```
Assignment No.3
Name : Shivani Gaikwad
Roll No. 43315
Batch : P-11
```

In [1]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

1. Loading and Preprocessing the image data

In [2]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

In [3]:
```
print(f"train data shape : {x_train.shape}")
print(f"label train shape : {y_train.shape}")
print(f"test data shape : {x_test.shape}")
print(f"label test shape : {y_test.shape}")
```

```
train data shape : (50000, 32, 32, 3)
label train shape : (50000, 1)
test data shape : (10000, 32, 32, 3)
label test shape : (10000, 1)
```

In [4]:
```
y_train[0]
```

Out[4]:
```
array([6], dtype=uint8)
```

In [5]:
```
num_classes = 10
y_train = tf.keras.utils.to_categorical(y_train, num_classes=num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=num_classes)
```

In [6]:
```
print(f"label train shape : {y_train.shape}")
print(f"label test shape : {y_test.shape}")
```

```
label train shape : (50000, 10)
label test shape : (10000, 10)
```

In [7]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [8]:
```
def get_generator_aug():
    data_generator_aug = ImageDataGenerator(rescale=(1/255.0), rotation_range=35,
    return data_generator_aug
```

In [10]:
```
data_generator = ImageDataGenerator(rescale=(1/255.0))
data_generator.fit(x_train)
img_generator = data_generator.flow(x_train, y_train, batch_size=10, shuffle=Fals
```

2. Defining model architecture

In [11]:
```
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Model
```

In [12]:
```
def get_model(input_shape):
    input_layer = Input(input_shape)
    layer1 = Conv2D(32, 8, activation="relu", padding='SAME')(input_layer)
    layer2 = MaxPooling2D((2,2))(layer1)
    layer3 = Conv2D(32, 4, activation="relu", padding='SAME')(layer2)
    layer4 = MaxPooling2D((2,2))(layer3)
    layer5 = Flatten()(layer4)
    layer6 = Dense(16, activation="relu")(layer5)
    output_layer = Dense(10, activation="softmax")(layer6)

    model = Model(inputs=input_layer, outputs= output_layer)

    model.compile(optimizer=tf.keras.optimizers.Adam(3e-4), loss='categorical_cr

    return model
```

In [13]:
```
model = get_model((32, 32, 3))
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d (Conv2D)             (None, 32, 32, 32)        6176

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 32)        16416

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 16)                32784

 dense_1 (Dense)             (None, 10)                170

=================================================================
Total params: 55,546
Trainable params: 55,546
Non-trainable params: 0
_________________________________________________________________
```

3. Training the model

In [14]:
```
from tensorflow.keras.callbacks import EarlyStopping

train_steps_per_epoch = img_generator.n
```

In [15]:
```
history = model.fit(img_generator, steps_per_epoch=train_steps_per_epoch, validat
```

```
Epoch 1/50
 5000/50000 [==>...........................] - ETA: 19:44 - loss: 1.7033 - cate
gorical_accuracy: 0.3703WARNING:tensorflow:Your input ran out of data; interrup
ting training. Make sure that your dataset or generator can generate at least `
steps_per_epoch * epochs` batches (in this case, 2500000 batches). You may need
to use the repeat() function when building your dataset.
50000/50000 [==============================] - 220s 4ms/step - loss: 1.7033 - c
ategorical_accuracy: 0.3703 - val_loss: 201.6341 - val_categorical_accuracy: 0.
4260
```

4. Evaluating the performance

In [16]:
```
from sklearn.metrics import accuracy_score
```

In [17]:
```
y_pred = model.predict(x_test)
```

In [18]:
```
y_pred = np.argmax(y_pred, axis=1)
```

In [19]:
```
y_test = np.argmax(y_test, axis=1)
```

In [20]:
```
print(accuracy_score(y_test, y_pred))
```

```
0.426
```

In [ ]: