Convolutional Networks

William Hu

What is Convolution

- Math operation applied on 2 functions, that derives a third function expressing how the shape of one is modified by the other.
- Matrix convolution is the one that we are interested in.
- If we want to calculate A * B, we will call A the input matrix, B kernel or filter.
- Three steps calculation.
    - Element-wise product
    - Sum up
    - Slide

Convolution of two matrices

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

0	0	1
0	0	1
0	1	1

dot

1	0	1
0	1	0
1	0	1

Convolution of two matrices

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{ dot } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 2$$

Convolution of two matrices

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

Convolution of two matrix

Input = $\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

Filter = $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \text{ dot } \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 3$$

Convolution of two matrix

Input =

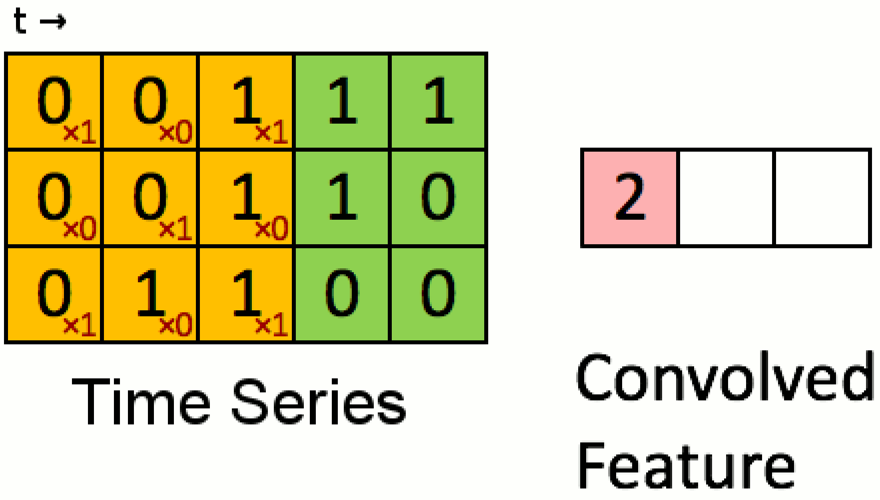
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

0	0	1
0	0	1
0	1	1

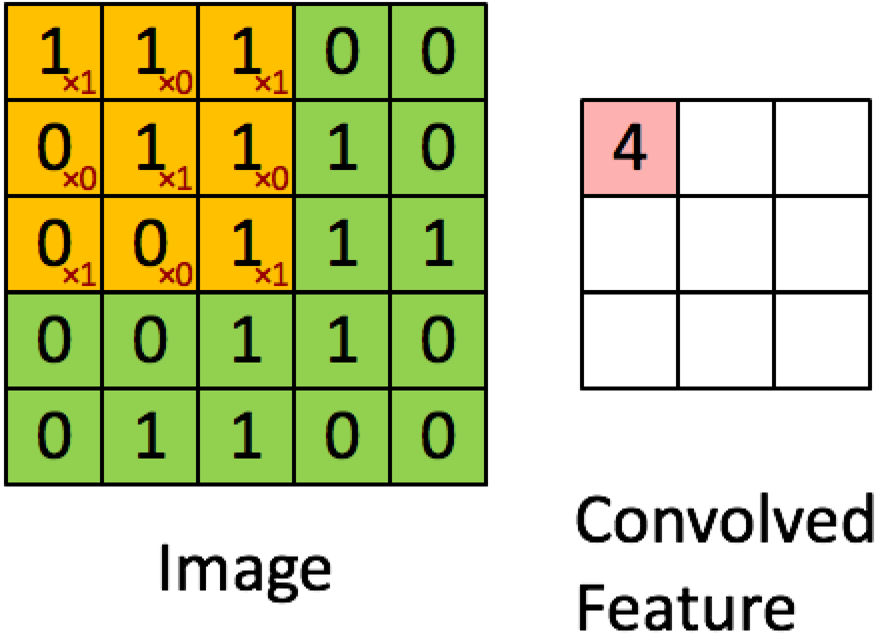
Filter =

1	0	1
0	1	0
1	0	1

How to compute a convolution? (3D time series)



How to compute a convolution?

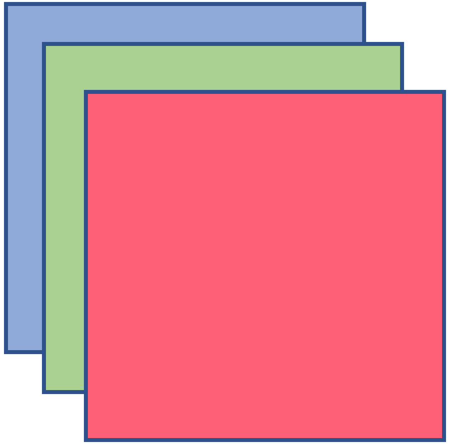


How to compute a convolution?

1. Given an input, overlay a small window (known as the convolution kernel) on the input. Find the dot product between the kernel and the segment of the input that was covered by the kernel.
2. Slide the kernel across the input and perform the previous step for all the different kernel locations.
3. Perform steps 1 and 2 for all output channels, each with a different kernel.

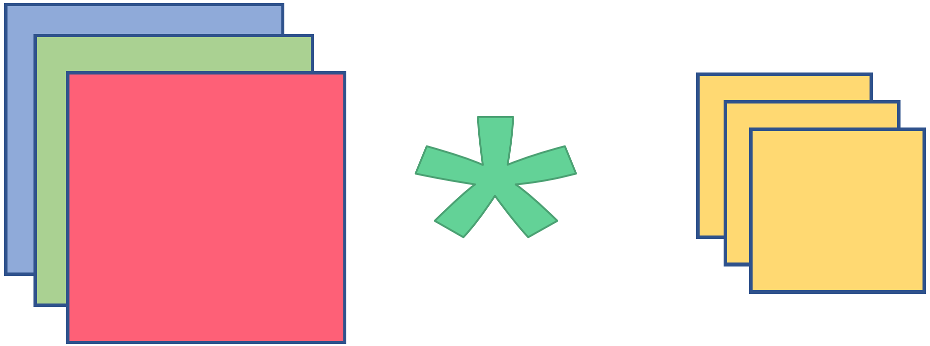
Volume Convolution

- Images has with 3 channels, which is a 3d matrix.



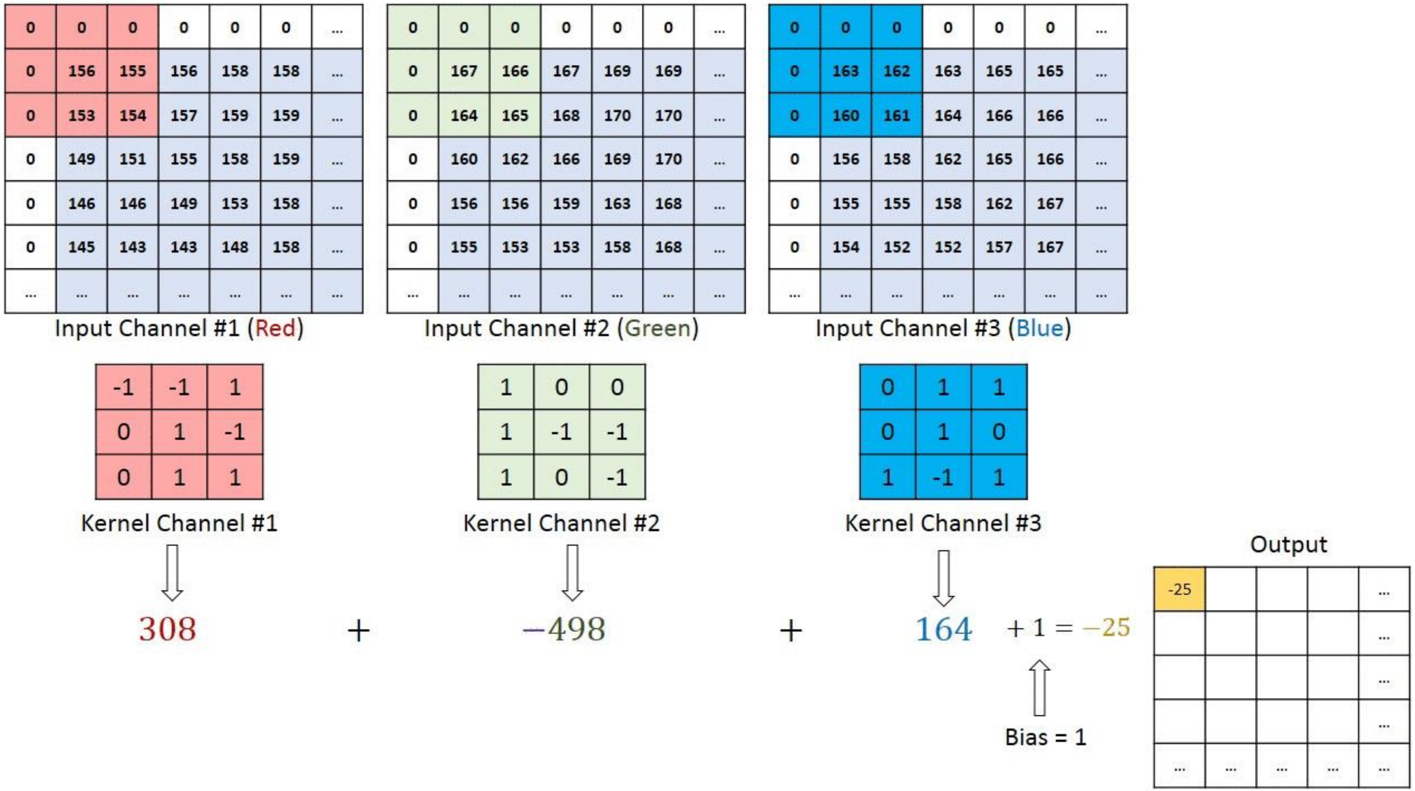
Volume Convolution

- We will need multiple filters and stack them together.

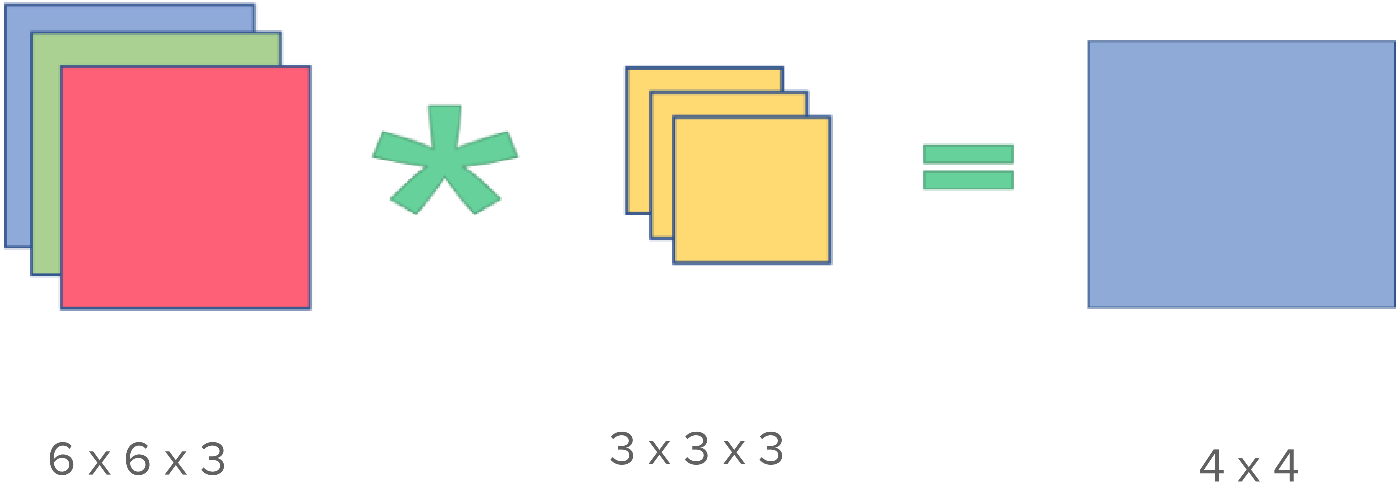


Volume convolution

How to compute a convolution? (Image, 3 channels)



Story so far



How to compute a convolution?

Good questions to ask:

- Does the kernel have to completely fit inside the input?
  - Can apply **padding** to the input
- Do we have to place the kernel at all possible locations? Or can we just place them at regular intervals?
  - Can adjust the **stride** of the convolution

Animations of padding and striding, as well as other sometimes useful concepts such as transposition and dilation:

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Math Equations

- Input volume is n x n x c
- Filter volume will f x f x c
- The output image (matrix) will be m x m.

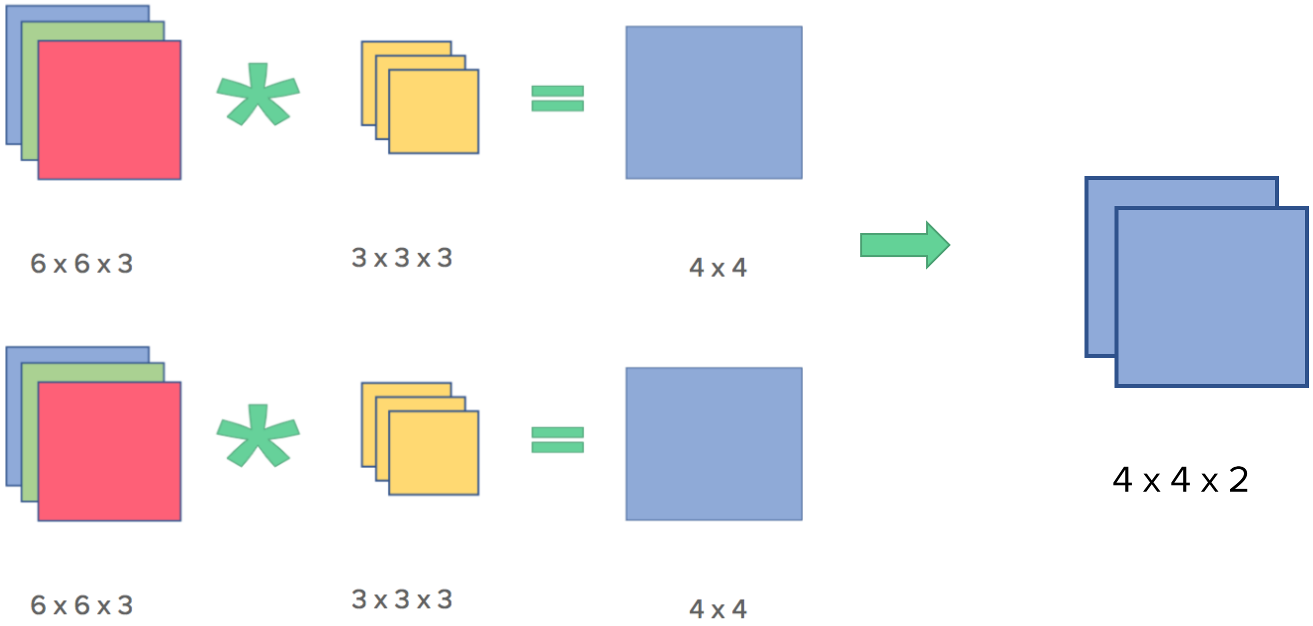
$$m = \left\lfloor \frac{n + 2p - f}{s} \right\rfloor + 1$$

- Where s is stride size, and p is the padding on a single side.

Convolutional Neural network

- Input size (bs, n, n, c)
- Filter size (bs, f, f, c)
- The output size (bs, m, m, k)
    - Where k is the number of filters in the convolutional network

Convolutional Neural Network



What does convolution really do?

Sounds a little arbitrary to slide a window across an input, right?

What do we even get out of this?

1. Convolution as a way to extract local features
2. Convolution as a way to reduce model complexity

Convolution as a feature detector

How do you find the vertical edges in an image?

How do you find the horizontal edges?

How can we combine the two to find the locations of all the edges?

Convolution as a feature detector

How do you find the vertical edges in an image?

-1	0	1
-2	0	2
-1	0	1

How do you find the horizontal edges?

1	2	1
0	0	0
-1	-2	-1

How can we combine the two to find the locations of all the edges?

|*horizontal edge intensity*| + |*vertical edge intensity*|

$\sqrt{(\textit{horizontal edge intensity}^2 + \textit{vertical edge intensity}^2)}$

Convolution as a feature detector

Original

All edges

Vertical edges

Horizontal edges

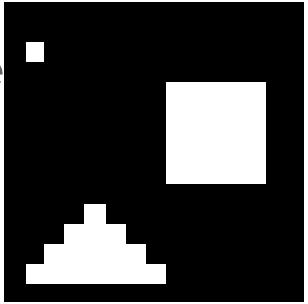
Convolution as a feature detector

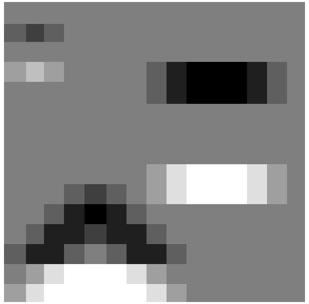
Original Horizontal Vertical

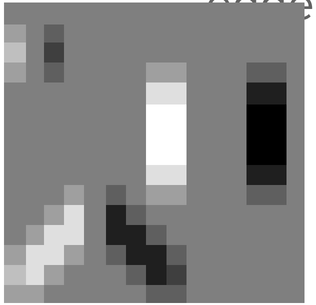
5x5 boxes

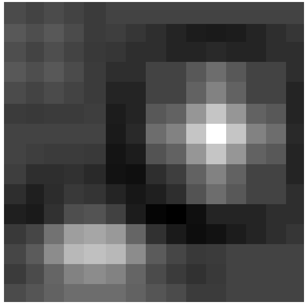
edges

e









Convolution as a feature detector

A convolution is used to extract simple features in the local neighborhood of each position in the input.

- An image ➜ Edges, corners, dots…
- Words ➜ Prepositions, short phrases, compound words…
- Sound intensity waveform ➜ Phonemes, tones, inflection…

Perform convolutions on convolutions (with a nonlinear function in between) to find larger and more complex features in the input.

Convolution to reduce complexity

Let’s consider MLP and CNN. Input dimension of a frame is 40

Let the context size be 10. Hence, one input to the NN is (21, 40). The output dimension is 138

Number of parameters to be learned for a 3 layer MLP with hidden layer size of 128 is

Convolution to reduce complexity

Let's consider MLP and CNN. Input dimension of a frame is 40

Let the context size be 10. Hence, one input to the NN is (21, 40). The output dimension is 138

Number of parameters to be learned for a 3 layer MLP with hidden layer size of 128 is

$$(21*40*128 + 128*128 + 128*128 + 128*138) \sim \mathbf{160k}$$

Convolution to reduce complexity

Let's consider MLP and CNN for MFCC. Input dimension of a frame is 40

Let the context size be 10. Hence, one input to the NN is (21, 40). The output dimension is 138

Number of parameters to learn for a 10 layer CNN with with 3 filters each of size 3x3, stride=1 and no padding

Convolution to reduce complexity

Let’s consider MLP and CNN for MFCC. Input dimension of a frame is 40

Let the context size be 10. Hence, one input to the NN is (21, 40). The output dimension is 138

Number of parameters to learn for a 10 layer CNN with with 3 filters each of size 3x3, stride=1 and no padding. Fully connected layer in the end.

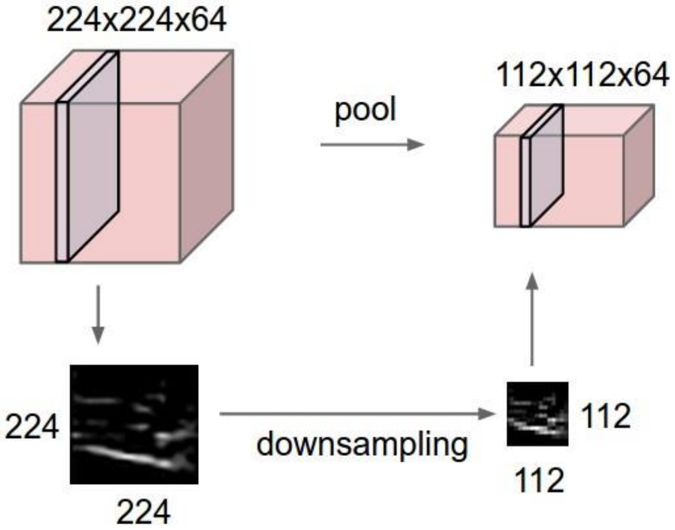
$$((3*3*1*3) + (9 * (3*3*3*3)) + (19*38*138)) \sim \mathbf{100k}$$

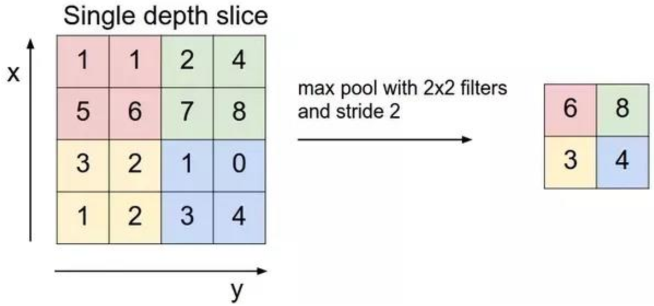
Pooling

A feature map essentially tell us:

- Whether a feature exists in the image (high activation)
- If so, approximately where in the image was it found

We can preserve both pieces of information pretty well with max-pooling





Convolutional Backward