# Kategoryzacja chorób jabłoni na podstawie wyglądu ich liści

Mateusz Garczyński Zuzanna Twardowska
Wiktor Wołek Dawid Wysocki Sonya Pobiedimska

Rysunek 1: Choroby widoczne na liściach jabłoni. Źródło: kaggle [**?**].

## 1 Progress

During the forth sprint we have concentrated on training existing models and trying new models that were similar to the already working ones. Our purpose was improving the accuracy and checking the influence of the single parameters on the accuracy. The main goal was not only to improve the accuracy but also to receive models working for 6 classes instead of 12.

## 2 Big decisions

At the beginning of this sprint we have made a big decision considering the future of our project - we have decided to drop the idea of leaf detection as the first part of the algorithm. After further research we have found out that this idea was simply outdated and is neither used nor significant in nowadays mcahine learning. Therefore any models including leaf detection by their contours were not continued.

## 3 Small problems

We were not able to check as many model variations as we wanted. Our input dataset is too big to allow us to quickly iterate between models. Therefore, tra-

ining new models was time consuming. Additional modifications (randomshear, randomrotate etc.) were not helpful regarding time problems. However, we have still managed to prepare and train enough models to make conclusions from the changing parameters.

# 4 Models

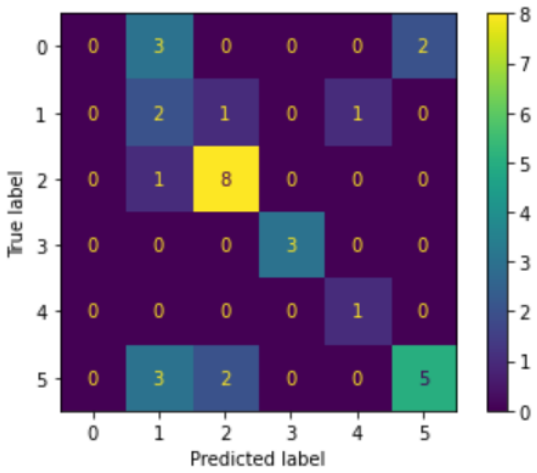
Model 4a was using VGG16 instead of inception v3. This idea seemed really promising as VGG16 is considered to be one the best vision model architectures. Instead of having a large number of hyper-parameter it is focused on having convolution layers of 3x3 filter with a stride 1 and always uses the same padding and maxpool layer of 2x2 filter of stride 2. Nevertheless, it was a complete failure. The accuracy was lower than ever, therefore we decided to stop training that model.

```
Epoch 1/5
200/200 [==============================] - 676s 3s/step - loss: 2.0128 - accuracy: 0.3181 - val_loss: 1.8828 - val_accuracy: 0.3188
Epoch 2/5
200/200 [==============================] - ETA: 0s - loss: 2.4570 - accuracy: 0.3247
```

Model 4b was using inception v3 - the same as our best old model. We decided to increase the number of epochs from 1 to 5. Input data randomization was moved outside of the model. The accuracy results were were better than the old model. The accuracy was around 0.64.

```
Epoch 1/5
2022-05-25 20:49:46.894796: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8400
2022-05-25 20:49:47.401227: I tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child process: No such file or directory
  1/466 [..............................] - ETA: 1:05:40 - loss: 2.1782 - accuracy: 0.1250
2022-05-25 20:49:48.569165: W tensorflow/core/common_runtime/bfc_allocator.cc:360] Garbage collection: deallocate free memory regions (i.e., allocatio
ns) so that we can re-allocate a larger region to avoid OOM due to memory fragmentation. If you see this message frequently, you are running near the
threshold of the available device memory and re-allocation may incur great performance overhead. You may try smaller batch sizes to observe the perfor
mance impact. Set TF_ENABLE_GPU_GARBAGE_COLLECTION=false if you'd like to disable this feature.
466/466 [==============================] - 1728s 4s/step - loss: 1.3131 - accuracy: 0.5475 - val_loss: 1.1740 - val_accuracy: 0.5888
Epoch 2/5
466/466 [==============================] - 1701s 4s/step - loss: 1.1423 - accuracy: 0.6200 - val_loss: 1.0373 - val_accuracy: 0.6559
Epoch 3/5
466/466 [==============================] - 1689s 4s/step - loss: 1.1168 - accuracy: 0.6329 - val_loss: 1.3044 - val_accuracy: 0.5368
Epoch 4/5
466/466 [==============================] - 1701s 4s/step - loss: 1.0824 - accuracy: 0.6466 - val_loss: 1.0392 - val_accuracy: 0.6498
Epoch 5/5
466/466 [==============================] - 1656s 4s/step - loss: 1.0834 - accuracy: 0.6460 - val_loss: 1.0585 - val_accuracy: 0.6428
```

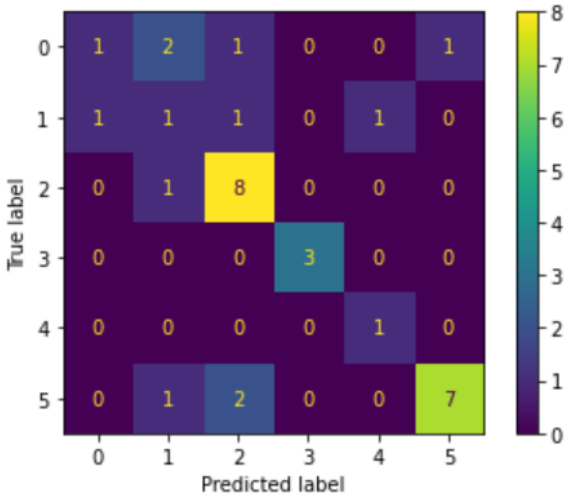
The models confusion matrix:




In model 4c we decided to use rmsprop as an optimizer. The results were good with evaluated accuracy equal to 0.6608.

```
Evaluate
117/117 [==============================] - 328s 3s/step - loss: 1.0259 - accuracy: 0.6608
{'loss': 1.0258911848068237, 'accuracy': 0.6607621908187866}
```

The models confusion matrix:




Then we have tried running the old model for 5 epochs with similar results. The evaluated accuracy was equal to 0.6669.

```
Epoch 1/5
466/466 [==============================] - 2654s 6s/step - loss: 1.3300 - accuracy: 0.5694 - val_loss: 1.2444 - val_accuracy: 0.5312
Epoch 2/5
466/466 [==============================] - 1618s 3s/step - loss: 1.1447 - accuracy: 0.6299 - val_loss: 1.1442 - val_accuracy: 0.5312
Epoch 3/5
466/466 [==============================] - 1613s 3s/step - loss: 1.1033 - accuracy: 0.6385 - val_loss: 1.1285 - val_accuracy: 0.5625
Epoch 4/5
466/466 [==============================] - 1612s 3s/step - loss: 1.0679 - accuracy: 0.6499 - val_loss: 1.0350 - val_accuracy: 0.5312
Epoch 5/5
466/466 [==============================] - 1615s 3s/step - loss: 1.0509 - accuracy: 0.6573 - val_loss: 1.0763 - val_accuracy: 0.6250
INFO:tensorflow:Assets written to: model/m4/assets
```

```
print("Evaluate")

result = model4.evaluate(validation_generator)
dict(zip(model4.metrics_names, result))
```

```
Evaluate
117/117 [==============================] - 397s 3s/step - loss: 1.0149 - accuracy: 0.6669
{'loss': 1.0149343013763428, 'accuracy': 0.6669350266456604}
```

The next model had additional dropout layer. After running it for one epoch the results were not satisfying so we moved to the next model.

In the last model we have tried changing the optimizer from Adam to NAdam and running it for 1 epoch.

```
466/466 [==============================] - 1692s 4s/step - loss: 1.3812 - accuracy: 0.5421 - val_loss: 1.1540 - val_accuracy: 0.5938
INFO:tensorflow:Assets written to: model/m5/assets
```

# 5 Conclusion

We have improved our accuracy during that sprint. Whats more important we were finally able to create models that used only 6 classes. Additionally, due to our further research, we were able to drop the idea of detecting leaf contours what left us with the consistent idea for the solution.