Import Data

```
 import zipfile

# Download zip file
 !wget https://storage.googleapis.com/ztm_tf_course/food_vision/pizza_steak.zip

 # Unzip file
 zip_ref = zipfile.ZipFile('pizza_steak.zip')
 zip_ref.extractall()
 zip_ref.close()
```

```
--2022-09-21 03:59:58--  https://storage.googleapis.com/ztm_tf_course/food_vision/pizza_
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.127.128, 142.251.18
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.127.128|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 109540975 (104M) [application/zip]
Saving to: ‘pizza_steak.zip’

pizza_steak.zip     100%[===================>] 104.47M  57.3MB/s    in 1.8s

2022-09-21 04:00:01 (57.3 MB/s) - ‘pizza_steak.zip’ saved [109540975/109540975]
```

Inspect the File

```
!ls pizza_steak
```

```
test  train
```

```
# Identify the path
!ls pizza_steak/train/steak/
```

```
1400760.jpg  1976160.jpg  2534567.jpg  3115772.jpg  3671877.jpg  703356.jpg
1403005.jpg  1984271.jpg  2535431.jpg  3116018.jpg  368073.jpg   703909.jpg
1404770.jpg  1987213.jpg  2535456.jpg  3128952.jpg  368162.jpg   704316.jpg
140832.jpg   1987639.jpg  2538000.jpg  3130412.jpg  368170.jpg   714298.jpg
141056.jpg   1995118.jpg  2543081.jpg  3136.jpg     3693649.jpg  720060.jpg
141135.jpg   1995252.jpg  2544643.jpg  313851.jpg   3700079.jpg  726083.jpg
1413972.jpg  199754.jpg   2547797.jpg  3140083.jpg  3704103.jpg  728020.jpg
1421393.jpg  2002400.jpg  2548974.jpg  3140147.jpg  3707493.jpg  732986.jpg
1428947.jpg  2011264.jpg  2549316.jpg  3142045.jpg  3716881.jpg  734445.jpg
1433912.jpg  2012996.jpg  2561199.jpg  3142618.jpg  3724677.jpg  735441.jpg
143490.jpg   2013535.jpg  2563233.jpg  3142674.jpg  3727036.jpg  740090.jpg
1445352.jpg  2017387.jpg  256592.jpg   3143192.jpg  3727491.jpg  745189.jpg
1446401.jpg  2018173.jpg  2568848.jpg  314359.jpg   3736065.jpg  752203.jpg
1453991.jpg  2020613.jpg  2573392.jpg  3157832.jpg  37384.jpg    75537.jpg
1456841.jpg  2032669.jpg  2592401.jpg  3159818.jpg  3743286.jpg  756655.jpg
146833.jpg   203450.jpg   2599817.jpg  3162376.jpg  3745515.jpg  762210.jpg
```

```
140855.jpg   205456.jpg   2599817.jpg   3162576.jpg   3745515.jpg   762210.jpg
1476404.jpg  2034628.jpg  2603058.jpg  3168620.jpg  3750472.jpg  763690.jpg
1485083.jpg  2036920.jpg  2606444.jpg  3171085.jpg  3752362.jpg  767442.jpg
1487113.jpg  2038418.jpg  2614189.jpg  317206.jpg   3766099.jpg  786409.jpg
148916.jpg   2042975.jpg  2614649.jpg  3173444.jpg  3770370.jpg  80215.jpg
149087.jpg   2045647.jpg  2615718.jpg  3180182.jpg  377190.jpg   802348.jpg
1493169.jpg  2050584.jpg  2619625.jpg  31881.jpg    3777020.jpg  804684.jpg
149682.jpg   2052542.jpg  2622140.jpg  3191589.jpg  3777482.jpg  812163.jpg
1508094.jpg  2056627.jpg  262321.jpg   3204977.jpg  3781152.jpg  813486.jpg
1512226.jpg  2062248.jpg  2625330.jpg  320658.jpg   3787809.jpg  819027.jpg
1512347.jpg  2081995.jpg  2628106.jpg  3209173.jpg  3788729.jpg  822550.jpg
1524526.jpg  2087958.jpg  2629750.jpg  3223400.jpg  3790962.jpg  823766.jpg
1530833.jpg  2088030.jpg  2643906.jpg  3223601.jpg  3792514.jpg  827764.jpg
1539499.jpg  2088195.jpg  2644457.jpg  3241894.jpg  379737.jpg   830007.jpg
1541672.jpg  2090493.jpg  2648423.jpg  3245533.jpg  3807440.jpg  838344.jpg
1548239.jpg  2090504.jpg  2651300.jpg  3245622.jpg  381162.jpg   853327.jpg
1550997.jpg  2125877.jpg  2653594.jpg  3247009.jpg  3812039.jpg  854150.jpg
1552530.jpg  2129685.jpg  2661577.jpg  3253588.jpg  3829392.jpg  864997.jpg
15580.jpg    2133717.jpg  2668916.jpg  3260624.jpg  3830872.jpg  885571.jpg
1559052.jpg  2136662.jpg  268444.jpg   326587.jpg   38442.jpg    907107.jpg
1563266.jpg  213765.jpg   2691461.jpg  32693.jpg    3855584.jpg  908261.jpg
1567554.jpg  2138335.jpg  2706403.jpg  3271253.jpg  3857508.jpg  910672.jpg
1575322.jpg  2140776.jpg  270687.jpg   3274423.jpg  386335.jpg   911803.jpg
1588879.jpg  214320.jpg   2707522.jpg  3280453.jpg  3867460.jpg  91432.jpg
1594719.jpg  2146963.jpg  2711806.jpg  3298495.jpg  3868959.jpg  914570.jpg
1595869.jpg  215222.jpg   2716993.jpg  330182.jpg   3869679.jpg  922752.jpg
1598345.jpg  2154126.jpg  2724554.jpg  3306627.jpg  388776.jpg   923772.jpg
1598885.jpg  2154779.jpg  2738227.jpg  3315727.jpg  3890465.jpg  926414.jpg
1600179.jpg  2159975.jpg  2748917.jpg  331860.jpg   3894222.jpg  931356.jpg
1600794.jpg  2163079.jpg  2760475.jpg  332232.jpg   3895825.jpg  937133.jpg
160552.jpg   217250.jpg   2761427.jpg  3322909.jpg  389739.jpg   945791.jpg
1606596.jpg  2172600.jpg  2765887.jpg  332557.jpg   3916407.jpg  947877.jpg
1615395.jpg  2173084.jpg  2768451.jpg  3326734.jpg  393349.jpg   952407.jpg
1618011.jpg  217996.jpg   2771149.jpg  3330642.jpg  393494.jpg   952437.jpg
1619357.jpg  2193684.jpg  2779040.jpg  3333128.jpg  398288.jpg   955466.jpg
1621763.jpg  220341.jpg   2788312.jpg  3333735.jpg  40094.jpg    9555.jpg
1623325.jpg  22080.jpg    2788759.jpg  3334973.jpg  401094.jpg   961341.jpg
1624450.jpg  2216146.jpg  2796102.jpg  3335013.jpg  401144.jpg   97656.jpg
1624747.jpg  2222018.jpg  280284.jpg   3335267.jpg  401651.jpg   979110.jpg
1628861.jpg  2223787.jpg  2807888.jpg  3346787.jpg  405173.jpg   980247.jpg
1632774.jpg  2230959.jpg  2815172.jpg  3364420.jpg  405794.jpg   982988.jpg
1636831.jpg  2232310.jpg  2818805.jpg  336637.jpg   40762.jpg    987732.jpg
1645470.jpg  2233395.jpg  2823872.jpg  3372616.jpg  413325.jpg   996684.jpg
```

```
import os

# list number of files in directory
for dirpath,dirnames,filenames in os.walk('pizza_steak'):
  print(f"There are {len(dirnames)} directories and {len(filenames)} in '{dirpath}'.")
```

```
There are 2 directories and 0 in 'pizza_steak'.
There are 2 directories and 0 in 'pizza_steak/test'.
There are 0 directories and 250 in 'pizza_steak/test/steak'.
There are 0 directories and 250 in 'pizza_steak/test/pizza'.
There are 2 directories and 0 in 'pizza_steak/train'.
```

```
    There are 0 directories and 750 in 'pizza_steak/train/steak'.
    There are 0 directories and 750 in 'pizza_steak/train/pizza'.
```

```
num_steak_images_train = len(os.listdir('pizza_steak/train/steak'))
```

```
# Number of steak training images
num_steak_images_train
```

```
    750
```

```
import pathlib
import numpy as np
data_dir = pathlib.Path('pizza_steak/train')
class_names = np.array(sorted([item.name for item in data_dir.glob('*')]))
```

```
print(class_names)
```

```
    ['pizza' 'steak']
```

View an image

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random

def view_random_image(target_dir, target_class):
  # create target directory
  target_folder = target_dir + '/' + target_class

  random_image = random.sample(os.listdir(target_folder), 1)

  img = mpimg.imread(target_folder + '/' + random_image[0])

  plt.imshow(img)

  plt.title(target_class)
  plt.axis('off');

  print(f"Image shape: {img.shape}")
  return img
```

```
# view a random image from the data set
img = view_random_image(target_dir = 'pizza_steak/train/',
                  target_class='steak')
```

```
Image shape: (512, 512, 3)
```


```
plt.figure()
plt.subplot(1,2,1)
steak_img = view_random_image('pizza_steak/train/','steak') # View an image from the steak tr
plt.subplot(1,2,2)
pizza_img = view_random_image('pizza_steak/train/','pizza') # View an image from the pizza tr
```

```
Image shape: (384, 512, 3)
Image shape: (512, 512, 3)
```


```
# view image data
import tensorflow as tf
tf.constant(img)
```

```
<tf.Tensor: shape=(512, 512, 3), dtype=uint8, numpy=
array([[[165, 117,  71],
        [172, 124,  78],
        [176, 129,  85],
        ...,
        [138,  14,   4],
        [154,  41,  23],
        [189,  84,  62]],

       [[159, 108,  65],
        [161, 110,  67],
        [159, 110,  67],
        ...,
        [148,  27,  16],
        [177,  70,  50],
```

```
            [219, 119,  95]],

           [[134,  78,  41],
            [132,  78,  40],
            [133,  79,  41],
            ...,
            [163,  51,  37],
            [207, 108,  85],
            [247, 157, 130]],

           ...,

           [[ 16,   5,   3],
            [ 16,   5,   3],
            [ 16,   6,   4],
            ...,
            [  5,   1,   0],
            [  5,   1,   0],
            [  6,   2,   1]],

           [[ 14,   3,   1],
            [ 14,   3,   1],
            [ 15,   4,   2],
            ...,
            [  6,   2,   1],
            [  6,   2,   1],
            [  6,   2,   1]],

           [[ 11,   0,   0],
            [ 11,   0,   0],
            [ 12,   1,   0],
            ...,
            [  6,   2,   1],
            [  7,   3,   2],
            [  7,   3,   2]]], dtype=uint8)>
```

```
img.shape
```

```
     (512, 512, 3)
```

```
# Normalized image data
img/255.
```

```
     array([[[0.64705882, 0.45882353, 0.27843137],
             [0.6745098 , 0.48627451, 0.30588235],
             [0.69019608, 0.50588235, 0.33333333],
             ...,
             [0.54117647, 0.05490196, 0.01568627],
             [0.60392157, 0.16078431, 0.09019608],
             [0.74117647, 0.32941176, 0.24313725]],

            [[0.62352941, 0.42352941, 0.25490196],
             [0.63137255, 0.43137255, 0.2627451 ],
             [0.62352941, 0.43137255, 0.2627451 ],
```

```
        ...,
        [0.58039216, 0.10588235, 0.0627451 ],
        [0.69411765, 0.2745098 , 0.19607843],
        [0.85882353, 0.46666667, 0.37254902]],

       [[0.5254902 , 0.30588235, 0.16078431],
        [0.51764706, 0.30588235, 0.15686275],
        [0.52156863, 0.30980392, 0.16078431],
        ...,
        [0.63921569, 0.2       , 0.14509804],
        [0.81176471, 0.42352941, 0.33333333],
        [0.96862745, 0.61568627, 0.50980392]],

       ...,

       [[0.0627451 , 0.01960784, 0.01176471],
        [0.0627451 , 0.01960784, 0.01176471],
        [0.0627451 , 0.02352941, 0.01568627],
        ...,
        [0.01960784, 0.00392157, 0.        ],
        [0.01960784, 0.00392157, 0.        ],
        [0.02352941, 0.00784314, 0.00392157]],

       [[0.05490196, 0.01176471, 0.00392157],
        [0.05490196, 0.01176471, 0.00392157],
        [0.05882353, 0.01568627, 0.00784314],
        ...,
        [0.02352941, 0.00784314, 0.00392157],
        [0.02352941, 0.00784314, 0.00392157],
        [0.02352941, 0.00784314, 0.00392157]],

       [[0.04313725, 0.        , 0.        ],
        [0.04313725, 0.        , 0.        ],
        [0.04705882, 0.00392157, 0.        ],
        ...,
        [0.02352941, 0.00784314, 0.00392157],
        [0.02745098, 0.01176471, 0.00784314],
        [0.02745098, 0.01176471, 0.00784314]]])
```

Create train and test data sets

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


tf.random.set_seed(42)

# Preprocess data
train_datagen = ImageDataGenerator(rescale=1./255)
valid_datagen = ImageDataGenerator(rescale=1./255)

# Setup directories
train_dir = '/content/pizza_steak/train'
```

```
test_dir = 'pizza_steak/test'

train_data = train_datagen.flow_from_directory(directory=train_dir,
                                               batch_size=32,
                                               target_size=(224,224), # change image size/sha
                                               class_mode='binary',
                                               seed=42)
valid_data = valid_datagen.flow_from_directory(directory=test_dir,
                                               batch_size=32,
                                               target_size=(224,224),
                                               class_mode='binary',
                                               seed=42)
```

```
Found 1500 images belonging to 2 classes.
Found 500 images belonging to 2 classes.
```

Model 1 is a simple example of a neural network. This model performs no better than random chance.

```
tf.random.set_seed(42)

model_1 = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(224,224,3)),
    tf.keras.layers.Dense(4, activation='relu'),
    tf.keras.layers.Dense(4, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

#Compile model
model_1.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

# Fit model
history_1 = model_1.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=valid_data,
                        validation_steps=len(valid_data))
```

```
Epoch 1/5
47/47 [==============================] - 13s 186ms/step - loss: 0.8251 - accuracy: 0.494
Epoch 2/5
47/47 [==============================] - 9s 184ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 3/5
47/47 [==============================] - 9s 184ms/step - loss: 0.6932 - accuracy: 0.5000
Epoch 4/5
47/47 [==============================] - 9s 195ms/step - loss: 0.6932 - accuracy: 0.4827
```

```
Epoch 5/5
47/47 [==============================] - 9s 183ms/step - loss: 0.6932 - accuracy: 0.5000
```

```
model_1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 150528)            0

 dense (Dense)               (None, 4)                 602116

 dense_1 (Dense)             (None, 4)                 20

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 602,141
Trainable params: 602,141
Non-trainable params: 0
_________________________________________________________________
```

Model 2 increases the number of trainable parameters by increasing the dimensionality of the output space. This does significantly increase the accuracy of the model.

```
tf.random.set_seed(42)

model_2 = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(224,224,3)),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model_2.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

history_2 = model_2.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=valid_data,
                        validation_steps=len(valid_data))
```

```
Epoch 1/5
47/47 [==============================] - 10s 209ms/step - loss: 2.9554 - accuracy: 0.643
Epoch 2/5
```

```
47/47 [==============================] - 8s 181ms/step - loss: 0.8232 - accuracy: 0.7307
Epoch 3/5
47/47 [==============================] - 8s 180ms/step - loss: 0.9018 - accuracy: 0.7140
Epoch 4/5
47/47 [==============================] - 9s 183ms/step - loss: 0.5541 - accuracy: 0.7687
Epoch 5/5
47/47 [==============================] - 9s 182ms/step - loss: 0.4646 - accuracy: 0.7960
```

```
model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 150528)            0

 dense_3 (Dense)             (None, 100)               15052900

 dense_4 (Dense)             (None, 100)               10100

 dense_5 (Dense)             (None, 100)               10100

 dense_6 (Dense)             (None, 1)                 101

=================================================================
Total params: 15,073,201
Trainable params: 15,073,201
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Activation
from tensorflow.keras import Sequential
from pyparsing.core import StringEnd
```

Model 3 is a small 3 layer CNN

```
model_3 = Sequential([
      Conv2D(filters=10,
             kernel_size=3,
             strides=1,
             padding='valid',
             activation='relu',
             input_shape=(224,224,3)), # input layer
      Conv2D(10,3,activation='relu'),
      Conv2D(10,3,activation='relu'),
      Flatten(),
      Dense(1,activation='sigmoid') # output layer
```

```
])
```

```
model_3.compile(loss='binary_crossentropy',
                optimizer=Adam(),
                metrics=['accuracy'])
```

```
model_3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 222, 222, 10)      280

 conv2d_1 (Conv2D)            (None, 220, 220, 10)      910

 conv2d_2 (Conv2D)            (None, 218, 218, 10)      910

 flatten_2 (Flatten)          (None, 475240)            0

 dense_7 (Dense)              (None, 1)                 475241

=================================================================
Total params: 477,341
Trainable params: 477,341
Non-trainable params: 0
_________________________________________________________________
```

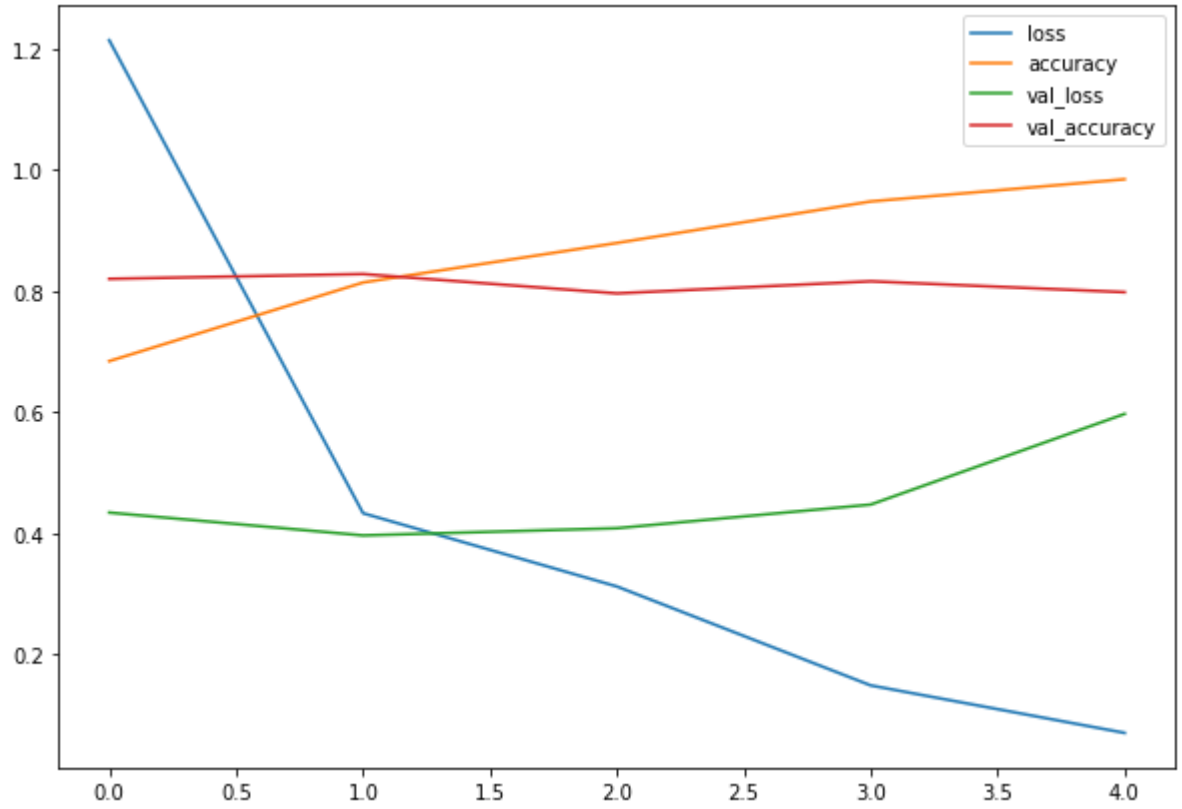
Model 3 improves on Model 2, even though it greatly reduces the number of trainable parameters.

```
history_3 = model_3.fit(train_data,
            epochs=5,
            steps_per_epoch=len(train_data),
            validation_data=valid_data,
            validation_steps=len(valid_data))
```

```
Epoch 1/5
47/47 [==============================] - 19s 213ms/step - loss: 1.2145 - accuracy: 0.684
Epoch 2/5
47/47 [==============================] - 9s 198ms/step - loss: 0.4326 - accuracy: 0.8140
Epoch 3/5
47/47 [==============================] - 10s 202ms/step - loss: 0.3114 - accuracy: 0.879
Epoch 4/5
47/47 [==============================] - 9s 198ms/step - loss: 0.1479 - accuracy: 0.9480
Epoch 5/5
47/47 [==============================] - 9s 198ms/step - loss: 0.0694 - accuracy: 0.9847
```

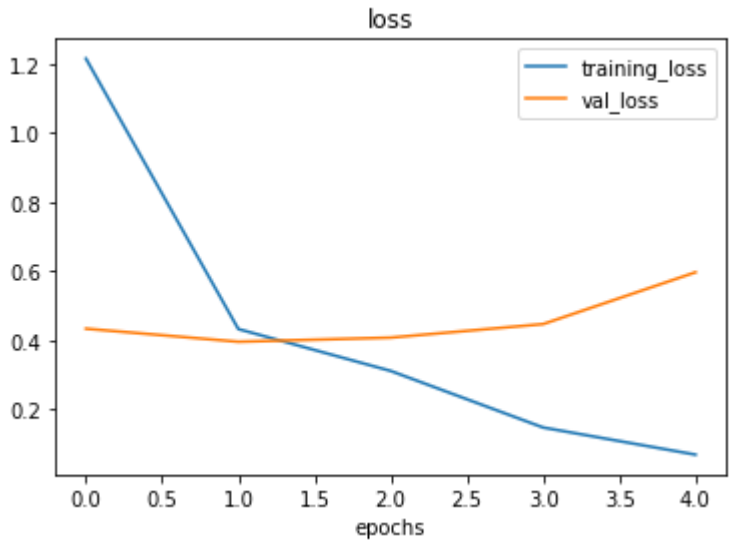
```
import pandas as pd
pd.DataFrame(history_3.history).plot(figsize=(10,7)) #plot loss and accuracy of model
```

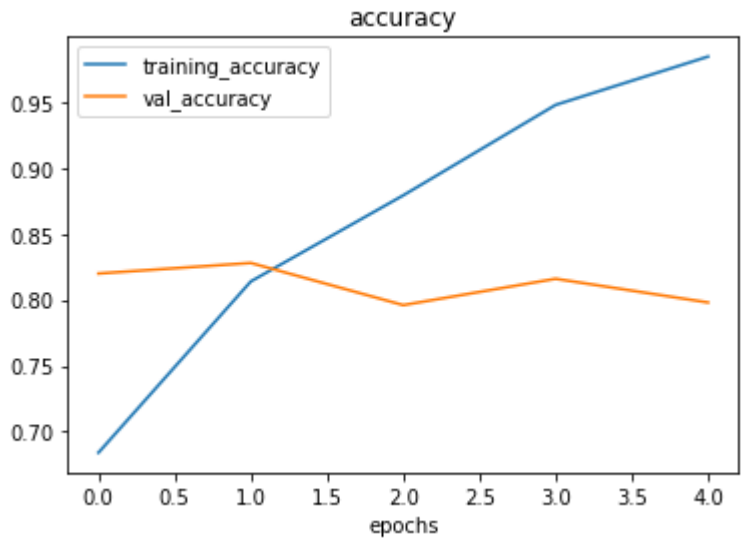
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7ec84fbe50>
```




```
def plot_loss_curves(history):
  """
  Returns separate loss curves for training and validation metrics.
  """
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  accuracy = history.history['accuracy']
  val_accuracy = history.history['val_accuracy']

  epochs = range(len(history.history['loss']))

  #plot loss
  plt.plot(epochs, loss, label='training_loss')
  plt.plot(epochs,val_loss,label='val_loss')
  plt.title('loss')
  plt.xlabel('epochs')
  plt.legend()

  #plot accuracy
  plt.figure()
  plt.plot(epochs, accuracy, label='training_accuracy')
  plt.plot(epochs,val_accuracy,label='val_accuracy')
  plt.title('accuracy')
```

```
  plt.xlabel('epochs')
  plt.legend();


plot_loss_curves(history_3)

#some overfitting appearing in this model
```





Model 4 adds in 2 MaxPool layers to reduce overfitting

```
model_4 = Sequential([
      Conv2D(10,3,activation='relu',input_shape=(224,224,3)),
      MaxPool2D(pool_size=2),
      Conv2D(10,3,activation='relu'),
      MaxPool2D(),
      Conv2D(10,3,activation='relu'),
      MaxPool2D(),
      Flatten(),
      Dense(1,activation='sigmoid')
])
```

```
model_4.compile(loss='binary_crossentropy',
                optimizer=Adam(),
                metrics=['accuracy'])
```

```
history_4 = model_4.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=valid_data,
                        validation_steps=len(valid_data)
)
```

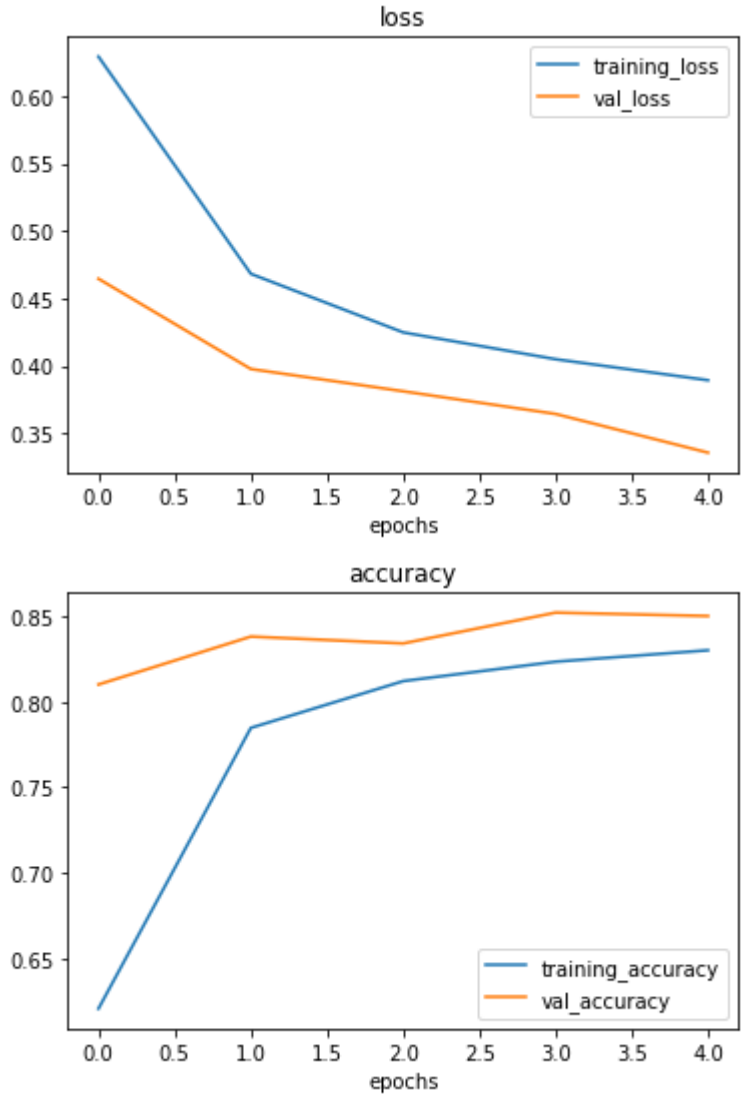
```
Epoch 1/5
47/47 [==============================] - 10s 193ms/step - loss: 0.6293 - accuracy: 0.626
Epoch 2/5
47/47 [==============================] - 9s 188ms/step - loss: 0.4682 - accuracy: 0.7847
Epoch 3/5
47/47 [==============================] - 9s 189ms/step - loss: 0.4247 - accuracy: 0.8120
Epoch 4/5
47/47 [==============================] - 9s 189ms/step - loss: 0.4048 - accuracy: 0.8233
Epoch 5/5
47/47 [==============================] - 9s 189ms/step - loss: 0.3892 - accuracy: 0.8300
```

```
model_4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 222, 222, 10)      280

 max_pooling2d (MaxPooling2D  (None, 111, 111, 10)     0
 )

 conv2d_4 (Conv2D)           (None, 109, 109, 10)      910

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 10)       0
 2D)

 conv2d_5 (Conv2D)           (None, 52, 52, 10)        910

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 10)       0
 2D)

 flatten_3 (Flatten)         (None, 6760)              0

 dense_8 (Dense)             (None, 1)                 6761
```

```
=================================================================
Total params: 8,861
Trainable params: 8,861
Non-trainable params: 0
_________________________________________________________________
```

```
plot_loss_curves(history_4)

#this model slightly improved accuracy while reducing overfitting.
```



Data Augmentation: we will attempt to reduce overfitting even more by augmenting the training data

```
#Augment data with rotation, shear, zoom, shift, and flip
train_datagen_augmented = ImageDataGenerator(rescale=1/255.,
                                             rotation_range=0.2,
                                             shear_range=0.2,
                                             zoom_range=0.2,
                                             width_shift_range=0.2,
```

```
                                                height_shift_range=0.2,
                                                horizontal_flip=True)

train_datagen = ImageDataGenerator(rescale=1/255.)

test_datagen = ImageDataGenerator(rescale=1/255.)


print('Augmented training data')
train_data_augmented = train_datagen_augmented.flow_from_directory(train_dir,
                                                   target_size=(224,224),
                                                   batch_size=32,
                                                   class_mode='binary',
                                                   shuffle=False) #False for image example below,
print('Non-augmented training data')
train_data = train_datagen.flow_from_directory(train_dir,
                                                   target_size=(224,224),
                                                   batch_size=32,
                                                   class_mode='binary',
                                                   shuffle=False) #False for image example below,

print('Non-augmented test data')
test_data = test_datagen.flow_from_directory(test_dir,
                                                target_size=(224,224),
                                                batch_size=32,
                                                class_mode='binary')


     Augmented training data
     Found 1500 images belonging to 2 classes.
     Non-augmented training data
     Found 1500 images belonging to 2 classes.
     Non-augmented test data
     Found 500 images belonging to 2 classes.


images, labels = train_data.next()
augmented_images, augmented_labels = train_data_augmented.next()


#Randomly choose an image and show the original and augmented version
random_number = random.randint(0,31)
print(f'Showing image number: {random_number}')
plt.imshow(images[random_number])
plt.title(f'Original image')
plt.axis(False)
plt.figure()
plt.imshow(augmented_images[random_number])
plt.title(f'Augmented images')
plt.axis(False);
```

```
Showing image number: 14
```

Original image

Augmented images

```
#Reshuffling training data
train_data_augmented = train_datagen_augmented.flow_from_directory(train_dir,
                                                   target_size=(224,224),
                                                   batch_size=32,
                                                   class_mode='binary',
                                                   shuffle=True)
train_data = train_datagen.flow_from_directory(train_dir,
                                                   target_size=(224,224),
                                                   batch_size=32,
                                                   class_mode='binary',
                                                   shuffle=True)
```

```
Found 1500 images belonging to 2 classes.
Found 1500 images belonging to 2 classes.
```

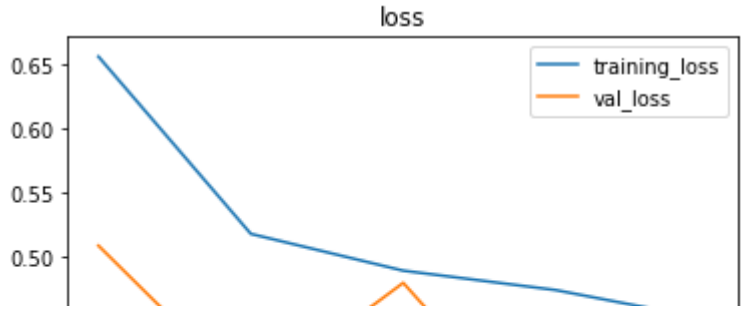
Model 5 is identical to Model 4 but uses augmented data

```
model_5 = Sequential([
          Conv2D(10,3,activation='relu'),
          MaxPool2D(pool_size=2),
          Conv2D(10,3,activation='relu'),
          MaxPool2D(pool_size=2),
```

```
          Conv2D(10,3,activation='relu'),
          MaxPool2D(pool_size=2),
          Flatten(),
          Dense(1,activation='sigmoid')
])

model_5.compile(loss='binary_crossentropy',
                optimizer=Adam(),
                metrics=['accuracy'])

history_5 = model_5.fit(train_data_augmented,
                        epochs=5,
                        steps_per_epoch=len(train_data_augmented),
                        validation_data=test_data,
                        validation_steps=len(test_data))
```

```
Epoch 1/5
47/47 [==============================] - 22s 466ms/step - loss: 0.6557 - accuracy: 0.620
Epoch 2/5
47/47 [==============================] - 22s 461ms/step - loss: 0.5181 - accuracy: 0.742
Epoch 3/5
47/47 [==============================] - 22s 459ms/step - loss: 0.4895 - accuracy: 0.774
Epoch 4/5
47/47 [==============================] - 22s 459ms/step - loss: 0.4744 - accuracy: 0.778
Epoch 5/5
47/47 [==============================] - 22s 460ms/step - loss: 0.4516 - accuracy: 0.796
```

```
plot_loss_curves(history_5)
```



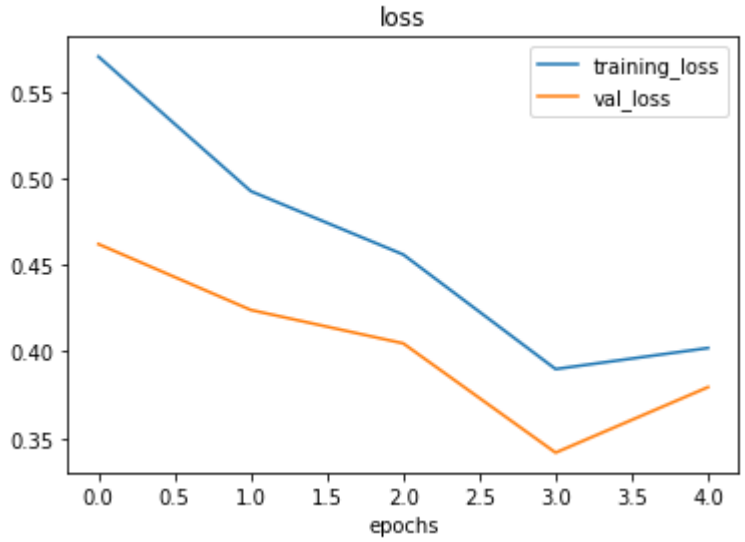
Model 6 no longer augments data and adds additional Conv2D layers. This structure matches the Tiny VGG: https://poloclub.github.io/cnn-explainer/

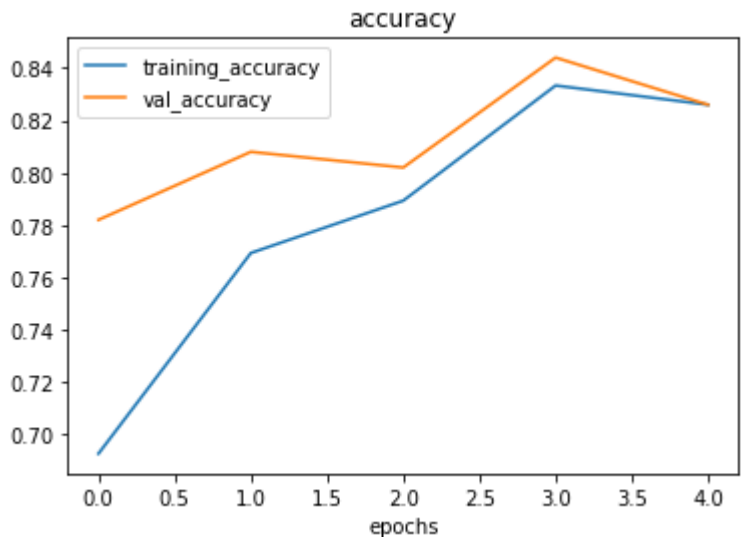
0.35

```
model_6 = Sequential([
  Conv2D(10, 3, activation='relu', input_shape=(224, 224, 3)), # same input shape as our imag
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Conv2D(10, 3, activation='relu'),
  Conv2D(10, 3, activation='relu'),
  MaxPool2D(),
  Flatten(),
  Dense(1, activation='sigmoid')
])

model_6.compile(loss="binary_crossentropy",
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])

history_6 = model_6.fit(train_data,
                        epochs=5,
                        steps_per_epoch=len(train_data),
                        validation_data=test_data,
                        validation_steps=len(test_data))
```

```
Epoch 1/5
47/47 [==============================] - 10s 199ms/step - loss: 0.5703 - accuracy: 0.692
Epoch 2/5
47/47 [==============================] - 9s 194ms/step - loss: 0.4926 - accuracy: 0.7693
Epoch 3/5
47/47 [==============================] - 9s 194ms/step - loss: 0.4560 - accuracy: 0.7893
Epoch 4/5
47/47 [==============================] - 9s 196ms/step - loss: 0.3899 - accuracy: 0.8333
Epoch 5/5
47/47 [==============================] - 9s 198ms/step - loss: 0.4020 - accuracy: 0.8266
```

```
plot_loss_curves(history_6)

# Overfitting is greatly reduced
```






```
model_6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 222, 222, 10)      280

 conv2d_10 (Conv2D)          (None, 220, 220, 10)      910

 max_pooling2d_6 (MaxPooling  (None, 110, 110, 10)     0
 2D)

 conv2d_11 (Conv2D)          (None, 108, 108, 10)      910

 conv2d_12 (Conv2D)          (None, 106, 106, 10)      910

 max_pooling2d_7 (MaxPooling  (None, 53, 53, 10)       0
 2D)

 flatten_5 (Flatten)         (None, 28090)             0

 dense_10 (Dense)            (None, 1)                 28091
```

```
================================================================
Total params: 31,101
Trainable params: 31,101
Non-trainable params: 0
_________________________________________________________________
```