Convolutional Neural Network

Astrini Sie

asie@seattleu.edu

Week 9

ECEGR4750 - Introduction to Machine Learning
Seattle University

November 16, 2023

Recap and Updates

- Lab Take Home Assignment due next Wednesday 11/22/23 at 11.59pm
- Office Hours: please email me your desired timeslot and to make appointments
- Final Homework 5 due: 12/1/23 at 11.59pm
- Final week of class: review + in class "exam" on 11/30/23 as bonus
- Final project due: 12/8/23 at 11.59pm
- Please review the notes for this lecture from the original source (a very good lecture and class)

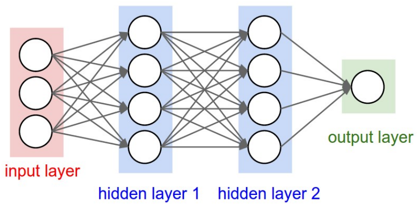
Overview

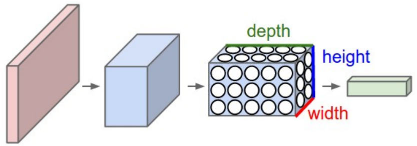
Convolutional Neural Network (CNN or ConvNet)

- Made from an assumption where the inputs are images, which by nature have 2D properties.

Convolutional Neural Network (CNN or ConvNet)

- Made from an assumption where the inputs are images, which by nature have 2D properties.

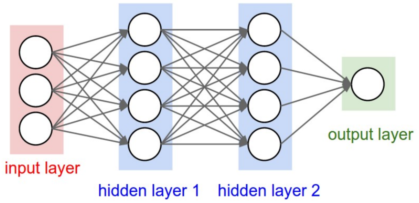


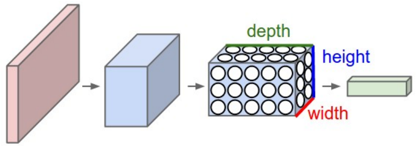


source

Convolutional Neural Network (CNN or ConvNet)

- Made from an assumption where the inputs are images, which by nature have 2D properties.



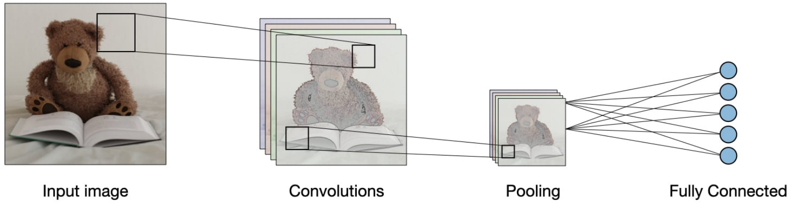


source

- The neurons in CNN are arranged in 3D (width, depth, height). (e.g: the red input layer has width = horizontal pixels, height = vertical pixels, depth = R, G, B channels)

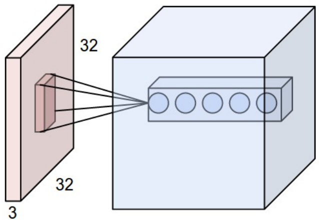
Layers of a CNN

1. **Convolutional Layer** (CONV)
2. **Pooling Layer** (POOL)
3. **Fully-Connected Layer** (FC)
4. **Activation Layer (usually ReLU)** (RELU)



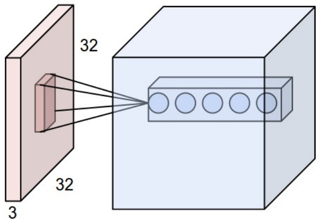
Convolutional Layer

Core building block of a CNN that does most of the computational heavy lifting.



Convolutional Layer

Local Connectivity



Each neuron is connected to only a local region (along width and height in the 2D space), but fully along the depth axis, of the input volume. The spatial extent is called the **receptive field** or the filter size.

Red input volume: 32x32x3. Blue convolutional layer with receptive field: 5x5. Each neuron in the CONV layer will have 5x5x3 weights.

Convolutional Layer

Spatial Arrangement

Hyperparameters:

1. **Depth**. Depth of the convolutional layer. Corresponds to the number of filters we are using, each learning to look for something different in the input, such as: oriented edges, blobs of color, etc. The set of neurons that are all looking at the same region of the input is called the *depth column*.
2. **Receptive Field (Kernel / Filter) Size**. The width and height of the kernel or filter that slides across the input volume..
3. **Stride**. Stride in which we slide the filter. If stride is 1, we move 1 pixel at a time. Usually stride is set to 2.
4. **Zero Padding**. Pad the input volume with zeroes around the border.

Convolutional Layer

Spatial Arrangement

Number of Output Neurons

$$\frac{W - F + 2P}{S} + 1$$

where:

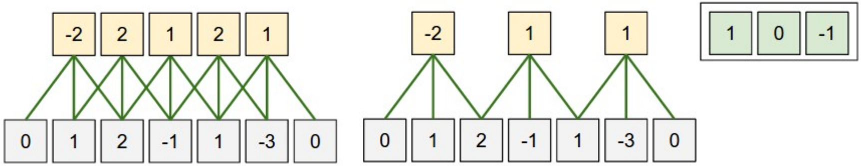
- W: input volume size
- F: receptive field size
- P: amount of zero padding
- S: stride

These parameters have to be carefully selected such that the number of the output neurons is an integer.

Convolutional Layer

Spatial Arrangement

Input volume: grey. Kernel: green. Output volume: yellow.



Left:

- Input size $W = 5$
- Receptive field $F = 3$
- Zero Padding $P = 1$
- Stride $S = 1$
- Output size $\frac{5-3+2}{1} + 1 = 5$

Right:

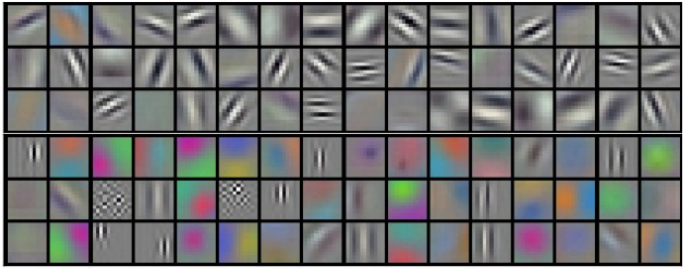
- Input size $W = 5$
- Receptive field $F = 3$
- Zero Padding $P = 1$
- Stride $S = 2$
- Output size $\frac{5-3+2}{2} + 1 = 3$

Convolutional Layer

Parameter Sharing

Neurons from a single 2D slice of depth '*depth slice*' share the same parameters (weights and biases).

ImageNet Challenge 2012 winner, Krizhevsky et al. with output volume of 55x55x96 and receptive field of 11x11x3. Each of the 96 filters below is 11x11x3, and each one is shared by the 55x55 neurons in one depth slice.



source

Convolutional Layer

Parameter Sharing

Parameter sharing make sense when:
Detecting a horizontal edge is important at some location in the image, it should be useful at some other location as well (translational invariance). There is no need to relearn to detect a horizontal edge at every one of the distinct locations.

Parameter sharing doesn't make sense when:
The input images to a CNN have some specific centered structure. For example, when the input are faces that have been centered in the image. Eye-specific or hair-specific features should be learned in different spatial locations. In this case, the parameter sharing scheme is relaxed and the layer is called a **Locally-Connected Layer**.

Convolutional Layer

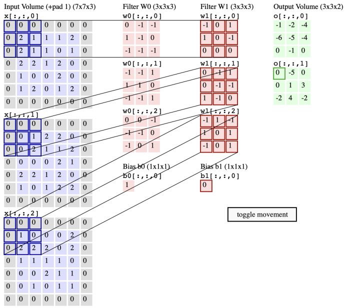
Summary

1. Accepts an input volume of size $W_1 \times H_1 \times D_1$
2. Requires 4 hyperparameters:
    1. Receptive field size F
    2. Amount of zero padding P
    3. Stride S
    4. Number of filters K
3. Produces an output volume of size $W_2 \times H_2 \times D_2$ where:
    - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
    - $H_2 = \frac{H_1 - F + 2P}{S} + 1$
    - $D_2 = K$
4. With parameters sharing:
    1. Number of weights per filter: $F \cdot F \cdot D_1$
    2. Total weights: $(F \cdot F \cdot D_1) \cdot K$ and K biases
5. Common hyperparameter setting: $F = 3$, $S = 1$, $P = 1$

Convolutional Layer

Computation

The visualization below iterates over the output activations (green), and shows that each element is computed by elementwise multiplying the highlighted input (blue) with the filter (red), summing it up, and then offsetting the result by the bias.



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	2	1	1	1	2	0
0	2	1	1	0	1	0
0	2	2	1	2	0	0
0	1	0	0	2	0	0
0	1	2	0	1	2	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	1	2	2	0	0
0	0	2	1	1	2	0
0	0	0	0	2	1	0
0	2	2	1	2	0	0
0	1	0	1	2	2	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	1	0	0	0	2	0
0	2	2	2	0	2	0
0	1	1	1	1	0	0
0	0	0	2	1	1	0
0	0	1	0	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

0	-1	-1
1	-1	0
-1	-1	-1

w0[:,:,1]

-1	-1	1
1	1	0
-1	-1	1

w0[:,:,2]

0	0	-1
-1	-1	0
-1	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	0	1
1	0	-1
0	0	0

w1[:,:,1]

0	-1	1
0	-1	-1
1	-1	0

w1[:,:,2]

-1	1	-1
1	0	1
-1	0	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

-1	-2	-4
-6	-5	-4
0	-1	0

o[:,:,1]

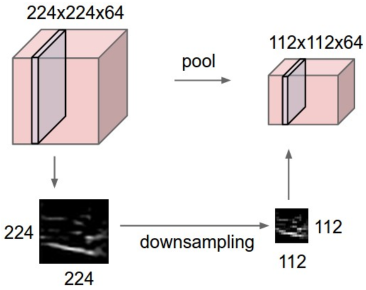
0	-5	0
0	1	3
-2	4	-2

See Full Animation Here

Pooling Layer

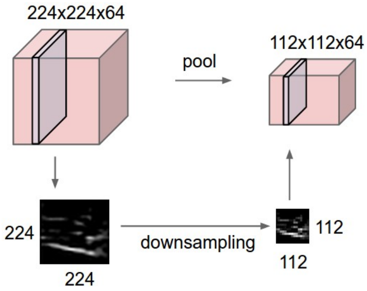
Pooling layer is inserted in-between successive Convolutional layers to:

- Progressively reduce the spatial size of the representation
- Reduce the amount of parameters and computation in the network
- Control overfitting



Pooling Layer

Spatial Arrangement



1. Pooling layer downsamples the volume independently in each depth slice. The **hyperparameters** are:
    1. **Spatial Extent** F
    2. **Stride** S
    3. Typically there is no zero padding in the input of a pooling layer
    4. In practice, $F = 3$, $S = 2$ or $F = 2$, $S = 2$ are commonly used.

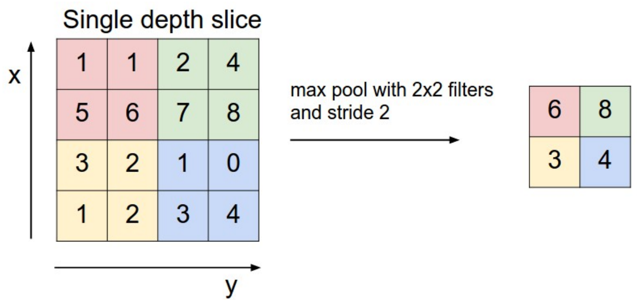
Pooling Layer

Summary

1. Accepts an input volume of size $W_1 \times H_1 \times D_1$
2. Requires 2 hyperparameters:
   1. Receptive field size F
   2. Stride S
3. Produces an output volume of size $W_2 \times H_2 \times D_2$ where:
   - $W_2 = \frac{W_1 - F}{S} + 1$
   - $H_2 = \frac{H_1 - F}{S} + 1$
   - $D_2 = D_1$
4. Common hyperparameter setting: $F = 3$, $S = 2$, or $F = 2$, $S = 2$

Pooling Layer

Computation



The most common downsampling operation is *max*, which yields **max-pooling** method, which is proven to be better than *average-pooling* or *L2-norm pooling*.

Pooling Layer

No pooling

Striving for Simplicity: The All Convolutional Net proposes to discard the pooling layer. To reduce the size of the representation, they proposed using a larger stride once in a while. Discarding pooling is importnat in training good generative models such as VAE or GANs.

Some Examples

1. LeNet
2. AlexNet
3. ZF Net
4. GoogLeNet
5. VGGNet
6. ResNet

References

Fei-Fei Li, Yunzhu Li, Ruohan Gao (2023)
CSE231n Deep Learning for Computer Vision
Stanford University