Convolutional Neural Network

### Overview

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization.

### Architecture

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r = 3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m-n+1.Each map is the subsampled typically with mean or max pooling over pxp contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger input inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. The figure below illustrates a full layer in a CNN consisting of convolutional and subsampling sublayers. Units of the same color have tied weights.

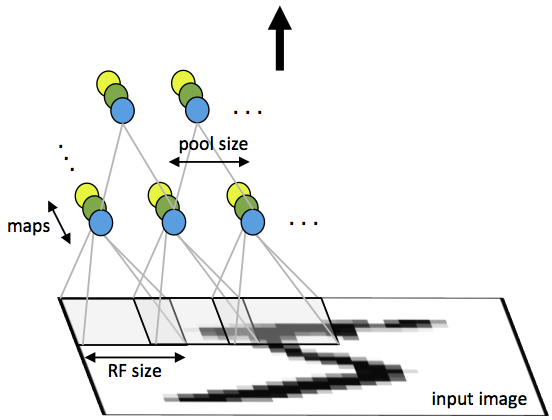


Fig1: First layer of a convolutional neural network with pooling. Units of the same color have tied weights and units of different color represent different filter maps.

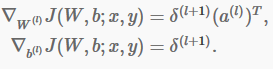
After the convolutional layers there may be any number of fully connected layers. The densely connected layers are identical to the layers in a standard multilayer neural network.

### Back Propagation

Let be the error term for the (l+1)-st layer in the network with a cost function  where  are the parameters and (x,y) are the training data and label pairs. If the l-th layer is densely connected to the (l+1)-st layer, then the error for the l-th layer is computed as



And the gradients are

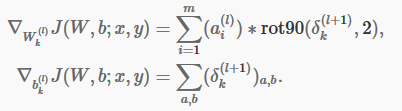


If the l-th layer is a convolutional and subsampling layer the the error is propagated through as



Where k indexes the filter number and is the derivative of the activation function. The upsample operation has to propagate the error through the pooling layer by calculating the error w.r.t to each uint incoming to the pooling layer. For example, if we have mean pooling then upsample simply uniformly distributes the error for a single pooling unit among the units which feed into it in the previous layer. In max pooling the unit which was chosen as the max receives all the error since very small changes in input would perturb the result only through that unit.

Finally, to calculate the gradient w.r.t to the filter maps , we rely on the border handling convolution operation again and filp the error matrix the same way we filp the filters in the convolutional layer.



Where is the input to the l-th layer , and is the input image. The operation is the “valid” convolution between i-th input in the l-th layer and the error w.r.t. the k-th filter.