Renduchintala Navya - AIML - 2306AML112

Design a Neural Network using ANN algorithm on CIFAR 10 dataset

```
https://www.cs.toronto.edu/~kriz/cifar.html
```

In [1]:
```
# pip install tensorflow - installed
```

In [2]:
```
import tensorflow as tf

# Display the version
print(tf.__version__)

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt # plotting library
%matplotlib inline

from keras.models import Sequential
from keras.layers import Dense , Activation, Dropout
from keras.optimizers import Adam ,RMSprop
from keras import  backend as K
```

```
2.13.0
```

In [3]:
```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

In [4]:
```
# Load in the data
cifar10 = tf.keras.datasets.cifar10
# Distribute it to train and test set
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# count the number of unique train labels
unique, counts = np.unique(y_train, return_counts=True)
print("Train labels: ", dict(zip(unique, counts)))

# count the number of unique test labels
unique, counts = np.unique(y_test, return_counts=True)
print("\nTest labels: ", dict(zip(unique, counts)))
```

```
Train labels:  {0: 5000, 1: 5000, 2: 5000, 3: 5000, 4: 5000, 5: 5000, 6: 5000, 7: 5000, 8: 5000, 9: 5000}

Test labels:  {0: 1000, 1: 1000, 2: 1000, 3: 1000, 4: 1000, 5: 1000, 6: 1000, 7: 1000, 8: 1000, 9: 1000}
```

In [5]:
```
y_train
```

Out[5]:
```
array([[6],
       [9],
       [9],
       ...,
       [9],
       [1],
       [1]], dtype=uint8)
```

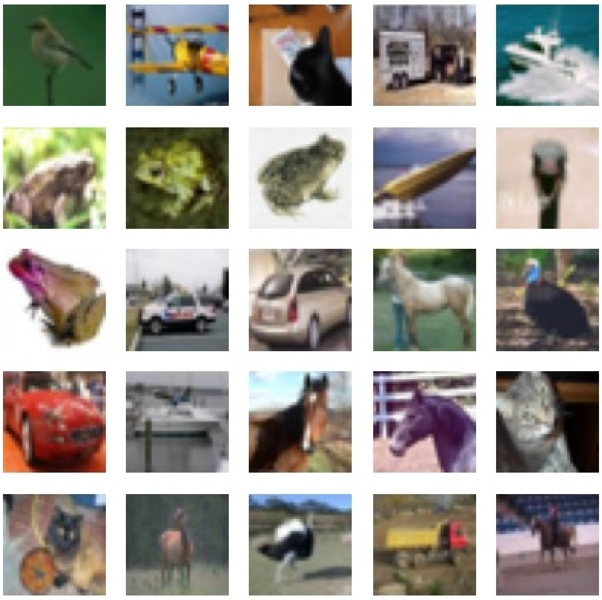
In [6]:
```
y_test
```

Out[6]:
```
array([[3],
       [8],
       [8],
       ...,
       [5],
       [1],
       [7]], dtype=uint8)
```

In [7]:
```
indexes = np.random.randint(0, x_train.shape[0], size=25)
images = x_train[indexes]
labels = y_train[indexes]

plt.figure(figsize=(5,5))
for i in range(len(indexes)):
    plt.subplot(5, 5, i + 1)
    image = images[i]
    plt.imshow(image, cmap='gray')
    plt.axis('off')

plt.show()
#plt.savefig("mnist-samples.png")
plt.close('all')
```



In [8]:
```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.utils import to_categorical, plot_model
```

In [9]:
```
# compute the number of labels
num_labels = len(np.unique(y_train))
```

In [10]:
```
# image dimensions (assumed square)
image_size = x_train.shape[1]
input_size = image_size * image_size
input_size
```

Out[10]:
```
1024
```

In [11]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

(x_train, y_train) , (x_test, y_test) = datasets.cifar10.load_data()
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255.0
x_test /= 255.0

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.InputLayer(input_shape=(32,32,3)))
model.add(tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2,2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

model.fit(x_train, y_train, batch_size=32, epochs=10) #Confined it to 10 epochs as my system is only having 4GB
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0         
 D)                                                              
                                                                 
 flatten (Flatten)           (None, 8192)              0         
                                                                 
 dense (Dense)               (None, 10)                81930     
                                                                 
=================================================================
Total params: 82826 (323.54 KB)
Trainable params: 82826 (323.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
1563/1563 [==============================] - 63s 32ms/step - loss: 1.4791 - accuracy: 0.4823
Epoch 2/10
1563/1563 [==============================] - 48s 31ms/step - loss: 1.2062 - accuracy: 0.5817
Epoch 3/10
1563/1563 [==============================] - 47s 30ms/step - loss: 1.1079 - accuracy: 0.6159
Epoch 4/10
1563/1563 [==============================] - 49s 32ms/step - loss: 1.0384 - accuracy: 0.6414
Epoch 5/10
1563/1563 [==============================] - 50s 32ms/step - loss: 0.9894 - accuracy: 0.6577
Epoch 6/10
1563/1563 [==============================] - 44s 28ms/step - loss: 0.9441 - accuracy: 0.6744
Epoch 7/10
1563/1563 [==============================] - 42s 27ms/step - loss: 0.9043 - accuracy: 0.6872
Epoch 8/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.8677 - accuracy: 0.7019
Epoch 9/10
1563/1563 [==============================] - 49s 32ms/step - loss: 0.8372 - accuracy: 0.7116
Epoch 10/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.8082 - accuracy: 0.7221
```

Out[11]:

```
<keras.src.callbacks.History at 0x189926096f0>
```

In [12]:

```
model.metrics_names
```

Out[12]:

```
['loss', 'accuracy']
```

In [13]:

```
loss, acc = model.evaluate(x_test, y_test, batch_size=32)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
```

```
313/313 [==============================] - 4s 10ms/step - loss: 1.0950 - accuracy: 0.6337

Test accuracy: 63.4%
```

In [14]:

```
from sklearn.metrics import classification_report

import numpy as np
y_predict = np.argmax(model.predict(x_test), axis=-1)
y_predict
```

```
313/313 [==============================] - 3s 9ms/step
```

Out[14]:

```
array([3, 8, 0, ..., 5, 1, 7], dtype=int64)
```