1. The size of the output layers and weights are calculated in the figure 1.

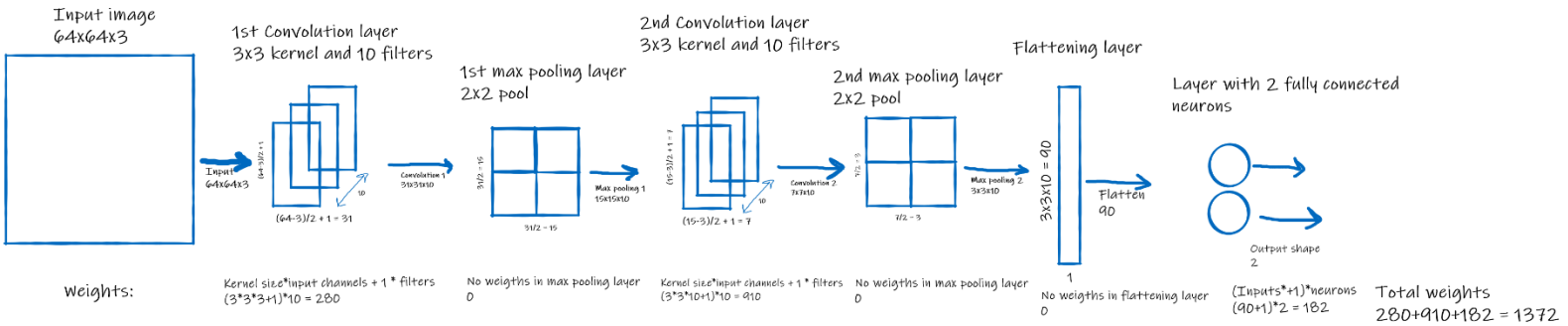


Figure 1: Output sizes and weights of the network

2. Model is defined as shown in figure 2.

```
# declare input shape
input = tf.keras.Input(shape=(64, 64, 3))

# Block 1 (convolution + max pooling)
conv1 = tf.keras.layers.Conv2D(10, (3, 3), strides=2, activation="relu")(input)
max1 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv1)

# Block 2 (convolution2 + max pooling2)
conv2 = tf.keras.layers.Conv2D(10, (3, 3), strides=2, activation="relu")(max1)
max2 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv2)

# Flattening
fc = tf.keras.layers.Flatten()(max2)

# Finally, we add a classification layer.
output = tf.keras.layers.Dense(2, activation="sigmoid")(fc)

# bind all
cnn_model = tf.keras.Model(input, output)

# This loss takes care of one-hot encoding
loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=True)

cnn_model.compile(loss=loss_fn, optimizer="SGD", metrics=["accuracy"])
cnn_model.summary()
```

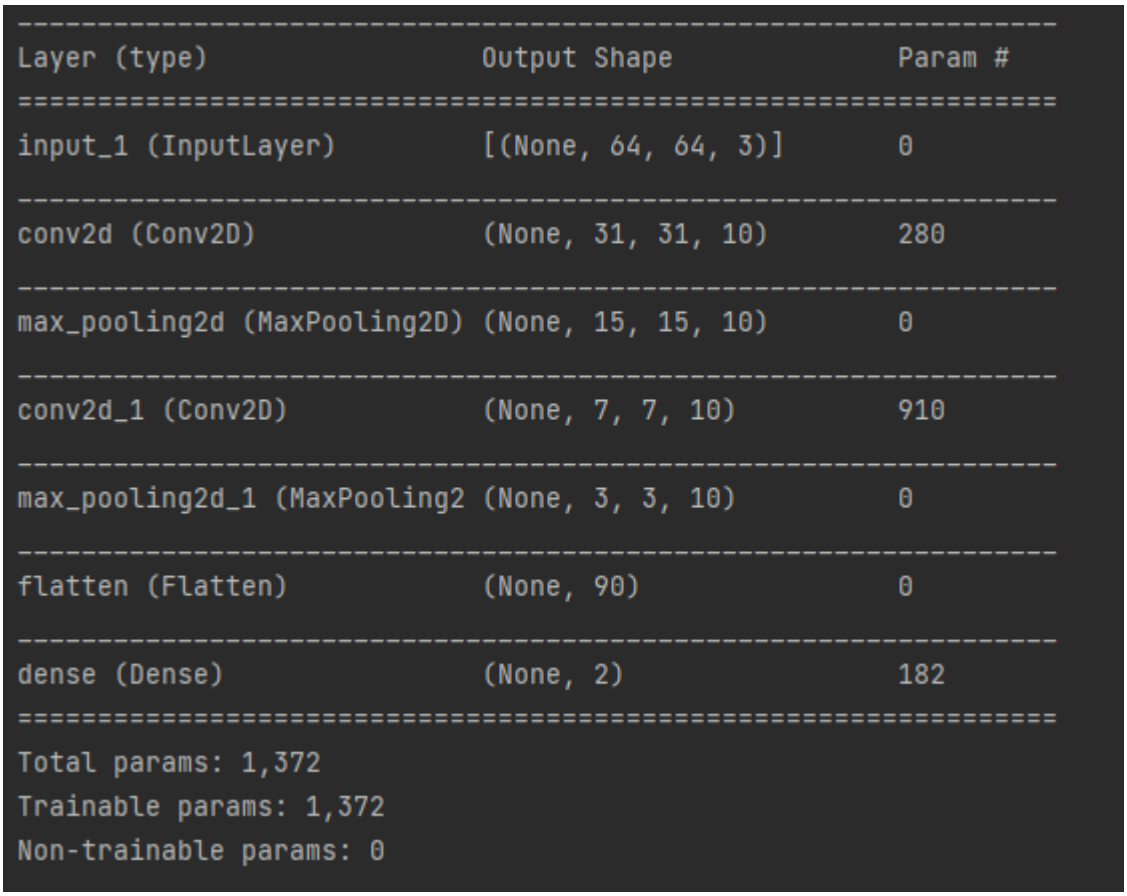
Figure 2: Definition of the cnn

3. Computing the test accuracy we get around 69% accuracy after the 20 epochs. It also looks like the loss hasn't really gone down much after the 7th epoch so more training epochs probably wont help to increase the accuracy anymore. The tr3aining epochs and testing results are shown in figure 3.

```
17/17 [==============================] - 1s 45ms/step - loss: 0.6908 - accuracy: 0.5473
Epoch 2/20
17/17 [==============================] - 1s 44ms/step - loss: 0.6828 - accuracy: 0.7330
Epoch 3/20
17/17 [==============================] - 1s 45ms/step - loss: 0.6773 - accuracy: 0.6989
Epoch 4/20
17/17 [==============================] - 1s 45ms/step - loss: 0.6711 - accuracy: 0.6799
Epoch 5/20
17/17 [==============================] - 1s 48ms/step - loss: 0.6663 - accuracy: 0.6780
Epoch 6/20
17/17 [==============================] - 1s 49ms/step - loss: 0.6620 - accuracy: 0.6837
Epoch 7/20
17/17 [==============================] - 1s 50ms/step - loss: 0.6573 - accuracy: 0.6818
Epoch 8/20
17/17 [==============================] - 1s 49ms/step - loss: 0.6527 - accuracy: 0.6818
Epoch 9/20
17/17 [==============================] - 1s 48ms/step - loss: 0.6475 - accuracy: 0.6818
Epoch 10/20
17/17 [==============================] - 1s 47ms/step - loss: 0.6421 - accuracy: 0.6818
Epoch 11/20
17/17 [==============================] - 1s 46ms/step - loss: 0.6377 - accuracy: 0.6818
Epoch 12/20
17/17 [==============================] - 1s 51ms/step - loss: 0.6326 - accuracy: 0.6818
Epoch 13/20
17/17 [==============================] - 1s 46ms/step - loss: 0.6306 - accuracy: 0.6818
Epoch 14/20
17/17 [==============================] - 1s 44ms/step - loss: 0.6248 - accuracy: 0.6818
Epoch 15/20
17/17 [==============================] - 1s 41ms/step - loss: 0.6188 - accuracy: 0.6818
Epoch 16/20
17/17 [==============================] - 1s 43ms/step - loss: 0.6114 - accuracy: 0.6818
Epoch 17/20
17/17 [==============================] - 1s 45ms/step - loss: 0.6076 - accuracy: 0.6818
Epoch 18/20
17/17 [==============================] - 1s 43ms/step - loss: 0.5994 - accuracy: 0.6856
Epoch 19/20
17/17 [==============================] - 1s 42ms/step - loss: 0.5934 - accuracy: 0.6875
Epoch 20/20
17/17 [==============================] - 1s 42ms/step - loss: 0.5872 - accuracy: 0.6856
5/5 [==============================] - 0s 37ms/step - loss: 0.5795 - accuracy: 0.6894
Validation loss: 0.5795310139656067
Validation accuracy: 0.689393937587738
```

Figure 3: Training the model and testing it.

Calling the summary for the model we can check if our output shape and weight calculations are correct in figure 1. This summary is presented in figure 4. Looks like they match.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 64, 64, 3)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 31, 31, 10)        280
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 10)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 10)          910
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 10)          0
_________________________________________________________________
flatten (Flatten)            (None, 90)                0
_________________________________________________________________
dense (Dense)                (None, 2)                 182
=================================================================
Total params: 1,372
Trainable params: 1,372
Non-trainable params: 0
```

Figure 4: Summary of the defined model