Building ConvNets to create a mood classifier and identify sign language digits (Convolutional Neural Networks: Application)

- Build a ConvNet to identify sign language digits using the TF Keras Functional API
- Build and train a ConvNet in TensorFlow for a **multiclass** classification problem

In [8]:
```
import math
import numpy as np
import h5py
import matplotlib.pyplot as plt
from matplotlib.pyplot import imread
import scipy
from PIL import Image
import pandas as pd
import tensorflow as tf
import tensorflow.keras.layers as tfl
from tensorflow.python.framework import ops
from cnn_utils import *
from test_utils_digits import summary, comparator

%matplotlib inline
np.random.seed(1)
```

The Functional API

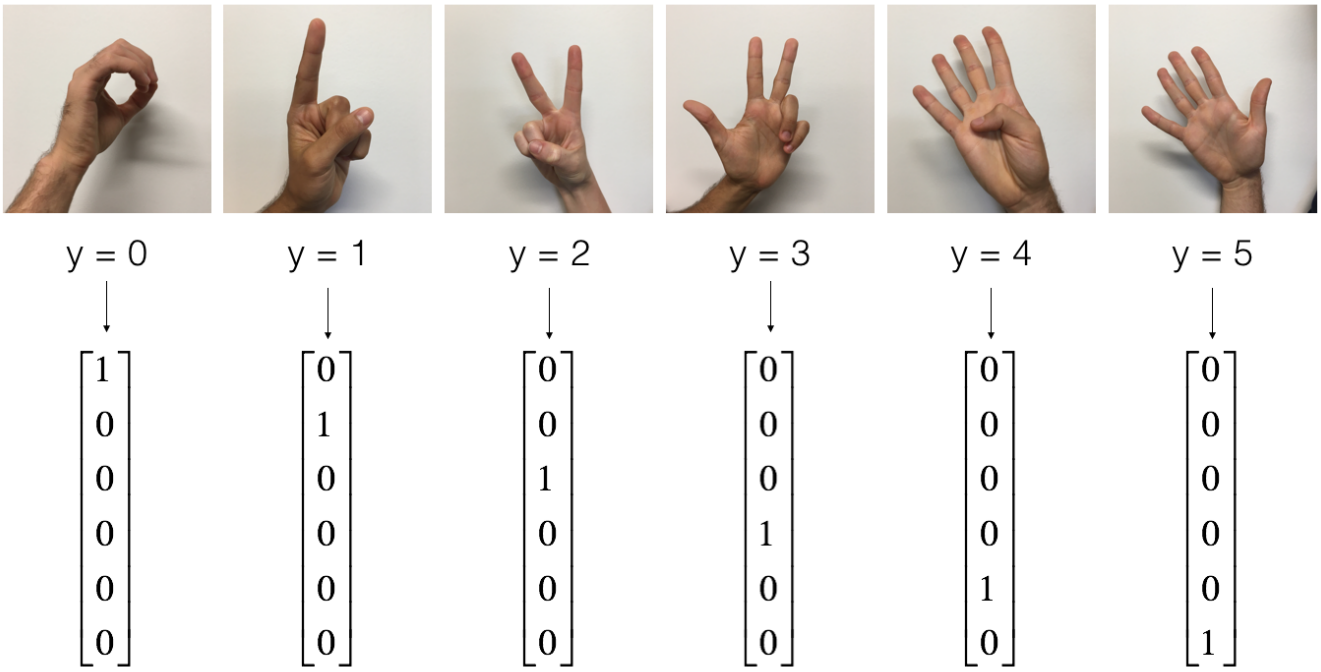
using Keras' flexible Functional APIto build a ConvNet that can differentiate between 6 sign language digits. (creating a graph of layers)

step 1: Load the SIGNS Dataset

the SIGNS dataset is a collection of 6 signs representing numbers from 0 to 5.

In [9]:
```
#Loading the data (signs)
X_train_orig, Y_train_orig, X_test_orig, Y_test_orig, classes = load_signs_dataset()
```

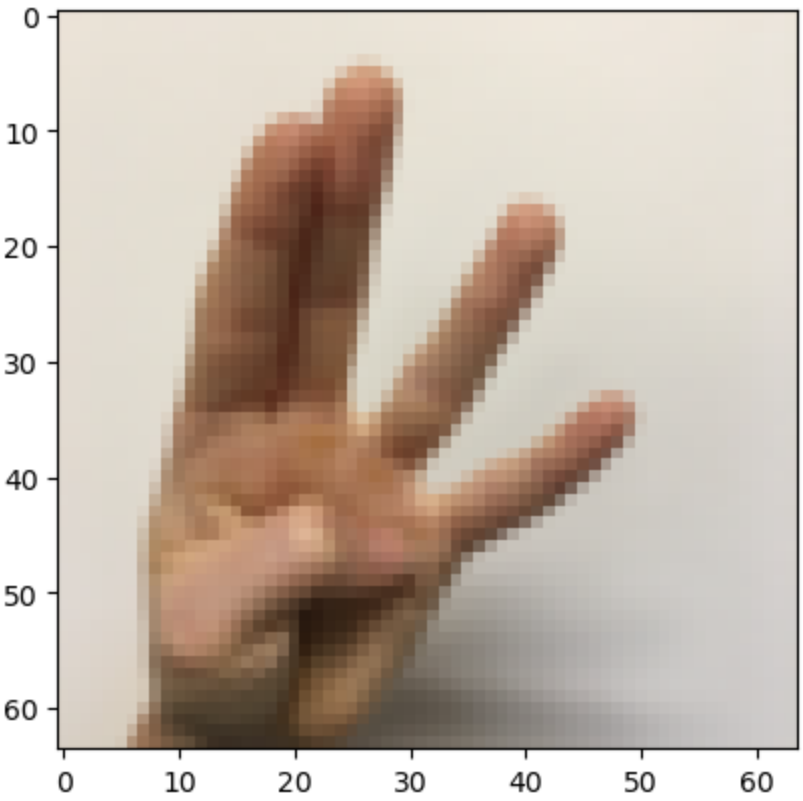
This is our signs:



The following cell will show you an example of a labelled image in the dataset:

In [11]:
```
# Example of an image from the dataset
index = 9
plt.imshow(X_train_orig[index])
print ("y = " + str(np.squeeze(Y_train_orig[:, index])))
```

```
y = 4
```



step2: Split the Data into Train/Test Sets

In [12]:
```
X_train = X_train_orig/255.
```

```
Y_train = convert_to_one_hot(Y_train_orig, 6).T
Y_test = convert_to_one_hot(Y_test_orig, 6).T
print ("number of training examples = " + str(X_train.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
print ("Y_train shape: " + str(Y_train.shape))
print ("X_test shape: " + str(X_test.shape))
print ("Y_test shape: " + str(Y_test.shape))
```

```
number of training examples = 1080
number of test examples = 120
X_train shape: (1080, 64, 64, 3)
Y_train shape: (1080, 6)
X_test shape: (120, 64, 64, 3)
Y_test shape: (120, 6)
```

step3: convolutional_model

Implementing the `convolutional_model` function below to build the following model: `CONV2D -> RELU -> MAXPOOL -> CONV2D -> RELU -> MAXPOOL -> FLATTEN -> DENSE`.

- the Functional APIs return a TF Keras model object.

In [16]:
```
def convolutional_model(input_shape):
    """
    Implements the forward propagation for the model:
    CONV2D -> RELU -> MAXPOOL -> CONV2D -> RELU -> MAXPOOL -> FLATTEN -> DENSE

    Note that for simplicity and grading purposes, you'll hard-code some values
    such as the stride and kernel (filter) sizes.
    Normally, functions should take these values as function parameters.

    Arguments:
    input_img -- input dataset, of shape (input_shape)

    Returns:
    model -- TF Keras model (object containing the information for the entire training p
    """

    input_img = tf.keras.Input(shape=input_shape)

    Z1 = tfl.Conv2D(8, 4, activation='linear', padding='same', strides=1)(input_img)
    A1 = tfl.ReLU()(Z1)
    P1 = tfl.MaxPool2D(pool_size=(8, 8), strides=(8, 8), padding='same')(A1)
    Z2 = tfl.Conv2D(16, 2, activation='linear', padding='same', strides=1)(P1)
    A2 = tfl.ReLU()(Z2)
    P2 = tfl.MaxPool2D(pool_size=(4, 4), strides=(4, 4), padding='same')(A2)
    F = tfl.Flatten()(P2)
    outputs = tfl.Dense(6, activation='softmax')(F)

    model = tf.keras.Model(inputs=input_img, outputs=outputs)
    return model
```

In [17]:
```
conv_model = convolutional_model((64, 64, 3))
conv_model.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
conv_model.summary()

output = [['InputLayer', [(None, 64, 64, 3)], 0],
        ['Conv2D', (None, 64, 64, 8), 392, 'same', 'linear', 'GlorotUniform'],
                                                  8), 0],
```

```
        ['MaxPooling2D', (None, 8, 8, 8), 0, (8, 8), (8, 8), 'same'],
        ['Conv2D', (None, 8, 8, 16), 528, 'same', 'linear', 'GlorotUniform'],
        ['ReLU', (None, 8, 8, 16), 0],
        ['MaxPooling2D', (None, 2, 2, 16), 0, (4, 4), (4, 4), 'same'],
        ['Flatten', (None, 64), 0],
        ['Dense', (None, 6), 390, 'softmax']]

comparator(summary(conv_model), output)
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 64, 64, 3)]       0

 conv2d_4 (Conv2D)           (None, 64, 64, 8)         392

 re_lu_4 (ReLU)              (None, 64, 64, 8)         0

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 8)           0
 g2D)

 conv2d_5 (Conv2D)           (None, 8, 8, 16)          528

 re_lu_5 (ReLU)              (None, 8, 8, 16)          0

 max_pooling2d_5 (MaxPoolin  (None, 2, 2, 16)          0
 g2D)

 flatten_2 (Flatten)         (None, 64)                0

 dense_2 (Dense)             (None, 6)                 390

=================================================================
Total params: 1310 (5.12 KB)
Trainable params: 1310 (5.12 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
All tests passed!
```

step4: Train the Model

In [19]:

```
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, Y_train)).batch(64)
test_dataset = tf.data.Dataset.from_tensor_slices((X_test, Y_test)).batch(64)
history = conv_model.fit(train_dataset, epochs=100, validation_data=test_dataset)
```

```
Epoch 1/100
17/17 [==============================] - 0s 10ms/step - loss: 0.4550 - accuracy: 0.8528
- val_loss: 0.5114 - val_accuracy: 0.8250
Epoch 2/100
17/17 [==============================] - 0s 9ms/step - loss: 0.4514 - accuracy: 0.8519 -
val_loss: 0.5085 - val_accuracy: 0.8250
Epoch 3/100
17/17 [==============================] - 0s 10ms/step - loss: 0.4477 - accuracy: 0.8537
- val_loss: 0.5054 - val_accuracy: 0.8250
Epoch 4/100
17/17 [==============================] - 0s 10ms/step - loss: 0.4438 - accuracy: 0.8556
- val_loss: 0.5029 - val_accuracy: 0.8250
Epoch 5/100
17/17 [==============================] - 0s 10ms/step - loss: 0.4401 - accuracy: 0.8556
- val_loss: 0.4997 - val_accuracy: 0.8250
Epoch 6/100
17/17 [==============================] - 0s 11ms/step - loss: 0.4365 - accuracy: 0.8593
- val_loss: 0.4975 - val_accuracy: 0.8250
Epoch 7/100
17/17 [==============================] - 0s 9ms/step - loss: 0.4329 - accuracy: 0.8602 -
val_loss: 0.4945 - val_accuracy: 0.8250
Epoch 8/100
17/17 [==============================] - 0s 9ms/step - loss: 0.4296 - accuracy: 0.8630 -
val_loss: 0.4920 - val_accuracy: 0.8250
Epoch 9/100
17/17 [==============================] - 0s 8ms/step - loss: 0.4260 - accuracy: 0.8685 -
val_loss: 0.4896 - val_accuracy: 0.8250
Epoch 10/100
17/17 [==============================] - 0s 8ms/step - loss: 0.4225 - accuracy: 0.8694 -
val_loss: 0.4874 - val_accuracy: 0.8250
Epoch 11/100
17/17 [==============================] - 0s 7ms/step - loss: 0.4190 - accuracy: 0.8704 -
val_loss: 0.4853 - val_accuracy: 0.8250
Epoch 12/100
17/17 [==============================] - 0s 8ms/step - loss: 0.4158 - accuracy: 0.8694 -
val_loss: 0.4831 - val_accuracy: 0.8250
Epoch 13/100
17/17 [==============================] - 0s 8ms/step - loss: 0.4126 - accuracy: 0.8722 -
val_loss: 0.4808 - val_accuracy: 0.8250
Epoch 14/100
17/17 [==============================] - 0s 7ms/step - loss: 0.4094 - accuracy: 0.8731 -
val_loss: 0.4784 - val_accuracy: 0.8417
Epoch 15/100
17/17 [==============================] - 0s 7ms/step - loss: 0.4063 - accuracy: 0.8750 -
val_loss: 0.4760 - val_accuracy: 0.8500
Epoch 16/100
17/17 [==============================] - 0s 9ms/step - loss: 0.4031 - accuracy: 0.8769 -
val_loss: 0.4737 - val_accuracy: 0.8500
Epoch 17/100
17/17 [==============================] - 0s 9ms/step - loss: 0.4001 - accuracy: 0.8778 -
val_loss: 0.4716 - val_accuracy: 0.8500
Epoch 18/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3968 - accuracy: 0.8769 -
val_loss: 0.4683 - val_accuracy: 0.8500
Epoch 19/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3938 - accuracy: 0.8778 -
val_loss: 0.4673 - val_accuracy: 0.8500
Epoch 20/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3909 - accuracy: 0.8778 -
val_loss: 0.4650 - val_accuracy: 0.8500
Epoch 21/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3879 - accuracy: 0.8815 -
val_loss: 0.4633 - val_accuracy: 0.8500
Epoch 22/100
```

```
17/17 [==============================] - 0s 8ms/step - loss: 0.3849 - accuracy: 0.8824 -
val_loss: 0.4610 - val_accuracy: 0.8500
Epoch 23/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3821 - accuracy: 0.8824 -
val_loss: 0.4595 - val_accuracy: 0.8500
Epoch 24/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3794 - accuracy: 0.8815 -
val_loss: 0.4574 - val_accuracy: 0.8500
Epoch 25/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3764 - accuracy: 0.8833 -
val_loss: 0.4551 - val_accuracy: 0.8500
Epoch 26/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3738 - accuracy: 0.8852 -
val_loss: 0.4540 - val_accuracy: 0.8500
Epoch 27/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3711 - accuracy: 0.8852 -
val_loss: 0.4516 - val_accuracy: 0.8500
Epoch 28/100
17/17 [==============================] - 0s 7ms/step - loss: 0.3683 - accuracy: 0.8870 -
val_loss: 0.4504 - val_accuracy: 0.8500
Epoch 29/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3658 - accuracy: 0.8889 -
val_loss: 0.4488 - val_accuracy: 0.8583
Epoch 30/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3631 - accuracy: 0.8880 -
val_loss: 0.4469 - val_accuracy: 0.8583
Epoch 31/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3606 - accuracy: 0.8907 -
val_loss: 0.4457 - val_accuracy: 0.8583
Epoch 32/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3580 - accuracy: 0.8917 -
val_loss: 0.4435 - val_accuracy: 0.8583
Epoch 33/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3554 - accuracy: 0.8926 -
val_loss: 0.4423 - val_accuracy: 0.8583
Epoch 34/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3529 - accuracy: 0.8954 -
val_loss: 0.4401 - val_accuracy: 0.8583
Epoch 35/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3507 - accuracy: 0.8935 -
val_loss: 0.4393 - val_accuracy: 0.8750
Epoch 36/100
17/17 [==============================] - 0s 7ms/step - loss: 0.3481 - accuracy: 0.8963 -
val_loss: 0.4369 - val_accuracy: 0.8750
Epoch 37/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3457 - accuracy: 0.8972 -
val_loss: 0.4357 - val_accuracy: 0.8750
Epoch 38/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3435 - accuracy: 0.8972 -
val_loss: 0.4336 - val_accuracy: 0.8750
Epoch 39/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3412 - accuracy: 0.8972 -
val_loss: 0.4325 - val_accuracy: 0.8750
Epoch 40/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3389 - accuracy: 0.8972 -
val_loss: 0.4304 - val_accuracy: 0.8750
Epoch 41/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3367 - accuracy: 0.8963 -
val_loss: 0.4306 - val_accuracy: 0.8750
Epoch 42/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3347 - accuracy: 0.8991 -
val_loss: 0.4302 - val_accuracy: 0.8750
Epoch 43/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3325 - accuracy: 0.8981 -
```

```
val_loss: 0.4287 - val_accuracy: 0.8750
Epoch 44/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3300 - accuracy: 0.9000 -
val_loss: 0.4270 - val_accuracy: 0.8750
Epoch 45/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3280 - accuracy: 0.9000 -
val_loss: 0.4261 - val_accuracy: 0.8750
Epoch 46/100
17/17 [==============================] - 0s 7ms/step - loss: 0.3257 - accuracy: 0.9009 -
val_loss: 0.4248 - val_accuracy: 0.8750
Epoch 47/100
17/17 [==============================] - 0s 7ms/step - loss: 0.3236 - accuracy: 0.9019 -
val_loss: 0.4233 - val_accuracy: 0.8750
Epoch 48/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3216 - accuracy: 0.9028 -
val_loss: 0.4221 - val_accuracy: 0.8833
Epoch 49/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3195 - accuracy: 0.9028 -
val_loss: 0.4209 - val_accuracy: 0.8833
Epoch 50/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3174 - accuracy: 0.9065 -
val_loss: 0.4197 - val_accuracy: 0.8833
Epoch 51/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3154 - accuracy: 0.9056 -
val_loss: 0.4181 - val_accuracy: 0.8833
Epoch 52/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3133 - accuracy: 0.9074 -
val_loss: 0.4176 - val_accuracy: 0.8833
Epoch 53/100
17/17 [==============================] - 0s 7ms/step - loss: 0.3115 - accuracy: 0.9065 -
val_loss: 0.4166 - val_accuracy: 0.8833
Epoch 54/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3097 - accuracy: 0.9102 -
val_loss: 0.4151 - val_accuracy: 0.8833
Epoch 55/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3077 - accuracy: 0.9102 -
val_loss: 0.4144 - val_accuracy: 0.8833
Epoch 56/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3059 - accuracy: 0.9093 -
val_loss: 0.4134 - val_accuracy: 0.8833
Epoch 57/100
17/17 [==============================] - 0s 9ms/step - loss: 0.3040 - accuracy: 0.9111 -
val_loss: 0.4124 - val_accuracy: 0.8833
Epoch 58/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3023 - accuracy: 0.9111 -
val_loss: 0.4120 - val_accuracy: 0.8833
Epoch 59/100
17/17 [==============================] - 0s 8ms/step - loss: 0.3004 - accuracy: 0.9130 -
val_loss: 0.4098 - val_accuracy: 0.8833
Epoch 60/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2986 - accuracy: 0.9139 -
val_loss: 0.4089 - val_accuracy: 0.8833
Epoch 61/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2966 - accuracy: 0.9148 -
val_loss: 0.4080 - val_accuracy: 0.8833
Epoch 62/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2949 - accuracy: 0.9167 -
val_loss: 0.4076 - val_accuracy: 0.8833
Epoch 63/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2929 - accuracy: 0.9167 -
val_loss: 0.4057 - val_accuracy: 0.8833
Epoch 64/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2913 - accuracy: 0.9167 -
val_loss: 0.4054 - val_accuracy: 0.8833
```

```
Epoch 65/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2894 - accuracy: 0.9167 -
val_loss: 0.4045 - val_accuracy: 0.8833
Epoch 66/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2877 - accuracy: 0.9167 -
val_loss: 0.4035 - val_accuracy: 0.8833
Epoch 67/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2859 - accuracy: 0.9176 -
val_loss: 0.4026 - val_accuracy: 0.8833
Epoch 68/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2840 - accuracy: 0.9176 -
val_loss: 0.4016 - val_accuracy: 0.8833
Epoch 69/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2823 - accuracy: 0.9176 -
val_loss: 0.4008 - val_accuracy: 0.8833
Epoch 70/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2809 - accuracy: 0.9176 -
val_loss: 0.4002 - val_accuracy: 0.8833
Epoch 71/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2790 - accuracy: 0.9185 -
val_loss: 0.3987 - val_accuracy: 0.8833
Epoch 72/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2774 - accuracy: 0.9194 -
val_loss: 0.3987 - val_accuracy: 0.8833
Epoch 73/100
17/17 [==============================] - 0s 7ms/step - loss: 0.2758 - accuracy: 0.9194 -
val_loss: 0.3975 - val_accuracy: 0.8833
Epoch 74/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2742 - accuracy: 0.9204 -
val_loss: 0.3966 - val_accuracy: 0.8833
Epoch 75/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2727 - accuracy: 0.9213 -
val_loss: 0.3959 - val_accuracy: 0.8833
Epoch 76/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2712 - accuracy: 0.9204 -
val_loss: 0.3948 - val_accuracy: 0.8833
Epoch 77/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2695 - accuracy: 0.9204 -
val_loss: 0.3945 - val_accuracy: 0.8833
Epoch 78/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2679 - accuracy: 0.9213 -
val_loss: 0.3936 - val_accuracy: 0.8833
Epoch 79/100
17/17 [==============================] - 0s 10ms/step - loss: 0.2666 - accuracy: 0.9222
- val_loss: 0.3931 - val_accuracy: 0.8833
Epoch 80/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2649 - accuracy: 0.9222 -
val_loss: 0.3920 - val_accuracy: 0.8833
Epoch 81/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2636 - accuracy: 0.9222 -
val_loss: 0.3914 - val_accuracy: 0.8833
Epoch 82/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2620 - accuracy: 0.9222 -
val_loss: 0.3908 - val_accuracy: 0.8833
Epoch 83/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2606 - accuracy: 0.9231 -
val_loss: 0.3902 - val_accuracy: 0.8833
Epoch 84/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2591 - accuracy: 0.9241 -
val_loss: 0.3891 - val_accuracy: 0.8833
Epoch 85/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2577 - accuracy: 0.9241 -
val_loss: 0.3892 - val_accuracy: 0.8833
Epoch 86/100
```

```
17/17 [==============================] - 0s 9ms/step - loss: 0.2563 - accuracy: 0.9250 -
val_loss: 0.3880 - val_accuracy: 0.8833
Epoch 87/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2551 - accuracy: 0.9259 -
val_loss: 0.3876 - val_accuracy: 0.8833
Epoch 88/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2534 - accuracy: 0.9250 -
val_loss: 0.3863 - val_accuracy: 0.8833
Epoch 89/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2523 - accuracy: 0.9259 -
val_loss: 0.3862 - val_accuracy: 0.8833
Epoch 90/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2508 - accuracy: 0.9250 -
val_loss: 0.3853 - val_accuracy: 0.8833
Epoch 91/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2496 - accuracy: 0.9259 -
val_loss: 0.3844 - val_accuracy: 0.8833
Epoch 92/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2482 - accuracy: 0.9259 -
val_loss: 0.3835 - val_accuracy: 0.8833
Epoch 93/100
17/17 [==============================] - 0s 8ms/step - loss: 0.2469 - accuracy: 0.9278 -
val_loss: 0.3838 - val_accuracy: 0.8833
Epoch 94/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2457 - accuracy: 0.9278 -
val_loss: 0.3825 - val_accuracy: 0.8750
Epoch 95/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2444 - accuracy: 0.9278 -
val_loss: 0.3828 - val_accuracy: 0.8750
Epoch 96/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2434 - accuracy: 0.9306 -
val_loss: 0.3821 - val_accuracy: 0.8833
Epoch 97/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2419 - accuracy: 0.9306 -
val_loss: 0.3814 - val_accuracy: 0.8750
Epoch 98/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2406 - accuracy: 0.9315 -
val_loss: 0.3815 - val_accuracy: 0.8750
Epoch 99/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2395 - accuracy: 0.9333 -
val_loss: 0.3807 - val_accuracy: 0.8750
Epoch 100/100
17/17 [==============================] - 0s 9ms/step - loss: 0.2381 - accuracy: 0.9324 -
val_loss: 0.3808 - val_accuracy: 0.8750
```

step5: History Object

The history object is an output of the `.fit()` operation, and provides a record of all the loss and metric values in memory.

In [20]:
```
history.history
```

Out[20]:

```
{'loss': [0.45498088002204895,
  0.4513610303401947,
  0.4476611614227295,
  0.44384947419166565,
  0.4400864243507385,
  0.43649885058403015,
  0.43291231989860535,
  0.4295520782470703,
  0.4259725511074066,
  0.4224812388420105,
  0.41897985339164734,
  0.4158307909965515,
  0.4126322567462921,
  0.4094254672527313,
  0.4063203036785126,
  0.4030573070049286,
  0.4000851809978485,
  0.3967605233192444,
  0.3937791883945465,
  0.3908597528934479,
  0.38792142271995544,
  0.38487371802330017,
  0.3821130394935608,
  0.3794204890727997,
  0.3764365315437317,
  0.3738376200199127,
  0.3710717558860779,
  0.36834201216697693,
  0.3657686114311218,
  0.36305680871009827,
  0.3606323003768921,
  0.35795050859451294,
  0.35542359948158264,
  0.352912962436676,
  0.35070428252220154,
  0.34808772802352905,
  0.3456805646419525,
  0.3434962034225464,
  0.3411964476108551,
  0.33891433477401733,
  0.33666399121284485,
  0.3346875309944153,
  0.3324844241142273,
  0.33004382252693176,
  0.32795026898384094,
  0.3256860375404358,
  0.32355672121047974,
  0.32159024477005005,
  0.319495290517807,
  0.31743186712265015,
  0.31536129117012024,
  0.31334036588668823,
  0.3115352392196655,
  0.3097466230392456,
  0.3076777458190918,
  0.3058916926383972,
  0.30395397543907166,
  0.3023427724838257,
  0.300392210483551,
  0.2985687255859375,
  0.29658931493759155,
  0.29494336247444153,
  0.2929050922393799,
  0.2912933826446533,
```

```
 0.2894275486469269,
 0.28770971298217773,
 0.2859257757663727,
 0.2840254306793213,
 0.282316654920578,
 0.2808993458747864,
 0.27904921770095825,
 0.2773685157299042,
 0.27582022547721863,
 0.27418795228004456,
 0.27269062399864197,
 0.2711831033229828,
 0.2694801390171051,
 0.26794734597206116,
 0.2665925920009613,
 0.26487696170806885,
 0.26357418298721313,
 0.2620081305503845,
 0.26055940985679626,
 0.259120374917984,
 0.2576722204685211,
 0.2563108205795288,
 0.25511571764945984,
 0.2534380257129669,
 0.25226324796676636,
 0.2508046627044678,
 0.2496384233236313,
 0.2482336312532425,
 0.24686256051063538,
 0.24574226140975952,
 0.24440957605838776,
 0.24342264235019684,
 0.24194291234016418,
 0.24063271284103394,
 0.23952899873256683,
 0.23809851706027985],
'accuracy': [0.8527777791023254,
 0.8518518805503845,
 0.8537036776542664,
 0.855555534362793,
 0.855555534362793,
 0.8592592477798462,
 0.8601852059364319,
 0.8629629611968994,
 0.8685185313224792,
 0.8694444298744202,
 0.8703703880310059,
 0.8694444298744202,
 0.8722222447395325,
 0.8731481432914734,
 0.875,
 0.8768518567085266,
 0.8777777552604675,
 0.8768518567085266,
 0.8777777552604675,
 0.8777777552604675,
 0.8814814686775208,
 0.8824074268341064,
 0.8824074268341064,
 0.8814814686775208,
 0.8833333253860474,
 0.885185182094574,
 0.885185182094574,
 0.8870370388031006,
```

0.8888888955116272,
0.8879629373550415,
0.8907407522201538,
0.8916666507720947,
0.8925926089286804,
0.895370364189148,
0.8935185074806213,
0.8962963223457336,
0.8972222208976746,
0.8972222208976746,
0.8972222208976746,
0.8972222208976746,
0.8962963223457336,
0.8990740776062012,
0.8981481194496155,
0.8999999761581421,
0.8999999761581421,
0.9009259343147278,
0.9018518328666687,
0.9027777910232544,
0.9027777910232544,
0.9064815044403076,
0.9055555462837219,
0.9074074029922485,
0.9064815044403076,
0.9101851582527161,
0.9101851582527161,
0.9092592597007751,
0.9111111164093018,
0.9111111164093018,
0.9129629731178284,
0.9138888716697693,
0.914814829826355,
0.9166666865348816,
0.9166666865348816,
0.9166666865348816,
0.9166666865348816,
0.9166666865348816,
0.9175925850868225,
0.9175925850868225,
0.9175925850868225,
0.9175925850868225,
0.9185185432434082,
0.9194444417953491,
0.9194444417953491,
0.9203703999519348,
0.9212962985038757,
0.9203703999519348,
0.9203703999519348,
0.9212962985038757,
0.9222221970558167,
0.9222221970558167,
0.9222221970558167,
0.9222221970558167,
0.9231481552124023,
0.9240740537643433,
0.9240740537643433,
0.925000011920929,
0.9259259104728699,
0.925000011920929,
0.9259259104728699,
0.925000011920929,
0.9259259104728699,
0.9259259104728699,

```
 0.9277777671813965,
 0.9277777671813965,
 0.9277777671813965,
 0.9305555820465088,
 0.9305555820465088,
 0.9314814805984497,
 0.9333333373069763,
 0.9324073791503906],
'val_loss': [0.511431097984314,
 0.508465588092804,
 0.5053831934928894,
 0.5028645992279053,
 0.4996529221534729,
 0.49750080704689026,
 0.49451935291290283,
 0.4919939339160919,
 0.48956242203712463,
 0.4873666763305664,
 0.48531609773635864,
 0.4831386208534241,
 0.4808046221733093,
 0.4783601462841034,
 0.4759831726551056,
 0.47368767857551575,
 0.4716239273548126,
 0.46828487515449524,
 0.46732938289642334,
 0.4650042653083801,
 0.46330389380455017,
 0.46101418137550354,
 0.45947960019111633,
 0.4574183523654938,
 0.4551481306552887,
 0.4540451467037201,
 0.4516145884990692,
 0.45035991072654724,
 0.4487977921962738,
 0.44688931107521057,
 0.44568735361099243,
 0.44353950023651123,
 0.4422570466995239,
 0.44005683064460754,
 0.4392617642879486,
 0.43686968088150024,
 0.4356532394886017,
 0.43358534574508667,
 0.43253293633461,
 0.43040338158607483,
 0.43056631088256836,
 0.43018901348114014,
 0.4287014901638031,
 0.4269765317440033,
 0.426104873418808,
 0.42483335733413696,
 0.4233371019363403,
 0.422067254781723,
 0.4209333062171936,
 0.41967883706092834,
 0.4180683493614197,
 0.41759905219078064,
 0.4165533185005188,
 0.4151020050048828,
 0.4144417643547058,
 0.4133646786212921,
```

```
 0.41235461831092834,
 0.4120120406150818,
 0.4098167419433594,
 0.40891531109809875,
 0.4079742431640625,
 0.4075808525085449,
 0.40567857027053833,
 0.4054197669029236,
 0.40448838472366333,
 0.4034976363182068,
 0.40259668231010437,
 0.401641845703125,
 0.40075868368148804,
 0.40024954080581665,
 0.3987489640712738,
 0.39872533082962036,
 0.3975272476673126,
 0.39660975337028503,
 0.39592254161834717,
 0.39479580521583557,
 0.3944854140281677,
 0.3935789465904236,
 0.39307835698127747,
 0.39196905493736267,
 0.39143529534339905,
 0.390825480222702,
 0.39022400975227356,
 0.38905802369117737,
 0.38922908902168274,
 0.38798654079437256,
 0.38763946294784546,
 0.38625475764274597,
 0.38622787594795227,
 0.3853186070919037,
 0.384432852268219,
 0.3835156261920929,
 0.38383951783180237,
 0.38253268599510193,
 0.38282865285873413,
 0.38211503624916077,
 0.3814336061477661,
 0.38152244687080383,
 0.3807063102722168,
 0.3808368742465973],
'val_accuracy': [0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.824999988079071,
 0.8416666388511658,
 0.8500000238418579,
 0.8500000238418579,
 0.8500000238418579,
 0.8500000238418579,
 0.8500000238418579,
 0.8500000238418579,
```

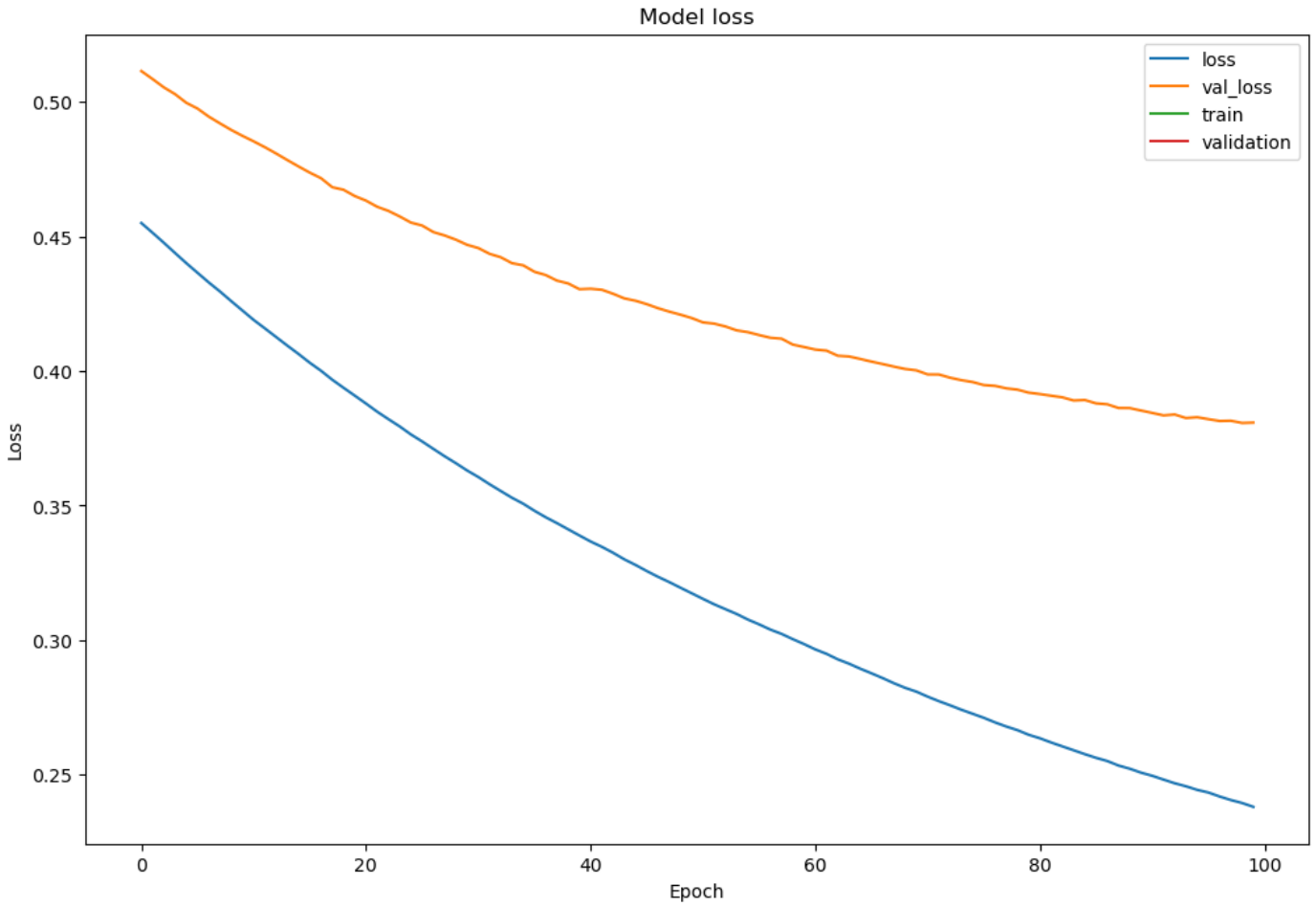
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.8500000238418579,
0.858333492279053,
0.858333492279053,
0.858333492279053,
0.858333492279053,
0.858333492279053,
0.858333492279053,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.875,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,
0.883333253860474,

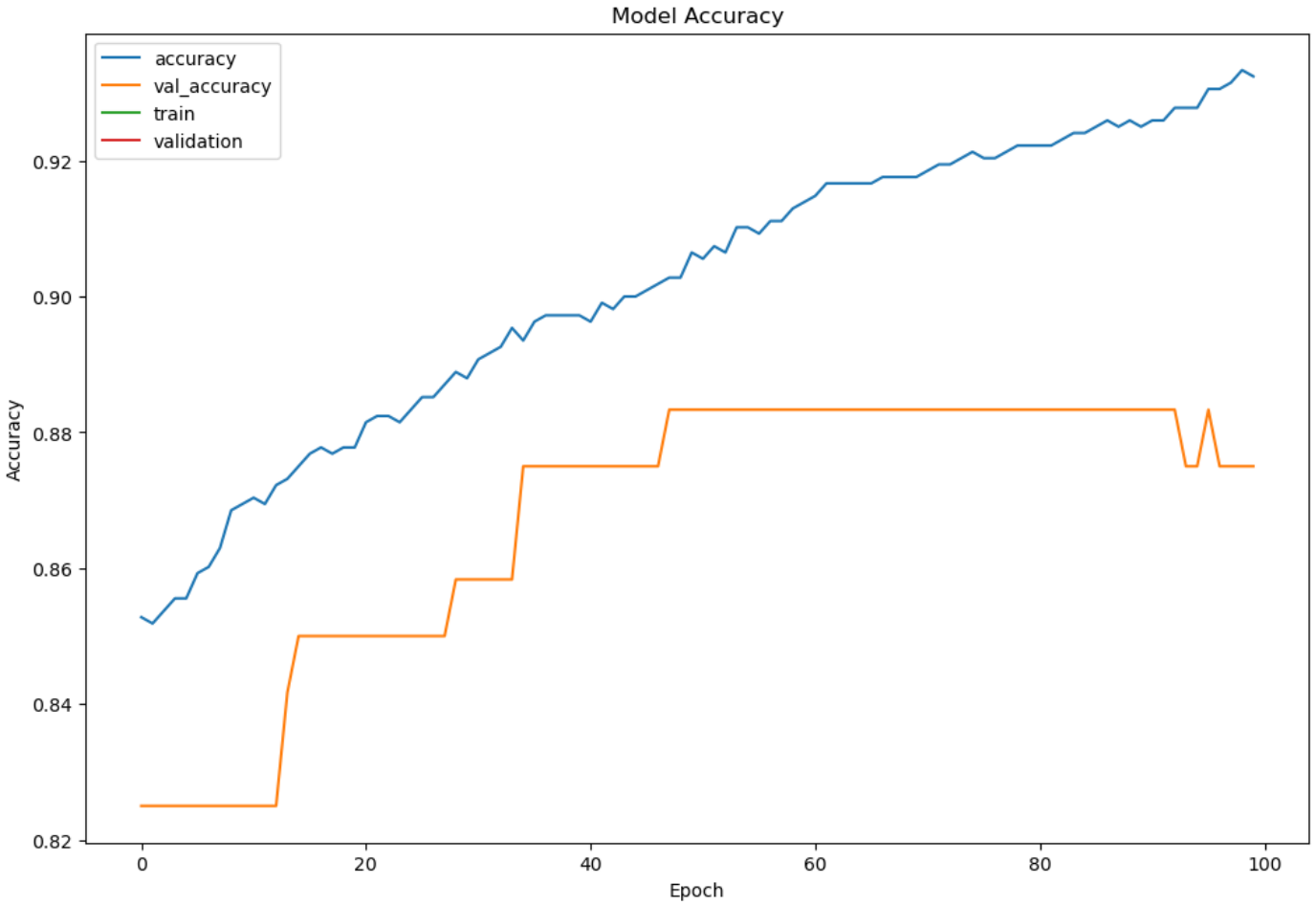
```
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.8833333253860474,
0.875,
0.875,
0.8833333253860474,
0.875,
0.875,
0.875,
0.875]}
```

Now visualizing the loss over time using `history.history`:

In [24]:
```
# The history.history["loss"] entry is a dictionary with as many values as epochs that t
# model was trained on.
df_loss_acc = pd.DataFrame(history.history)
df_loss= df_loss_acc[['loss','val_loss']]
df_loss.loc[:, ['train', 'validation']] = df_loss[['loss', 'val_loss']]
df_acc= df_loss_acc[['accuracy','val_accuracy']]
df_acc.loc[:, ['train', 'validation']] = df_acc[['accuracy', 'val_accuracy']]
df_loss.plot(title='Model loss',figsize=(12,8)).set(xlabel='Epoch',ylabel='Loss')
df_acc.plot(title='Model Accuracy',figsize=(12,8)).set(xlabel='Epoch',ylabel='Accuracy')
```

Out[24]:
```
[Text(0.5, 0, 'Epoch'), Text(0, 0.5, 'Accuracy')]
```





In [ ]: