Image Classification

```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf
import pathlib

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Explore Dataset

This dataset contains images of flowers, split into 5 classes. The expected prediction is which species of flower is pictured.

```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/examp
data_dir = tf.keras.utils.get_file('flower_photos.tar', origin=dataset_url,
data_dir = pathlib.Path(data_dir).with_suffix('')

image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflo
228813984/228813984 [==============================] - 2s 0us/step
3670
```

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

```
train_ds = tf.keras.utils.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 3670 files belonging to 5 classes.
Using 2936 files for training.
```

✓ 3s completed at 5:02 PM

```
val_ds = tf.keras.utils.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 3670 files belonging to 5 classes.
Using 734 files for validation.
```

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

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
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")
```



Tune and Run Model

```
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.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]
print(np.min(first_image), np.max(first_image))
```

```
    0.0 0.9993303
```

```
num_classes = len(class_names)

model = Sequential([
  layers.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.Flatten(),
```

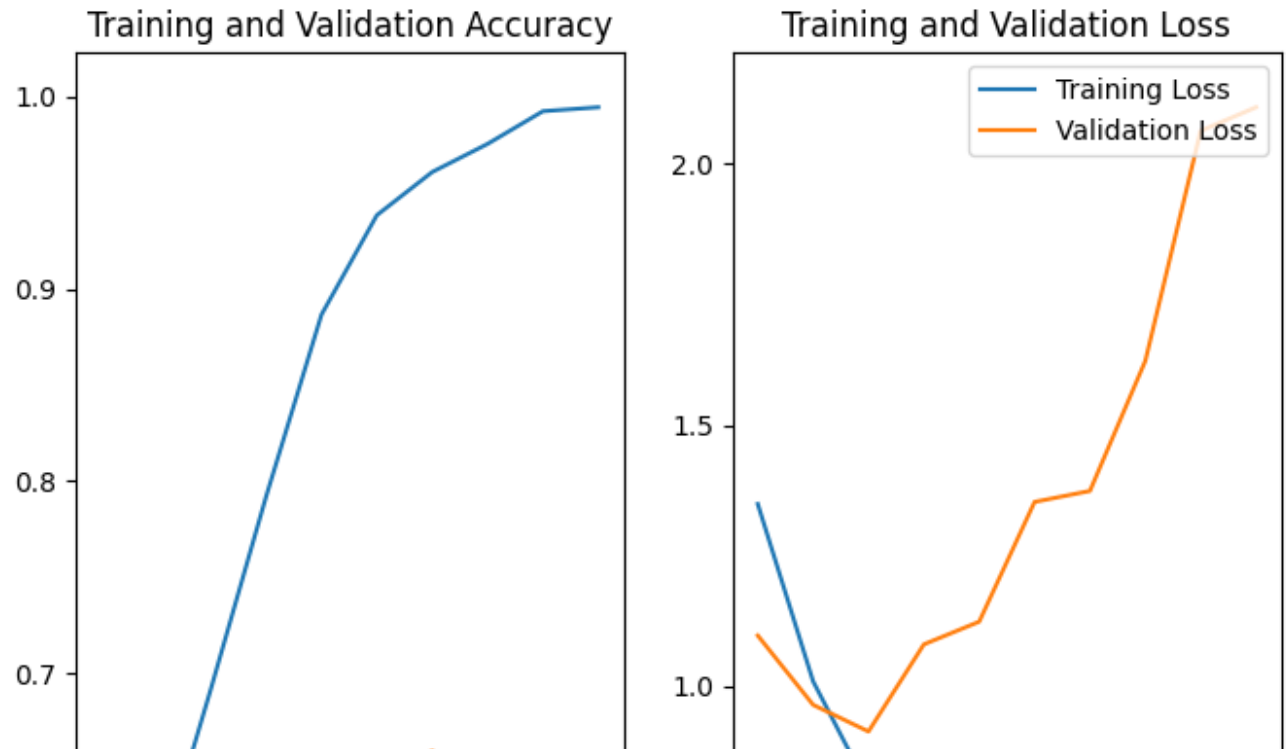
```
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])


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


model.summary()
```

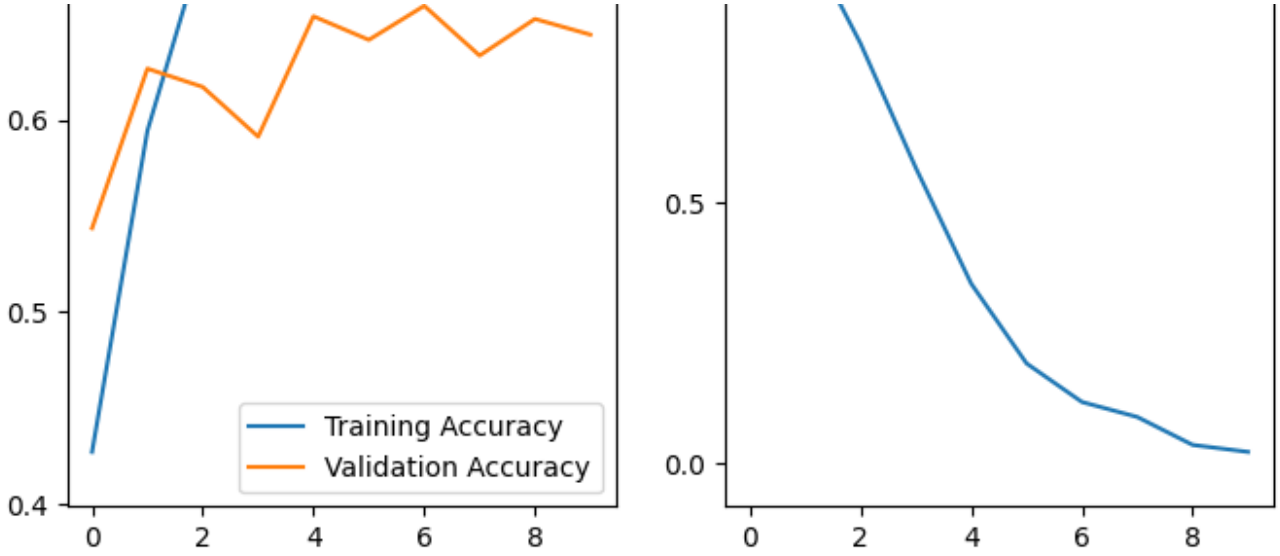
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 30976)             0

 dense_2 (Dense)             (None, 128)               3965056

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

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

```
Epoch 8/10
92/92 [==============================] - 105s 1s/step - loss: 0.0893 -
Epoch 9/10
92/92 [==============================] - 105s 1s/step - loss: 0.0358 -
Epoch 10/10
92/92 [==============================] - 110s 1s/step - loss: 0.0225 -
```

```
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()
```





ψ

```
The accuracy of this model is 60%. Th
through further training and data aug
```

The accuracy of this model is 60%. This can be improved through further training and data augmentation.

Transfer Learning

```
base_model = tf.keras.applications.MobileNetV2(input_shape=(img_height, img_
                                               include_top=False,
                                               weights='imagenet')

base_model.trainable = False
base_model.summary()
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows`
Downloading data from https://storage.googleapis.com/tensorflow/keras-a
9406464/9406464 [==============================] - 0s 0us/step
Model: "mobilenetv2_1.00_224"
```

Layer (type)	Output Shape	Param #	Connec
input_1 (InputLayer)	[(None, 180, 180, 3)]	0	[]
Conv1 (Conv2D)	(None, 90, 90, 32)	864	['inpu
bn_Conv1 (BatchNormalization)	(None, 90, 90, 32)	128	['Conv
Conv1_relu (ReLU)	(None, 90, 90, 32)	0	['bn_C

expanded_conv_depthwise (DepthwiseConv2D)	(None, 90, 90, 32)	288	['Conv
expanded_conv_depthwise_BN (BatchNormalization)	(None, 90, 90, 32)	128	['expa
expanded_conv_depthwise_relu (ReLU)	(None, 90, 90, 32)	0	['expa]']
expanded_conv_project (Conv2D)	(None, 90, 90, 16)	512	['expa [0]']
expanded_conv_project_BN (BatchNormalization)	(None, 90, 90, 16)	64	['expa
block_1_expand (Conv2D)	(None, 90, 90, 96)	1536	['expa]
block_1_expand_BN (BatchNormalization)	(None, 90, 90, 96)	384	['bloc
block_1_expand_relu (ReLU)	(None, 90, 90, 96)	0	['bloc
block_1_pad (ZeroPadding2D)	(None, 91, 91, 96)	0	['bloc
block_1_depthwise (DepthwiseConv2D)	(None, 45, 45, 96)	864	['bloc
block_1_depthwise_BN (BatchNormalization)	(None, 45, 45, 96)	384	['bloc
block_1_depthwise_relu (ReLU)	(None, 45, 45, 96)	0	['bloc
block_1_project (Conv2D)	(None, 45, 45, 24)	2304	['bloc
block_1_project_BN (BatchNormalization)	(None, 45, 45, 24)	96	['bloc
block_2_expand (Conv2D)	(None, 45, 45, 144)	3456	['bloc

Colab paid products - Cancel contracts here