

CS480/680: Introduction to Machine Learning

Lec 09: Convolutional Neural Network

Yaoliang Yu



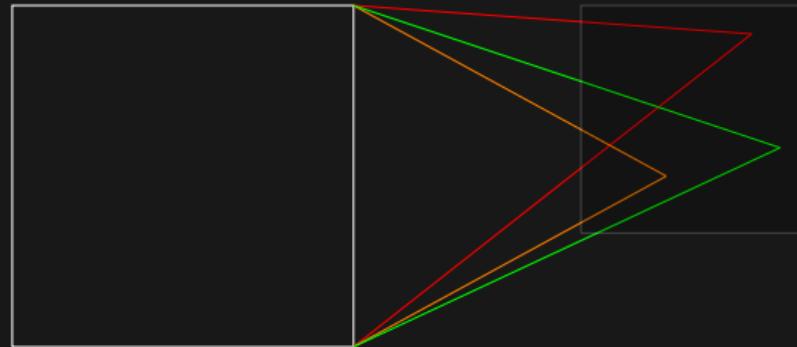
UNIVERSITY OF
WATERLOO

FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL
OF COMPUTER SCIENCE

June 05, 2024

MLP Rearranged

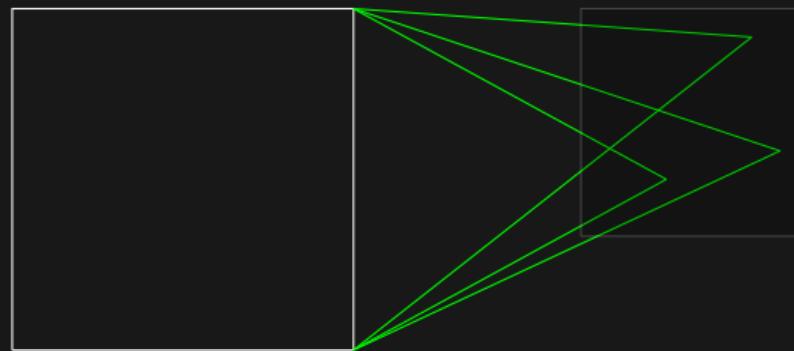
input image
or input feature map output feature maps



- Fully connected; say input/output dim $m \times n \times d / p \times q$
 - how many weights are there?

Weight Sharing

input image
or input feature map output feature maps



- Share weights; say input/output dim $m \times n \times d / p \times q$
 - how many weights are there?
 - is there any new issue?

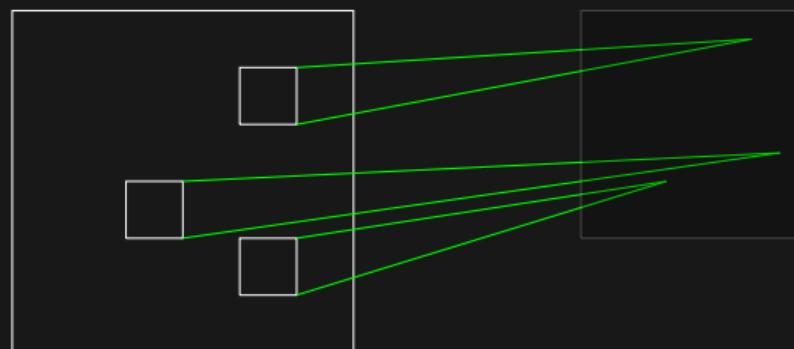
Enumerating All Possibilities

$$h = \sigma(\langle w, x \rangle + b)$$

- Same x , different w : MLP
- Same x , same w
- Different x , same w
- Different x , different w

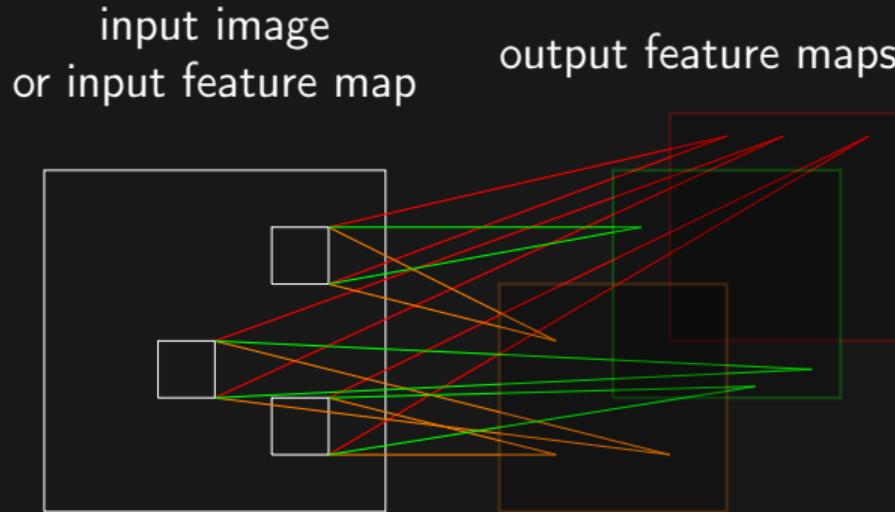
Weight Sharing with A Restricted Field

input image
or input feature map output feature maps



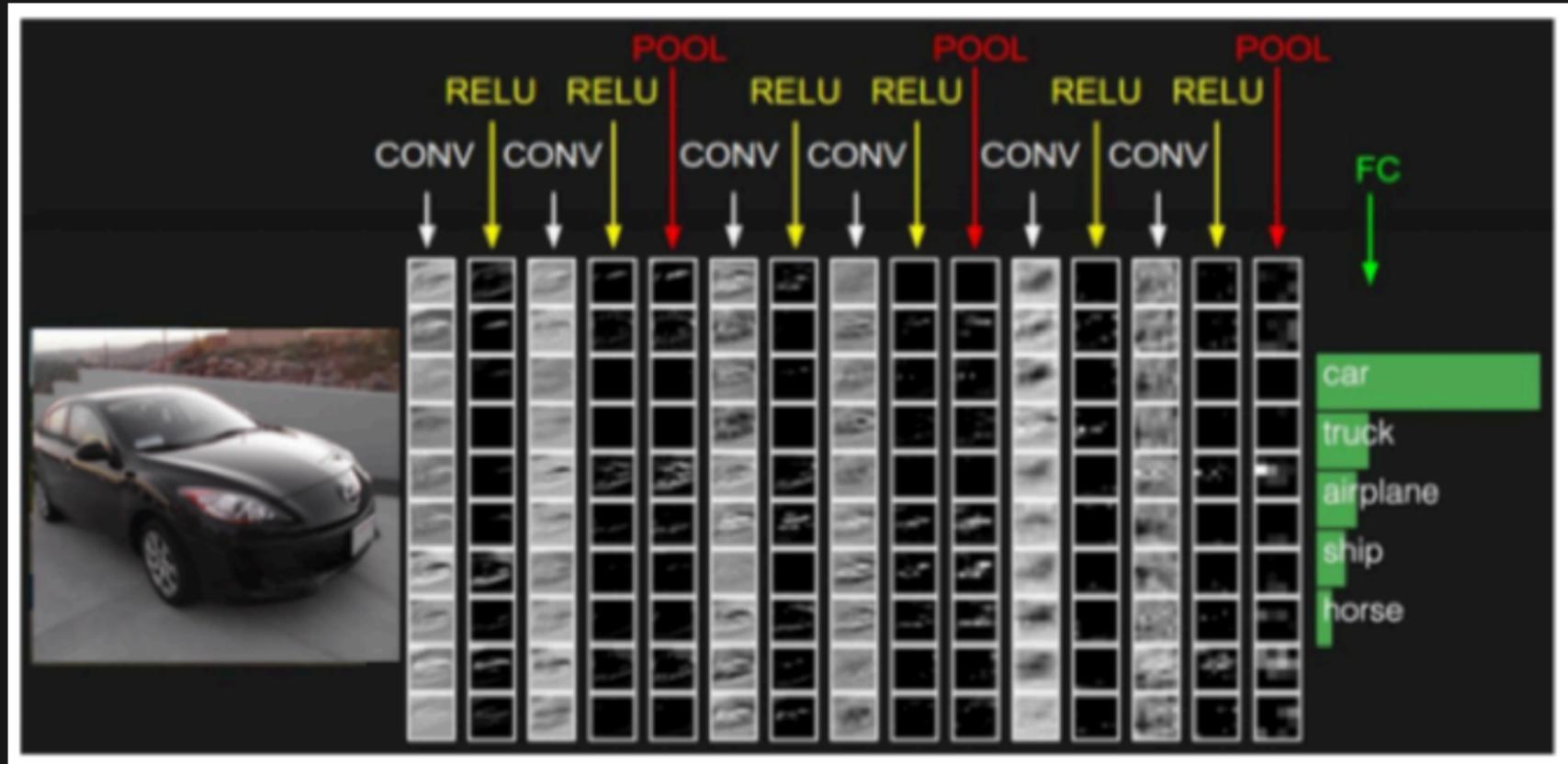
- Share weights; say input/output dim $m \times n \times d / p \times q$, filter size $a \times b$
 - how many weights are there?
 - is there any new issue?

Weight Sharing and Convolution

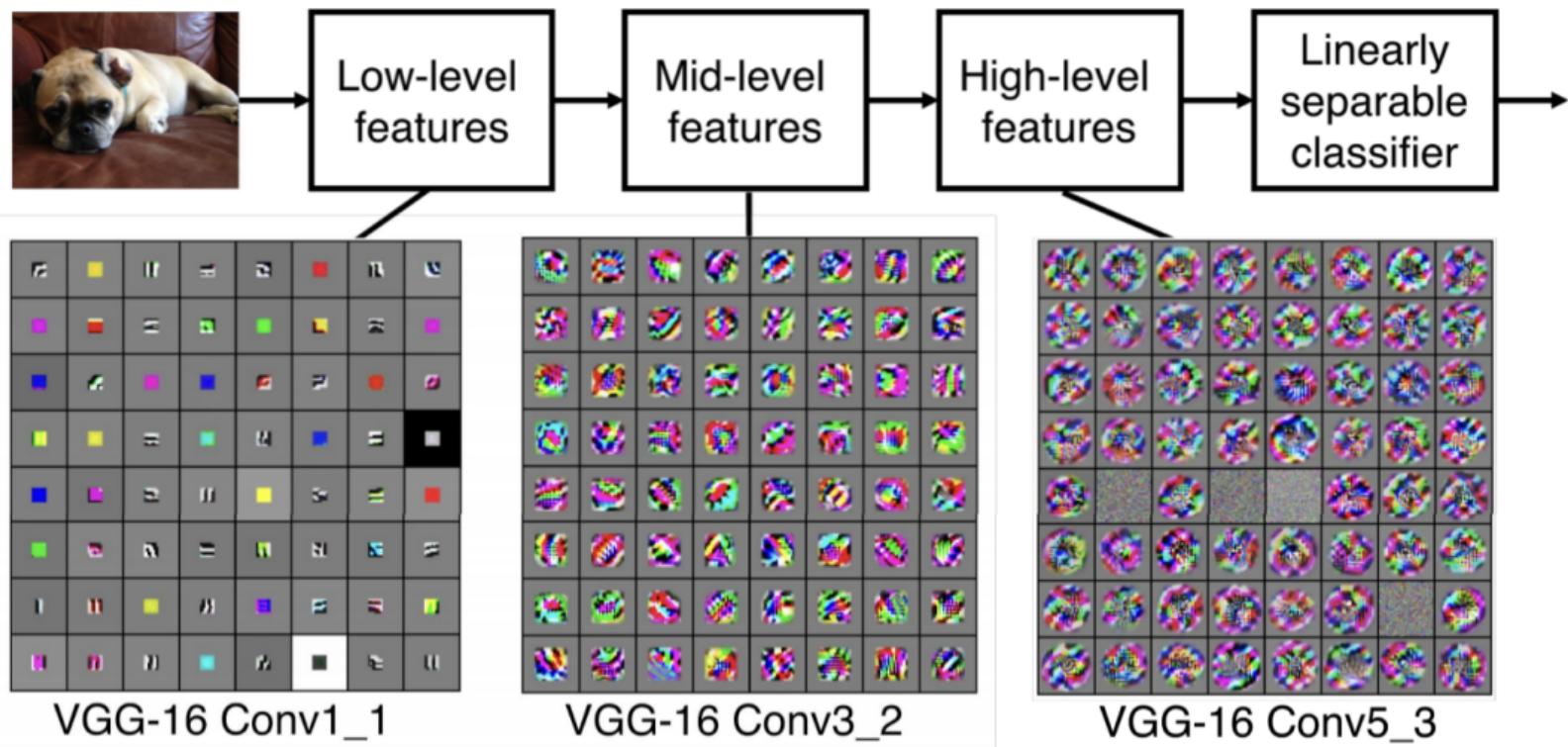


- Share weights; say input/output dim $m \times n \times d / p \times q \times c$, filter size $a \times b$
 - how many weights are there?

Layers in Convolutional Neural Networks (CNN)



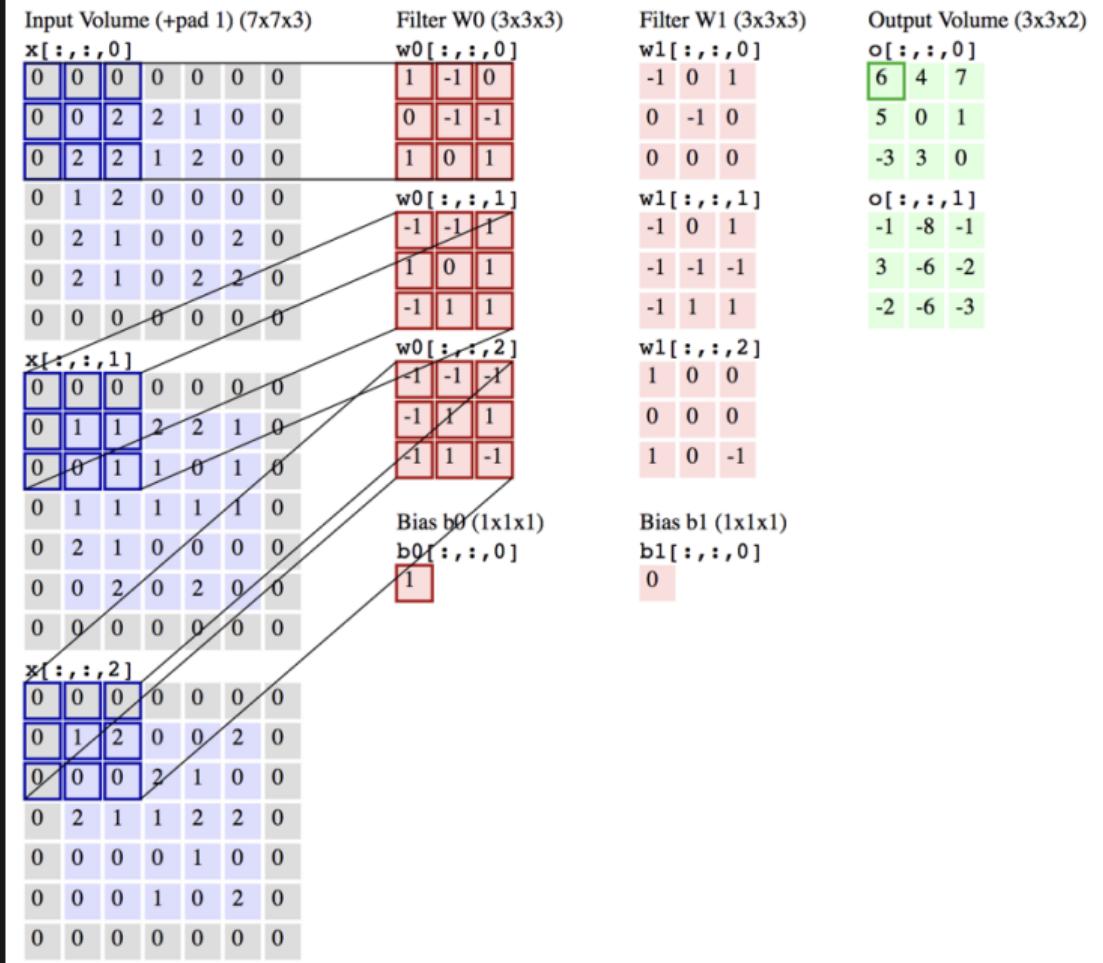
Hierarchical Feature Representation



M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: European Conference on Computer Vision (ECCV). 2014, pp. 818–833.

Controlling the Convolution

- Filter size: width \times height, e.g. 3×3 or 5×5 ; by default, depth of each filter is the same as that of the input
- Number of filters: weights are not shared between filters; determine depth (channel) of output
- Stride: how many pixels to move the filter 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 (or any other value) around boundary of input
 - make the output size more standard (e.g. same as input, or 2^k for some k)



Size Calculation

Input size: $m \times n \times c$, filter size: $a \times b$, stride: $s \times t$, padding: $p \times q$

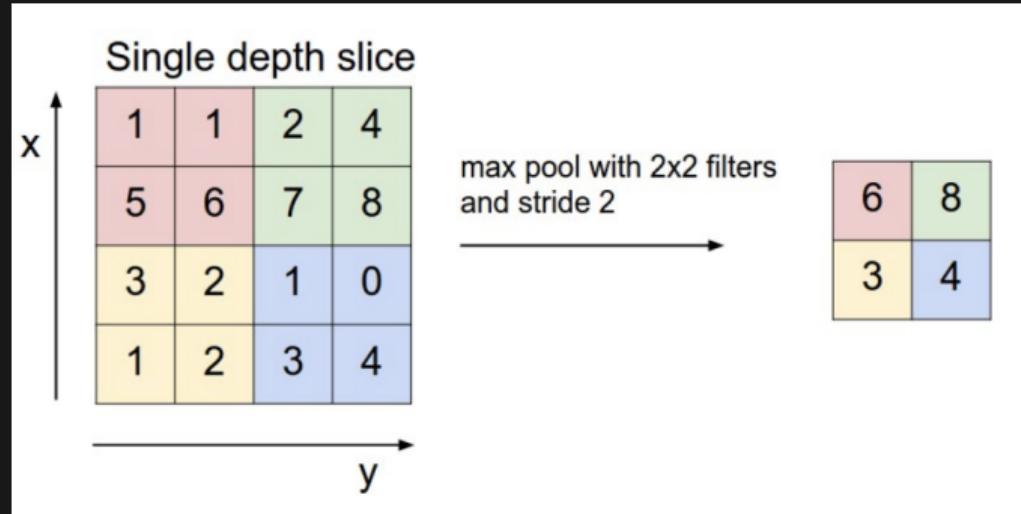
- Pad p pixels on left/right and q pixels on top/bottom (typically $p = q$)
- Filter size is $a \times b \times c$ but we omit the last dimension
- Move s pixels horizontally and t pixels vertically
- Output size: $\left\lfloor 1 + \frac{m+2p-a}{s} \right\rfloor \times \left\lfloor 1 + \frac{n+2q-b}{t} \right\rfloor$
- With $p = \left\lceil \frac{m(s-1)+a-s}{2} \right\rceil$ and $q = \left\lceil \frac{n(t-1)+b-t}{2} \right\rceil$, output size = input size

Receptive Field

Input size: $m \times n \times c$, filter size: $a \times b$, stride: $s \times t$, padding: $p \times q$

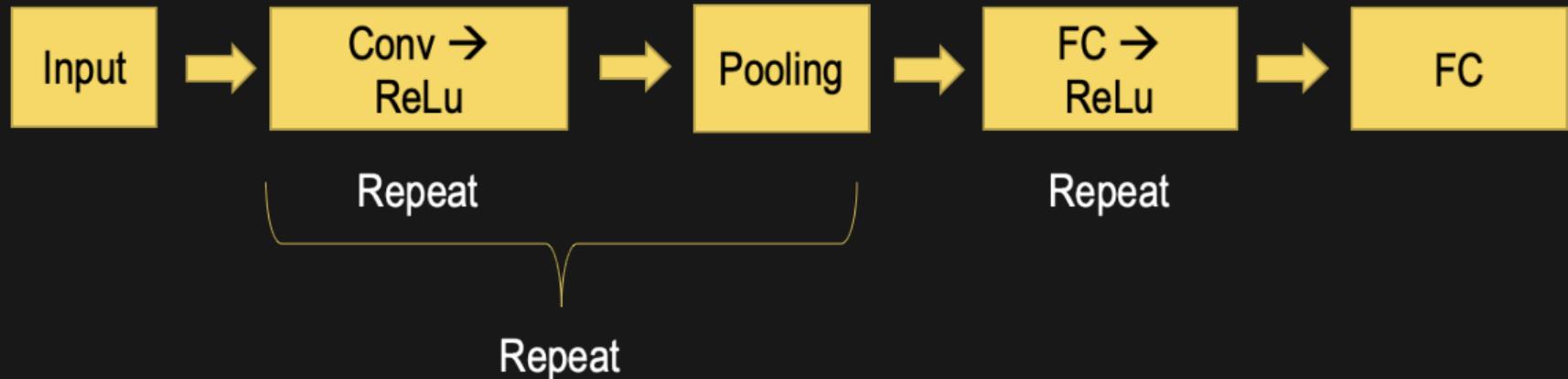
- How many pixels in the input can each output pixel “see” (i.e. depend on)?
 - obviously $a \times b$ (filter size); again, omitting channels by default and ignoring boundary
- How many pixels in the input can a $i \times j$ window in output “see”?
 - $[(i - 1)s + a] \times [(j - 1)t + b]$, assuming stride \leq filter size and ignoring boundary
- How many pixels in layer k can a $i \times j$ window in layer $l \geq k$ “see”?
 - $i \cdot \prod_{r=k+1}^l s_r + \sum_{u=k+1}^l (a_u - s_u) \prod_{v=k+1}^{u-1} s_v$, assuming stride \leq filter size and ignoring boundary
 - $i \cdot s^{l-k} + (a - s) \frac{s^{l-k} - 1}{s - 1}$, and $i + (a - 1)(l - k)$ if $s \equiv 1$, i.e. increase by $a - 1$ for every layer

Pooling



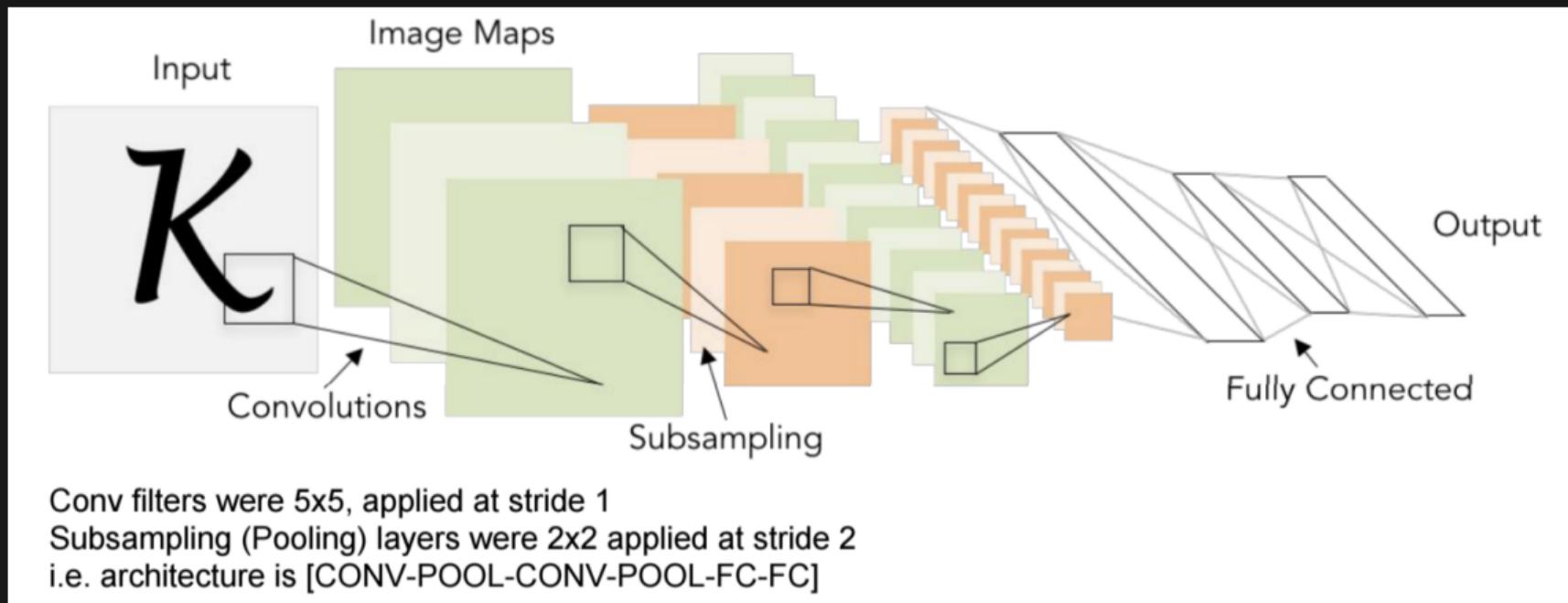
- Down-sample input size to reduce computation and memory
- Pooling by default is **performed on each slice separately**
 - hence output depth = input depth
 - max-pool, average-pool, (averaged) ℓ_p -norm pool
- Size and stride as in convolution; **no parameter**; typically no padding
- **Global pooling**: pooling size = input size

CNN Architecture



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

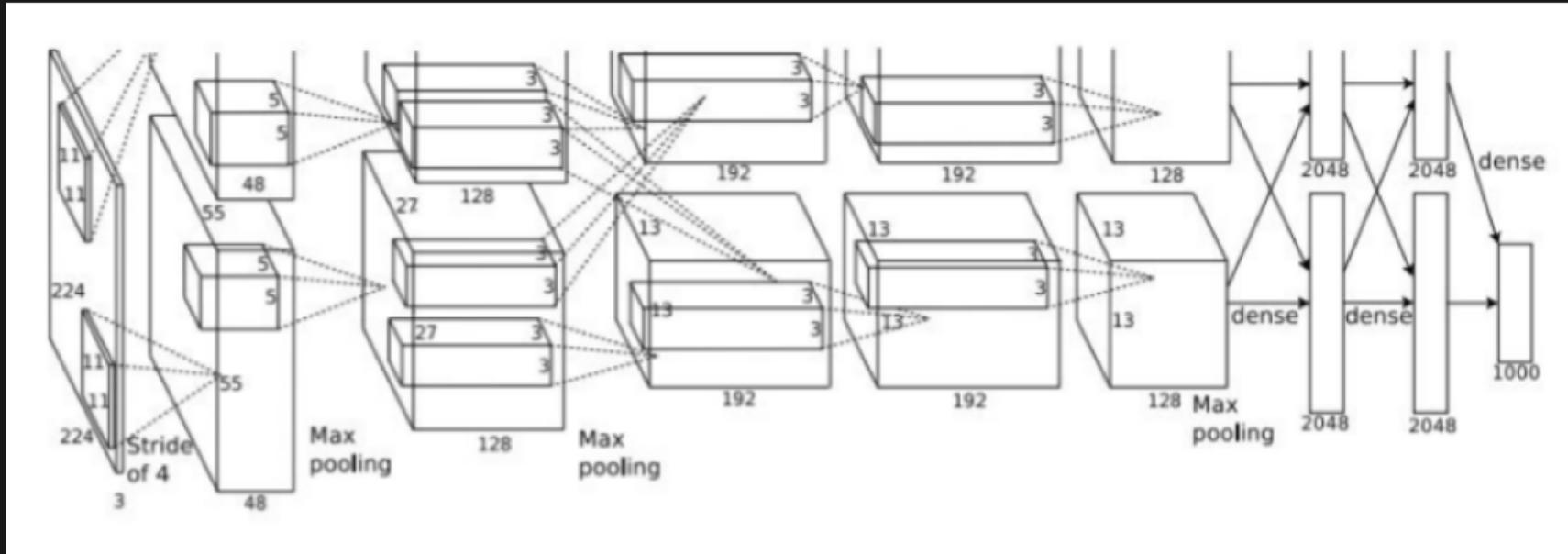
LeNet



Conv filters were 5x5, applied at stride 1

Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

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.

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

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

- 7x7 filter vs. 3x3 filter x 3 (stride 1)
 - both have receptive field 7x7
 - params: $O(7 \times 7)$ vs. $O(3 \times 3 \times 3)$
- nxn vs. nx1 followed by 1xn ?



INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800K$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400K$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200K$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

Note:

Most memory is in early CONV

Most params are in late FC

TOTAL memory: $24M * 4$ bytes $\approx 96MB / \text{image}$ (only forward! ~ 2 for bwd)

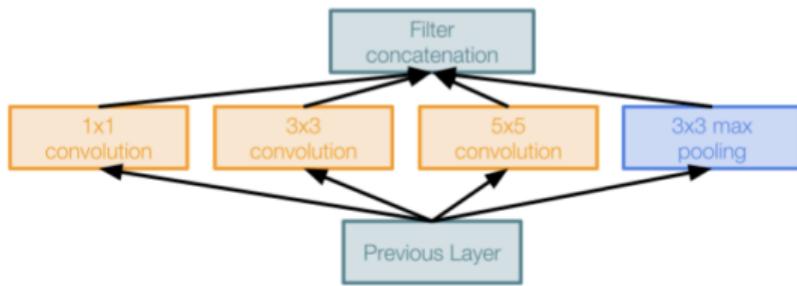
TOTAL params: 138M parameters

THAT'S NOT ENOUGH

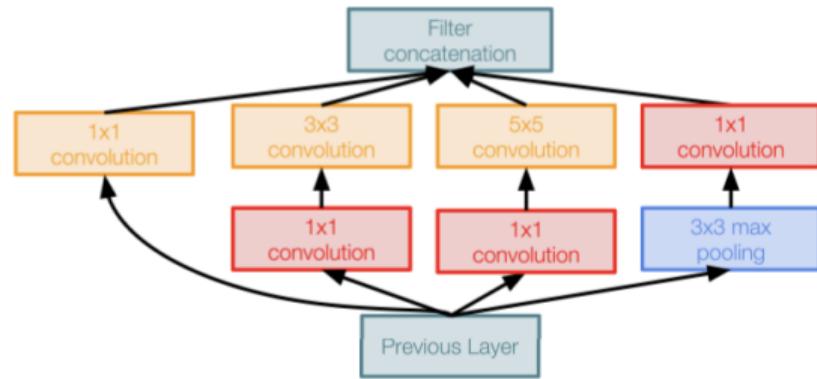
WE HAVE TO GO DEEPER

1x1 Convolution

1x1 conv “bottleneck” layers



Naive Inception module



Inception module with dimension reduction

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

naïve 3x3 conv

3x3 conv first, 1x1 conv second

1x1 conv first, 3x3 conv second

$m \times n \times c$

$3 \times 3 \times c \times k \rightarrow m \times n \times k$

$O(9mnck)$

$m \times n \times c$

$3 \times 3 \times c \times d \rightarrow m \times n \times d$

$1 \times 1 \times d \times k$

$O(9mncd + mnck)$

$m \times n \times c$

$1 \times 1 \times c \times d \rightarrow m \times n \times d$

$3 \times 3 \times d \times k$

$O(mnkd + 9mnck)$

output dim k > intermediate dim d input dim c > output dim k

1x1 conv first, 3x3 pool second

$m \times n \times c$

$1 \times 1 \times c \times k \rightarrow m \times n \times k$

$3 \times 3 \times k$

$O(mnck + mnk)$

3x3 pool first, 1x1 conv second

$m \times n \times c$

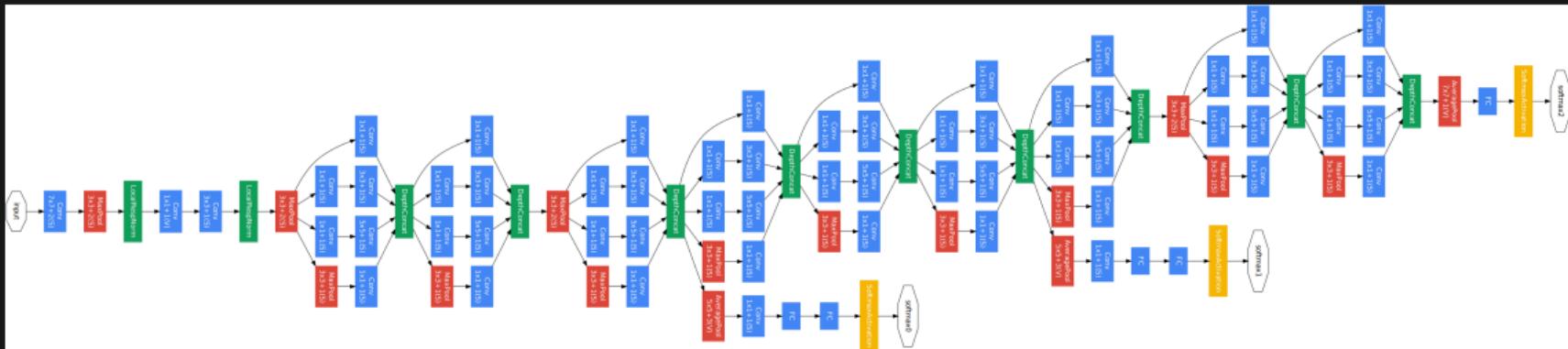
$3 \times 3 \times c \rightarrow m/3 \times n/3 \times c$

$1 \times 1 \times c \times k$

$O(mnc + mnck/9)$

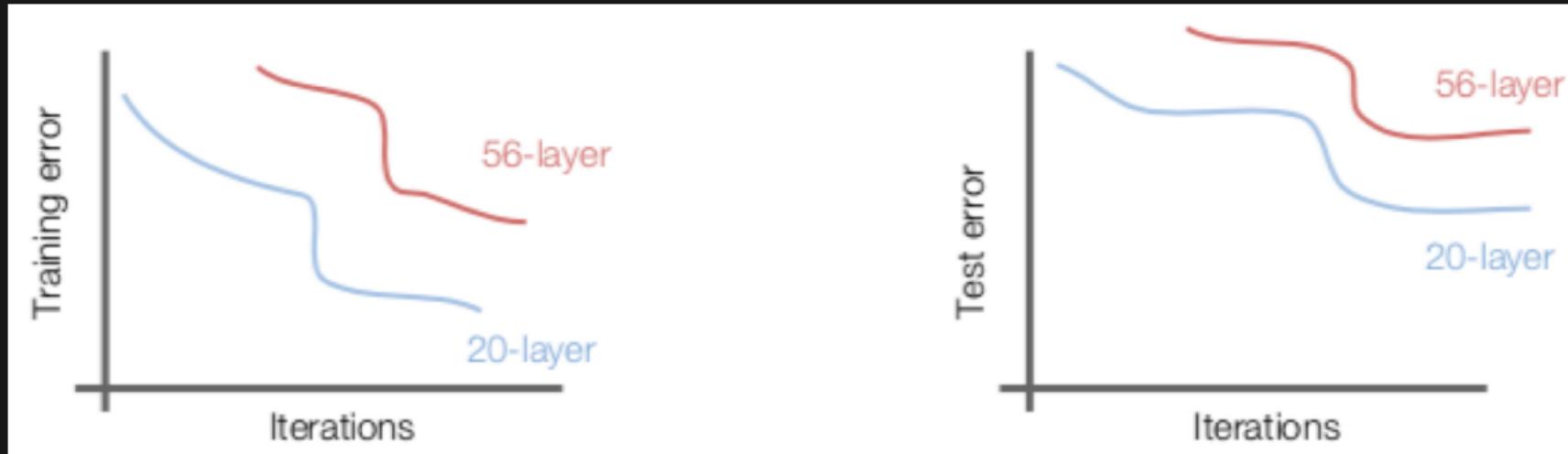
- 1x1 conv allows easy control of the depth (channels)
- Can save computation without affecting output size

GoogLeNet



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

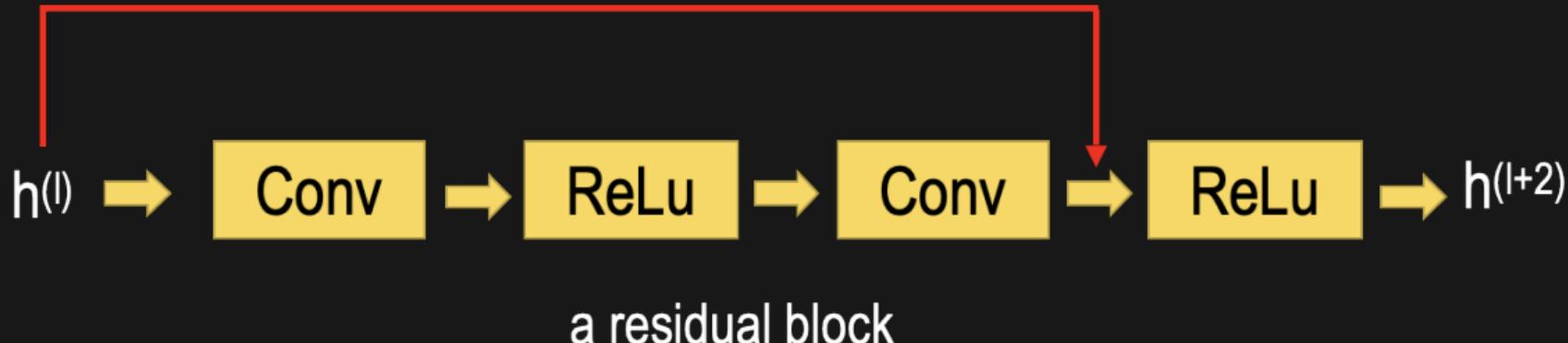
The Deeper, the Better, but More Difficult to Train



- 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 backpropagation 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)

