Project Objectives

Image Pre-processing:

- Image pre-processing are **the steps taken to format images before they are used by model training and inference**. This includes, but is not limited to, resizing, orienting, and colour corrections
- As a Machine Learning Engineer, data pre-processing or data cleansing is a crucial step and most of the ML engineers spend a good amount of time in data pre-processing before building the model. Some examples for data pre-processing includes outlier detection, missing value treatments and remove the unwanted or noisy data.
- Similarly, Image pre-processing is the term for operations on images at the lowest level of abstraction. These operations do not increase image information content but they decrease it if entropy is an information measure. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task.

There are 4 different types of Image Pre-Processing techniques and they are listed below.

1. Pixel brightness transformations/ Brightness corrections
2. Geometric Transformations
3. Image Filtering and Segmentation
4. Fourier transform and Image restauration

Convolutional Neural Network

The Convolutional Neural Network takes a different approach, mimicking the way we perceive our environment with our eyes. When we see an image, we automatically divide it into many small sub-images and analyze them one by one. By assembling these sub-images, we process and interpret the image. How can this principle be implemented in a Convolutional Neural Network?

The work happens in the so-called **convolution layer**. To do this, we define a filter that determines how large the partial images we are looking at should be, and a step length that decides how many pixels we continue between calculations, i.e. how close the partial images are to each other. By taking this step, we have greatly reduced the dimensionality of the image.

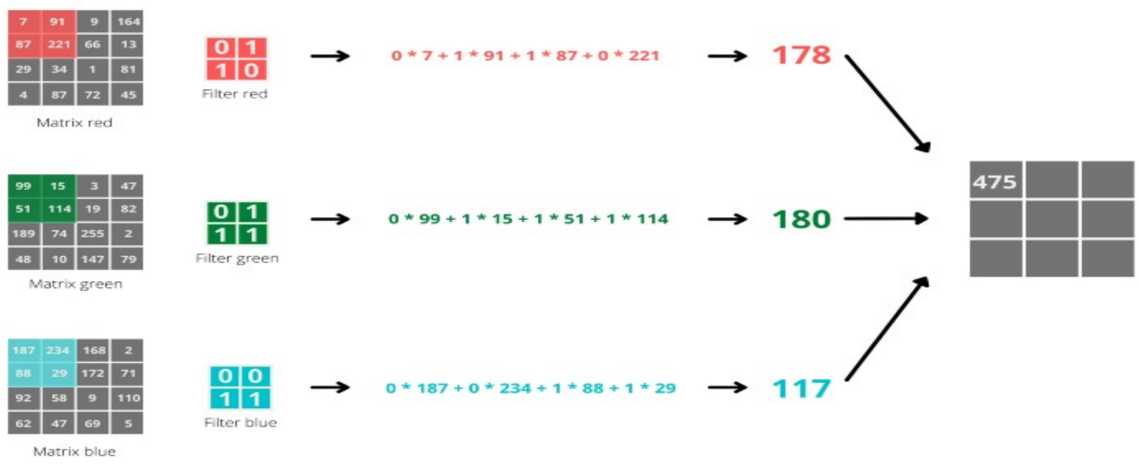
The next step is the **pooling layer**. From a purely computational point of view, the same thing happens here as in the convolution layer, with the difference that we only take either the average or maximum value from the result, depending on the application. This preserves small features in a few pixels that are crucial for the task solution.

Finally, there is a **fully-connected layer**, as we already know it from the normal neural networks. Now that we have greatly reduced the dimensions of the image, we can use the tightly meshed layers. Here, the individual sub-images are linked again in order to recognize the connections and to carry out the classification.

Now that we have a basic understanding of what the individual layers roughly do, we can look in detail at how an image becomes a classification. For this purpose, we try to recognize from a 4x4x3 image whether there is a dog in it.

Detail: Convolution Layer

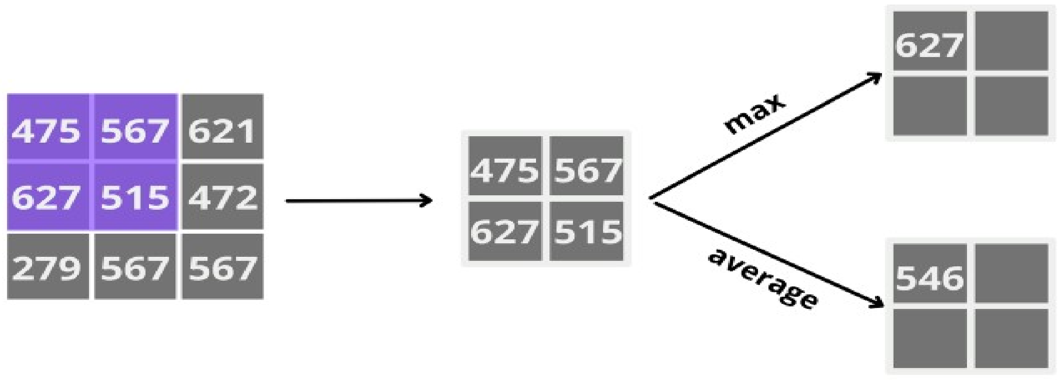
In the first step, we want to reduce the dimensions of the 4x4x3 image. For this purpose, we define a filter with the dimension 2x2 for each color. In addition, we want a step length of 1, i.e. after each calculation step, the filter should be moved forward by exactly one pixel. This will not reduce the dimension as much, but details of the image will be preserved. If we migrate a 4x4 matrix with a 2x2 and advance one column or one row in each step, our Convolutional Layer will have a 3x3 matrix as output. The individual values of the matrix are calculated by taking the scalar product of the 2x2 matrices, as shown in the graphic.



Convolution Layer

Detail: Pooling Layer

The (Max) Pooling Layer takes the 3x3 matrix of the convolution layer as input and tries to reduce the dimensionality further and additionally take the important features in the image. We want to generate a 2x2 matrix as the output of this layer, so we divide the input into all possible 2x2 partial matrices and search for the highest value in these fields. This will be the value in the field of the output matrix. If we were to use the average pooling layer instead of a max-pooling layer, we would calculate the average of the four fields instead.



Pooling Layer

The pooling layer also filters out noise from the image, i.e. elements of the image that do not contribute to the classification. For example, whether the dog is standing in front of a house or in front of a forest is not important at first.

Detail: Fully-Connected Layer

The fully-connected layer now does exactly what we intended to do with the whole image at the beginning. We create a neuron for each entry in the smaller 2x2 matrix and connect them to all neurons in the next layer. This gives us significantly fewer dimensions and requires fewer resources in training.

This layer then finally learns which parts of the image are needed to make the classification dog or non-dog. If we have images that are much larger than our 5x5x3 example, it is of course also possible to set the convolution layer and pooling layer several times in a row before going into the fully-connected layer. This way you can reduce the dimensionality far enough to reduce the training effort.

How deep neural networks detect the disease

Deep learning techniques, and in particular Convolutional Neural Networks (CNNs), have led to significant progress in image processing. Since 2016, many applications for the automatic identification of crop diseases have been developed. These applications could serve as a basis for the development of expertise assistance or automatic screening tools. Such tools could contribute to more sustainable agricultural practices and greater food production security. To assess the potential of these networks for such applications, we survey 19 studies that relied on CNNs to automatically identify crop diseases. We describe their profiles, their main implementation aspects and their performance. Our survey allows us to identify the major issues and shortcomings of works in this research area. We also provide guidelines to improve the use of CNNs in operational contexts as well as some directions for future research.

Accuracy:

Accuracy is a metric used in classification problems used to tell the percentage of accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

This formula provides an easy-to-understand definition that assumes a **binary classification** problem. (We discuss **multiclass** and **multilabel** problems in the second part of this article.)

In the binary classification case, we can express accuracy in **True/False Positive/Negative** values.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Building a webpage using python

There are many modules or frameworks which allow building your webpage using python like a bottle, Django, Flask, etc. But the real popular ones are Flask and Django. Django is easy to use as compared to Flask but Flask provides you with the versatility to program with.
To understand what Flask is you have to understand a few general terms.

1. **WSGI:**

   Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

2. **Werkzeug**:

   It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

3. **jinja2 :**

   jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

Flask:

It is a web application framework written in Python. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Installation:

We will require two packages to set up your environment. *virtualenv* for a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries and the next will be *Flask* itself.