What does 1x1 convolution mean in a neural network?

I am currently doing the Udacity Deep Learning Tutorial. In Lesson 3, they talk about a 1x1 convolution. This 1x1 convolution is used in Google Inception Module. I'm having trouble understanding what is a 1x1 convolution.

I have also seen this post by Yann Lecun.

Could someone kindly explain this to me?

neural-networks | deep-learning | convolution | conv-neural-network

edited Feb 7 '16 at 17:11
gung ♦
93.8k 26 216 452

asked Feb 5 '16 at 3:33
jkschin
363 1 4 6

Also see a related question – gkcn Nov 1 '16 at 10:57

Here's a blog post on these modules that went into detail on the 1x1 convolutions: hackathonprojects.wordpress.com/2016/09/25/… – Tommy Nov 15 '16 at 6:46

3 Answers

Suppose that I have a conv layer which outputs an (N, F, H, W) shaped tensor where:

- N is the batch size
- F is the number of convolutional filters
- H, W are the spatial dimensions

Suppose this output is fed into a conv layer with F_1 1x1 filters, zero padding and stride 1. Then the output of this 1x1 conv layer will have shape (N, F_1, H, W).

So 1x1 conv filters can be used to change the dimensionality in the filter space. If $F_1 > F$ then we are increasing dimensionality, if $F_1 < F$ we are decreasing dimensionality, in the filter dimension.

Indeed, in the Google Inception article Going Deeper with Convolutions, they state (bold is mine, not by original authors):

> One big problem with the above modules, at least in this naive form, is that even a modest number of 5x5 convolutions can be prohibitively expensive on top of a convolutional layer with a large number of filters.
>
> This leads to the second idea of the proposed architecture: judiciously applying dimension reductions and projections wherever the computational requirements would increase too much otherwise. This is based on the success of embeddings: even low dimensional embeddings might contain a lot of information about a relatively large image patch...**1x1 convolutions are used to compute reductions before the expensive 3x3 and 5x5 convolutions.** Besides being used as reductions, they also include the use of rectified linear activation which makes them dual-purpose.

So in the Inception architecture, we use the 1x1 convolutional filters to reduce dimensionality in the filter dimension. As I explained above, these 1x1 conv layers can be used in general to change the filter space dimensionality (either increase or decrease) and in the Inception architecture we see how effective these 1x1 filters can be for dimensionality reduction, explicitly in the filter dimension space, not the spatial dimension space.

Perhaps there are other interpretations of 1x1 conv filters, but I prefer this explanation, especially in the context of the Google Inception architecture.

edited Feb 7 '16 at 16:58

answered Feb 7 '16 at 16:45
Indie AI

3,688 1 17 26

1 wonderful explanation ! – London guy May 8 '16 at 16:03

3 Is it that the 1x1 conv compresses the previous filter dimension to 1, before implementing the 5x5 conv ? – Leonard Loo Dec 13 '16 at 7:51

nice explanation. – Alwyn Mathew Mar 19 at 18:19

1 @LeonardLoo each 1x1 kernel reduces filter dimension to 1, but you can have multiple kernels in one 1x1 convolution, so the number of "filters" can be arbitrary of you choice. – Fazzolini May 19 at 8:58

A 1x1 convolution simply maps an input pixel with all it's channels to an output pixel, not looking at anything around itself. It is often used to reduce the number of depth channels, since it is often very slow to multiply volumes with extremely large depths.

```
input (256 depth) -> 1x1 convolution (64 depth) -> 4x4 convolution (256 depth)

input (256 depth) -> 4x4 convolution (256 depth)
```

The bottom one is about ~3.7x slower.

Theoretically the neural network can 'choose' which input 'colors' to look at using this, instead of brute force multiplying everything.

edited May 29 at 4:25

answered May 18 at 6:57


Free Debreuil
131 1 3

6 I would say that 1x1 maps not just one pixel to an output pixel, but it collapses all input pixel channels to one pixel. In your example in the first line, there are 256 channels for input, and each of the 64 1x1 kernels collapses all 256 input channels to just one "pixel" (real number). The result is that you have 64 channels now instead of 256 with the same spacial dimension, which makes 4x4 convolution computationally cheaper than in your second line example. – Fazzolini May 19 at 9:03

Good point, will update the post :) – Free Debreuil May 29 at 4:19

One more idea about dimensionality reduction in the context of 1x1 filters:

Take for example an 4096x8x8 fc7 layer from FCN. What happens if the next layer (call it fc8) is 2048x8x8 with filter size 1? fc7 is very deep inside the network, each of its 4096 features is semantically rich, but each neuron (e.g. input image is 250x250x3) has a large receptive field. In other words, if a neuron is very active, we know that somewhere in its semantic field there's a corresponding feature present.

Take for example a left-uppermost neuron in fc8 with a 1x1 filter. It connects to all 4096 neurons/features only in the same receptive field (upper-left corner of the image), each of which is activated by a single feature. Some (let's same 500) of them are very active. If the resulting neuron is also very active, it means it probably learnt to identify 1 or more features in this receptive field. After you've done this 2048 times for left-uppermost neurons in fc8, quite a few of them (e.g. 250) will be very active, meaning they 'collected' features from the same receptive field through fc7, and many very likely more than one.

If you keep reducing the dimensionality, a decreasing number of neurons will be learning an increasing number of features from the same receptive field. And since spatial parameters 8x8 remain the same, we do not change the 'view' of each neuron, thus do not decrease the spatial coarseness.

You may want to have a look at 'Fully Convolutional Networks' by Long, Shelhamer and Darrel.

answered Jul 30 at 3:14


Alex
175 8