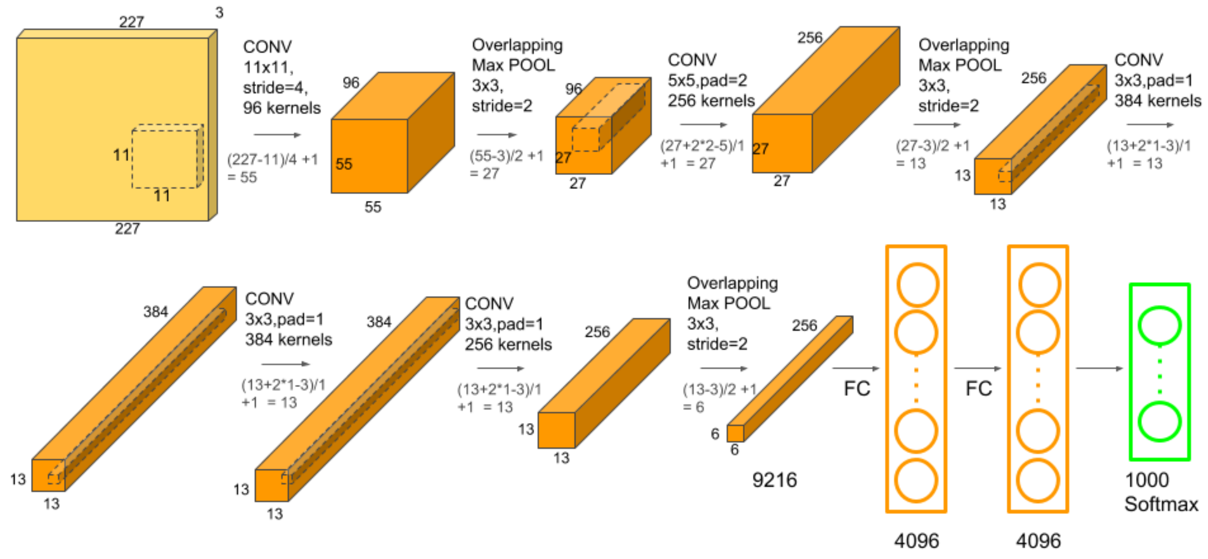
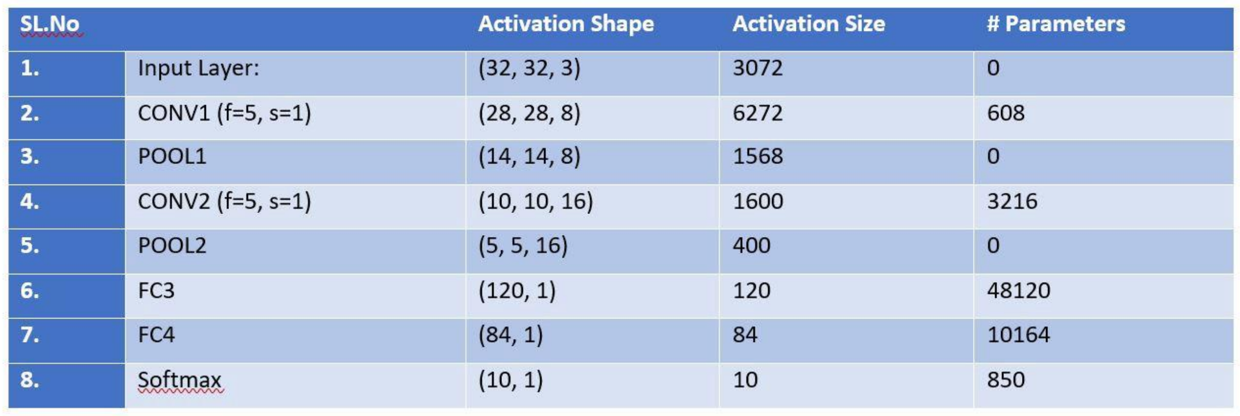
# Understanding and calculating the number of Parameters in Convolution Neural Networks (CNNs)



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



| SL.No | | Activation Shape | Activation Size | # Parameters |
|---|---|---|---|---|
| 1. | Input Layer: | (32, 32, 3) | 3072 | 0 |
| 2. | CONV1 (f=5, s=1) | (28, 28, 8) | 6272 | 608 |
| 3. | POOL1 | (14, 14, 8) | 1568 | 0 |
| 4. | CONV2 (f=5, s=1) | (10, 10, 16) | 1600 | 3216 |
| 5. | POOL2 | (5, 5, 16) | 400 | 0 |
| 6. | FC3 | (120, 1) | 120 | 48120 |
| 7. | FC4 | (84, 1) | 84 | 10164 |
| 8. | Softmax | (10, 1) | 10 | 850 |

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

**FYI: The above image does not represent correct number of parameters. Please refer to the section titled "CORRECTION". You may skip to that section if you just want the numbers.**

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 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):**

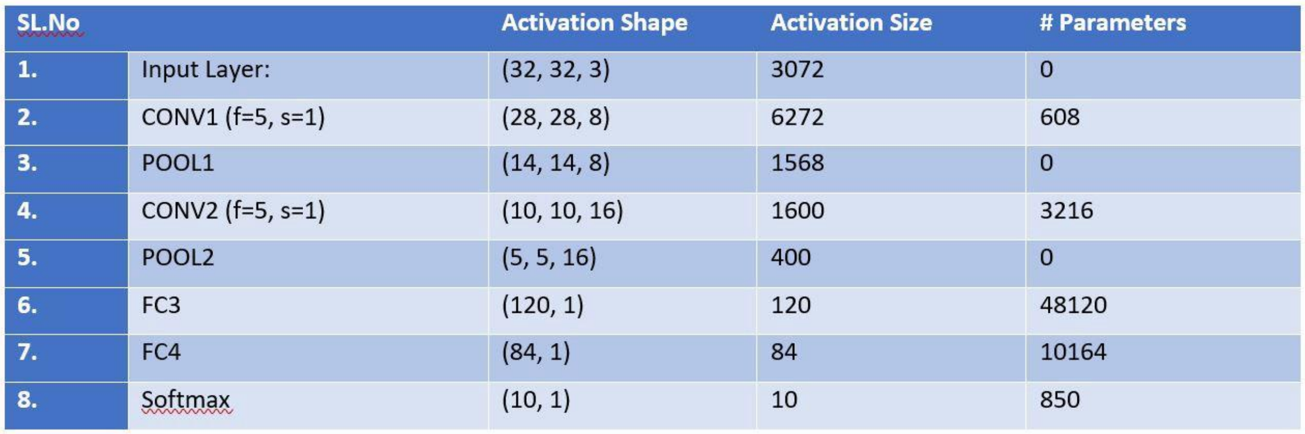
**Let me ask you this question : How does a CNN learn?**

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.

Parameters in general are weights that are learnt during training. They are weight matrices that contribute to model’s predictive power, changed during back-propagation process. Who governs the change? Well, the training algorithm you choose, particularly the optimization strategy makes them change their values.

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.



| SL.No | | Activation Shape | Activation Size | # Parameters |
|---|---|---|---|---|
| 1. | Input Layer: | (32, 32, 3) | 3072 | 0 |
| 2. | CONV1 (f=5, s=1) | (28, 28, 8) | 6272 | 608 |
| 3. | POOL1 | (14, 14, 8) | 1568 | 0 |
| 4. | CONV2 (f=5, s=1) | (10, 10, 16) | 1600 | 3216 |
| 5. | POOL2 | (5, 5, 16) | 400 | 0 |
| 6. | FC3 | (120, 1) | 120 | 48120 |
| 7. | FC4 | (84, 1) | 84 | 10164 |
| 8. | Softmax | (10, 1) | 10 | 850 |

Example taken from
Coursera: 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**.

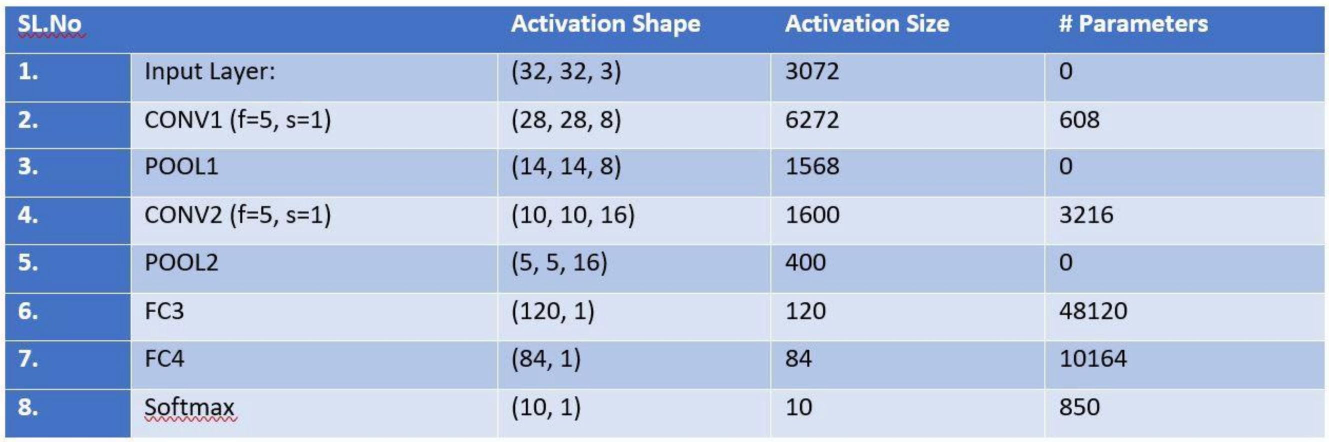
2. **CONV layer:** This is where CNN learns, so certainly we'll have weight matrices. To calculate the learnable parameters here, all we have to do is just multiply the by the shape of **width m**, **height n**, **previous layer's filters d** and account for all such filters **k in the current layer**. Don't forget the bias term for each of the filter. Number of parameters in a CONV layer would be : **((m * n * d)+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 * number of filters in the previous layer+1)*number of filters).** Where the term "filter" refer to the number of filters in the current layer.

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, it is the product of the number of neurons in the current layer **c** and the number of neurons on the previous layer **p** and as always, do not forget the bias term. Thus number of parameters here are: **((current layer neurons c * previous layer neurons p)+1*c)**.

**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?**



| SL.No | | Activation Shape | Activation Size | # Parameters |
|---|---|---|---|---|
| 1. | Input Layer: | (32, 32, 3) | 3072 | 0 |
| 2. | CONV1 (f=5, s=1) | (28, 28, 8) | 6272 | 608 |
| 3. | POOL1 | (14, 14, 8) | 1568 | 0 |
| 4. | CONV2 (f=5, s=1) | (10, 10, 16) | 1600 | 3216 |
| 5. | POOL2 | (5, 5, 16) | 400 | 0 |
| 6. | FC3 | (120, 1) | 120 | 48120 |
| 7. | FC4 | (84, 1) | 84 | 10164 |
| 8. | Softmax | (10, 1) | 10 | 850 |

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

1. The first **input layer** has no parameters. You know why.
2. Parameters in the second **CONV1(filter shape =5*5, stride=1) layer is: ((shape of width of filter*shape of height filter*number of filters in the previous layer+1)*number of filters) = (((5*5*3)+1)*8) = 608.**
3. The third **POOL1 layer** has no parameters. You know why.
4. Parameters in the fourth **CONV2(filter shape =5*5, stride=1) layer is**: **((shape of width of filter * shape of height filter * number of filters in the previous layer+1) * number of filters) = (((5*5*8)+1)*16) = 3216.**

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

6. Parameters in the Sixth **FC3 layer is((current layer c*previous layer p)+1*c) = 120*400+1*120= 48120.**

7. Parameters in the Seventh **FC4 layer is: ((current layer c*previous layer p)+1*c) = 84*120+1* 84 = 10164.**

8. The Eighth **Softmax layer has ((current layer c*previous layer p)+1*c) parameters = 10*84+1*10 = 850.**

Update V2:

Thanks for the comments by observant readers. Appreciate the corrections. Changed the image for better understanding.

**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 means, we saw how to calculate the activation size, also we understood how to calculate the number of parameters in a CNN.