BSPro-Classification of flowers with Convolutional Neural Networks Summary

Sunday 21st June, 2020 - 14:38

Licina Esada
University of Luxembourg
Email: esada.licina.001@student.uni.lu

Kieffer Emmanuel
University of Luxembourg
Email: emmanuel.kieffer@uni.lu

***Abstract*—This summary of the final report made by the student Esada Licina and by the support of hers tutor Emmanuel Kieffer. Here will explain the scientific and technical part of the project and for more information, read the final report once.**

1. Introduction

This is a project of a flower world where all flowers are mixed up. However, the task is to recognize and categorize all plants with the help of a Neural Network model.

To understand better the theme of the project, it will be firstly explained the meaning of the title. Classification is a structured universe of the given data. Here, for example, is a data set of flowers that will be split into 5 classes (tulips, sunflowers, roses, dandelion, daisy). This data will be used by the neural network. One of them is called the convolutional neural network (CNN), which describes a process to recognize and categorize samples. It is able to find the logic behind a picture and to extract the knowledge. However, it is also good to know that this project is a part of deep learning and machine learning.

The main objectives of this project are to find an effective way to classify data sets and to look at how CNN's works. There will be also a practical work where the code will be written in PyCharm with help of the FastAi library.

2. Project description

2.0.1. Convolutional neural networks. Convolutional neural networks (CNN) are a space invariant artificial neural network (SIANN), which is an extension of Neural Network and that have the task to analyze visual imagery. Every picture can be represented as a matrix of pixel values with features like color, figure, etc..., so the CNN works with them to recognize the images. It checks the whole pictures by using filters, and by doing the feature learning. When it works with more filter layers and creates more features maps then it will output better results. This is also called a deeper CNN because it is more abstract and that's the reason why the project is a part of deep learning.

3. Prerequisites

3.1. Scientific

The most significant task that was needed to do before, is to make a sheet of the project plan. It helps to be organized. The first task was to make researches about the topic and to find a data set of flowers in Kaggle.

3.2. Technical

After having enough knowledge, it is time to program. At first, it will be split the data into train and test folders and then it will build a model with them.

4. Scientific Deliverable: The brain of convolution neural networks

4.1. Requirements

Functional requirements:

1) Data processing of flowers.

Non-Functional requirements:

1) Efficiency of the CNN.

4.2. Design & Production

The picture below shows the CNN structure.

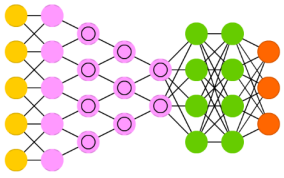

Figure 1. Convolutional Neural Network model

1 **Convolutional Layer (Feature Learning Part):** After putting the image in the model (yellow), the scalar product will be used on the matrix from the image with a filter matrix. Then an activation function called ReLU will be used after each convolutional layer. One more information is that an image consists of pixel values that form a matrix.
2 **Pooling Layer (area with circles: Feature Learning Part):** This ensures that the matrix will be refined.
3 **Fully Connected Layer (Classification Part):** After this run, the result will be predicted.

The model will use a ResNet34 architecture, which is explained below.

Layer name	ResNet34
conv1	$7 \times 7, 64$, stride 2
pool1	3×3, max pool stride 2
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$
fc1	$9 \times 1, 512$, stride 1
pool_time	$1 \times N$, avg pool stride 1

Figure 2. ResNet34 architecture

In the first step, we see that we have a convolutional layer with 64 kernels (layer depth), which has the size 7x7 and stride 2. Then do the dot product with the filter/kernel and the input matrix, which produced the feature map. Then after using the max pooling, the matrix will be simplifying to 3x3, and the depth rests the same. Then it continues with more convolutional operation and every time the filter/kernel size still the same but the number of filters (layer depth) double until the end. The multiplication number on the right side of the matrix says how many layers it will have of this type. At first, we have 2 convolutional operation x 3 so we get 6 layers and so on. In the end, it needs to have 34 layers in general because it uses ResNet34. After it uses an activation function ReLU and in the end it uses an average pooling and the matrix will be simplifying again.

4.3. Assessment

The requirements were successfully satisfied.

5. Technical Deliverable: Classification process in PyCharm

5.1. Requirements

Functional requirements:

1) Build of a model.
2) Real-time classification.

Non-Functional requirements:

1) Usability of PyCharm.
2) Performance.

5.2. Design & Production

In the picture below, it is seen the most significant steps that are used to create the code.

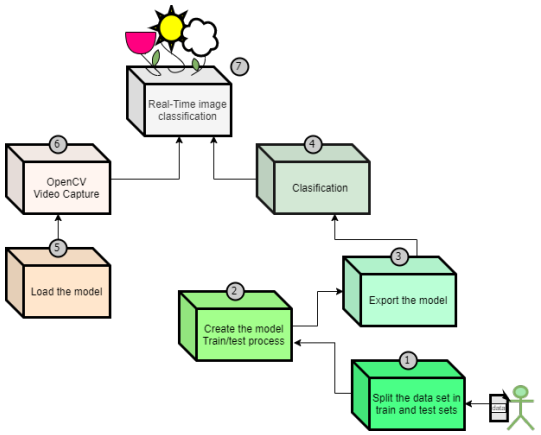

Figure 3. Classification Progress diagram

1 Firstly, the data set will be divided into two different folders. One for training (80% of the data set) and the other for testing (20% of the data set). Here the function np.split () will be used of the numpy library.
2 Then the model will be build from the detest, using the pre-trained ResNet34 architecture. Here the function cnn _learner () from FastAi is useful in this area.
3 The model will be exported so that it can be used. Here you can use learn.export () from FastAi.
4 Then the model will be tested with an unknown flower images to determine whether it is working properly. The function learn.predict () makes this possible.
5 Then the second part of the code continues. The model, which was exported will be loaded, so that it can be used.
6 Now a video recording will be made with the help of the OpenCV library. The video will be converted into images and classified by the model.
7 In the picture below, can be seen the result, where the prediction of the model appears above the pink window.

Figure 4. Video-Capture output

5.3. Assessment

The requirements were successfully satisfied, the pictures above show the results.

6. Acknowledge and conclusion

Finally the goal is reached, it is created a well working classification model, which can be expanded to connect it with android. In the near future I will continue to work in this domain. I would to thanks a lot my project tutor Emmanuel Kieffer, the BICS team, my family and my friend for their support during this semester.