Name: Tarun Tanmay

Class: MBATech CE

Roll No: N049

Sem: 6

SAP ID: 70471018055

In [ ]:

```
#Experiment 6
#CNN for Gray Iamges
#classification of 70,000 images of handwritten digits (from 0 to 9)
```

Importing Libraries:

In [49]:

```
from keras.datasets import mnist #taking inbuilt dataset #mnist named dataset has some gr
ay images
from keras.models import Sequential #to feed data sequentially from layer to layer in our
CNN model
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten
from keras.utils import np_utils
```

In [50]:

```
from sklearn.metrics import accuracy_score
```

In [51]:

```
(X_tr, y_tr),(X_ts,y_ts)=mnist.load_data() #loading the dataset
```

In [52]:

```
X_tr.shape
```

Out[52]:

```
(60000, 28, 28)
```

In [53]:

```
#there are 60,000 images for training, and each image is gray image
```

In [54]:

```
X_ts.shape
```

Out[54]:

```
(10000, 28, 28)
```

In [ ]:

```
#there are 10,000 images for testing
```

In [55]:

```
y_tr.shape
```

Out[55]:

```
(60000,)
```

In [56]:

```
y_tr[0]
```

Out[56]:

5

In [ ]:

```
#if we train this network, then the zeroth image is of 5
```

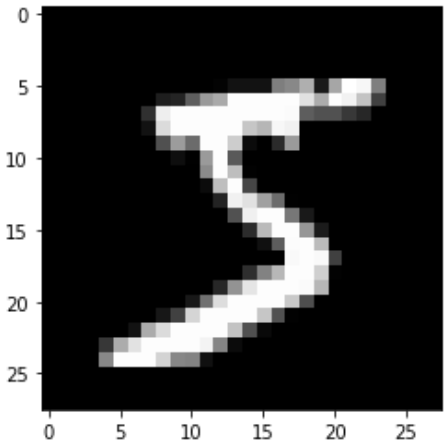
Plotting the Output:

In [58]:

```
import matplotlib.pyplot as plt
plt.imshow(X_tr[0], cmap='gray') #displaying the first training image
```

Out[58]:

<matplotlib.image.AxesImage at 0x7fbe00c0d750>



In [59]:

```
y_tr[3] #checking the label of the fourth training image
```

Out[59]:

1

In [ ]:

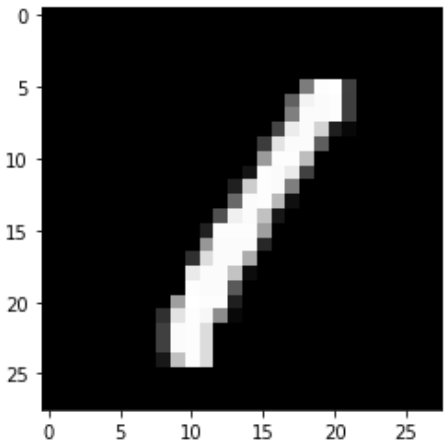
```
#the third image in training set is of 1
```

In [61]:

```
plt.imshow(X_tr[3],cmap='gray') #displaying the fourth training image
```

Out[61]:

<matplotlib.image.AxesImage at 0x7fbe00b5afd0>



In [62]:

```
X_train=X_tr.reshape(X_tr.shape[0],28,28,1) #reshaping the training set
#adding 1 more dimension so that CNN is told about the planes in the input image
X_test=X_ts.reshape(X_ts.shape[0],28,28,1)
```

In [63]:

```
X_tr.shape #original dimensions of training set
```

Out[63]:

```
(60000, 28, 28)
```

In [64]:

```
X_train.shape  #new dimensions for CNN
```

Out[64]:

```
(60000, 28, 28, 1)
```

In [65]:

```
X_test.shape
```

Out[65]:

```
(10000, 28, 28, 1)
```

In [ ]:

```
#the shape of the image is redefined since CNN is told how many plains are there in the i
nput image
#Therefore, it is specified here that there is 1 plain
```

In [66]:

```
#converting them into float type from integer values
X_train=X_train.astype('float32')
X_test=X_test.astype('float32')
```

Normalization:

In [ ]:

```
#commonly used images are 8 bit images ( from 0 to 255)
#we are converting it to 0 and 1
```

In [67]:

```
X_train=X_train/255.0
X_test=X_test/255.0
```

In [ ]:

```
#since our digits are from 0 to 9, therefore we have 10 classes
```

In [68]:

```
#performing one hot encoding on the labels
num_classes=10
y_train_one_hot=np_utils.to_categorical(y_tr,num_classes=num_classes) #converting into ca
tegorical data
#originally there was only 1 column, now it has columns=no. of classes
y_test_one_hot=np_utils.to_categorical(y_ts,num_classes=num_classes)
```

In [69]:

```
y_train_one_hot.shape #now we have 10 columns for 10 neurons
```

Out[69]:

```
(60000, 10)
```

In [70]:

```
y_ts.shape #earlier it(output) had only 1 column
```

Out[70]:

```
(10000,)
```

In [71]:

```
y_test_one_hot.shape #10 columns for 10 neurons
```

Out[71]:

```
(10000, 10)
```

In [ ]:

```
#each neuron is either 0 or 1, so 1 is given to that column which has label and others ar
e given 0
```

In [72]:

```
y_tr[3] #original output of training set, for third image
```

Out[72]:

```
1
```

In [73]:

```
y_train_one_hot[3]
```

Out[73]:

```
array([0., 1., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

Creating the Model:

In [75]:

```
#Building the Model
model=Sequential()
#first layer is Convolution layer
model.add(Conv2D(25,kernel_size=(3,3),strides=(1,1),padding='valid', activation='relu',i
nput_shape=(28,28,1)))
#we apply 25 filters of each 3*3 size on the image
#we use stride of 1*1 for each row and column for the convolution
#then we use activation function as relu
#Pooling
model.add(MaxPool2D(pool_size=(2,2)))
#flatten
#it is a 2D matrix, the next layer is not Convolution layer thus we flatten it
model.add(Flatten())
#the output after flattening is converted into one dimensional vector
#Dense Layer
model.add(Dense(100,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

Compiling the Model:

In [76]:

```
model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
#we compile the model and check for loss and accuracy
#we check the loss using categorical_crossentropy
```

Fitting the Model:

In [77]:

```
model.fit(X_train,y_train_one_hot,batch_size=128,epochs=5,validation_data=(X_test,y_test
_one_hot))
```

```
Epoch 1/5
469/469 [==============================] - 22s 47ms/step - loss: 0.4631 - accuracy: 0.868
8 - val_loss: 0.0875 - val_accuracy: 0.9735
Epoch 2/5
469/469 [==============================] - 21s 44ms/step - loss: 0.0819 - accuracy: 0.976
4 - val_loss: 0.0625 - val_accuracy: 0.9794
Epoch 3/5
469/469 [==============================] - 20s 43ms/step - loss: 0.0533 - accuracy: 0.984
2 - val_loss: 0.0503 - val_accuracy: 0.9832
Epoch 4/5
469/469 [==============================] - 21s 44ms/step - loss: 0.0383 - accuracy: 0.989
2 - val_loss: 0.0451 - val_accuracy: 0.9844
Epoch 5/5
469/469 [==============================] - 20s 43ms/step - loss: 0.0276 - accuracy: 0.991
8 - val_loss: 0.0492 - val_accuracy: 0.9838
```

Out[77]:

```
<tensorflow.python.keras.callbacks.History at 0x7fbe00a57510>
```

In [ ]:

```
#our trained network has 985 accuracy with 5 epochs training
```

Testing: Predicted Output

In [78]:

```
predict=model.predict_classes(X_test) #predicting the X_test
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:450:
UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01
. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi
-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.pre
dict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it
uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

In [79]:

```
predict[0]
```

Out[79]:

```
7
```

In [80]:

```
y_ts[0]
```

Out[80]:

```
7
```

In [81]:

```
y_test_one_hot[0]
```

Out[81]:

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

In [82]:

```
predict[6]
```

Out[82]:

```
4
```

In [83]:

```
y_ts[6]
```

Out[83]:

```
4
```

In [84]:

```
y_test_one_hot[6]
```

Out[84]:

```
array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.], dtype=float32)
```

In [85]:

```
#CIFAR-10 of color images
from keras.datasets import cifar10
from keras.models import Sequential #to feed data sequentially from layer to layer in our CNN model
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten
from keras.utils import np_utils
```

In [86]:

```
from sklearn.metrics import accuracy_score
```

In [87]:

```
(X_tr1, y_tr1),(X_ts1,y_ts1)=cifar10.load_data() #loading the dataset
```

In [88]:

```
X_tr1.shape
```

Out[88]:

```
(50000, 32, 32, 3)
```

In [89]:

```
X_ts1.shape
```

Out[89]:

```
(10000, 32, 32, 3)
```

In [ ]:

```
#the value 3 in dimension 4 tells that it is RGB image
```

In [90]:

```
y_tr1.shape
```

Out[90]:

```
(50000, 1)
```

In [91]:

```
y_ts1.shape
```

Out[91]:

```
(10000, 1)
```

In [93]:

```
# checking the label of the first training image
y_tr1[0]
```

Out[93]:

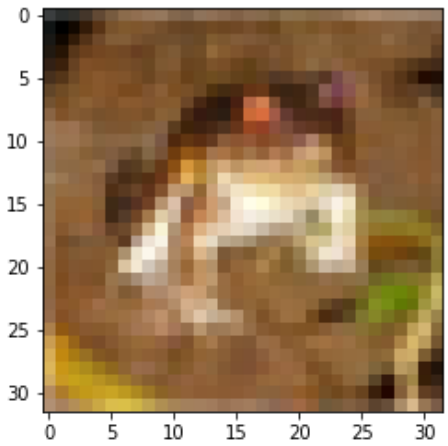
```
array([6], dtype=uint8)
```

In [95]:

```
plt.imshow(X_tr1[0])
# displaying the first training image
```

Out[95]:

```
<matplotlib.image.AxesImage at 0x7fbe00a603d0>
```



In [96]:

```
# checking the label of the fourth training image
y_tr1[3]
```

Out[96]:

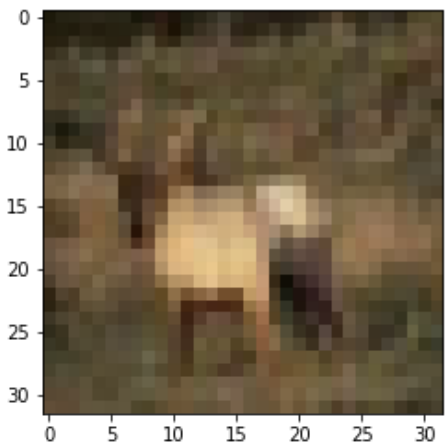
```
array([4], dtype=uint8)
```

In [97]:

```
# displaying the fourth training image
plt.imshow(X_tr1[3])
```

Out[97]:

```
<matplotlib.image.AxesImage at 0x7fbe00ed5e50>
```



In [99]:

```
X_train1=X_tr1.reshape(X_tr1.shape[0],32,32,3) #reshaping the training set
X_train1.shape
```

Out[99]:

```
(50000, 32, 32, 3)
```

In [100]:

```
X_test1=X_ts1.reshape(X_ts.shape[0],32,32,3)# reshaping x_test to specify the number of planes in the image
X_test1.shape
```

Out[100]:

```
(10000, 32, 32, 3)
```

In [108]:

```
# chaning the data type of the training and testing dataset
X_train1=X_train1.astype('float32')
X_test1=X_test1.astype('float32')
#normalizing the dataset
X_train1=X_train1/255.0
X_test1=X_test1/255.0
num_classes=10
y_train1_one_hot=np_utils.to_categorical(y_tr1,num_classes=num_classes)
y_test1_one_hot=np_utils.to_categorical(y_ts1,num_classes=num_classes)
print(y_train1_one_hot.shape)
print(y_test1_one_hot.shape)
```

```
(50000, 10)
(10000, 10)
```

In [109]:

```
# checking the change in the label after one hot encoding
print(y_tr1[3])
print(y_train1_one_hot[3])
```

```
[4]
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```

In [110]:

```
# building the model
model1 = Sequential()
model1.add(Conv2D(50, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu', input_shape=(32,32,3)))
# adding input layer with input size 28,28,1 and 50 filters
model1.add(MaxPool2D(pool_size=(2,2))) # add a maxpooling layer with a 2x2 pool size
model1.add(Conv2D(75, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
# adding convolutional layer with 75 filters
model1.add(MaxPool2D(pool_size=(2,2))) # add a maxpooling layer with a 2x2 pool size
model1.add(Dropout(0.25)) # removing 25% of connections
model1.add(Conv2D(125, kernel_size=(3,3), strides=(1,1), padding='same', activation='relu'))
# adding convolutional layer with 75 filters
model1.add(MaxPool2D(pool_size=(2,2))) # add a maxpooling layer with a 2x2 pool size
model1.add(Dropout(0.25)) # removing 25% of connections
model1.add(Flatten()) # converting a two dimentional image to single dimension
model1.add(Dense(500, activation='relu')) # adding a dense layer with 500 neurons
model1.add(Dropout(0.4)) # removing 40% of connections
model1.add(Dense(250, activation='relu')) # adding a dense layer with 250 neurons
model1.add(Dropout(0.3)) # removing 30% of connections
model1.add(Dense(10, activation='softmax')) # adding output layer with 10 output neurons
```

In [116]:

```
#Compiling the Model
model1.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')
```

In [117]:

```
model1.fit(X_train1,y_train1_one_hot,batch_size=128,epochs=2,validation_data=(X_test1,y_test1_one_hot))
```

```
Epoch 1/2
391/391 [==============================] - 213s 542ms/step - loss: 1.9054 - accuracy: 0.2834 - val_loss: 1.2717 - val_accuracy: 0.5418
Epoch 2/2
391/391 [==============================] - 211s 539ms/step - loss: 1.2968 - accuracy: 0.5328 - val_loss: 1.0983 - val_accuracy: 0.6012
```

Out[117]:

```
<tensorflow.python.keras.callbacks.History at 0x7fbe00871e50>
```

In [123]:

```
predict1=model1.predict_classes(X_test1)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

In [124]:

```
predict1[1]
```

Out[124]:

```
8
```

In [125]:

```
y_ts1[1]
```

Out[125]:

```
array([8], dtype=uint8)
```

In [127]:

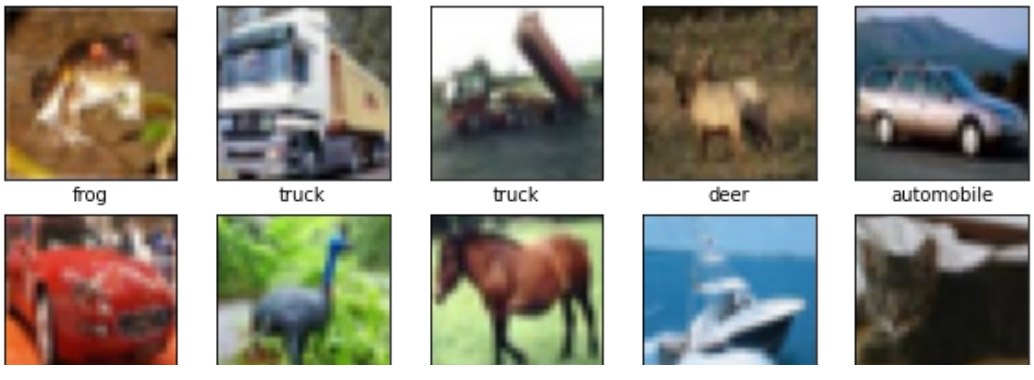
```
y_test1_one_hot[1]
```

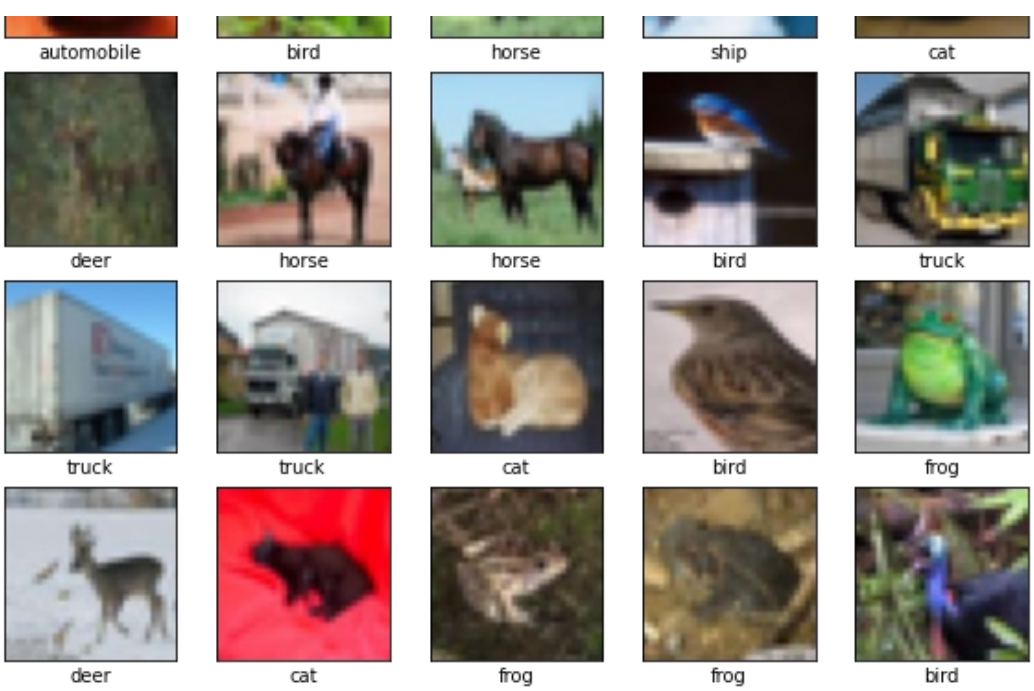
Out[127]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 1., 0.], dtype=float32)
```

In [128]:

```
class_names=['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']
plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(X_train1[i],cmap='binary')
  plt.xlabel(class_names[y_tr1[i][0]])
```





Conclusion:

1)For MNIST dataset, 1 convolutional layer, and 2 fully connected layers are used, and the model gets 98.30%. When hand written digit is 7, model predicts it as 7.

2)CNN model is trained for CIFAR-10 dataset, 3 convolutional layers are used, and 3 fully connected layers are used for the CNN, for 2 epochs the accuracy is 63.80%. The given model has 10 classes, which are plotted for better understanding