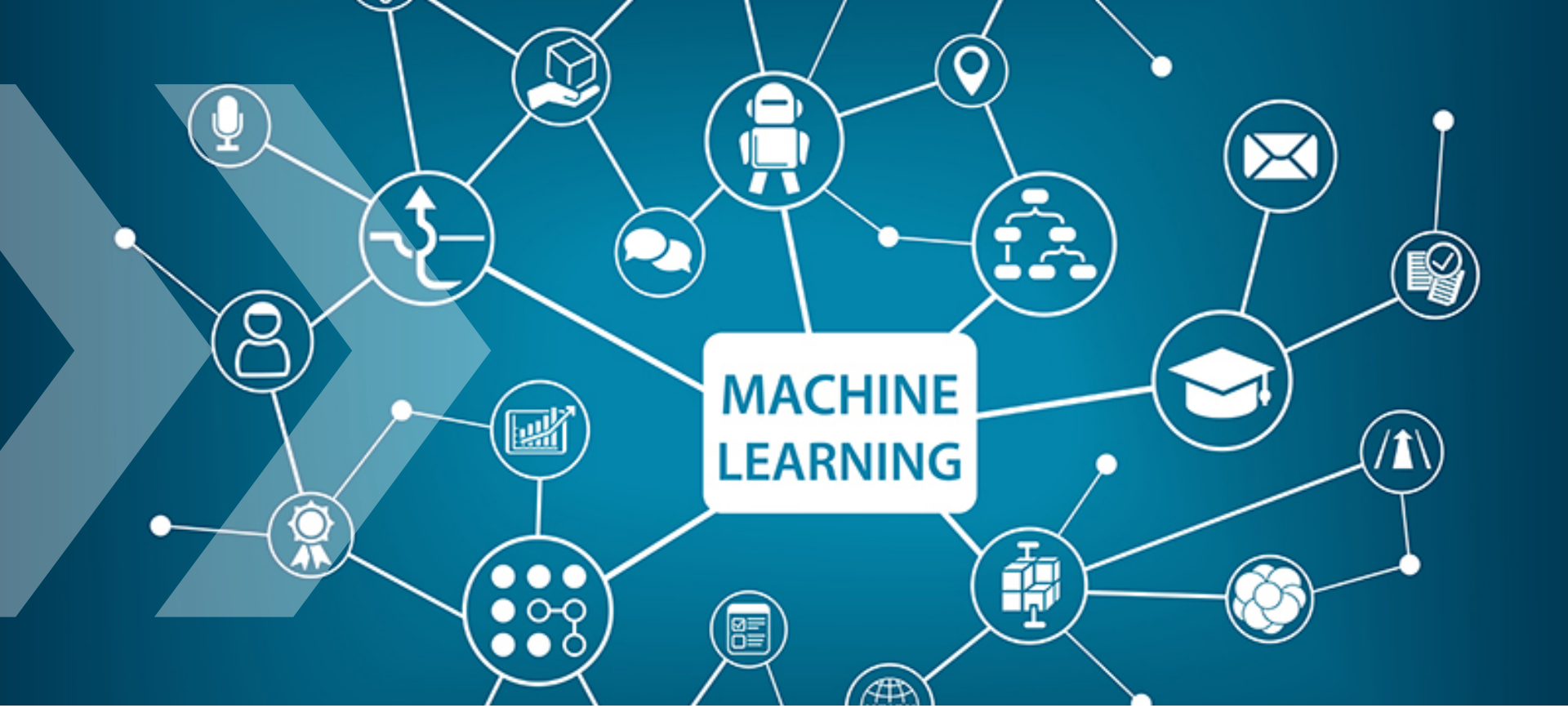


CS480/680: Intro to ML

Lecture 14: Convolutional Neural Networks (CNNs)

UNIVERSITY OF WATERLOO

some slides are adapted from Stanford cs231n course slides

Convolution

$$y(i) = (x * w)(t)$$

- Continuous case

$$y(i) = \int x(t)w(i-t)dt$$

- Discrete case

$$y(i) = \sum_{t=-\infty}^{\infty} x(t)w(i-t)$$

Convolutions for feature extraction

- In neural networks, a **convolution** denotes the linear combination of a **subset of units** based on a **specific pattern of weights**

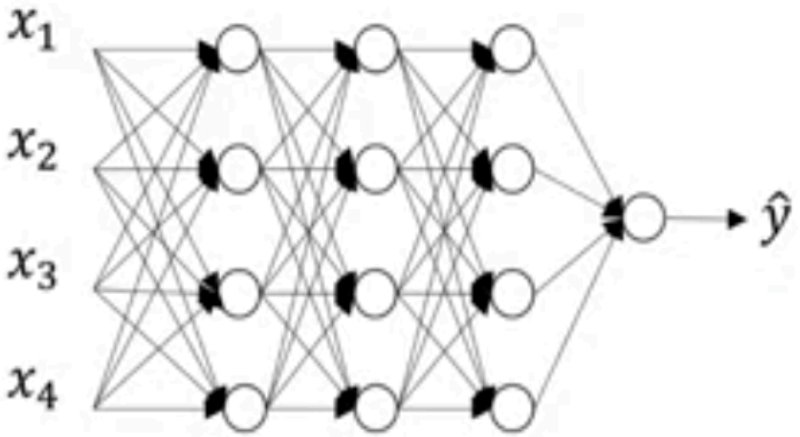
$$z_j = \sum_i w_{ji} h_i$$

- Convolutions are often combined with an activation function to produce a feature

$$h_j = f(z_j) = f\left(\sum_i w_{ji} h_i\right)$$

Recap: MLPs

- Hidden unit is fully connected to all units in the previous layer
- Units in a hidden layer do not share any connections
- MLPs do not scale well to full images
  - E.g., consider an image with the size 200x200x3. Then each hidden unit in layer 1 would have 200*200*3 = 120,000 weights
  - Huge number of parameters would lead to overfitting



MLP

Convolutional neural networks (CNNs)

- CNNs are primarily used in the field of pattern recognition with images, which aim to exploit the strong spatially local correlation present in natural images

- Instead of using fully connected weights, a convolutional layer has kernels (or filters) that are only connected to a small region of the previous layer via dot products (similar to discrete convolutional operator)

- Each filter can be thought of as a feature identifier

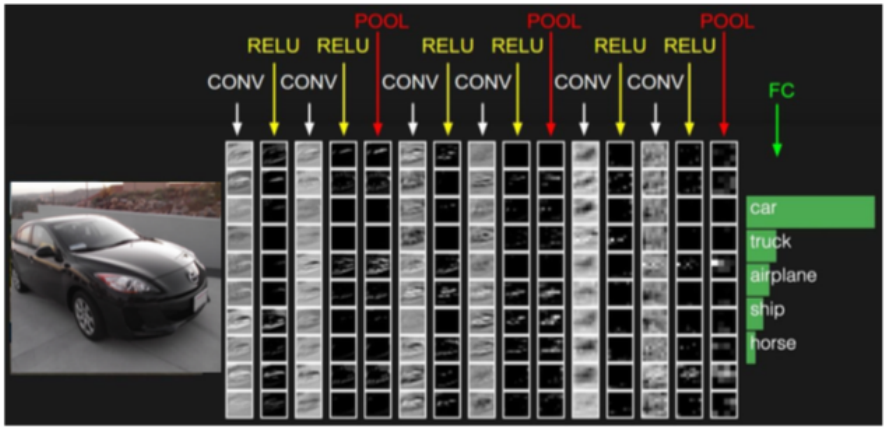
Input/output dimension

Consider the input as images.

- Units are arranged in 3 dimensions
  - **Width, height, and depth**
  - For classification problem, the final output layer would have dimensions 1x1xC for class scores, where C is the number of classes

Layers in CNNs

- Fully-connected layer (FC)
  - ➢ Each unit in this layer is connected to all units in the previous layer.
- Convolutional layer (Conv)
- Pooling layer



UNIVERSITY OF WATERLOO

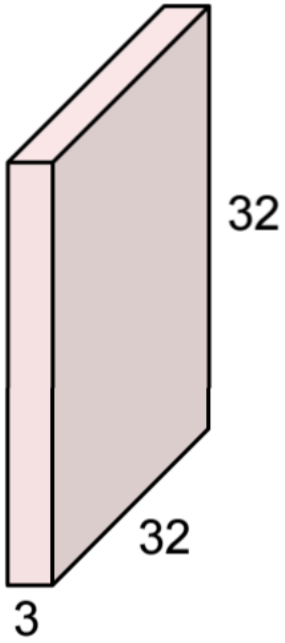
Conv layers: filter

Include a set of filters

- Local connectivity: The connections are local in space (along width and height), but always full along the entire depth of the input volume

- Hyperparameter: receptive field of unit (or filter size)

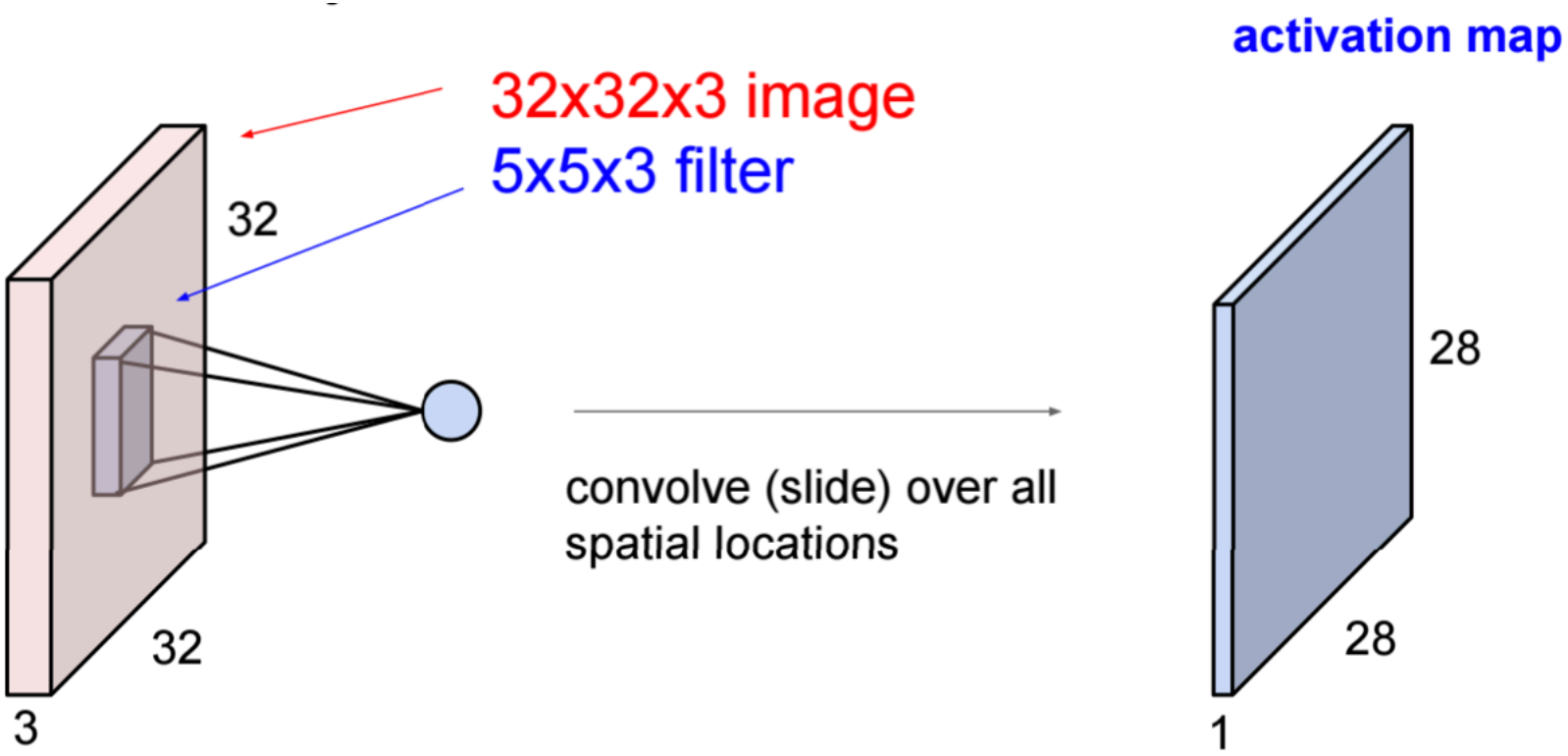
32x32x3 image



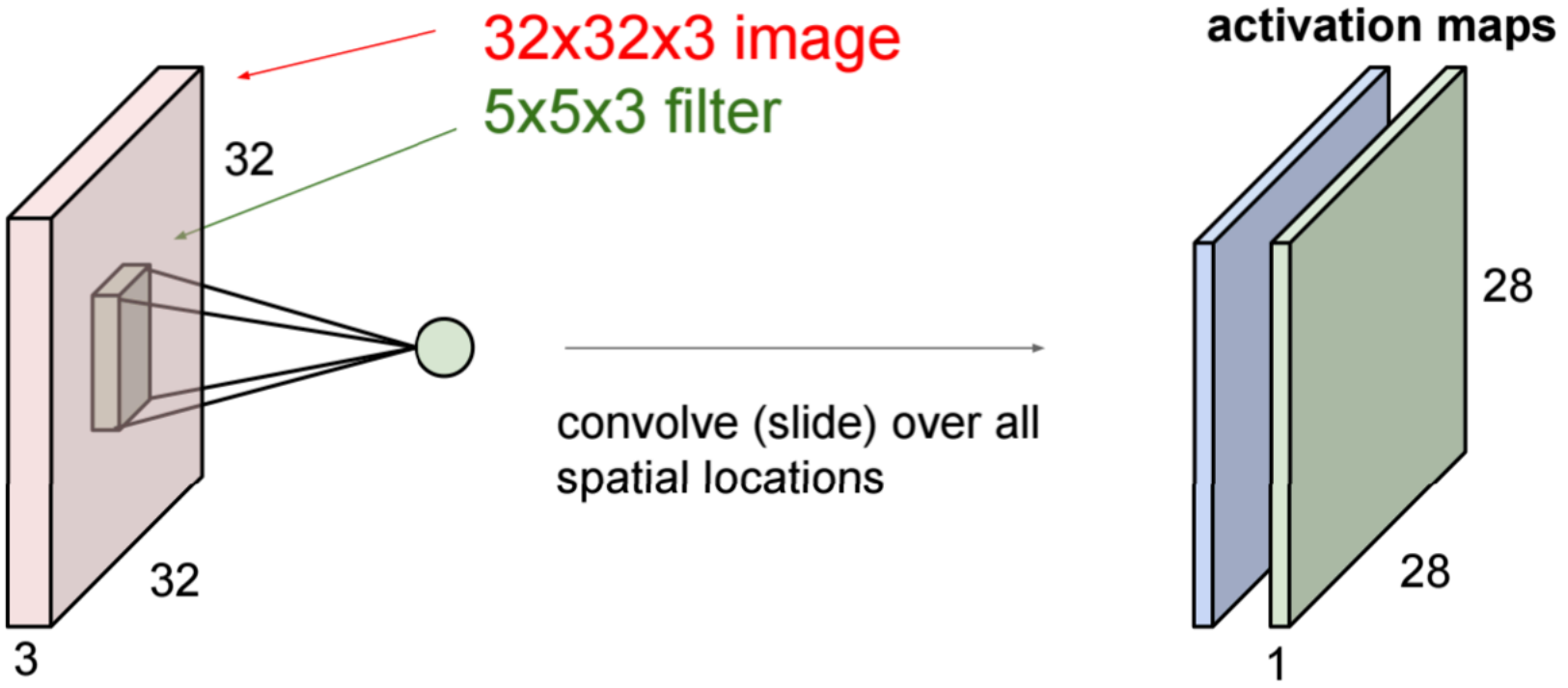
5x5x3 filter

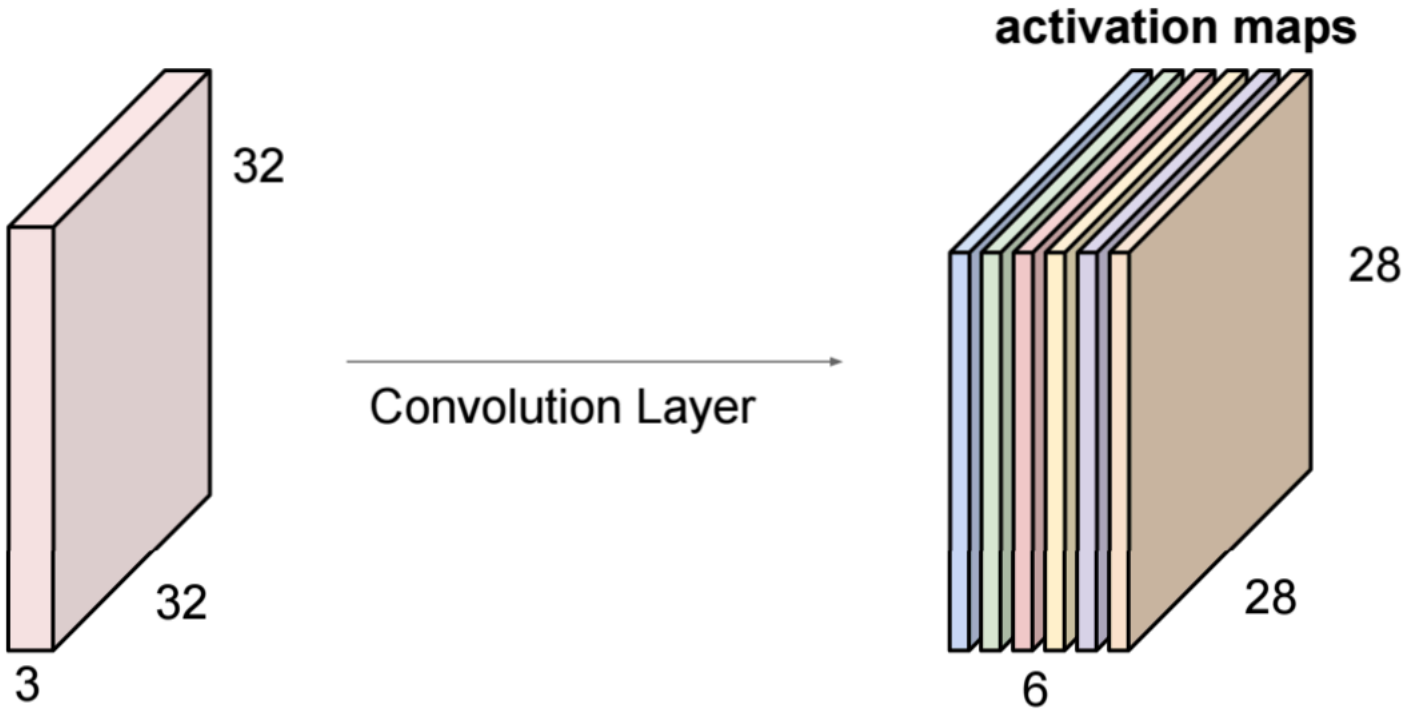
Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

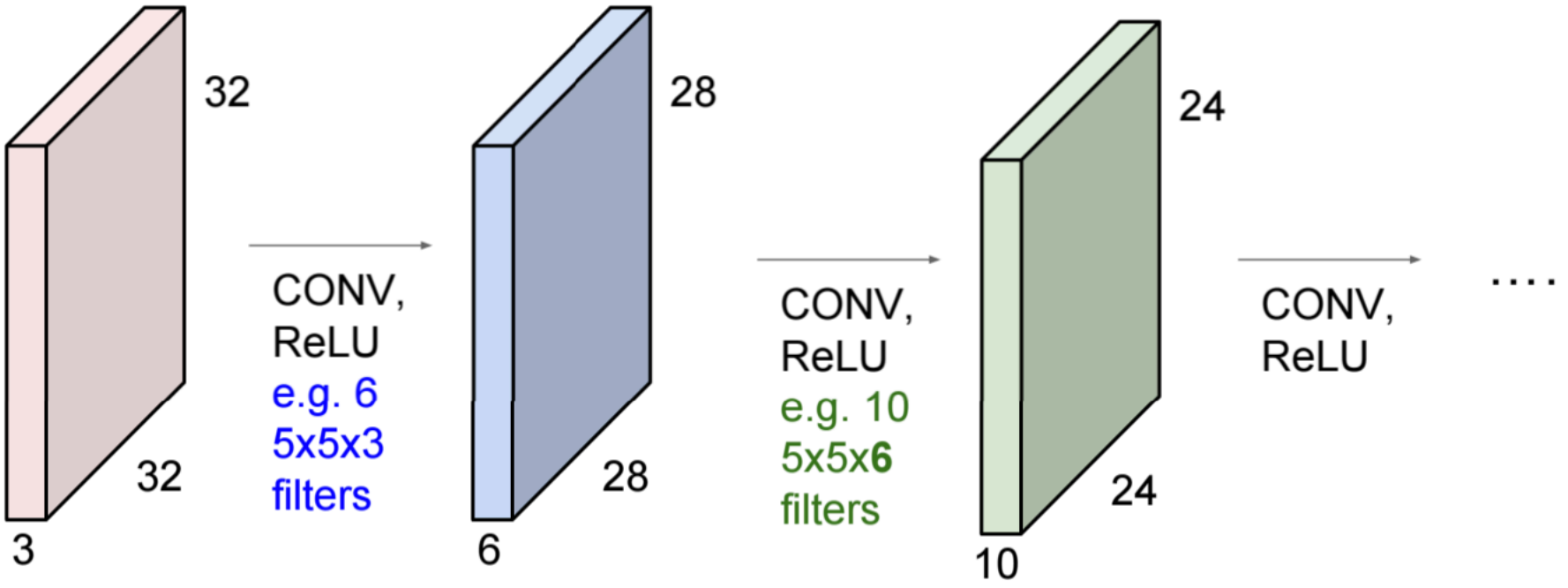
Main idea 1: restricted receptive field

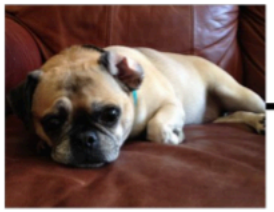


Main idea 2: weight sharing







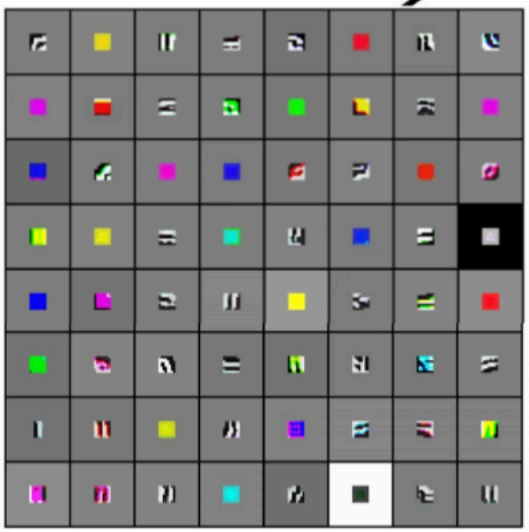


Low-level features

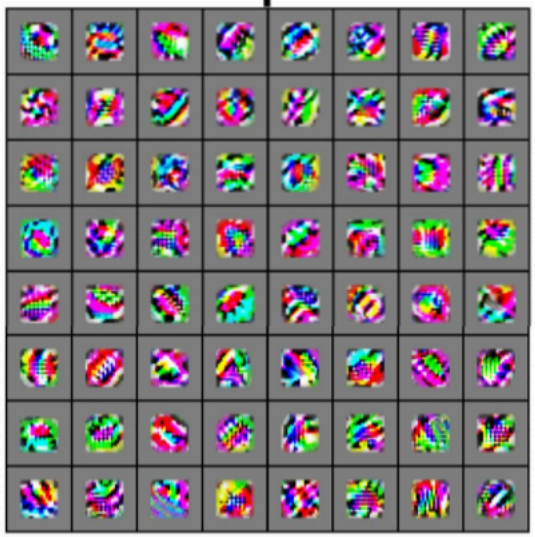
Mid-level features

High-level features

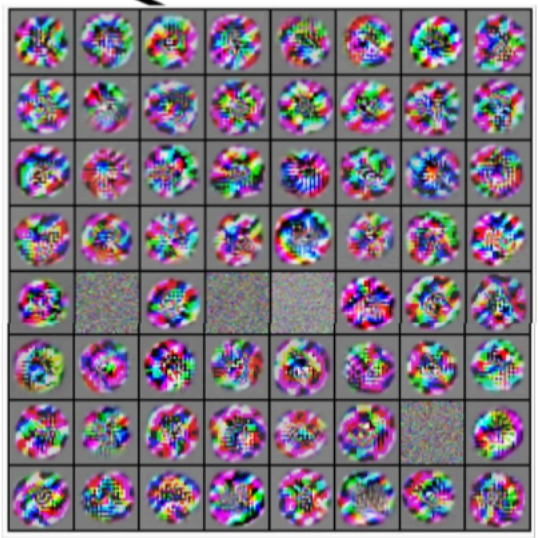
Linearly separable classifier



VGG-16 Conv1_1



VGG-16 Conv3_2



VGG-16 Conv5_3

Conv layers: output

Three hyperparameters control the output size

- **Depth:** the number of filters (determine the depth of output)
- **Stride:** how to slide the filters
  - Stride = 1: move the filters 1 pixel at a time
  - Stride = 2: move the filters 2 pixels at a time
- **Zero-padding:** pad the input with zeros around the border
  - Allow us to control the spatial size of the output (most commonly we preserve the spatial size of the input)

Example

The input volume has size 32x32x3, (e.g. an RGB CIFAR-10 image). If the filter size is 5x5, then how many weights does each unit in the Conv layer have?

Spatial size of output

Spatial size: (W+2P-F)/S+1 (output should be an integer)

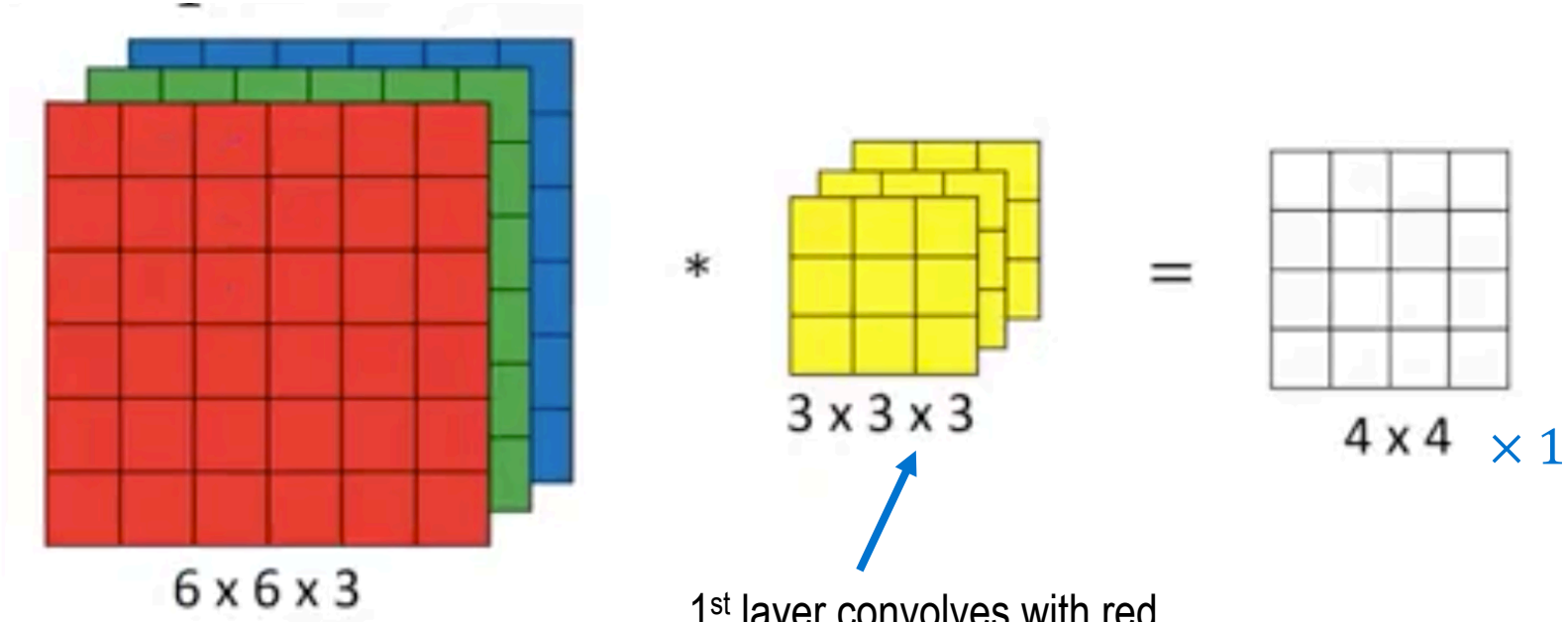
- W: the input size (width or height; assume they have the same dimension)
- F: the filter size (width or height; assume they have the same dimension)
- P: the amount of zero padding used on the border
- S: stride

In general, set P=(F−1)/2 when S=1 to make sure that the input and output have the same spatial size

Spatial size: 2-D examples

- For a 7x7 input and a 3x3 filter with stride 1 and pad 0, what is the spatial size of the output? With stride 2?

Convolutions on 3-D



The convolution operation essentially performs dot products between the filters and local regions of the input

Output size: 3-D examples

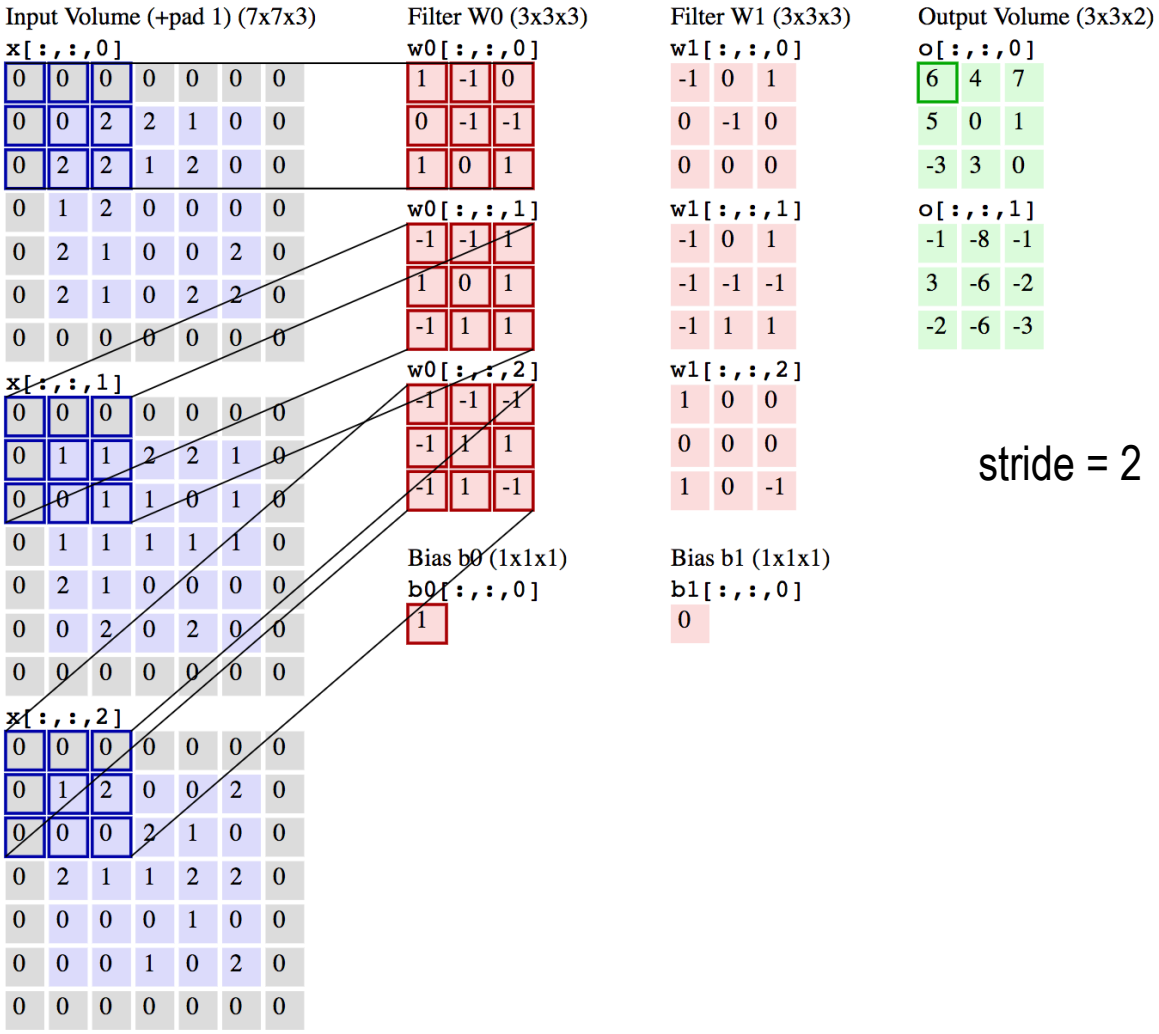
- Consider the input size as 227x227x3. In the first conv layer, use 96 filters each with the size F=11, stride S=4 and no zero-padding P=0. What is the output size of this conv layer?

Parameter sharing

- Motivation: control the number of parameters
  - Use the previous example. Consider 55*55*96 = 290,400 units in the first conv layer, and each has 11*11*3 = 363 weights and 1 bias. In total, 290400 * 364 = 105,705,600 parameters in the first layer.

- Idea: constrain the units in each depth slice to use the same weights and bias.
  - With parameter sharing, the first conv layer would now have only 96 unique set of weights (one for each depth slice). In total, 96*11*11*3 + 96 = 34,944 parameters.

Illustration



UNIVERSITY OF WATERLOO

Summary of Conv layer

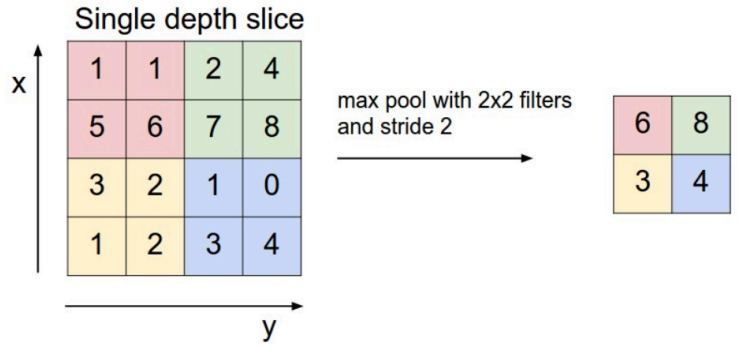
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it 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.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Difference between FC and Conv layers

- Units in the Conv layer are connected only to a local region in the input
- Units in the Conv layer share parameters

Pooling layers

- Idea: perform a down-sampling operation along the spatial dimensions (width and height)
  - Reduce the amount of parameters and computation in the network
  - Control overfitting
- Common operations
  - Max-pooling (most popular): e.g., use max-pooling with 2x2 filter size (i.e. F=2), and with stride of 2 (i.e. S=2)
  - Average pooling
  - L2-norm pooling



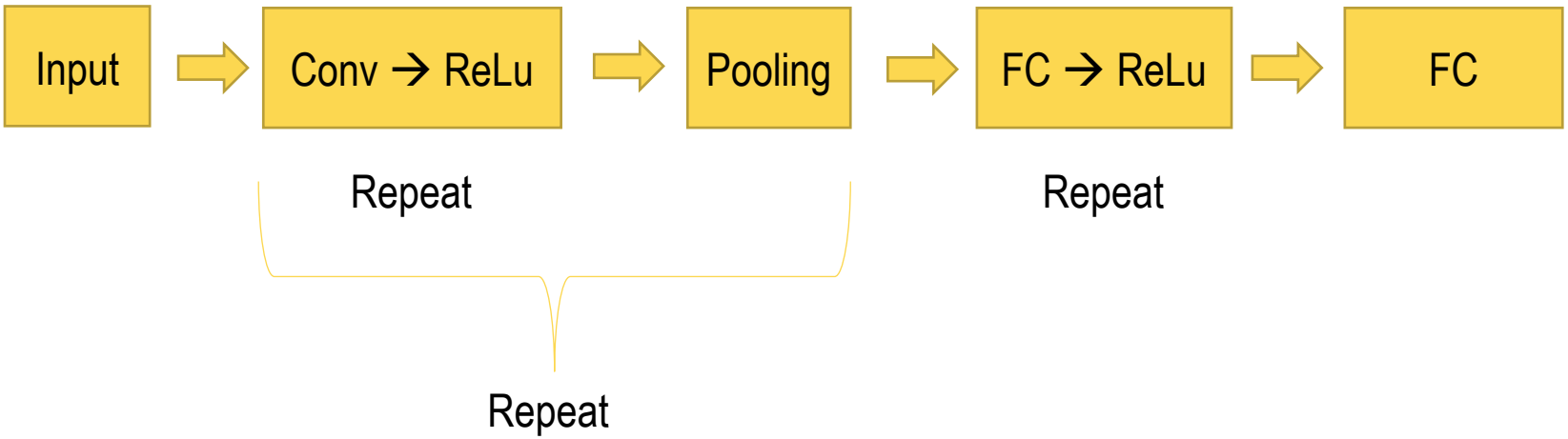
Summary of pooling layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires two hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Example

Consider the input size as 55*55*96. This input is fed to a pooling layer with 3*3 filters and a stride of 2. What is the output size of this pooling layer? How many parameters in this layer?

ConvNet architecture

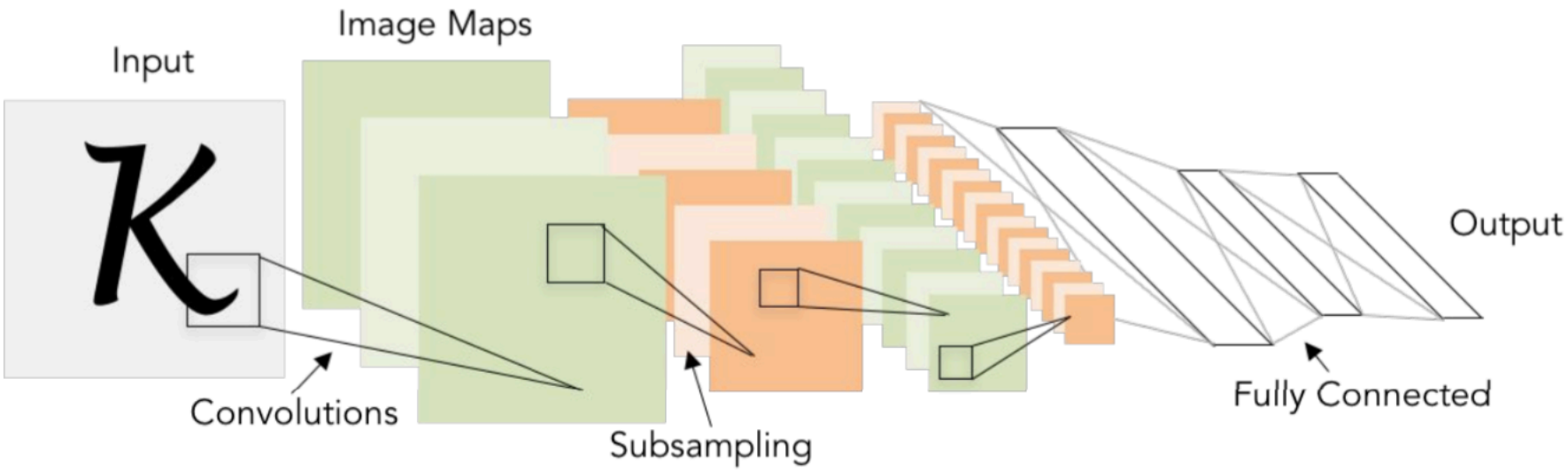


Several famous architectures:
LeNet, AlexNet, ZFNet, GoogLeNet, VGGNet, ResNet, etc.

Try different network structures and find one that suits your problem

LeNet

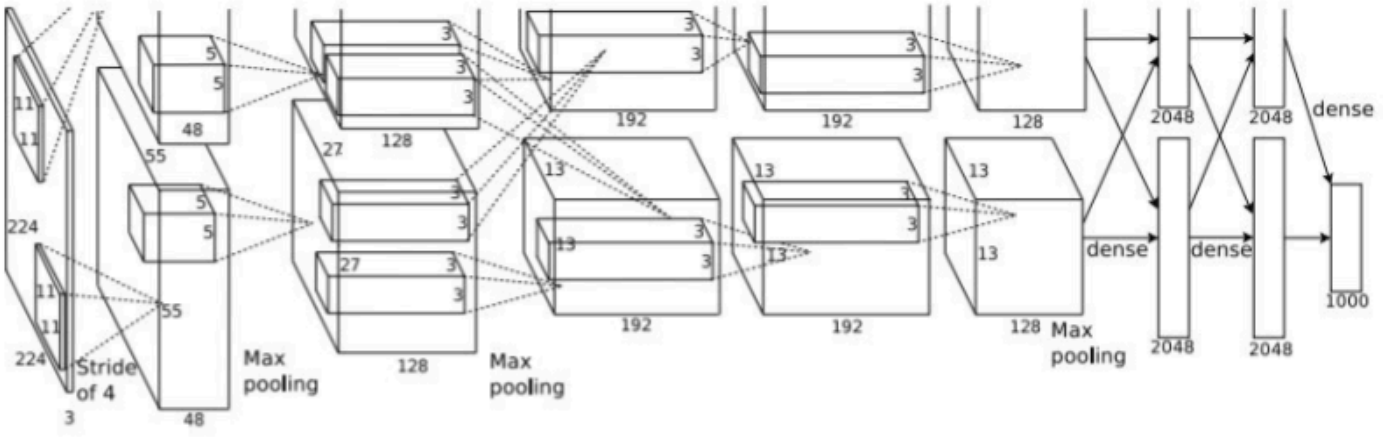
[LeCun et al., 1998]



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

[Krizhevsky et al. 2012]



Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

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

[Simonyan and Zisserman, 2014]

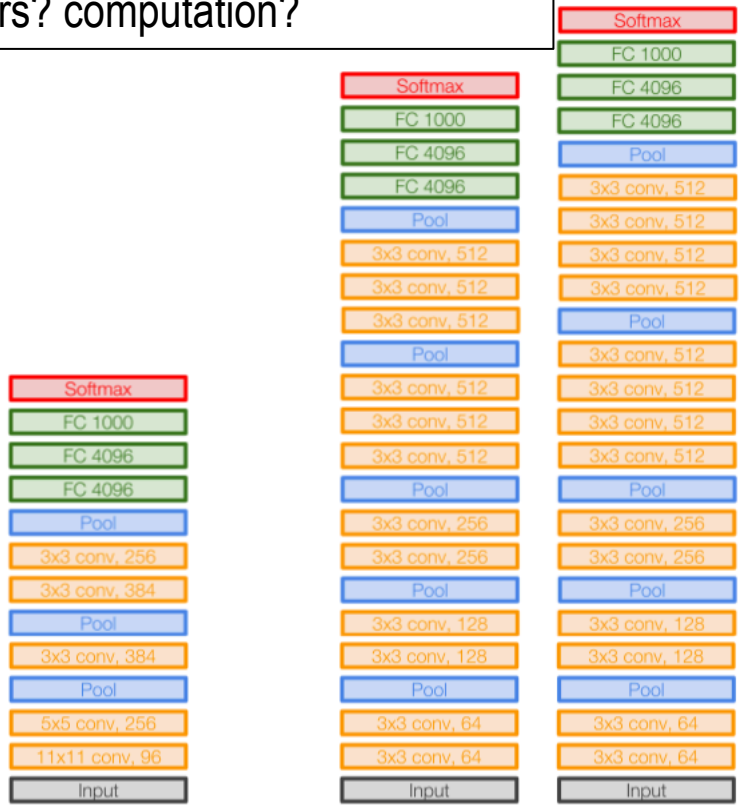
Compare: nxn-conv vs 1xn-conv followed by nx1-conv
receptive field? parameters? computation?

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-conv

receptive field: 7 x 7
params/comp: O(7*7)

3x3-conv, x3

receptive field: 7 x 7
params/comp: O(3*3*3)

receptive field: number of input pixels each final output unit can "see"

VGGNet

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
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

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

Note:

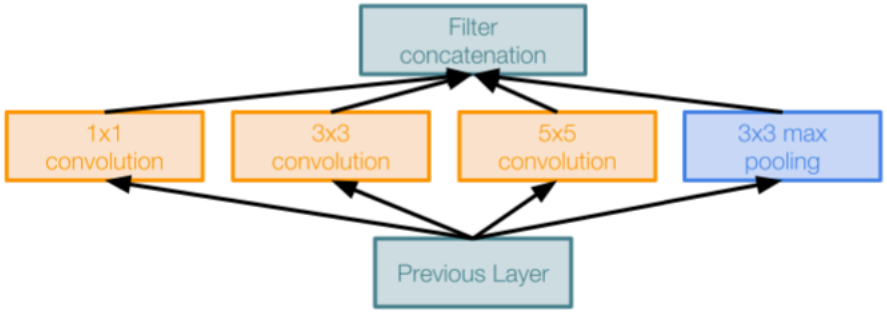
Most memory is in early CONV

Most params are in late FC

GoogLeNet

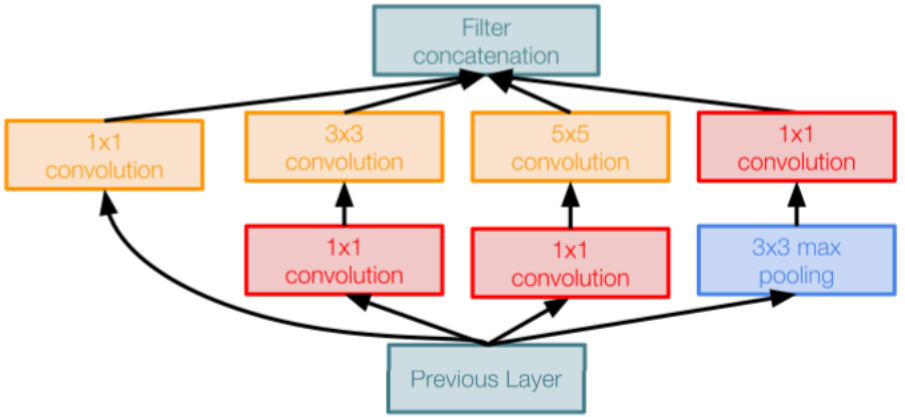
GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers



Inception module with dimension reduction

naïve 3x3 conv

m x n x c
3 x 3 x c x k --> m x n x k

O(9mnck)

3x3 conv first, 1x1 conv second

m x n x c
3 x 3 x c x d --> m x n x d
1 x 1 x d x k

O(9mncd + mndk)

1x1 conv first, 3x3 conv second

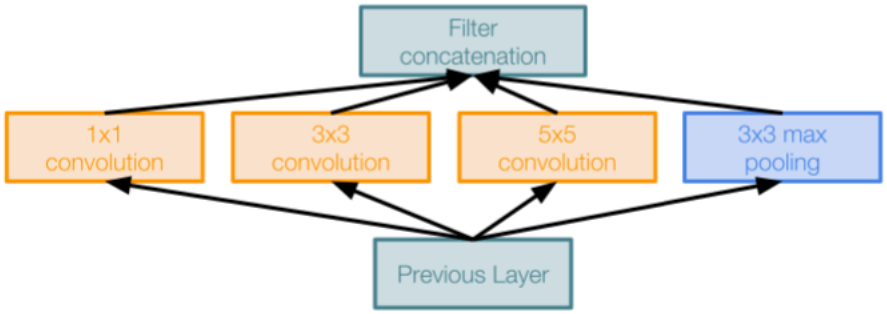
m x n x c
1 x 1 x c x d --> m x n x d
3 x 3 x d x k

O(mncd + 9mndk)

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

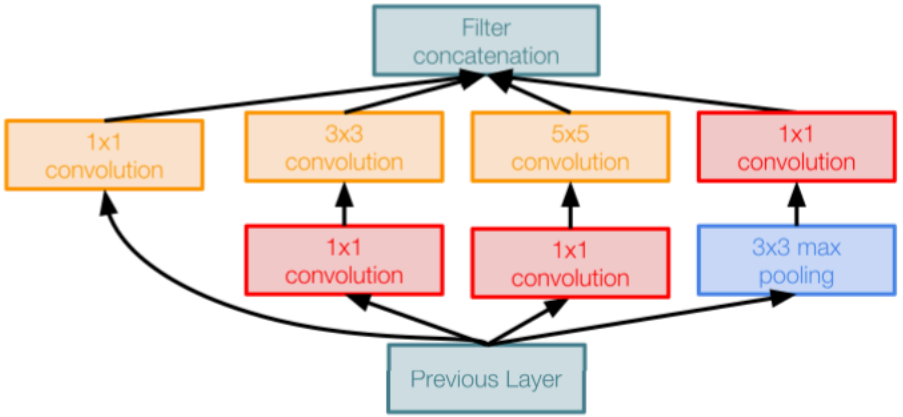
GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers



Inception module with dimension reduction

1x1 conv first, 3x3 pool second

m x n x c
1 x 1 x c x k --> m x n x k
3 x 3 x k

O(mnck + mnk)

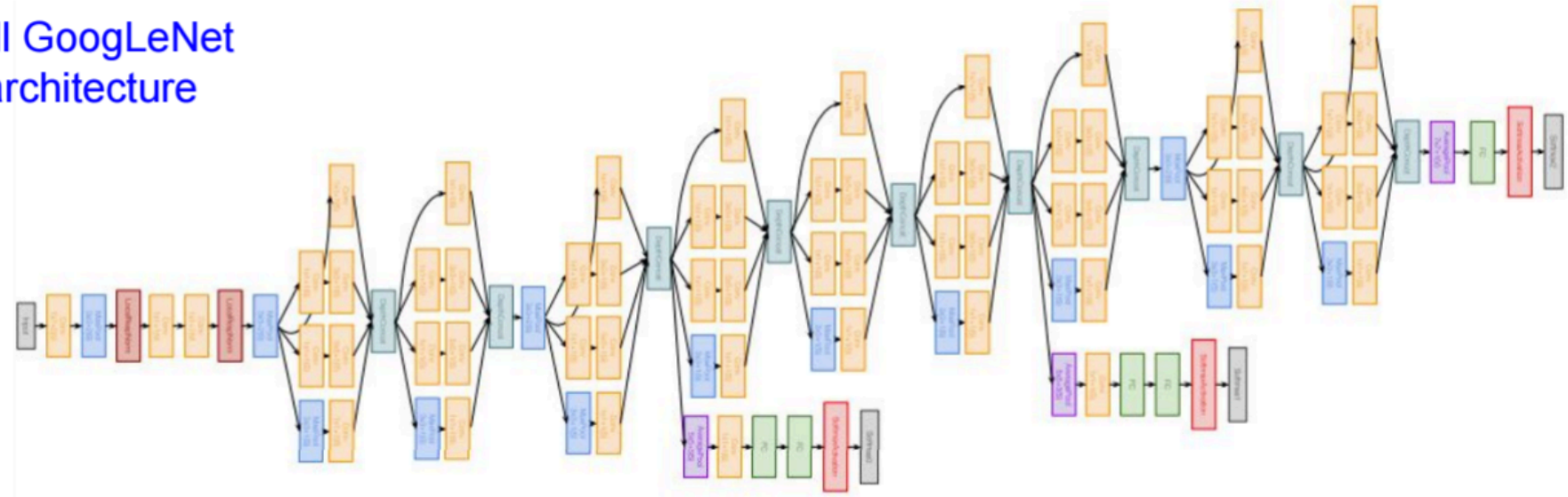
3x3 pool first, 1x1 conv second

m x n x c
3 x 3 x c --> m/3 x n/3 x c
1 x 1 x c x k

O(mnc + mnck/9)

GoogLeNet

Full GoogLeNet architecture

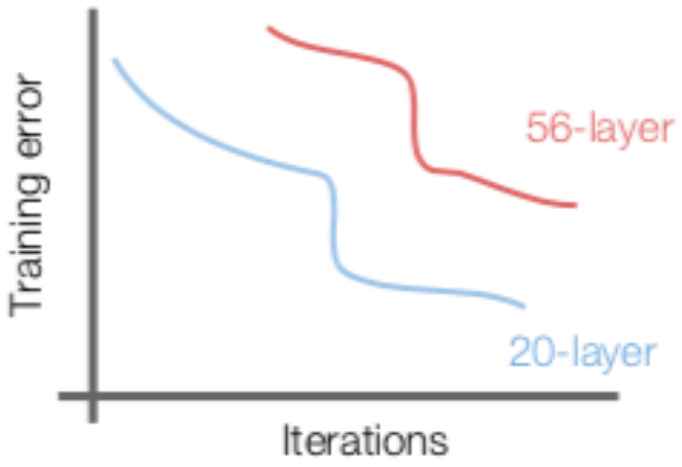


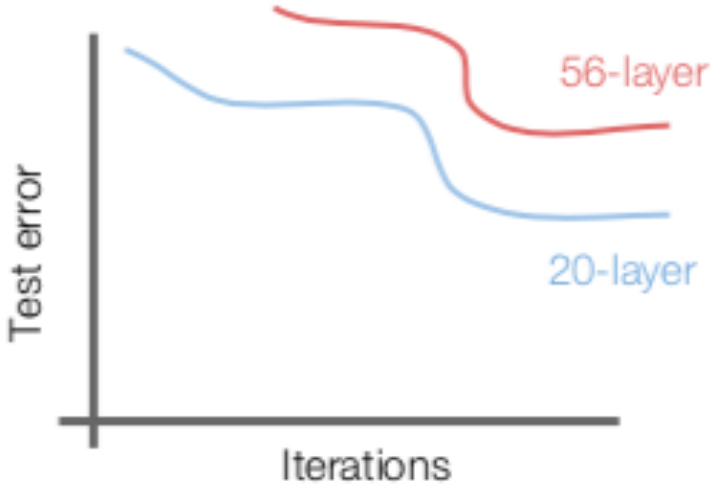
- Deeper but more efficient
- No FC layers
- Better performance

Case studies: ResNet

Fact: deeper models are harder to optimize

(e.g., gradient vanishing and exploding)

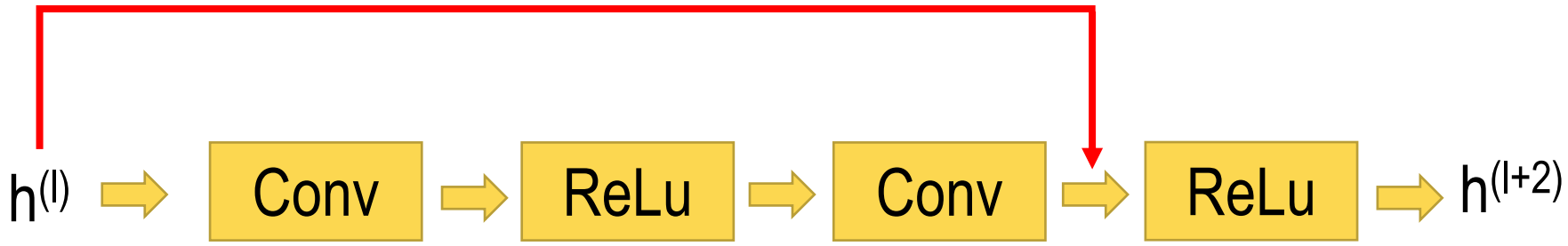




increasing network depth leads to worse performance in reality

Solution

Add an ``identity shortcut connection'' that skips one or more layers



a residual block

ResNet: very deep networks using residual blocks

- Can support up to hundreds or even thousands of layers and still achieve compelling performance
- Swept all classification/detection competitions in ILSVRC'15 and COCO'15

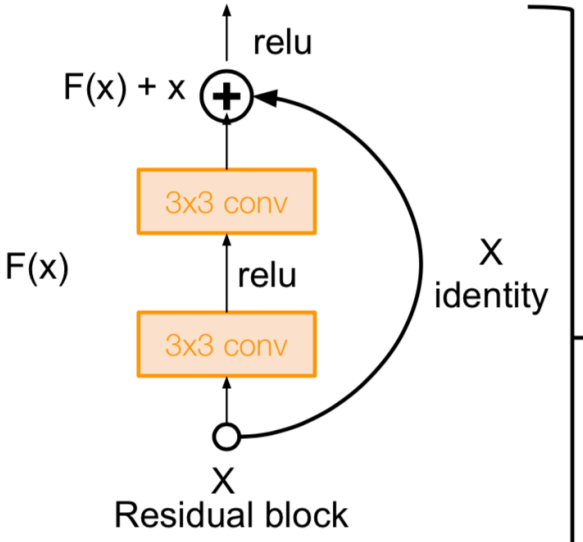
Illustration

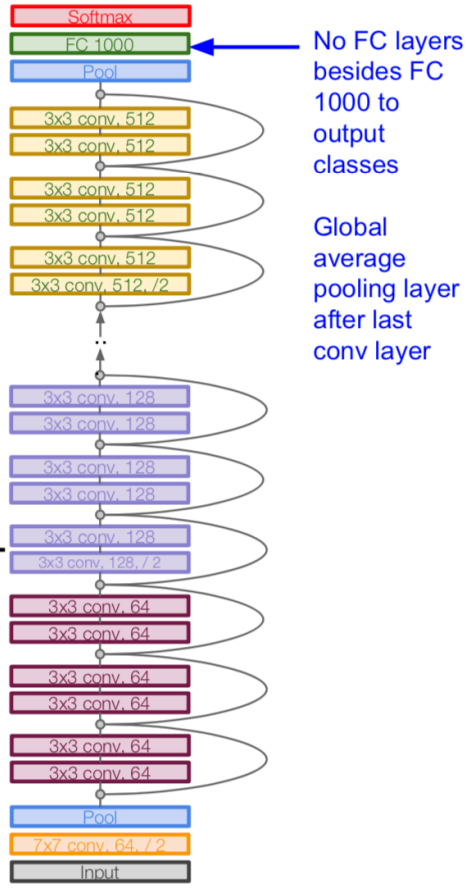
Case Study: ResNet

[He et al., 2015]

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)





UNIVERSITY OF WATERLOO

Questions?