BSPro-Classification of flowers with Convolutional Neural Networks

Sunday 21st June, 2020 - 14:43

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

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

Abstract

This second-semester project of BICS, made by the Student Esada Licina and with the support of her tutor Emmanuel Kieffer, is about recognition of flowers with convolutional neural networks. Consequently, it will be presented the incredible world of machine learning and deep learning, that are described during the scientific sections. There exists also a technical part, where would be studying how the code is built-in PyCharm.

1. Introduction

A man named Thomas Mann said one day " You ask what is the use of classification, arrangement, systemization? I answer you: order and simplification are the first steps toward the mastery of a subject. The actual enemy is the unknown ". Therefore, the flower classification project is a perfect example and consists of using convolutionary networks. In this way, the flowers will be analyzed and the unknown will be fought. Before starting, it is needed to understand better the project, it is important to know what the title means. At first, classification is a structured universe of the given data. Here it is a data set of flowers that will be split into 5 classes (tulips, sunflowers, roses, dandelions, daisies). 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 extracts 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 work. There will be also a practical work where the code will be written in PyCharm with help of the FastAi library.

These main objectives will be present in the scientific and technical part, which makes 80 percent of the whole report.

On the scientific side, some questions are asked, for example how and why using convolutional neural networks to recognize images? In this part are also very significant pictures that explain better the scientific process.

Then to the technical side of the report, the whole code will be programmed in Python using PyCharm. This is an integrated development environment for the programming language Python. In the beginning, to test the code, I used a data set of Kaggle and after it will experiment on the own pictures that are taken from the internet or from outside. But the most impressive piece of the code will be shown at the end of the technical part.

2. Project description

2.1. Scientific domain

Artificial intelligence (AI) is the most significant domain of this project because the meaning and goal of the project are connected with AI. It is a part of computer science, that creates a machine that has similar features as humans, like recognizing objects. To achieve such a feature, AI relies on mathematics theory and complex algorithms to solve this issue and that's what this project is doing. In the following, AI is having two important subsets, where the report will be more focused. The first one is machine learning and the second one is the deep learning which is the subset of machine learning.

2.1.1. Machine Learning [4] and Deep Learning [3]. Machine learning (ML) is part of artificial intelligence, it is built on top of mathematics processes to train a machine to learn a pattern. In the past, machines are not available to do something themselves without human programming. Now the time has come where a machine can learn from experience like a human. One example is when the input will be a data set of face reaction (sad, happy,..). However, to reach that a machine can say to themselves which face reaction it is, there are a few steps needed:

1. Input the familiar data set.
2. Then the machine will analyze the data.
3. It finds some useful features of each class (sad, happy...).
4. After this, it will input unknown images that don't exist in the data and it will try to say which reaction it is with the help of the features of the classes.
5. In the end, it can get some feedback if this is false, like this it can learn from the error and try it again, so it collects experience.

Machine learning has three types do to it:

a. Supervised → The machine learns from the trained data that will be given to it and on the end it gets feedback. This type is the one that the project will use.
b. Unsupervised → Here the machine has no specified data, but it learns from the features of the unknown data input and classifies them.
c. Reinforcement → Here the machine also doesn't have a specified data set, just one unknown input, so it learns in his own with help of the feedback.

It continues with deep learning, which is an important part of machine learning and convolutional neural networks. It imitates biological neural networks to learn efficient and independent. It also uses it to recognize patterns in the data, like this it can employ them in new inputs and make by every try the pattern more refined and accurate. Deep learning can also be called as features learning.

2.1.2. Convolutional neural networks [1]. Convolutional neural networks (CNN) is a space invariant artificial neural network (SIANN), which are extensions of Neural Network and that have the task to analyze visual imagery.

At first, it will be explained quickly how the neural network accepts the pictures. Every picture can be represented as a matrix of pixel values with features like color, figure, etc..., so the NN works with them to recognize the images.

There is one traditional neural network also called multi-layer perception (MLP), but this is not an efficient way to classify images, so we use CNN. To understand the reason, it will be explained with an image. When the MPL get an input picture and the flower appears for example on the bottom right, but in the other input picture the flower appears on the bottom left, then the MLP will assume that the flower will be always in this part of the image and it doesn't analyse the other areas. To recognize images, it's needed to analyze the whole picture. It can also happen that some information about the pictures gets lost during the process in an MLP.

The CNN hasn't this problem, 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.

2.2. Technical domain

The whole code of the project will be written with the aid of the IDE PyCharm. The code is split into two modules, one to build a model with ResNet34 and another for the real-time classification.

2.2.1. Python/PyCharm. Python is a beginner-friendly programming language, which is available for everybody without having a payable license. It was developed by Guido van Rossum in the year 1991 and it is one of the fastest learnable programming languages because of his easy understandable syntax. Additionally, it posses multiple programming paradigms, which helps python to find the right way of code to the solution. There exist three different styles, the object-oriented, the aspect-oriented, and the functional programming. They contains a lot of helpful libraries, that were chosen to help the developer to write easier a program.

However, there is something called an integrated development environment(IDE) which is a software environment that includes tools, debuggers, build, and a code editor with the Python shell. It is easier to work with it because if something is wrong on the code, there are many possibilities to solve it simpler and the code is more readable. In the IDE, it can be created more files together and it is easier connected with other codes. It can be said that in an IDE is everything together in one window. When using for example a programming language like Python, it has two windows, but Python has also an IDE PyCharm that can be used. It exists also the Integrated Development and Learning Environment (IDLE) which contains the basic package of the IDE. It can be said that IDLE is an IDE just IDLE is more for learning processes. In this project, it will be used the IDE PyCharm, which has been developed by JetBrains in 2010. It has many useful features, for example, it has access to Google's app engine, support from the web framework Django and it helps to complete and create the code.

2.2.2. FastAi [9]. Nowadays, humans still have difficulties to learn, implement, and develop the AI algorithms. Therefore two employees from the University of San Francisco, Rachel Thomas, and Jeremy Howard established a startup of FastAi which is supported by PyTorch. It is a famous deep learning library, which provides a lot of features and which avoid using complex algorithms. One example is to build an easily readable classification code in this project. The use of this library will be shown in the technical part by the build of the model.

2.2.3. OpenCV. OpenCV also named as Open Source Computer Vision, is a library which is mainly intended for the domains of image processing and machine learning. It contains a lot of useful ML algorithms, which a user didn't need to implement anymore. However, one of the features that this project is used from OpenCV are the functions of Video capture where it can be created a real-time classification.

3. Pre-requisites

Before starting a project, it is necessary to search at first for a lot of information to understand the topic. Therefore it will be shown the scientific and technical previous knowledge which you should know and do when you are interested to make this experience.

3.1. Scientific prerequisites

Scientific knowledge is an important one because everything is built on them. It is like the theory that is needed to learn before beginning to understand or to program. At first, are should not forget that everything that will be done, needs to be written on an excel sheet for example for organizing ourselves. This sheet should be submitted every second week in BMT, where it also needs to check the meetings with the tutor. In the first days, it was hard to understand how machine learning works. With the time it was needed to make new researches to understand the topic better because it is a large domain where often some little significant information gets forgotten. In general, it is the first time for me to work with such a topic and where I need to be focused on this and to do every week the required work.

In most explanation on the internet is it difficult to understand because there is no visual construction to imagine. In this situation, videos helped a lot to imagine a CNN and how it was built and how it works. To begin there are some researches about AI in general then it continues with machine learning and deep learning. After I understood these big topics, the NN's were in the line to be focused on. Most time was spent on the researches and writing for this report, because when someone has never done something like this then this domain can be difficult to learn but not impossible. In these weeks I learned a lot about AI in general and my English writing skills are also better now.

3.2. Technical prerequisites

After having the necessary information about the project, it can be started with the programming. At first, it is needed to install Python with PyCharm which was already on the laptop, of the first semester in BICS. Then it comes the tricky part to download libraries like torch and FastAi in a windows laptop. When having a Linux, it will be simpler, but with the help of the tutor, I learned how to solve it. Then it is needed to choose a data set, which was found in Kaggle with a large number of images. Then it was time for programming, so loading the images in PyCharm and start splitting the data into train and test folders. This part took time because I didn't have an image in my head of how it will look like but with the time, I understood how it works, and with the help of my tutor.

Afterward, I continue with the code and always followed the plan what was created before starting with this part of the project. It should make a train and test process and build a model with these images which were split. There was some situation where it took a long time to build the model because the laptop didn't have enough RAM, but in the end, it was fine. Then it was added one more code for the real-time classification, where I got a lot of help from my tutor. This part was easier because the model was already built and there not much to do. In the end, it was everything loaded in GitLab, where it is simpler for the tutor to look over the code. In this time, I learned a lot about python and GitLab.

In the end, it should be making two videos in two different languages (English, German), where will be explained and showed the knowledge and results which were gleaned from the project. This will be done with PowerPoint for the first time.

4. Scientific Deliverable: The brain of convolution neural networks

In the scientific deliverable will be answered two significant questions, that gives us more information about this project.

1. What is the semantics of a CNN?
2. What can a CNN be useful for?

4.1. Functional Requirements

This section presents the function of the project, it is a little explanation about this work, but more details will be found during this section.

4.1.1. Data processing of flowers. CNN is a very useful and interesting machine, which has a lot of functions. One of the main capacities is in the moment where the user inputs an unknown image of a flower in CNN, then it should categorize the plant with the class. Before this, the machine needs to learn from the train images that it takes from the data set. After there will be input the test images to look if it works fine. However, this deliverable should show how this machine analyse the images.

4.2. Non-Functional Requirements

Here will be shown the expected properties of the project but more details will be found during this section.

4.2.1. Efficiency of the CNN. Scientists have shown that CNN is one of the best models for classifier data because it uses deep learning features. It works precisely by analyzing the whole area of the data, like this, we get better results which are shown in the pictures during this section. However, the experience shows that the efficiency depends on the architecture, which says that ResNet34 gives better outputs then a lower ResNet. Depending on the computer it can take time to build the model with ResNet34.

4.3. Design

4.3.1. ConvNet mechanism. A ConvNet (CNN), which was mention in the description part, is a space invariant artificial neural network and it can be also defined as a deep learning algorithm. It is one of the popular image processing and classification tools from the Neural Networks group, that's why it is used in this project. Additionally, it can also be

used for audio signals application. There exists one better NN which is more specified for sounds, which is called recurrent neural network (RNN).

To categorize the images, CNN will use the supervised type of machine learning, where it gets an input (the train data) and learns from features of the images. After the training process, it gets the test data, to check if the CNN can recognize most of the pictures and put them to the right class. In this way, it learns from his experience, like humans, and makes fewer faults.

A CNN works similarly to a classic NN with some special unique properties. The mechanism of a CNN will be better understood with the explanation of the basis from a classic NN. At first, it will be analyzing the core of the NN and that is the architecture, which is created to imitate the human brain connectivity design of neurons.

A NN contains 3 main layers:

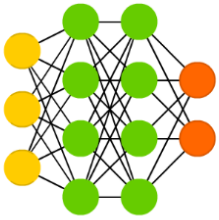


Fig. 1. Classical Neural Network model (2)

1. **The input layer**: Accepts input of different forms from the outside world.
2. **Hidden layer**: They do all the calculation on the input to get an useful output. In this section will be just explained what they do, the formulas will be shown in the production section.
3. **Output layer**: They hold the outcome of the calculation and extraction and give us a readable output.

How knowing, NN contains artificial neurons and each layer contains neurons that are fully connected, which can be observed in the above picture. This means that all neuron relates to the neurons in the previous layer. Every neuron has his own weight and bias (which will be explained after), where in CNN all neurons have the same weight. This full connection doesn't allow NN to make any assumption about the input data, and so a CNN is a better image processor.

Now we look over the architecture of a CNN, which is built of three main layers that are hidden layers [5]:

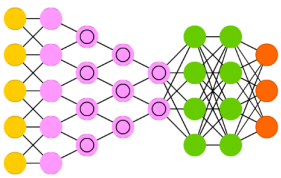


Fig. 2. Convolutions Neural Network model (2)

1. **Convolutional Layer (Feature Learning Part)**: Knowing that a computer sees images as an array of pixels. The NN's works with them, this is the part where linear algebra appears. However, the neurons of the layers are the most important ones, each of them has the same weight and bias and they are not fully connected, they are just connected with the close ones. This connection can be observed in the picture above. In this way, the CNN can make an assumption about the data because every set of neurons can take different regions of the image and not just one part.
The weight of the neurons is used to build automatically a matrix also called filters here, which will be used to go through the images matrices and create a new output matrix. A bias [2] is a constant (bias=1), which is important for the function of this process. It allows the activation function to move the left and right, in general, it helps us to get better results.
In the picture above we see the yellow input layer, where the input data is a set of neurons that are 3 dimensional (height*width*color). The color is created of 3 channels (Red, Green, Blue) and when it is a black/white picture then the color gets the value 1. The height and width build the matrix which will be used as input for the filter part.

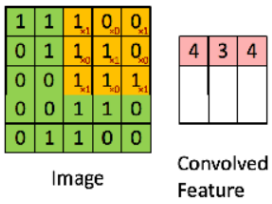


Fig. 3. Dot-Product between two matrices (1)

The filters are neurons, that build a matrix that is chosen by the CNN as explained before. By the picture above, it can be observed the dot product calculation between the filter matrix (Yellow with red values) and the image matrix (Green). The whole image will be analyzed. Here the CNN starts to learn from the features of the images from the matrices and save the information like a brain. As an output, we get the Convolved Features map which will be used as input for the pooling layer.

2. **Pooling Layer (area with circles: Feature Learning Part)**: This part of the architecture helps to simplify the input Convolved Features maps. It reduces the parameters of the image when they are too large, in detail, it reduces the dimensionality of the input maps and keeps the important information. This method can be called downsampling and there exist three different types the max pooling, the average pooling, and the sum pooling. Max pooling takes the biggest elements from the map, the average pooling takes also the biggest element and

the sum pooling takes the sum of the elements. In most cases, CNN uses max pooling, which is the simplest way.

3. **Fully Connected Layer (Classification Part)**:

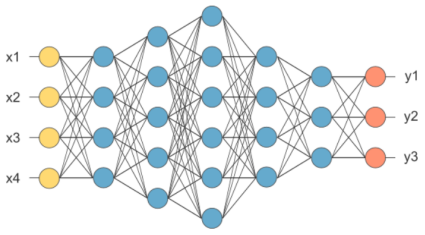


Fig. 4. Full-Connected Layer (2)

This picture is a zoom of the orange area. This process is the same as from a classical NN which is also fully connected. The (x1 x2 x3 x4) are the outputs from the pooling layer. In the yellow part, the input matrices will be flattened into vectors, like this, the vectors can be used in the blue area. In this part, the model will be built from the features, which were saved during the process. After the model was created, it will be using an activation function to make the classification, where (y1 y2 y3) the classifier outputs are.

This is the architecture of a CNN and steps 1 and 2 can be done multiple times because CNN is multi-layer networks. CNN has a lot of types of architectures and this project has used one time the ResNet34. In the production section, it will be explained the structure of ResNet34.

4.4. Production

4.4.1. A deep look in ConvNet with Architecture ResNet34. At first, we explain some significant terms which help to understand the CNN mechanisms. After it will be talked about the places where these terms play a role.

- **Stride:** This is useful for the dot product operation. We have the input matrix and the filter, the stride says how many pixels the filter should take to move through the input matrix. When the stride gets bigger than it will create smaller feature maps.
- **Depth:** The depth of layers is depended on the numbers of filters that we have for convolution operation.
- **Kernels:** Kernel is a filter, that produces channels (It is used in Convolutional layers).
- **Filter:** The neurons in the layers that create a matrix are called filter/kernel/ feature detector. It checks the color composition or brightness of the input matrix to create together a new matrix.
- **Channels:** Channels are matrices that depend on the images, when it is a colored image then it has three channels for Red, Green, and Blue in the beginning. When going deeper with the layers, the numbers of channels increase. It has values of pixels between 0 to 255. To calculate the number of channels, it can be said that the number of Kernels is equal to them. In the picture below, it can be seen that the filters created with the input matrix three channels and when adding the channels together we get the feature map.

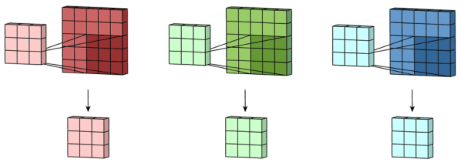


Fig. 5. RGB Channels (1)

- **Layer:** This is a domain where the parameters of neurons are placed, and which builds the architecture by a CNN. It is a general word for the input storage of the previous step.
- **Gradient [8]:** Gradient is the rate at which values change concerning weight or bias. This gradient is most used in the training process during the back-propagation algorithm. This part is important because sometimes it is difficult or impossible to train complex network architectures. However, the gradient is defined by the deep learning layers so by the convolutional layers. In most cases it appears the gradients problem:
  - vanishing gradient problem: If the start values in the layers are small then the gradient goes closer to zero. It will be harder to train the model because it will be difficult to move through space.
- **Back-propagation:** The training process has different algorithms and one of them is the back-propagation algorithm, which gives accurate and fast results. To understand the algorithm, it will be shown some steps.
  1. To begin it is important to set random values for the filters and parameters in the layers.
  2. Then put an image for training in the CNN, that goes through the forward propagation (Convolution, ReLU, pooling and in the end the FC layers).
  3. When having the outcome probabilities for each class, and the first outcomes are not similar to the real ones, what is normal here because it was chosen random values for the in the beginning so it will also be a random output.
  4. To get good probabilities, it is important to calculate the total error of the last layer [7].

$$\text{Total Error} = \sum \tfrac{1}{2} (\text{target probability} - \text{output probability})$$

  5. Knowing that the total error is large, it is important for our training process to minimize the error, and that will be done with back-propagation. It calculates the gradient of the error and uses the gradient descent to update the filters and parameters. That means that the weights will be set propositional to the error, so it

can get better outcomes which are closer to the real results. When putting the same picture again in the training process then the filters matrix and connected weights are actualized with new values where CNN can learn better from them. One more information is that just these two values will be changed, the filter size, number of filters, and the architecture still the same, they are fixed from the beginning.

6. Now we can repeat the steps from 2-5 with all images.

How seen, CNN learns from the updated parameters and filters because they are closer to the real ones. When adding one unknown image then hopefully it will recognize the same parameters in the new one.

- **ReLU (Rectified Linear Unit) [10]:** It is an activation functional layer that will be used after every convolutional layer and it can avoid the vanishing gradient problem. There is a function that avoids the problem better and that is the leaky ReLU. The ReLU is an element-wise operation which avoid every negative value in the convolutional layer and transform it into zeros. It is as a non-linear function, but it looks like a linear function however it has a derivative function and it is allowed for back-propagation.

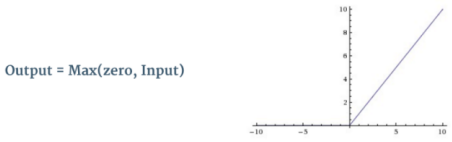
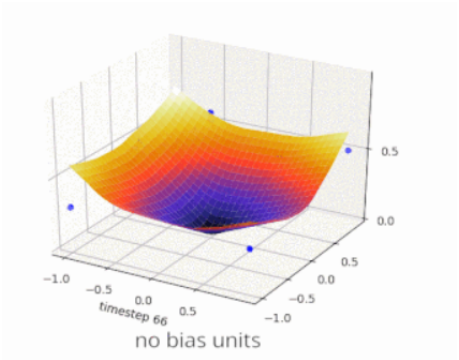
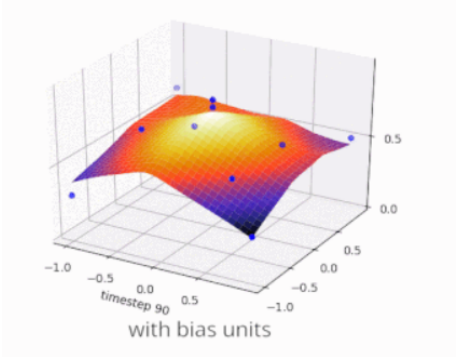


Fig. 6. ReLU Diagram

- **Bias [2]:** How was explained in the design section a bias is useful to make our outcome better. It is one bias neuron in every layer and with the help of them, the network can have arbitrary outputs. It influences the dimension, because without this it would be a simple dot product result. The bias has always the number 1 and so it avoids that the value 0 appears. As know the bias allows the activation function to move through the axis so it can move the function as it needs. (3)





⇒ **Architecture ResNet34 (Residual Network) [6]:** ResNet was the winner for image classification, detection, and localization in 2015 and the most used architecture in CNN's. In the picture below it can be seeing the structure of the architecture.

Layer name	ResNet34
conv1	7×7, 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

Fig. 7. ResNet34 structure (5)

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. In the picture below we have the formula to calculate the size of the output image after a convolutional operation.

Fig. 8. Formula for the output size

Here we see the difference between max pooling and average pooling:

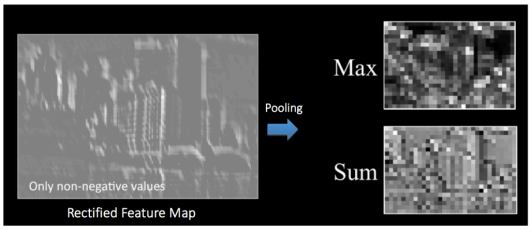


Fig. 9. Max and average pooling

4.5. Assessment

It can be said that all requirements are satisfied. This section shows how CNN works and his ResNet34 architecture. It explained all steps of the whole structure with pictures that allow us to imagine and understand this type of machine learning and deep learning better. We learn a lot of significant terms that play a big role in this domain. Now it is simpler to create the code, after knowing what is needed and why it works.

5. Technical Deliverable: Classification process in PyCharm

5.1. Functional Requirements

This section presents the function of the project, more details will be found during this section.

5.1.1. Build of a model. The creation of a model in our generation isn't anymore difficult because of our useful libraries that are developed to help programmers to build a model for the classification. The library that this project use is FastAI with his functions cnn_learner() and learner.fit_one_cycle() that build for us a model. The code will also use a pre-trained model of ResNet34.

5.1.2. Real-time classification. The second PyCharm module is for real-time classification. When this module runs then it should appear large windows and small pink windows where the user can show through his front camera a flower. In this time the user can start the classification with one keyboard click and stop it. When the classification starts, the program should write automatically above the pink window which flower it is. There will also exist one click to stop video capturing.

5.2. Non-Functional Requirements

Here will be shown the expected properties of the project and which experience got during the work, but more details will be found during this section.

5.2.1. Usability of PyCharm. If a user using this project it will be easy because he needs just to run the project. Everything goes from alone with some clicks and it is also understandable what PyCharm does.

If a user tries to program on his own, then PyCharm is an easily usable and readable program because of his syntax. It contains also a lot of functions that help us to reach our goals faster, to make fewer mistakes, and to simplify the code. My experience with PyCharm was in the first days not directly easy to use because of my missing programming skills but with the time it was an comfortable habitue.

5.2.2. Performance. The performance of a program depends on the computer that the user has. The response time in my PC (HP Pavilion 14-ce3011ng) for the classification takes more time then by other PCs. In general, for the video capture file it takes just 5 sec to run it, but for some function like the learner.fit_one_cycle() from the FastAI library, takes a lot of time when using a higher architecture like the ResNet34. However, PyCharm has not a bad performance, it is very efficient for codes.

5.3. Design

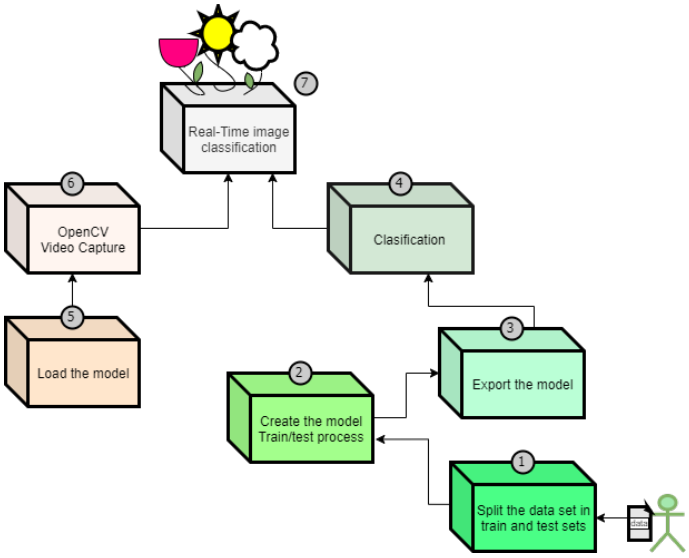


Fig. 10. Classification Progress diagram

5.3.1. Classification progress in PyCharm. In the picture above, the chart of the classification progress is presented. In this section, it will be explained which steps are used to reach the goal of a real-time flower classification. The functions that are used will not be explained in details, just why they

are used and where. In general, they are 6 significant steps to satisfy to reach the main step (7). There are two parts, which are in python two different modules, one for the general classification, and one for the video capture code. The green one on the right side (general classification), is the starting point, after when it finished the green blocks until number 4, then it goes over to the orange side (video capture). Here are also some modules that are used during this process:

- **FastAi and OpenCV** are explained in the description section of this report.
- **Random:** Chose random objects, numbers....
- **OS:** An Operation System works with computer resources like a disk drive and it organize the user interface. It supports also services for software application.
- **Shutil:** The shutil module helps to copying files and directories.
- **NumPy:** NumPy is a package for scientific computing.
- **PIL:** PIL is a python imaging library that helps to open, edit and save different types of images.

1. **Split the data set into train and test sets:** The user needs to choose a data set of images, this project takes flowers for the classification. This data set was downloaded from Kaggle because its faster to take prepared data than to make his own that contains over 3000 images. This set has 5 categories (tulip, sunflower, daisy, rose, dandelion), which has 1000 images in each of them.
   After the data needs to be loaded in PyCharm, which he can use for the split.To begin the code, it is needed to know how many percents the train set and test set obtain from each class. In general, it is usually that the trainset is 80% of the data and the test set is 20% from the data. Then it is important to take of each class the same number of images. In this project, it is taken 50 images of each category to spend less time and to look at how it works. Of course, the user can take more images of each class just the duration of the training process will take more time.
   Then it is time to begin with the split of the classes, after this it is necessary to create two folders to save the images that are taken. In this project, the folders in the PC are called train_set and valid_set (valid=train). In this step, are used some packages that PyCharm offers, like FastAi, the OS (operation system), random, shutil and NumPy modules.
2. **Create the model-Train/Test process:** Before building the model, it is necessary to load and split the train and test data which are created and ensure that every image has the same size which is important for the classification process. This part will have all information about the sets, which will be useful for CNN.
   However, to build the model, the user needs to choose a type of architecture of CNN. This project used the ResNet18, which was pre-trained [11] (more details in the production part). Now it can be started to create a CNN model with the use of FastAi functions.
3. **Export the model:** There exist two ways to get the created model. The first one is useless, it saves the model in the PC and after it loads it. This method is not so effective, because it saves only the parameters but not the whole architecture which is needed to produce something. Therefore, one should use the export method, where it saves the model with the architecture and all useful information of the data after the export file can be loaded to produce a classifier. Here are also some useful functions that can get from FastAi.
4. **Classification:** This is the end of our classification code. Here the user can take his own pictures and try out if CNN works fine. FastAi will help to load your pictures and to put it on CNN.
5. **Load the model:** Now it is time to work on the second file where the real-life classification takes place. After creating our CNN model, this part will be easy. It is just needed to load the export file and to use it, like before.
6. **OpenCV-Video Capture:** In this part, it is needed to create a small function to realize a video classification. Here the package cv2 is very helpful because it was developed for such stuff. It can create visual windows that relate to the front camera, where the user can show the real flowers that he wants to classify. In this visual window, it is needed to be written which category of flower it is. However, when the video capture runs, the code needs to transform the life video into images that can be put into the model. In this case, it can be used the PIL package.
7. **Real-time Image classification:** Now the goal is reached, the user can go out and take a flower and look at which type it can be. One more information is that the real-time classification file doesn't classifier during the whole life video because the user can choose when the code should start with the process. He needs just to click "s" on the keyboard to start and stop.

5.4. Production

In this section, it will be focused on the different functions which was used in the steps 1-7 on the design section.

1. – **os. path. exists ():** This is a function of the library os and it helps the code to check if a path exists or not. This we used to check if already exists folders for the train and valid images.
   – **os. mkdir ():** If didn't exist a folder then it can be created folders with this function, with just writing the path with the folder name.
   – **os. path. join ():** This function helps to get the class paths to take after the images from this class. At first, it will be written the path and then the folder name of the classes (example: tulip). In this project are 5 categories (tulip, sunflower, daisy, rose, dandelion).
   – **random. sample ():** Here it will be taken the desired amount of images from each class because it is important to have the same number of images from

each category of flowers. In this project, we take 200 images of each class.

- **np. split ():** This is the most important part because this split our 200 images of each class to a train and valid sets. The train folder has 80% of the images (160 of 200) and the valid folder has 20% (40 of 200). Important information is also that it is needed to use a big amount of images for the train/valid process to get better results because CNN uses a supervised method to learn.
- **np. array:** it will be used with the function before and make from the images an array. After in a new line it will be used the function to. list to get a list with the separate elements of the array. In the end, it is needed to have the images and not arrays.
- **shutil. copy ():** In the end, it will be taken every single image and copy it with the help of shutil in the train/valid folder and with in right category.

2. Those functions are from the FastAi library!

   - **ImageDataBunch.from_folder ():** This is a class where you can put anything that represents an image into it, for the test, train, or validation domain.
   - **plt. show ():** This function is helpful to show diagrams or images that are created.
   - **cnn_learner (data, resnet34, metrics=error_rate, pretrained=True):** Here is the part where the model will be created with the architecture resnet34 and it will be used a pre-trained model. The pre-trained model is significant because it simplifies the code part. With a pre-trained model, it is not needed to spend a lot of time building an algorithm for the training process, it is used already a model that was trained from someone else. It is like help for the beginning train process of the model.
   - **learn. fit_one_cycle ():** Here it will be looking at how good the model is. It shows the train and valid loses and it will run 50 times until it has a good result (the losses decrease).
   - **ClassificationInterpretation. from_learner ():** This class produces a confusion matrix where it shows how many images are correctly classified and which not. Then it can also make a table of images with the most losses. After there are special functions that need to be added to show the results of this function.

3. - **load_learner (export):** This is the part where the code uses the model that we had created. The model was saved in a folder named export and this will be taken to start the classification. Here it is important to use the learning.export and not the save.export, because when we save the model, we don't get the architecture of the model. The function export gives only the architecture and that's all that is necessary for the classification.

4. - **open_image ():** This function will be used to get only the .jpg image that is chosen randomly from the internet, which will be used to test our model.
   - **learn. predict ():** Here it will be put a single image and it gives an output, which defines the tensor(class) and how near this prediction is by the truth of this tensor. In the picture below it can be seen the output of the code and how excellent the results are. The first line says which flower picture was used to classify and the two-line below is the prediction and it is every time correct.

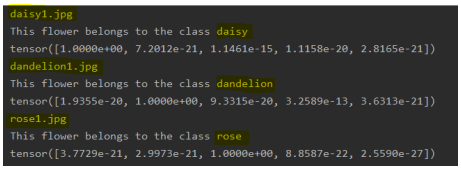


Fig. 11. Output1

5. Here will be repeating the step 3), where the model will be used to make a real-time classification.

6. - **cv2. VideoCapture (camera_port):** This function creates a video capture object which allows capturing a live video. When the camera_port=0 then it used the first camera and if the camera_port=1 then it is the second camera and so on. In this code, it will be set to 0 and that's the front camera of the laptop.
   - **cv2. namedWindow ():** Here it will be given a name for a specified window and to set the window it will be used this function cv2.WND_PROP_FULLSCREEN inside. This means it uses the whole screen of the laptop.
   - **cv2. VideoCapture (camera_port). read ():** This function reads all frames of the video, which will be needed after.
   - **cv2. rectangle ():** This function draws a rectangle on the frames that are chosen, for example, the frames of the whole screen of the laptop. This rectangle needs also some sizes, it can be used the two parameters upper_left and bottom_right to set them. It will be a pink rectangle in the middle of the screen where the flower needs to appear, which will be classifying.
   - **cv2. imshow ():** This allows the rectangle to appear on the screen where the user can be seen from the front camera.
   - **if cv2. waitKey (1) == ord ('s'):** This is useful when the user wants to stop the video capture or the classification. It is just needed to click on the keyboard by 's' or any other letter, which was chosen.
   - **Image ():** This function is from the PIL library and transforms the frames in images that will be used to classify.
   - **cv2. FONT_HERSHEY_SIMPLEX:** This function will be used for the putText function to say that the text will be on the front of the window.
   - **cv2. putText ():** This is helpful to put a text on the

screen because, during the classification, it should be written on the screen what the prediction is.
  - **cv2. VideoCapture (camera_port). release ():** This is at the end of the code to stop the video capture.
  - **cv2. destroyAllWindows ():** This function should also be at the end of the code to remove the windows and the code will not run anymore.
7. This is the output, of this part of code. It can be seen the whole screen with a little pink rectangle in the middle. Then there exists a little 'Selection' window, which shows what appear on the pink rectangle. After starting the classification, the flower name will be written above the pink rectangle.

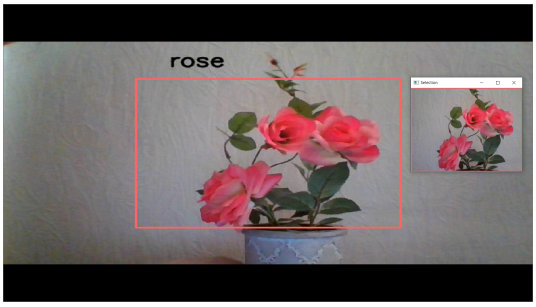


Fig. 12. Video-Capture output

5.5. Assessment

It can be joyfully said that all requirements are satisfied by the project. In the design section, it can be seen the way which is taken to create this project in Python. There is exactly explained what was done and after in the production section, was explained the in detail of the different functions that are used to reach the goal. In the pictures can be seen that the code works fine with good results of resnet34. In general, it can be seen that the programming language was not difficult to use and it worked everything effectively. When problems appear, one can quickly found a solution on the Internet, because most people work with Python when they make a project about machine learning.

Acknowledgment

After all this time, my biggest thanks go to my tutor Emmanuel Kieffer. He is one of the best tutors I have had because of the patience and effort he had to help me when I had trouble. I learned a lot about AI and Python, and I will continue to deal with AI in the future.

My thanks also go to my professor Nicolas Guelfi who made this experience possible and helped me to understand the project better. I also thank the BMT management for the great work and help they provided us with.

6. Conclusion

In the end, the project was a great achievement, where everything was accomplished, even if it was difficult in our current situation with the Corona Virus to work effectively. However, work was input and the results show the positive output. Now the classification worked with the flowers and we can expand it because we now have the basis. It can even create an app where it will connect with the code and make a real-time classification from the mobile phone.

References

[1] **Convolutional Neural Network:**
https://cs231n.github.io/ ,
https://www.youtube.com/watch?v=m8pOnJxOcqY
https://www.youtube.com/watch?v=K_BHmztRTpA
[2] **Bias of CNN:**
https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/
https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks
https://missinglink.ai/guides/neural-network-concepts/neural-network-bias-bias-neuron-overfitting-underfitting/
[3] **Deep Learning:**
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
[4] **Machine learning:**
https://machinelearningmastery.com/crash-course-convolutional-neural-networks/
[5] **CNN main layers:**
https://machinelearningmastery.com/crash-course-convolutional-neural-networks/
[6] **ResNet34 architecture:**
https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
[7] **Mathematical formula:**
https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/: :text=In%20a%20CNN%2C%20each%20layer,weights%20of%20the%20Conv%20Layer.
[8] **Vanishing Gradient:**
https://cv-tricks.com/keras/understand-implement-resnets/
[9] **FastAi:**
https://docs.fast.ai/vision.models.html
[10] **ReLU:**
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
[11] **Pre-trained Model:**
https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

7. Appendix

Source of the image used in NetBeans.

1) **Matrices:**
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
2) **Structures of NN's:**
https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
3) **Bias graph:**
https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks
4) **Mathematics Formula:**
https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/: :text=In%20a%20CNN%2C%20each%20layer, weights%20of%20the%20Conv%20Layer.
5) **ResNet architecture:**
https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
6) **Video images:**
https://giphy.com/search/intro