Assignment 3

Name: Shubham Saraf

Roll No: 43367

Division: BE11

Batch: S11

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

```
    2.5.0
```

1. Loading and Preprocessing the image data

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

```
    Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
    170500096/170498071 [==============================] - 45s 0us/step
```

```
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)
```

```
y_train[0]
```

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

```
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)
```

```
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)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
def get_generator_aug():
    data_generator_aug = ImageDataGenerator(rescale=(1/255.0),
                                            rotation_range=35,
                                            width_shift_range=0.3,
                                            height_shift_range=0.3,
                                            fill_mode='nearest',
                                            brightness_range=(0.2,0.8),
                                            shear_range=45.0,
                                            horizontal_flip=True,
                                            vertical_flip=True,
                                            zoom_range=[0.5, 1.5])
    return data_generator_aug
```

```
data_generator_aug = get_generator_aug()
data_generator_aug.fit(x_train)
train_generator_aug = data_generator_aug.flow(x_train, y_train, batch_size=10, shuffle=True)

data_generator_aug_test = get_generator_aug()
data_generator_aug_test.fit(x_test)
train_generator_aug = data_generator_aug.flow(x_train, y_train, batch_size=10, shuffle=True)
```

```
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=False)
```

2. Defining model architecture

```
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Model
```

```
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_crossentropy', metrics=[tf.keras.metrics.CategoricalAccurac
  return model
```

```
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 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
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

```
from tensorflow.keras.callbacks import EarlyStopping

train_steps_per_epoch = img_generator.n // img_generator.batch_size


history = model.fit(img_generator, steps_per_epoch=train_steps_per_epoch, validation_data=(x_
```

```
Epoch 1/10
5000/5000 [==============================] - 75s 15ms/step - loss: 1.6294 - categorical_
Epoch 2/10
5000/5000 [==============================] - 80s 16ms/step - loss: 1.2849 - categorical_
Epoch 3/10
5000/5000 [==============================] - 64s 13ms/step - loss: 1.1438 - categorical_
Epoch 4/10
5000/5000 [==============================] - 71s 14ms/step - loss: 1.0519 - categorical_
Epoch 5/10
```

```
5000/5000 [==============================] - 66s 13ms/step - loss: 0.9889 - categorical_
Epoch 6/10
5000/5000 [==============================] - 70s 14ms/step - loss: 0.9361 - categorical_
Epoch 7/10
5000/5000 [==============================] - 64s 13ms/step - loss: 0.8957 - categorical_
Epoch 8/10
5000/5000 [==============================] - 65s 13ms/step - loss: 0.8589 - categorical_
Epoch 9/10
5000/5000 [==============================] - 65s 13ms/step - loss: 0.8274 - categorical_
Epoch 10/10
5000/5000 [==============================] - 63s 13ms/step - loss: 0.7975 - categorical_
```

4. Evaluating its performance

```
from sklearn.metrics import accuracy_score


y_pred = model.predict(x_test)


y_pred = np.argmax(y_pred, axis=1)


y_test = np.argmax(y_test, axis=1)


print(accuracy_score(y_test, y_pred))
```

```
0.539
```