236606 - Deep Learning

Tutorial 6

Yonatan Geifman  Yair Feldman
Ran El-Yaniv

Computer Science Department
Technion - Israel Institute of Technology

Winter 2018-2019
Table of Contents

1 Convolution

2 Convolutional Layer

3 Convolutional Neural Networks

4 Backprop of Convolutional Layers
Table of Contents

1 Convolution
2 Convolutional Layer
3 Convolutional Neural Networks
4 Backprop of Convolutional Layers
Convolution For Signal Processing

- Given a signal $x$ (a vector) and a filter $f$ (kernel), the operation $x \ast f$ is defined to be a sliding window pass of $f$ over $x$ where at each point the output is the inner product of the overlap of $f$ and $x$.

- This concept can be easily extended to 2D.
2D Convolution Example
Convolution Parameters

- Padding \((p)\)- adding \(p\) layers of zeros around \(x\) to control the output size, usually padding is used to keep the output map at the same size as input.
- Strides \((s)\)- step size \((s)\)of the sliding window
- Examples - Powerpoint
Table of Contents

1 Convolution

2 Convolutional Layer

3 Convolutional Neural Networks

4 Backprop of Convolutional Layers
A convolutional layer for input map whose dimension is $N \times N \times N_c$, is a kernel of size $f \times f \times N_c$.

It is spatially connecting a region of $f \times f$ across all feature maps (depth).

Each filter will produce a feature map of depth 1 in the output layer.

For the feature map of each convolution we add a bias and apply an activation function.

The number of filters $K$ is the depth of the output.

The feature map size depends on the strides and padding.
Convolutional Layer Structure

\[
\text{ReLU} \left( \begin{array}{ccc}
\ast & + b_1 \\
\ast & + b_2 \\
\ast & + b_3 \\
\ast & + b_k \\
\end{array} \right) = \text{ReLU} \left( \begin{array}{ccc}
\ast & + b_1 \\
\ast & + b_2 \\
\ast & + b_3 \\
\ast & + b_k \\
\end{array} \right)
\]

Geifman, Golan, Feldman, El-Yaniv
Given a feature map of input shape $N \times N \times D$ and a filter with shape $f \times f \times D$.

- A padding $p =$? and Strides $s = 1$
- The adjusted input size is $N + 2p$
- The output size is $N + 2p - 2\lfloor f/2 \rfloor$
- What if we set $s = 2$?
Pooling Layers

- In general our purpose is to reduce dimensionality while keeping class depended information.

- How can we reduce dimensionality using the tools we have learned so far?

- Pooling layers spatially subsample the feature maps in a defined region.
Max-pooling

- Max pooling is an operation applied spatially that runs the 2D MAX operation for certain filter size.

- It has a kernel size \((f)\) and strides \((s)\), it simply returns the max value of the input region.

- It is applied on each feature map (in depth) independently.

- There are some other variants of pooling layers (Average pooling), but max-pooling is the most common.
Max-pooling Example

Source: Stanford c231 convolutional neural networks for visual recognition
Geifman, Golan, Feldman, El-Yaniv
Table of Contents

1 Convolution
2 Convolutional Layer
3 Convolutional Neural Networks
4 Backprop of Convolutional Layers
Analyzing Alexnet

Consider the following architecture

Let's calculate the number of weights
Analyzing Alexnet
# Table of Contents

1. Convolution
2. Convolutional Layer
3. Convolutional Neural Networks
4. Backprop of Convolutional Layers
Convolution in Computation Graph

- Convolution operation can be written as matrix multiplication.

- The kernel is decomposed to a doubly blocked circulant matrix, which is a special case of a Touplitz matrix.

- Forward and backward passes can be applied with a standard matrix multiplication.
Convolution in Computation Graph - Example

- Consider a $4 \times 4$ image, and a $3 \times 3$ filter ($w$)

- The image is flatten to a column vector and multiplied by the matrix

$$
\begin{pmatrix}
w_{0,0} & 0 & 0 & 0 \\
w_{0,1} & w_{0,0} & 0 & 0 \\
w_{0,2} & w_{0,1} & 0 & 0 \\
0 & w_{0,2} & 0 & 0 \\
w_{1,0} & 0 & w_{0,0} & 0 \\
w_{1,1} & w_{1,0} & w_{0,1} & w_{0,0} \\
w_{1,2} & w_{1,1} & w_{0,2} & w_{0,1} \\
0 & w_{1,2} & 0 & w_{0,2} \\
w_{2,0} & 0 & w_{1,0} & 0 \\
w_{2,1} & w_{2,0} & w_{1,1} & w_{1,0} \\
w_{2,2} & w_{2,1} & w_{1,2} & w_{1,1} \\
0 & w_{2,2} & 0 & w_{1,2} \\
0 & 0 & w_{2,0} & 0 \\
0 & 0 & w_{2,1} & w_{2,0} \\
0 & 0 & w_{2,2} & w_{2,1} \\
0 & 0 & 0 & w_{2,2}
\end{pmatrix}^T
$$