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

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

```
import pathlib

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

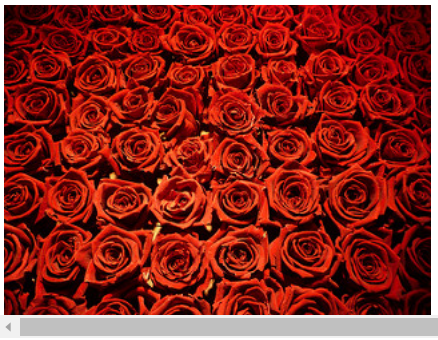
```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
```

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

```
3670
```

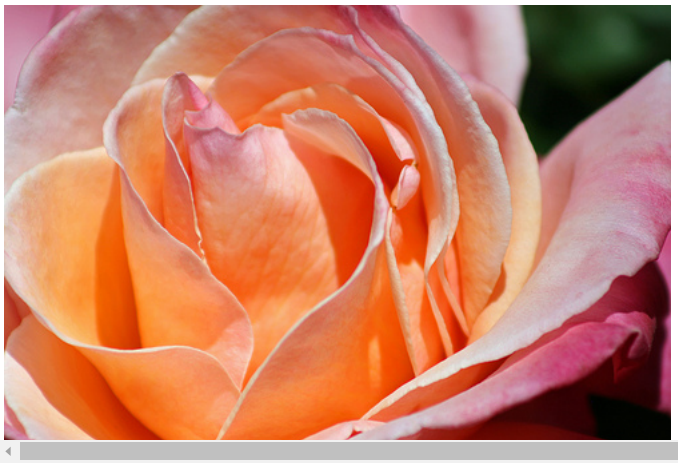
```
roses = list(data_dir.glob('roses/*'))
PIL.Image.open(str(roses[0]))
```

```
PIL.Image.open(str(roses[1]))
```

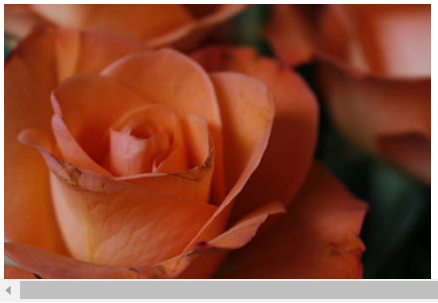


```
PIL.Image.open(str(roses[2]))
```

```
PIL.Image.open(str(roses[4]))
```

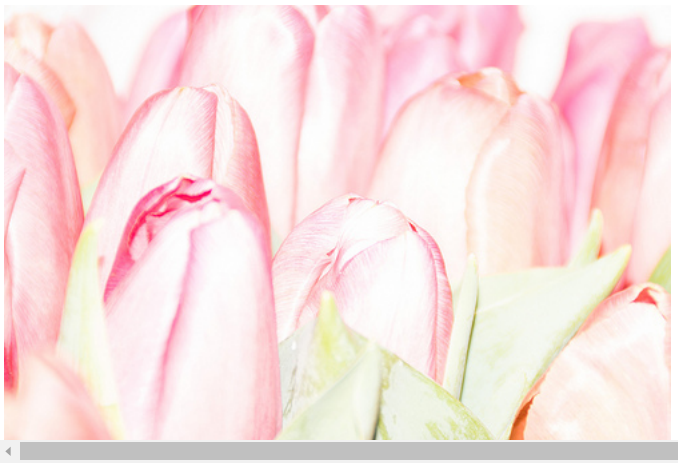


```
PIL.Image.open(str(roses[6]))
```



```
tulips = list(data_dir.glob('tulips/*'))
PIL.Image.open(str(tulips[0]))
```

```
PIL.Image.open(str(tulips[8]))
```



```
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.
```

```
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)
```

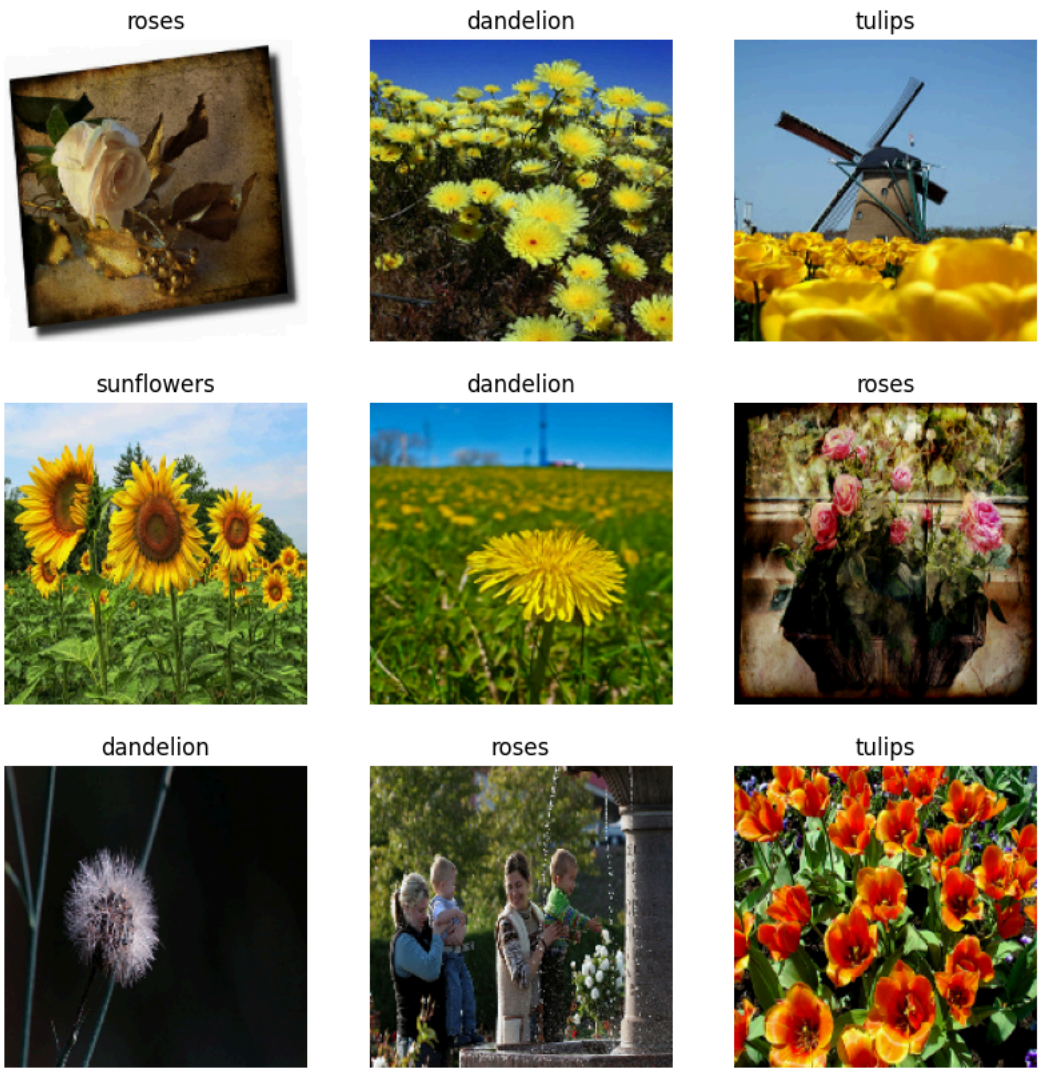
```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

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

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

```
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")
```




```
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
(32, 180, 180, 3)
(32,)
```

```
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]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

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

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/preprocessing/tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`i
  super().__init__(**kwargs)
```

Start coding or generate with AI.

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

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3,965,056
dense_1 (Dense)	(None, 5)	645

Total params: 3,989,285 (15.22 MB)
Trainable params: 3,989,285 (15.22 MB)
Non-trainable params: 0 (0.00 B)

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

```
Epoch 1/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 120s 1s/step - accuracy: 0.3325 - loss: 1.8630 - val_accuracy: 0.5218 - val_loss: 1.1903
Epoch 2/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 112s 1s/step - accuracy: 0.5256 - loss: 1.1126 - val_accuracy: 0.5858 - val_loss: 1.0183
Epoch 3/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 103s 1s/step - accuracy: 0.6572 - loss: 0.9332 - val_accuracy: 0.6362 - val_loss: 0.9392
Epoch 4/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 143s 1s/step - accuracy: 0.7217 - loss: 0.7370 - val_accuracy: 0.6580 - val_loss: 0.8976
Epoch 5/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 105s 1s/step - accuracy: 0.8161 - loss: 0.5304 - val_accuracy: 0.6390 - val_loss: 0.9796
Epoch 6/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 153s 1s/step - accuracy: 0.8730 - loss: 0.3558 - val_accuracy: 0.6485 - val_loss: 1.0056
Epoch 7/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 112s 1s/step - accuracy: 0.9247 - loss: 0.2347 - val_accuracy: 0.6403 - val_loss: 1.2627
Epoch 8/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 137s 1s/step - accuracy: 0.9633 - loss: 0.1237 - val_accuracy: 0.6540 - val_loss: 1.5436
Epoch 9/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 143s 1s/step - accuracy: 0.9747 - loss: 0.0860 - val_accuracy: 0.6431 - val_loss: 1.7792
Epoch 10/10
92/92 ━━━━━━━━━━━━━━━━━━━━ 138s 1s/step - accuracy: 0.9903 - loss: 0.0403 - val_accuracy: 0.6458 - val_loss: 1.9156
```

```
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()
```


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

```
model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255),
  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.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes, name="outputs")
])
```

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

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential_1 (Sequential)	(None, 180, 180, 3)	0
rescaling_2 (Rescaling)	?	0 (unbuilt)
conv2d_3 (Conv2D)	?	0 (unbuilt)
max_pooling2d_3 (MaxPooling2D)	?	0 (unbuilt)
conv2d_4 (Conv2D)	?	0 (unbuilt)
max_pooling2d_4 (MaxPooling2D)	?	0 (unbuilt)
conv2d_5 (Conv2D)	?	0 (unbuilt)
max_pooling2d_5 (MaxPooling2D)	?	0 (unbuilt)

```
epochs = 15
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 128s 1s/step - accuracy: 0.3874 - loss: 1.4080 - val_accuracy: 0.5613 - val_loss: 1.0745
Epoch 2/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 124s 1s/step - accuracy: 0.6060 - loss: 1.0335 - val_accuracy: 0.6431 - val_loss: 0.9357
Epoch 3/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 142s 1s/step - accuracy: 0.6466 - loss: 0.9031 - val_accuracy: 0.5967 - val_loss: 1.0082
Epoch 4/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 139s 1s/step - accuracy: 0.6799 - loss: 0.8374 - val_accuracy: 0.6567 - val_loss: 0.8291
Epoch 5/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 142s 1s/step - accuracy: 0.6969 - loss: 0.7854 - val_accuracy: 0.6826 - val_loss: 0.8077
Epoch 6/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 141s 1s/step - accuracy: 0.7211 - loss: 0.7135 - val_accuracy: 0.7016 - val_loss: 0.8000
Epoch 7/15
92/92 ━━━━━━━━━━━━━━━━━━━━ 120s 1s/step - accuracy: 0.7335 - loss: 0.7056 - val_accuracy: 0.7084 - val_loss: 0.7428
Epoch 8/15
```