DATA SCIENCE II: Machine Learning MTH 9899 Baruch College

Lecture 7: Convolutional Neural Networks

Adrian Sisser

May 17, 2017

Outline

Outline

CNNs

Convolution Neural Networks, or CNNs/ConvNets, are a NN architecture who's most obvious application is to images. In a nutshell, this is what makes image recognition work.

- While the applications of CNNs to finance are less clear, they are a very interesting area of research.
- Just this week, there was a paper published on using CNNs to predict timeseries.

Image Representation

- Images are represented as a $M \times N \times D$ matrix
- D is typically 3 and represents the color depth of the image
- For simplicity's sake, we'll assume that D is one and we're dealing with grayscale images.

One of the main reasons CNNs are so successful is that they massively reduce the number of weights we need to learn by only being interested in locally connected regions. We will take a set of small input filters, f, and apply them over all regions of the image to get a hidden layer. We convolve the filter with the underlying input by doing element-wise multiplication of filter with x and taking the sum of the elements.

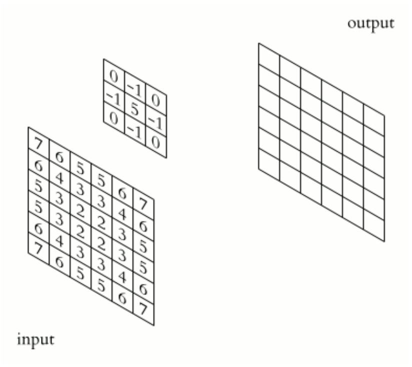
The weights for the filter are shared across the entire underlying image.

Let's assume:

- Input matrix, X is $N \times F$
- Filter, f, is $(2C+1) \times (2C+1)$, we'll make it square to make our lives easier.
- We will index f in a strange way, where the 'center' is $(0,0)$ and can be indexed from $(-C,-C)$ to (C,C)
- Assume that $X_{ij} = 0$ if $i < 0$ or $i >= N$ or $j < 0$ or $j >= F$.
- Our output matrix, O, will be the same shape as X

$$o_{i,j} = \sum_{r=-C \ldots C, s=-C \ldots C} f_{r,s} x_{i+r,j+s}$$

We can extend this by moving the filter more than 1 unit at a time which we'll call the stride.



[1] Source: https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif

The filters become "feature detectors". For example, you can have a filter like:

$$f = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Horizontal Line Detector

$$f = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

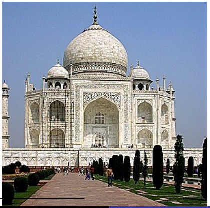
Vertical Line Detector

$$f = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Right Angle Detector

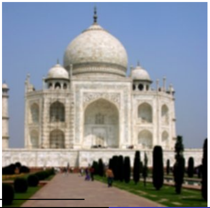
Original

Sharpen $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$



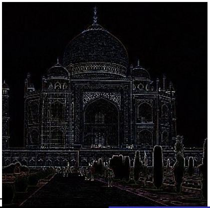
Original

Blur $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



Original

Edge Detection $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$



So why are filters interesting?

- Because of the way we apply them, they are translation invariant
- We learn the weights of the filter, we don't start off with the types of filters mentioned before
- We can train many filters, and by starting them off with different random weights, we will end up with different features

Outline

So what next? After we've calculated the convolution output, we have to apply a non-linearity, just like a normal NN. We typically choose ReLU:

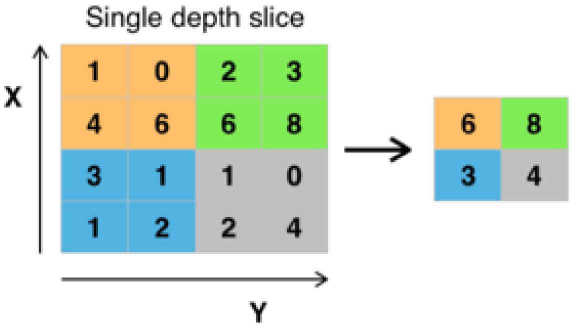
$$O' = \mathcal{A}(O)$$

All of the usual reasons apply for why we have to do something non-linear and why ReLU is a good choice.

Outline

Now that we have a non-linear version of the convolved layer, we apply a new layer type - pooling.

- The idea of max pooling is to take to define a rectangular pool size and tile it across the feature map.
- Then we, take all of the elements in that pool and apply a simple operation to them, such as max or mean.
- In practice, max works best.
- The big value of max pooling is that we can shrink the size of our inputs between layers
- While we shrink the layers, we are extracting the 'largest' features.



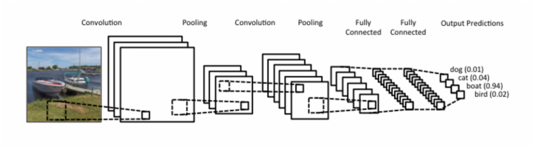
[1] Source: https://commons.wikimedia.org/wiki/File:Max_pooling.png

Outline

We have now introduced 2 new layers, and talked about one we already know about:

- Convolutional Layer
- ReLU
- Pooling Layer

These layers have been able to extract the essential features of an image. Now, we can put them together with a smaller, fully connected network to do image classification.

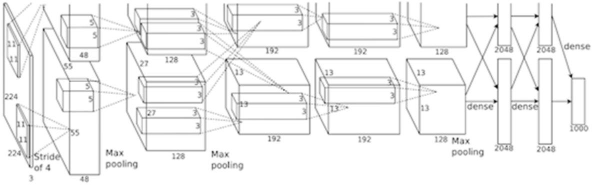


[1]Source: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

AlexNet

AlexNet was a deep CNN developed by Alex Krizhevsky that won the ILSVRC (image classification) contest in 2012. The net consisted of:

- 5 convolutional layers
- 3 fully connected layers
- The first layer had 96 filters, and they were $11 \times 11 \times 3$ with a 'stride' of 4. This was followed by max pooling.
- The second layer has 256 filters, also 11×11. This was followed by max pooling.
- The 3rd, 4th, and 5th layers were 3×3 convolutional layers, with no max pooling in between, just at the end.
- This was followed by 3 fully connected layers
- Dropout was also used.



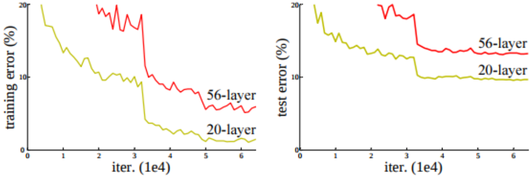
AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

[1] Source: http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghe

ResNet

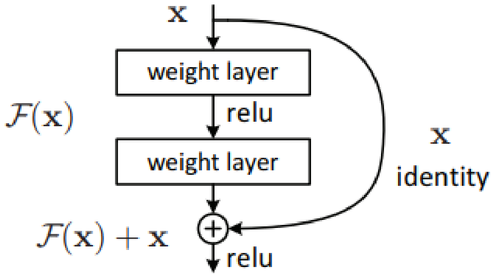
Residual Networks, aka ResNet, are a new type of network introduced in 2015 by He, et al at Microsoft Research. The underlying question is, why can't we train very deep networks?

- We know that a deep network can be much narrower yet have more explanatory power
- Better activation functions allows deeper networks, ie ReLu.
- Batch Normalization allows deeper networks.



One very interesting point - a deeper network hurts *training error* as well as test error. Why?

[1] Source: Deep Residual Learning for Image Recognition; 2015; He, Zhang, Ren, Sun


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

- ResNet adds in a 'skip' layer.
- Instead of learning $\mathcal{H}(x)$, we learn $\mathcal{F}(x) = \mathcal{H}(x) - x$. This is why it's referred to as a "Residual Network".
- In a degenerate case, if the new layer adds no value, we'll simply set it's weights to 0.

[1] Source: Deep Residual Learning for Image Recognition; 2015; He, Zhang, Ren, Sun

So does it work? **YES.**

- ResNets built a 152-layer deep NN that won the 2015 ImageNet Large Scale Visual Recognition Challenge.
- Successful networks on the order of thousands of layers can be built with this.
- This technique effectively removes the depth constrains on NNs.