**SGD TUTORIAL**

Learning means finding the best parameter values with the least error.

The convolutional neural network is divided into two parts: Feature extractions, and a fully connected layer. All together is called the feature learning

SGD, often referred to as the cornerstone for deep learning, is an algorithm for training a wide range of models in machine learning. [Deep learning](https://en.wikipedia.org/wiki/Deep_learning) is a machine learning technique that teaches computers to do what comes naturally to humans. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Models are trained by using a large set of labeled data and neural network architectures that contain many layers. Neural networks make up the backbone of deep learning algorithms. A neural network that consists of more than three layers which would be inclusive of the inputs and the output can be considered a deep learning algorithm. Due to SGD’s efficiency in dealing with large scale datasets, it is the most common method for training [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks). Furthermore, SGD has received considerable attention and is applied to text classification and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing). It is best suited for unconstrained optimization problems and is the main way to train large linear models on very large data sets.

These hyperparameters, learning rate, and epoch are related to the Stochastic Gradient Descent Algorithm and it is an optimization algorithm.

I think one of the reasons why CNN was used is not to lose spatial information when images are flattened. for the feed-forward process.

Pooling is used to reduce the size of the image. Thus reducing the number of parameters.

Output size = (inputSize - filterSize + 2 \* padding) Stride + 1

OutputSize = [ (I – F +2P) \*S + 1 ]

Image Processing is a method for converting input images into digital form images, carrying out some operations on them, obtaining enhanced images, or extracting some information so that it becomes output following the intended use.

**Quantization** in image processing refers to the process of reducing the number of distinct color or brightness levels in an image. This is usually done to reduce the amount of memory required to store or transmit an image, or to simplify the image for processing or analysis.

In digital images, each pixel has a value that represents its color or brightness. The number of distinct values that a pixel can have is determined by the number of bits used to represent it. Quantization involves mapping the original pixel values to a smaller set of values.

**Convolutional Neural Network**

For these reasons, 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 regular 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 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 the 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 | Photo: Author

**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 | Photo: Author

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.