Deep Learning Study Lecture Note

Lecture 5 : What is CNN?

Kim Tae Young

January 13, 2020

5.1 What is CNN?

CNN(Convolutional Neural Network) is one other model like MLP. Unlike MLP, where we use Fully Connected Layer as connection between nodes, here we use convolution layer where the name CNN comes from.

Definition 5.1.1. *A convolution of function $f, g : \mathbb{R} \to \mathbb{R}, f * g(x) = \int f(x-y)g(y)dy$. This is definition in analysis.*
A convolution of function $f, g : \mathbb{N} \to \mathbb{R}, f * g(x) = \sum_{b|n, b\in\mathbb{N}} f(b)g(\frac{n}{b})$. This is definition in number theory.
A convolution layer is defined in terms of 'filter', where filter is n dimensional tensor. Here, I will simply show 1 dimensional convolution and 2 dimensional convolution.

$$\begin{pmatrix}1\\3\\1\\4\\2\\5\end{pmatrix} * \begin{pmatrix}-1\\2\\-1\end{pmatrix} = \begin{pmatrix}1\times -1 + 3\times 2 + 1\times -1\\ 3\times -1 + 1\times 2 + 4\times -1\\ 1\times -1 + 4\times 2 + 2\times -1\\ 4\times -1 + 2\times 2 + 5\times -1\end{pmatrix} = \begin{pmatrix}4\\-5\\5\\-5\end{pmatrix}$$

$$\begin{pmatrix}1&2&3&4\\2&3&4&5\\3&4&5&6\\4&5&6&7\\5&6&7&8\end{pmatrix} * \begin{pmatrix}3&-1\\-2&1\end{pmatrix} = \begin{pmatrix}0&1&2\\1&2&3\\2&3&4\\3&4&5\end{pmatrix}$$

Usually when using convolutional layer, as you can see we shrink the size of input tensor. However we usually don't wants to have this kind of behavior, we append padding to input layer. Usual one is zero padding, which is computed as following.

$$\begin{pmatrix}1\\3\\1\\4\\2\\5\end{pmatrix} * \begin{pmatrix}-1\\2\\-1\end{pmatrix} = \begin{pmatrix}0\times -1 + 3\times -1 + 1\times 2\\ 1\times -1 + 3\times 2 + 1\times -1\\ 3\times -1 + 1\times 2 + 4\times -1\\ 1\times -1 + 4\times 2 + 2\times -1\\ 4\times -1 + 2\times 2 + 5\times -1\\ 2\times -1 + 5\times 2 + 0\times -1\end{pmatrix} = \begin{pmatrix}-1\\4\\-5\\5\\-5\\8\end{pmatrix}$$

Also, we does not apply one filter for each connection. For example, if our input is $28\times 28\times 3$ RGB pixel image, and wants our output to be $28\times 28\times 16$ tensor, we use 48 filters of size 3×3.
We can use different size of filter, also we can change stride. Meaning, if we use stride 2 instead 1 as in above,

$$\begin{pmatrix}1&2&3&4\\2&3&4&5\\3&4&5&6\\4&5&6&7\\5&6&7&8\\6&7&8&9\end{pmatrix} * \begin{pmatrix}3&-1\\-2&1\end{pmatrix} = \begin{pmatrix}0&2\\2&4\\4&6\end{pmatrix}$$

The other main layer we use in CNN is pooling layer. Since applying convolutional layer does not decrease size of tensor, pooling layer does main role as decreasing the size of tensor. The main two pooling layer is max pooling layer and average pooling layer.

$$MaxPool(\begin{pmatrix}1&2&-1&0\\3&2&0&4\\-3&5&2&-1\\0&2&-1&4\end{pmatrix},2,2) = \begin{pmatrix}3&4\\5&4\end{pmatrix}$$

$$AveragePool(\begin{pmatrix}1&2&-1&0\\3&2&0&4\\-3&5&2&-1\\0&2&-1&4\end{pmatrix},2,2) = \begin{pmatrix}2&0.75\\1&1\end{pmatrix}$$

Again, the backpropagation of neural network can be computed since convolution layer is just simple multiplication. However, you can see that backpropagation is also a convolution.
Since applying convolutional layer only outputs n dimensional tensor, not vector(which is 1 dimensional tensor), we need final fully connected layer to output. The main structure of CNN is as following.

$$IN \to [[CONV \to ACT]^N \to POOL]^M \to [FC \to ACT]^K \to FC \to OUT$$

Followings are instance of VGGnet, which is the runner up network at ILSVRC2014.

$$\begin{aligned}&VGG11[64, M, 128, M, 256^2, M, 512^2, M, 512^2, M]\\&VGG13[64^2, M, 128^2, M, 256^2, M, 512^2, M, 512^2, M]\\&VGG16[64^2, M, 128^2, M, 256^3, M, 512^3, M, 512^3, M]\\&VGG19[64^2, M, 128^2, M, 256^4, M, 512^4, M, 512^4, M]\end{aligned}$$

Here, number means we will have that number of output dimension. M means max pooling layer, and exponent means applying that layer n times. Every filter is 3×3 filter with 1 stride and zero padding, and every max pooling is of size2×2.
The neuroscientific narrative of CNN is the sensory system, especially the vision. As in CNN, in vision system, each neurons are locally connected, not fully connected as MLP. And this explains why CNN is better, we only requires local informations in each layer in vision problem however MLP fully connects.

5.2 Methods in CNN

So in VGGnet, we only used 3×3 filters. Consider compressing 5×5 information to one scalar. If we use 5×5 filter, we require 25 paramters. However if we use two 3×3 filter, we only require 18 paramters however having equal input size. This makes us to increase performance of network with less number of parameter, meaning we require less time to train.
Similar to MLP, we can both use batch normalization and dropout in CNN. However, it is experimentally proved that normalization does not work in CNN, so normalization is not used well.

5.3 Inception

Inception is module used in advanced CNN. It was first introduced in GoogLeNet, which won first place at ILSVRC2014. As we talked right before, using 3×3 filter was better then using larger filter, However, Inception uses both, as parallel layer. So Inception is computed as following, where $\circ$ is concatenation.

$$\begin{aligned} NaiveInception(in) =& Conv(in, 1, 1, 128) \\ & \circ Conv(in, 3, 3, 192) \\ & \circ Conv(in, 5, 5, 96) \\ & \circ MaxPool(3, 3) \end{aligned}$$

$$\begin{tikzcd}
 & & \substack{Input \\ (28\times28\times256)} \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
\substack{1\times1Conv \\ 128} \arrow[drr] & \substack{3\times3Conv \\ 192} \arrow[dr] & & \substack{5\times5Conv \\ 96} \arrow[dl] & 3 \times 3 MaxPool \arrow[dll] \\
 & & \substack{Concat \\ (28\times28\times672)} & &
\end{tikzcd}$$

Figure 5.1: Naive Inception

Now consider number of parameter here. Assume input is size $28 \times 28 \times 256$, and we have 128 1×1 filters, 192 3×3 filters, 96 5×5 filters. Then we have $28 \times 28 \times 256 \times (1 \times 1 \times 128 + 3 \times 3 \times 194 + 5 \times 5 \times 96)$ convolutional operation, which is approximately 854 million operations. So instead, we append 1×1 Conv to reduce depth of input.

$$\begin{aligned}Inception(in) = &Conv(in, 1, 1, 128)\\ &\circ Conv(Conv(in, 1, 1, 64), 3, 3, 192)\\ &\circ Conv(Conv(in, 1, 1, 64), 5, 5, 96)\\ &\circ Conv(MaxPool(in, 3, 3), 1, 1, 64)\end{aligned}$$

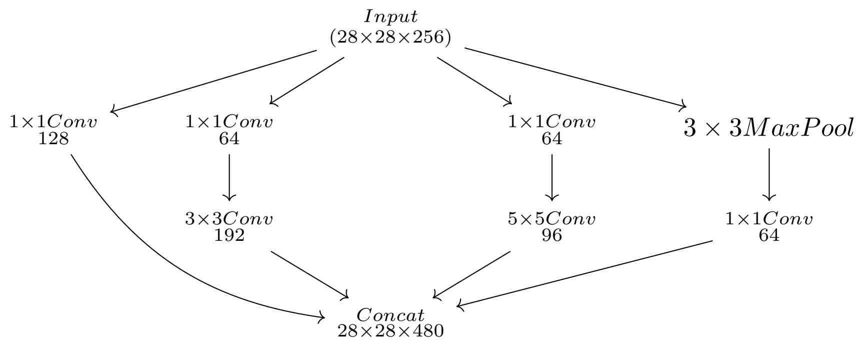

Figure 5.2: Inception

5.4 Auxiliary classifiers

Since ResNet has large depth, back propagating gradient is a main concern. The motivation here was that as shallow networks also works, middle of the ResNet would be also discriminative enough. Then by adding auxiliary classifiers in middle of networks, it encourage discrimination in earlier stages, increase gradient signal. The auxiliary classifiers are of the form small convolutional networks, and their loss is added to total loss with discount weight.

5.5 Residual Connection

Residual Connection is another module used in advanced CNN. It was first introduced in ResNet, which won first place at ILSVRC2015, in all classification. The motivation of ResNet was simple, as we saw in MLP, they increased depth. However as deeper the network is, since gradient vanishing occurs, it is hard to train. To overcome such phenomena, ResNet used residual connection.

$$Res(in) = Conv(ReLU(Conv(in))) + in$$

$$\begin{tikzcd}
input \arrow[r] \arrow[rrr, bend left] & Conv \arrow[r, "ReLU"] & Conv \arrow[r] & output
\end{tikzcd}$$

Figure 5.3: Residual Connection

Since the network tries to learn residual, not the full function, it is easier to train. In deeper network, we use bottleneck layer, which is similar to residual connection

$$Bottleneck(in) = Conv(Conv(Conv(in, 1, 1, 64), 3, 3, 64), 1, 1, 256) + in$$

$$\begin{tikzcd}
\substack{input \\ 28\times28\times256} \arrow[r] \arrow[rrrr, bend left] & \substack{1\times1 Conv \\ 64} \arrow[r] & \substack{3\times3 Conv \\ 64} \arrow[r] & \substack{1\times1 Conv \\ 256} \arrow[r] & output
\end{tikzcd}$$

Figure 5.4: Bottleneck