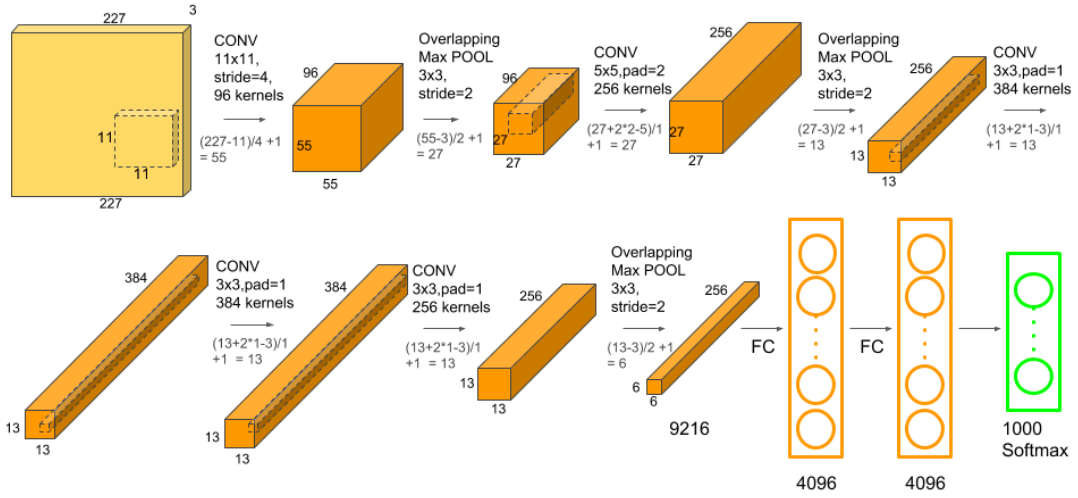
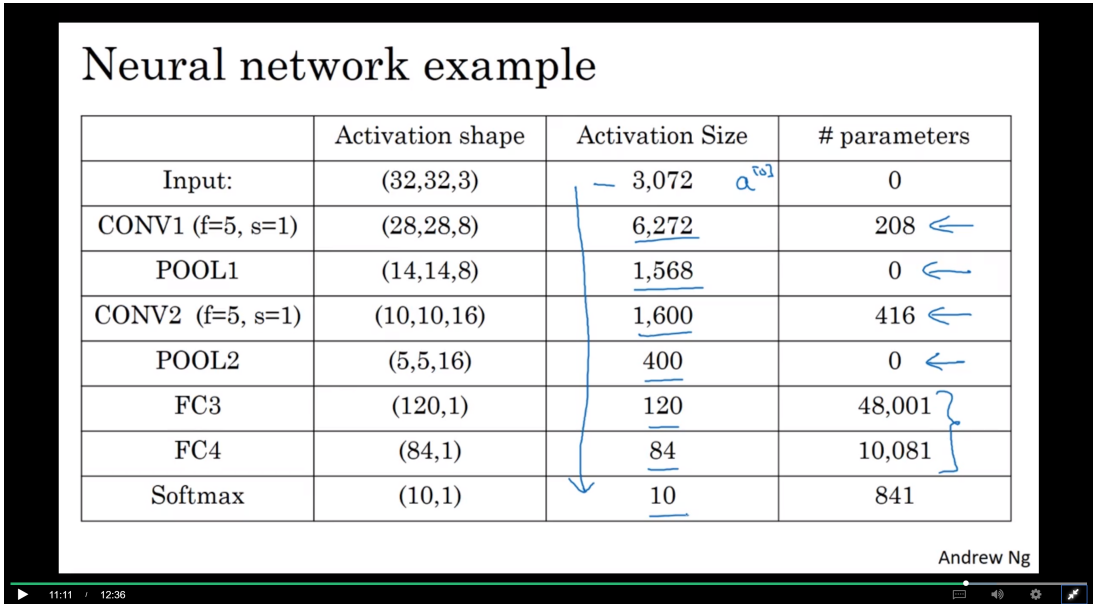
Understanding and Calculating the number of Parameters in Convolution Neural Networks (CNNs)

Rakshith Vasudev Follow
Feb 11 · 5 min read



https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png



https://www.coursera.org/learn/convolutional-neural-networks/lecture/uRYL1/cnn-example

If you've been playing with CNN's it is common to encounter a summary of parameters as seen in the above image. We all know it is easy to calculate the activation size, considering it's merely the product of width, height and the number of channels in that layer.

For example, as shown in the above image from coursera, the input layer's shape is (32, 32, 3), the activation size of that layer is 32 * 32 * 3 = 3072. The same holds good if you want to calculate the activation shape of any other layer. Say, we want to calculate the activation size for CONV2. All we have to do is just multiply (10,10,16) , i.e 10*10*16 = 1600, and you're a done calculating the activation size.

However, what sometimes may get tricky, is the approach to calculate the number of parameters in a given layer. With that said, here are some simple ideas to keep in my mind to do the same.

Some context (Skip this if you know the meaning of the term "parameter" in our context):

Let me ask you this question : How does a CNN learn?

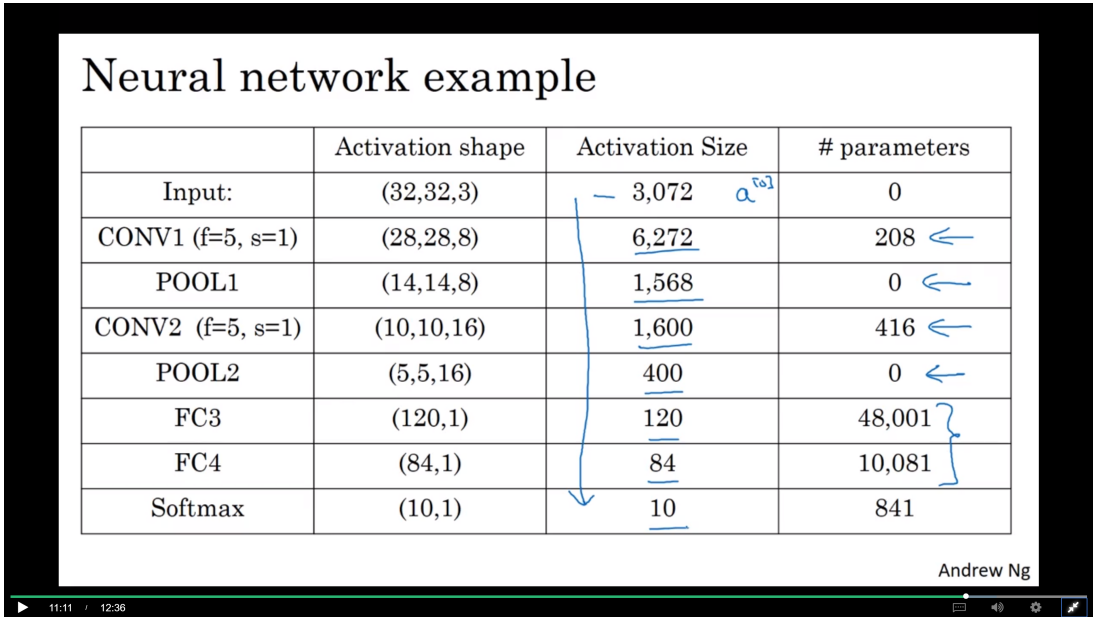
This goes back to the idea of understanding what we are doing with a convolution neural net, which is basically trying to learn the values of filter(s) using backprop. In other words, if a layer has weight matrices, that is a "learnable" layer.

Basically, the number of parameters in a given layer is the count of "learnable" (assuming such a word exists) elements for a filter aka parameters for the filter for that layer.

Say suppose you have 5, 4*4 filters in a layer, how many parameters can you learn in that layer?

You have 4*4 parameters for each filter and you have 5 such filters. Thus, total parameters you could learn would be 4*4*5 = 80 parameters, for that entire layer.

Now that you know what “parameters” are, let’s dive into calculating the number of parameters in the sample image we saw above. But, I’d want to include that image again here to avoid your scrolling effort and time.



https://www.coursera.org/learn/convolutional-neural-networks/lecture/uRYL1/cnn-example

1. **Input layer:** Input layer has nothing to learn, at it’s core, what it does is just provide the input image’s shape. So no learnable parameters here. Thus number of parameters = 0.
2. **CONV layer:** This is where CNN learns, so certainly we’ll have weight matrices. To calculate the learnable parameters here, all we have to do is just multiply the by the shape of width m, height n and account for all such filters k. Don’t forget the bias term for each of the filter. Number of parameters in a CONV layer would be : **((m * n)+1)*k)**, added 1

because of the bias term for each filter. The same expression can be written as follows: **((shape of width of the filter*shape of height of the filter+1)*number of filters).**

3. **POOL layer:** This has got no learnable parameters because all it does is calculate a specific number, no backprop learning involved! Thus number of parameters = 0.
4. **Fully Connected Layer (FC):** This certainly has learnable parameters, matter of fact, in comparison to the other layers, this category of layers has the highest number of parameters, why? because, every neuron is connected to every other neuron! So, how to calculate the number of parameters here? You probably know, take the product of the number of neurons in the current layer and the number of neurons on the previous layer. Thus number of parameters here : **((current layer n*previous layer n)+1)**. As always, do not forget the bias term!

Now let's follow these pointers and calculate the number of parameters, shall we?

Remember the drill? We don't want to scroll, do we?

https://www.coursera.org/learn/convolutional-neural-networks/lecture/uRYL1/cnn-example

1. The first **input layer** has no parameters. You know why.
2. The second **CONV1(filter shape =5*5, stride=1) layer** has how many parameters? Let's do the math according to the formula: it is **((shape of width of filter*shape of height filter+1)*number of filters) = (((5*5)+1)*8) = 208.**
3. The third **POOL1 layer** has no parameters. You know why.
4. The fourth **CONV2(filter shape =5*5, stride=1) layer** has how many parameters? Let's do the math according to the formula: it is **((shape of width of filter*shape of height filter+1)*number of filters) = (((5*5)+1)*16) = 416.**

5. The fifth **POOL2 layer** has no parameters. You know why.

6. The Sixth **FC3 layer has ((current layer n*previous layer n)+1) parameters = 120*400+1 = 48,001.**

7. The Seventh **FC4 layer has ((current layer n*previous layer n)+1) parameters = 84*120+1 = 10,081.**

8. The Eighth **Softmax layer has ((current layer n*previous layer n)+1) parameters = 10*84+1 = 841.**

FYI:

1. I've used the term "layer" very loosely to explain the separation. Ideally, CONV + Pooling is termed as a layer.

2. Just because there are no parameters in the pooling layer, it does not imply that pooling has no role in backprop. Pooling layer is responsible for passing on the values to the next and previous layers during forward and backward propagation respectively.

In this article we saw what a parameter in CNN means, we saw how to calculate the activation size, also we understood how to calculate the number of parameters in a CNN.

If you have the bandwidth, I'd recommend you to consume Dr. Andrew's Coursera course linked below.

Sources:

How to calculate the number of parameters for convolutional neural network?

I'm using Lasagne to create a CNN for the MNIST dataset. I'm

stackoverflow.com

Convolutional Neural Networks | Coursera

Convolutional Neural Networks from deeplearning.ai. This course will teach you how to build convolutional neural...

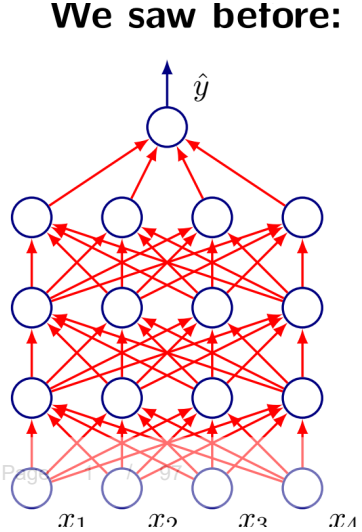
www.coursera.org

Lecture 7
Convolutional Neural Networks
CMSC 35246: Deep Learning

Shubhendu Trivedi
&
Risi Kondor

University of Chicago

April 17, 2017



If you liked this article, then clap it up! :) Maybe a follow?

Connect with me on Linkedin :

https://www.linkedin.com/in/rakshith-vasudev/

Machine Learning Convolutional Network Deep Learning Convolution Neural Net Andrew Ng