LAB - 9 IMAGE CLASSIFICATION USING CNN FOR CIFAR -10 DATA

R. SUJITHA
215229140

PART -I BASELINE MODEL

1. IMPORT LIBRARIES

In [1]:

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
from keras.utils import np_utils
```

1. LOAD YOUR DATA AND PRINT THE SHAPE OF TRAINING AND TEST SAMPLES

In [3]:

```
(X_train,y_train),(X_test, y_test)= cifar10.load_data()
print('X_train shape:',X_train.shape)
print(X_train.shape[0], 'Train Samples')
print(X_test.shape[0],'Test Samples')
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
 (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
170500096/170498071 [==============================] - 3s 0us/step
170508288/170498071 [==============================] - 3s 0us/step
X_train shape: (50000, 32, 32, 3)
50000 Train Samples
10000 Test Samples
```

1. PRINT THE SHAPE OF ONE IMAGE (IS IT 32X32X3 NUMPY ARRAY?)

In [4]:

```
X_train[444].shape
```

Out[4]:

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

yes it is 32 x32 x3 numpy array

1. DISPLAY ONE IMAGE USING IMSHOW() FUNCTION

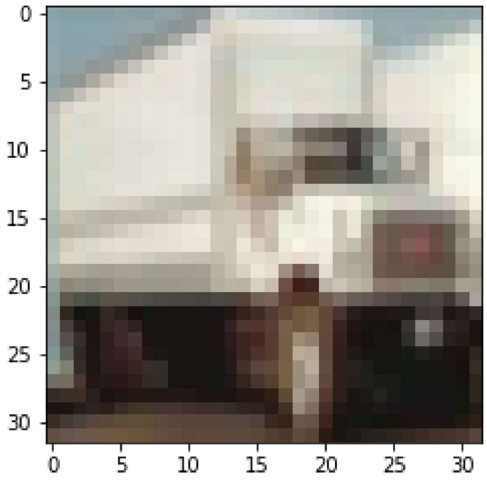
In [5]:

```
print(y_train[444])
plt.imshow(X_train[444])
```

```
[9]
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x7f28a13f6850>
```



1. CONVERT Y_TRAIN AND Y_TEST INTO CATEGORICAL VALUES

In [6]:

```
num_classes = 10
y_train = keras.utils.np_utils.to_categorical(y_train,num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test,num_classes)
```

In [7]:

```
y_train[444]
```

Out[7]:

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

1. CONVERT TRAIN DATA INTO FLOAT AND SCALE

In [8]:

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /=255
X_test /= 255
```

1. BUILD YOUR FIRST CNN

In [9]:

```
model = Sequential()
model.add(Conv2D(32,kernel_size=(5,5), strides = 2, activation='relu',padding='sa
model.add(Conv2D(32,kernel_size=(5,5), strides = 2, activation='relu',padding='sa
model.add(MaxPooling2D(2,2))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(512,input_shape=(32,32,3),activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(10,activation='softmax'))
```

1. PRINT SUMMARY AND VERIFY YOUR CONFIGURATION

In [10]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 16, 16, 32)        2432

 conv2d_1 (Conv2D)           (None, 8, 8, 32)          25632

 max_pooling2d (MaxPooling2D  (None, 4, 4, 32)         0
 )

 dropout (Dropout)           (None, 4, 4, 32)          0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 295,850
Trainable params: 295,850
Non-trainable params: 0
_________________________________________________________________
```

1. COMPILE AND FIT AND VALIDATE YOUR MODEL WITH THE FOLLOWING PARAMETERS

In [11]:

```
from tensorflow import keras
opt = keras.optimizers.RMSprop(learning_rate=0.001, decay =1e-6)
```

In [12]:

```

model.compile(loss='categorical_crossentropy',optimizer=opt,metrics =['accuracy']
```

In [13]:

```
model.fit(X_train,y_train,batch_size=32, epochs=15, shuffle=True)
```

```
Epoch 1/15
1563/1563 [==============================] - 53s 33ms/step - loss: 1.6463 - acc
uracy: 0.4033
Epoch 2/15
1563/1563 [==============================] - 47s 30ms/step - loss: 1.3625 - acc
uracy: 0.5161
Epoch 3/15
1563/1563 [==============================] - 45s 29ms/step - loss: 1.2716 - acc
uracy: 0.5519
Epoch 4/15
1563/1563 [==============================] - 44s 28ms/step - loss: 1.2394 - acc
uracy: 0.5701
Epoch 5/15
1563/1563 [==============================] - 47s 30ms/step - loss: 1.2281 - acc
uracy: 0.5773
Epoch 6/15
1563/1563 [==============================] - 46s 30ms/step - loss: 1.2310 - acc
uracy: 0.5778
Epoch 7/15
1563/1563 [==============================] - 48s 31ms/step - loss: 1.2374 - acc
uracy: 0.5802
Epoch 8/15
1563/1563 [==============================] - 46s 30ms/step - loss: 1.2547 - acc
uracy: 0.5749
Epoch 9/15
1563/1563 [==============================] - 47s 30ms/step - loss: 1.2697 - acc
uracy: 0.5722
Epoch 10/15
1563/1563 [==============================] - 46s 29ms/step - loss: 1.2909 - acc
uracy: 0.5674
Epoch 11/15
1563/1563 [==============================] - 47s 30ms/step - loss: 1.3154 - acc
uracy: 0.5609
Epoch 12/15
1563/1563 [==============================] - 46s 29ms/step - loss: 1.3341 - acc
uracy: 0.5527
Epoch 13/15
1563/1563 [==============================] - 47s 30ms/step - loss: 1.3589 - acc
uracy: 0.5424
Epoch 14/15
1563/1563 [==============================] - 45s 29ms/step - loss: 1.3726 - acc
uracy: 0.5407
Epoch 15/15
1563/1563 [==============================] - 46s 30ms/step - loss: 1.3973 - acc
uracy: 0.5333
```

Out[13]:

```
<keras.callbacks.History at 0x7f289cd50650>
```

PART -II MODEL IMPROVEMENTS

1. BUILD A MORE COMPLICATED MODEL WITH THE FOLLOWING PATTERN: CONV - CONV -MAXPOOL -CONV - CONV - MAXPOOL - FLATTEN - DENSE - FINAL CLASSIFICATION

1. USE STRIDES OF 1 FOR ALL CONVOLUTIONAL LAYERS

In [14]:
```
model1 = Sequential()
model1.add(Conv2D(32,kernel_size=(5,5), strides = 1, activation='relu',padding='s
model1.add(Conv2D(32,kernel_size=(5,5), strides = 1, activation='relu',padding='s
model1.add(MaxPooling2D(2,2))
model1.add(Conv2D(32,kernel_size=(5,5), strides = 1, activation='relu',padding='s
model1.add(Conv2D(32,kernel_size=(5,5), strides = 1, activation='relu',padding='s
model1.add(MaxPooling2D(2,2))
model1.add(Flatten())
model1.add(Dense(512,input_shape=(32,32,3),activation='relu'))
model1.add(Dense(10,activation='softmax'))
```

1. HOW MANY PARAMS DOES YOUR MODEL HAVE ? HOW DOES THAT COMPARE TO THE PREVIOUS MODEL ?

In [15]:
```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 32, 32, 32)        2432

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        25632

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 32)        25632

 conv2d_5 (Conv2D)           (None, 16, 16, 32)        25632

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 512)               1049088

 dense_3 (Dense)             (None, 10)                5130

=================================================================
Total params: 1,133,546
Trainable params: 1,133,546
Non-trainable params: 0
_________________________________________________________________
```

No of params increased

1. TRAIN IT FOR 5 EPOCHS . WHAT DO YOU NOTICE ABOUT THE TRAINING TIME, LOSS, ACCURACY NUMBERS

In [16]:

```
model1.compile(loss='categorical_crossentropy',optimizer=opt,metrics =['accuracy'
```

In [17]:

```
model1.fit(X_train,y_train,batch_size=128, epochs=5, shuffle=True)
```

```
Epoch 1/5
391/391 [==============================] - 461s 1s/step - loss: 1.7678 - accura
cy: 0.3676
Epoch 2/5
391/391 [==============================] - 459s 1s/step - loss: 1.2405 - accura
cy: 0.5608
Epoch 3/5
391/391 [==============================] - 460s 1s/step - loss: 0.9941 - accura
cy: 0.6512
Epoch 4/5
391/391 [==============================] - 471s 1s/step - loss: 0.8147 - accura
cy: 0.7167
Epoch 5/5
391/391 [==============================] - 460s 1s/step - loss: 0.6702 - accura
cy: 0.7683
```

Out[17]:

```
<keras.callbacks.History at 0x7f289cc94090>
```

1. TRY DIFFERENT STRUCTURES AND RUNTIMES

In [18]:

```
qa1model2 = Sequential()
model2.add(Conv2D(32,kernel_size=(5,5), strides = 1, activation='relu',padding='s
model2.add(MaxPooling2D(2,2))
model2.add(Flatten())
model2.add(Dense(512,input_shape=(32,32,3),activation='relu'))
model2.add(Dense(10,activation='softmax'))

model2.compile(loss='categorical_crossentropy',optimizer=opt,metrics =['accuracy'
model2.fit(X_train,y_train,batch_size=128, epochs=5, shuffle=True)
model2.summary()
```

```
Epoch 1/5
391/391 [==============================] - 92s 234ms/step - loss: 1.6348 - accu
racy: 0.4387
Epoch 2/5
391/391 [==============================] - 92s 235ms/step - loss: 1.2026 - accu
racy: 0.5778
Epoch 3/5
391/391 [==============================] - 92s 236ms/step - loss: 1.0189 - accu
racy: 0.6449
Epoch 4/5
391/391 [==============================] - 92s 235ms/step - loss: 0.8725 - accu
racy: 0.6978
Epoch 5/5
391/391 [==============================] - 101s 258ms/step - loss: 0.7446 - acc
uracy: 0.7410
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 32)        2432

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 8192)              0

 dense_4 (Dense)             (None, 512)               4194816

 dense_5 (Dense)             (None, 10)                5130

=================================================================
Total params: 4,202,378
Trainable params: 4,202,378
Non-trainable params: 0
_________________________________________________________________
```