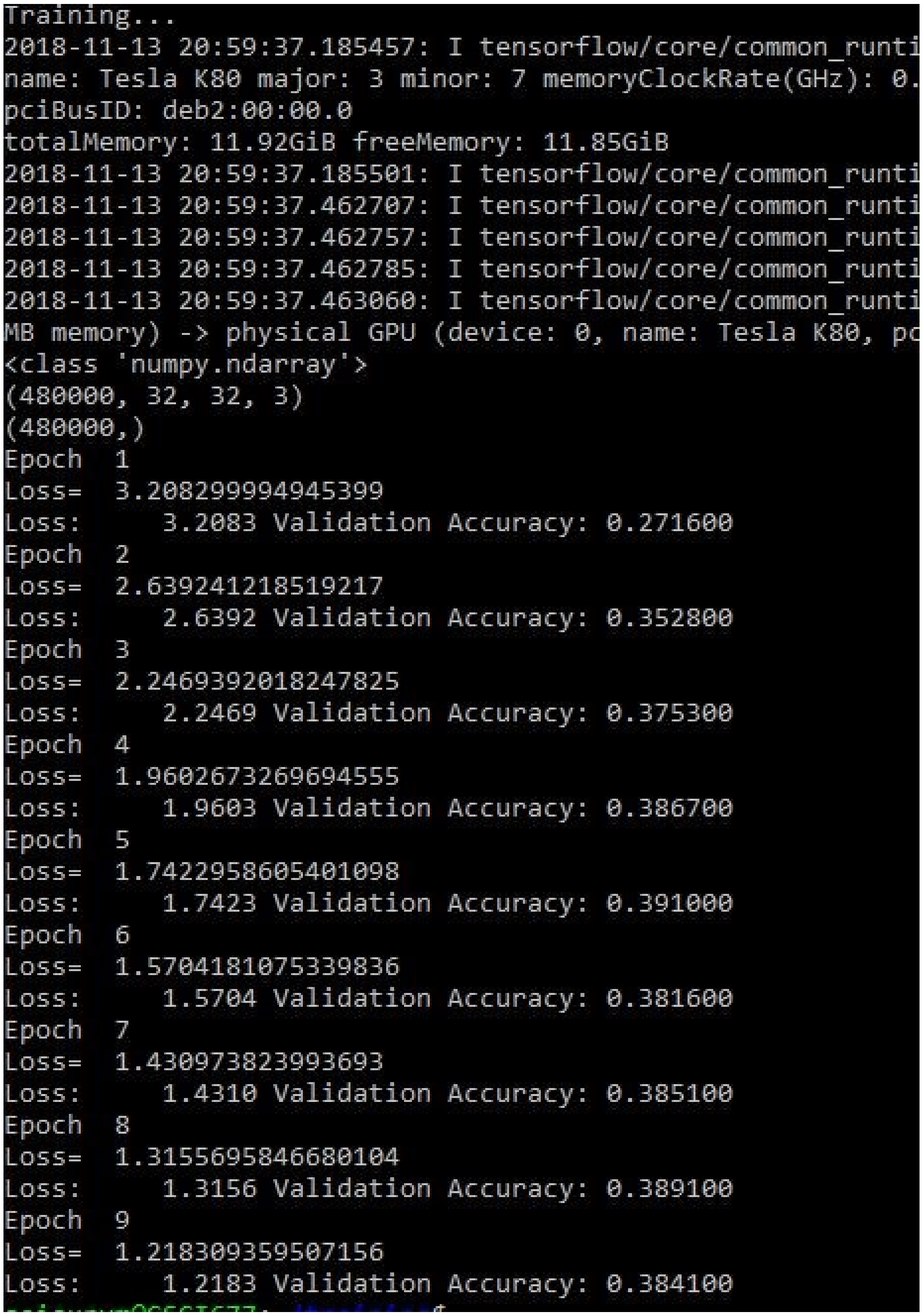
```
#epochs, batchSize,Learning rate
epochs = 2
batch_size = 64
learning_rate = 0.001

# Output of model
logits = leNet(x)
model = tf.identity(logits, name='logits')

#Loss function & Optimization Algorithm
cost = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logits, labels=y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)


#Prediction and Accuracy
correct_pred = tf.equal(tf.argmax(logits, 1),tf.cast(y,tf.int64))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32), name='accuracy')
```

In [ ]:

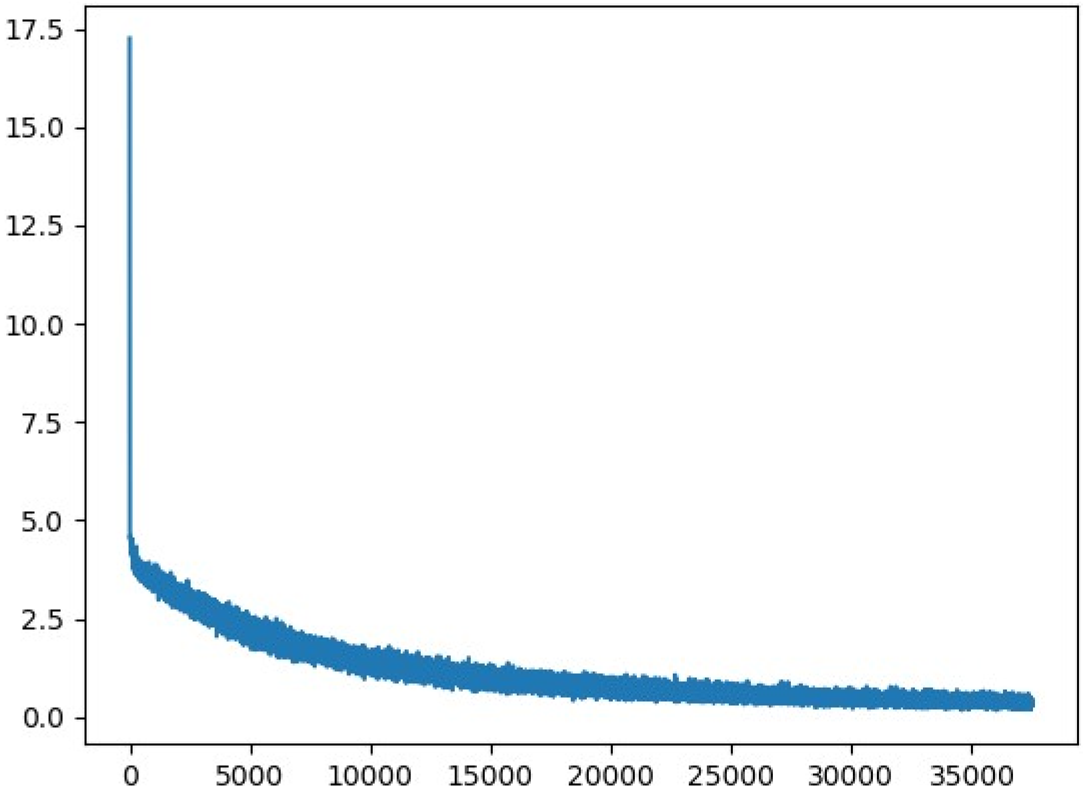
```
# Shuffling Data
def batch_features_labels(features, labels, batch_size):

    rand_index=np.random.choice(len(features),size=len(features))
    for start in range(0,len(features),batch_size):
        end=min(start+batch_size,len(features))
        tmp=np.array(rand_index[start:end])
        yield features[tmp],labels[tmp]
```

In [ ]:

```
#saving model
model_path = './image_classification'


#Creating Session
print('Training...')
with tf.Session() as sess:
    #inistalizing Global_variables
    sess.run(tf.global_variables_initializer())
    ###########

    # Data Augmentation
    data_tf=tf.convert_to_tensor(Xtrain,np.float32)
    big=tf.image.resize_images(Xtrain,(36,36))
    top_r=tf.image.crop_to_bounding_box(big,0,0,32,32)
    top_l=tf.image.crop_to_bounding_box(big,0,4,32,32)
    bot_r=tf.image.crop_to_bounding_box(big,4,0,32,32)
    bot_l=tf.image.crop_to_bounding_box(big, 4, 4, 32, 32)
    cen=tf.image.crop_to_bounding_box(big,2,2,32,32)
    flip=tf.image.flip_left_right(data_tf)

    fbig = tf.image.resize_images(flip, (36, 36))
    ftop_r = tf.image.crop_to_bounding_box(fbig, 0, 0, 32, 32)
    ftop_l = tf.image.crop_to_bounding_box(fbig, 0, 4, 32, 32)
    fbot_r = tf.image.crop_to_bounding_box(fbig, 4, 0, 32, 32)
    fbot_l = tf.image.crop_to_bounding_box(fbig, 4, 4, 32, 32)
    fcen = tf.image.crop_to_bounding_box(fbig, 2, 2, 32, 32)


    Xtrain1 = tf.concat([data_tf, top_r, top_l, bot_l, bot_r, cen, flip], axis
=0)
    Xtrain2=tf.concat([ftop_l,ftop_r,fbot_l,fbot_r,fcen],axis=0)

    # Xtrain=tf.concat([Xtrain1,Xtrain2],axis=0)

    sess.run(Xtrain1)
    sess.run(Xtrain2)
    Xtrain1=Xtrain1.eval()
    Xtrain2=Xtrain2.eval()
    Xtrain=np.concatenate((Xtrain1,Xtrain2))
    ytrain=np.concatenate((ytrain,ytrain,ytrain,ytrain,ytrain,ytrain,ytrain,yt
rain,ytrain,ytrain,ytrain,ytrain))
    # Xtrain=Xtrain.eval()
    print(type(Xtrain))
    print(Xtrain.shape)
    print(ytrain.shape)

    #Calculating mean of the dataSet
    sub = np.mean(Xtrain, axis=0)
    Xtrain = Xtrain - sub
    Xval=Xval-sub
    np.save("mean_vec",sub)

    # Traning and Validation
    for epoch in range(0,epochs):
        for batch_features, batch_labels in batch_features_labels(Xtrain, ytra
in, batch_size):
            sess.run(optimizer,
                            feed_dict={
                                x: batch_features,
                                y: batch_labels
                            })

            loss = sess.run(cost,
                            feed_dict={
                                x: batch_features,
                                y: batch_labels
                            })
            loss_graph_list.append(loss)

            if(epoch==epc):
                epc+=1
                print('Epoch {:>2}\n'.format(epoch), end='')
                print(sum(loss_graph_list)/len(loss_graph_list))

            if(epoch==count):
                count+=1
                valid_acc = sess.run(accuracy,
                                     feed_dict={
                                         x: Xval,
                                         y: yval})

                print('Loss: {:>10.4f} Validation Accuracy: {:.6f}'.format(los
s, valid_acc))


    # Saving model and plotting Loss Graph
    plt.plot(loss_graph_list)
    plt.show()
    plt.savefig("myfig")
    model_saver = tf.train.Saver()
    save_path_model = model_saver.save(sess, model_path)
```



```
Training...
2018-11-13 20:59:37.185457: I tensorflow/core/common_runti
name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0.
pciBusID: deb2:00:00.0
totalMemory: 11.92GiB freeMemory: 11.85GiB
2018-11-13 20:59:37.185501: I tensorflow/core/common_runti
2018-11-13 20:59:37.462707: I tensorflow/core/common_runti
2018-11-13 20:59:37.462757: I tensorflow/core/common_runti
2018-11-13 20:59:37.462785: I tensorflow/core/common_runti
2018-11-13 20:59:37.463060: I tensorflow/core/common_runti
MB memory) -> physical GPU (device: 0, name: Tesla K80, pc
<class 'numpy.ndarray'>
(480000, 32, 32, 3)
(480000,)
Epoch  1
Loss=  3.208299994945399
Loss:     3.2083 Validation Accuracy: 0.271600
Epoch  2
Loss=  2.639241218519217
Loss:     2.6392 Validation Accuracy: 0.352800
Epoch  3
Loss=  2.2469392018247825
Loss:     2.2469 Validation Accuracy: 0.375300
Epoch  4
Loss=  1.9602673269694555
Loss:     1.9603 Validation Accuracy: 0.386700
Epoch  5
Loss=  1.7422958605401098
Loss:     1.7423 Validation Accuracy: 0.391000
Epoch  6
Loss=  1.5704181075339836
Loss:     1.5704 Validation Accuracy: 0.381600
Epoch  7
Loss=  1.430973823993693
Loss:     1.4310 Validation Accuracy: 0.385100
Epoch  8
Loss=  1.3155695846680104
Loss:     1.3156 Validation Accuracy: 0.389100
Epoch  9
Loss=  1.218309359507156
Loss:     1.2183 Validation Accuracy: 0.384100
```


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
5000
10000
15000
20000
25000
30000
35000

Name: Sai Surya Salalith Mantha

USC_ID: 9463553709

Email_ID: Saisurym@usc.edu

In [1]:

```
import tensorflow as tf
import pickle
import random
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
np.set_printoptions(threshold=np.nan)
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWa
rning: Conversion of the second argument of issubdtype from `float` to `np.fl
oating` is deprecated. In future, it will be treated as `np.float64 == np.dty
pe(float).type`.
  from ._conv import register_converters as _register_converters
```

In [2]:

```
model_path = './image_classification'
batch_size = 64
n_samples = 10
top_n_predictions = 5


def unpickle(file):

    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

train=unpickle("cifar-100-python/train")
test=unpickle("cifar-100-python/test")
meta=unpickle("cifar-100-python/meta")

Xtest=test[b'data'].reshape((len(test[b'data']), 3, 32, 32)).transpose(0, 2, 3
, 1)
ytest=np.array(test[b'fine_labels'])

Xtest=Xtest.astype(np.float32)


sub=np.load("mean_vec.npy")
Xtest=Xtest-sub

ytestC=np.array(test[b'coarse_labels'])

lab={}
for i in range(0,20):
    lab[i]=set()
for i in range(0,len(ytest)):
    lab[ytestC[i]].add(ytest[i])


name_100_labels=[str(i)[2:-1] for i in meta[b'fine_label_names']]
name_20_labels=[str(i)[2:-1] for i in meta[b'coarse_label_names']]
```

In [3]:

```
def batch_features_labels(features, labels, batch_size):
    for start in range(0, len(features), batch_size):
        end = min(start + batch_size, len(features))
        yield features[start:end], labels[start:end]
```

In [4]:

```
loaded_graph = tf.Graph()
with tf.Session(graph=loaded_graph) as sess:

    loader = tf.train.import_meta_graph(model_path + '.meta')
    loader.restore(sess, model_path)
    loaded_x = loaded_graph.get_tensor_by_name('input_x:0')
    loaded_y = loaded_graph.get_tensor_by_name('output_y:0')
    loaded_logits = loaded_graph.get_tensor_by_name('logits:0')
    loaded_acc = loaded_graph.get_tensor_by_name('accuracy:0')

    label_predictor = sess.run(
            tf.nn.top_k(tf.nn.softmax(loaded_logits),1),
            feed_dict={loaded_x: Xtest, loaded_y: ytest})
#         print(label_predictor[1])

    matFine=confusion_matrix(ytest,label_predictor[1])
    accFine=accuracy_score(ytest,label_predictor[1])

    print("---------------------------100 classes------------------")




    print("\nAccuracy and confusion matrix for 100 labels")
    print("----------------------------------Accuracy=",100*accFine,"---------
---------------")
    print("\nConfusion-Matrix:\n",matFine)




#     print("\nPredicted_labels:\n",label_predictor[1].reshape(len(label_predi
ctor[1])).tolist())
#     print("\n Correct_labels:\n",ytest)
    print("\n\n\n\n\n")



    coarse_pred=[]
    pred=np.reshape(label_predictor[1],(len(label_predictor[1]))).tolist()
    for i in range(0,len(pred)):
        for pos in lab.keys():
            if(pred[i] in lab[pos]):
                coarse_pred.append(pos)
    matCoa = confusion_matrix(ytestC,coarse_pred)
    accCoa = accuracy_score(ytestC, coarse_pred)
    print("---------------------------20  classes------------------")
    print("Accuracy and confusion matrix for 20 labels")
    print("----------------------------------Accuracy=",100*accCoa,"----------
--------------")
    print("Confusion-Matrix:\n",matCoa)



#     random_test_features, random_test_labels = tuple(zip(*random.sample(list
(zip(Xtest, ytest)), n_samples)))
```

```
    Top_3_pred= sess.run(
        tf.nn.top_k(tf.nn.softmax(loaded_logits), 1),
        feed_dict={loaded_x: Xtest, loaded_y: ytest})
    print("---------------------------------------Considering Top-3----------
-----------------\n")
#     print("Top 5 Classes:\n"Top_5_pred[1])
    acc_top_3=0
    for i in range(0,len(Top_3_pred[1])):
        if(ytest[i] in Top_3_pred[1][i]):
            acc_top_3+=1
    print("------------------------Accuracy for 100 classes= ",100*acc_top_3/l
en(ytest))

    cacc_top_3=0
    for i in range(0,len(ytest)):
        temp=[]
        for j in lab.keys():
            if(ytest[i] in lab[j]):
                temp.append(lab[j])
                break
#         print(temp)
#         print(Top_5_pred[1][i])
        if(len(temp)!=0):
            temp2=set(temp[0]).intersection(Top_3_pred[1][i])
#             print(temp2)
            if(len(temp2)!=0):
                cacc_top_3+=1
    print("------------------------Accuracy for 20 classes= ",100*cacc_top_3/l
en(ytest))


    Top_5_pred= sess.run(
        tf.nn.top_k(tf.nn.softmax(loaded_logits), top_n_predictions),
        feed_dict={loaded_x: Xtest, loaded_y: ytest})
    print("---------------------------------------Considering Top-5----------
-----------------\n")
#     print("Top 5 Classes:\n"Top_5_pred[1])
    acc_top_5=0
    for i in range(0,len(Top_5_pred[1])):
        if(ytest[i] in Top_5_pred[1][i]):
            acc_top_5+=1
    print("------------------------Accuracy for 100 classes= ",100*acc_top_5/l
en(ytest))

    cacc_top_5=0
    for i in range(0,len(ytest)):
        temp=[]
        for j in lab.keys():
            if(ytest[i] in lab[j]):
                temp.append(list(lab[j]))
                break
        if(len(temp)!=0):
            temp2=set(temp[0]).intersection(Top_5_pred[1][i])
            if(len(temp2)!=0):
                cacc_top_5+=1
    print("------------------------Accuracy for 20 classes= ",100*cacc_top_5/l
en(ytest))
```

```
INFO:tensorflow:Restoring parameters from ./image_classification
---------------------------100 classes------------------

Accuracy and confusion matrix for 100 labels
----------------------------------Accuracy= 39.08 ------------------------

Confusion-Matrix:
 [[57  2  0  0  0  0  0  0  0  1  3  0  0  0  2  1  0  0  0  0  0  0  0  0
   0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0
   0  0  0  0  0  3  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  5  0
   0  0  0  0  0  0  2  0  0  0  0  9  0  0  0  0  0  0  0  0  4  0  0  0
   0  0  0  0]
 [ 0 48  1  0  0  0  0  1  2  0  1  0  0  0  1  0  1  0  1  0  0  0  0  0
   0  0  3  0  0  0  0  0  3  1  2  0  0  0  0  0  0  1  0  0  3  0  1  0
   1  0  0  2  0  1  3  0  0  0  0  0  0  0  0  0  0  0  0  2  0  1  1  0
   0  1  0  0  0  0  3  0  1  1  1  1  0  0  2  0  0  0  0  2  3  4  0  0
   0  0  0  0]
 [ 1  1 20  1  0  0  1  0  2  0  0  7  0  0  0  0  1  0  0  0  0  2  0  0
   0  0  5  0  1  0  0  0  2  0  1 15  3  1  0  1  0  1  0  0  0  3  4  0
   0  0  0  0  0  0  3  0  0  1  1  0  0  2  0  0  1  1  0  0  0  0  2  0
   0  0  1  0  0  0  1  0  2  0  0  1  3  0  0  0  0  0  1  1  1  0  1  0
   0  2  2  0]
 [ 0  0  0 25  7  0  3  0  0  0  0  0  0  0  1  3  0  0  1  2  0  8  0  1
   0  1  0  0  0  1  0  2  2  0  1  0  0  0  1  0  1  1  1  3  0  0  0  0
   1  1  0  1  0  0  1  3  0  0  0  0  0  0  0  3  0  0  2  1  0  0  0  0
   5  0  4  2  0  0  1  0  2  1  0  0  0  0  1  1  1  1  0  0  0  0  0  1
   0  1  1  0]
 [ 0  1  0  3 22  0  2  2  0  0  0  0  0  1  0  1  0  0  0  1  0  1  0  0
   0  1  2  0  0  0  0  0  1  2  2  1  0  0  0  0  0  0  5  2  2  0  0  0
   1  0  4  1  0  0  0  4  0  0  0  0  0  0  0  3  2  0  2  1  0  0  0  0
   2  0 12  0  0  3  3  2  0  0  0  0  1  0  1  1  3  0  1  0  0  1  0  0
   0  0  0  0]
 [ 0  0  1  0  1 20  0  0  0  0  3  3  2  3  0  1  4  0  0  0  8  1  1  0
   0  8  2  0  0  0  0  0  1  1  1  1  1  2  1  1  0  0  1  0  0  1  1  0
   0  0  2  0  0  0  0  0  1  0  2  0  0  0  0  0  0  1  0  0  0  1  0  0
   0  0  0  0  0  1  1  1  0  0  0  0  4  1  2  4  0  2  1  0  1  0  2  0
   1  0  0  2]
 [ 0  1  0  0  1  0 53  3  0  0  0  0  0  0  6  1  0  0  0  0  0  0  0  0
   1  0  5  1  0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  0  1  0  0  0
   0  0  1  1  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  1  1  0  0
   0  0  0  0  0  1  1  3  1  0  1  3  0  0  0  0  5  0  1  0  1  0  0  0
   0  0  0  0]
 [ 0  1  1  0  1  0  2 52  0  0  2  0  0  0  2  1  0  0  2  0  0  0  0  1
   8  0  3  1  0  1  0  0  0  0  0  1  0  0  1  0  0  1  0  0  0  1  0  0
   0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  2  1  0  0  0
   0  1  1  1  0  0  1  2  0  1  0  2  1  0  0  0  1  0  0  0  1  0  0  1
   0  0  0  1]
 [ 0  0  1  0  0  0  0  1 41  0  0  1  2  0  1  0  1  0  0  0  0  0  0  0
   0  0  0  0  0  2  1  1  0  0  0  0  0  2  1  1  0  2  1  0  0  0  2  0
  15  0  0  0  0  0  1  0  0  0  0  2  0  0  0  0  1  0  0  0  0  0  0  0
   2  0  0  0  0  0  1  0  0  1  0  0  2  3  1  0  1  5  1  0  1  0  2  0
   0  0  0  0]
 [ 1  0  1  0  0  1  1  0  0 48  0  2  1  0  1  0  1  0  0  0  0  1  0  0
   0  0  0  0  1  0  0  0  0  0  1  1  0  1  0  1  3  1  0  0  1  0  2  0
   0  0  0  2  0  1  1  1  0  0  2  1  0  0  0  0  0  0  1  0  0  1  0  0
   0  0  0  1  3  0  0  0  0  1  1  2  1  0  4  1  0  2  1  0  0  0  2  0
   0  1  0  0]
```

```
[ 2  0  1  0  1  2  0  2  0  0 25  1  0  0  1  0  4  0  0  0  0  0  6  0
  0  1  3  0  6  0  0  1  2  0  0  2  1  0  0  0  1  1  0  0  0  0  1  0
  0  0  1  0  0  0  0  0  0  0  0  1  7  1  0  0  0  0  0  0  0  0  1  0
  0  1  2  1  0  2  5  0  0  0  1  2  2  0  3  1  0  0  0  3  1  0  0  0
  0  0  0  1]
[ 0  3  8  1  0  0  0  0  0  1  0 26  1  1  0  0  0  0  0  0  1  3  0  0
  0  0  0  0  0  1  0  0  2  0  1  9  0  0  0  1  1  3  0  2  0  0 12  0
  0  0  0  1  0  0  1  1  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  3  0  0  3  0  0  0  2  0  0  0  1  0  0  0
  1  1  4  2]
[ 0  0  0  0  0  0  0  0  0  1  0  0 45  3  1  0  0  3  1  1  1  0  0  0
  0  1  0  2  0  0  0  0  0  0  0  1  0  8  1  0  0  0  0  0  0  0  0  0
  1  1  0  0  1  0  0  0  2  0  2  0  0  0  0  0  0  0  0  0  2  2  0  0
  0  0  0  0  1  0  0  0  1  8  0  0  0  3  1  0  0  1  3  1  0  0  0  0
  0  0  0  1]
[ 0  0  0  0  0  0  1  0  1  0  1  0  2 29  0  0  1  1  1  0  0  0  0  0
  1  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0
  2  0  0  0  0  0  0  0  0 14  0  1  0  0  0  0  0  1  0  1  0  0  0  0
  0  2  0  0  2  0  0  0  0 19  1  0  1  0  2  1  0  2  6  0  1  0  1  0
  0  0  0  0]
[ 0  4  1  0  0  1  4  1  1  0  0  0  0  0 42  0  1  0  2  0  1  1  1  0
  0  0  3  0  2  0  0  0  0  1  1  0  0  0  0  0  0  0  0  1  0  0  0  0
  0  0  1  0  0  0  1  0  0  0  0  0  0  1  2  2  1  0  0  0  0  0  0  0
  1  0  2  1  0  3  3  2  0  0  1  3  0  0  0  0  3  1  1  0  2  0  0  0
  0  0  0  1]
[ 0  1  0  0  5  0  1  1  2  0  1  0  0  0  3 38  1  2  0  3  0  0  0  0
  1  1  2  0  0  2  0  2  0  0  3  1  0  2  4  0  1  0  0  3  1  0  0  1
  0  0  0  1  0  1  0  1  0  0  0  0  0  0  0  1  1  0  3  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  2  4  0  0  0  0  0  0  0
  0  1  1  0]
[ 0  0  0  0  1  1  0  0  1  2  4  0  0  3  0  1 40  0  1  1  0  0  5  0
  0  0  0  0  3  1  0  0  0  0  0  3  0  0  0  1  1  0  0  1  0  0  1  0
  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0
  1  0  0  0  0  0  1  0  0  5  1  2  1  1  3  3  0  0  2  1  0  0  1  0
  0  0  2  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  2  1 47  0  0  0  0  0  0
  0  1  0  0  0  0  1  1  0  1  0  0  0 18  0  1  0  0  0  0  0  0  1  0
  1  2  0  0  0  0  0  0  3  0  0  1  2  0  0  0  0  0  0  0  1  0  0  0
  0  0  0  0  4  0  0  0  0  0  0  0  0  2  1  0  1  3  1  0  0  0  0  0
  1  0  0  0]
[ 1  1  1  0  1  0  4  5  1  2  0  0  0  0  3  1  0  0 32  0  0  1  0  0
  0  0  0  4  1  1  0  0  0  2  1  1  0  0  0  1  1  2  0  0  3  1  0  1
  0  0  1  0  0  0  1  1  1  0  0  1  0  1  0  0  0  0  1  0  0  0  1  0
  1  0  2  1  0  3  2  2  3  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
  1  1  0  2]
[ 0  0  0  1  1  0  0  0  2  0  1  1  0  1  4 10  0  0  2 26  1  2  0  0
  0  0  3  1  0  1  0  3  1  1  2  0  0  2  4  0  0  0  0  1  0  0  1  0
  1  0  0  2  1  0  0  1  0  0  1  0  0  0  0  2  0  0  2  0  0  0  0  1
  1  1  0  1  0  0  1  1  0  1  0  0  2  0  0  0  3  1  0  0  0  0  0  0
  0  3  2  0]
[ 1  0  0  0  0  0  1  1  0  0  0  0  0  1  1  4  3  0  0  0 63  0  1  0
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1  0  1  0  0  0  1  0  0
  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  1  0  0
  0  0  0  0  0  0  1  0  0  0  0  0  3  0  2  3  1  1  0  2  0  0  0  0
  0  1  1  0]
[ 0  1  1  8  2  0  1  2  0  0  1  0  1  0  2  1  1  0  1  1  0 51  0  0
  0  0  0  0  0  1  0  0  2  0  0  1  0  0  0  0  0  0  1  1  0  0  0  0
```

```
   1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  3  0  3  0  0  0  0  0
   2  0  2  0  0  0  0  0  0  1  0  0  0  0  2  1  2  0  0  0  0  0  0  0
   0  1  1  0]
 [ 0  1  1  0  1  0  0  2  1  0  5  1  0  0  0  1  1  0  1  0  1  1 36  0
   0  0  2  1  1  0  0  0  1  0  0  1  0  0  0  2  1  0  0  1  0  0  1  0
   0  0  1  0  0  0  0  0  0  1  3  0  0  5  0  0  0  1  2  0  0  0  0  0
   0  0  0  0  0  1  4  0  0  0  0  1  2  0  5  1  2  1  0  2  1  0  0  0
   0  1  0  2]
 [ 0  0  0  0  0  0  0  0  0  0  2  0  1  0  1  0  0  0  0  0  0  1  0 45
   0  1  1  1  0  1  2  0  1  0  1  0  0  1  0  5  1  0  0  1  0  0  1  0
   0  8  0  0  0  0  0  0  0  0  1  0  4  0  0  0  1  1  0  2  0  2  0  5
   3  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  1  0  0
   1  0  0  1]
 [ 0  0  0  2  1  0  2  6  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  53  0  3  0  0  0  0  0  0  0  0  3  0  0  0  0  1  2  0  0  1  0  0  0
   0  0  3  1  0  0  0  2  1  1  0  0  0  0  1  0  0  0  0  0  0  1  0  0
   0  0  1  2  1  2  0  3  2  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0
   0  0  0  1]
 [ 0  0  2  1  0  6  0  0  1  1  4  2  2  2  0  1  2  0  0  0  2  0  1  0
   0 27  2  0  0  0  0  0  0  4  0  0  1  1  0  1  4  0  0  0  0  0  0  0
   1  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  1  0  1  0  0  2  1
   0  0  0  1  0  0  1  0  1  4  0  1  1  0  2  6  0  2  0  0  2  1  1  0
   1  0  1  0]
 [ 0  1  0  0  1  1  2  1  1  0  2  1  0  0  1  0  3  0  0  0  0  0  2  0
   1  0 41  0  0  2  0  0  0  0  1  2  2  1  0  0  0  0  1  0  2  2  0  0
   0  0  0  2  0  0  0  1  1  0  0  0  0  1  0  0  1  0  1  0  0  0  0  0
   2  0  3  0  0  0  5  3  0  0  0  2  0  0  0  0  0  1  0  2  1  1  0  0
   1  0  1  0]
 [ 0  0  2  0  2  0  0  0  0  0  0  0  0  0  0  2  0  0  2  0  0  0  0  2
   0  3  1 17  0  1  1  1  0  6  1  0  0  2  3  1  1  0  1  0  4  0  1  0
   0  0  1  1  0  0  0  3  0  0  0  1  0  0  0  2  0  1  0  0  1  0  0  1
   1  2  5  2  0  1  3  1  1  1  1  0  0  2  0  0  3  1  1  4  0  0  0  2
   1  0  1  2]
 [ 0  0  1  1  0  1  0  0  0  1  2  1  0  1  0  1  2  0  2  2  0  0  0  1
   0  1  0  0 52  0  0  0  0  0  1  1  0  0  0  0  6  0  0  0  1  1  0  0
   0  0  0  0  0  2  0  1  0  0  0  1  0  1  2  0  0  0  0  0  0  2  0  0
   0  0  0  0  0  1  0  0  0  0  0  0  1  0  5  1  0  0  0  0  0  1  0  1
   0  0  0  2]
 [ 0  2  0  2  2  0  3  0  3  0  0  2  0  0  0  6  1  0  0  0  0  0  0  0
   0  0  1  1  0 36  0  0  0  0  2  0  0  4  1  0  1  1  1  1  4  1  1  1
   1  0  0  2  0  0  0  1  1  0  0  0  0  0  0  1  0  0  1  0  0  1  0  0
   2  0  0  0  0  0  4  1  1  0  0  0  0  0  0  0  1  0  0  3  0  0  0  1
   0  0  1  1]
 [ 0  1  2  0  1  0  0  0  1  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  1  0  0 37  0  1  0  0  1  1  0  0  1  0  0  0  0  0  1  0  0
   0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  3  0  4  0  0
   4 10  0  1  1  1  1  0  0  2  0  0  0  0  0  0  0  0  0  1  0  2  0 15
   0  1  0  1]
 [ 0  0  0  1  0  1  0  1  0  0  0  2  0  0  0  8  0  1  1  4  0  3  0  1
   0  0  0  1  0  2  0 32  0  4  0  1  0  2  2  0  0  0  1  0  0  1  2  0
   2  0  0  0  0  0  0  2  1  0  0  2  0  0  0  0  2  0  1  0  0  0  0  0
   3  0  2  1  0  0  1  0  2  1  1  0  0  0  2  0  0  3  0  0  0  0  0  0
   0  3  0  0]
 [ 0  2  0  0  2  0  0  0  0  0  3  2  0  0  0  1  0  0  1  0  0  2  1  0
   0  1  2  2  0  1  0  0 33  0  0  1  1  1  1  0  0  0  3  0  0  0  2  0
   0  0  3  0  0  0  0  2  0  2  0  0  0  1  0  0  0  0  1  2  0  0  0  0
   3  0  1  0  0  0  1  0  2  2  0  0  0  1  3  2  2  0  2  3  1  2  0  0
   2  0  0  0]
 [ 0  0  0  0  1  0  3  1  0  1  0  0  0  0  0  1  0  1  1  0  0  0  0  1
   0  0  0  1  0  0  0  0  0 36  0  0  0  1  1  0  0  0  0  0  0  0  0  6
   1  4  0  2  0  0  0  0  1  0  1  5  2  0  1  1  2  1  1  0  0  0  0  0
   0  0  1  0  0  0  0  0  0  3  1  0  1  0  0  1  1  3  2  0  1  0  1  1
   7  0  0  0]
 [ 0  3  0  1  2  0  6  1  0  0  0  0  0  0  1  2  0  0  0  2  0  0  0  0
   0  1  3  2  0  3  0  1  1  0 26  2  1  0  5  0  0  0  4  4  1  1  2  0
   0  0  2  3  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0
   0  0  3  0  0  3  0  0  3  0  0  0  0  0  1  0  4  1  0  0  1  1  0  0
   0  1  0  0]
 [ 0  0  2  0  0  0  1  1  1  0  0  9  0  0  2  1  1  0  0  1  0  1  1  0
   2  0  1  0  0  0  0  0  0  0  0 35  0  0  0  2  1  0  1  1  0  1  5  0
   1  0  2  0  0  1  1  0  0  0  1  0  0  1  0  0  1  1  1  0  0  0  0  0
   0  0  1  0  0  0  0  1  0  0  0  0  2  1  0  0  0  0  0  0  1  1  0  1
   0  4  8  0]
 [ 0  2  2  1  3  0  2  0  0  0  2  3  0  0  1  3  0  0  0  0  0  0  1  1
   0  0  2  0  0  0  1  0  1  0  3  4 30  0  1  0  1  0  1  2  0  0  4  0
   0  0  2  1  0  0  0  0  0  1  0  0  0  0  0  1  3  8  0  0  0  0  0  0
   0  0  2  0  0  2  0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  2  0  0
   0  1  2  0]
 [ 0  0  0  1  0  1  0  0  0  0  0  0  1  2  0  0  4  5  0  1  1  0  1  0
   0  1  0  0  0  2  0  0  0  0  0  1  0 43  0  3  0  0  0  0  0  1  0  0
   0  1  0  0  0  0  0  0  2  0  0  4  1  0  0  0  0  0  0  0  1  0  0  0
   0  0  0  0  1  0  1  0  0 10  0  0  1  3  0  0  1  2  1  0  0  0  0  0
   1  1  1  0]
 [ 0  0  0  4  2  0  1  1  0  0  0  0  0  0  2  3  0  2  2  1  0  0  0  1
   0  0  1  4  1  0  0  1  0  3  3  0  0  0 18  0  0  1  5  1  4  0  0  0
   1  0  1  5  0  1  0  4  0  0  0  1  0  0  0  2  1  0  1  0  0  1  0  0
   2  0  1  1  0  1  2  3  1  0  0  0  0  2  0  0  3  1  1  0  0  0  0  0
   0  1  1  1]
 [ 0  0  0  0  0  1  2  0  1  1  1  0  1  0  1  0  1  0  0  0  0  1  0  3
   0  1  0  2  0  0  1  0  0  2  0  0  0  0  0 59  0  0  0  0  0  0  0  0
   0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  1
   0  0  0  0  1  0  4  0  0  5  0  1  0  1  2  0  1  0  0  0  0  1  1  0
   0  0  0  1]
 [ 0  0  0  1  0  0  0  2  0  1  2  0  3  0  0  2  2  0  1  1  1  0  0  0
   0  2  0  0  2  1  0  0  0  0  1  0  0  0  0  0 39  1  0  0  0  0  5  1
   0  0  0  0  0  0  1  0  1  2  0  0  0  0  0  0  0  1  0  0  0  3  1  0
   0  0  0  1  1  0  2  2  0  1  0  1  3  2  1  2  0  0  0  1  0  1  1  0
   0  0  0  4]
 [ 0  3  1  0  0  1  0  4  0  2  0  1  0  1  0  0  0  0  1  0  1  0  1  0
   0  0  1  2  0  0  0  0  0  0  0  0  0  1  2  0  0 56  1  0  0  1  0  0
   7  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  1  0  0  0  0  0  3  0  0  0  2  0  1  1  2  0  0  1  0  0  0
   0  0  0  0]
 [ 0  0  0  0  3  0  1  0  0  0  0  0  0  0  0  1  0  0  2  1  0  0  1  1
   0  0  1  1  0  0  0  0  0  1  5  1  0  0  1  1  0  0 36  3  1  0  0  0
   0  0  2  1  0  0  0  2  0  0  0  0  0  0  0  1  4  0  5  1  0  0  0  0
   0  0  2  0  0  0  3  1  1  1  0  0  1  0  0  0 10  1  1  0  0  0  0  0
   1  1  0  0]
 [ 0  0  1  2  4  0  3  1  0  0  0  0  0  0  1  3  0  0  0  3  0  0  0  0
   0  0  1  0  0  0  0  0  0  0  3  1  1  0  0  0  0  0  7 40  0  0  1  0
   0  0  1  9  0  1  0  0  0  1  0  0  0  0  0  2  0  1  1  0  0  0  0  0
   0  0  1  0  0  2  1  0  1  0  1  0  0  0  1  0  2  2  0  0  0  0  0  0
   0  1  0  0]
 [ 0  1  0  0  2  0  2  3  3  0  1  2  0  0  1  2  0  0  3  0  0  0  0  0
```

```
  2  0  2  7  0  2  1  0  0  2  0  1  0  0  2  3  0  1  3  0 15  1  0  0
  0  0  1  0  0  0  0  2  0  0  0  1  0  0  0  0  0  0  0  0  1  1  0  0
  0  0  2  0  0  2  9  1  0  0  0  1  0  0  1  1  1  1  2  0  0  2  0  1
  0  1  0  7]
 [ 0  1  0  0  1  1  4  2  0  0  1  0  0  0  3  0  0  0  3  0  0  0  0  1
  2  1  9  1  0  2  0  1  1  0  1  0  0  0  0  2  0  3  0  0  9 18  0  0
  2  0  1  2  1  0  1  0  1  0  0  1  0  0  1  0  1  0  0  1  0  0  2  1
  0  0  3  0  0  2  1  2  1  1  0  3  0  0  0  0  1  1  0  0  0  1  0  0
  0  0  1  1]
 [ 1  0  3  0  1  1  2  1  0  1  1  7  1  0  0  1  1  0  0  1  0  1  1  0
  0  1  2  0  2  0  0  0  1  0  2  7  2  0  0  0  0  0  1  3  0  1 21  0
  2  0  0  0  0  0  0  1  0  0  2  1  0  0  0  1  2  2  0  0  0  0  1  0
  1  0  0  1  0  1  2  1  0  1  0  0  0  0  3  2  0  0  2  0  0  0  0  1
  1  0  3  2]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  2  0  0  1  0  0  1  2  0  0  0  2  0  0  0  0  1  0  0  0  0 48
  0  0  0  0 17  0  0  0  0  0  0 10  0  0  0  0  0  0  1  0  0  0  1  0
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  1  0  0  1  0  0  0
  9  1  0  0]
 [ 0  1  0  0  0  0  1  2  1  0  1  2  1  1  2  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0  2  0  0  0
 72  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  1  0
  0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  0  0  2  1  0  1  1  0  0
  0  0  0  0]
 [ 0  1  0  1  2  0  0  1  1  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
  0  1  1  0  0  1  1  0  0  1  0  1  0  3  0  1  0  0  0  0  0  0  0  0
  0 59  0  1  0  0  0  0  0  1  2  1  2  0  0  0  0  0  1  0  0  0  0  3
  1  0  0  0  2  0  0  0  1  1  0  0  0  0  0  0  0  0  4  0  0  1  0  1
  0  0  0  1]
 [ 0  0  0  1  2  0  1  0  1  0  1  3  0  0  1  1  0  1  1  0  0  1  0  0
  0  0  1  1  1  0  0  0  2  0  0  0  2  0  3  1  0  0  1  0  3  2  0  0
  0  1 21  0  0  0  2  2  0  1  0  0  0  1  1  3  8  2  3  0  0  0  0  0
  1  0  5  0  0  3  5  1  0  0  0  0  0  1  1  0  2  0  0  2  0  1  0  0
  0  2  0  0]
 [ 0  4  0  1  1  0  4  0  0  1  0  0  0  1  0  2  1  0  0  4  0  0  0  0
  0  0  4  1  0  0  0  0  0  0  2  1  1  0  1  0  0  0  1  2  2  1  2  0
  0  0  2 32  0  0  3  2  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0
  1  0  1  0  0  1  3  1  1  0  0  2  1  0  0  0  3  0  1  0  1  2  0  0
  0  2  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0
  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15
  0  0  0  0 49  0  0  0  2  0  0 26  0  0  0  1  0  0  0  0  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  3  0  0  0]
 [ 3  3  0  0  0  0  0  0  0  0  0  1  0  0  3  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
  0  0  0  0  0 65  1  0  0  3  0  0  0  0  4  0  0  0  0  0  0  0  3  0
  0  0  0  0  0  0  0  0  0  0  1  7  0  0  0  0  0  0  0  0  2  0  0  0
  0  0  1  0]
 [ 1  0  1  0  0  0  1  0  0  0  0  0  1  0  1  0  0  0  0  1  1  0  0  0
  0  0  2  0  1  0  0  0  0  0  0  4  0  0  0  0  1  0  0  0  1  0  1  0
  0  0  1  0  0  1 53  0  0  0  0  0  0  1  0  0  3  1  0  1  0  0  7  0
  0  0  1  0  0  0  3  0  0  1  1  2  0  0  0  0  0  0  0  0  5  1  0  0
  0  0  0  1]
 [ 0  0  0  7  7  1  0  1  0  0  0  0  0  0  0  2  0  0  2  0  0  1  1  0
  0  0  0  2  0  2  3  2  1  0  1  0  0  1  4  0  0  0  2  0  2  0  0  0
  0  1  0  4  0  0  0  9  1  1  0  1  0  0  0  3  0  3  0  0  0  2  0  0
  3  2  7  1  0  1  2  1  2  2  0  0  0  1  2  0  2  0  2  2  0  1  0  1
  0  1  0  0]
[ 0  0  0  0  0  0  0  0  1  0  0  0  1  1  0  1  0  0  1  0  0  0  1  0
  0  0  0  0  1  1  1  1  0  2  0  0  0  3  0  0  1  0  0  0  0  0  0  3
  0  0  0  1  2  0  2  0 60  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0
  1  0  0  0  0  0  0  0  0  2  0  0  0  0  0  1  1  2  2  0  0  0  0  0
  1  0  0  0]
[ 1  0  1  1  0  0  2  1  0  0  1  1  0  0  3  0  1  0  0  1  0  0  0  0
  2  0  0  0  0  2  1  0  0  0  2  3  0  0  1  0  0  0  0  0  1  0  1  1
  1  0  0  1  0  5  5  1  0 42  0  0  0  1  0  0  0  0  0  1  0  1  1  0
  0  0  0  0  0  2  3  0  0  0  2  4  1  0  0  0  0  0  0  0  0  1  0  0
  0  0  0  1]
[ 0  1  0  0  0  0  0  1  0  0  0  0  1  8  0  0  2  0  0  0  0  0  0  0
  0  0  1  0  0  0  0  0  0  1  0  0  0  3  0  0  0  2  0  0  0  2  0  0
  0  0  0  0  0  0  0  0  0  0 50  0  1  0  0  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  1  0  0  6  0  0  1  4  0  1  0  6  6  1  0  0  0  0
  0  0  0  0]
[ 0  0  0  0  0  0  1  0  0  0  0  0  1  1  0  1  0  3  2  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  1  0  0  0  2  0  1  0  0  3  1  0  0  1  3
  0  2  0  0 10  0  0  1  4  0  0 44  0  0  0  0  0  0  1  1  0  0  0  1
  1  0  0  0  1  0  0  0  0  0  1  0  0  4  0  0  0  0  1  0  0  2  0  1
  4  0  0  0]
[ 0  1  0  0  0  0  1  0  0  0  1  0  1  0  0  0  0  1  0  0  0  0  0  3
  0  0  0  0  0  0  0  0  0  4  0  0  0  3  0  0  0  0  0  0  0  0  0  0
  0  3  0  0  0  0  0  0  0  0  0  1 63  0  0  0  0  0  0  0  1  3  1  7
  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1  0  3  0  0  0  0  0  0
  0  0  0  0]
[ 0  0  1  0  0  0  1  0  0  0 13  0  0  0  0  0  1  0  0  0  0  0 15  0
  1  0  0  0  2  0  0  0  0  0  0  1  1  0  0  0  1  0  0  0  0  0  2  0
  0  0  0  0  0  0  1  1  0  0  0  0  0 53  0  1  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0
  0  0  0  1]
[ 0  0  0  0  0  0  1  0  0  2  2  0  0  0  3  0  1  0  0  0  0  0  0  0
  0  0  2  0  0  0  0  0  0  0  0  2  0  0  0  0  0  1  0  0  0  0  0  0
  0  0  0  0  0  7  2  0  0  2  0  0  0  0 35  0  0  0  0  0  0  0 12  0
  0  0  0  0  0  0  0  0  0  0  3  8  0  0  0  0  0  0  1  1 15  0  0  0
  0  0  0  0]
[ 0  1  0  5  7  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  2  0  0
  0  0  3  3  0  0  0  2  2  1  0  0  0  1  2  0  0  0  5  0  0  1  1  0
  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0 37  1  0  1  0  0  0  0  0
  2  0  7  0  0  4  0  0  3  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0
  0  0  1  1]
[ 0  1  1  0  2  0  0  0  0  0  0  0  0  0  3  3  0  0  0  3  0  0  0  0
  0  0  0  4  0  2  0  0  2  0  0  4  2  1  3  0  0  0  2  1  1  0  0  0
  1  0  4  0  0  0  0  4  0  0  2  1  0  0  0  4 13  4  3  0  0  0  0  0
  2  1  2  0  0  3  2  0  5  0  0  0  0  0  0  0  1  1  2  1  1  1  0  0
  0  7  0  0]
[ 1  0  1  2  3  0  1  0  0  2  0  1  1  0  1  1  0  1  1  0  0  2  2  1
  1  1  2  1  0  2  0  0  2  1  1  0  3  0  3  1  0  0  0  1  3  0  0  0
  1  0  1  3  0  0  0  2  0  0  0  0  0  1  1  2  5 19  3  1  0  0  0  0
  1  0  2  0  0  2  4  2  0  0  0  0  0  0  1  0  0  0  1  1  0  0  0  1
  0  1  0  2]
[ 0  1  0  2  2  0  1  1  1  0  1  0  0  0  0  0  0  0  0  2  0  1  1  0
  1  0  1  0  0  2  0  0  1  1  2  1  1  1  0  0  0  0  2  0  1  0  0  0
  1  0  1  0  0  0  0  0  0  0  0  2  0  0  0  2  2  0 39  0  0  0  0  0
  3  0  2  2  0  0  4  1  3  0  0  0  0  0  1  0  1  2  0  0  0  0  0  0
  0  6  0  1]
```

```
[ 0  2  1  1  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  1
  0  0  1  2  1  2  2  0  3  2  0  0  0  0  0  0  0  1  0  0  2  0  0  0
  0  3  1  1  0  0  0  0  0  0  0  2  0  0  0  3  1  1  2 29  0  1  0  1
  3  7  1  0  1  3  2  0  0  0  0  0  0  0  0  0  0  0  2  2  0  4  0  3
  0  0  0  3]
[ 0  0  0  2  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  1  0  0  0  0
  0  0  0  0  0  1  1  0  0  2  0  1  0  1  1  0  0  0  0  0  0  1  1  0
  1  2  0  0  0  0  0  0  0  0  2  1  2  1  0  0  0  0  0  0 67  0  0  4
  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0
  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  3  1  0  0  0  1  3
  0  1  0  1  0  0  1  0  0  0  0  0  0  1  1  0  3  0  0  0  1  1  0  0
  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0 56  0  0
  0  2  0  0  4  0  0  0  0  0  0  0  0  2  1  2  0  0  0  0  0  0  2  1
  0  0  0  2]
[ 0  1  1  0  0  0  0  0  0  0  1  1  0  0  0  0  1  0  0  0  0  0  0  0
  1  0  2  0  0  1  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  1  0  0
  1  1  0  1  0  2  6  0  0  2  1  0  0  2  1  0  2  1  0  0  0  0 41  0
  0  0  0  0  0  1  3  1  0  0  0  4  1  0  1  0  0  0  0  1 15  0  0  0
  0  0  0  0]
[ 0  0  0  0  0  0  0  0  0  0  2  0  1  1  0  0  0  1  0  0  0  0  0  3
  0  0  0  1  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  3  0  3  1  0  0  0  1  0  0  0 12  0  0  0  0  0  0  1  0  1  0 55
  0  4  0  0  2  1  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  2
  0  0  0  0]
[ 0  0  1  4  2  0  0  0  0  0  0  1  0  0  0  3  1  0  1  2  0  2  0  1
  0  1  2  2  0  1  1  1  2  0  0  2  0  0  0  0  0  1  0  0  1  0  0  0
  0  3  0  1  0  0  0  7  2  0  1  0  0  0  0  1  1  0  1  2  0  1  0  0
 17  1  2  0  0  3  2  1  3  0  0  0  0  5  0  0  0  1  3  0  0  1  0  4
  0  1  3  1]
[ 0  3  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0  0  1
  0  1  0  1  0  0  8  0  1  1  1  0  1  0  0  1  0  1  0  0  1  0  0  0
  1  0  1  1  0  0  0  0  0  0  0  0  0  1  0  0  1  0  1 10  0  0  0  1
  4 36  1  2  0  1  0  0  1  0  0  0  0  0  0  1  0  1  0  2  0  2  0  5
  0  0  0  3]
[ 0  0  0  1  3  0  2  3  0  0  1  2  1  0  0  2  0  0  1  0  0  0  0  0
  0  1  2  4  0  0  0  0  1  2  1  0  0  0  0  0  0  1  0  1  2  1  0  0
  0  0  6  2  0  0  0  1  0  0  0  0  0  0  0  4  1  3  0  0  0  0  0  0
  1  0 30  1  0  3  6  1  1  0  0  1  0  0  0  0  4  0  0  1  0  0  0  0
  0  0  0  2]
[ 0  0  0  0  2  0  0  3  0  0  0  0  1  0  0  0  0  0  0  3  0  0  1  0
  0  0  0  1  0  1  0  0  0  0  1  0  0  1  0  1  1  0  2  0  2  0  0  0
  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  2  5  0  0  0  0  0
  1  0  2 56  0  1  0  0  0  1  0  0  0  0  2  0  0  1  0  0  0  2  0  0
  0  3  0  1]
[ 0  0  0  0  0  0  0  0  0  1  0  0  5  0  0  0  0  6  0  0  1  0  0  3
  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1
  0  2  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0
  1  1  1  0 66  0  0  1  1  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0
  0  0  0  0]
[ 0  1  1  3  0  1  2  5  1  1  0  0  0  1  4  1  0  0  2  0  0  0  1  0
  1  1  1  0  0  1  1  0  0  0  0  2  3  1  1  0  1  1  3  0  6  2  1  0
  1  0  1  1  0  0  1  2  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0
  0  0  6  0  0 25  0  0  1  0  0  0  1  0  0  0  3  1  0  0  0  1  0  0
  0  0  2  1]
[ 0  0  0  0  1  0  1  3  0  0  0  1  0  0  0  2  0  0  5  0  0  1  3  0
  2  0  3  0  0  1  0  0  0  1  0  1  0  0  0  2  0  0  2  1  1  1  1  0
```

```
   2  0  1  0  0  0  0  1  0  2  0  1  0  1  0  0  1  0  1  0  0  1  0  0
   2  0  2  2  0  5 26  0  1  0  0  1  0  0  0  0  5  0  1  1  0  0  0  0
   0  1  0  9]
 [ 0  0  0  0  2  0  6  4  0  0  0  0  0  0  2  0  0  0  3  0  0  0  1  1
   0  0  6  1  0  1  0  0  0  2  2  2  0  0  3  0  0  0  0  0  4  0  1  0
   0  0  0  0  0  0  4  1  4  0  0  0  0  0  0  1  0  2  2  0  0  1  0  0
   1  0  2  1  0  1  3 31  0  0  0  0  1  0  0  1  0  1  1  0  0  0  0  0
   0  1  0  0]
 [ 0  1  0  3  1  1  1  0  0  1  0  0  0  1  2  3  0  1  2  0  0  0  1  1
   0  0  1  3  0  0  0  1  0  1  3  0  0  0  8  0  0  0  4  4  0  1  0  1
   0  0  0  2  0  0  0  2  1  0  0  0  0  1  0  0  5  3  0  1  0  1  0  0
   3  0  4  1  1  4  1  2 12  2  0  0  0  0  1  0  2  1  0  0  0  1  0  0
   1  2  0  0]
 [ 0  0  0  0  0  1  0  0  1  0  0  0  1  9  0  0  0  2  0  0  0  0  0  0
   0  1  0  0  0  0  0  0  2  0  0  0  0  2  0  0  0  1  1  0  0  0  0  0
   1  0  0  0  0  0  0  0  0  0  8  0  0  0  0  0  0  0  0  0  1  0  0  1
   0  0  0  0  0  0  0  0  0 50  0  0  0  4  1  1  0  4  6  0  0  0  2  0
   0  0  0  0]
 [ 0  0  0  0  0  0  4  1  1  0  0  0  0  1  2  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1
   0  0  0  0  0  1  0  0  0  3  0  0  1  0  3  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  1  0  0  0 73  2  0  0  0  0  0  2  0  0  3  0  0  0
   0  0  0  0]
 [ 3  2  0  0  0  0  0  1  0  2  2  1  0  1  1  1  2  0  0  0  0  1  1  0
   0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  2  0  0
   0  0  0  0  0  8  4  0  0  4  0  1  0  2  4  0  0  0  0  0  0  0  3  0
   0  0  0  0  0  0  1  0  0  0  0 36  0  0  1  1  0  1  0  1  7  0  0  0
   0  1  1  0]
 [ 0  0  1  0  0  2  0  0  1  1  2  4  2  2  1  2  4  1  1  2  2  0  0  0
   0  1  1  0  0  0  0  0  0  0  3  0  0  2  2  1  4  1  0  0  1  0  4  0
   0  0  0  1  0  0  0  0  0  0  2  0  0  1  0  0  3  0  1  0  0  0  0  1
   0  0  0  1  1  0  2  1  0  1  0  0 28  1  1  2  0  1  2  0  0  0  0  0
   0  0  2  0]
 [ 0  1  0  0  1  0  0  1  0  0  0  0  2  0  0  2  0  1  0  2  0  0  0  0
   0  0  1  2  0  1  0  0  0  1  0  0  1  4  0  0  1  0  1  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  7  3  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  1  1  1  0  1  0  0  3  0  0  0 51  0  0  1  6  2  0  0  1  0  0
   0  0  0  0]
 [ 0  0  0  0  0  1  0  2  0  0  1  1  0  2  1  0  4  0  0  0  0  0  3  0
   0  1  0  0  2  1  0  0  1  0  0  1  0  0  0  5  3  1  0  0  0  1  0  0
   0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  1  0  3  0  0  2  0  0
   0  0  1  1  0  0  2  0  0  0  1  0  0  1 40  8  0  2  0  0  0  1  1  0
   0  0  1  2]
 [ 0  0  0  0  0  0  0  0  0  0  2  0  3  2  0  1  1  0  0  0  2  0  2  0
   0  2  0  0  0  0  0  1  0  3  0  0  0  1  0  0  1  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  2  0  0  0  1  0  0  0  1  0  0  0  0  0
   0  0  0  0  2  0  0  0  0  6  1  0  2  0  5 48  2  2  0  0  0  0  7  0
   0  0  0  0]
 [ 0  1  0  1  2  1  3  0  4  0  1  1  0  0  0  3  0  0  1  1  0  0  0  0
   0  0  3  0  0  3  0  1  1  0  4  1  0  0  1  0  0  0  8  8  0  0  1  0
   0  1  3  3  0  0  0  0  0  0  0  0  0  1  0  1  2  0  1  0  0  0  0  0
   0  0  0  0  0  1  5  0  0  0  0  0  0  0  0  0 29  2  0  0  0  0  0  0
   0  1  0  0]
 [ 0  0  0  0  0  0  1  1  3  0  0  0  0  3  1  0  1  2  1  0  0  0  0  0
   0  0  4  0  0  1  0  0  2  1  0  0  0  3  0  0  0  1  0  0  1  1  1  0
   6  0  0  0  0  0  0  0  0  2  1  1  0  0  0  0  0  0  0  0  0  0  0  0
   1  0  0  1  0  0  0  1  1  2  0  0  1  2  0  0  2 50  0  0  1  0  0  0
   0  0  0  0]
 [ 0  2  1  0  1  0  1  1  0  0  1  0  1  1  0  0  0  0  0  0  0  1  0  0
   0  0  0  1  1  0  0  0  1  1  0  1  0  2  0  1  0  0  0  0  0  1  0  0
   4  2  0  0  0  0  0  0  1  0  5  0  0  0  0  0  0  0  1  0  0  0  0  1
   1  0  0  2  1  0  1  0  0 16  0  0  0  4  0  2  0  5 33  0  0  2  0  0
   0  0  0  0]
 [ 0  2  0  0  0  0  0  1  0  0  0  1  1  0  1  0  0  0  2  1  0  0  0  0
   1  0  2  3  0  0  1  0  0  1  0  0  0  0  0  0  1  0  0  0  5  0  1  0
   0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  2  0  0  2  0  0
   2  2  1  3  0  1  8  1  0  0  0  0  2  1  0  0  0  2  0 46  0  1  0  1
   0  0  0  0]
 [ 1  0  0  0  0  0  3  1  0  1  1  0  0  0  3  0  0  0  0  0  1  0  0  0
   0  0  3  0  1  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  1
   1  0  0  0  0  4 12  0  0  3  0  0  0  0  4  0  0  0  0  0  0  0 13  0
   0  0  0  0  0  1  0  1  0  0  2  6  0  0  0  0  0  0  1  0 33  0  0  0
   0  0  0  1]
 [ 0  2  0  1  4  0  1  3  2  0  0  0  1  0  0  1  0  0  0  0  1  0  1  1
   1  0  6  0  0  0  2  1  1  1  0  0  0  0  2  0  1  0  0  0  1  2  1  0
   1  1  1  1  0  0  1  3  0  0  0  0  0  0  0  2  3  0  2  7  0  1  0  0
   3  2  4  1  0  0  3  0  0  0  0  0  0  0  1  0  0  0  0  3  0 19  0  2
   0  2  0  0]
 [ 0  0  0  0  0  3  0  0  0  0  1  0  0  0  0  0  2  0  0  0  1  0  1  0
   0  2  0  0  0  0  0  0  0  2  0  0  1  1  0  0  0  0  1  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
   0  0  0  0  1  1  1  0  0  2  0  0  4  0  5  3  0  0  0  0  0  0 65  0
   1  0  0  0]
 [ 0  2  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0  1  0 12  0  1  0  0  1  0  1  0  3  0  0  0  0  0  0  0  0
   1  1  1  0  0  0  1  4  1  0  0  0  0  1  0  0  0  0  0  2  0  3  0  0
   2  6  0  2  0  0  0  0  0  0  0  0  1  0  2  0  0  0  0  0  0  1  0 45
   0  0  0  2]
 [ 0  0  0  1  1  0  0  0  1  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1
   0  1  0  0  0  0  0  0  0  8  1  0  0  1  0  1  0  0  1  1  0  0  0 18
   0  1  0  0 18  0  0  0  1  0  0 10  0  0  0  1  0  0  0  1  1  0  1  0
   0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0
  25  0  0  0]
 [ 0  1  0  0  2  0  0  1  0  0  0  0  0  0  0  1  2  0  0  2  0  1  1  1
   0  0  1  0  1  1  0  0  0  4  1  0  0  1  4  1  0  0  0  1  0  0  1  0
   0  0  2  0  0  0  0  0  2  0  0  0  0  0  0  2  2  0  5  0  0  0  0  0
   2  0  4  0  0  0  2  0  2  0  0  0  1  0  0  0  1  1  0  0  0  0  0  0
   0 44  1  1]
 [ 0  1  5  1  0  1  2  0  1  0  1  4  0  0  1  1  0  0  0  0  0  1  2  0
   1  0  5  0  0  0  0  0  1  0  0 18  1  1  0  0  1  0  0  0  0  1  8  0
   0  0  0  3  0  2  1  1  1  1  0  0  0  0  0  0  3  1  1  0  0  0  1  0
   0  0  0  1  0  0  1  0  0  0  0  0  1  1  3  0  1  1  0  0  0  0  0  0
   0  0 17  1]
 [ 2  0  1  0  1  0  0  2  1  0  2  0  2  1  0  0  0  0  0  0  0  0  6  1
   0  2  3  0  1  0  1  0  0  0  1  0  0  0  1  4  2  1  1  0  1  1  0  0
   0  0  1  1  0  0  0  0  0  0  0  0  0  3  0  0  1  0  1  2  0  3  0  0
   0  1  1  1  0  0  7  1  0  0  0  0  0  0  1  1  0  1  0  0  0  0  0  1
   0  0  0 35]]
```

```
---------------------------20  classes------------------
Accuracy and confusion matrix for 20 labels
---------------------------------Accuracy= 51.690000000000005 --------------
----------
Confusion-Matrix:
 [[202  40   1   3   7  12   5   8  31   7   9  20  21  21  12  27  35   6
   11  22]
 [ 37 241   9  10   8  11   5  11   6   3  12   6  19  20  10  49  16   4
    8  15]
 [  0   4 339  14  50   2   3  21   0   1   2   1   5  16  12  10   5   2
    7   6]
 [  7   7  11 268  15  61  15  10   3   5   2   7   8  11  24  12   7   2
   11  14]
 [  6  13  51  17 293   5   2  19  11   0   0  13   6  18  14  18   5   3
    4   2]
 [  2   6   7  32   8 263  29  12   8  12   9   8  11  15  12  20   4   3
   11  28]
 [  1   5   5  28   3  46 253   5   5  16  10  14  10  13  23   8   9   6
   20  20]
 [ 10  10  17  11  13   4   3 283  15   3   4   7  12  42   8  22  18   5
    4   9]
 [ 31   7   2   5  16   8   4  18 226   1  10  39  47  11  10  19  27   3
    6  10]
 [  4   3   0  10   0  11   8   2   6 333  23   8   1   4   6   6   3  18
   17  37]
 [ 16  13   5   6   7  11   4   8   3  21 324   3   7   5   2   7   4  24
   13  17]
 [ 26   6   1   6  10   7   6  26  46  12  11 217  34  13  15  21   9   6
   12  16]
 [ 28  14   3   1   3   8   2  18  52   5   2  32 197  24  12  33  48   4
    8   6]
 [ 15  10  10  14  13  22   9  50  13   4   6   7  17 185  15  49  28   9
    9  15]
 [  7  12  13  13  12  19  14  14  21   4   0  13  17  27 259   6  17   5
   16  11]
 [ 39  28   2   3   8  16   4  32  26   9  15  24  21  48  15 156  21   6
   16  11]
 [ 27  15   4  11  12  12   3  21  37   6   8  29  48  36  22  39 156   3
    4   7]
 [  6   4   7   1   1   6   1   5  10  19  18   3   5   2   1   5   1 388
    7  10]
 [  5   8   7   8   0  10   7  13   2  25   7   4   8   4   8  12   0   3
  291  78]
 [  4  10   3   3   2  12  10  11   7  35   7   7   4  12   3  11   3   5
   56 295]]
--------------------------------------Considering Top-3--------------------
-------

-----------------------Accuracy for 100 classes=  39.08
-----------------------Accuracy for 20 classes=  51.69
--------------------------------------Considering Top-5--------------------
-------

-----------------------Accuracy for 100 classes=  67.66
-----------------------Accuracy for 20 classes=  80.71
```