6 DeepLearning

Our treatment of neural networks has been basically the state of the art for at least 20 or 30 years, though this area is now receiving considerable attention under the name of deep learning. Quite simply, deep learning refers to neural networks with many layers, which will of course lead to more complex models and hence it is likely that they can outperform simpler models for complex tasks. While this was already clear in the 1990s, we have only been able to train such networks in more complex tasks in the last few years. This advancement is due to several factors, but specifically though the availability of large supervised datasets, the enormous increase of computational power in particular though GPUs, and some more sophisticated applications of regularization techniques. We will briefly outline some of these issues here while basically following some of the tutorials in Tensorflow.

6.1 Convolution

Most of the success in deep neural networks have been demonstrated with image processing and involve convolutional neural networks, and this this section we will just review basic convolutions. A convolution in one dimension is defined as

$$f * g = \int_{-\infty}^{\infty} g(t')f(t-t')dt' \tag{6.1}$$

6.2 CNN

The idea of using convolutions in neural network have been first used by Fukushima in 1980. Fukushima worked at this time for NHK (the Japanese public broadcaster) together with physiologists as NHK was interested to understand the mechanisms of human vision. It was well known since the early 1960s form the experiments by Hubel and Wiesel that some neurons in the primary visual cortex (the first stage of visual processing in the cortex) are edge detectors.

Edge detectors are also the workhorse of computer vision, and we discussed in the first section how such filters are implemented with convolutions. The neural networks that we discussed before had to learn individual weights to each pixel location. Even if this network would learn to represent an edge detector, such detectors have to be learned for each location in an image since edges could usually appear in all locations. So another way of thinking about convolution is that a neuron (specific filter) is applied to every possible location in the image. This leads us to a **convolutional neural network (CNN)**.

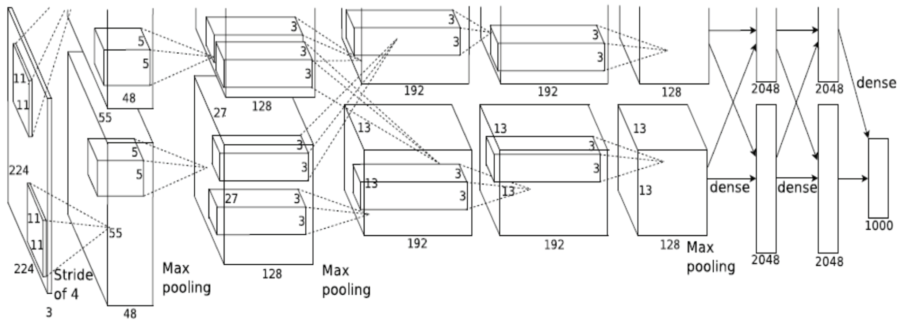

Fig. 6.1 A famous implementation (so called AlexNet) of a convolutional neural network for image classification Representation (Krizhevsky, Sutskever, Hinton, 2012).

A famous example called AlexNet is shown that was used to classify a big database of images from many different classes is shown in Fig. 6.1. This network takes three dimensional images (e.g. RGB values of pixels) and applies a layer of filters onto it. There are actually several layers of filters (so called filter banks) applied to the input image. The network consists actually os several layers of such convolutional computations. Also, to help with the computational coast of the operations we are sometimes not just shifting the filter by one pixel but might shift it by s pixels. This is called a lemphstride.

If we would only apply convolutions on convolutions, then we would end with filtered images of filtered images. However, what we really hope to achieve are high level representation such as nodes that represent class labels. Such labels are likely not to depend on individual pixels and rather represent a highly compressed summary of an image. To help with this we add **pooling layers** after each convolution layer. A pooling operation is usually just taking the average or the maximum of the responses in a certain area of the filtered image, thus compressing the images down by only considering the average or maximal feature represented by the filter in this area. At the end we use a regular network, now often called a **fully connected layer**, to gather all the information and make the final classification based on the features extracted by the network.

While we have just outlined the operation of this network, an important part of using this network is of course the training. Indeed, for our further discussion it is good to realize that the filters are not chosen by hand but are learned from examples. This training was trained with the back propagation algorithm. This is fairly straight forward except when back propagating though the pooling layers. One approach is thereby to give all credit for the error (average pooling), or juts change the winning unit (max pooling).

6.3 Tensorflow

To implement deep networks we use the Google's Tensorflow package. Please make sure it is implemented. Please follow the first tutorials

`https://www.tensorflow.org/versions/r0.11/tutorials/mnist`
followed by
`/beginners/index.html#mnist-for-ml-beginners`
for the beginner tutorial, and by
`/pros/index.html#deep-mnist-for-experts` for the expert one.