

Special topics: Convolutional Neural Networks

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.

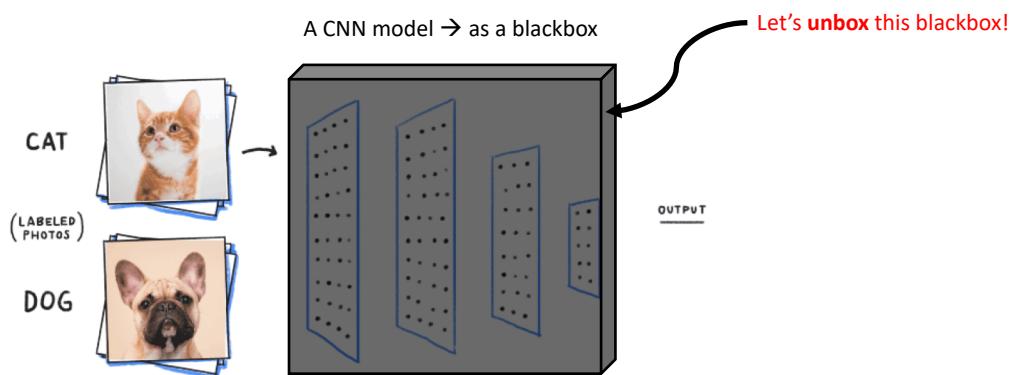
Email: ngocdiep@vnu.edu.vn

Slide & Code: https://github.com/chupibk/INT3414_22

Schedule

Week	Content	Class hour	Self-study hour
1	Introduction CNNs in Computer Vision	2	1
2	Foundations of CNNs Case study: Image classification problem Basics of Neural networks Training with backpropagation Implementation with PyTorch	2	2-6
3	Training and tuning parameters Data augmentation - Data generator Foundations of CNNs Transfer learning	2	2-6
4	Object detection	2	2-6
5	Segmentation	2	2-6
6, 7	Mid-term presentations	2	2-6
8, 9	Advanced topics using CNNs	2	2-6
10, 11, 12	Final project presentations	1	2-6
13	Class summarization	1-3	open

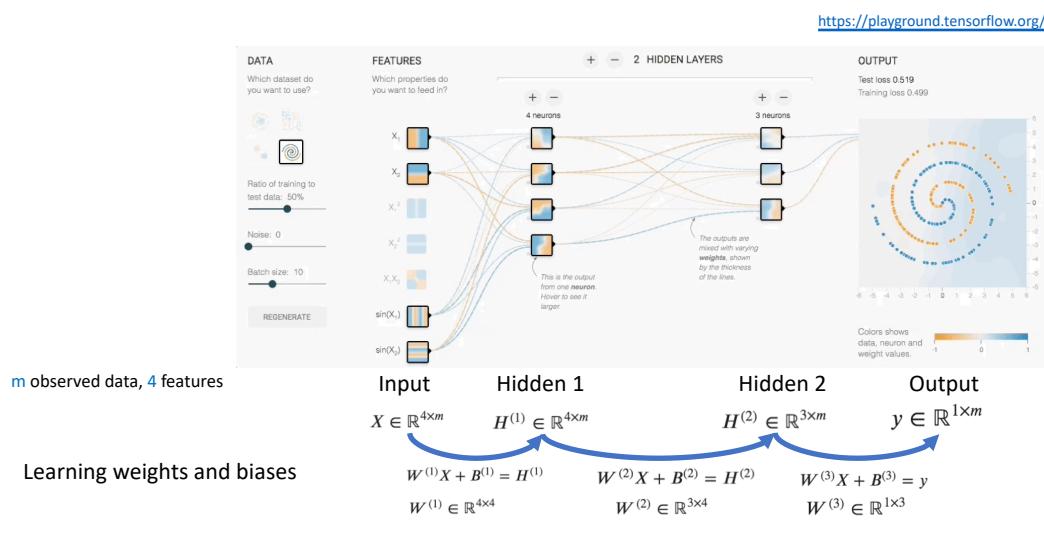
Recall week 1



The most important: design the problem to a solvable task using the blackbox

Image credit: <https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8>

Recall week 2: Build a neural network



Recall week 2: Optimizers

Updating weights iteratively

```

for t in range(num_epochs):
    # Forward pass
    y_pred = model(x_train_t)

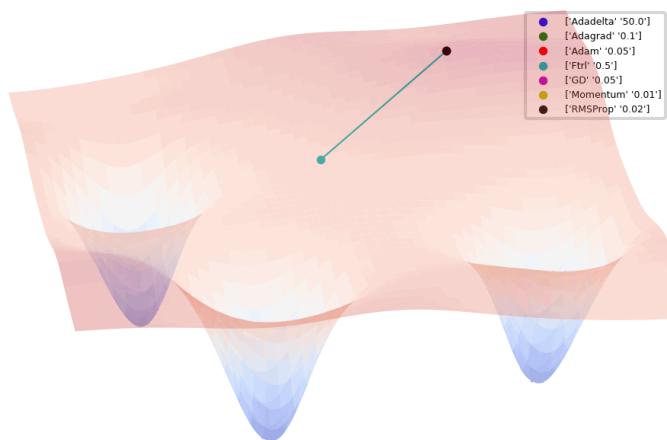
    # Compute loss
    loss = loss_fn(y_pred, y_train_t)

    # Clear the buffers
    optimizer.zero_grad()

    # Backward pass: compute gradients
    loss.backward()

    # update the parameters
    optimizer.step()

```



Visualization credit: <https://github.com/Jaewan-Yun/optimizer-visualization>

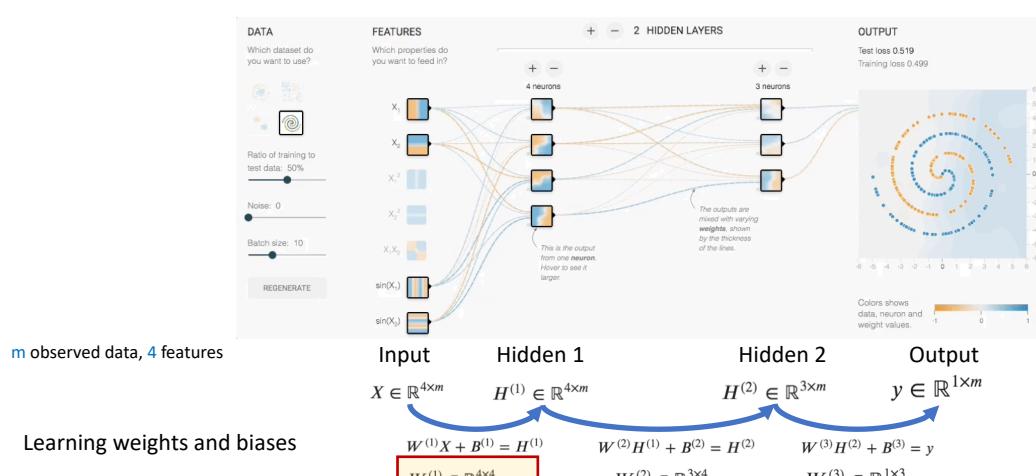
Foundations of CNNs

Week 3

Code @Google Colab

- Link: <https://colab.research.google.com/drive/1hLRa43CbzXeE0-DY7UlTrbDUeITsS7Pe>

From Neural Networks to Convolutional NNs



From NNs to CNNs



Input: 32x32

$$X \in \mathbb{R}^{1024 \times m}$$

$$\text{Hidden layer 1: } W^{(1)} \in \mathbb{R}^{1024 \times 4}$$

Input: 1024x1024

$$X \in \mathbb{R}^{1048576 \times m}$$

$$W^{(1)} \in \mathbb{R}^{1048576 \times 4}$$

From NNs to CNNs: Color images



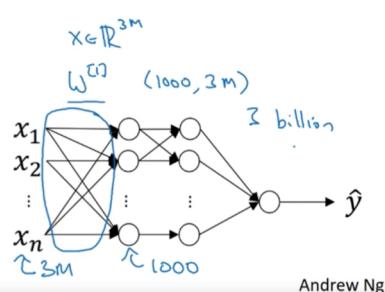
\longrightarrow Cat? (0/1)

64x64x3

12288



$1000 \times 1000 \times 3$
 $= 3 \text{ million}$



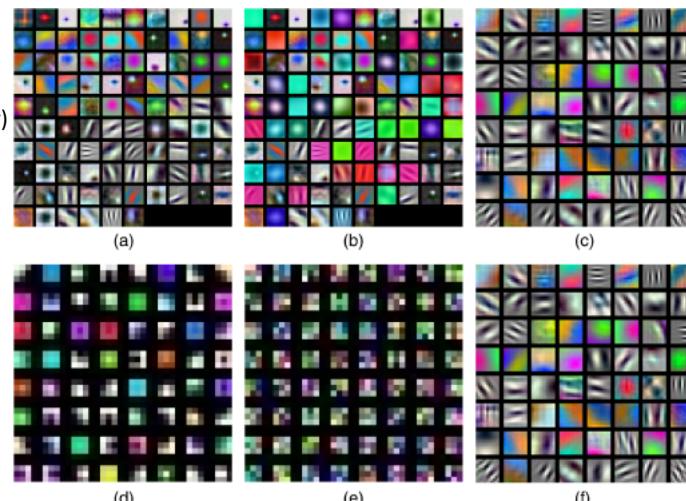
Andrew Ng

Convolutional operations as the fundamental building block of computational neural networks

CNNs for image representations

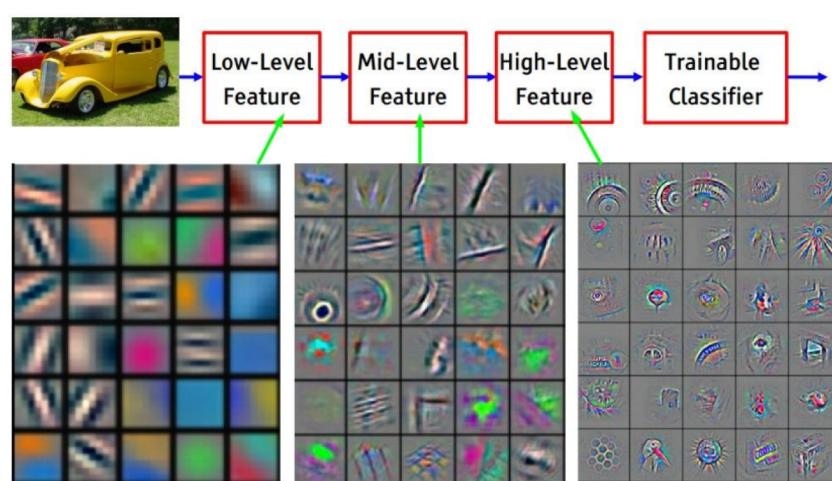
Learned convolutional kernels (first layer)

- a) AlexNet (ImageNet)
- b) AlexNet (CDB)
- c) VGG-f (ImageNet)
- d) VGG-16 (ImageNet)
- e) VGG-16 (CDB)
- f) VGG-f (CDB)



Ref: Georg Wimmer, Andreas Vécsei, Michael Häfner, Andreas Uhl, "Fisher encoding of convolutional neural network features for endoscopic image classification," J. Med. Imag. 5(3) 034504 (24 September 2018)

CNNs for image representations

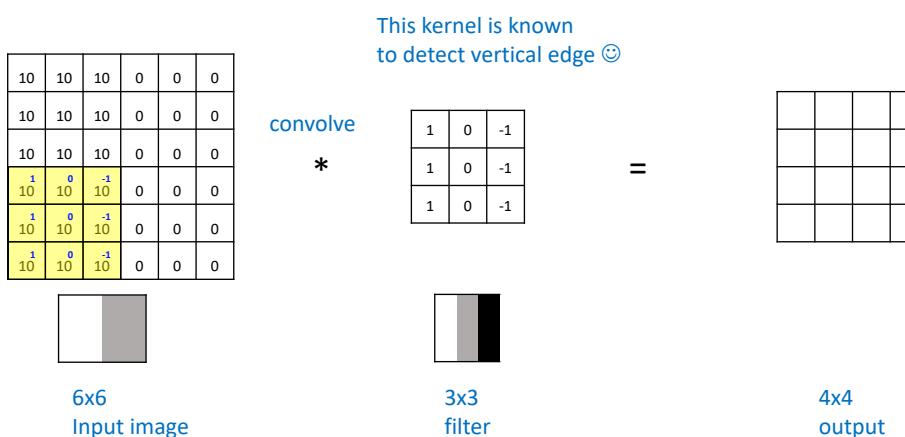


Source: Internet

Question:

What "convolution" operation does to make that representation and learning possible?

Convolution example



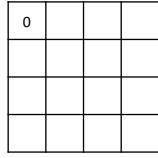
Convolution example

Start from top left...

$$\begin{array}{|c|c|c|c|c|c|} \hline
 1 & 0 & -1 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 1 & 0 & -1 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & & & \\ \hline \end{array}$$


 6x6 image


 3x3 filter


 4x4 output

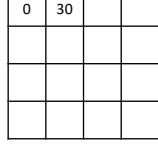
Convolution example

Move one pixel to the next...

$$\begin{array}{|c|c|c|c|c|c|} \hline
 10 & 1 & 0 & -1 & 0 & 0 \\ \hline
 10 & 1 & 0 & -1 & 0 & 0 \\ \hline
 10 & 1 & 0 & -1 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & 30 & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array}$$


 6x6 image


 3x3 filter


 4x4 output


 4x4 output

Convolution example

Next in the row...

$$\begin{array}{|c|c|c|c|c|c|} \hline
 10 & 10 & 10 & \textcolor{blue}{1} & 0 & -1 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & \textcolor{blue}{1} & 0 & -1 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 \end{array} = \begin{array}{|c|c|c|c|} \hline
 0 & 30 & 30 & 0 \\ \hline
 \end{array}$$

6x6 image 3x3 filter 4x4 output

Convolution example

Next to the right...

$$\begin{array}{|c|c|c|c|c|c|} \hline
 10 & 10 & 10 & \textcolor{blue}{1} & 0 & -1 \\ \hline
 10 & 10 & 10 & 0 & \textcolor{blue}{1} & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & \textcolor{blue}{-1} \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 10 & 10 & 10 & 0 & 0 & 0 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 \end{array} = \begin{array}{|c|c|c|c|} \hline
 0 & 30 & 30 & 0 \\ \hline
 \end{array}$$

6x6 image 3x3 filter 4x4 output

Convolution example

Move down one pixel
and begin from the leftmost...

$$\begin{array}{|c|c|c|c|c|c|c|} \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & 30 & 30 & 0 \\ \hline 0 & & & \\ \hline \vdots & & & \\ \hline \vdots & & & \\ \hline \end{array}$$

 6x6 image
 3x3 filter
 4x4 output

Convolution example

Patiently
to the bottom end...

$$\begin{array}{|c|c|c|c|c|c|c|} \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline \end{array}$$

 6x6 image
 3x3 filter
 4x4 output

The thick line found is just the artifact because
the input is only 6x6 pixels.
If the image is larger, the found edge will be very small.

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

$$* \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Light to dark transition



0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

$$* \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Dark to light transition



Credit: Andrew Ng

Vertical and horizontal edge detection

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

Vertical

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

Horizontal

1	1	1	0	0	0
0	0	0	0	0	0
-1	-1	-1	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

6x6

$$* \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

3x3

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

4x4

Learning kernels

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

Vertical

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

Horizontal

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

Sobel

3	0	3
10	0	-10
3	0	-3

Scharr

Any many more

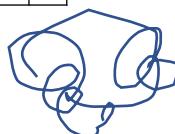
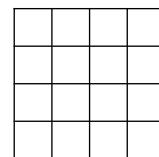
Learning values of kernels automatically

3	0	10	5	24	2
8	4	5	6	10	7
23	3	1	0	0	0
21	4	0	0	0	23
23	0	0	0	45	23
0	3	0	12	34	32

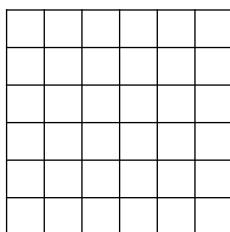
*

W ₁₁	W ₁₂	W ₁₃
W ₂₁	W ₂₂	W ₂₃
W ₃₁	W ₃₂	W ₃₃

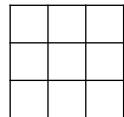
=



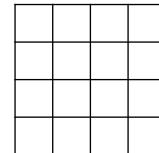
Dealing with boundaries



*



=



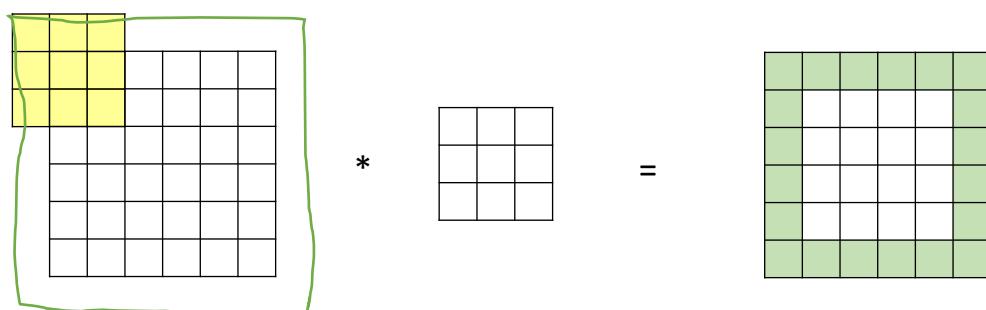
$$n \times n * f \times f \Rightarrow (n - f + 1) \times (n - f + 1)$$

$$6 \times 6 * 3 \times 3 \Rightarrow 4 \times 4$$

Two problems

- 1. After convolution, image is shrunked
- 2. Losing information at the boundaries of the image

Padding



p padding pixel around the boundaries

$$n \times n * f \times f \Rightarrow (n - f + 2p + 1) \times (n - f + 2p + 1)$$

“Valid” and “same” convolutions

“Valid” = no padding

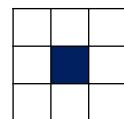
$$\begin{aligned} n \times n * f \times f &\Rightarrow (n - f + 2p + 1) \times (n - f + 2p + 1) \\ 6 \times 6 * 3 \times 3 &\Rightarrow (6 - 3 + 0 + 1) \times (6 - 3 + 0 + 1) = 4 \times 4 \end{aligned}$$

“Same” = Pad so that output size is the same as the input size

$$n \times n * f \times f \Rightarrow (n - f + 2p + 1) \times (n - f + 2p + 1)$$

$$\begin{aligned} n - f + 2p + 1 &= n \\ \Rightarrow p &= (f-1)/2 \end{aligned}$$

\Rightarrow Kernels is often odd!
 $\Rightarrow 3 \times 3, 5 \times 5, 7 \times 7$



Strided convolution

3	0	10	5	24	2
8	4	5	6	10	7
23	3	1	0	0	0
21	4	0	0	0	23
23	0	0	0	45	23
0	3	0	12	34	32

3	0	10	5	24	2
8	4	5	6	10	7
23	3	1	0	0	0
21	4	0	0	0	23
23	0	0	0	45	23
0	3	0	12	34	32

3	0	10	5	24	2
8	4	5	6	10	7
23	3	1	0	0	0
21	4	0	0	0	23
23	0	0	0	45	23
0	3	0	12	34	32

3	0	10	5	24	2
8	4	5	6	10	7
23	3	1	0	0	0
21	4	0	0	0	23
23	0	0	0	45	23
0	3	0	12	34	32

Stride = 2
Padding = 0

$(6 - 3) / 2 + 1 \Rightarrow$ output: 2x2

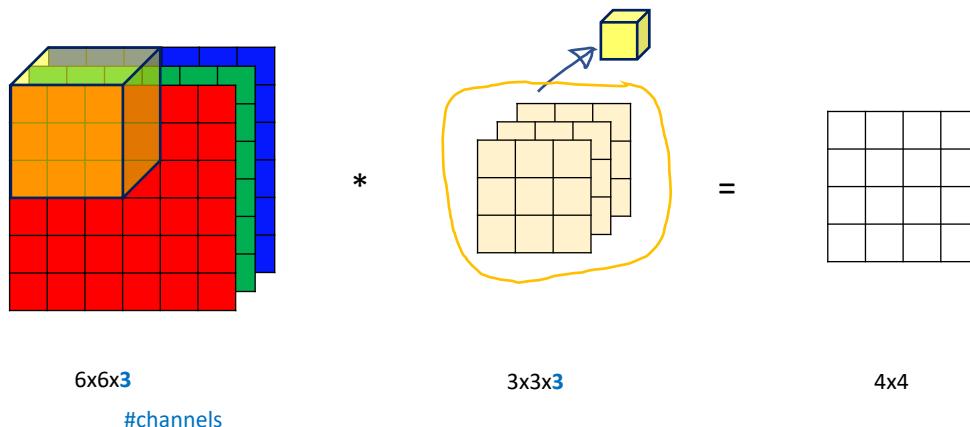
Summary of convolutions

$n \times n$ image $f \times f$ filter

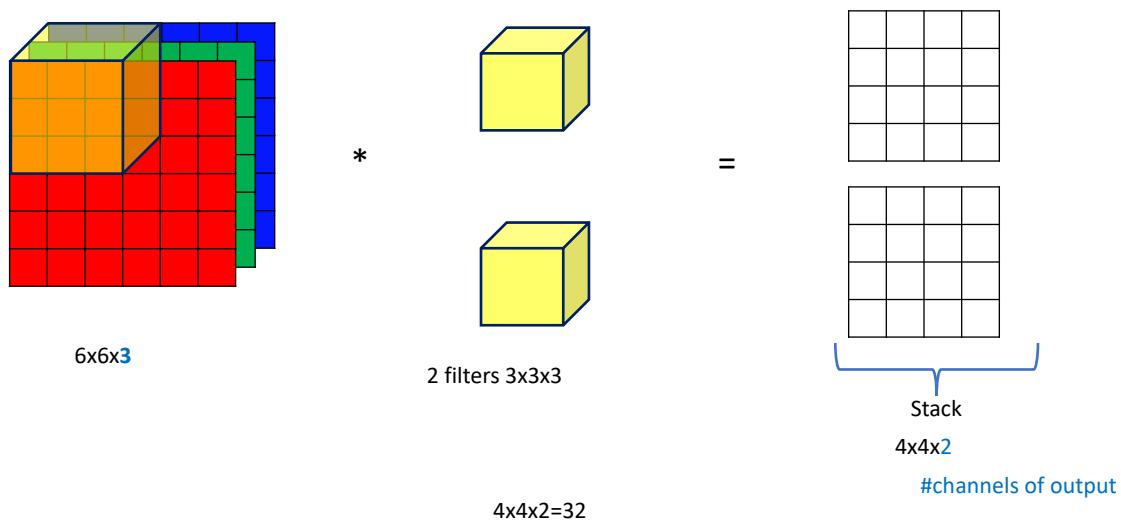
padding p stride s

Size of output: $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \quad \times \quad \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Convolutions over volumes



Multiple filters



Parameters of a Convolution layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters: K ,
 - Spatial extent (filter size): F ,
 - Stride: S ,
 - Amount of zero padding: P
- Produces a volume of size $W_2 \times H_2 \times D_2$
- Introduces $F \cdot F \cdot D_1$ weights per filter
- For a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases

$$\left. \begin{array}{l} W_2 = (W_1 - F + 2P)/S + 1 \\ H_2 = (H_1 - F + 2P)/S + 1 \\ D_2 = K \end{array} \right\}$$

Reference: <http://cs231n.github.io/convolutional-networks/>

Types of layer in a CNN

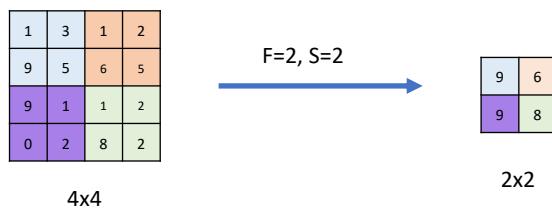
- Convolution (Conv)
- Pooling (Pool)
- Fully connected (FC)

Pooling layer: Max pooling

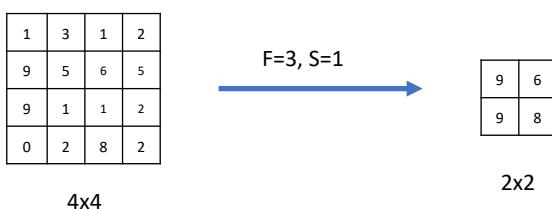
1	3	1	2
9	5	6	5
9	1	1	2
0	2	8	2

4x4

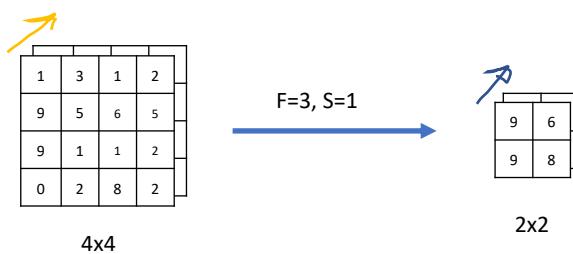
Pooling layer: Max pooling



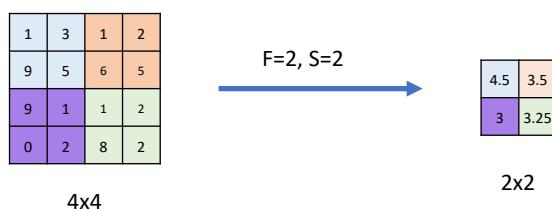
Pooling layer: Max pooling



Pooling layer: Max pooling over volume



Pooling layer: Average pooling



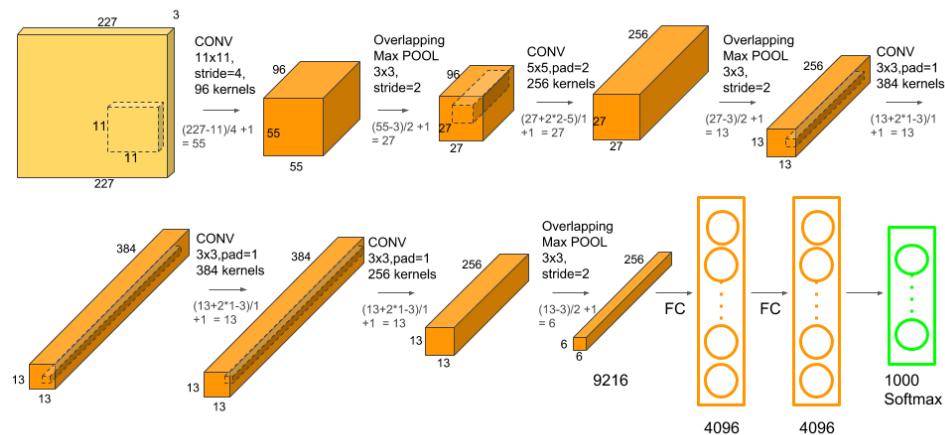
Summary of pooling

- Only hyperparameters:
 - F: filter size
 - S: stride
 - Not usually using padding P
- No parameters to learn!

Advantages of CNNs

- Parameter sharing or translation invariant
 - Once learnt, the filter can apply at any position
- Local connectivity or sparsity connection
 - One neuron can only relate to its neighborhood
 - More effective than using FC

CNN example: AlexNet



Implementation

- Link: <https://colab.research.google.com/drive/1hLRa43CbzXeE0-DY7UITrbDUeITsS7Pe>

```
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,  
          dilation=1, groups=1, bias=True, padding_mode='zeros')
```

Mid-term assignment

- Individual report:
 - Each student chooses a paper and write a report of 2-page A4
- Group presentation
 - Students choose a same paper will automatically form a group
 - Present your “discovery” at the class ☺
- Check this for more information:
 - <https://forms.gle/ZUZgtHL6rLq7RNjU9>

11 groups

<https://www.dropbox.com/sh/hl6hkdp02bjemyk/AADpE5u-2TicqdhgS9ajiLqya?dl=0>

- | | |
|--|---------------------------------------|
| <ol style="list-style-type: none"> 1. VGG 2. MobileNet 3. ResNet 4. ShuffleNet 5. GoogLeNet 6. Inception-v3 7. Inception-v4 | Datasets: CIFAR10
Present together |
| <ol style="list-style-type: none"> 8. Xception 9. ResNext 10. DenseNet 11. EfficientNet | |

<https://forms.gle/ZUZgtHL6rLq7RNjU9>

- What you need to do:

1. Choose a paper (if the choices among students are too imbalanced, a paper will be assigned to you randomly)
2. Read the paper carefully
3. Implement the model and train it for CIFAR-10 dataset.
4. Write a report about your discovery (using the below template)
5. Submit the report (submission method will be announced later)
6. Group with other students who read the same paper to make an awesome presentation

Midterm report template

1. Motivation

Describe the motivation of the paper

2. Method

Describe how authors design the architecture or algorithm to achieve that goal.

--> Demonstrate it by an interesting figure, diagram or code!

3. Evaluation

Describe the experiments the authors use to showcase the advantages of their proposed method.

And describe the disadvantages of the method (either stated by the authors or by your observation)

4. Reproduction

Describe your code and the results when you train the model for CIFAR10 classification dataset.

Note: it's okay to re-use code of others, but don't forget to make a reference and its correctness.