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


_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'

path_to_zip = tf.keras.utils.get_file('cats_and_dogs.zip', origin=_URL, extract=True)
PATH = os.path.join(os.path.dirname(path_to_zip), 'cats_and_dogs_filtered')

train_dir = os.path.join(PATH, 'train')
validation_dir = os.path.join(PATH, 'validation')

BATCH_SIZE = 32
IMG_SIZE = (160, 160)

train_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,shuffle=True,batch_s
```

```
Downloading data from https://storage.googleapis.com/mledu-datasets/cats_and_dogs_fi
68606236/68606236 [==============================] - 0s 0us/step
Found 2000 files belonging to 2 classes.
```

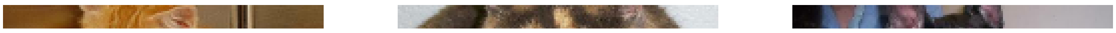
```
validation_dataset = tf.keras.utils.image_dataset_from_directory(validation_dir,
                                                                 shuffle=True,
                                                                 batch_size=BATCH_SIZE,
                                                                 image_size=IMG_SIZE)
```

```
Found 1000 files belonging to 2 classes.
```

```
class_names = train_dataset.class_names

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.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")
```


```
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
```



```
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_dat
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 26
Number of test batches: 6
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2),
])
```

```
for image, _ in train_dataset.take(1):
  plt.figure(figsize=(10, 10))
  first_image = image[0]
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    augmented_image = data_augmentation(tf.expand_dims(first_image, 0))
    plt.imshow(augmented_image[0] / 255)
    plt.axis('off')
```

```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

```
rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)
```

```
# Create the base model from the pre-trained model MobileNet V2
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mc
9406464/9406464 [==============================] - 0s 0us/step
```

```
image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)
print(feature_batch.shape)
```

```
(32, 5, 5, 1280)
```

```
base_model.trainable = False
```

```
# Let's take a look at the base model architecture
```

```
base_model.summary()
```

```
Model: "mobilenetv2_1.00_160"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 input_1 (InputLayer)           [(None, 160, 160, 3  0           []                               
                                )]                                                                
                                                                                                  
 Conv1 (Conv2D)                 (None, 80, 80, 32)   864         ['input_1[0][0]']
                                                                                                  
 bn_Conv1 (BatchNormalization)  (None, 80, 80, 32)   128         ['Conv1[0][0]']
                                                                                                  
 Conv1_relu (ReLU)              (None, 80, 80, 32)   0           ['bn_Conv1[0][0]
                                                                                                  
 expanded_conv_depthwise (Depth  (None, 80, 80, 32)  288         ['Conv1_relu[0][(
 wiseConv2D)                                                                                      
                                                                                                  
 expanded_conv_depthwise_BN (Ba  (None, 80, 80, 32)  128         ['expanded_conv_(
 tchNormalization)                                                                                
                                                                                                  
 expanded_conv_depthwise_relu (  (None, 80, 80, 32)  0           ['expanded_conv_(
 ReLU)                                                           ]']
                                                                                                  
 expanded_conv_project (Conv2D)  (None, 80, 80, 16)  512         ['expanded_conv_(
                                                                 [0]']
                                                                                                  
 expanded_conv_project_BN (Batc  (None, 80, 80, 16)  64          ['expanded_conv_p
 hNormalization)                                                                                  
                                                                                                  
 block_1_expand (Conv2D)        (None, 80, 80, 96)   1536        ['expanded_conv_p
                                                                 ]
                                                                                                  
 block_1_expand_BN (BatchNormal  (None, 80, 80, 96)  384         ['block_1_expand[
 ization)                                                                                         
                                                                                                  
 block_1_expand_relu (ReLU)     (None, 80, 80, 96)   0           ['block_1_expand_
                                                                                                  
 block_1_pad (ZeroPadding2D)    (None, 81, 81, 96)   0           ['block_1_expand_
                                                                                                  
 block_1_depthwise (DepthwiseCo  (None, 40, 40, 96)  864         ['block_1_pad[0][
 nv2D)                                                                                            
                                                                                                  
 block_1_depthwise_BN (BatchNor  (None, 40, 40, 96)  384         ['block_1_depthwi
 malization)                                                                                      
                                                                                                  
 block_1_depthwise_relu (ReLU)  (None, 40, 40, 96)   0           ['block_1_depthwi
                                                                                                  
 block_1_project (Conv2D)       (None, 40, 40, 24)   2304        ['block_1_depthwi
                                                                                                  
 block_1_project_BN (BatchNorma  (None, 40, 40, 24)  96          ['block_1_project
 lization)                                                                                        
                                                                                                  
 block_2_expand (Conv2D)        (None, 40, 40, 144)  3456        ['block_1_project
                                                                                                  
 block_2_expand_BN (BatchNormal  (None, 40, 40, 144)  576        ['block_2_expand[
 ization)                                                                                         
                                                                                                  
 block_2_expand_relu (ReLU)     (None, 40, 40, 144)  0           ['block_2_expan
```

```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
```

```
(32, 1280)
```

```
prediction_layer = tf.keras.layers.Dense(1)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
```

```
(32, 1)
```

```
inputs = tf.keras.Input(shape=(160, 160, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

```
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 160, 160, 3)]     0         
                                                                 
 sequential (Sequential)     (None, 160, 160, 3)       0         
                                                                 
 tf.math.truediv (TFOpLambda  (None, 160, 160, 3)      0         
 )                                                               
                                                                 
 tf.math.subtract (TFOpLambd  (None, 160, 160, 3)      0         
 a)                                                              
                                                                 
 mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)       2257984   
 ional)                                                          
                                                                 
 global_average_pooling2d (G  (None, 1280)             0         
 lobalAveragePooling2D)                                          
                                                                 
 dropout (Dropout)           (None, 1280)              0         
                                                                 
 dense (Dense)               (None, 1)                 1281      
                                                                 
=================================================================
```

```
Total params: 2,259,265
Trainable params: 1,281
Non-trainable params: 2,257,984
_________________________________________________________________
```

```
len(model.trainable_variables)
```

```
2
```

```
initial_epochs = 10

loss0, accuracy0 = model.evaluate(validation_dataset)
```

```
26/26 [==============================] - 24s 742ms/step - loss: 0.9479 - accuracy: 0
```

```
print("initial loss: {:.2f}".format(loss0))
print("initial accuracy: {:.2f}".format(accuracy0))
```

```
initial loss: 0.95
initial accuracy: 0.33
```

```
history = model.fit(train_dataset,
                    epochs=initial_epochs,
                    validation_data=validation_dataset)
```

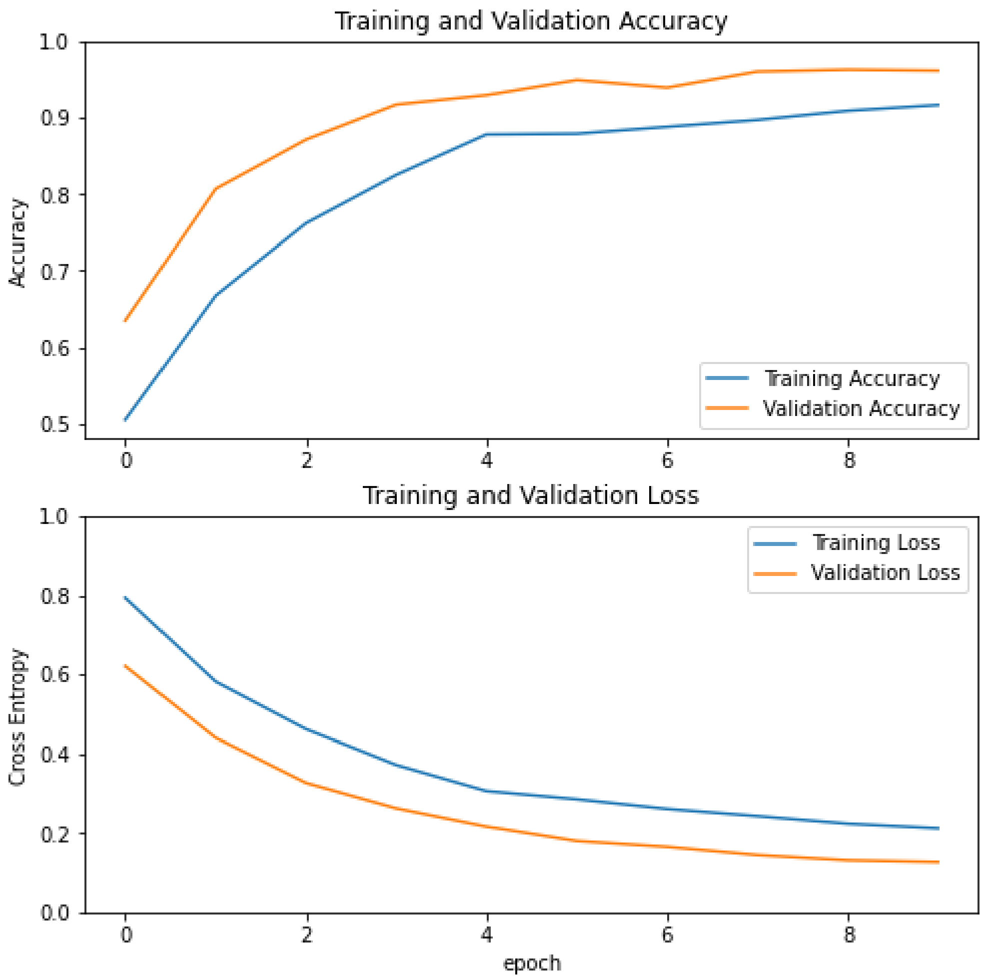
```
Epoch 1/10
63/63 [==============================] - 69s 1s/step - loss: 0.7938 - accuracy: 0.505
Epoch 2/10
63/63 [==============================] - 63s 994ms/step - loss: 0.5820 - accuracy: 0
Epoch 3/10
63/63 [==============================] - 61s 961ms/step - loss: 0.4624 - accuracy: 0
Epoch 4/10
63/63 [==============================] - 58s 914ms/step - loss: 0.3716 - accuracy: 0
Epoch 5/10
63/63 [==============================] - 58s 924ms/step - loss: 0.3060 - accuracy: 0
Epoch 6/10
63/63 [==============================] - 60s 949ms/step - loss: 0.2852 - accuracy: 0
Epoch 7/10
63/63 [==============================] - 59s 931ms/step - loss: 0.2608 - accuracy: 0
Epoch 8/10
63/63 [==============================] - 58s 926ms/step - loss: 0.2427 - accuracy: 0
Epoch 9/10
63/63 [==============================] - 60s 950ms/step - loss: 0.2235 - accuracy: 0
Epoch 10/10
63/63 [==============================] - 58s 923ms/step - loss: 0.2120 - accuracy: 0
```

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

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

```
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```




```
base_model.trainable = True
```

```
# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
```

```
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable = False
```

```
Number of layers in the base model:  154
```

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer = tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate/10)
              metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 160, 160, 3)]     0

 sequential (Sequential)     (None, 160, 160, 3)       0

 tf.math.truediv (TFOpLambda  (None, 160, 160, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 160, 160, 3)      0
 a)

 mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 1)                 1281

=================================================================
Total params: 2,259,265
Trainable params: 1,862,721
Non-trainable params: 396,544
_________________________________________________________________
```

```
len(model.trainable_variables)
```

```
56
```

```
fine_tune_epochs = 10
total_epochs =  initial_epochs + fine_tune_epochs
history_fine = model.fit(train_dataset,epochs=total_epochs,initial_epoch=history.epoch[-1]
```

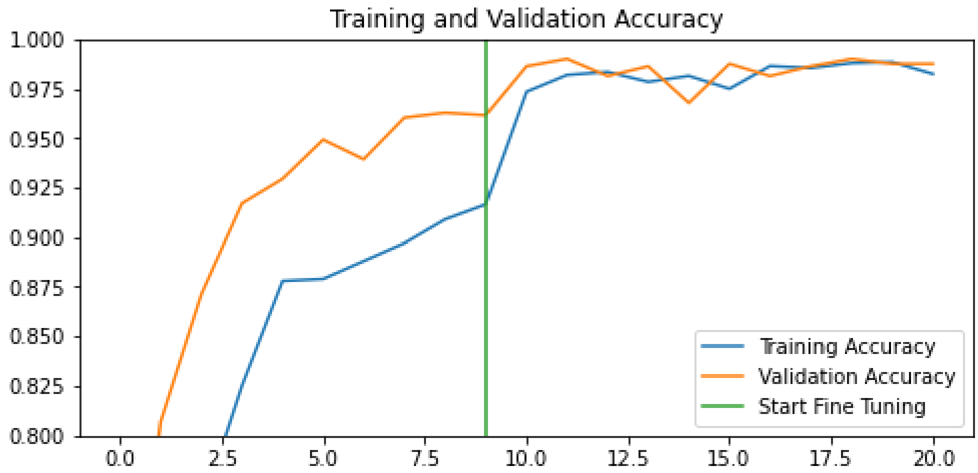
```
Epoch 10/20
63/63 [==============================] - 86s 1s/step - loss: 0.0625 - accuracy: 0.97
Epoch 11/20
```

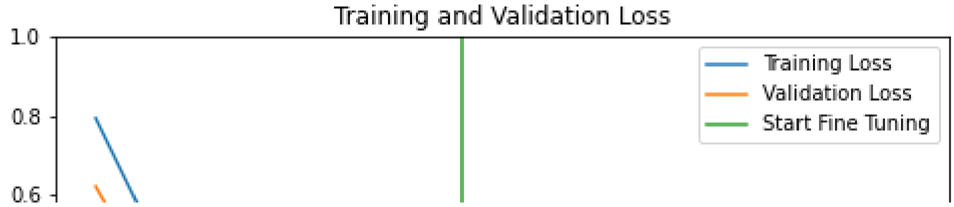
```
63/63 [==============================] - 87s 1s/step - loss: 0.0453 - accuracy: 0.982
Epoch 12/20
63/63 [==============================] - 85s 1s/step - loss: 0.0432 - accuracy: 0.983
Epoch 13/20
63/63 [==============================] - 86s 1s/step - loss: 0.0569 - accuracy: 0.978
Epoch 14/20
63/63 [==============================] - 86s 1s/step - loss: 0.0504 - accuracy: 0.981
Epoch 15/20
63/63 [==============================] - 85s 1s/step - loss: 0.0542 - accuracy: 0.975
Epoch 16/20
63/63 [==============================] - 85s 1s/step - loss: 0.0396 - accuracy: 0.986
Epoch 17/20
63/63 [==============================] - 84s 1s/step - loss: 0.0398 - accuracy: 0.985
Epoch 18/20
63/63 [==============================] - 85s 1s/step - loss: 0.0338 - accuracy: 0.988
Epoch 19/20
63/63 [==============================] - 87s 1s/step - loss: 0.0319 - accuracy: 0.988
Epoch 20/20
63/63 [==============================] - 83s 1s/step - loss: 0.0419 - accuracy: 0.982
```

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

loss += history_fine.history['loss']
val_loss += history_fine.history['val_loss']
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.ylim([0.8, 1])
plt.plot([initial_epochs-1,initial_epochs-1],
          plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1.0])
plt.plot([initial_epochs-1,initial_epochs-1],
         plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```






```
base_model.trainable = True
```

```
# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable = False
```

```
Number of layers in the base model:  154
```

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer = tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate/10)
              metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 160, 160, 3)]     0

 sequential (Sequential)     (None, 160, 160, 3)       0

 tf.math.truediv (TFOpLambda  (None, 160, 160, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 160, 160, 3)      0
 a)

 mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)       2257984
```

```
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 1)                 1281

=================================================================
Total params: 2,259,265
Trainable params: 1,862,721
Non-trainable params: 396,544
_________________________________________________________________
```

```
len(model.trainable_variables)
```

```
56
```

```
fine_tune_epochs = 10
total_epochs =  initial_epochs + fine_tune_epochs

history_fine = model.fit(train_dataset,
                         epochs=total_epochs,
                         initial_epoch=history.epoch[-1],
                         validation_data=validation_dataset)
```

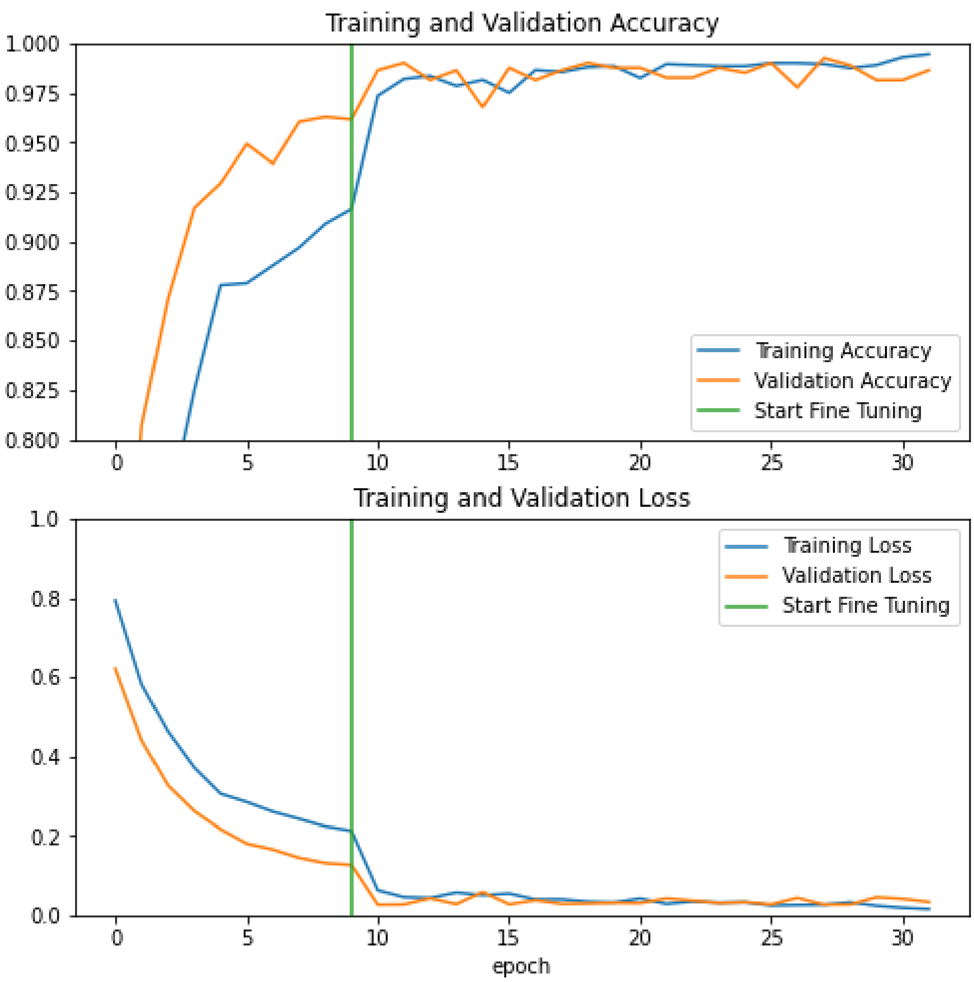
```
Epoch 10/20
63/63 [==============================] - 93s 1s/step - loss: 0.0285 - accuracy: 0.989
Epoch 11/20
63/63 [==============================] - 83s 1s/step - loss: 0.0344 - accuracy: 0.989
Epoch 12/20
63/63 [==============================] - 86s 1s/step - loss: 0.0313 - accuracy: 0.988
Epoch 13/20
63/63 [==============================] - 84s 1s/step - loss: 0.0326 - accuracy: 0.988
Epoch 14/20
63/63 [==============================] - 84s 1s/step - loss: 0.0244 - accuracy: 0.996
Epoch 15/20
63/63 [==============================] - 86s 1s/step - loss: 0.0254 - accuracy: 0.996
Epoch 16/20
63/63 [==============================] - 88s 1s/step - loss: 0.0261 - accuracy: 0.989
Epoch 17/20
63/63 [==============================] - 84s 1s/step - loss: 0.0323 - accuracy: 0.987
Epoch 18/20
63/63 [==============================] - 83s 1s/step - loss: 0.0237 - accuracy: 0.989
Epoch 19/20
63/63 [==============================] - 87s 1s/step - loss: 0.0184 - accuracy: 0.993
Epoch 20/20
63/63 [==============================] - 84s 1s/step - loss: 0.0153 - accuracy: 0.994
```

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

loss += history_fine.history['loss']
val_loss += history_fine.history['val_loss']
```

```
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.ylim([0.8, 1])
plt.plot([initial_epochs-1,initial_epochs-1],
          plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1.0])
plt.plot([initial_epochs-1,initial_epochs-1],
         plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```




```
loss, accuracy = model.evaluate(test_dataset)
print('Test accuracy :', accuracy)
```

```
6/6 [==============================] - 6s 856ms/step - loss: 0.0223 - accuracy: 0.994
```

```
Test accuracy : 0.9947916865348816
```

```
# Retrieve a batch of images from the test set
image_batch, label_batch = test_dataset.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch).flatten()

# Apply a sigmoid since our model returns logits
predictions = tf.nn.sigmoid(predictions)
predictions = tf.where(predictions < 0.5, 0, 1)

print('Predictions:\n', predictions.numpy())
print('Labels:\n', label_batch)

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

```
Predictions:
 [0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0]
Labels:
 [0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0]
```

✓ 4s completed at 8:48 PM