**Report project 3: “De maakindustrie”**

Nils Moerman

23/06/2023

**Image classification: Recognize defects in apples.**

Introduction:

* **Project**
* Assignment
* Goals
* **EDA**
* Manual data check
* Augmentation
* **CNN model**
* Creating the model
* Training the model
* Testing the model
* Comparing with Resnet50
* **AQL and chatbot**
* AQL function
* Chatbot
* **Conclusion**
* What have I learned
* Overall thoughts of the course

**Project**

Assignment

For this project we had to create a convolutional neural network (CNN) that can detect apples and the flaws they have by looking at apple images. The model needs to be able to make a difference between good,blotch,rot and scab apples. For this the teachers gave us a dataset of apple images to work with. We also have to implement an AQL inspection using the model we created and gather the information using a chatbot.

Goals

My goals for this project:

* Gather the data.
* Make decision and don’t try to fix every image (be picky about the usage).
* Transform the data.
* Create a model and fully understand how it works and why.
* Keep everything structured and in order.

**EDA**

Manual data check

The first thing I do after receiving the dataset is looking at the images by hand. I quickly found out there are a lot of unusable images. There are animated apples, an painting of an apple, a lot of apples with watermarks, duplicates, etc.. I decide to just delete all of them since I’m going to create more of them in the augmentation.

Augmentation

*For this please check the EDA.ipynb notebook*

Before I start with the augmentation I want to gather my images to rename and resize them. I then save them in a different folder so the original dataset isn’t affected. After that I check the dataset manually again and notice some images that had rectangle shapes now look weird and stretched. I decide to just remove those as well.

During the training and testing phase I realised I don’t have a lot of normal apples in my dataset compared to the rest, so I came back to this notebook to add more normal apples (gathered from the internet) and apply the same steps (as shown in the EDA.ipynb notebook).

*Next please check the augment.py file*

Next I want to create more images, since my model is only going to see the number values of the images I could mirror and rotate them and the model would see it as a completely different image. So after loading in the images from the new location, I start by flipping all the images to mirror them. By doing this I double my data set already. Then I decide to rotate the images 3 times and save every rotation as a separate image. To keep the background roughly the same I decided I would rotate the images in steps of 90 degrees.

Example for 1 image:

* Flip it (creates 2 in total)
* Rotate both images 90 degrees (creates 4 in total)
* Rotate both images 180 degrees (creates 6 total)
* Rotate both images 270 degrees (creates 8 total)

So this means my dataset got 8 times larger!

**CNN model**

Creating the model

*For this please check the apple\_classifyer.py file*

So after gathering en transforming all my data I went on and started with my CNN.

I started with the layout and decided I wanted 2 convolutional filters and 1 pooling filter. After creating that and looking at some examples online I realised that would be a bit small even for a basic model. So I decided to follow those steps a second time.

After that I flattened the outcome and added the fully connected linear layers, I decided 3 would be fine. Since I want my model to predict 1 of 4 outputs (normal, blotch, rot or scab) I put my outputs as 4. After that I created a fit function for training and validation. For my loss function I uses CrossEntropyLoss.

I created this class inside the train\_test\_cnn.ipynb notebook to start with. It’s a bit more convenient to train and test while still tweaking some stuff. After a while I moved it to it’s own file and imported it to the notebook.

Training the model

*For this please check the train\_test\_cnn.ipynb notebook*

So I started by creating a class to create a dataset with. I then call my data put it in the DatasetApples class and transform every image to a tensor (this is what the model works with). The class uses a function that automatically creates a label for each folder inside the path I give it. As you can see in the notebook I resize my images again. This is because I encountered problems with my GPU memory and to fix that I made the images smaller, this also caused my model to train much faster.

I then split the dataset 60% train, 20% test, 20% validate and put these inside the data loader and started to train! After experimenting with different learning rates and adjusting the filters a bit I ended with what you see now with a max accuracy of 86%.

Testing the model

To test the model I created a function inside my class that can predict the label of a single image. As you can see it’s not perfect but I decided it would be enough.

Labels:

* Blotch = 0
* Normal = 1
* Rot = 2
* Scab = 3

I also added a confusion matrix function and made it so it’s shown after a full training cycle.

After testing I save the model using pickle.

Comparing model with Resnet50

*For this please use the resnet\_tryout.ipynb notebook*

I really wanted something to compare my model to. So I decided to use the predefined and trained model called Resnet50.

So to start I created the dataset, I had to use a different size for my images since Resnet50 is used to training on 224 by 224 pixels. It also needed mean and std like shown in the notebook. After that I Created the data loaders but decided to make them smaller since the images were a lot bigger and it would take a very long time to train an cost a lot of memory.

Then I created a class for the Resnet model. I want to train the model on my dataset, so for this I used the same functions (with a few minor adjustments) as I used with my own model. After that I save the model and load in my own model to compare the two next to each other. The funny thing is, even though my Resnet model performed with a much higher accuracy of more than 90%, my own model actually performed better with the normal images I gave them!

I decided to use my own model for the AQL and chatbot part of this project.

**AQL and Chatbot**

AQL function

*For this please check the apple\_classifyer.py file*

To be able to do an AQL inspection I created a function inside the AppleClassifyer class. It takes a sample and uses the model to predict the outcome of every apple, it counts every normal apple as good and all the others as bad. At the end it returns the sample label and the amount of good and bad apples in the given sample (as predicted by the model).

*Next please check the aql\_and\_chatbot.ipynb notebook*

Next I created a folder with 20 apples in it (14 good, 2, blotch, 1 rot and 3 scab) specifically to test my AQL function. Since the assignment says the lot size is between 250 and 500 apples, and the inspection level is “I” the inspection code letter is F and the sample size has to be 20.

I then used my function and put the results in variables for later use.

Chatbot

*Please continue to use the aql\_and\_chatbot.ipynb notebook*

Now that I have all the information I need I can implement the chatbot. For this I use the SentenceTransformer(‘all-MiniLM-L12-v2’). Next I use the variables I created earlier to help give the best possible answers.

**Conclusion**

What have I learned

I learned:

* How to read and transform images.
* To not overthink to much and only use the usable data.
* How to structure and build a CNN.
* How to read and perform an AQL inspection.
* How to implement a simple chatbot.

Overall thoughts of the course

I’ve had the best time these past 5 months and I’m sorry it has to come to an end. I’ve learned so much in such little time it’s almost impossible to wrap my head around it.

Of course there were difficult moments, but the teachers and classmates were always there to support those moments. This course gave me in particular a BIG confidence boost and looking back on what I have achieved I’m extremely proud of myself!

Thank you Frank, Jeroen, Ruud, Gwen and Laura for everything!