Convolutional Neural Networks

compiled by Alvin Wan from Professor Jitendra Malik's lecture

1 Convolution

We take as input one matrix, for example a 4×4 and an array called a **filter** (a.k.a., or "convolution kernel" or "mask"). We can think of this filter, a weighting function, as the receptive field of the output node, or as the weights w_{ij}. We multiply point-wise $\sum w_{ij} x_i$. This exploits *local connectivity.*

We then shift the array by one. Note that even for different nodes, the weights stay the same, as we shift this mask along the image. This exploits *shift invariance.* We trade off the complexity of each layer with the number of layers, as each layer may have a simple set of weights.

We consider the result of the convolution kernel f_i applied to I to be $I \times f_i$.

1.1 Motivation

We consider two layers l_1 and l_2. Say we connect three pixels along a diagonal, in l_1 to one node n in l_2, where each of these edges has positive weight 10. Then, connect all surrounding pixels in l_1 to n, with negative weight 5. We note that if the three pixels are fired and its surrounding pixels do not receive input, then n is activated. If the line is any thicker, surround input may cancel center input. If the same pattern of three pixels in a diagonal is activated elsewhere, then notice that n is still not activated. In other words, n detects a specific position at a specific orientation. This is rather naive.

1.2 Feature Detection

If we instead model this "mask" as a convolution filter, we can get this output in many different locations. This family of responses is called a convolution, and this allows us to find the same pattern at different positions. As we increase the number of convolutional layers, we see that the receptive field of a node increases. This is the reason why we can associate convolutions with various high-level features of an image. In a sense, we develop a tolerance for small variances. Features may be as trivial as edges or, with sufficiently many convolutions, as as interesting as a window or hat.

2 Early Models

2.1 NeoCognitron

In 1980, Kunihiko Fukushima conceived of a "neocognitron", which we now know as a neural network. He proposed multiple layers, which correspond to the convolutions we know.

Fukushima presented a few ideas. Consider the max function of the same filter applied to different positions. We call this technique **max pooling**. This technique allows us to decrease the size of the convolution, which we call **subsampling**. For an extreme example, consider taking the maximum over all nodes in the previous layer. Every node would then have the exact same value, allowing us to reduce the entire layer down to one node.

2.2 MNIST

In 1989, Yann LeCun used back propagation to compute neural networks and demonstrated it using MNIST. LeCun alternated convolution layers with subsampling layers for a total of 6 filters. However, we have very few parameters. LeCun did not design the filters and instead allowed them to emerge from backpropagation.

3 AlexNet

AlexNet used a similar model but with fully-connected layers.