CS480/680: Introduction to Machine Learning

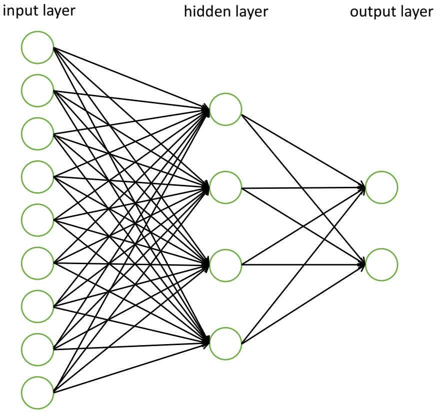
Lecture 10: Convolutional Neural Network

Hongyang Zhang

UNIVERSITY OF WATERLOO

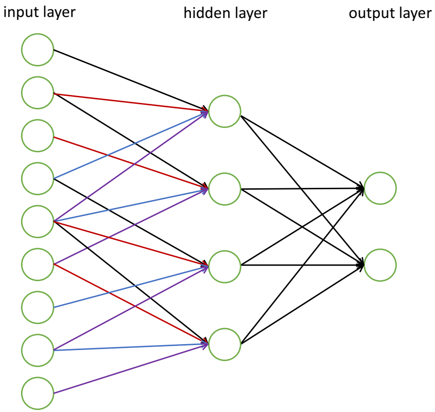
Feb 15, 2024

MLP Recap



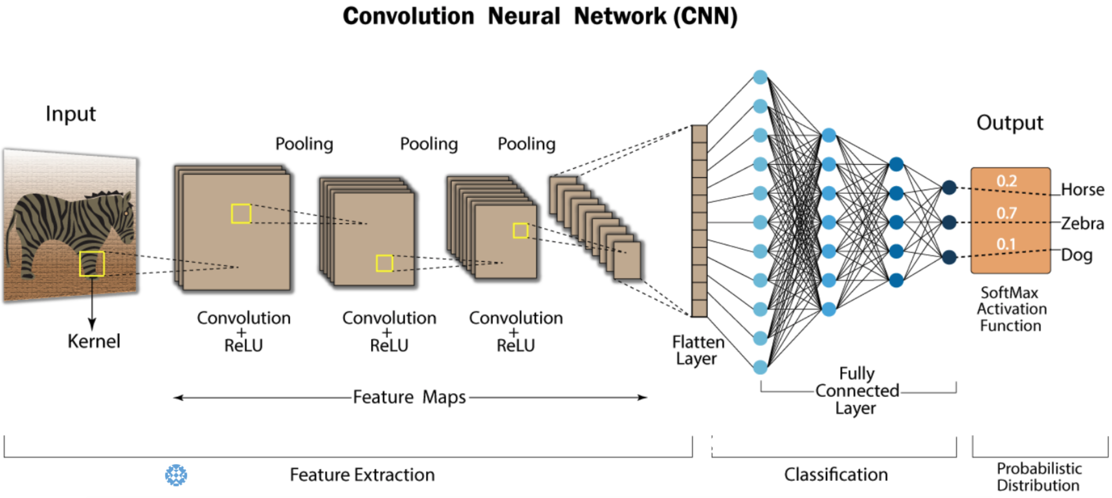
- $f(\mathbf{x}) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$
- Dense weights; Each connection represents a weight to be learned
- Easy to overfit to training data

Convolutional Neural Network



How about considering weight sharing and sparse weight matrix?

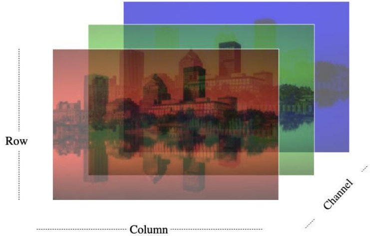
Layers in Convolutional Neural Networks (CNN)

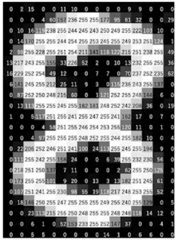


The Form of Image Data

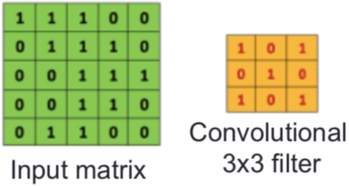
Original image

Red-Green-Blue channels

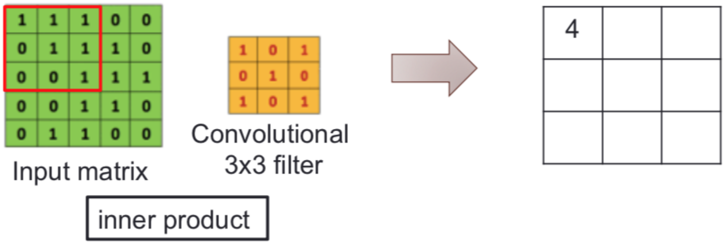




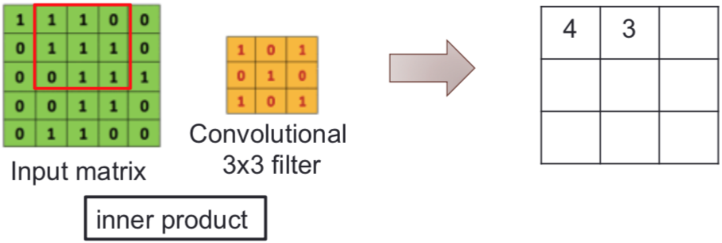
Convolution (One-Channel Input)



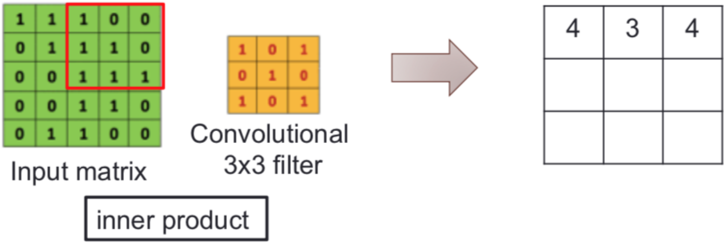
Convolution (One-Channel Input)



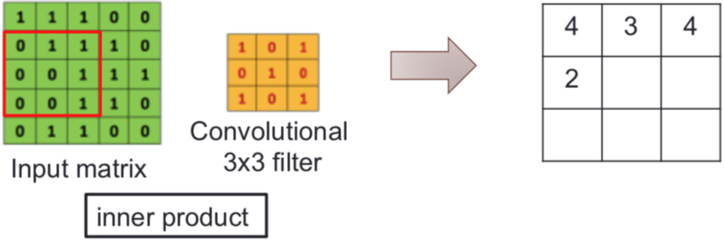
Convolution (One-Channel Input)



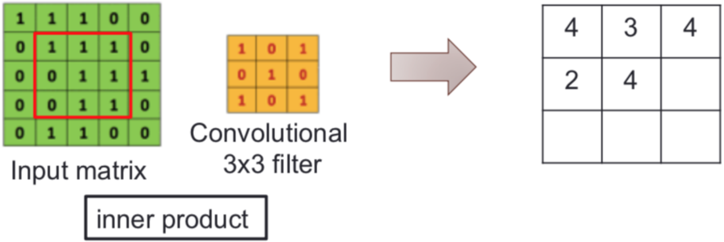
Convolution (One-Channel Input)



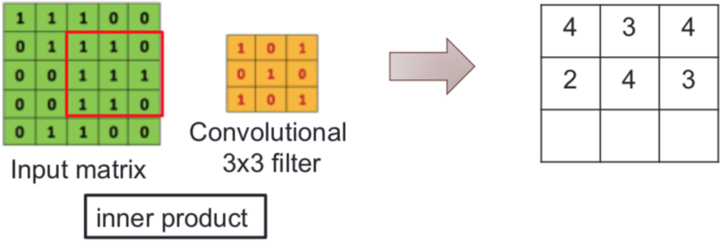
Convolution (One-Channel Input)



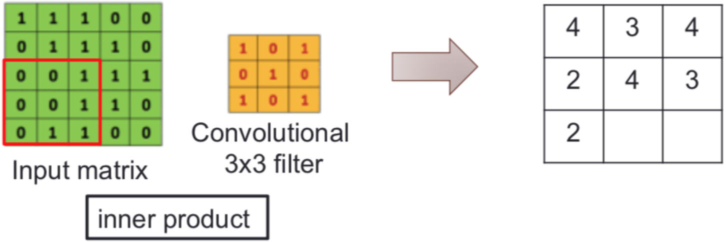
Convolution (One-Channel Input)



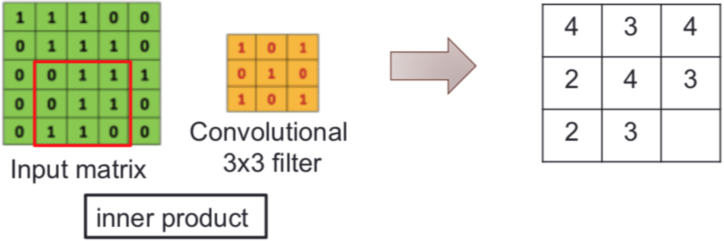
Convolution (One-Channel Input)



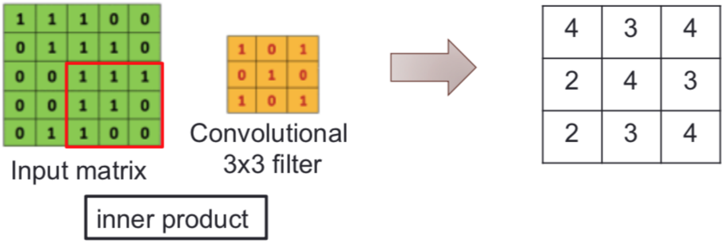
Convolution (One-Channel Input)



Convolution (One-Channel Input)



Convolution (One-Channel Input)



Why Convolution?

- Brain science tells us human visual system is using convolution operation
- Traditional image processing algorithms use convolution operation:

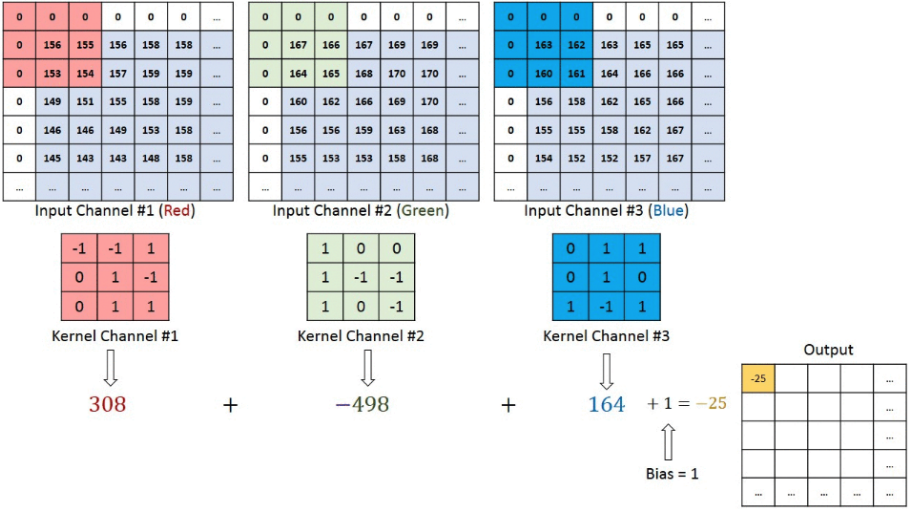
Edge detection (Sobel):

$$* \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} =$$

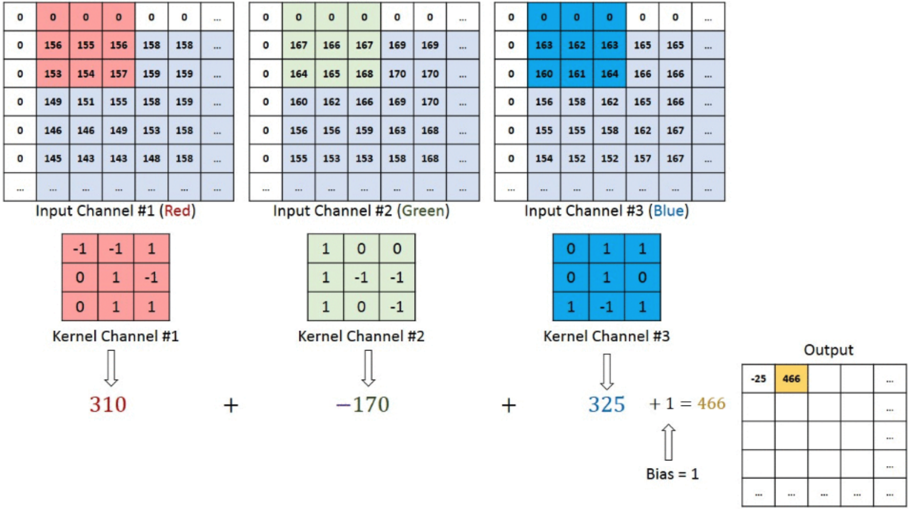
Gaussian smoothing:

$$* \frac{1}{273} \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array} =$$

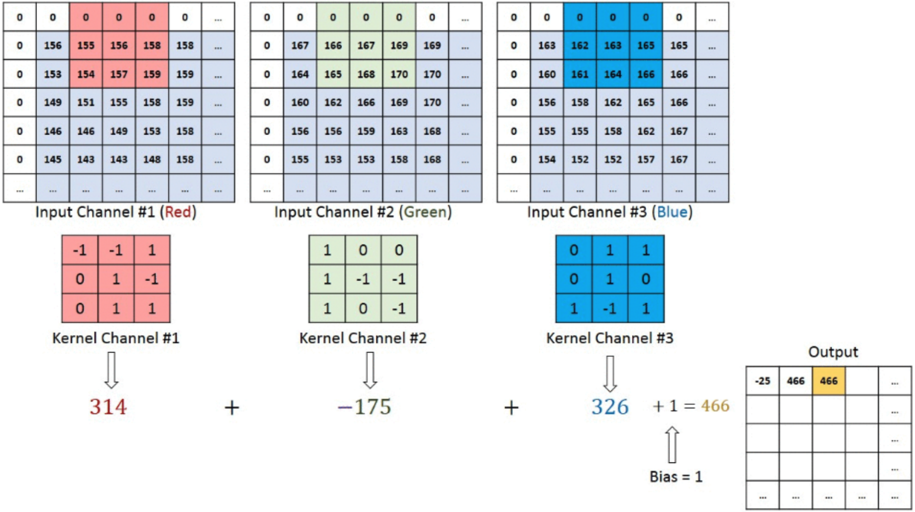
Convolution (Multi-Channel Input)



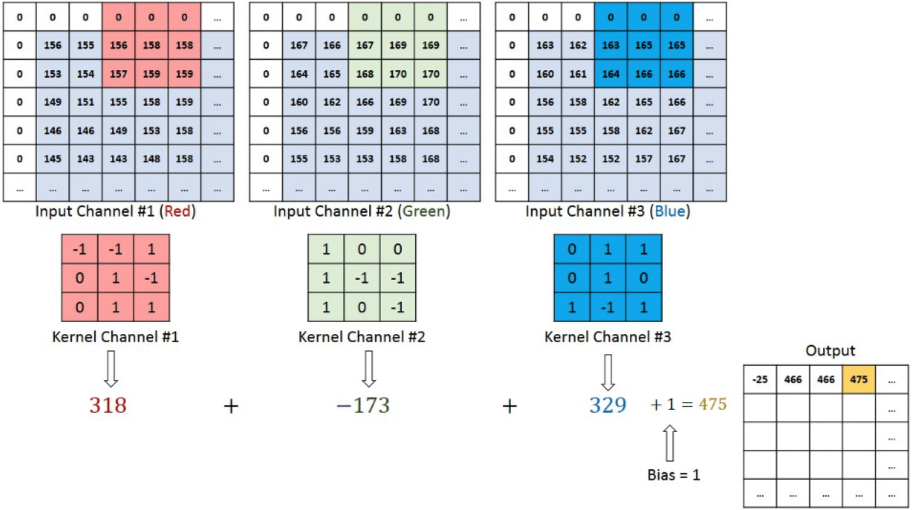
Convolution (Multi-Channel Input)



Convolution (Multi-Channel Input)



Convolution (Multi-Channel Input)



Convolution (Multi-Channel Input)

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

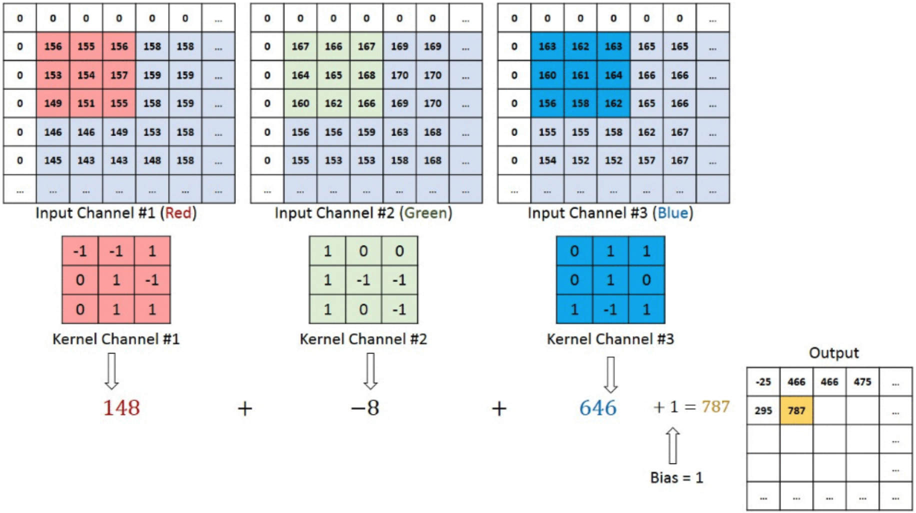
$$298 \quad + \quad -491 \quad + \quad 487 \quad +1 = 295$$

Bias = 1

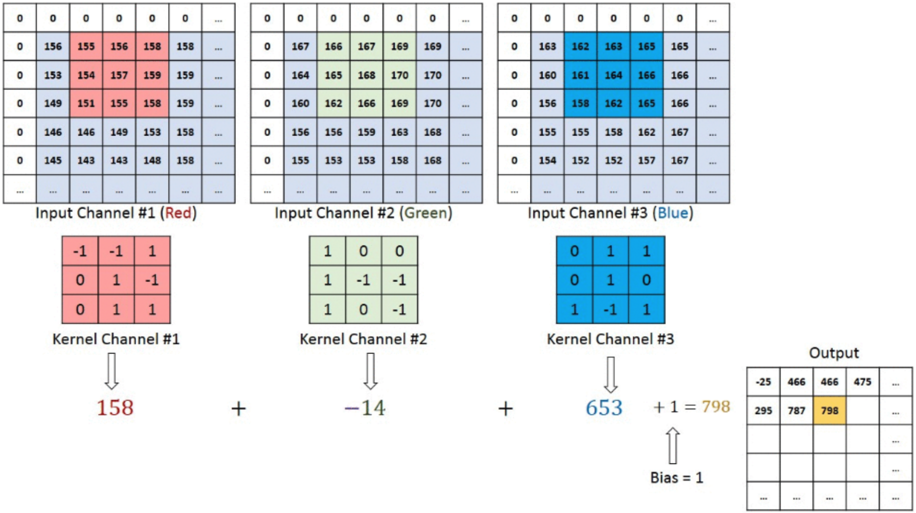
Output

-25	466	466	475	...
295				...
				...
				...
...	...	...	...	...

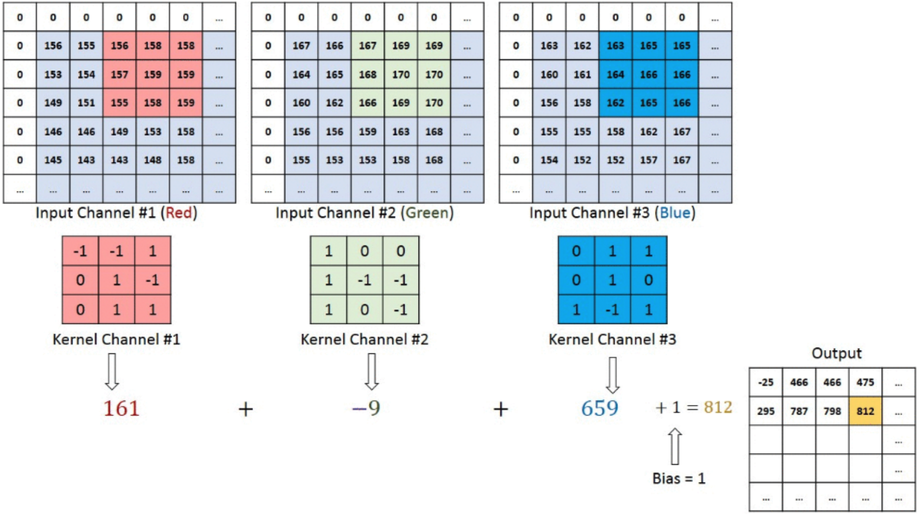
Convolution (Multi-Channel Input)



Convolution (Multi-Channel Input)



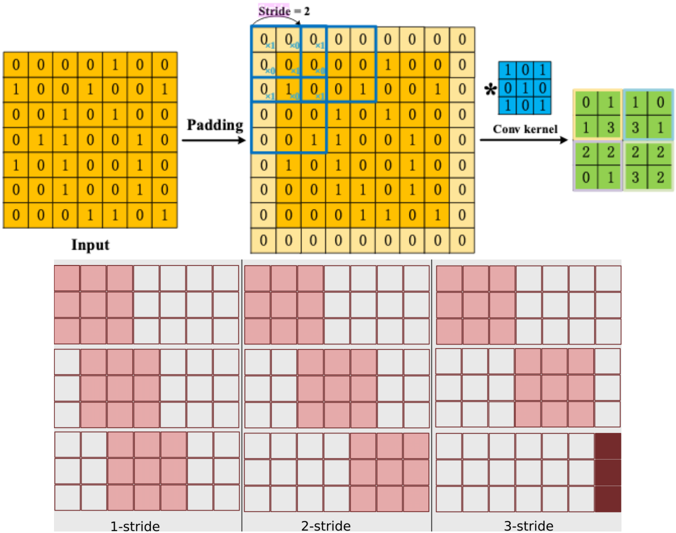
Convolution (Multi-Channel Input)



Controlling the Convolution

- Filter (kernel) size: width x height, e.g. 3 x 3 or 5 x 5; by default, number of channels of each filter is the same as that of the input (a.k.a. c_{in})
- Number of kernels: weights are not shared between different filters; determine the number of channels of output (a.k.a. c_{out})
- Stride: how many pixels the filter moves each time
  - typically stride $\leq$ filter size so as to leave no "gap"
  - larger stride makes neighboring outputs less similar due to less overlap in the input window
- Padding: add zeros around boundary of input
  - keep boundary information lossless

Padding and Stride



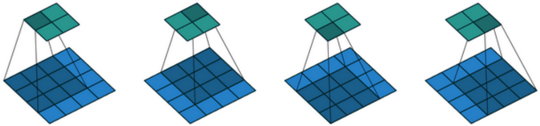
Size Calculation

Input size: $m \times n \times c_{in}$, filter size: $a \times b \times c_{in}$, stride: $s \times t$, padding: $p \times q$ (let the first number refer to the height and the second number refer to the width)

- Pad p pixels on top/bottom and q pixels on left/right
- Move s pixels vertically and t pixels horizontally

- Output size: $\left\lfloor 1 + \frac{m+2p-a}{s} \right\rfloor \times \left\lfloor 1 + \frac{n+2q-b}{t} \right\rfloor$
- With $p = \left\lceil \frac{m(s-1)+a-s}{2} \right\rceil$ and $q = \left\lceil \frac{n(t-1)+b-t}{2} \right\rceil$, you have "output size = input size"

Convolution Layer (One Kernel)=FC Layer with Weight Sharing



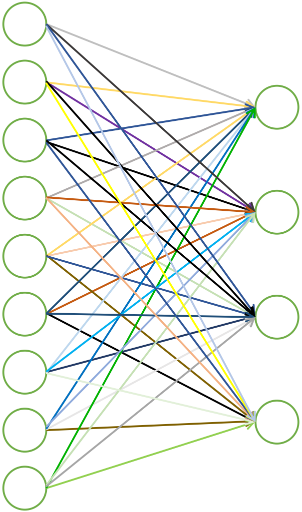
$$\mathbf{W} = \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \end{bmatrix} \in \mathbb{R}^{2\times 2}, \quad \mathbf{X} = \begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ x_{20} & x_{21} & x_{22} \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

$$\mathbf{W}_{\text{circ}} = \begin{bmatrix} w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 & 0 & 0 & 0 \\ 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} & 0 \\ 0 & 0 & 0 & 0 & w_{00} & w_{01} & 0 & w_{10} & w_{11} \end{bmatrix} \in \mathbb{R}^{4\times 9} \text{ (circulant matrix)}$$

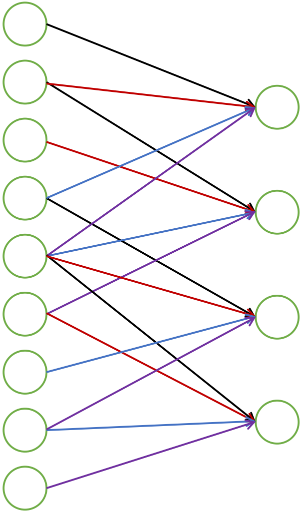
$$\text{Vector}(\mathbf{X}) = [x_{00}, x_{01}, x_{02}, x_{10}, x_{11}, x_{12}, x_{20}, x_{21}, x_{22}]^T \in \mathbb{R}^9$$

$$\text{Vector}(\mathbf{W} * \mathbf{x}) = \mathbf{W}_{\text{circ}}\text{Vector}(\mathbf{X}) \in \mathbb{R}^4$$

Convolution Layer (One Kernel)=FC Layer with Weight Sharing

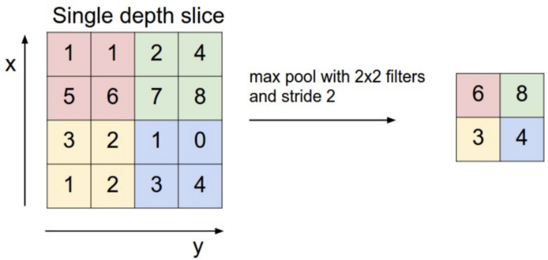


9×4 parameters to be leaned



4 parameters to be leaned

Pooling

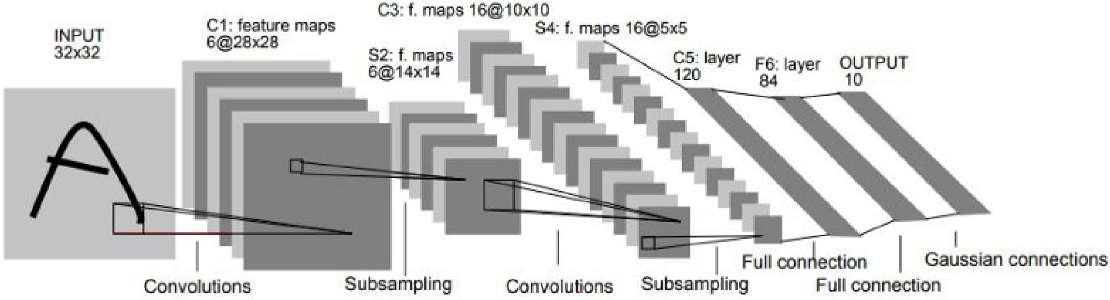


- Down-sample input size to reduce computation and memory
- Pooling by default is performed on each slice separately
  - hence output #channel = input #channel
  - max-pool, average-pool
- Size and stride as in convolution; no parameter; typically no padding
- Global pooling: take the max or average of the whole input slice; output size is 1×1

Putting Everything Together

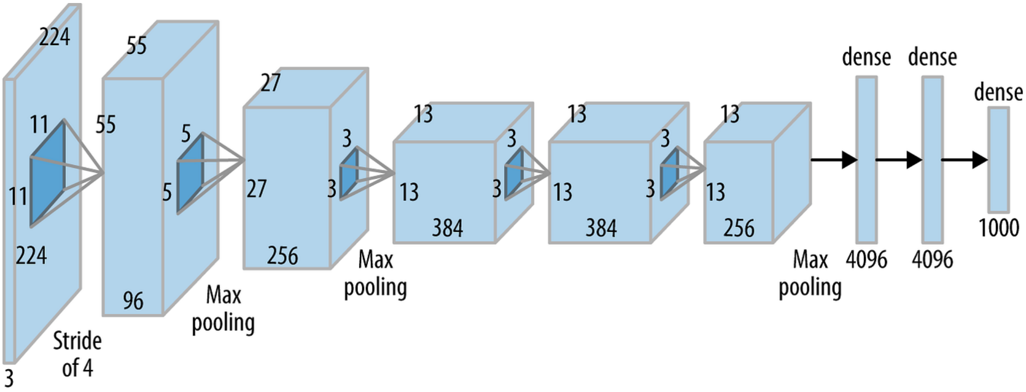
- Several standard architectures to choose (examples to follow)
- Try and adapt to fit your problem

LeNet



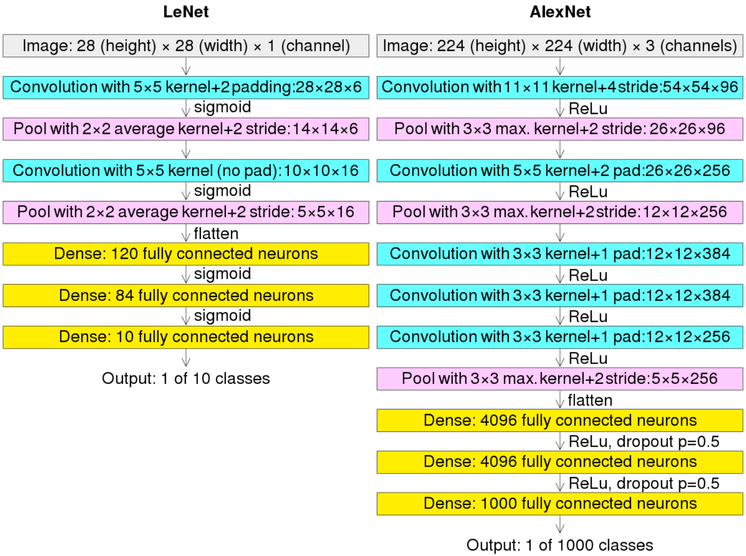
Y. LeCun et al. "Gradient-based learning applied to document recognition". Proceedings of the IEEE, vol. 86, no. 11 (1998), pp. 2278–2324.

AlexNet



A. Krizhevsky et al. “ImageNet Classification with Deep Convolutional Neural Networks”. In: Advances in Neural Information Processing Systems 25. Ed. by F. Pereira et al. 2012, pp. 1097–1105.

Comparisons of LeNet and AlexNet



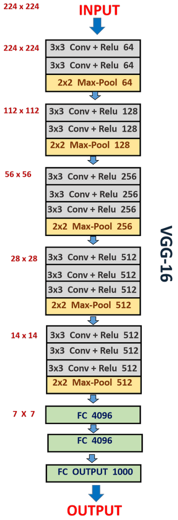
VGGNet

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-scale Image Recognition". In: ICLR. 2015.

Memory

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Note:

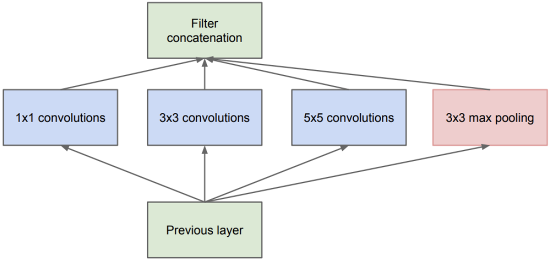
Most memory is in early CONV

Most params are in late FC

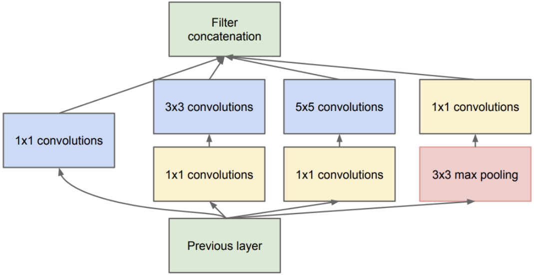
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Let's go even deeper!

Inception



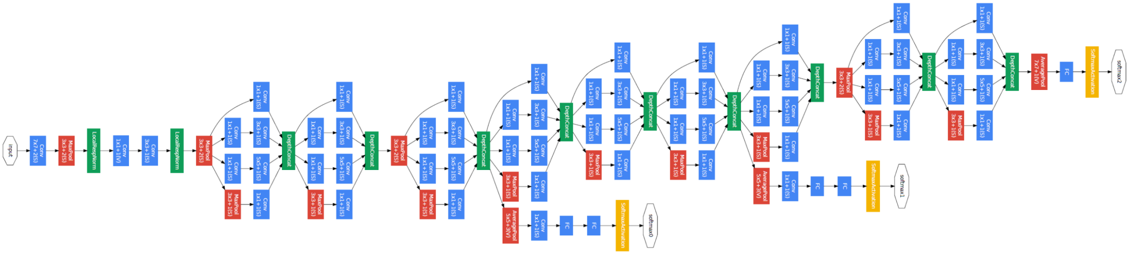
(a) Inception module, naïve version



(b) Inception module with dimension reductions

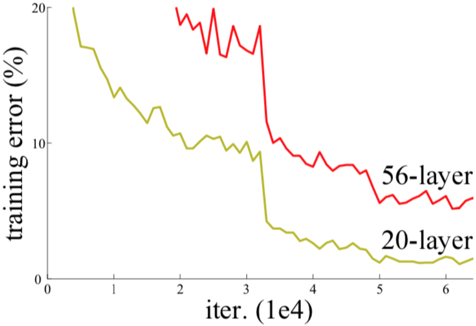
C. Szegedy et al. "Going deeper with convolutions". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015, pp. 1–9.

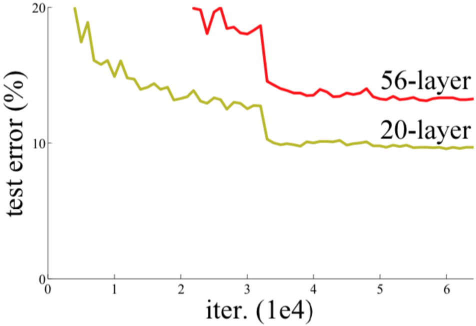
GoogLeNet



- No fully connected (FC) layers
- Deeper but more efficient and better performance

The Deeper, the Better, but More Difficult to Train

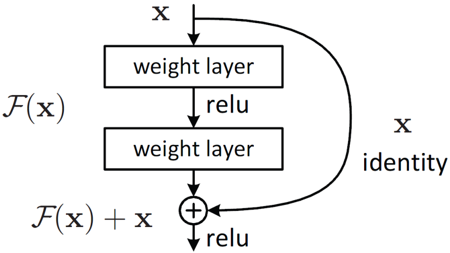




- Deeper models are harder to train due to vanishing / exploding gradient
- Can be worse than shallower networks if not properly trained!

K. He et al. "Deep Residual Learning for Image Recognition". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, pp. 770–778.

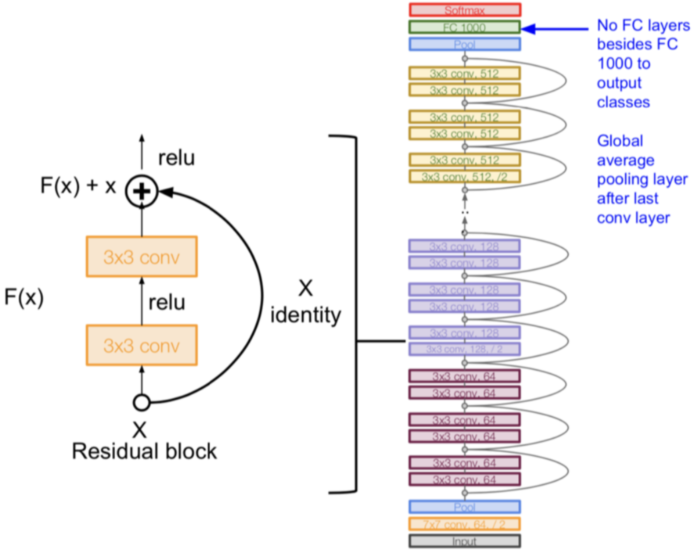
Residual Block



- Add a shortcut connection that allows "skipping" one or more layers
- Effectively turning the block into learning residual: output - input
- Allows more direct backpropogation of the gradient through the "shortcut"
- Can also concatenate or add a linear layer if dimensions mismatch

Residual Network (ResNet)

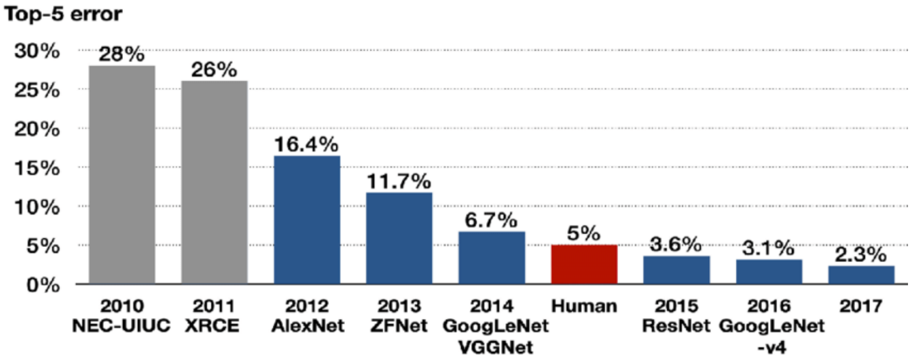
Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)



ImageNet (ILSVRC) Competition

- Training set: 1.28M images
- Validation set: 50K images
- Test set: 100K images
- #Classes: 1K

ImageNet (ILSVRC) Competition




Questions
Answers