Understanding Convolutional Neural Networks for NLP

<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

http://www.wildml.com/deep-learning-glossary/

odavde uzet samo dio koji govori općenito o konvolucijskim mrežama

When we hear about Convolutional Neural Network (CNNs), we typically think of Computer Vision. CNNs were responsible for major breakthroughs in Image Classification and are the core of most Computer Vision systems today, from Facebook’s automated photo tagging to self-driving cars.

## WHAT IS CONVOLUTION

Easiest way to understand a *convolution* is by thinking of it as a sliding window function applied to a matrix.

[*http://deeplearning.stanford.edu/wiki/index.php/Feature\_extraction\_using\_convolution*](http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution)

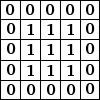


*Convolution with 3×3 Filter.*

Imagine that the matrix on the left represents a black and white image. Each entry corresponds to one pixel, 0 for black and 1 for white (typically it’s between 0 and 255 for grayscale images). The sliding window is called a *kernel,* *filter,*or*feature detector.* Here we use a 3×3 filter, multiply its values element-wise with the original matrix, then sum them up. To get the full convolution we do this for each element by sliding the filter over the whole matrix.

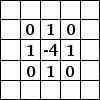
**USE-CASES:**

#### Averaging each pixel with its neighboring values blurs an image:

#### Taking the difference between a pixel and its neighbors detects edges:

(To understand this one intuitively, think about what happens in parts of the image that are smooth, where a pixel color equals that of its neighbors: The additions cancel and the resulting value is 0, or black. If there’s a sharp edge in intensity, a transition from white to black for example, you get a large difference and a resulting white value)

## WHAT ARE CONVOLUTIONAL NEURAL NETWORKS?

Now you know what convolutions are. CNNs are basically just several layers of convolutions with nonlinear activation functions like [ReLU](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)) applied to the results. In a traditional feedforward neural network we connect each input neuron to each output neuron in the next layer.

That’s also called a fully connected layer, or affine layer. In CNNs we don’t do that. Instead, we use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output.

Each layer applies different filters, typically hundreds or thousands like the ones showed above, and combines their results.

During the training phase, **a CNN** **automatically learns the values of its filters**based on the task you want to perform. For example, in Image Classification a CNN may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features.



There are two aspects of this computation worth paying attention to: **Location Invariance** and **Compositionality**.

 Let’s say you want to classify whether or not there’s an elephant in an image. Because you are sliding your filters over the whole image you don’t really care *where*the elephant occurs. In practice,  *pooling* also gives you invariance to translation, rotation and scaling.

compositionality

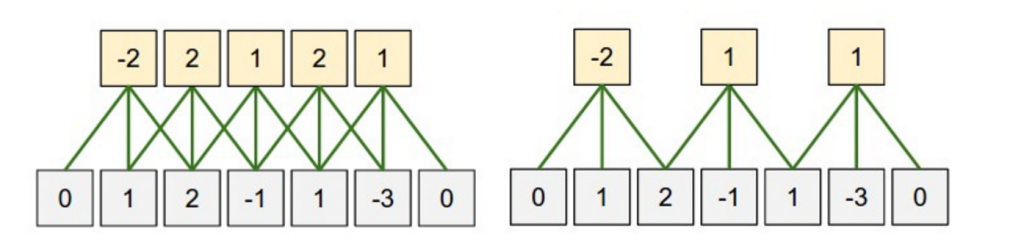
The second key aspect is (local) compositionality. Each filter *composes* a local patch of lower-level features into higher-level representation. That’s why CNNs are so powerful in Computer Vision. It makes intuitive sense that you build edges from pixels, shapes from edges, and more complex objects from shapes.

### NARROW VS WILD CONVOLUTION

Samo pročitano

### STRIDE SIZE

Another hyperparameter for your convolutions is the stride size, defining by how much you want to shift your filter at each step.  In all the examples above the stride size was 1, and consecutive applications of the filter overlapped. A larger stride size leads to fewer applications of the filter and a smaller output size. The following from the [Stanford cs231 website](http://cs231n.github.io/convolutional-networks/) shows stride sizes of 1 and 2 applied to a one-dimensional input:

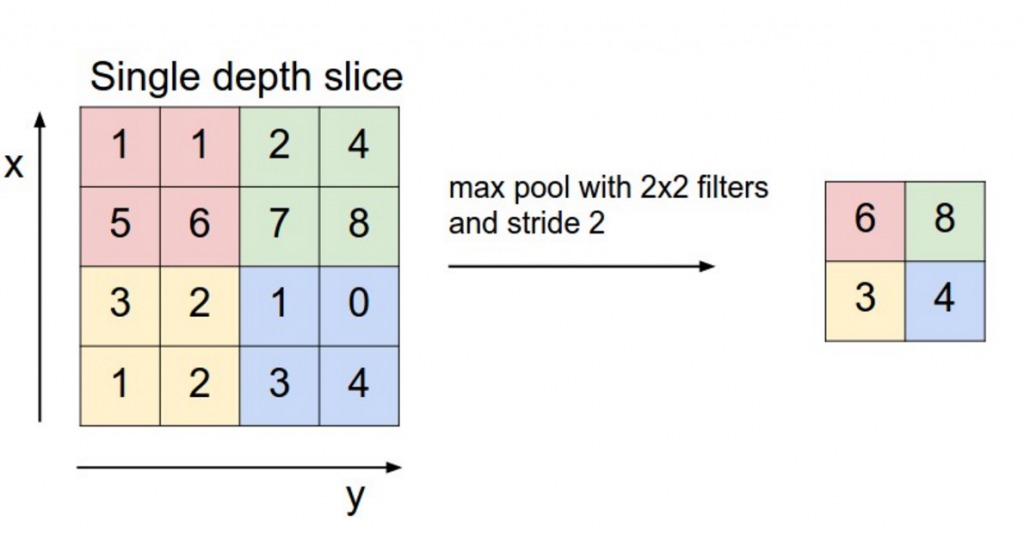


*Convolution Stride Size. Left: Stride size 1. Right: Stride size 2.*

[*http://cs231n.github.io/convolutional-networks/*](http://cs231n.github.io/convolutional-networks/)

### POOLING LAYERS

A key aspect of Convolutional Neural Networks are *pooling layers,* typically applied after the convolutional layers. Pooling layers subsample their input. The most common way to do pooling it to apply a max operation to the result of each filter. You don’t necessarily need to pool over the complete matrix, you could also pool over a window. For example, the following shows max pooling for a 2×2 window:



*Max pooling in CNN. Source:*[*http://cs231n.github.io/convolutional-networks/#pool*](http://cs231n.github.io/convolutional-networks/#pool)

**REASONS FOR POOLING**

One property of pooling is that it provides a fixed size output matrix, which typically is required for classification. For example, if you have 1,000 filters and you apply max pooling to each, you will get a 1000-dimensional output, regardless of the size of your filters, or the size of your input. This allows you to use variable size sentences, and variable size filters, but always get the same output dimensions to feed into a classifier.

Pooling also reduces the output dimensionality but (hopefully) keeps the most salient information

In imagine recognition, pooling also provides basic invariance to translating (shifting) and rotation. When you are pooling over a region, the output will stay approximately the same even if you shift/rotate the image by a few pixels, because the max operations will pick out the same value regardless.

### CHANNELS

The last concept we need to understand are *channels*.Channels are different “views” of your input data. For example, in image recognition you typically have RGB (red, green, blue) channels. You can apply convolutions across channels, either with different or equal weights.