Deep Learning Chapter 9 CNN

Jee Dong Jun

23.4.2020

What is CNN?

- Convolutional neural network
- Specialized at processing grid like data
- Convolution layer and pooling layer

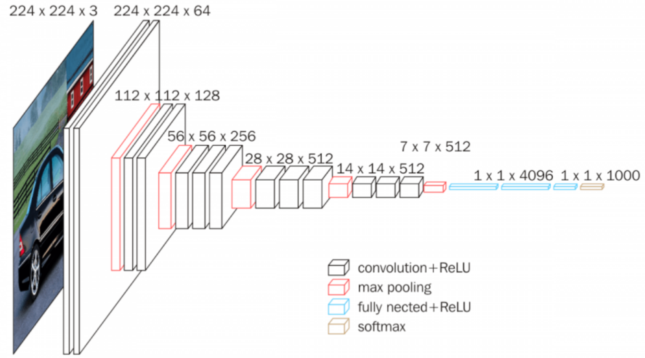

Figure 1: Vggnet

Input Data

- Audio data, volumetric data such as CT scans, colour image data and colour video data
- Mainly we will discuss 2D image data
- 2D image data is actually represented by 3D tensor. Two axis are for horizontal axis and vertical axis. One axis is for colour channel
- For grayscale image one channel is enough but for coloured image we need 3. Each of them is represented by 8bit integer.

Convolution operation

- From two functions x, w convolution operation is defined as $s(t) = \int_{-\infty}^{+\infty} x(a)w(t-a)da$
- Note this operation is commutative (use chain rule)

Using 2D image I as input with two-dimensional kernel

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- We have flipped the kernel relative to the input.

Cross-correlation

- Same as convolution without flipping the kernel

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

- For convolution we are learning kernel. Therefore, convolution or cross-correlation is equivalent.
- We can view discrete convolution as multiplication by matrix. (sparse matrix)

3 ideas of convolution

- Sparse interactions
- In fully connected layer, every input is connected to every output.
- By making the kernel smaller size than input, we can achieve sparse interactions.
- we need to store only k*n parameters and O(k*n) operations

3 ideas of convolution

- Parameter sharing
- Note kernel size is usually much smaller than the input image size
- Instead of using parameter only once in fully connected layer, we reuse same kernel for all the location of images.
- We further reduce parameter size to k which we reuse over and over.

3 ideas of convolution

- Equivalence to translation
- If the input changes, output changes in same way.

$$f(g(x)) = g(f(x))$$

- Convolution is equivalent to translation

Pooling

- Typical layer of CNN consists of 3 stages.
- Stage 1 is convolution
- Stage 2 is activation function (like RELu). Also, called detector stage.
- Third stage, we use pooling function.
- Pooling function replaces the output with summary statistics of neighbourhood.

Pooling

- Max Pooling chooses the maximum output within a rectangular neighbourhood.
- Pooling becomes approximately invariant to small translations of the input
- Invariance to local translation can be useful property if we are interested about presence of certain feature.
- Pooling summarises the responses, we can reduce output size.

Pooling

- Pooling over spatial regions produces invariance to translation.
- If we pool over the outputs of separately parameterized convolution (or locally connected layer) Can learn which transformation to be invariant to.
- Use pooling to control the output size.

Infinitely Strong Prior

- Using prior probability distribution of parameters, we can encode our prior knowledge of models.
- Strong prior(stronger assumption) vs Weak prior (weaker assumption)
- We can think of convolution network as fully connected layer with infinitely strong prior.
- Enforces parameter sharing, equivalent sharing, invariant to translation and equivalent to translation.
- Can cause underfitting!

Variant of the Convolution Function

- Now we need to consider higher dimension convolution
- Input image is not actually 2D but 3D because of colour channels.
- Also, there are multiple output channels as seen from VGG16
- Let V be input image data, Z output data, K kernel, i output channel index, row j column k and l for input unit

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

- Think of it as summing convolution respective to all input channels

Variant of the Convolution Function

- Strided convolution skips some outputs

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,(j-1)*s+m-1,(k-1)*s+n} K_{i,l,m,n}$$

- We can think of this as downsampling of full convolution
- Discarding few outputs

Zero Padding

- In the boundary, we do not have values to calculate the output
- We use zero near the boundary to calculate the output.
- This is called zero padding.
- If there is no zero padding, output size is smaller than input size. (valid convolution.)
- we can use enough zero padding, output size is same as input size. (same convolution.)
- enough zero padding so each pixel is visited k times, output size is bigger than input size (full convolution)

Unshared convolution

- Locally connected parameter instead of fully connected
- However, we remove parameter sharing.

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} w_{i,j,k,l,m,n}$$

when no guarantee that same feature should occur acros sall of space but each feature depends only on local space not whole space.

Tiled convolution

- Compromise between unshared convlution and full convolution.
- However, we remove parameter sharing.

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,jmodulot+1,kmodulot+1,l,m,n}$$

Kernel takes rotation of few sets.

Visualizing CNN

- Because CNN uses image data we can actually try to visualize what CNN is doing.
- Called Feature Visualization
- We can take learned filter and visualize it.
- Also, we can choose image as input and observe output of each layer.
- We can find optimal input that maximize each filter.
- https://distill.pub/2017/feature-visualization/
- https://towardsdatascience.com/visual-interpretability-for-convolutional-neural-networks-2453856210ce