**Assignment - 3**
**Build CNN Model for Classification Of Flowers**

| | |
|---|---|
| Assignment submission | |
| Student Name | SIVA S |
| Student Roll Number | 951919CS092 |
| Maximum Marks | 2 Marks |

**1. Download the Dataset**

**2. Import required library**

```
import os
import zipfile
```

**3. Read dataset and do pre-processing**

```
Zip_ref = zipfile.ZipFile("/content/drive/MyDrive/IBM/Assignment - 3/Flowers-Dataset.zip")
Zip_ref.extractall("/tmp")
Zip_ref.close()
```

**4. Import required library**

```
import numpy as np
import os
import cv2
import shutil
import random as rn
from tqdm import tqdm
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

**5.Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output):**

```
data_dir ="/tmp/flowers"

print(os.listdir("/tmp/flowers"))
```

```
['sunflower', 'daisy', 'tulip', 'rose', 'dandelion']
```

```
batch_size = 32
img_height = 180
img_width = 180
```

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
data_dir,
validation_split=0.2,
subset="training",
seed=123,
image_size=(img_height, img_width),
batch_size=batch_size)
```

```
Found 4317 files belonging to 5 classes.
Using 3454 files for training.
```

---

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
data_dir,
validation_split=0.2,
subset="validation",
seed=123,
image_size=(img_height, img_width),
batch_size=batch_size)
```

```
Found 4317 files belonging to 5 classes.
Using 863 files for validation.
```

```
class_names = train_ds.class_names
print(class_names)
```

```
['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
for i in range(9):
ax = plt.subplot(3, 3, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(class_names[labels[i]])
plt.axis("off")
```


```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)



normalization_layer = layers.experimental.preprocessing.Rescaling(1./255)


normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixels values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

**6.Create the model:**

```
num_classes = 5
model = Sequential([
layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(32, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
```

```
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

## 7. Compile The Model:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

## 8. Fit The Model:

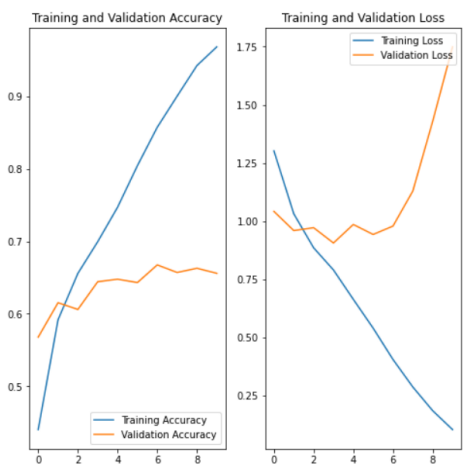
```
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
108/108 [==============================] - 96s 886ms/step - loss: 1.3016 - accuracy: 0.4404 - val_loss: 1.0411 - val_accuracy: 0.5678
Epoch 2/10
108/108 [==============================] - 94s 871ms/step - loss: 1.0308 - accuracy: 0.5918 - val_loss: 0.9587 - val_accuracy: 0.6153
Epoch 3/10
108/108 [==============================] - 93s 864ms/step - loss: 0.8844 - accuracy: 0.6558 - val_loss: 0.9710 - val_accuracy: 0.6060
Epoch 4/10
108/108 [==============================] - 103s 956ms/step - loss: 0.7890 - accuracy: 0.6995 - val_loss: 0.9051 - val_accuracy: 0.6443
Epoch 5/10
108/108 [==============================] - 95s 878ms/step - loss: 0.6627 - accuracy: 0.7470 - val_loss: 0.9848 - val_accuracy: 0.6477
Epoch 6/10
108/108 [==============================] - 94s 873ms/step - loss: 0.5386 - accuracy: 0.8043 - val_loss: 0.9419 - val_accuracy: 0.6431
Epoch 7/10
108/108 [==============================] - 93s 866ms/step - loss: 0.4038 - accuracy: 0.8573 - val_loss: 0.9779 - val_accuracy: 0.6674
Epoch 8/10
108/108 [==============================] - 94s 872ms/step - loss: 0.2854 - accuracy: 0.9001 - val_loss: 1.1288 - val_accuracy: 0.6570
Epoch 9/10
108/108 [==============================] - 94s 870ms/step - loss: 0.1834 - accuracy: 0.9424 - val_loss: 1.4286 - val_accuracy: 0.6628
Epoch 10/10
108/108 [==============================] - 94s 873ms/step - loss: 0.1022 - accuracy: 0.9682 - val_loss: 1.7492 - val_accuracy: 0.6559
```

## 9.Test the model to know the results :

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
```

```
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




**10.Image Augmentation:**

```
data_augmentation = keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal",
                                                 input_shape=(img_height,
                                                              img_width,
                                                              3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
  ]
)


plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```

**11.Save The Model:**

```
model.save('flowers_model2.h5')
from tensorflow.keras.models import load_model
model2 = load_model('flowers_model2.h5')
```