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 Introduction

The aim of this project was creating a tool that would be helpful during the apple tree's disease detection based on the photos of their leaves. Despite the development in the computer science and machine learning, the detection is still done manually and is very time consuming. Due to the significant impact of the diseases on the quantity of yield, early disease detection is crucial in increasing the production from the orchards.

In order to address this problem we have created a machine learning model that can be used in the detection and categorization of apple tree diseases. The project idea and dataset were provided by Kaggle as one of their challenges.

One of the the biggest challenges considering detecting diseases based on the leaf appearance was the variety of their symptoms. The visual differences between the representatives of the same group are caused mainly by the trees' age differences, different growing conditions and the age differences of the lesions.

2 Technologies and environments

The solution was implemented in python3 that is widely known as one of the most useful and popular tools in machine learning. The most crucial characteristic of that language for our project was the amount of the available libraries designed for the machine learning purposes.

The project was based on the Tensorflow library and its interface library, Keras.

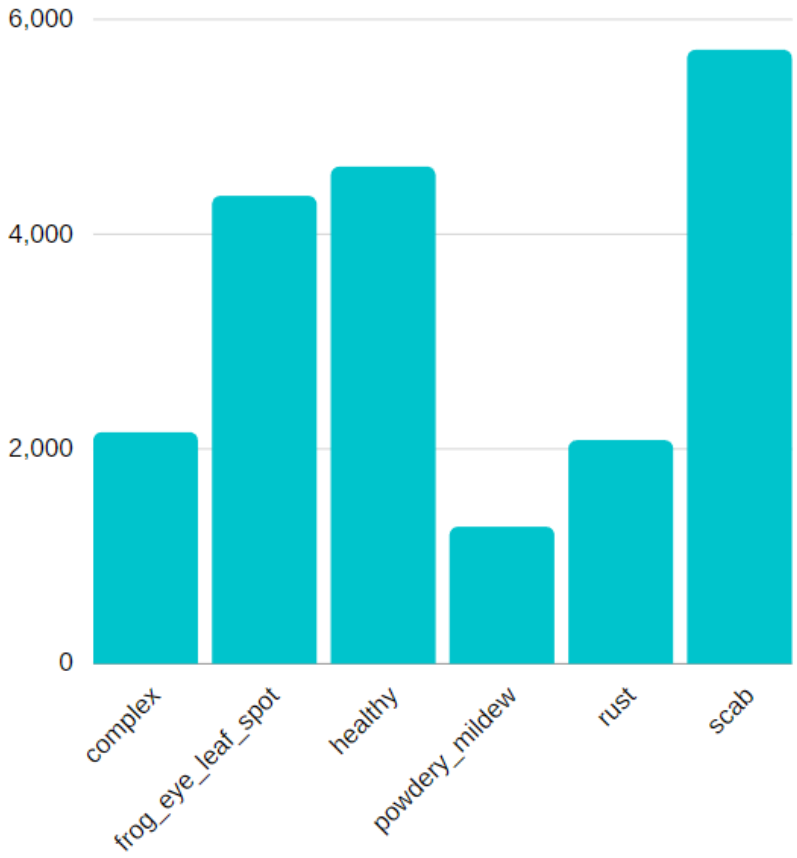
In order to automate as many parts of the development process as possible, in the beginning of the project we have focused on searching for the virtual environment/storage that would allow us to become less reliable on our CPUs. However, we have not found anything that was able to store and manage the dataset that big, therefore we have finally decided to use our PCs during the whole process.

3 Dataset

The dataset consisted of 18 600 photos presenting apple leaves. Photos were of different size and were saved in JPEG format. The photos in the dataset were categorized into 6 groups:

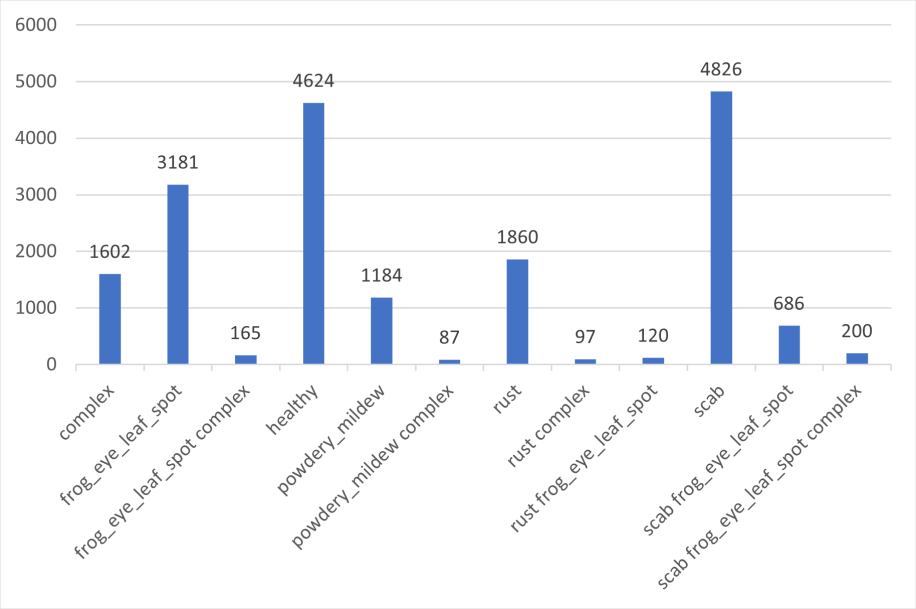
- scub,
- rust,
- powdery mildew,
- frog eye leaf spot,
- healthy,
- complex - too many disease were present to detect single ones.

To visualize those groups better we have created a chart that represented the amount of photos in each group.



Rysunek 2: Classes quantity chart for the original groups.

We have quickly realized that some of the photos were members of more than one group, therefore, we decided to add group combinations to the chart as well.



Rysunek 3: Classes quantity chart for all groups' permutations.

The analysis of the data distribution between the groups was sufficient to proceed with the implementation without removing any items from our dataset. From the second chart we have noticed that all of the newly created groups were much smaller than the original ones.

While analysing the photos we came to the realization that most of them contain many leaves in the background that do not necessarily have the same disease as the main leaf. Even though at the beginning it seemed to become a major problem, at the end created model was able do detect the main leaf correctly without any additional preprocessing.

4 Data preprocessing

The photos from the dataset had different sizes, so we decided to resize them before model training. We have chosen the size of 256x256 pixels, because their quality was still sufficient to detect diseases and additionally we were able to train our models much faster.

Moreover, during training we randomly rotated, flipped and zoomed photos to increase the size of our traing dataset and avoid overfitting the model.

5 First models

We have iterated over various models before settling on the one with the highest accuracy. We have started of with a small neural network that consisted of three

layers. It came as no surprise that the accuracy was close to random, therefore we have continued with more advanced models.

We have decided to try using well-proven and well-tested neural networks. We have tried using a neural net designed to solve common beginners problem of distinguishing cats from dogs. The problem of that solution was binary since there were only two classes, but after modifying last layers of the neural net and switching to categorical cross entropy as a loss function we have managed to obtain higher accuracy compared to the starting model.

Then we have experimented with the idea of using state of the art algorithms, designed to classify thousands of objects in worldwide competitions, VGG16, VGG19 and InceptionV3 among them. As they were designed to classify thousands of objects, we had to modify last layers to fit our problem: classifying between 6 labels. We have started of with the InceptionV3 which has provided very high accuracy from the start. As model was reaching as high as 71% during training, we were quite optimistic. However, it failed to reach our expectations - evaluation showed only 16% of accuracy. With randomizing data in the input layers we managed to boost the accuracy to 59 %, which still was not the result we were counting on.

VGG16 and VGG19 results were not even satisfactory in comparison to the InceptionV3. After some tries and hours of training we gave up the idea of using them in order to focus on modifying our best solution so far.

We have experimented with different optimizers: Adam, NAdam and RMSProp. Our starting optimizer Adam was able to reach higher accuracy in less time, so we ended up using that in the end. Modifying learning rate also didn't help with training. Making learning rates higher caused instability - we have experienced sudden rises and drop-offs in the accuracy during training. Lower learning rates slowed down training significantly, which was already much below the average.

We have also experimented with different amount of classes. We had 6 base classes with 12 permutations within the dataset. Classifying between 12 permutations showed marginally higher accuracy, therefore we decided to continue classifying between 12 permutations in our final set of the models.

6 Model structure

The final model consisted of three main layers: the input layer, the InceptionV3 layer and the output layer.

The input layer preprocessed raw images in sRGB. We have changed the size of images to be 256 by 256. We have also randomized data by flipping images horizontally, rotating and zooming. It helped drastically with accuracy, but it also made our model non deterministic.

InceptionV3 was used as our base model, which did most of the detection and recognition. It has more than 21 millions of parameters, which is still lower than VGG counterparts. We have used weights from the Imagenet competition and froze them during training. We cut off the last layer, since our problem required different amount of classes.

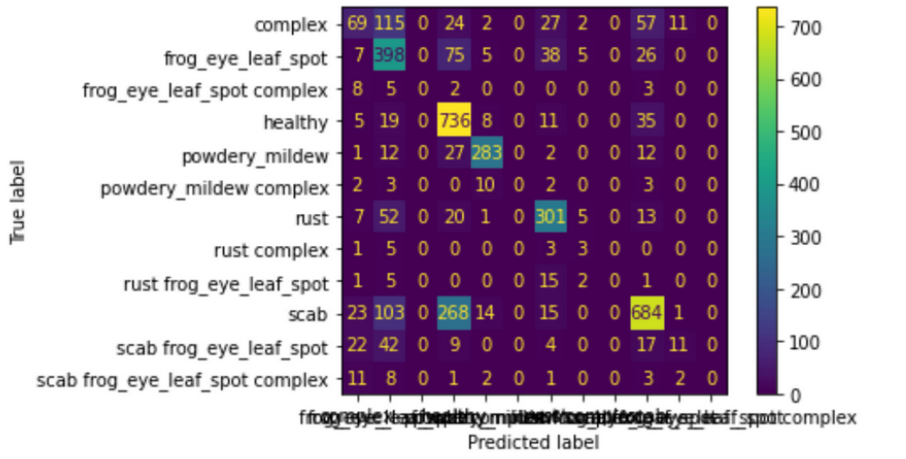
Last layer consisted of global average pooling 2D layer which connected the InceptionV3 model with our last layer which was a Dense layer with 12 nodes corresponding to our permuted classes. We have used softmax as our activation layer and trained with the uniform kernel initializer.

7 Results

Our final model's accuracy was equal 68% on our validation set that consisted of 3726 images selected randomly from our dataset. We have also evaluated it on our training dataset and obtained similar results. That meant that overfitting did not occur in our model's case, which was great news.

By analysing the confusion matrix we were able to determine how effective our model is. In general the model was classifying the photos correctly considering its accuracy.

In the confusion matrix it can be seen that one of the most frequent model's mistake was classifying 'scab' as 'healthy'. It is worth mentioning that in most cases scab's visual symptoms are not very visible and hard to notice even for the human eye, therefore, we were not surprised by that result.



Rysunek 4: Final model's confusion matrix, tested on the validation set

In order to present what exactly our model's was classifying we decided to show some of the test cases.

Rysunek 5: Picture of a scab-infected leaf, our model correctly labeled it as scab.

Rysunek 6: Picture of a healthy leaf, our model correctly labeled it as healthy.

Rysunek 7: Picture of a leaf with many different diseases, our model correctly labeled it as complex.

Rysunek 8: Picture of a leaf infected with powdery mildew, our model correctly labeled it as powdery mildew.

8 Conclusion

The final accuracy of 68% does not sound impressive, but given that the highest scores from the Kaggle's competition were reaching only around 86%, our accuracy seems to be more than satisfactory as for the first entry into the machine learning. especially given the huge size of the training data, which slowed down creative process of fine tuning the models.

Most of the problems we were facing were the results of the size of the dataset and the amount of the time that training the models took. The dataset

had more that 16GB and training it on modern PCs took at least 2 hours for the fastest models. As the result we were not able to experiment with as many different parameters and models.

The project itself gave us first hand experience with different aspects of the machine learning such as data analysis, building models, training on the big data and has showed us the importance of preprocessing.

We have also had to tackle popular problems such as overfitting and under-representation in the datasets. Moreover, we have read many scientific articles about machine learning in general and about different approaches and tools used in the discipline, which improved our knowledge about the field significantly.