In [1]:
```
print("Hello World!")
```

```
Hello World!
```

In [2]:
```
# 1
import pandas as pd
import numpy as np
import os
```

In [3]:
```
# Pillow for image processing
from PIL import Image

# Dataset came already divided into train/test, so we set directory paths
train_dir = (r"C:\Users\colto\Downloads\archive\train")
test_dir = (r"C:\Users\colto\Downloads\archive\train")
```

In [4]:
```
from itertools import chain
from matplotlib import pyplot as plt

dim = [100, 100] # Lots of different dimension images in the set, so we need to sca
```

In [5]:
```
# 2 & 3 (Sequential is on the way to CNN)
import keras
from keras import Sequential, backend
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization,
import tensorflow as tf

# Creating simple model
def simple_model():
    backend.clear_session() # clear model memory / similar to restarting kernel

    model = Sequential()

    # Convolution layers
    model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform', activation="relu",
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform', activation="relu")
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(128, (3,3), kernel_initializer='he_uniform', activation="relu"
    model.add(MaxPooling2D(2, 2))

    # Flattening output, adding Dense layers
    model.add(Flatten())
    model.add(Dense(512, activation="relu", kernel_initializer='he_uniform'))
    model.add(Dropout(0.5))
    model.add(Dense(256, activation="relu", kernel_initializer='he_uniform'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    model.summary()

    return model
```

In [6]:
```
# Creating advanced model
def advanced_model():
    # Clearning any model memory
    backend.clear_session()

    # Creating Sequential model
    model = Sequential()

    # Adding convolution layers
    model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform', input_shape=(*dim,
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(2, 2))
    model.add(Conv2D(128, (3,3), kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Conv2D(128, (3,3), kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Conv2D(128, (3,3), kernel_initializer='he_uniform'))
    model.add(MaxPooling2D(2, 2))

    # Flattening the output and adding Dense layers
    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
```

In [7]:
```
from keras.preprocessing.image import ImageDataGenerator

train_config = ImageDataGenerator( # Our data is augmented in accordance to the con
      rescale=1./255,
      rotation_range=50,
      width_shift_range=0.25,
      height_shift_range=0.25,
      shear_range=0.3,
      zoom_range=0.3,
      horizontal_flip=True,
      brightness_range=(0.8, 1.2),
      fill_mode='nearest'
)

input_pipeline = ImageDataGenerator(rescale=1./255) # Images flow from the director
```

```
train_generator = train_config.flow_from_directory(
        train_dir,  # Source directory
        target_size=tuple(dim),  # All images will be resized to dim (100x100)
        class_mode='binary') # Since we use binary_crossentropy loss, we need binar

valid_generator = input_pipeline.flow_from_directory(
        test_dir,  # This is the source directory for training images
        target_size=tuple(dim),  # All images will be resized to 100x100
        class_mode='binary')
```

```
Found 294 images belonging to 2 classes.
Found 294 images belonging to 2 classes.
```

In [8]:
```
model = simple_model() # Creating new model

# Parameters we can influence to get different models
BATCH_SIZE = 64
EPOCHS = 120

model.compile(loss='binary_crossentropy',
              optimizer="Adam", # Adam is used in the transfer learning example, so
              metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 98, 98, 32)        896

 max_pooling2d (MaxPooling2D  (None, 49, 49, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 47, 47, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 21, 21, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 10, 10, 128)      0
 2D)

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 512)               6554112

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 6,778,945
Trainable params: 6,778,945
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
# Training the model
history = model.fit(
      train_generator,
      validation_data=valid_generator,
      batch_size=BATCH_SIZE,
      epochs=EPOCHS,
      callbacks=[tf.keras.callbacks.LearningRateScheduler(lambda epoch, lr: lr if e
      verbose=1)
```

```
Epoch 1/120
10/10 [==============================] - 3s 217ms/step - loss: 1.7861 - accuracy: 0.
5476 - val_loss: 0.6503 - val_accuracy: 0.6020 - lr: 0.0010
Epoch 2/120
10/10 [==============================] - 2s 212ms/step - loss: 0.6723 - accuracy: 0.
5646 - val_loss: 0.6546 - val_accuracy: 0.5816 - lr: 0.0010
Epoch 3/120
10/10 [==============================] - 2s 201ms/step - loss: 0.6571 - accuracy: 0.
6122 - val_loss: 0.7073 - val_accuracy: 0.5374 - lr: 0.0010
Epoch 4/120
10/10 [==============================] - 2s 217ms/step - loss: 0.6393 - accuracy: 0.
6190 - val_loss: 0.6017 - val_accuracy: 0.6905 - lr: 0.0010
Epoch 5/120
10/10 [==============================] - 2s 215ms/step - loss: 0.6065 - accuracy: 0.
6973 - val_loss: 0.5794 - val_accuracy: 0.6939 - lr: 0.0010
Epoch 6/120
10/10 [==============================] - 2s 218ms/step - loss: 0.6457 - accuracy: 0.
6599 - val_loss: 0.5756 - val_accuracy: 0.7245 - lr: 0.0010
Epoch 7/120
10/10 [==============================] - 2s 208ms/step - loss: 0.6136 - accuracy: 0.
7007 - val_loss: 0.5832 - val_accuracy: 0.6939 - lr: 0.0010
Epoch 8/120
10/10 [==============================] - 2s 202ms/step - loss: 0.6689 - accuracy: 0.
6769 - val_loss: 0.5911 - val_accuracy: 0.6735 - lr: 0.0010
Epoch 9/120
10/10 [==============================] - 2s 239ms/step - loss: 0.6420 - accuracy: 0.
6190 - val_loss: 0.6307 - val_accuracy: 0.6190 - lr: 0.0010
Epoch 10/120
10/10 [==============================] - 2s 223ms/step - loss: 0.6325 - accuracy: 0.
6463 - val_loss: 0.7195 - val_accuracy: 0.5136 - lr: 0.0010
Epoch 11/120
10/10 [==============================] - 2s 213ms/step - loss: 0.6558 - accuracy: 0.
6190 - val_loss: 0.5989 - val_accuracy: 0.7245 - lr: 0.0010
Epoch 12/120
10/10 [==============================] - 2s 233ms/step - loss: 0.6187 - accuracy: 0.
6361 - val_loss: 0.5865 - val_accuracy: 0.6735 - lr: 0.0010
Epoch 13/120
10/10 [==============================] - 2s 231ms/step - loss: 0.5763 - accuracy: 0.
7007 - val_loss: 0.5332 - val_accuracy: 0.7245 - lr: 0.0010
Epoch 14/120
10/10 [==============================] - 2s 221ms/step - loss: 0.5873 - accuracy: 0.
7041 - val_loss: 0.5131 - val_accuracy: 0.7381 - lr: 0.0010
Epoch 15/120
10/10 [==============================] - 2s 206ms/step - loss: 0.5490 - accuracy: 0.
7449 - val_loss: 0.5110 - val_accuracy: 0.7551 - lr: 0.0010
Epoch 16/120
10/10 [==============================] - 2s 203ms/step - loss: 0.6065 - accuracy: 0.
6871 - val_loss: 0.5328 - val_accuracy: 0.7347 - lr: 0.0010
Epoch 17/120
10/10 [==============================] - 2s 210ms/step - loss: 0.5790 - accuracy: 0.
7109 - val_loss: 0.5626 - val_accuracy: 0.7279 - lr: 0.0010
Epoch 18/120
10/10 [==============================] - 2s 195ms/step - loss: 0.5456 - accuracy: 0.
7415 - val_loss: 0.4879 - val_accuracy: 0.7551 - lr: 0.0010
Epoch 19/120
10/10 [==============================] - 2s 213ms/step - loss: 0.5737 - accuracy: 0.
```

```
7619 - val_loss: 0.4855 - val_accuracy: 0.7755 - lr: 0.0010
Epoch 20/120
10/10 [==============================] - 2s 219ms/step - loss: 0.6028 - accuracy: 0.
7313 - val_loss: 0.5590 - val_accuracy: 0.6871 - lr: 0.0010
Epoch 21/120
10/10 [==============================] - 2s 206ms/step - loss: 0.5879 - accuracy: 0.
7143 - val_loss: 0.4827 - val_accuracy: 0.7755 - lr: 0.0010
Epoch 22/120
10/10 [==============================] - 2s 218ms/step - loss: 0.5580 - accuracy: 0.
7483 - val_loss: 0.5105 - val_accuracy: 0.7517 - lr: 0.0010
Epoch 23/120
10/10 [==============================] - 2s 210ms/step - loss: 0.4998 - accuracy: 0.
7789 - val_loss: 0.4922 - val_accuracy: 0.7721 - lr: 0.0010
Epoch 24/120
10/10 [==============================] - 2s 204ms/step - loss: 0.4952 - accuracy: 0.
7483 - val_loss: 0.4429 - val_accuracy: 0.7891 - lr: 0.0010
Epoch 25/120
10/10 [==============================] - 2s 191ms/step - loss: 0.5582 - accuracy: 0.
7551 - val_loss: 0.4962 - val_accuracy: 0.7891 - lr: 0.0010
Epoch 26/120
10/10 [==============================] - 2s 187ms/step - loss: 0.5186 - accuracy: 0.
7517 - val_loss: 0.4759 - val_accuracy: 0.7313 - lr: 0.0010
Epoch 27/120
10/10 [==============================] - 2s 196ms/step - loss: 0.5366 - accuracy: 0.
7007 - val_loss: 0.4537 - val_accuracy: 0.7551 - lr: 0.0010
Epoch 28/120
10/10 [==============================] - 2s 194ms/step - loss: 0.4793 - accuracy: 0.
7721 - val_loss: 0.4423 - val_accuracy: 0.7993 - lr: 0.0010
Epoch 29/120
10/10 [==============================] - 2s 197ms/step - loss: 0.5040 - accuracy: 0.
7415 - val_loss: 0.4417 - val_accuracy: 0.7925 - lr: 0.0010
Epoch 30/120
10/10 [==============================] - 2s 202ms/step - loss: 0.5241 - accuracy: 0.
7381 - val_loss: 0.4499 - val_accuracy: 0.7755 - lr: 0.0010
Epoch 31/120
10/10 [==============================] - 2s 200ms/step - loss: 0.4867 - accuracy: 0.
7789 - val_loss: 0.4180 - val_accuracy: 0.8027 - lr: 0.0010
Epoch 32/120
10/10 [==============================] - 2s 197ms/step - loss: 0.4956 - accuracy: 0.
7585 - val_loss: 0.4235 - val_accuracy: 0.7959 - lr: 0.0010
Epoch 33/120
10/10 [==============================] - 2s 210ms/step - loss: 0.5246 - accuracy: 0.
7279 - val_loss: 0.4977 - val_accuracy: 0.7245 - lr: 0.0010
Epoch 34/120
10/10 [==============================] - 2s 192ms/step - loss: 0.5286 - accuracy: 0.
7653 - val_loss: 0.4410 - val_accuracy: 0.8129 - lr: 0.0010
Epoch 35/120
10/10 [==============================] - 2s 192ms/step - loss: 0.5349 - accuracy: 0.
7891 - val_loss: 0.4191 - val_accuracy: 0.7789 - lr: 0.0010
Epoch 36/120
10/10 [==============================] - 2s 201ms/step - loss: 0.5115 - accuracy: 0.
7517 - val_loss: 0.4372 - val_accuracy: 0.7789 - lr: 0.0010
Epoch 37/120
10/10 [==============================] - 2s 214ms/step - loss: 0.4785 - accuracy: 0.
7687 - val_loss: 0.4757 - val_accuracy: 0.7551 - lr: 0.0010
Epoch 38/120
```

```
10/10 [==============================] - 2s 197ms/step - loss: 0.4635 - accuracy: 0.
7891 - val_loss: 0.4643 - val_accuracy: 0.7925 - lr: 0.0010
Epoch 39/120
10/10 [==============================] - 2s 204ms/step - loss: 0.4426 - accuracy: 0.
7993 - val_loss: 0.3658 - val_accuracy: 0.8537 - lr: 0.0010
Epoch 40/120
10/10 [==============================] - 2s 198ms/step - loss: 0.4299 - accuracy: 0.
7959 - val_loss: 0.3857 - val_accuracy: 0.8129 - lr: 0.0010
Epoch 41/120
10/10 [==============================] - 2s 197ms/step - loss: 0.5235 - accuracy: 0.
7755 - val_loss: 0.4255 - val_accuracy: 0.7993 - lr: 0.0010
Epoch 42/120
10/10 [==============================] - 2s 195ms/step - loss: 0.4795 - accuracy: 0.
7925 - val_loss: 0.4042 - val_accuracy: 0.8197 - lr: 0.0010
Epoch 43/120
10/10 [==============================] - 2s 194ms/step - loss: 0.4105 - accuracy: 0.
8061 - val_loss: 0.3713 - val_accuracy: 0.8401 - lr: 0.0010
Epoch 44/120
10/10 [==============================] - 2s 196ms/step - loss: 0.4930 - accuracy: 0.
7551 - val_loss: 0.4509 - val_accuracy: 0.7653 - lr: 0.0010
Epoch 45/120
10/10 [==============================] - 2s 210ms/step - loss: 0.4921 - accuracy: 0.
7483 - val_loss: 0.3772 - val_accuracy: 0.8333 - lr: 0.0010
Epoch 46/120
10/10 [==============================] - 2s 249ms/step - loss: 0.4609 - accuracy: 0.
7653 - val_loss: 0.3633 - val_accuracy: 0.8299 - lr: 0.0010
Epoch 47/120
10/10 [==============================] - 2s 206ms/step - loss: 0.4190 - accuracy: 0.
8367 - val_loss: 0.3429 - val_accuracy: 0.8503 - lr: 0.0010
Epoch 48/120
10/10 [==============================] - 2s 197ms/step - loss: 0.4537 - accuracy: 0.
7857 - val_loss: 0.3413 - val_accuracy: 0.8265 - lr: 0.0010
Epoch 49/120
10/10 [==============================] - 2s 223ms/step - loss: 0.3911 - accuracy: 0.
8197 - val_loss: 0.3378 - val_accuracy: 0.8537 - lr: 0.0010
Epoch 50/120
10/10 [==============================] - 2s 232ms/step - loss: 0.4123 - accuracy: 0.
7925 - val_loss: 0.4166 - val_accuracy: 0.8095 - lr: 0.0010
Epoch 51/120
10/10 [==============================] - 2s 217ms/step - loss: 0.4239 - accuracy: 0.
8061 - val_loss: 0.3646 - val_accuracy: 0.8299 - lr: 0.0010
Epoch 52/120
10/10 [==============================] - 2s 210ms/step - loss: 0.4631 - accuracy: 0.
7755 - val_loss: 0.3835 - val_accuracy: 0.8231 - lr: 0.0010
Epoch 53/120
10/10 [==============================] - 2s 210ms/step - loss: 0.4280 - accuracy: 0.
7721 - val_loss: 0.3405 - val_accuracy: 0.8435 - lr: 0.0010
Epoch 54/120
10/10 [==============================] - 2s 208ms/step - loss: 0.3954 - accuracy: 0.
8027 - val_loss: 0.3777 - val_accuracy: 0.8299 - lr: 0.0010
Epoch 55/120
10/10 [==============================] - 2s 208ms/step - loss: 0.4385 - accuracy: 0.
8027 - val_loss: 0.3709 - val_accuracy: 0.8333 - lr: 0.0010
Epoch 56/120
10/10 [==============================] - 3s 253ms/step - loss: 0.4534 - accuracy: 0.
7721 - val_loss: 0.4048 - val_accuracy: 0.8197 - lr: 0.0010
```

```
Epoch 57/120
10/10 [==============================] - 2s 209ms/step - loss: 0.4560 - accuracy: 0.
7959 - val_loss: 0.3502 - val_accuracy: 0.8435 - lr: 0.0010
Epoch 58/120
10/10 [==============================] - 2s 212ms/step - loss: 0.4386 - accuracy: 0.
7823 - val_loss: 0.3335 - val_accuracy: 0.8265 - lr: 0.0010
Epoch 59/120
10/10 [==============================] - 2s 202ms/step - loss: 0.4198 - accuracy: 0.
8129 - val_loss: 0.3750 - val_accuracy: 0.8299 - lr: 0.0010
Epoch 60/120
10/10 [==============================] - 2s 195ms/step - loss: 0.3930 - accuracy: 0.
8095 - val_loss: 0.3824 - val_accuracy: 0.8333 - lr: 0.0010
Epoch 61/120
10/10 [==============================] - 2s 200ms/step - loss: 0.3775 - accuracy: 0.
8197 - val_loss: 0.3085 - val_accuracy: 0.8605 - lr: 9.5123e-04
Epoch 62/120
10/10 [==============================] - 2s 202ms/step - loss: 0.3719 - accuracy: 0.
8367 - val_loss: 0.3206 - val_accuracy: 0.8571 - lr: 9.0484e-04
Epoch 63/120
10/10 [==============================] - 2s 209ms/step - loss: 0.3727 - accuracy: 0.
8095 - val_loss: 0.2808 - val_accuracy: 0.8878 - lr: 8.6071e-04
Epoch 64/120
10/10 [==============================] - 2s 217ms/step - loss: 0.4404 - accuracy: 0.
8163 - val_loss: 0.3090 - val_accuracy: 0.8639 - lr: 8.1873e-04
Epoch 65/120
10/10 [==============================] - 2s 208ms/step - loss: 0.3279 - accuracy: 0.
8673 - val_loss: 0.2820 - val_accuracy: 0.8605 - lr: 7.7880e-04
Epoch 66/120
10/10 [==============================] - 2s 221ms/step - loss: 0.4030 - accuracy: 0.
8095 - val_loss: 0.2772 - val_accuracy: 0.8741 - lr: 7.4082e-04
Epoch 67/120
10/10 [==============================] - 2s 210ms/step - loss: 0.3595 - accuracy: 0.
8367 - val_loss: 0.2656 - val_accuracy: 0.8673 - lr: 7.0469e-04
Epoch 68/120
10/10 [==============================] - 2s 204ms/step - loss: 0.3470 - accuracy: 0.
8435 - val_loss: 0.3002 - val_accuracy: 0.8707 - lr: 6.7032e-04
Epoch 69/120
10/10 [==============================] - 2s 207ms/step - loss: 0.3956 - accuracy: 0.
8061 - val_loss: 0.2875 - val_accuracy: 0.8673 - lr: 6.3763e-04
Epoch 70/120
10/10 [==============================] - 2s 201ms/step - loss: 0.3908 - accuracy: 0.
8163 - val_loss: 0.2606 - val_accuracy: 0.8673 - lr: 6.0653e-04
Epoch 71/120
10/10 [==============================] - 2s 204ms/step - loss: 0.3612 - accuracy: 0.
8503 - val_loss: 0.2857 - val_accuracy: 0.8639 - lr: 5.7695e-04
Epoch 72/120
10/10 [==============================] - 2s 199ms/step - loss: 0.3350 - accuracy: 0.
8401 - val_loss: 0.3477 - val_accuracy: 0.8299 - lr: 5.4881e-04
Epoch 73/120
10/10 [==============================] - 2s 206ms/step - loss: 0.3723 - accuracy: 0.
8095 - val_loss: 0.3061 - val_accuracy: 0.8537 - lr: 5.2205e-04
Epoch 74/120
10/10 [==============================] - 2s 205ms/step - loss: 0.3405 - accuracy: 0.
8333 - val_loss: 0.2900 - val_accuracy: 0.8741 - lr: 4.9659e-04
Epoch 75/120
10/10 [==============================] - 2s 210ms/step - loss: 0.3308 - accuracy: 0.
```

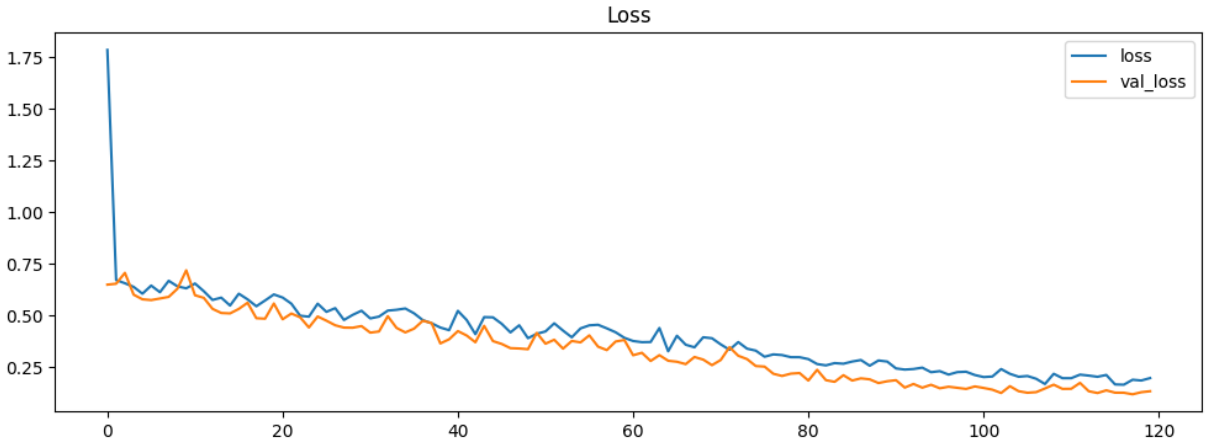
```
8639 - val_loss: 0.2564 - val_accuracy: 0.8810 - lr: 4.7237e-04
Epoch 76/120
10/10 [==============================] - 2s 201ms/step - loss: 0.3014 - accuracy: 0.
8503 - val_loss: 0.2537 - val_accuracy: 0.8776 - lr: 4.4933e-04
Epoch 77/120
10/10 [==============================] - 2s 212ms/step - loss: 0.3127 - accuracy: 0.
8639 - val_loss: 0.2191 - val_accuracy: 0.9184 - lr: 4.2742e-04
Epoch 78/120
10/10 [==============================] - 2s 198ms/step - loss: 0.3094 - accuracy: 0.
8537 - val_loss: 0.2084 - val_accuracy: 0.9218 - lr: 4.0657e-04
Epoch 79/120
10/10 [==============================] - 2s 200ms/step - loss: 0.2993 - accuracy: 0.
8673 - val_loss: 0.2196 - val_accuracy: 0.9116 - lr: 3.8674e-04
Epoch 80/120
10/10 [==============================] - 2s 206ms/step - loss: 0.2991 - accuracy: 0.
8707 - val_loss: 0.2224 - val_accuracy: 0.9218 - lr: 3.6788e-04
Epoch 81/120
10/10 [==============================] - 2s 204ms/step - loss: 0.2900 - accuracy: 0.
8673 - val_loss: 0.1861 - val_accuracy: 0.9252 - lr: 3.4994e-04
Epoch 82/120
10/10 [==============================] - 2s 209ms/step - loss: 0.2661 - accuracy: 0.
8571 - val_loss: 0.2382 - val_accuracy: 0.8912 - lr: 3.3287e-04
Epoch 83/120
10/10 [==============================] - 2s 209ms/step - loss: 0.2596 - accuracy: 0.
8707 - val_loss: 0.1881 - val_accuracy: 0.9252 - lr: 3.1664e-04
Epoch 84/120
10/10 [==============================] - 2s 205ms/step - loss: 0.2706 - accuracy: 0.
8741 - val_loss: 0.1800 - val_accuracy: 0.9320 - lr: 3.0119e-04
Epoch 85/120
10/10 [==============================] - 2s 200ms/step - loss: 0.2676 - accuracy: 0.
8776 - val_loss: 0.2120 - val_accuracy: 0.9082 - lr: 2.8651e-04
Epoch 86/120
10/10 [==============================] - 2s 216ms/step - loss: 0.2781 - accuracy: 0.
8537 - val_loss: 0.1863 - val_accuracy: 0.9116 - lr: 2.7253e-04
Epoch 87/120
10/10 [==============================] - 2s 203ms/step - loss: 0.2855 - accuracy: 0.
8639 - val_loss: 0.1969 - val_accuracy: 0.9082 - lr: 2.5924e-04
Epoch 88/120
10/10 [==============================] - 2s 210ms/step - loss: 0.2579 - accuracy: 0.
8844 - val_loss: 0.1920 - val_accuracy: 0.9150 - lr: 2.4660e-04
Epoch 89/120
10/10 [==============================] - 2s 205ms/step - loss: 0.2833 - accuracy: 0.
8571 - val_loss: 0.1741 - val_accuracy: 0.9320 - lr: 2.3457e-04
Epoch 90/120
10/10 [==============================] - 2s 192ms/step - loss: 0.2780 - accuracy: 0.
8741 - val_loss: 0.1827 - val_accuracy: 0.9354 - lr: 2.2313e-04
Epoch 91/120
10/10 [==============================] - 2s 196ms/step - loss: 0.2448 - accuracy: 0.
9048 - val_loss: 0.1879 - val_accuracy: 0.9218 - lr: 2.1225e-04
Epoch 92/120
10/10 [==============================] - 2s 201ms/step - loss: 0.2393 - accuracy: 0.
8844 - val_loss: 0.1520 - val_accuracy: 0.9388 - lr: 2.0190e-04
Epoch 93/120
10/10 [==============================] - 2s 208ms/step - loss: 0.2420 - accuracy: 0.
8741 - val_loss: 0.1696 - val_accuracy: 0.9354 - lr: 1.9205e-04
Epoch 94/120
```

```
10/10 [==============================] - 2s 218ms/step - loss: 0.2486 - accuracy: 0.
8707 - val_loss: 0.1516 - val_accuracy: 0.9422 - lr: 1.8268e-04
Epoch 95/120
10/10 [==============================] - 2s 199ms/step - loss: 0.2266 - accuracy: 0.
9014 - val_loss: 0.1658 - val_accuracy: 0.9354 - lr: 1.7377e-04
Epoch 96/120
10/10 [==============================] - 2s 227ms/step - loss: 0.2317 - accuracy: 0.
9014 - val_loss: 0.1487 - val_accuracy: 0.9524 - lr: 1.6530e-04
Epoch 97/120
10/10 [==============================] - 2s 210ms/step - loss: 0.2150 - accuracy: 0.
9218 - val_loss: 0.1562 - val_accuracy: 0.9558 - lr: 1.5724e-04
Epoch 98/120
10/10 [==============================] - 2s 215ms/step - loss: 0.2270 - accuracy: 0.
8946 - val_loss: 0.1511 - val_accuracy: 0.9558 - lr: 1.4957e-04
Epoch 99/120
10/10 [==============================] - 2s 202ms/step - loss: 0.2288 - accuracy: 0.
9150 - val_loss: 0.1457 - val_accuracy: 0.9558 - lr: 1.4227e-04
Epoch 100/120
10/10 [==============================] - 2s 204ms/step - loss: 0.2129 - accuracy: 0.
9116 - val_loss: 0.1576 - val_accuracy: 0.9592 - lr: 1.3534e-04
Epoch 101/120
10/10 [==============================] - 2s 196ms/step - loss: 0.2030 - accuracy: 0.
9252 - val_loss: 0.1504 - val_accuracy: 0.9626 - lr: 1.2874e-04
Epoch 102/120
10/10 [==============================] - 2s 204ms/step - loss: 0.2056 - accuracy: 0.
9116 - val_loss: 0.1420 - val_accuracy: 0.9626 - lr: 1.2246e-04
Epoch 103/120
10/10 [==============================] - 2s 207ms/step - loss: 0.2418 - accuracy: 0.
8810 - val_loss: 0.1259 - val_accuracy: 0.9524 - lr: 1.1648e-04
Epoch 104/120
10/10 [==============================] - 2s 195ms/step - loss: 0.2188 - accuracy: 0.
9014 - val_loss: 0.1589 - val_accuracy: 0.9490 - lr: 1.1080e-04
Epoch 105/120
10/10 [==============================] - 2s 195ms/step - loss: 0.2045 - accuracy: 0.
9082 - val_loss: 0.1346 - val_accuracy: 0.9660 - lr: 1.0540e-04
Epoch 106/120
10/10 [==============================] - 2s 194ms/step - loss: 0.2081 - accuracy: 0.
9014 - val_loss: 0.1272 - val_accuracy: 0.9660 - lr: 1.0026e-04
Epoch 107/120
10/10 [==============================] - 2s 208ms/step - loss: 0.1948 - accuracy: 0.
9014 - val_loss: 0.1301 - val_accuracy: 0.9694 - lr: 9.5369e-05
Epoch 108/120
10/10 [==============================] - 2s 217ms/step - loss: 0.1694 - accuracy: 0.
9252 - val_loss: 0.1478 - val_accuracy: 0.9558 - lr: 9.0718e-05
Epoch 109/120
10/10 [==============================] - 2s 213ms/step - loss: 0.2189 - accuracy: 0.
9082 - val_loss: 0.1663 - val_accuracy: 0.9422 - lr: 8.6294e-05
Epoch 110/120
10/10 [==============================] - 2s 206ms/step - loss: 0.1976 - accuracy: 0.
9218 - val_loss: 0.1459 - val_accuracy: 0.9456 - lr: 8.2085e-05
Epoch 111/120
10/10 [==============================] - 2s 207ms/step - loss: 0.1975 - accuracy: 0.
9116 - val_loss: 0.1463 - val_accuracy: 0.9388 - lr: 7.8082e-05
Epoch 112/120
10/10 [==============================] - 3s 274ms/step - loss: 0.2148 - accuracy: 0.
9184 - val_loss: 0.1749 - val_accuracy: 0.9388 - lr: 7.4274e-05
```

```
Epoch 113/120
10/10 [==============================] - 2s 240ms/step - loss: 0.2101 - accuracy: 0.
9048 - val_loss: 0.1344 - val_accuracy: 0.9490 - lr: 7.0651e-05
Epoch 114/120
10/10 [==============================] - 3s 269ms/step - loss: 0.2044 - accuracy: 0.
8980 - val_loss: 0.1258 - val_accuracy: 0.9524 - lr: 6.7206e-05
Epoch 115/120
10/10 [==============================] - 3s 255ms/step - loss: 0.2132 - accuracy: 0.
9082 - val_loss: 0.1388 - val_accuracy: 0.9524 - lr: 6.3928e-05
Epoch 116/120
10/10 [==============================] - 2s 247ms/step - loss: 0.1674 - accuracy: 0.
9320 - val_loss: 0.1275 - val_accuracy: 0.9626 - lr: 6.0810e-05
Epoch 117/120
10/10 [==============================] - 3s 272ms/step - loss: 0.1661 - accuracy: 0.
9252 - val_loss: 0.1272 - val_accuracy: 0.9626 - lr: 5.7844e-05
Epoch 118/120
10/10 [==============================] - 2s 223ms/step - loss: 0.1901 - accuracy: 0.
9218 - val_loss: 0.1196 - val_accuracy: 0.9592 - lr: 5.5023e-05
Epoch 119/120
10/10 [==============================] - 2s 208ms/step - loss: 0.1865 - accuracy: 0.
9116 - val_loss: 0.1298 - val_accuracy: 0.9626 - lr: 5.2340e-05
Epoch 120/120
10/10 [==============================] - 2s 224ms/step - loss: 0.1981 - accuracy: 0.
9218 - val_loss: 0.1345 - val_accuracy: 0.9660 - lr: 4.9787e-05
```

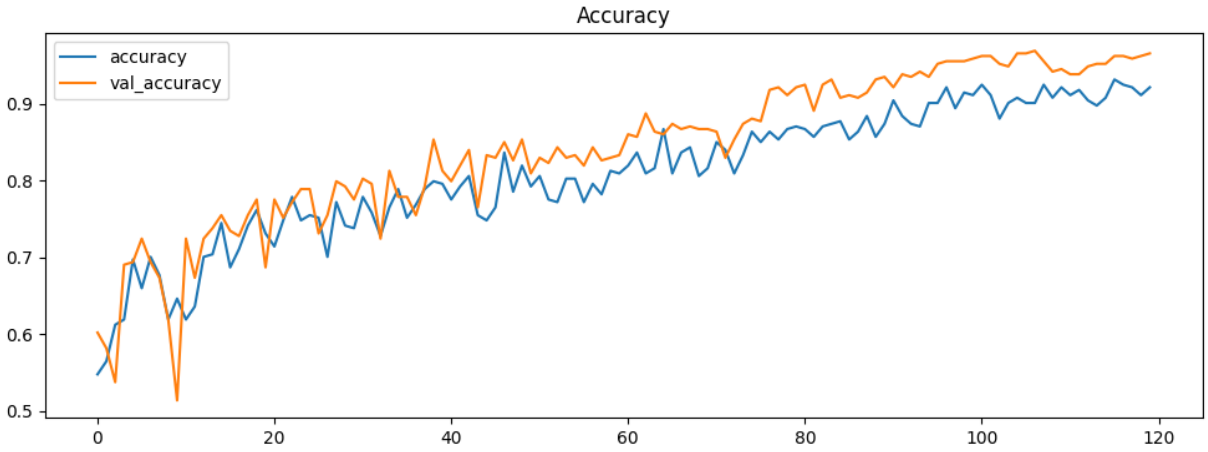
In [10]:
```python
import matplotlib.pyplot as plt

# Graphing Loss
plt.figure(figsize=(12,4))
plt.title("Loss")
plt.plot(history.history["loss"], label="loss")
plt.plot(history.history["val_loss"], label="val_loss")
plt.legend()
```

Out[10]:
```
<matplotlib.legend.Legend at 0x2883b298a50>
```



In [11]:
```python
# Graphing Accuracy
plt.figure(figsize=(12,4))
plt.title("Accuracy") # order of operations matter
plt.plot(history.history["accuracy"], label="accuracy")
plt.plot(history.history["val_accuracy"], label="val_accuracy")
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x2883d8fbbd0>



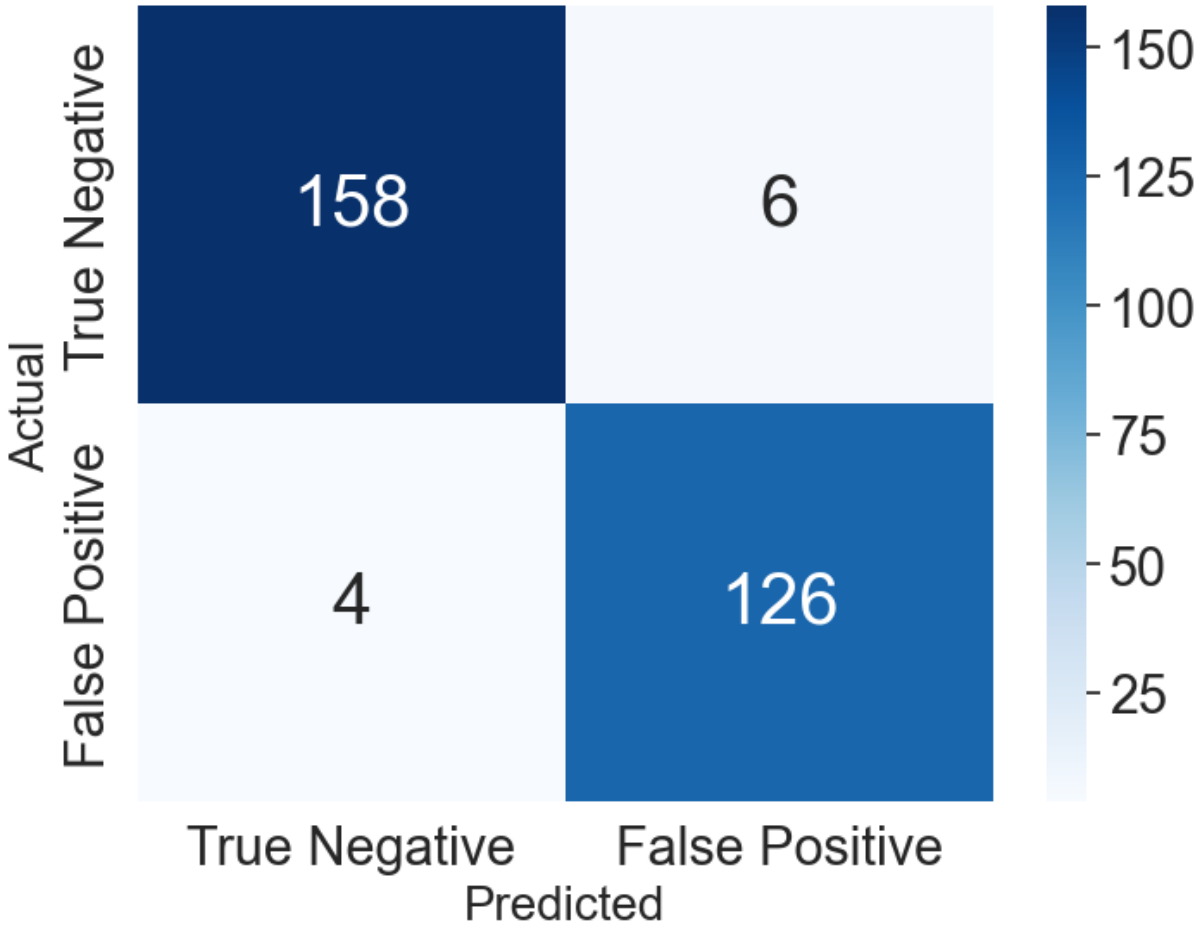
In [12]:
```
# New generator for test data
test_generator = input_pipeline.flow_from_directory(
        test_dir,
        target_size=(dim),
        class_mode='binary',
        shuffle=False)

# Predicting labels for all test images
y_pred = model.predict(test_generator)
y_pred[:10]
```

```
Found 294 images belonging to 2 classes.
10/10 [==============================] - 0s 29ms/step
```

Out[12]:
```
array([[9.1501331e-04],
       [1.5639281e-04],
       [8.1750371e-02],
       [6.8941707e-04],
       [2.3547868e-01],
       [4.9427040e-06],
       [3.3594231e-05],
       [1.8361539e-01],
       [2.0076232e-01],
       [7.4772290e-03]], dtype=float32)
```

In [13]:
```
from sklearn.metrics import confusion_matrix

y_true = test_generator.labels # Getting the true labels for the test data

y_pred = np.round(y_pred) # Converting the predicted labels to binary labels

confusion_matrix = confusion_matrix(y_true, y_pred) # Calculating the confusion mat

# Displaying the confusion matrix
import seaborn as sns
from seaborn import heatmap

plt.figure(figsize=(8,6))
sns.set(font_scale=2)
```

```
heatmap(confusion_matrix,
        annot=True,
        annot_kws={"size": 30},
        fmt='g',
        cmap='Blues',
        xticklabels=['True Negative', 'False Positive'],
        yticklabels=['True Negative', 'False Positive'])

plt.xlabel('Predicted', fontsize=20)
plt.ylabel('Actual', fontsize=20);
```



In [14]:
```
# 4
import random
#from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2, pr
```

In [15]:
```
# Hyperparameters
CFG = dict(
    seed = 55,
    batch_size = 16,
    img_size = (299,299),
    epochs = 20,
    patience = 10
)
```

In [16]:
```
# Augment train set only
train_data_generator = ImageDataGenerator(
        validation_split=0.15,
        rotation_range=15,
        width_shift_range=0.1,
        height_shift_range=0.1,
        preprocessing_function=preprocess_input,
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

val_data_generator = ImageDataGenerator(preprocessing_function=preprocess_input, va
test_data_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
```

In [17]:
```
# Connect generators to data in folders
train_generator = train_data_generator.flow_from_directory(train_dir, target_size=C
validation_generator = val_data_generator.flow_from_directory(train_dir, target_siz
test_generator = test_data_generator.flow_from_directory(test_dir, target_size=CFG[

# Number of samples and classes
nb_train_samples = train_generator.samples
nb_validation_samples = validation_generator.samples
nb_test_samples = test_generator.samples
classes = list(train_generator.class_indices.keys())
print('Classes:'+str(classes))
num_classes = len(classes)
```

```
Found 251 images belonging to 2 classes.
Found 43 images belonging to 2 classes.
Found 294 images belonging to 2 classes.
Classes:['apples', 'tomatoes']
```

In [18]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Pre-trained deep convolutional neural network
base_model = InceptionResNetV2(weights='imagenet', include_top=False, input_shape=(

# Add new layers
x = base_model.output
x = Flatten()(x)
x = Dense(100, activation='relu')(x)
predictions = Dense(num_classes, activation='softmax', kernel_initializer='random_u

# Build model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Define optimiser
optimizer = Adam()
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accur
```

In [19]:
```
# Save the best model
save_checkpoint = keras.callbacks.ModelCheckpoint(filepath='model.h5', monitor='val

# Early stopping
early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=CFG['pa
```

In [20]:
```
# Train model
history = model.fit(
        train_generator,
        steps_per_epoch=nb_train_samples // CFG['batch_size'],
        epochs=CFG['epochs'],
        callbacks=[save_checkpoint,early_stopping],
        validation_data=validation_generator,
        verbose=True,
        validation_steps=nb_validation_samples // CFG['batch_size'])
```

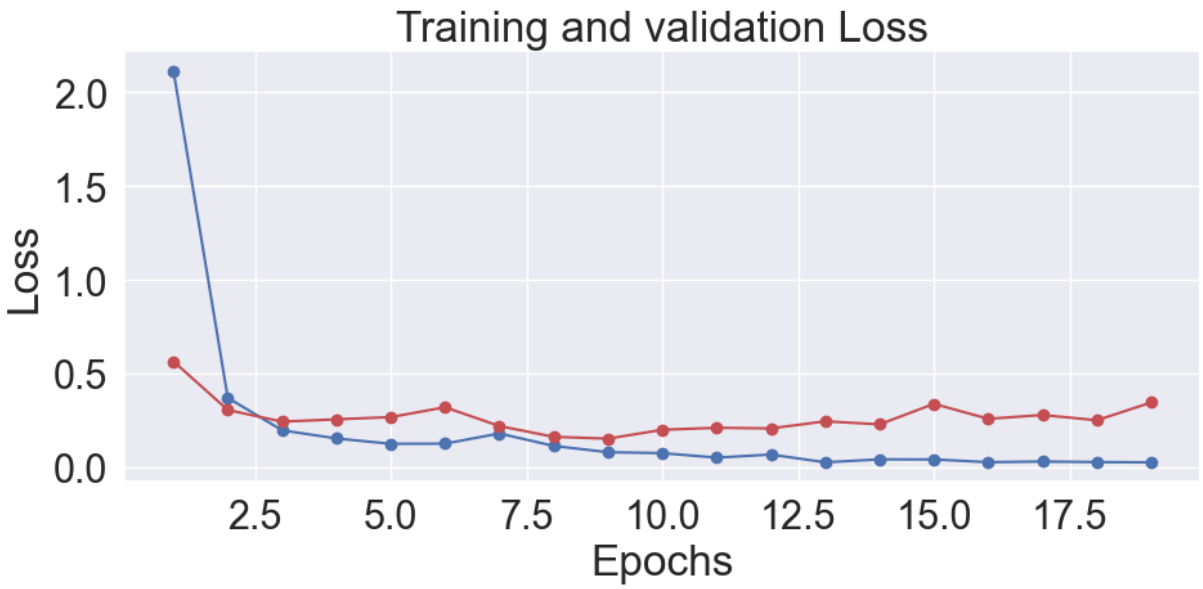
```
Epoch 1/20
15/15 [==============================] - ETA: 0s - loss: 2.1142 - accuracy: 0.6851
Epoch 1: val_loss improved from inf to 0.56465, saving model to model.h5
15/15 [==============================] - 34s 2s/step - loss: 2.1142 - accuracy: 0.68
51 - val_loss: 0.5647 - val_accuracy: 0.8125
Epoch 2/20
15/15 [==============================] - ETA: 0s - loss: 0.3700 - accuracy: 0.8298
Epoch 2: val_loss improved from 0.56465 to 0.30521, saving model to model.h5
15/15 [==============================] - 28s 2s/step - loss: 0.3700 - accuracy: 0.82
98 - val_loss: 0.3052 - val_accuracy: 0.9062
Epoch 3/20
15/15 [==============================] - ETA: 0s - loss: 0.1969 - accuracy: 0.9404
Epoch 3: val_loss improved from 0.30521 to 0.24296, saving model to model.h5
15/15 [==============================] - 26s 2s/step - loss: 0.1969 - accuracy: 0.94
04 - val_loss: 0.2430 - val_accuracy: 0.8750
Epoch 4/20
15/15 [==============================] - ETA: 0s - loss: 0.1532 - accuracy: 0.9574
Epoch 4: val_loss did not improve from 0.24296
15/15 [==============================] - 24s 2s/step - loss: 0.1532 - accuracy: 0.95
74 - val_loss: 0.2554 - val_accuracy: 0.9375
Epoch 5/20
15/15 [==============================] - ETA: 0s - loss: 0.1248 - accuracy: 0.9404
Epoch 5: val_loss did not improve from 0.24296
15/15 [==============================] - 24s 2s/step - loss: 0.1248 - accuracy: 0.94
04 - val_loss: 0.2672 - val_accuracy: 0.9375
Epoch 6/20
15/15 [==============================] - ETA: 0s - loss: 0.1256 - accuracy: 0.9489
Epoch 6: val_loss did not improve from 0.24296
15/15 [==============================] - 23s 2s/step - loss: 0.1256 - accuracy: 0.94
89 - val_loss: 0.3199 - val_accuracy: 0.9062
Epoch 7/20
15/15 [==============================] - ETA: 0s - loss: 0.1799 - accuracy: 0.9404
Epoch 7: val_loss improved from 0.24296 to 0.21959, saving model to model.h5
15/15 [==============================] - 24s 2s/step - loss: 0.1799 - accuracy: 0.94
04 - val_loss: 0.2196 - val_accuracy: 0.9375
Epoch 8/20
15/15 [==============================] - ETA: 0s - loss: 0.1134 - accuracy: 0.9574
Epoch 8: val_loss improved from 0.21959 to 0.16207, saving model to model.h5
15/15 [==============================] - 25s 2s/step - loss: 0.1134 - accuracy: 0.95
74 - val_loss: 0.1621 - val_accuracy: 0.9375
Epoch 9/20
15/15 [==============================] - ETA: 0s - loss: 0.0800 - accuracy: 0.9702
Epoch 9: val_loss improved from 0.16207 to 0.15216, saving model to model.h5
15/15 [==============================] - 24s 2s/step - loss: 0.0800 - accuracy: 0.97
02 - val_loss: 0.1522 - val_accuracy: 0.9375
Epoch 10/20
15/15 [==============================] - ETA: 0s - loss: 0.0749 - accuracy: 0.9702
Epoch 10: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0749 - accuracy: 0.97
02 - val_loss: 0.1998 - val_accuracy: 0.9375
Epoch 11/20
15/15 [==============================] - ETA: 0s - loss: 0.0512 - accuracy: 0.9872
Epoch 11: val_loss did not improve from 0.15216
15/15 [==============================] - 24s 2s/step - loss: 0.0512 - accuracy: 0.98
72 - val_loss: 0.2102 - val_accuracy: 0.9375
Epoch 12/20
```

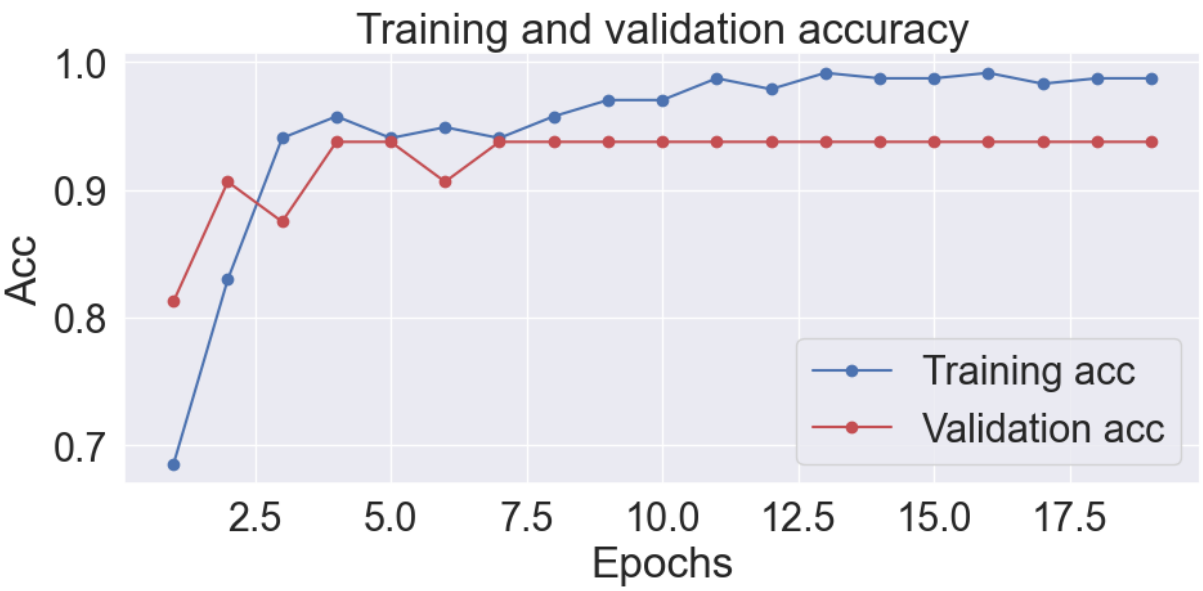
```
15/15 [==============================] - ETA: 0s - loss: 0.0675 - accuracy: 0.9787
Epoch 12: val_loss did not improve from 0.15216
15/15 [==============================] - 24s 2s/step - loss: 0.0675 - accuracy: 0.97
87 - val_loss: 0.2070 - val_accuracy: 0.9375
Epoch 13/20
15/15 [==============================] - ETA: 0s - loss: 0.0260 - accuracy: 0.9915
Epoch 13: val_loss did not improve from 0.15216
15/15 [==============================] - 25s 2s/step - loss: 0.0260 - accuracy: 0.99
15 - val_loss: 0.2449 - val_accuracy: 0.9375
Epoch 14/20
15/15 [==============================] - ETA: 0s - loss: 0.0419 - accuracy: 0.9872
Epoch 14: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0419 - accuracy: 0.98
72 - val_loss: 0.2286 - val_accuracy: 0.9375
Epoch 15/20
15/15 [==============================] - ETA: 0s - loss: 0.0415 - accuracy: 0.9872
Epoch 15: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0415 - accuracy: 0.98
72 - val_loss: 0.3379 - val_accuracy: 0.9375
Epoch 16/20
15/15 [==============================] - ETA: 0s - loss: 0.0268 - accuracy: 0.9915
Epoch 16: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0268 - accuracy: 0.99
15 - val_loss: 0.2581 - val_accuracy: 0.9375
Epoch 17/20
15/15 [==============================] - ETA: 0s - loss: 0.0304 - accuracy: 0.9830
Epoch 17: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0304 - accuracy: 0.98
30 - val_loss: 0.2785 - val_accuracy: 0.9375
Epoch 18/20
15/15 [==============================] - ETA: 0s - loss: 0.0273 - accuracy: 0.9872
Epoch 18: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0273 - accuracy: 0.98
72 - val_loss: 0.2505 - val_accuracy: 0.9375
Epoch 19/20
15/15 [==============================] - ETA: 0s - loss: 0.0259 - accuracy: 0.9872
Epoch 19: val_loss did not improve from 0.15216
15/15 [==============================] - 23s 2s/step - loss: 0.0259 - accuracy: 0.98
72 - val_loss: 0.3453 - val_accuracy: 0.9375
Epoch 19: early stopping
```

In [21]:

```python
# History
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

# Loss
epochs_x = range(1, len(loss_values) + 1)
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.plot(epochs_x, loss_values, 'b-o', label='Training loss')
plt.plot(epochs_x, val_loss_values, 'r-o', label='Validation loss')
plt.title('Training and validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
# Accuracy
plt.subplot(2,1,2)
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs_x, acc_values, 'b-o', label='Training acc')
plt.plot(epochs_x, val_acc_values, 'r-o', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.legend()
plt.tight_layout()
plt.show()
```





In [22]:
```
# Evaluate on validation dataset
score = model.evaluate(validation_generator, verbose=False)
print('Val loss:', score[0])
print('Val accuracy:', score[1])
```

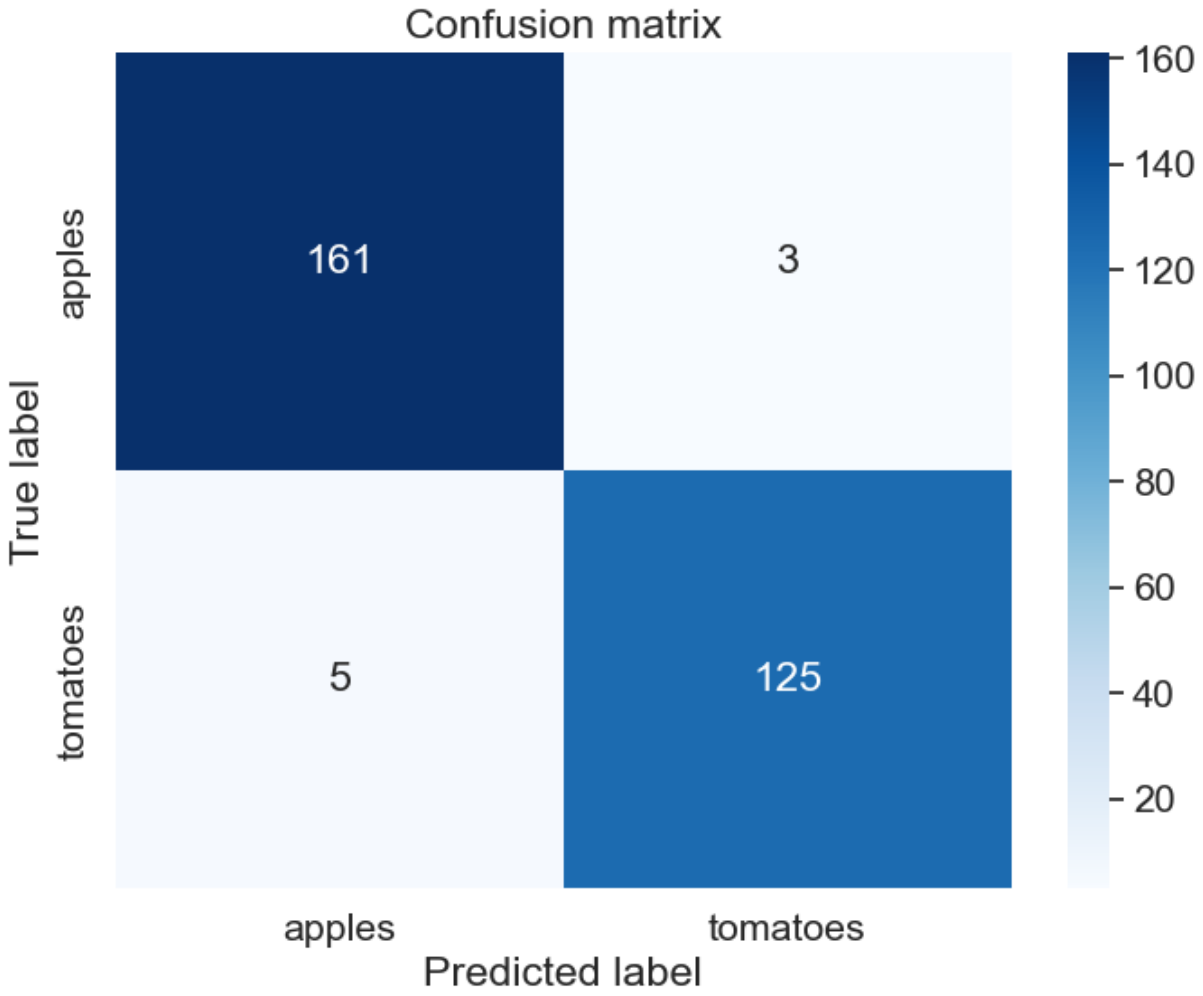
```
Val loss: 0.5205492377281189
Val accuracy: 0.8837209343910217
```

In [23]:
```python
# Evaluate on test dataset
score = model.evaluate(test_generator, verbose=False)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.0967201441526413
Test accuracy: 0.9727891087532043
```

In [27]:
```python
from sklearn.metrics import confusion_matrix # something breaks if we don't import

# Confusion matrix
y_pred = np.argmax(model.predict(test_generator), axis=1)
cm = confusion_matrix(test_generator.classes, y_pred)

# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cbar=True, cmap='Blues',xticklabels=classes, y
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion matrix')
plt.show()
```

```
19/19 [==============================] - 22s 1s/step
```



In [ ]:
```
# 5
# The Confusion Matrix suggests that transfer learning performs better than CNN. In
# the confusion matrix performed better than CNN every time. In other words, transf
# than CNN. This makes sense because transfer learning reuses is great with limited
# case, we had approximately 130/164 images of tomatoes/apples for the model to tra
# bigger, CNN could perform better. Regardless, both models performed very well wit
```

In [ ]:
```

```