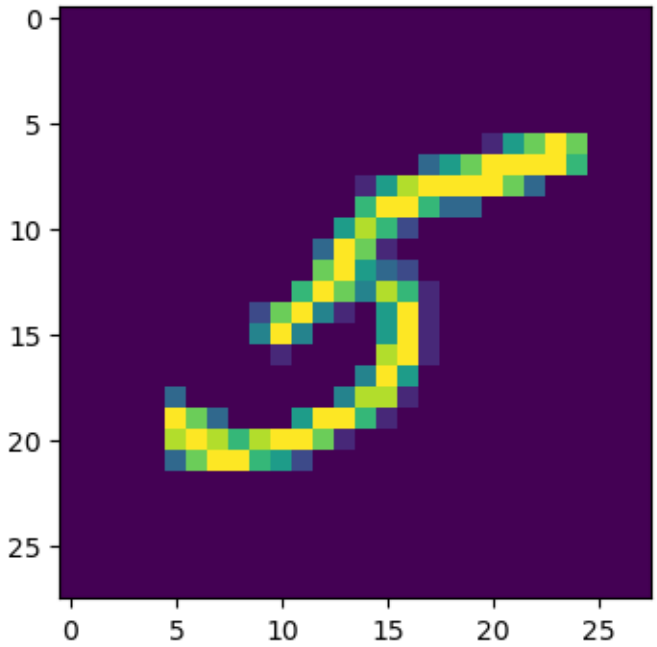
Madhuri_CS541_HW1_final

November 24, 2022

[1]:
```
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
import numpy as np
import time
plt.rcParams["figure.figsize"] = (5,4)
np.set_printoptions(linewidth=320)
(x_train,y_train), (x_test,y_test)= tf.keras.datasets.mnist.load_data()
```

[2]:
```
plt.imshow(x_train[35,:,:])
print("Given Label:" + str(y_train[35]))
```

```
Given Label:5
```



Dataset Preparation

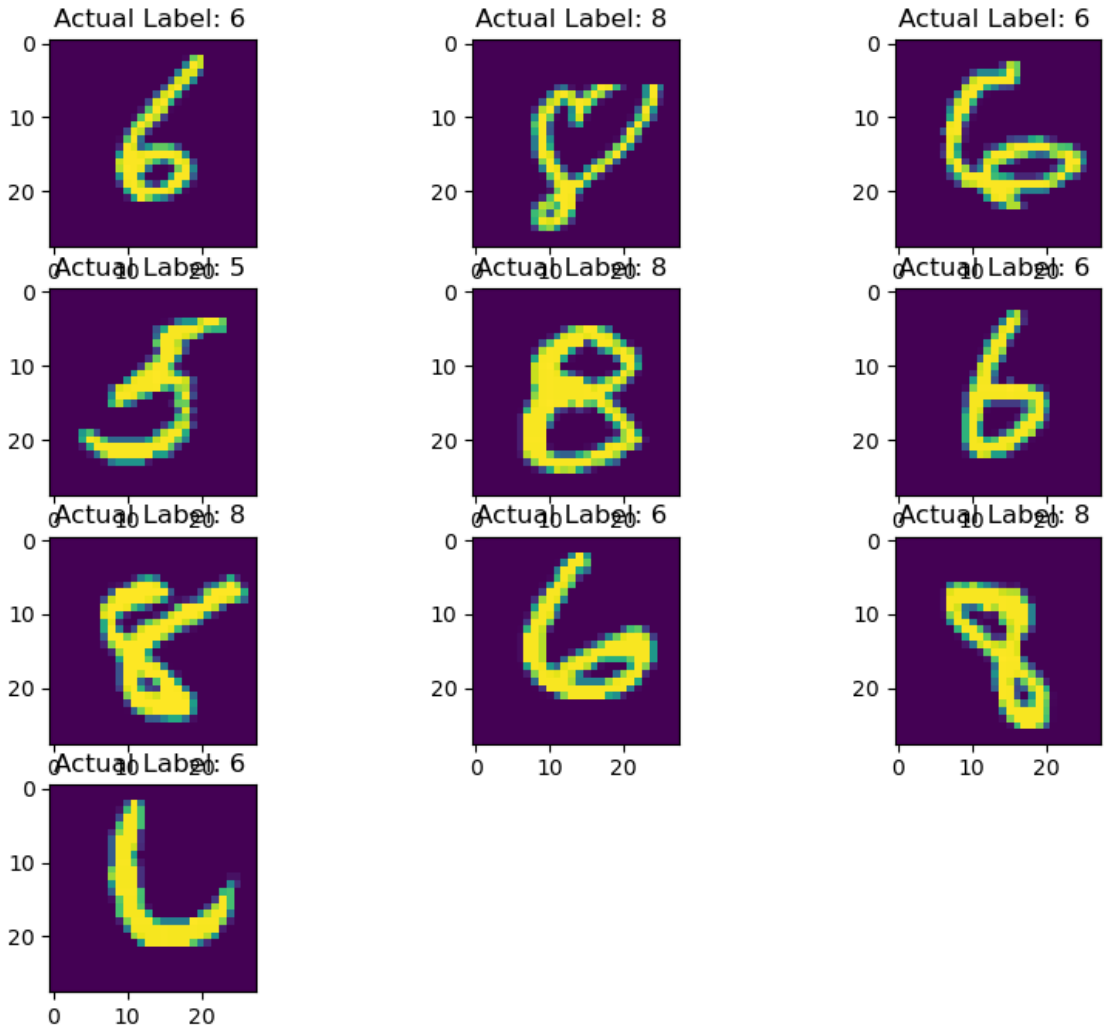
[3]:
```
indexes=[]
indexes1=[]
indexes.append(np.argwhere(y_train==5).tolist())
[indexes1.append(i[0]) for i in indexes[0]]
indexes=[]
indexes.append(np.argwhere(y_train==6).tolist())
[indexes1.append(i[0]) for i in indexes[0]]
indexes=[]
indexes.append(np.argwhere(y_train==8).tolist())
[indexes1.append(i[0]) for i in indexes[0]]

samples_15k=random.sample(indexes1,15000)
X_tra=x_train[samples_15k]
y_tra=y_train[samples_15k]

print(X_tra.shape)
```

```
(15000, 28, 28)
```

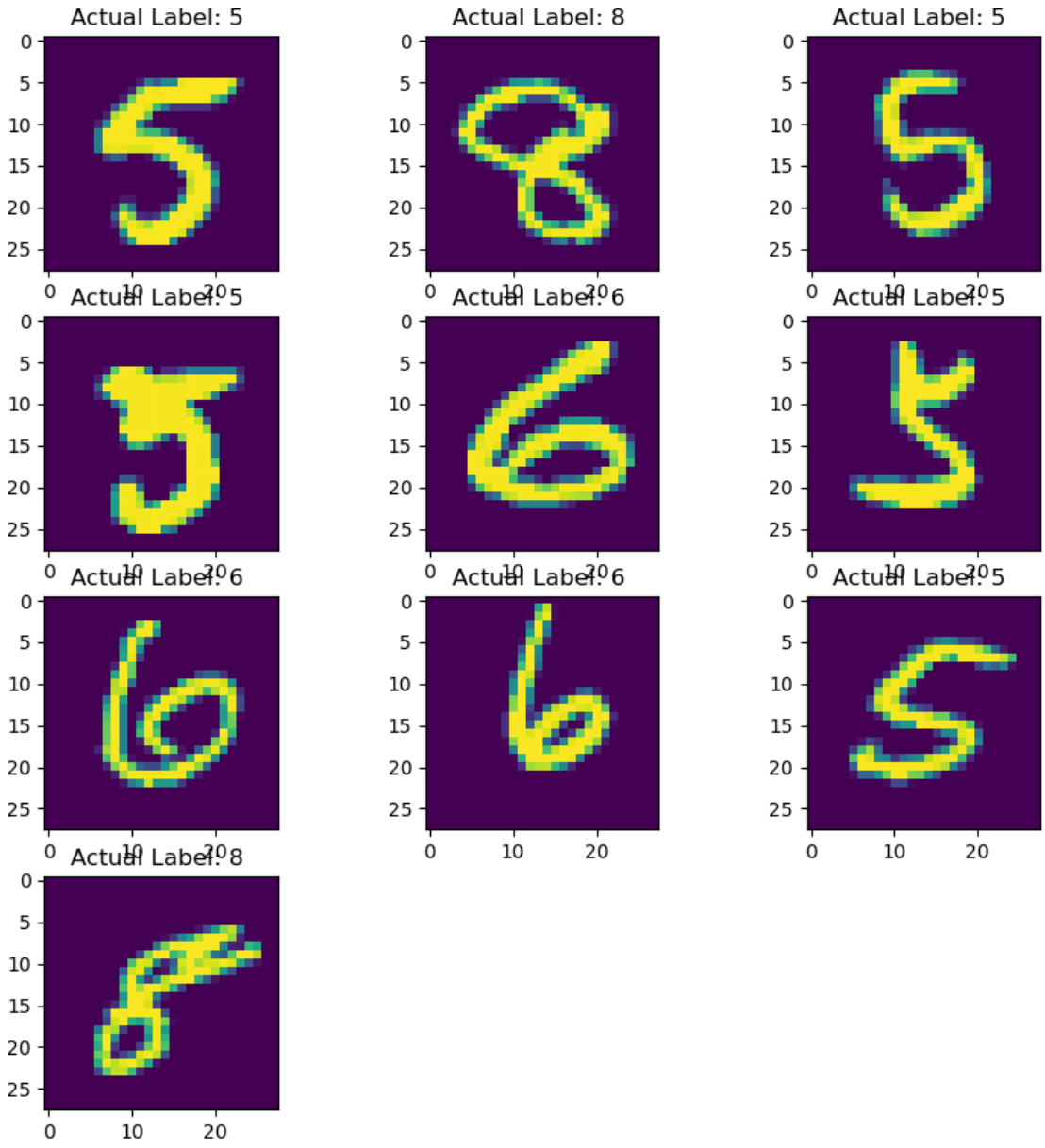
[4]:
```
indexes=[]
indexes1=[]
indexes.append(np.argwhere(y_test==5).tolist())
[indexes1.append(i[0]) for i in indexes[0]]
indexes=[]
indexes.append(np.argwhere(y_test==6).tolist())
[indexes1.append(i[0]) for i in indexes[0]]
indexes=[]
indexes.append(np.argwhere(y_test==8).tolist())
[indexes1.append(i[0]) for i in indexes[0]]

samples_2k=random.sample(indexes1,2500)
X_tes=x_test[samples_2k]
y_tes=y_test[samples_2k]
```

Randomly picking 10 images from X_train and X_test

[5]:
```
randomX_tra = np.random.randint(15000, size =10)
plt.figure(figsize=(10,8))
j=0
for i in randomX_tra:
    j=j+1
    ax=plt.subplot(4,3,j)
    plt.imshow(X_tra[i, :,:])
    plt.title("Actual Label: %s " % y_tra[i])
```



[6]:
```
randomX_tes = np.random.randint(2500, size=10)
plt.figure(figsize=(10,10))
j=0
for i in randomX_tes:
    j=j+1
    ax=plt.subplot(4,3,j)
    plt.imshow(X_tes[i, :])
    plt.title("Actual Label: %s " % y_tes[i])
```



Normalizing X_train

[7]:
```
X_tra=X_tra/255
```

Normalizing X_test

[8]:
```
X_tes = X_tes/255
```

Calculating Distance matrix

[9]:
```
#calculate L2
def distance_matrix(X_train, X_test):
    num_test = X_test.shape[0]
    num_train = X_train.shape[0]
    dists = np.zeros((num_test, num_train))
    for i in range(num_test):
        for j in range(num_train):
            dists[i][j] = np.sqrt(np.sum((X_test[i] - X_train[j]) ** 2))
    return dists
```

[10]:
```
M=distance_matrix(X_tra, X_tes)
```

Retrieving the closest k neighbours

[21]:
```
def retrieve_k(i,M,k):
    return M[i].argsort()[:k]
```

[22]:
```
#knn_ix=retrieve_k(1,M,9)
#closest_y = y_train[knn_ix]
# retreiving k(10) nearest labels for test image 1
closest_y=retrieve_k(1,M,10)
print(closest_y)
print(y_tra[closest_y])
```

```
[ 3425  7995 11829  3799 11646  5818 12188  7619  5184   187]
[6 6 6 6 6 6 6 6 6 6]
```

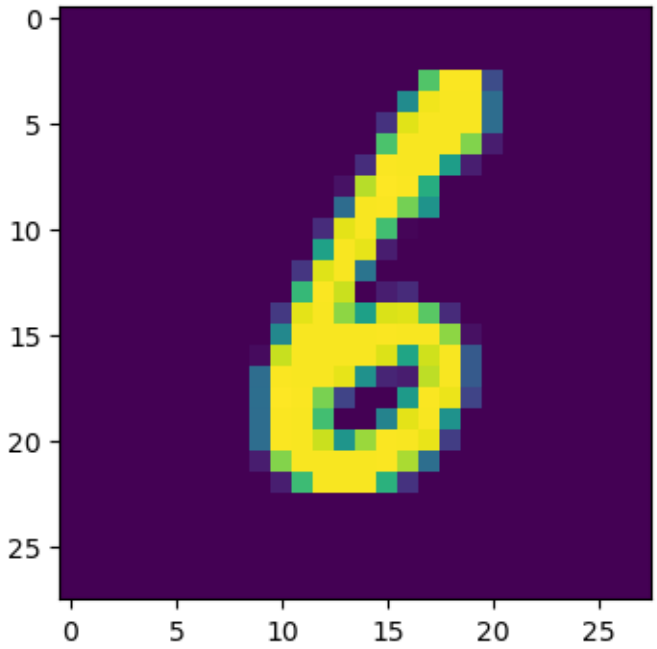
Calculating the precision

[23]:
```
def precision_k(y,y_train,closest_y):
    sum=0
    for i in closest_y:
        if y==y_train[i]:
            sum+=1
    return sum/closest_y.shape[0]
```

calculating precision for a single example with `label 6`

[24]:
```
plt.imshow(X_tes[1,:,:])
print("Original label:"+str(y_tes[1]))
```

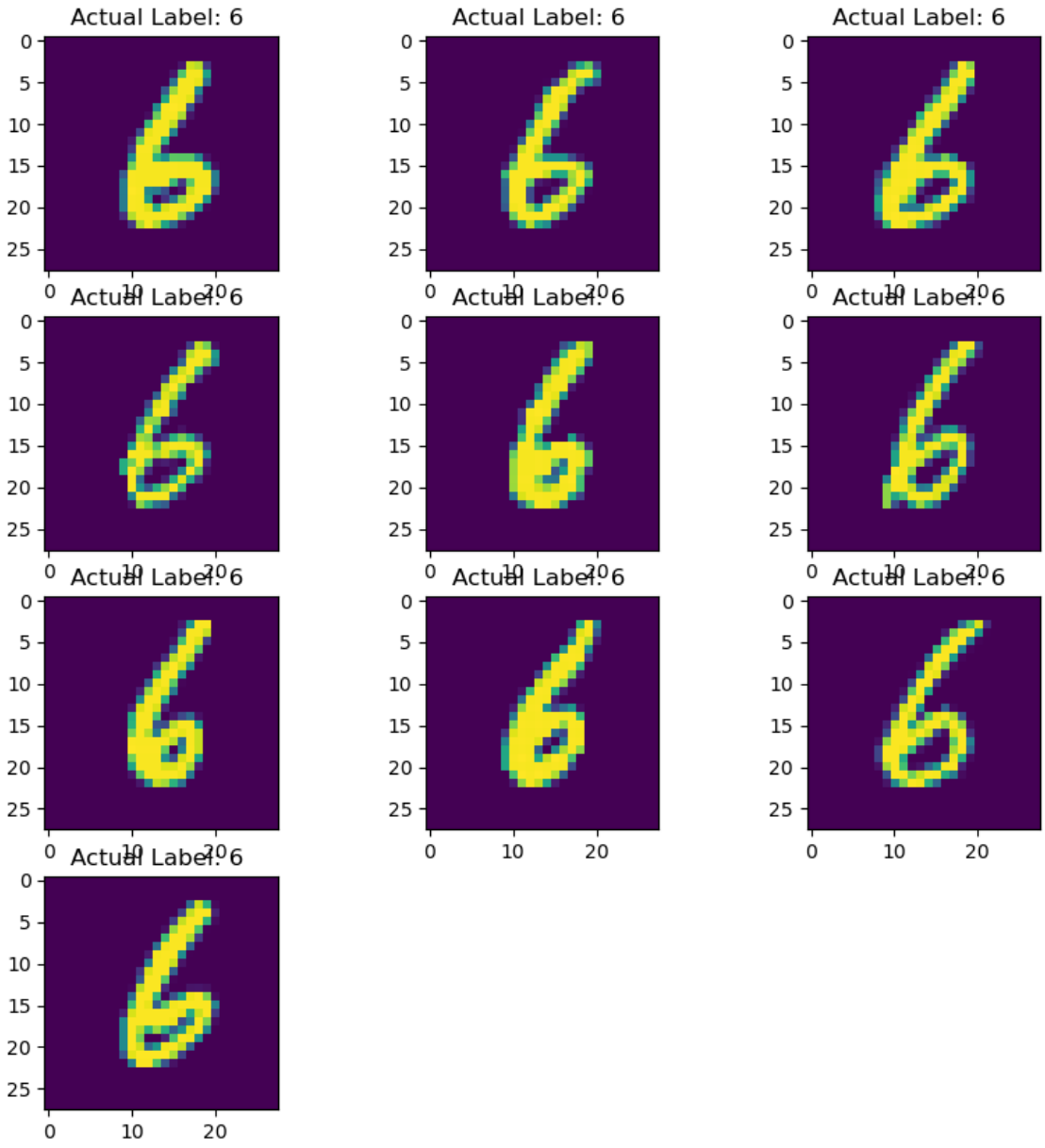
```
Original label:6
```



[31]:
```
closest_y1=retrieve_k(1,M,10)
precision_k(6,y_tra,closest_y1)
```

[31]: 1.0

[32]:
```
plt.figure(figsize=(10,10))
j=0
for i in closest_y1:
    j=j+1
    ax=plt.subplot(4,3,j)
    plt.imshow(X_tra[i, :])
    plt.title("Actual Label: %s " % y_tra[i])
```



Calculating the average precision

[33]:
```
def avg_precision_k(X_train,y_train,X_test,y_test,k):
    avg=0
    for i in range(X_test.shape[0]):
        closest=retrieve_k(i,M,k)
        pp_val=precision_k(y_test[i],y_train,closest)
        avg+=pp_val

    return avg/X_test.shape[0]
```

[34]:
```
k=10
avg_precision_k(X_tra,y_tra,X_tes,y_tes,k)
```

[34]:
```
0.9728800000000017
```

0.0.1 8.4 Retrieval Performance with Original Data

[35]:
```
k=[1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]

tim={}
precision={}
for i in k:
    t1=time.time()
    pp=avg_precision_k(X_tra,y_tra,X_tes,y_tes,i)
    t2=time.time()
    tim[i]=t2-t1
    precision[i]=pp
```

[36]:
```
plt.plot(k,list(precision.values()))
```

[36]:
```
[<matplotlib.lines.Line2D at 0x140516ed100>]
```