Name : Arul Kumar ARK

Roll No. : 225229102

Lab : 9 : Image Classification using CNN for CIFAR-10 Data

Baseline Model

In [2]:

```
import pandas
```

In [3]:

```
import tensorflow as tf
```

In [4]:

```
import keras
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten
```

In [5]:

```
from __future__ import print_function
```

In [6]:

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.backend import categorical_crossentropy
```

Dataset

In [8]:

```
from keras.datasets import cifar10
```

In [9]:

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [10]:

```
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

```
50000 train samples
10000 test samples
```

Print the shape and Display

In [11]:

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

Out[11]:

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

In [12]:

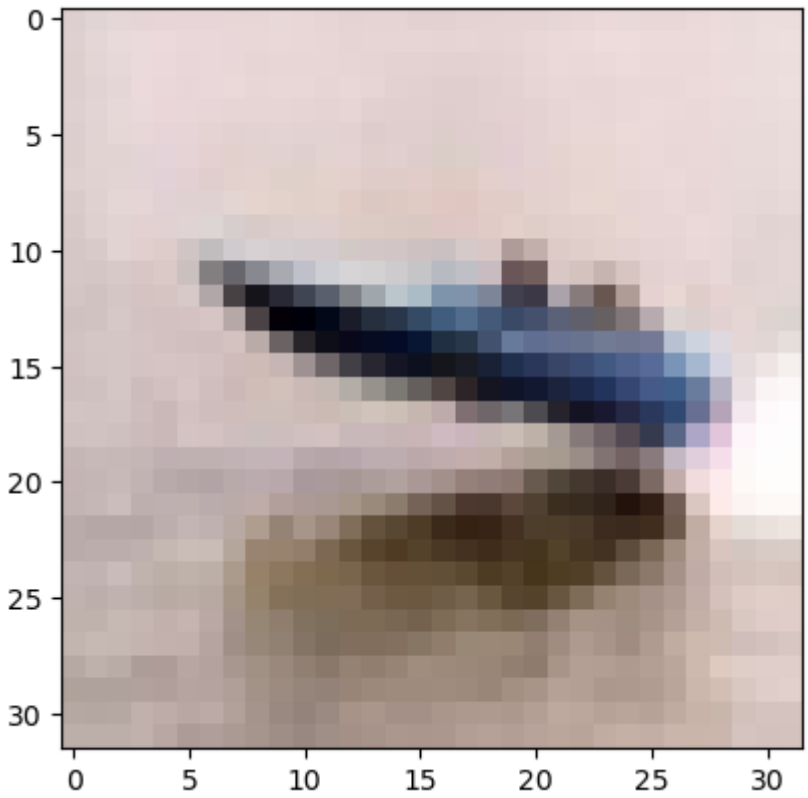
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [13]:

```
plt.imshow(X_train[441])
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x1868d5871d0>
```



Convert y_train and y_test

In [14]:

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

In [15]:

```
y_train[444]
```

Out[15]:

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

Convert train data

In [16]:

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

Build CNN Model

In [17]:

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import Adam
```

In [18]:

```
model = Sequential()
model.add(Conv2D(32, (5, 5), strides=(2, 2), activation='relu', padding='same', input_sh
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), strides=(2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

In [19]:

```
from tensorflow.keras.optimizers import legacy as legacy_optimizers
```

In [20]:

```
optimizer =tf.keras.optimizers.legacy.RMSprop(learning_rate=0.0005, decay=1e-6)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy']
```

In [21]:

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

Summary and Verify Czonfiguration

In [22]:

```
model.summary()
```

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

 max_pooling2d (MaxPooling2  (None, 8, 8, 32)          0
 D)

 conv2d_1 (Conv2D)           (None, 4, 4, 64)          18496

 max_pooling2d_1 (MaxPoolin  (None, 2, 2, 64)          0
 g2D)

 flatten (Flatten)           (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 512)               66048

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 125002 (488.29 KB)
Trainable params: 125002 (488.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [23]:

```
batch_size = 32
epochs = 15
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1,
```

```
Epoch 1/15
1407/1407 [==============================] - 102s 61ms/step - loss: 1.864
6 - accuracy: 0.3064 - val_loss: 1.7087 - val_accuracy: 0.3842
Epoch 2/15
1407/1407 [==============================] - 88s 63ms/step - loss: 1.5969
- accuracy: 0.4125 - val_loss: 1.4127 - val_accuracy: 0.4812
Epoch 3/15
1407/1407 [==============================] - 72s 51ms/step - loss: 1.4905
- accuracy: 0.4505 - val_loss: 1.3783 - val_accuracy: 0.4920
Epoch 4/15
1407/1407 [==============================] - 80s 57ms/step - loss: 1.4273
- accuracy: 0.4799 - val_loss: 1.3714 - val_accuracy: 0.4908
Epoch 5/15
1407/1407 [==============================] - 88s 63ms/step - loss: 1.3721
- accuracy: 0.5048 - val_loss: 1.2587 - val_accuracy: 0.5400
Epoch 6/15
1407/1407 [==============================] - 89s 63ms/step - loss: 1.3364
- accuracy: 0.5155 - val_loss: 1.3183 - val_accuracy: 0.5160
Epoch 7/15
1407/1407 [==============================] - 80s 57ms/step - loss: 1.3065
- accuracy: 0.5290 - val_loss: 1.4107 - val_accuracy: 0.5164
Epoch 8/15
1407/1407 [==============================] - 91s 64ms/step - loss: 1.2815
- accuracy: 0.5397 - val_loss: 1.1555 - val_accuracy: 0.5796
Epoch 9/15
1407/1407 [==============================] - 84s 59ms/step - loss: 1.2581
- accuracy: 0.5500 - val_loss: 1.1758 - val_accuracy: 0.5686
Epoch 10/15
1407/1407 [==============================] - 79s 56ms/step - loss: 1.2333
- accuracy: 0.5619 - val_loss: 1.2297 - val_accuracy: 0.5560
Epoch 11/15
1407/1407 [==============================] - 91s 65ms/step - loss: 1.2165
- accuracy: 0.5725 - val_loss: 1.1479 - val_accuracy: 0.5856
Epoch 12/15
1407/1407 [==============================] - 82s 58ms/step - loss: 1.2032
- accuracy: 0.5742 - val_loss: 1.0688 - val_accuracy: 0.6108
Epoch 13/15
1407/1407 [==============================] - 79s 56ms/step - loss: 1.1923
- accuracy: 0.5799 - val_loss: 1.0881 - val_accuracy: 0.6150
Epoch 14/15
1407/1407 [==============================] - 86s 61ms/step - loss: 1.1820
- accuracy: 0.5868 - val_loss: 1.1481 - val_accuracy: 0.5932
Epoch 15/15
1407/1407 [==============================] - 82s 58ms/step - loss: 1.1777
- accuracy: 0.5874 - val_loss: 1.0898 - val_accuracy: 0.6078
```

Out[23]:

```
<keras.src.callbacks.History at 0x1868d50b1d0>
```

In [24]:

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_accuracy}")
```

```
313/313 [==============================] - 11s 32ms/step - loss: 1.0983 -
accuracy: 0.6111
Test accuracy: 0.6111000180244446
```

Model Improvements

In [25]:

```
model1 = Sequential()
```

In [29]:

```
model1 = Sequential()
model1.add(Conv2D(filters=32, kernel_size=(5,5), strides=1, padding='same', activation='
model1.add(Conv2D(filters=32, kernel_size=(5,5), strides=1, padding='same', activation='
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Conv2D(filters=64, kernel_size=(5,5), strides=1, padding='same', activation='
model1.add(Conv2D(filters=64, kernel_size=(5,5), strides=1, padding='same', activation='
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.add(Dropout(0.25))
model1.add(Flatten())
model1.add(Dense(512, activation='relu'))
model1.add(Dropout(0.5))
model1.add(Dense(10, activation='softmax'))
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [31]:

```
history01 = model1.fit(X_train, y_train, shuffle=True, epochs=5, batch_size=32, validati
score01 = model1.evaluate(X_test, y_test, verbose=0)
```

```
Epoch 1/5
1250/1250 [==============================] - 791s 616ms/step - loss: 1.59
93 - accuracy: 0.4079 - val_loss: 1.2704 - val_accuracy: 0.5500
Epoch 2/5
1250/1250 [==============================] - 759s 607ms/step - loss: 1.23
92 - accuracy: 0.5586 - val_loss: 1.1334 - val_accuracy: 0.5967
Epoch 3/5
1250/1250 [==============================] - 445s 356ms/step - loss: 1.05
87 - accuracy: 0.6260 - val_loss: 0.9499 - val_accuracy: 0.6661
Epoch 4/5
1250/1250 [==============================] - 315s 252ms/step - loss: 0.93
71 - accuracy: 0.6696 - val_loss: 0.8695 - val_accuracy: 0.6970
Epoch 5/5
1250/1250 [==============================] - 253s 202ms/step - loss: 0.84
93 - accuracy: 0.7015 - val_loss: 0.8480 - val_accuracy: 0.7149
```

In [34]:

```
model1.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 32, 32, 32)        2432

 conv2d_11 (Conv2D)          (None, 32, 32, 32)        25632

 max_pooling2d_6 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_12 (Conv2D)          (None, 16, 16, 64)        51264

 conv2d_13 (Conv2D)          (None, 16, 16, 64)        102464

 max_pooling2d_7 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 dropout_6 (Dropout)         (None, 8, 8, 64)          0

 flatten_3 (Flatten)         (None, 4096)              0

 dense_7 (Dense)             (None, 512)               2097664

 dropout_7 (Dropout)         (None, 512)               0

 dense_8 (Dense)             (None, 10)                5130

=================================================================
Total params: 2284586 (8.72 MB)
Trainable params: 2284586 (8.72 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [35]:

```
print('Test loss:', score01[0])
print('Test accuracy:', score01[1])
```

```
Test loss: 0.8771228790283203
Test accuracy: 0.7027999758720398
```

In [ ]: