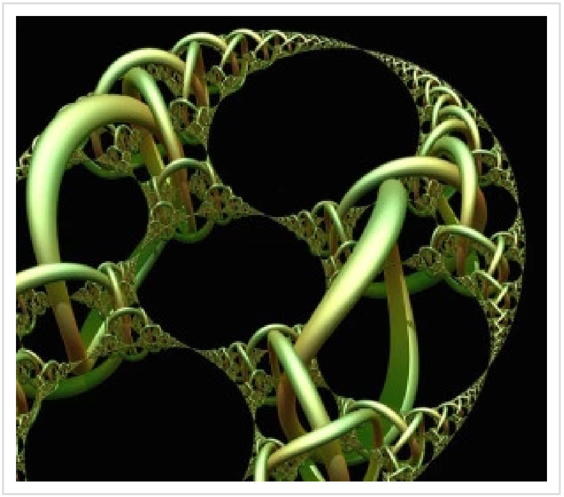
PERPETUAL ENIGMA

PERENNIAL FASCINATION WITH ALL THINGS TECH

Understanding Locally Connected Layers In Convolutional Neural Networks

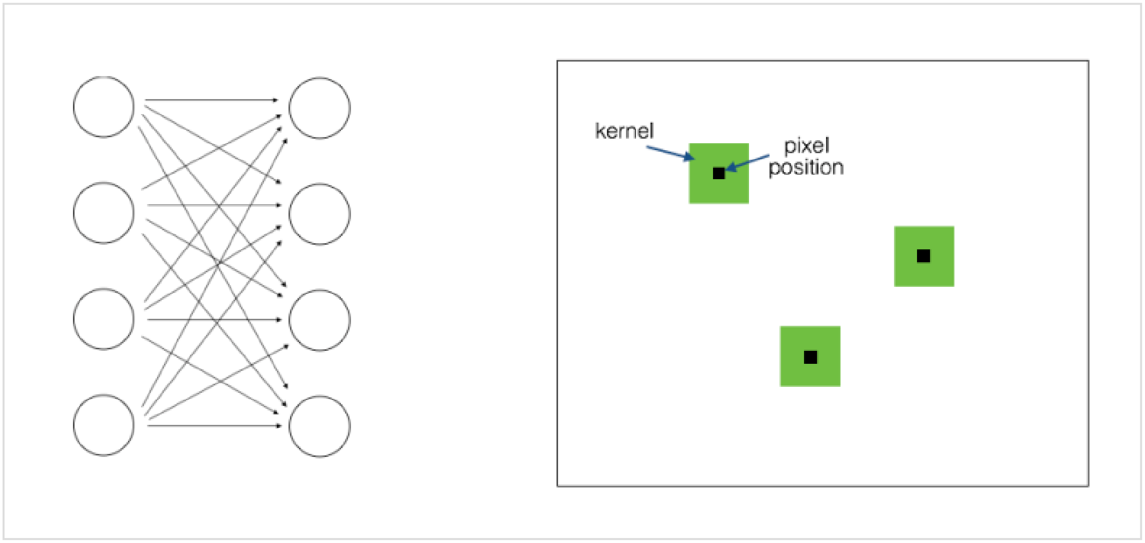
Posted on April 12, 2016



Convolutional Neural Networks (CNNs) have been phenomenal in the field of image recognition. Researchers have been focusing heavily on building deep learning models for various tasks and they just keeps getting better every year. As we know, a CNN is composed of many types of layers like convolution, pooling, fully connected, and so on. Convolutional layers are great at dealing with image data, but there are a couple of restrictions as well. The DeepFace network built by Facebook used another type of layer to speed up their training and get amazing results. This layer is called Locally Connected Layer with unshared weights. So what exactly does it do that other layers can't?

Fully connected and convolutional layers

Before we start discussing locally connected layers, we need to understand where it comes from. Back when neural networks started gaining traction, people were heavily into fully connected layers. It's basically connected all the neurons in one layer to all the neurons in the next layers. Let's see what a fully connected and convolutional layers look like:

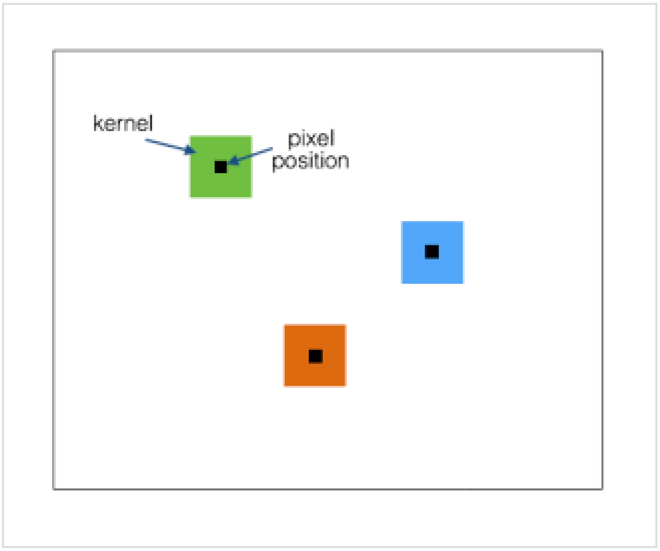


The one on the left is the fully connected layer. The figure on the right indicates convolutional layer operating on a 2D image. This gives a lot of freedom for the neural network to train and optimize all the parameters. But at the same time, it's computationally intensive! So in order to take advantage of the 2D structure of image data, researchers came up with convolutional layers.

On each 2D array of data, we train a whole bunch of N x N kernels. These kernels are nothing but filters that we run across the 2D array. For each position (x,y), we compute the dot product summation between this kernel and the image values around that point. This process is called convolution, hence the name convolutional neural networks. As we can see in the image above, the kernels will remain the same everywhere (as indicated by the green color).

We're like half way through and we still haven't talked about "locally connected" layers

Well, that's not exactly true! Convolutional layers are technically locally connected layers. To be precise, they are locally connected layers with shared weights. We run the same filter for all the (x,y) positions in the image. In other words, all the pixel positions "share" the same filter weights. We allow the network to tune the filter weights until we arrive at the desired performance. While this is great at image classification tasks, we tend to miss out on some subtle nuances of spatial arrangements.



So researchers tried out an approach where they have a different filter for each (x,y) position. There's no convolution as such! It's just a dot product at each pixel position. Let's say we want train 256 filters of size 5×5 at the current stage of the network.Here, each filter learns to detect different aspects of the image. The input image is of size, say, 128 x 128. We can fit 124 x 124 filters in that image grid (draw it on a piece of paper to verify). As seen from the figure to the left, the different colors indicate different filters. In a convolutional layer, we just need to train 5 x 5 x 256 number of parameters. But if we are dealing with locally connected layers with unshared weights, we will be dealing with 5 x 5 x 256 x 124 x 124.

That's a lot of parameters

Now you might think — Won't that result in an enormous number of hyperparameters? Because we are building a separate filter for each pixel position and there are a lot of positions in the image. The answer is yes! But they have optimized the methods and created a fast implementation on the GPU. Also, since every pixel position gets its own filter, they reduce the number of overall filters that need to be trained. The locally connected layers with unshared weights seemed to work well for the DeepFace architecture. It is yet to undergo rigorous testing to see if it can generalize well for other types of image recognition tasks.

In essence, this layer is just like a convolutional layer in the deep neural network, but without any sharing of weights. If you keep this fact aside, it's pretty much the same as a convolutional layer.

——

SHARE THIS:

Twitter | Press This | LinkedIn 2 | Facebook 5 | Google | Reddit

Reblog | Like

3 bloggers like this.

RELATED

What Is Local Response Normalization In Convolutional Neural Networks In "Machine Learning"	Deep Learning With Caffe In Python – Part I: Defining A Layer In "Machine Learning"	Deep Learning For Sequential Data – Part III: What Are Recurrent Neural Networks In "Machine Learning"

This entry was posted in **Machine Learning** and tagged **Artificial Intelligence**, **Convolutional Neural Networks** by **Prateek Joshi**. Bookmark the **permalink [https://prateekvjoshi.com/2016/04/12/understanding-locally-connected-layers-in-convolutional-neural-networks/]** .

5 THOUGHTS ON "UNDERSTANDING LOCALLY CONNECTED LAYERS IN CONVOLUTIONAL NEURAL NETWORKS"

Hossein
on **April 22, 2016 at 10:41 PM** said:

Great article
Thanks 😉

Prateek Joshi
on **April 23, 2016 at 4:41 PM** said:

Thanks Hossein! Glad you found it useful 🙂

dvigneshwer
on **April 25, 2016 at 8:08 AM** said:

Is the locally connected layer implementation in pylearn2 and cuda convnet same as deepface paper's implementation? (Theoritically they sound the same to me)

Do you know any library which implements locally connected layer?

BTW Great article 🙂

Prateek Joshi
on April 25, 2016 at 9:52 AM said:

Thanks 🙂 I think the cuda-convnet implementation comes pretty close to the original paper. Even though Caffe doesn't provide locally connected layers directly, you can implement it as shown here: https://github.com/BVLC/caffe/pull/1271. Given Facebook's success with locally connected layer, I hope all the major libraries will implement it soon.

Za
on January 24, 2017 at 6:46 PM said:

hi Prateek, what do you think the difference between local connected layer and multiple kernel(filter) and the difference between local conected layer and patch image?

I mean, different filter at different position, why not increase the number of the filters at the conv process? Too many parameters to estimated?

IF that was the reason, when we feed the network with image patches (cut different patch at different position in one image) to different single network (like we cut 2 patch of a face image, the mouth and the eyes, then feed one network with mouth patch image and one network with eyes patch images, then we esembel these two network together). Is it the same as local connected layer? Or worse than local connected layer?