Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Fachbereich 4: INFORMATIK, KOMMUNIKATION UND WIRTSCHAFT

# Course project

## Image classification using ConvNet

| | |
|---|---|
| Studiengang: | **Angewandte Informatik** |
| Lehrveranstaltung: | **Advanced Topics: Machine Learning** |

| | |
|---|---|
| Autor: | Chris Rebbelin, s0548921 |
| Dozent: | Herr Prof. Dr. Mykyta Kovalenko |
| Datum: | Berlin, 1. Juli 2018 |

# Chapter 1

# Task

A problem of classification or regression had to be selected, that could be solved with means of Machine learning. For the solution there were two approaches possible; the classical ML approach and the deep learning approach. For the chosen problem, the right approach had to be used and implemented. The topology and structure of the implementation had to be explained. After creating a model, it had to be evaluated and tested by at least two different model evaluation metrics. Finally a conclusion had to be drawn about the model in relation to the problem. The task description was available on Moodle.

# Chapter 2

# Problem

I chose to do the deep learning approach. I used the website kaggle.com to look for a dataset containing a larger amount of images to be able to classify them with a convolutional neural network (CNN). The dataset i selected was "Flowers Recognition" and contains in total 4242 images of five different types of flowers.[1]. The images were scraped from different image sources online like Google Image, flickr etc. Because of that, they are in different sizes and resolutions and need to preprocessed before feeding them into a CNN. Because of the large amount of available data, the deep learning approach was ideal for this type of problem.

[1] https://www.kaggle.com/alxmamaev/flowers-recognition/home

# Chapter 3

# Implementation

The code structure was taken from the one on moodle[1] and consits of two python code files. (inside the directory `code/`)

`code/my_cnn_train.py` includes the training of the convnet and basic loading function for reading and preparing the dataset. The created model is saved at the end as `mymodel.dat` and can be used to directly classify images without having to train the model first.

In `code/my_cnn_test.py`, the model that was created and saved in the previous file, is read by `keras` and used to classify some images. The results are shown with `openCV`, where every given image is printed with the most likely result label. (Again, this was mostly taken from the sample code on moodle)

The dataset is inside the directory `flowers`. It contains five subdirectories, one for each label / flower type. Each of those directories contains about 800 images in `JPG` format. The exact amount can be found in table 3.1.

| type | number of images |
|---|---|
| daisy | 765 |
| dandelion | 1095 |
| rose | 784 |
| sunflower | 734 |
| tulip | 984 |

Table 3.1: **distribution of flower types**

Detailed explanation of code is found in the source files as comments.
Everything was implemented and tested on Windows 10 (64 bit) with Visual Code.

## 3.1 training

```
import cv2
import numpy as np
from keras import backend as K
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Activation, Dense, Dropout, Flatten
```

[1] https://moodle.htw-berlin.de/mod/resource/view.php?id=417224

```
from keras.layers.normalization import BatchNormalization
from keras.models import Sequential
```

## 3.2 testing

```
import cv2
import numpy as np
from keras.models import load_model
from keras.preprocessing.image import img_to_array

MODEL = load_model("mymodel.dat")
CLASSES = os.listdir("flowers")
TEST_DIR = "flowers_test"
```

# Chapter 4

# Evaluation

To evaluate the created model, multiple metrics were used:

- accuracy
- Precision / recall
- AUC-ROC
- f1 score

## 4.1 accuracy

Accuracy is the simplest metric because its just right divied by wrong. I achieved a maximum accuracy of around XX percent.

## 4.2 precision/recall

precision and recall can be calculated as formeln hier The precision / recall curve can be seen in figure 4.1.

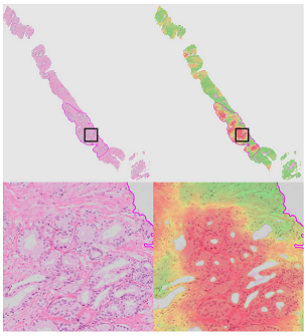

Figure 4.1: precision / recall curve

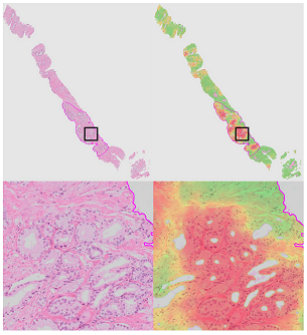

Figure 4.2: AUC_ROC

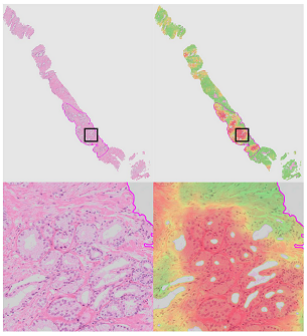

Figure 4.3: F1 score

## 4.3 AUC-ROC

AUC-ROC is an abbreviation for blabla. The AUC_ROC is shown in figure 4.2.

## 4.4 f1 score

The F1 score is displayed in figure 4.3.

# Chapter 5

# Conclusion

Using an image dataset of over 4000 images of flowers, a conv net was build and used to classify new images of flowers. The model achieved an accuracy of XX percent, as described in the last chapter.

Does the selected model fit the result? Yes because conv net is good for images, esp for many. Influeing paremters: Using dropout to make it indepent more and not prone to overfitting. Epochs: traing longer takes more time, wird es besser wenn man viel mehr nimmt?