Lec12 Deep Learning
12-3 Convolutional Network

Yang Shu

School of Data Science and Engineering

yshu@dase.ecnu.edu.cn

[Acknowledgement: Slides are adapted from Deep Learning Course, Mingsheng Long, THU]

Outline

- **Convolutional Neural Network**
- Practical Training Strategies
  - Weight Initialization
  - Batch Normalization
  - Data Augmentation
- Architecture Revolution

Computer Vision

Nose, Eyes, Mouth

A beautiful woman

Wheels, License Plate, Headlights

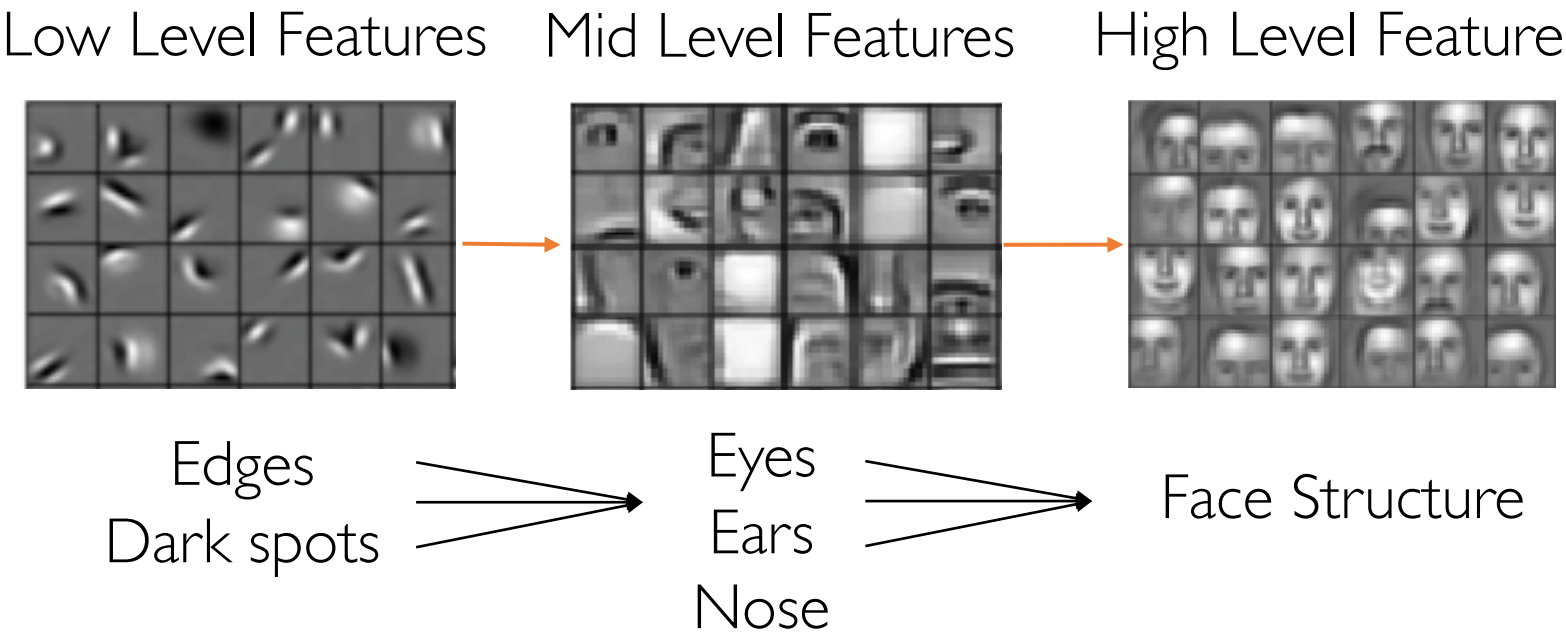
An vehicle off the road

Door, Windows

A simple villa

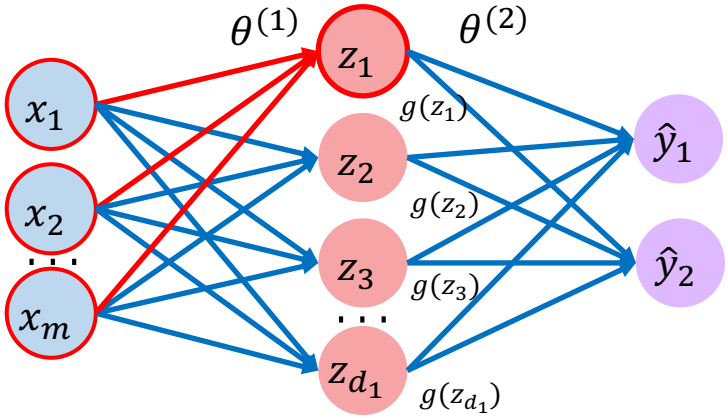
Humans are good at making multi-aspect descriptions of an image, but machines cannot.

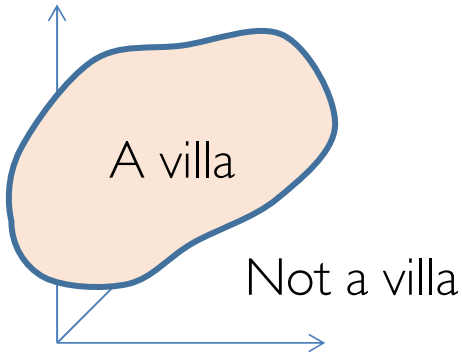
Hierarchical Representation Learning



To mimic the human visual system, we need to learn hierarchy of feature representations directly from data without hand engineering

MLP for Vision





Dimensions
372x372

Input

Hidden Layer

Output

Result

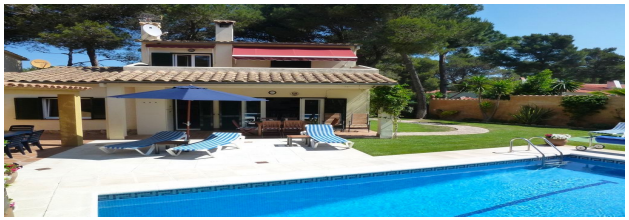
Spatial Info lost

Input
N= 138,384

k hidden layers
Layer 1: α×138,384
Layer 2: β×138,384
Layer 3: γ×138,384

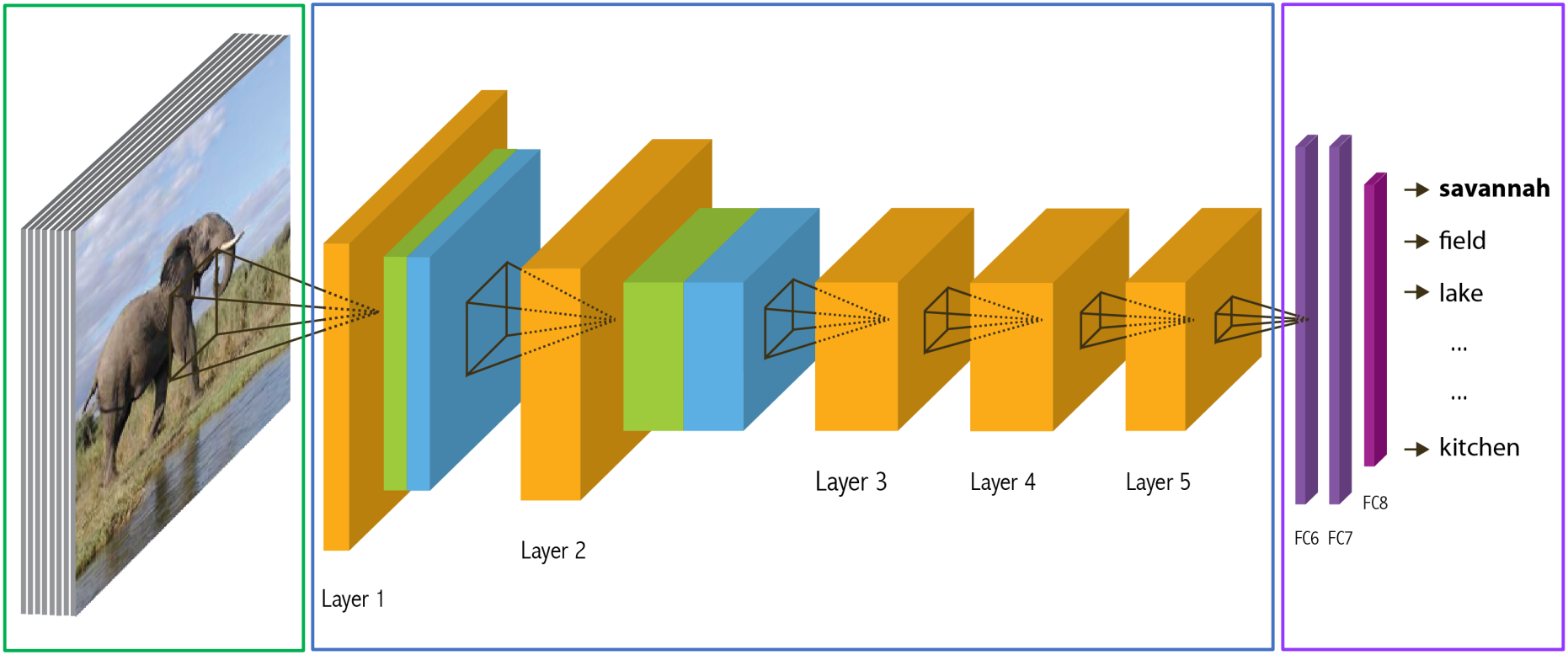
Output

Parameter explosion



It is hard to **adapt** scenarios of viewpoint variation, scale variation, illumination conditions and so forth.

Convolutional Neural Network

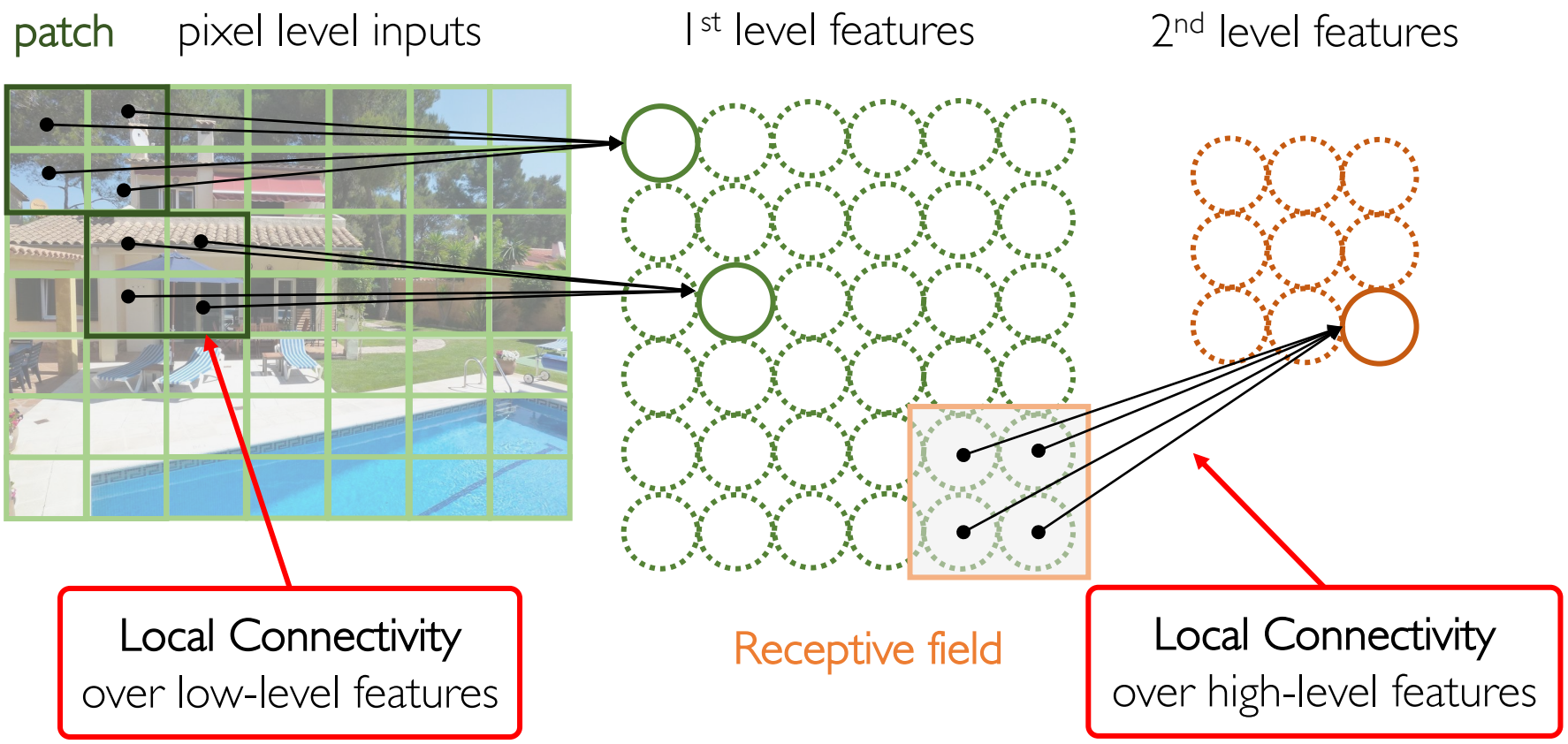


Scan
the image

Generate
hierarchy of features

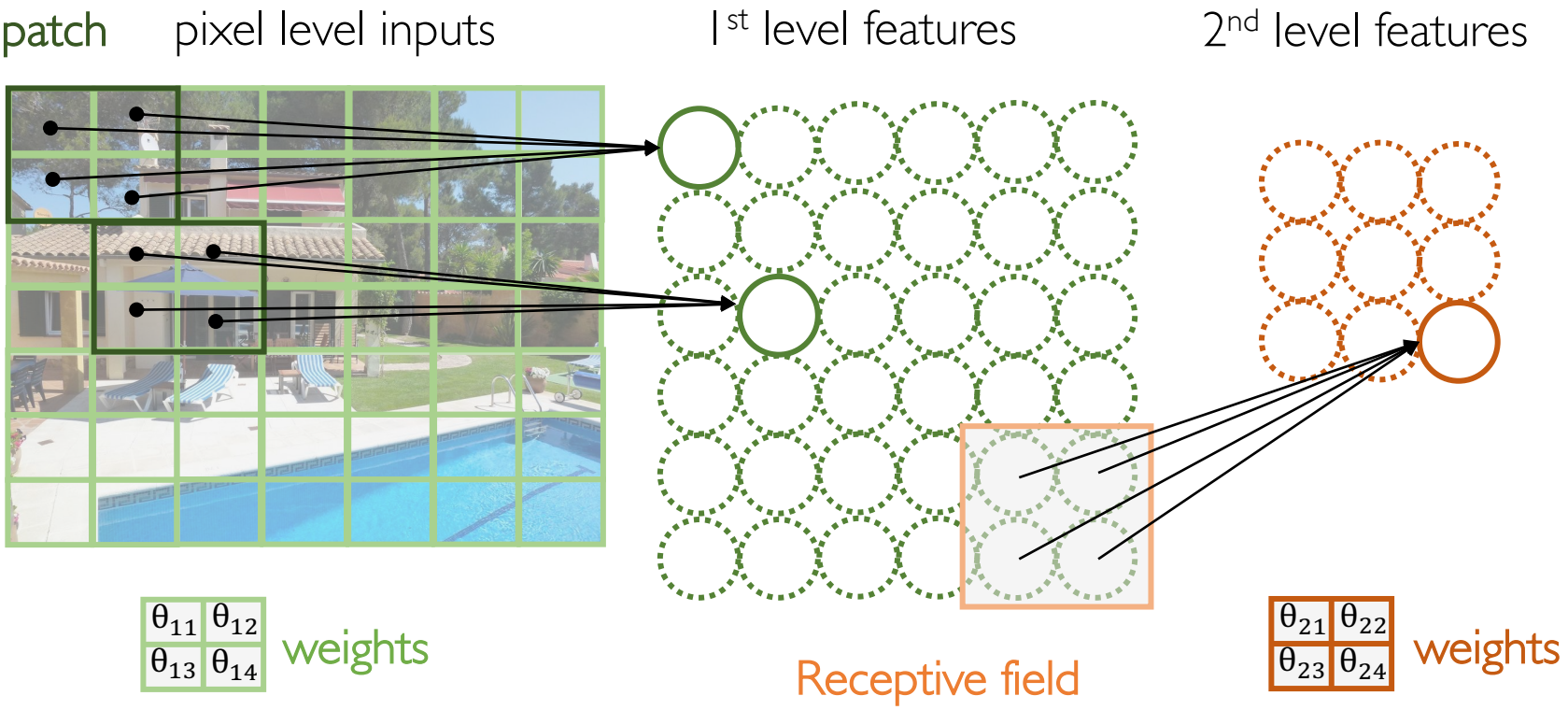
Recognize from
high level features

Idea 1: Local Connectivity



- **Locality Assumption:** local information is enough for recognition
- Connect each neuron to only a local region of the input volume

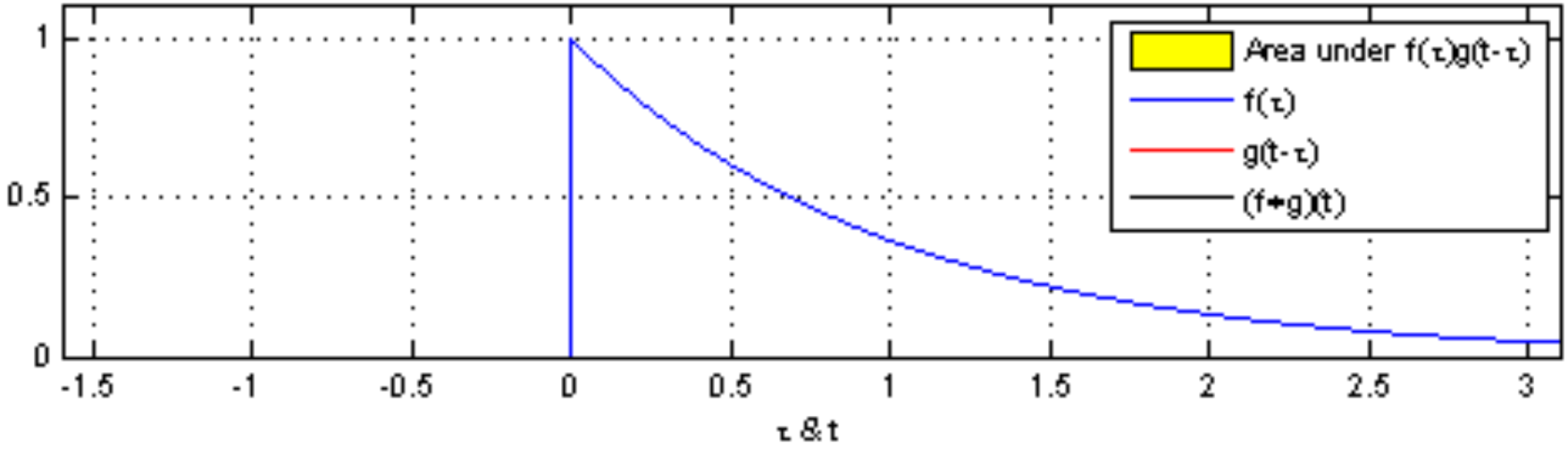
Idea 2: Parameter Sharing



- **Shift Invariance Assumption:** If a feature is useful at spatial position (x, y), then it should also be useful for other positions (x', y')
- Share the weights of sliding window over different spatial locations

Convolution

- Continuous functions: $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \mathrm{d}\tau$
- Discrete functions: $(f * g)(n) = \sum_{m=-\infty}^{\infty} f(m) g(n - m)$
- Weighting function g emphasizes different parts of input function f as t varies. **Convolution** shows how the shape of f is modified by g.

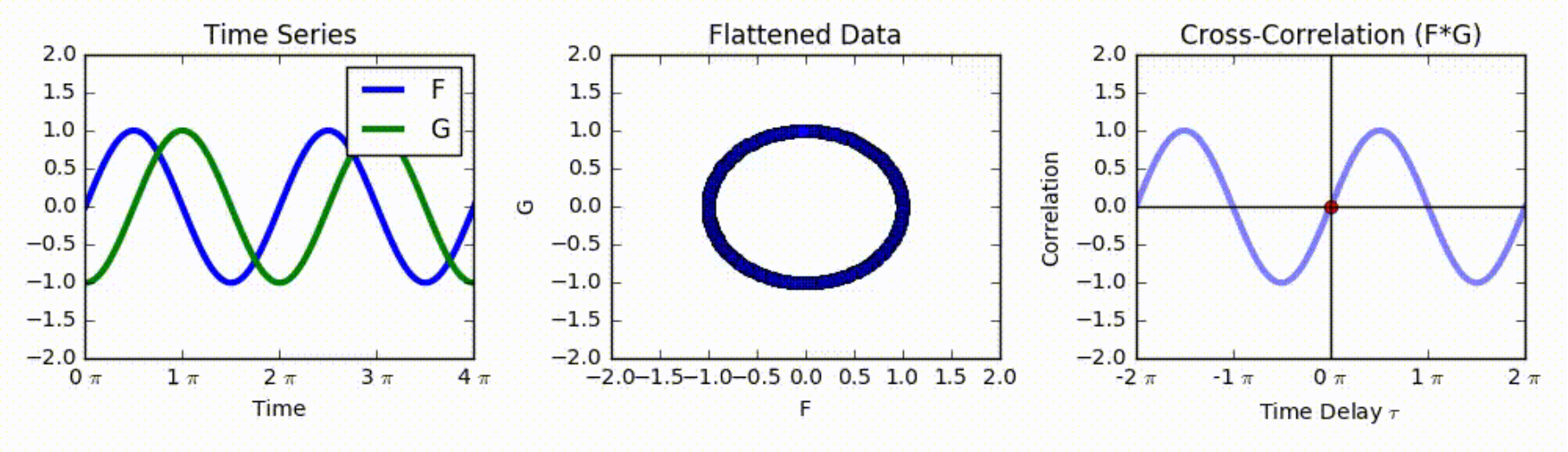


Cross-Correlation

- **Cross-correlation** is a similarity measure of two series $f(\tau), g(\tau)$ as a function of the lag (displacement) of one relative to the other.

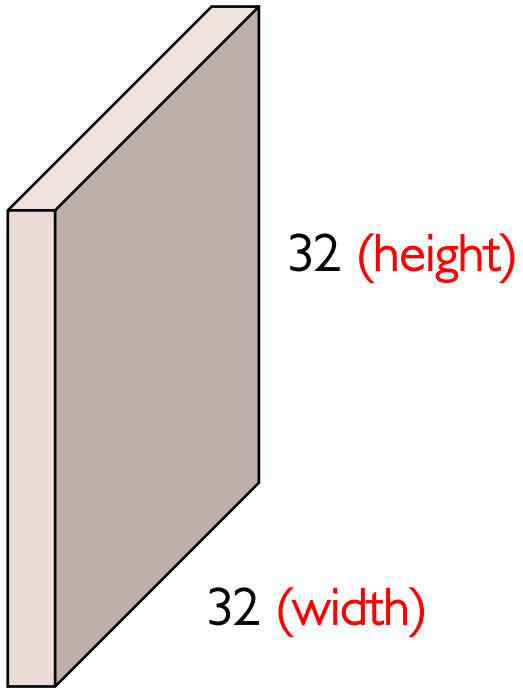
$$(f \star g)(\tau) = \int_{-\infty}^{\infty} \overline{f(t)} g(t+\tau) \mathrm{d}t$$

$$(f \star g)(n) = \sum_{m=-\infty}^{\infty} \overline{f(m)} g(m+n)$$



Notations

32x32x3 image (width x height x RGB)



32 (height)

32 (width)

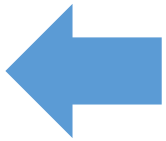
3

depth

channels

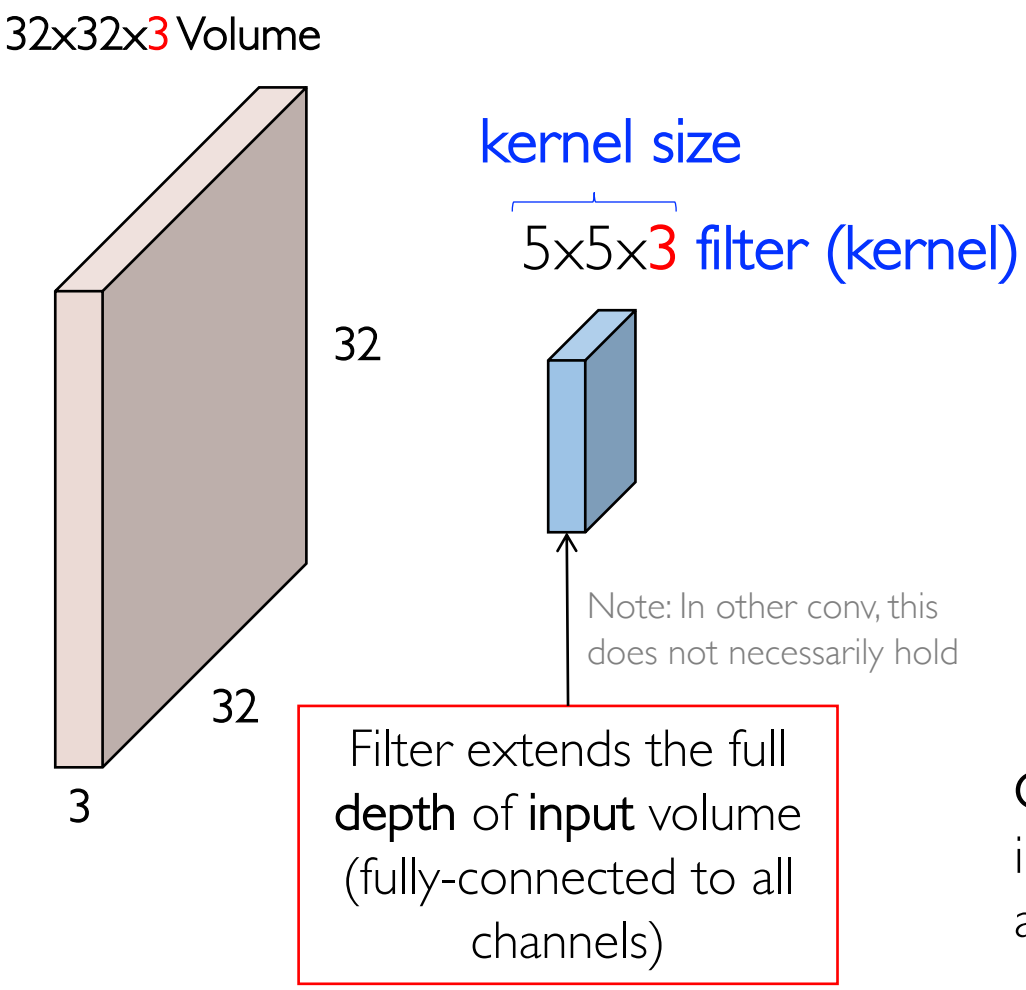
Volume

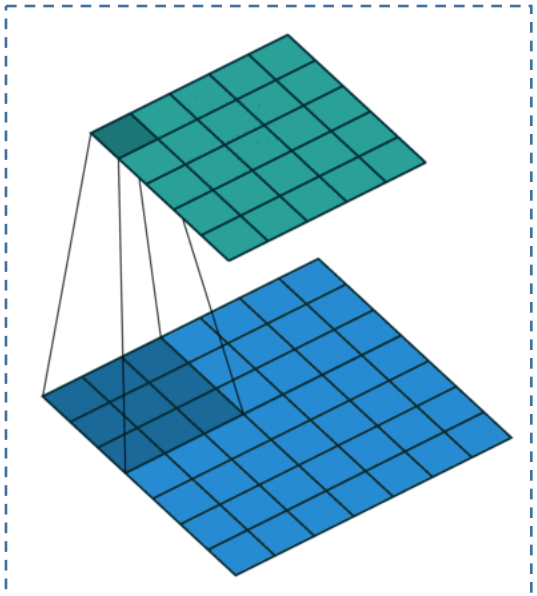
Tensor



Input channels: R/G/B

Convolution: Local Connection

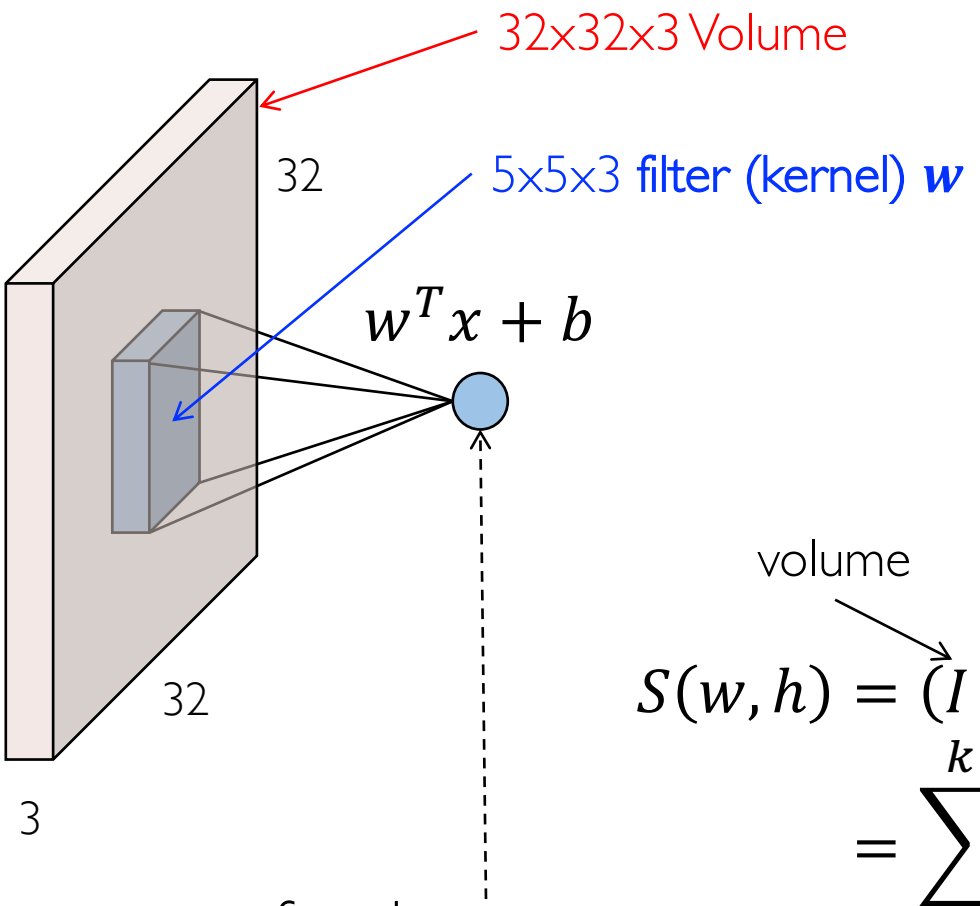


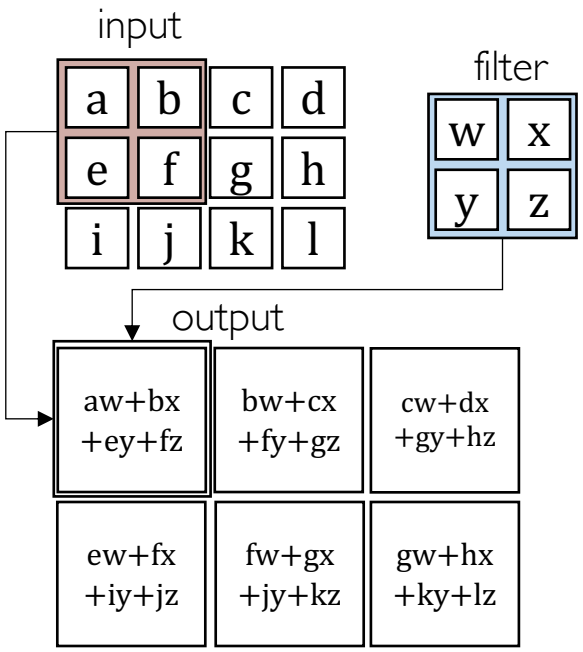


A simple picture of 3x3x1 convolution

Convolve the filter with the image i.e. **slide over** the image spatially, and computing the **dot products**

Convolution: Local Connection





volume filter

$$S(w,h) = (I \star K)(w,h)$$

$$= \sum_{i=1}^{k} \sum_{j=1}^{k} I(w+i, h+j) K(i,j)$$

- The output of each neuron:
  - Dot product between filter and a small 5x5x3 chunk of the volume

Convolution: Local Connection

$$S(w,h,d') = (I \star K)(w,h,d')$$

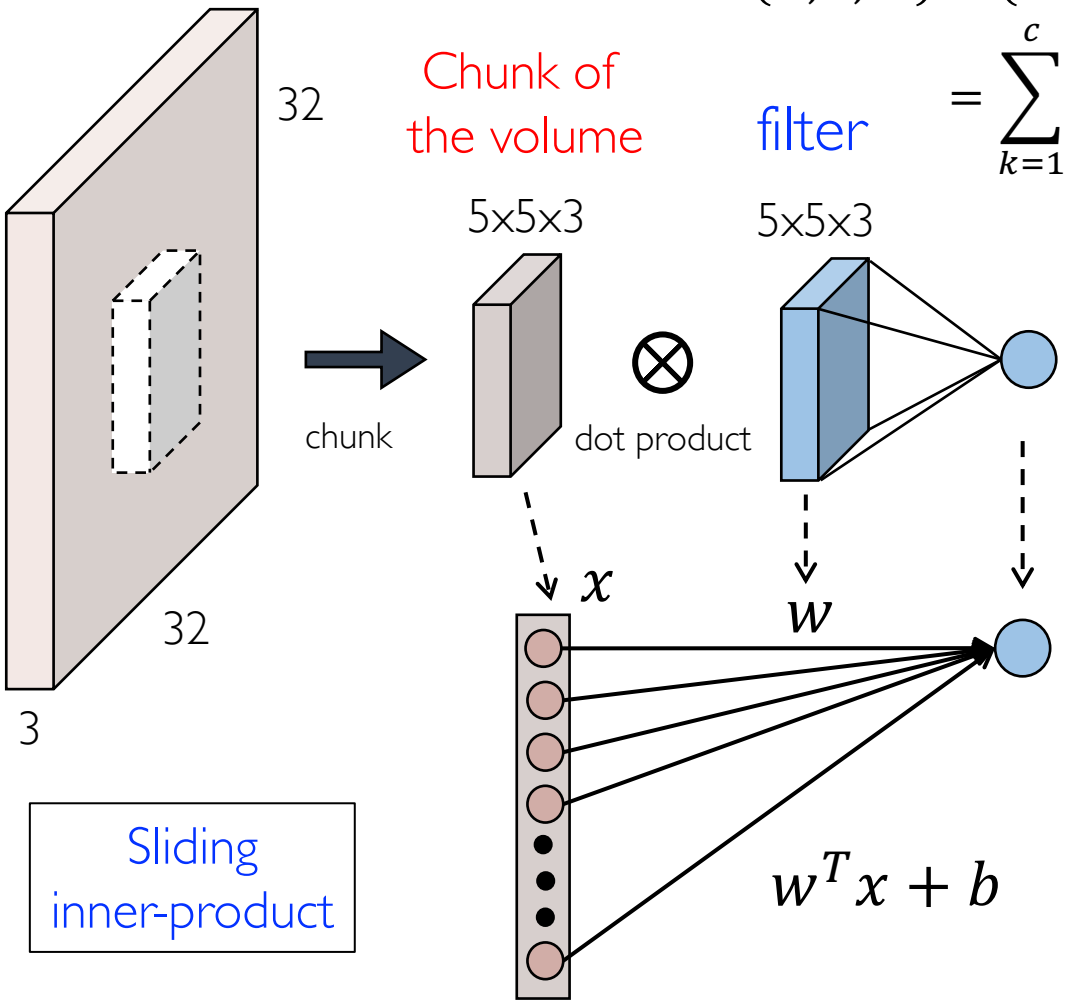
$$= \sum_{k=1}^{c} \sum_{i=1}^{k} \sum_{j=1}^{k} I(w+i, h+j, k) K(i,j,k)$$

Chunk of the volume

filter

Chunk of the volume

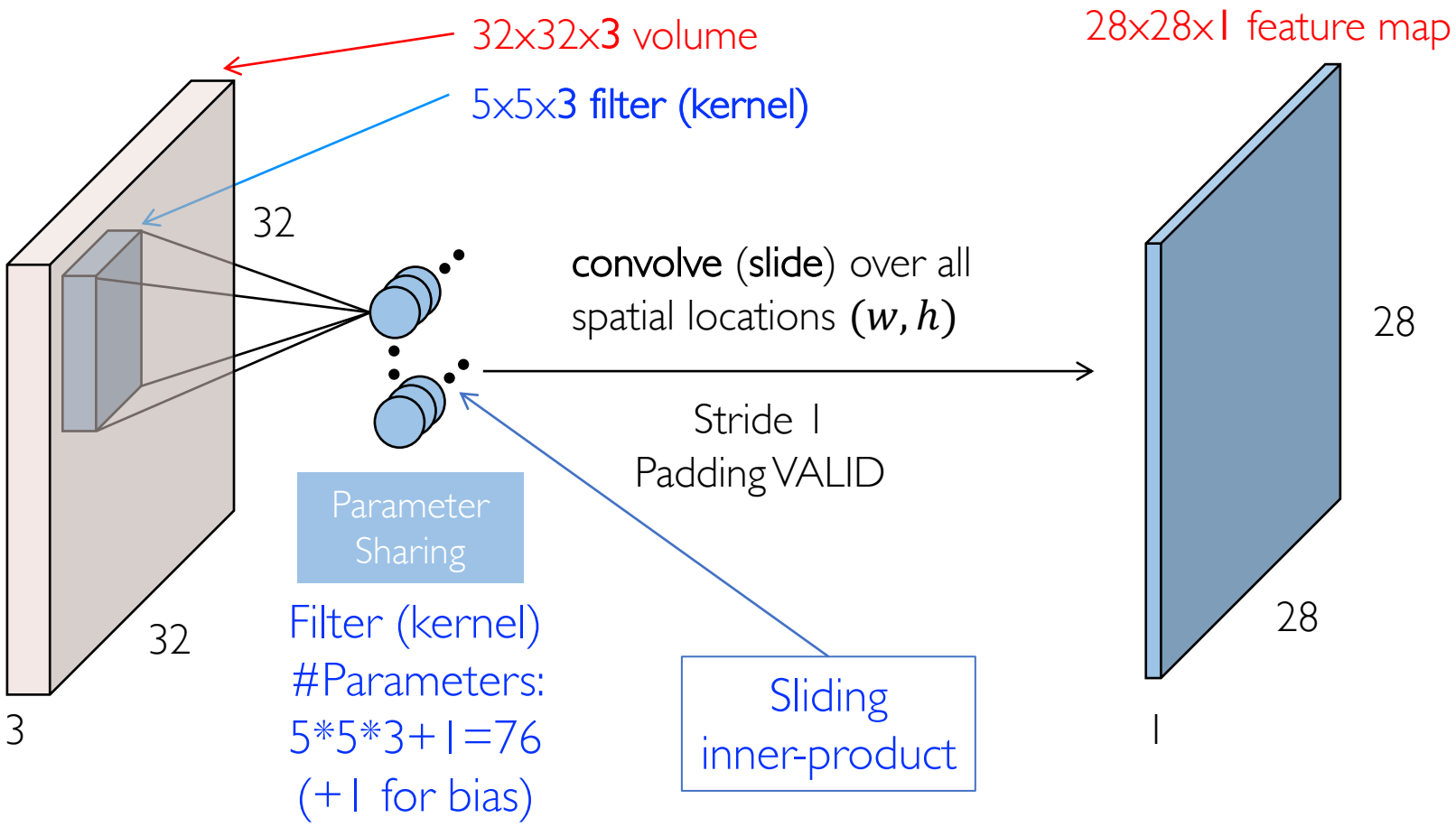
filter (kernel)



Sliding inner-product

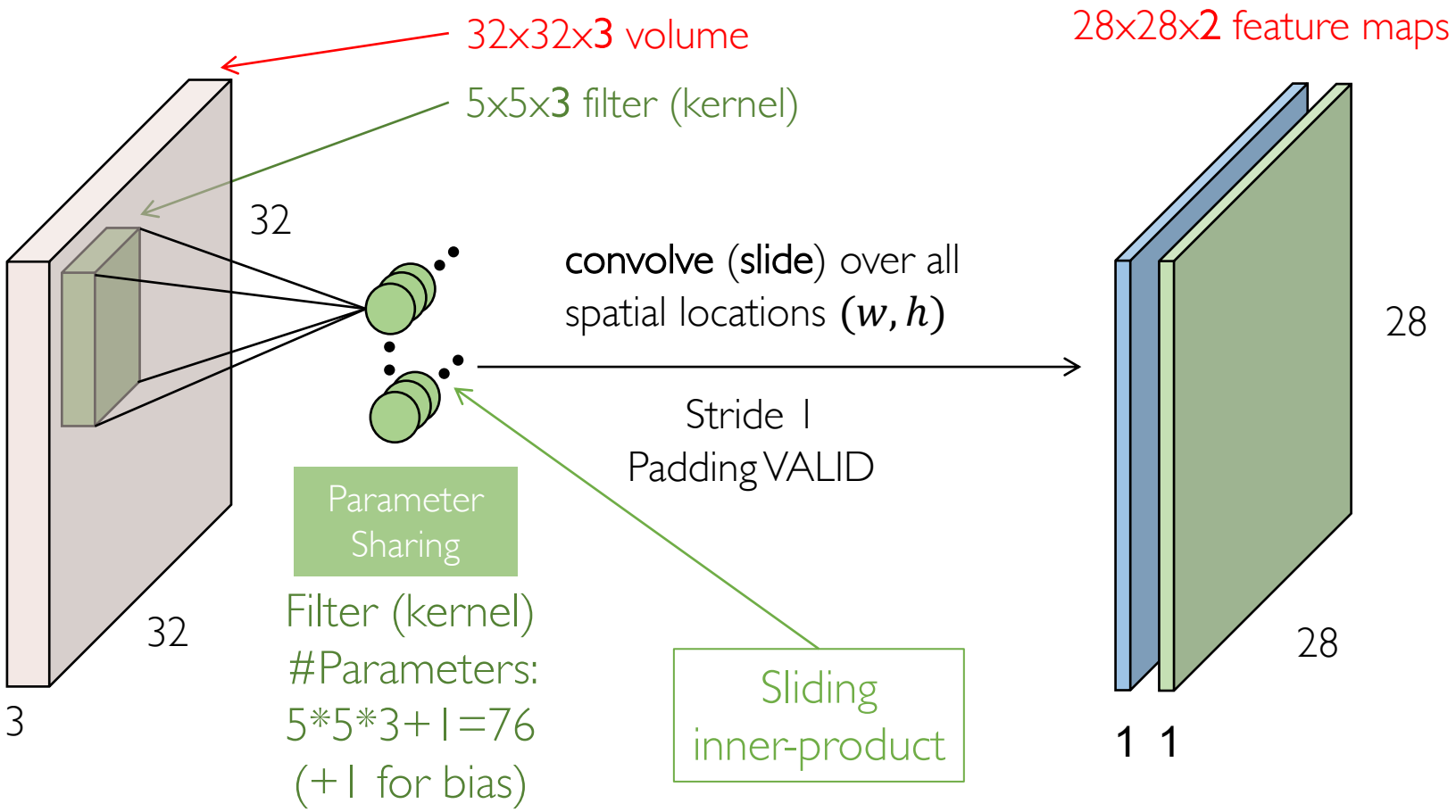
The output of each neuron: Dot product between filter and a small 5x5x3 chunk of the volume

Convolution: Parameter Sharing



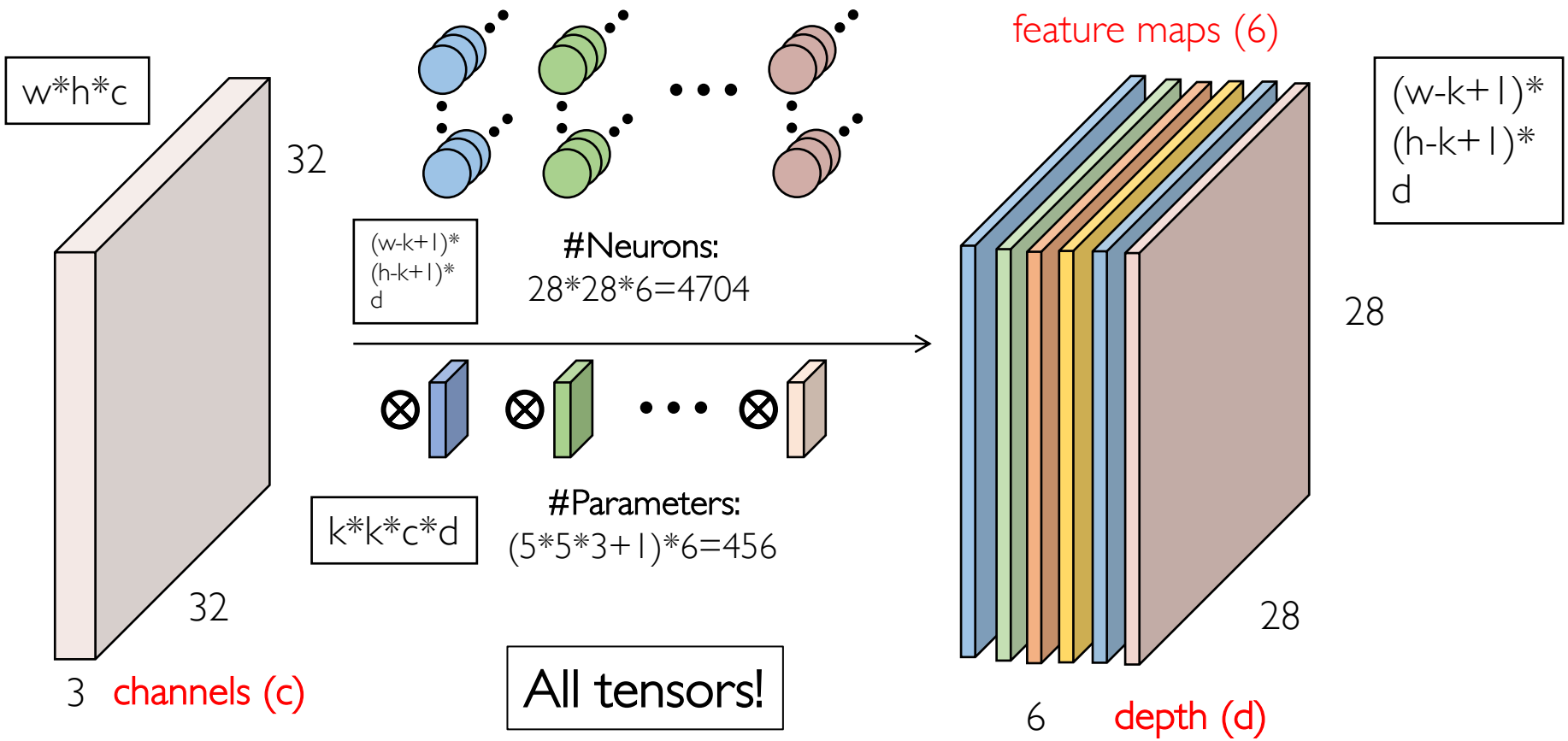
Not that no "sliding" in practice, we add neuron for each location (w, h)

Convolution: Parameter Sharing



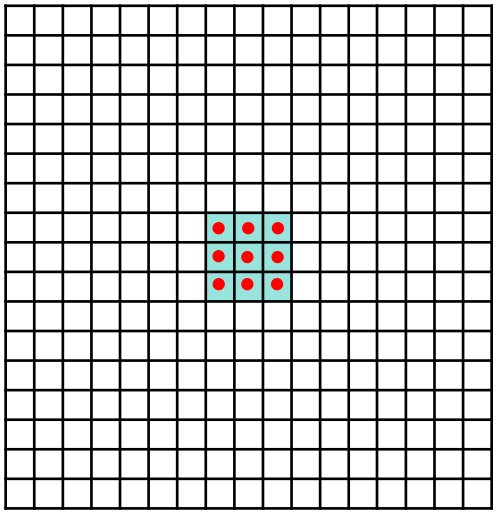
Not that no "sliding" in practice, we add neuron for each location (w, h)

Convolution: Feature Maps

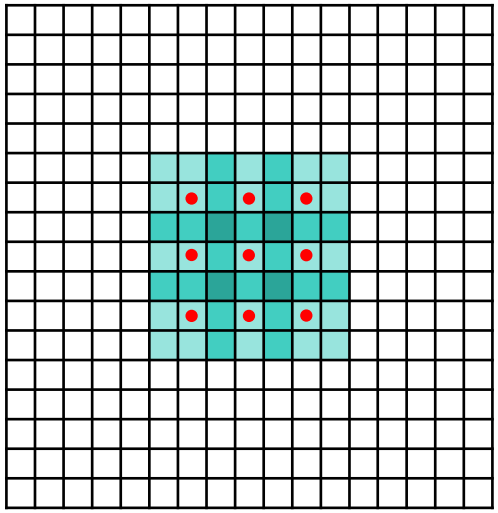


We stack all feature maps up to get a new image (representation) of size 28x28x6!
(However, we cannot see the new image since it has more than 3 R/G/B channels)

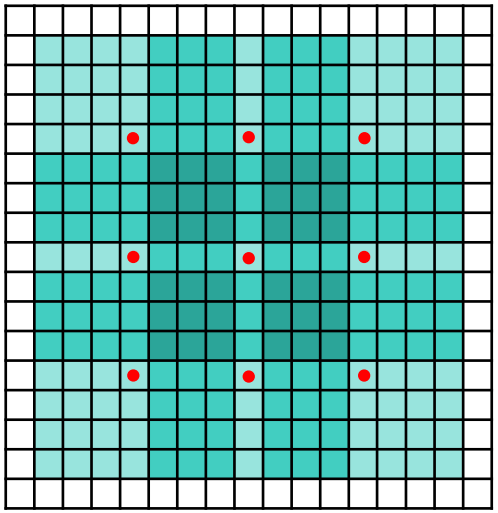
Dilated Convolution



3x3 **1-dilated** conv



3x3 **2-dilated** conv

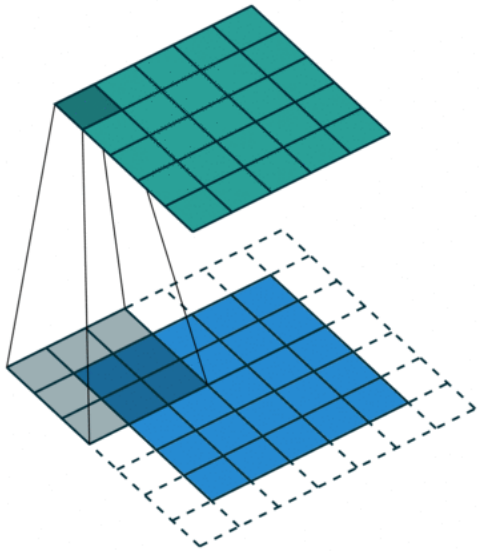


3x3 **4-dilated** conv

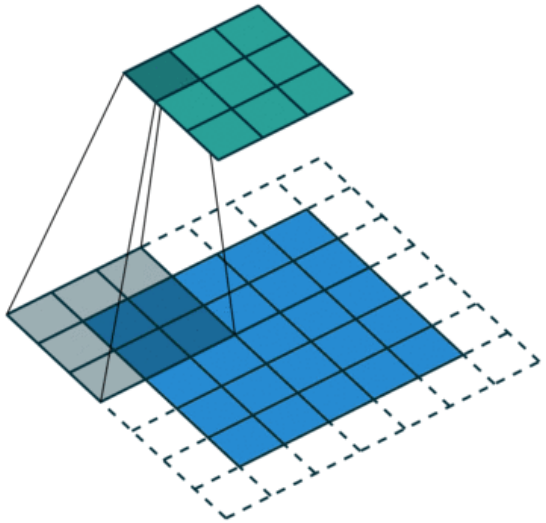
- Support exponential expansion of the **receptive field** without loss of resolution or coverage.

Fisher Yu, Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. ICLR, 2016.

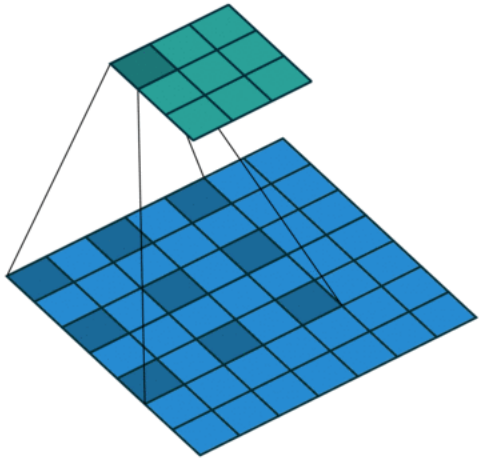
Convolution Operators



Convolution
Kernel Size: 3x3
Stride: 1
Padding: 1
Dilation: 1



Stride Convolution
Kernel Size: 3x3
Stride: 2
Padding: 1
Dilation: 1

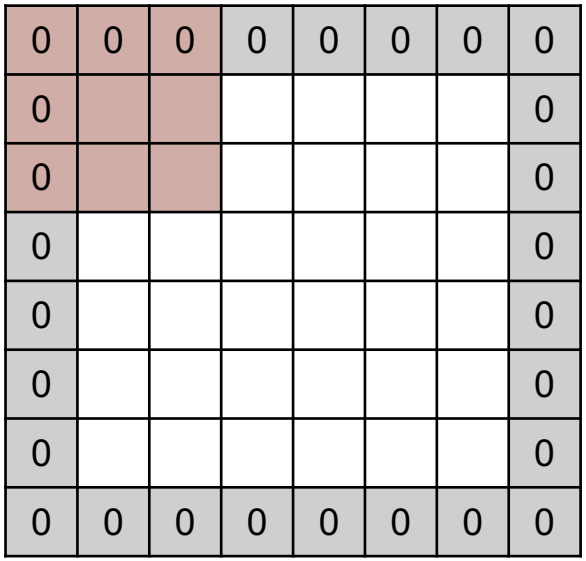


Dilated Convolution
Kernel Size: 3x3
Stride: 1
Padding: 0
Dilation: 2

http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

Padding

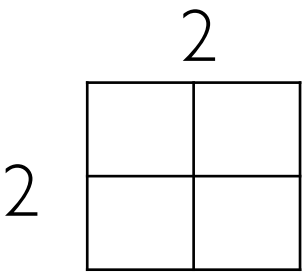
Input 7x7



(Pad P)

Add P circles of zeros outside of the original map

Stride 3
3x3 Filter



$$\text{height}_{\text{new}} = \lceil (\text{height-filter+2*pad}) / \text{stride} \rceil + 1$$

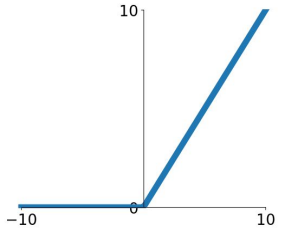
$$\text{width}_{\text{new}} = \lceil (\text{width-filter+2*pad}) / \text{stride} \rceil + 1$$

- Multiple layers of convolutions reduce the spatial sizes $W \times H$ and the information available at the boundary. **Padding** addresses this problem.

Activation Functions

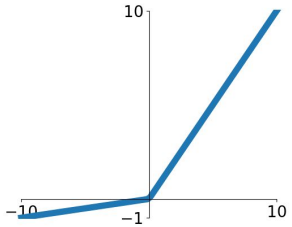
ReLU

$\max(0, x)$



Leaky ReLU

$\max(ax, x)$

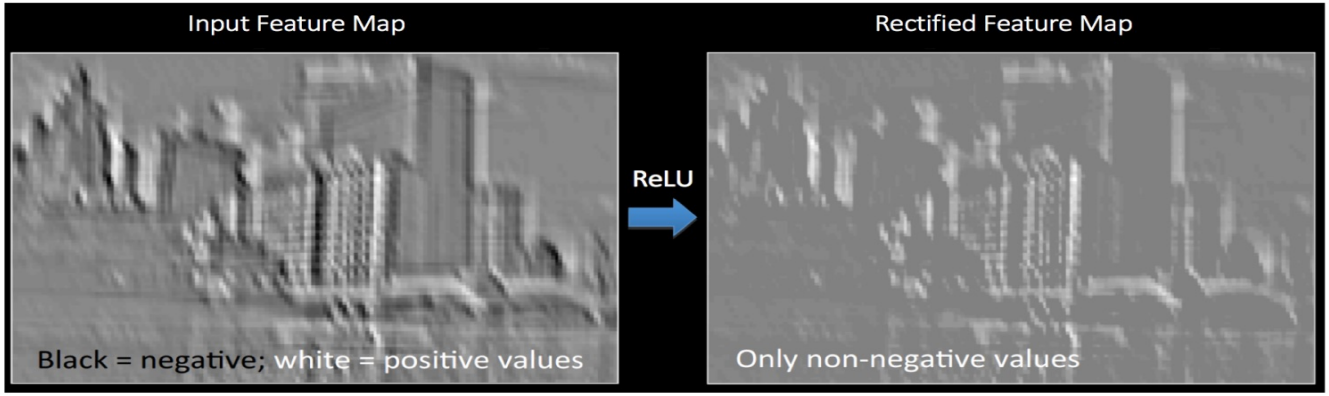


Advantages:

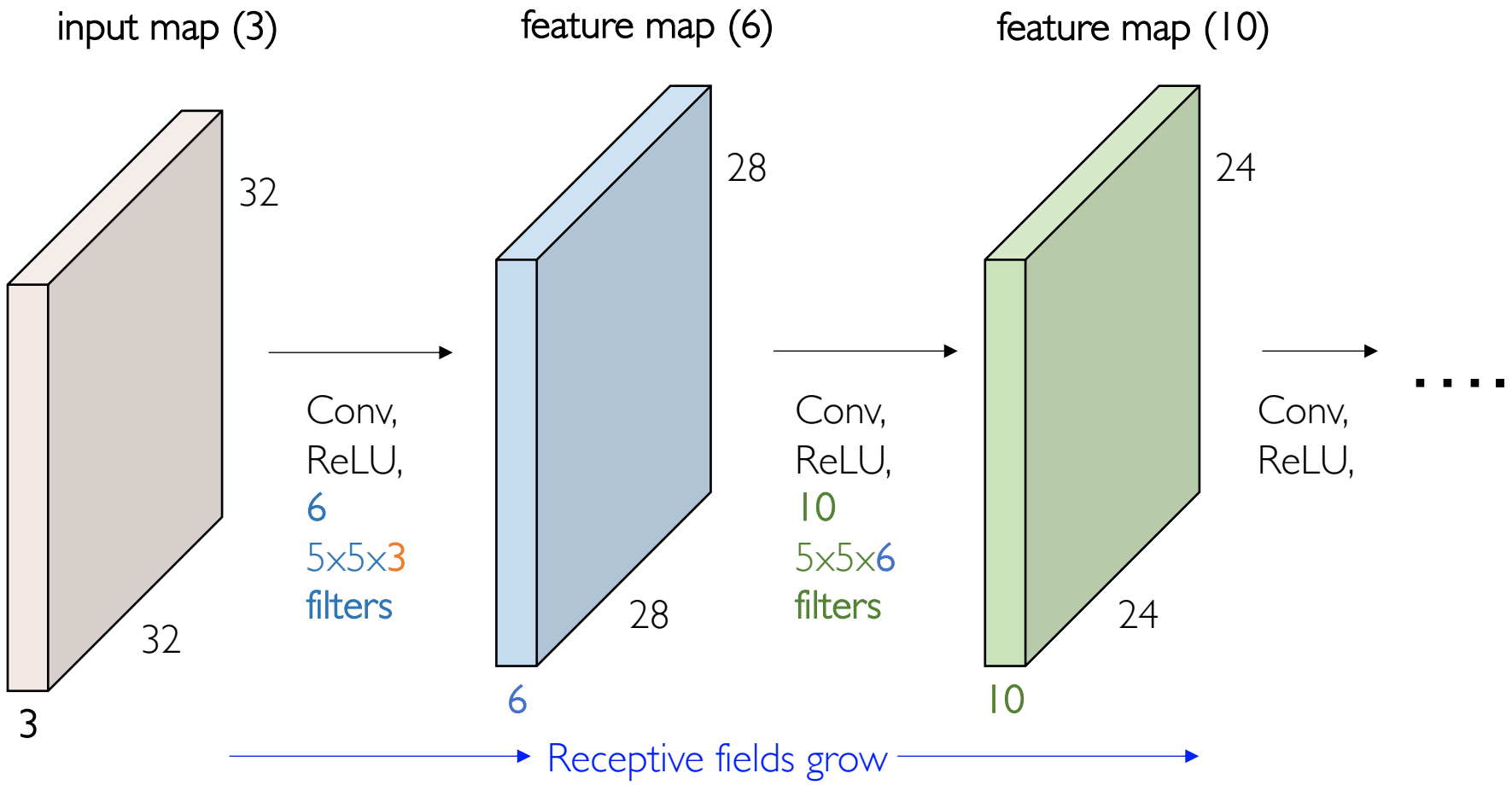
- Does not saturate (in positive region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

Disadvantages:

- **Not zero-centered output**
- **An annoyance → Leaky ReLU**



Convolutional Layer



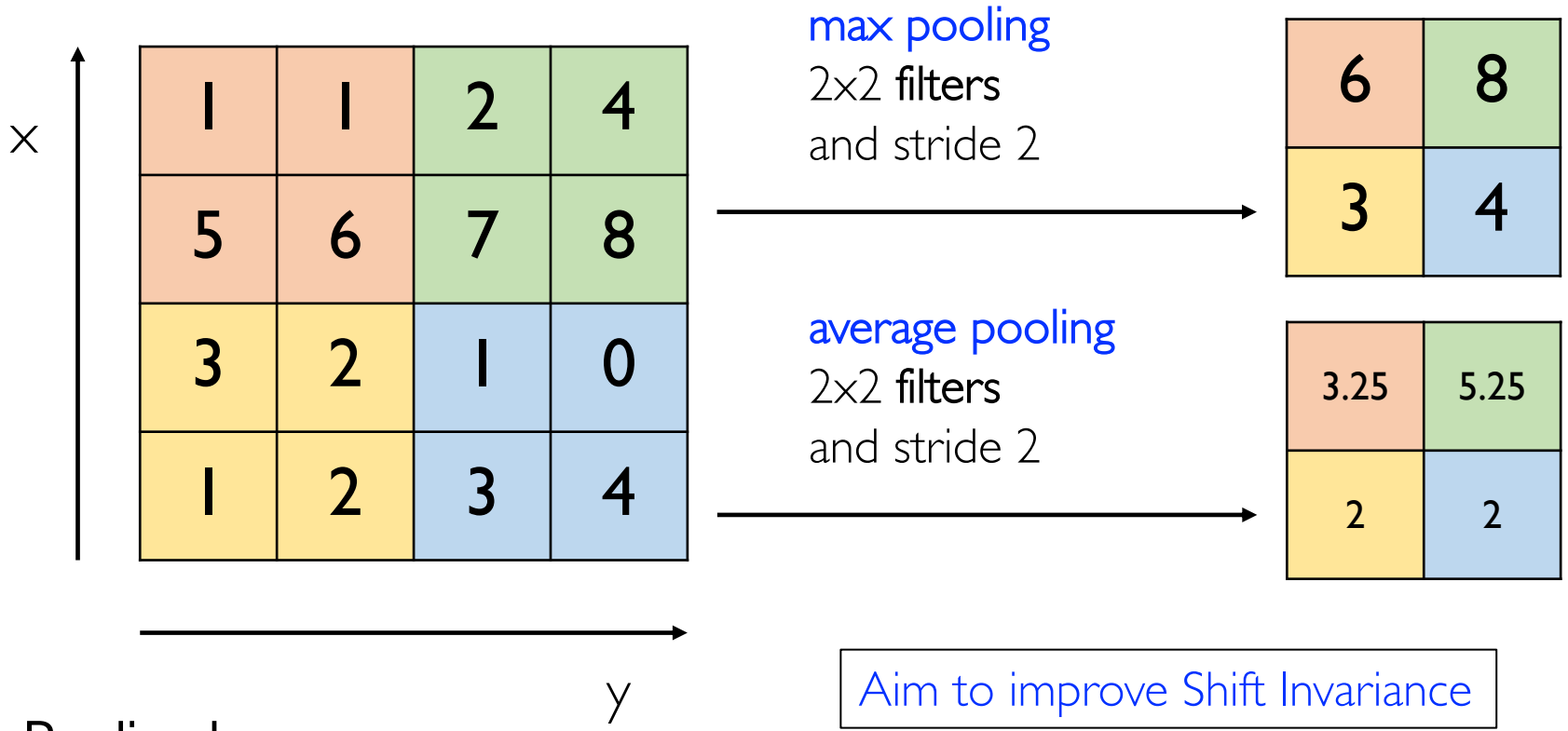
CNN is a stack of Convolutional Layers, interspersed with activation functions

Note: Shrinking too fast in the spatial dimensions is not good, doesn't work well.

Convolutional Layer: Hyperparameters

- Accepts an input volume of size $W_1 * H_1 * D_1$
- Requires four hyper-parameters
  - **Number of filters:** D_2 (depth of next layer)
  - **Receptive field (kernel size):** K
  - Stride: S
  - The amount of zero Padding: P
- Produce an output volume of size $W_2 * H_2 * D_2$
  - $W_2 = (W_1 - K + 2P) / S + 1$
  - $H_2 = (H_1 - K + 2P) / S + 1$
- With parameter sharing, it requires $(K * K * D_1 + 1) * D_2$ parameters

Pooling

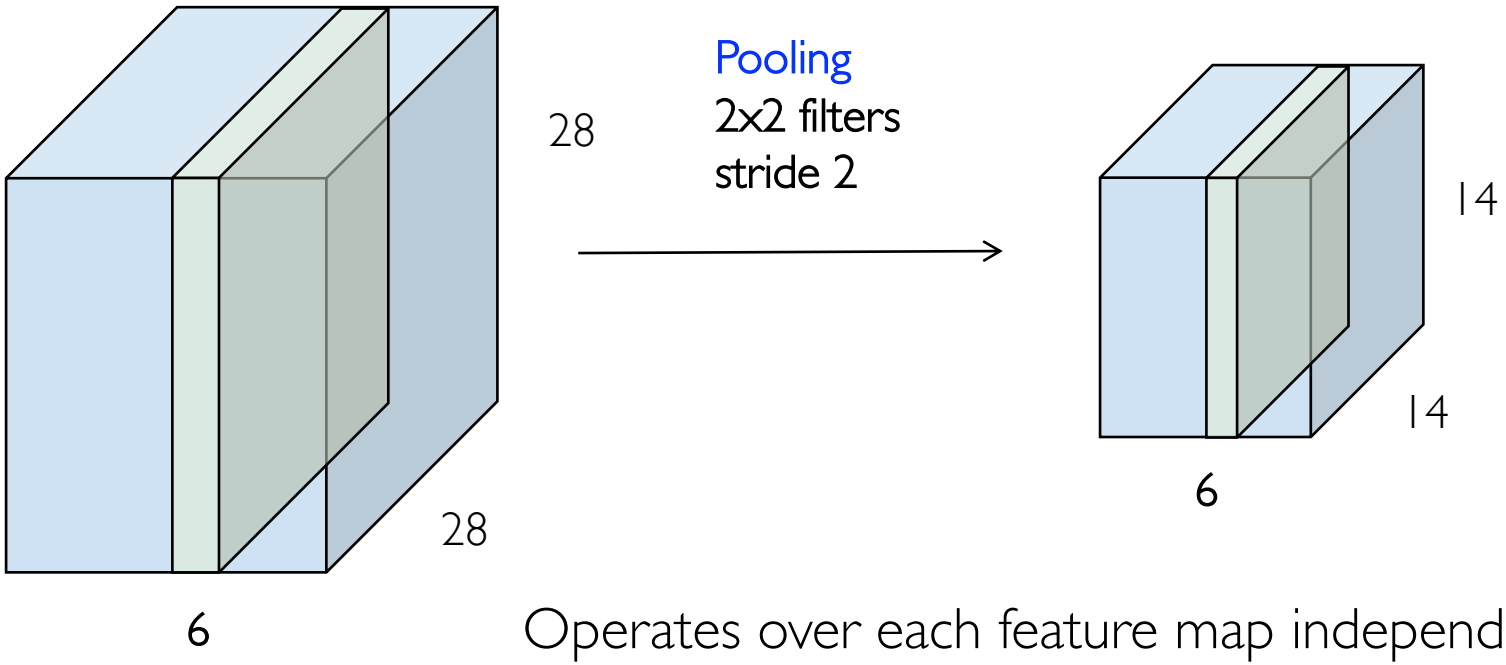


Aim to improve Shift Invariance

- Pooling layers
  - Alleviate excessive sensitivity of convolutional layers to locations
  - Reduce the resolution of images through the processing pipeline

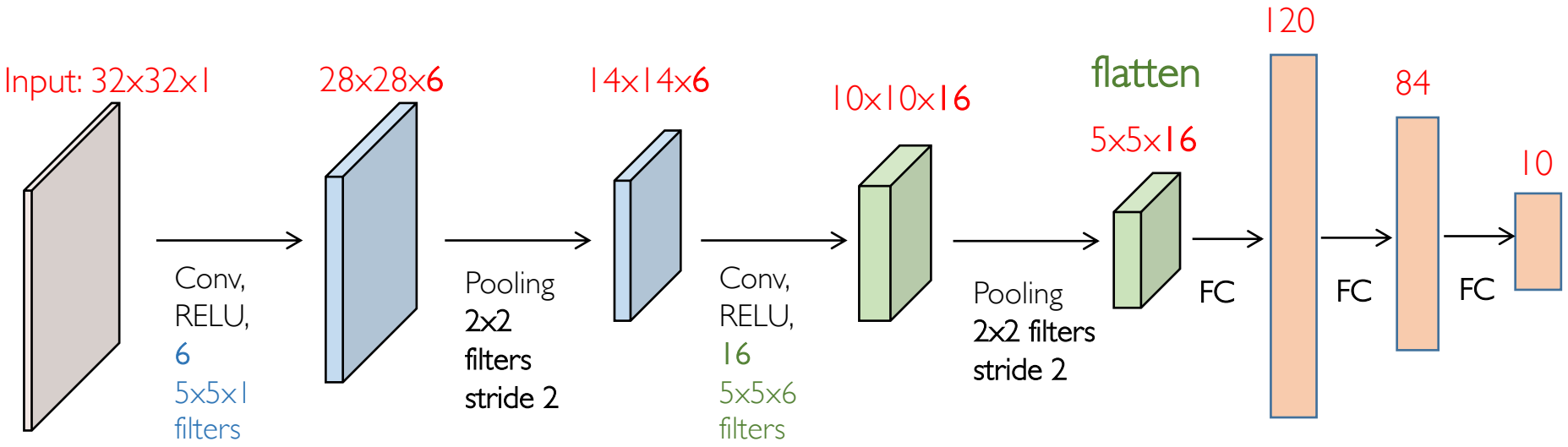
Pooling

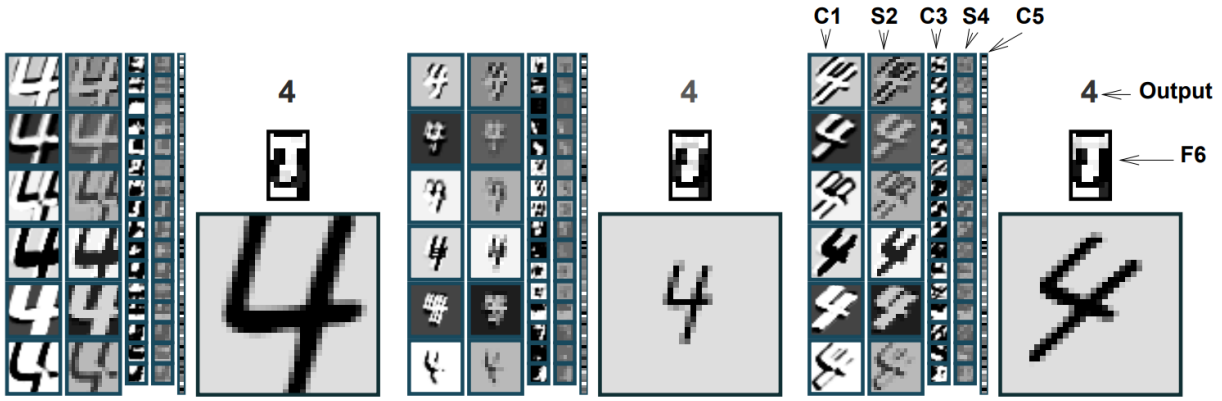
Aim to improve Shift Invariance



Operates over each feature map independently:
Pooling reduces dimensions of width and height.
Depth keeps unchanged before and after pooling

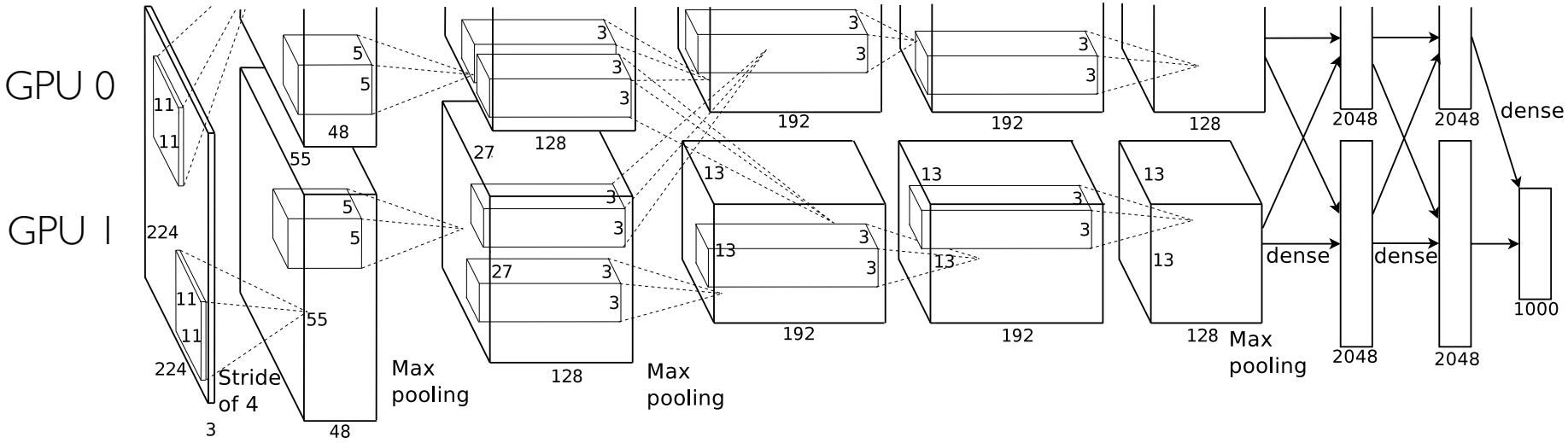
LeNet

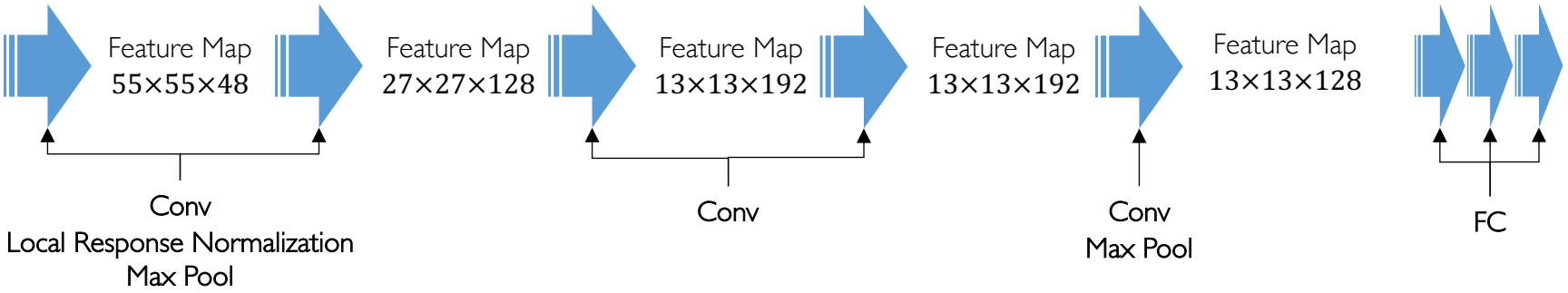




Y. LeCun, L. Bottou, Y. Bengio, Gradient-based learning applied to document recognition, Proc. IEEE, 1998. (Cited 48907)

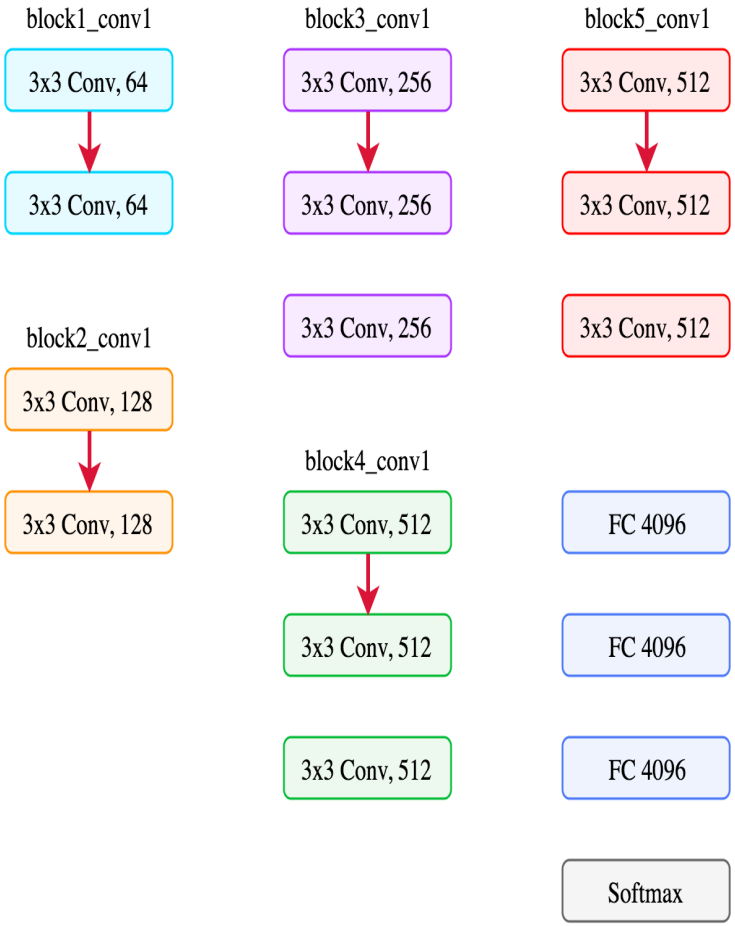
AlexNet

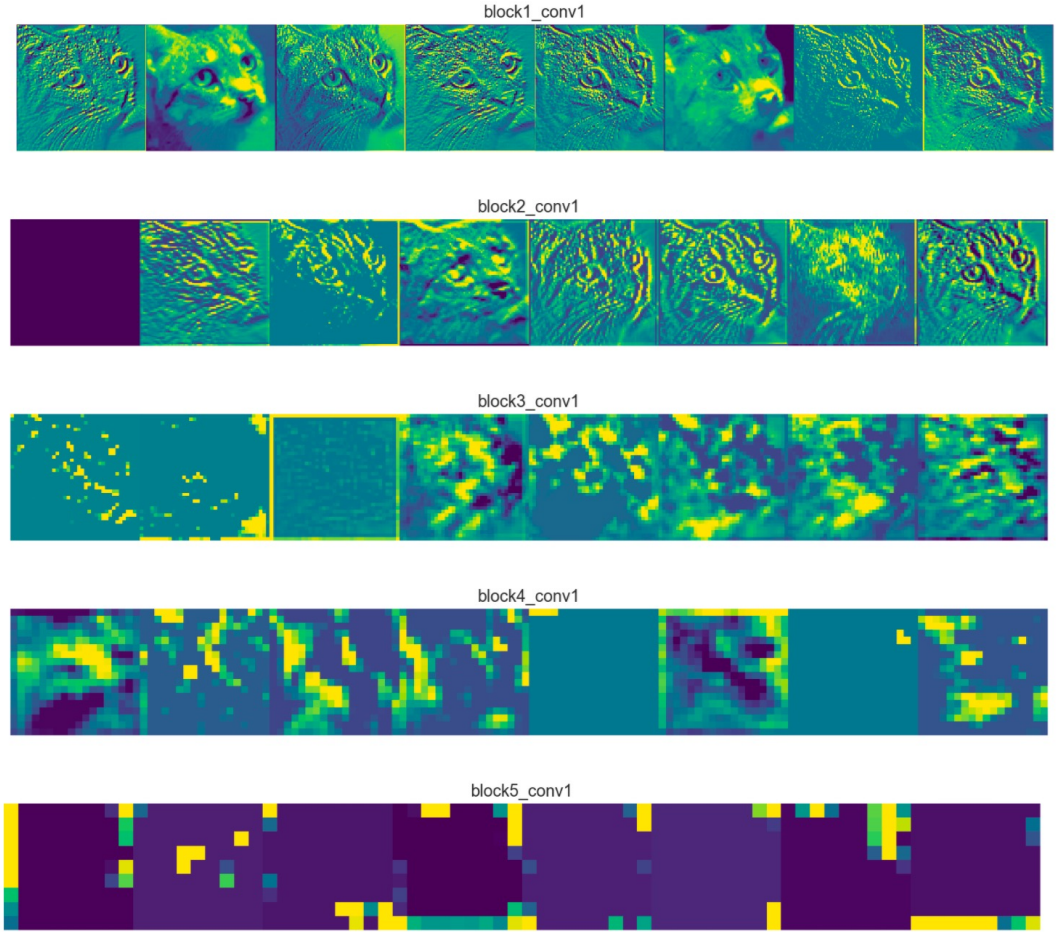




Krizhevsky, Alex, Ilya Sutskever, Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012. (Cited 116256)

Visualization





Display only eight of the 64/128/256/512 feature maps

https://github.com/utkuozbulak/pytorch-cnn-visualizations

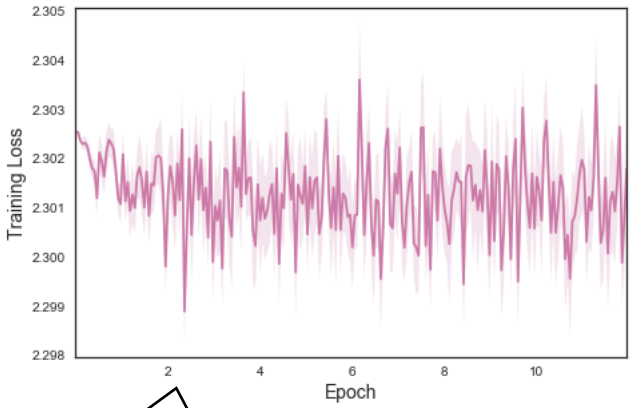
Outline

- Convolutional Neural Network
- **Practical Training Strategies**
  - **Weight Initialization**
  - Batch Normalization
  - Data Augmentation
- Architecture Revolution

Weight Initialization

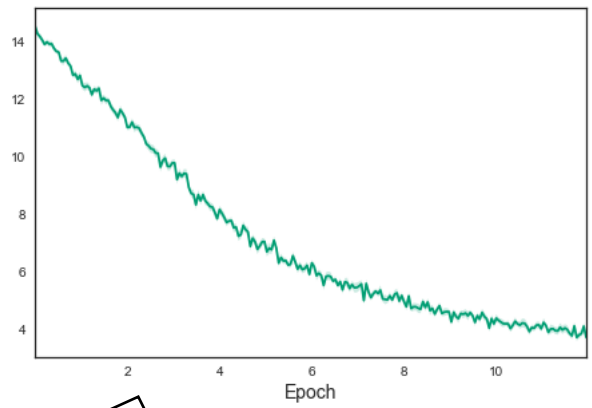
2-conv CNN on MNIST dataset

Initial Weights Set to Zero



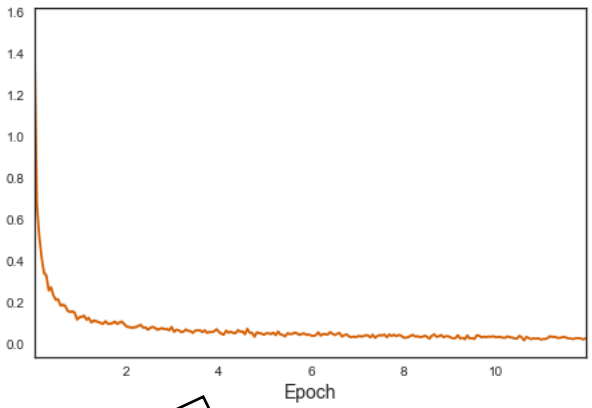
- Initialize W=0
- The network cannot learn at all

Initial Weights Drawn from $\mathbf{N}(0, \sigma = 0.4)$



- Rate of convergence is very low
- The network barely achieves a validation accuracy of about 88%

Initial Weights Drawn from $\mathbf{N}(0, \sigma \sim \sqrt{2/n_i})$



- Two orders of magnitude smaller loss
- A validation accuracy greater than 99%

https://github.com/Intoli/intoli-article-materials/tree/master/articles/neural-network-initialization

Xavier Initialization

- Suppose that $\mathbb{E}[x_j] = 0$, $\mathbb{E}[x_j^2] = \gamma$. We aim to keep expectation and variance invariant across layers.
- The hidden units h_i for some layer are given by:

$$h_i = \sum_{j=1}^{n_{\text{in}}} W_{ij} x_j$$

- We assume W_{ij} drawn independently from the same distribution with zero mean and variance σ^2. In this case:

$$\mathbb{E}[h_i] = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[W_{ij} x_j] = 0$$

$$\mathbb{E}[h_i^2] = \sum_{j=1}^{n_{\text{in}}} \mathbb{E}[W_{ij}^2]\mathbb{E}[x_j^2] = n_{\text{in}}\sigma^2\gamma$$

- One way to keep variance fixed is to set $\sigma^2 = 1/n_{\text{in}}$

Weight Initialization

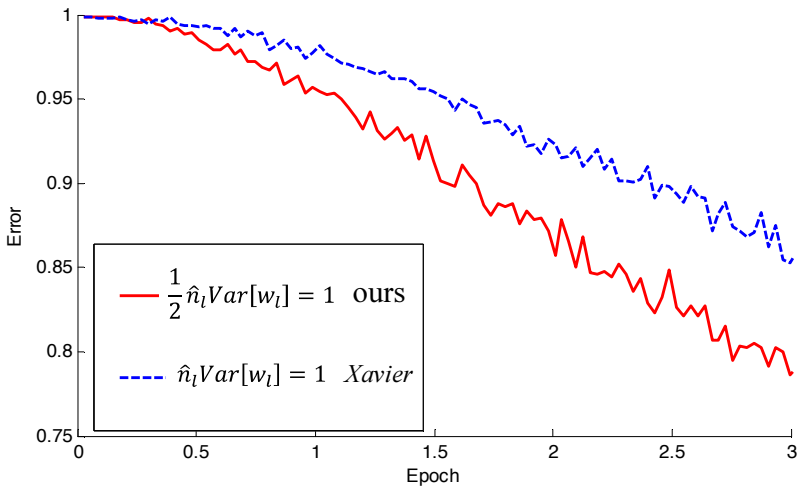
multiplicative effect through layers

Proper initialization avoids reducing or magnifying the magnitudes of signals exponentially

Xavier Initialization

(linear activations)

$$\mathrm{Var}(w) = 1 / n_{\text{in}}$$

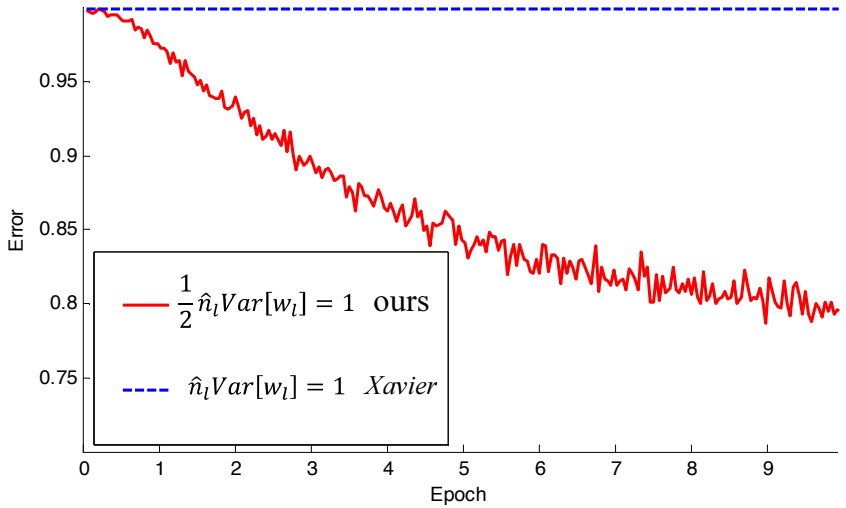


22-layer ReLU network

He Initialization

(ReLU activations)

$$\mathrm{Var}(w) = 2 / n_{\text{in}}$$



30-layer ReLU network

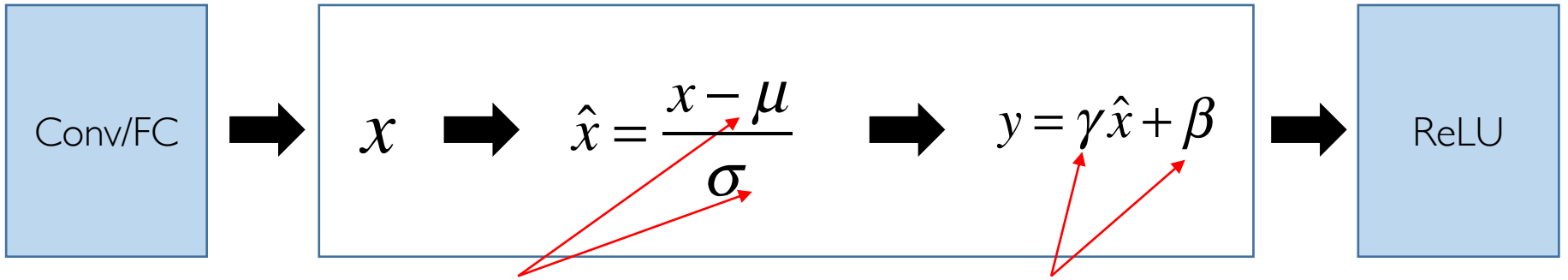
Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. JMLR, 2010, 9:249-256.

He K, Zhang X, Ren S, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification[J]. 2015:1026-1034.

Outline

- Convolutional Neural Network
- **Practical Training Strategies**
  - Weight Initialization
  - **Batch Normalization**
  - Data Augmentation
- Architecture Revolution

Batch Normalization



After FC or Conv layers

Mean and std of input in each running mini-batch

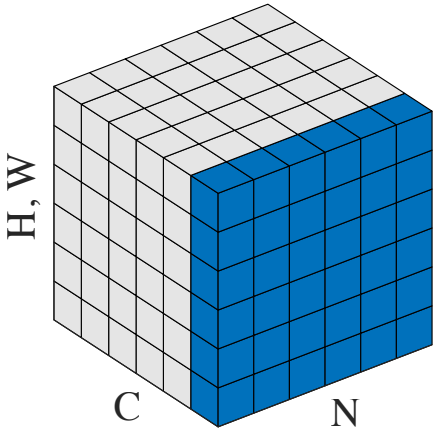
Parameters to be learned

If set $\gamma = \sqrt{\mathrm{Var}(x)}, \beta = \mathrm{E}(x)$

→ identity mapping

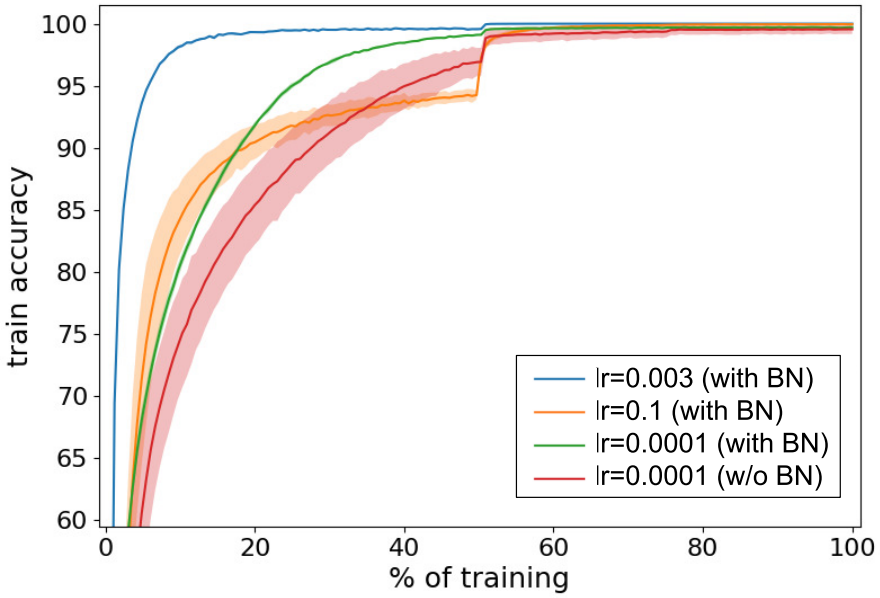
Before the nonlinearity

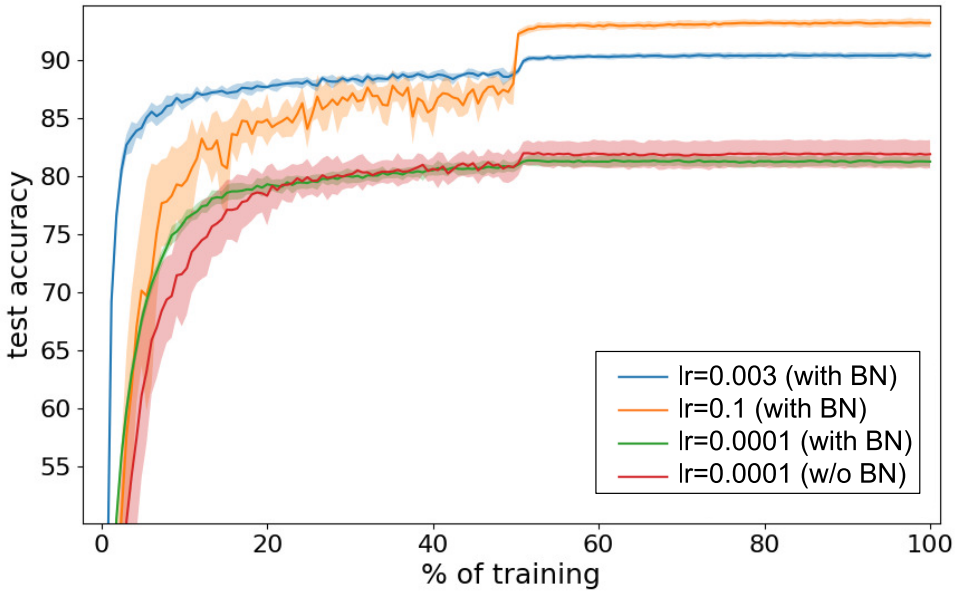
- Normalize over mini-batch examples
- At training: for a mini-batch of size n and feature maps of size $w \times h$, all $n \times w \times h$ elements are normalized, each feature map has a pair of γ & β
- At inference: use **EMA** to compute moving $\tilde{\mu}$ & $\tilde{\sigma}$



Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. **ICML**, 2015.

Batch Normalization

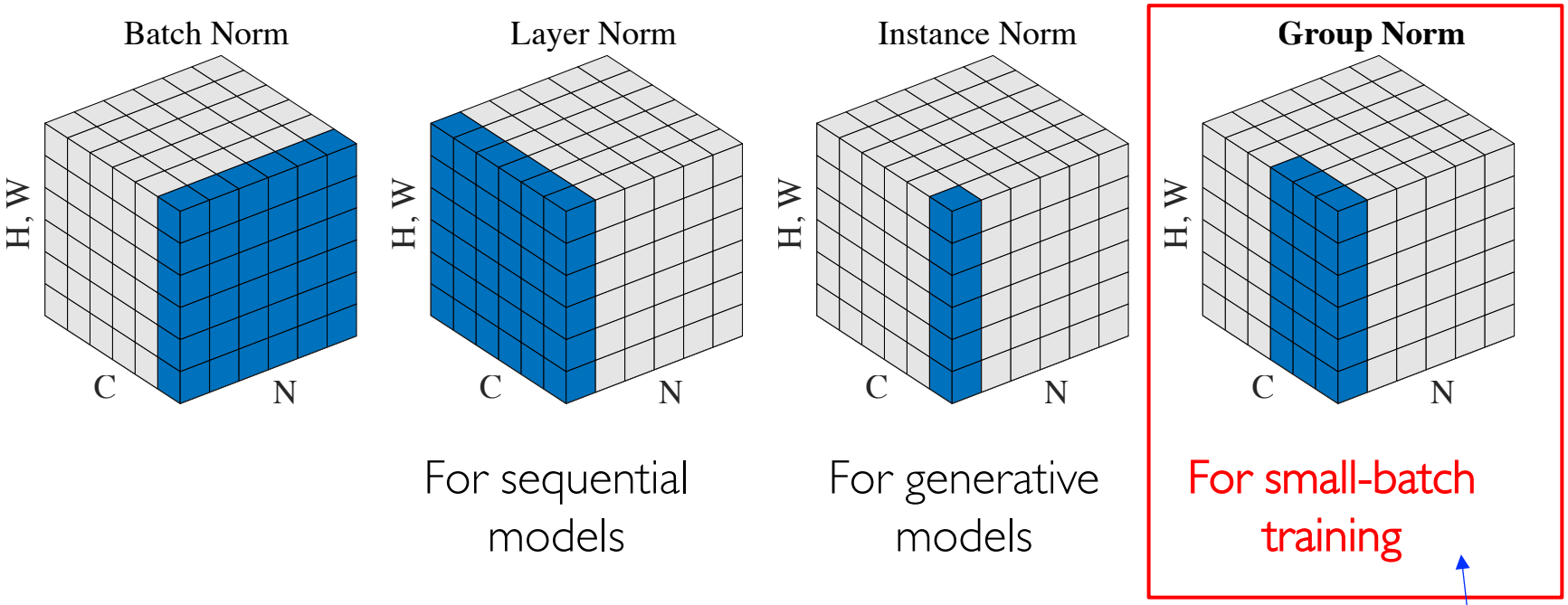




- Comparable learning rates result in comparable testing accuracies with and w/o BN.
- Larger learning rates yield higher test accuracy for BN networks, and diverge for networks w/o BN.

Bjorck, Nils, et al. Understanding batch normalization. NIPS 2018.

Group Normalization



For sequential models

For generative models

For small-batch training

- A small batch leads to inaccurate estimation of the batch statistics
- This limits BN for training larger models for detection and video.
- The computation of GN is independent of batch sizes.

Divides channels into G groups

He et al. Group Normalization. ECCV 2018.

Outline

- Convolutional Neural Network
- **Practical Training Strategies**
  - Weight Initialization
  - Batch Normalization
  - **Data Augmentation**
- Architecture Revolution

Data Augmentation

- Get more data for training
- Give the model high adaptability

Color Jittering

PCA Jittering

Random Scale

Random Crop

Scale jittering

Horizontal/Vertical Flip

Shift

Rotation/Reflection

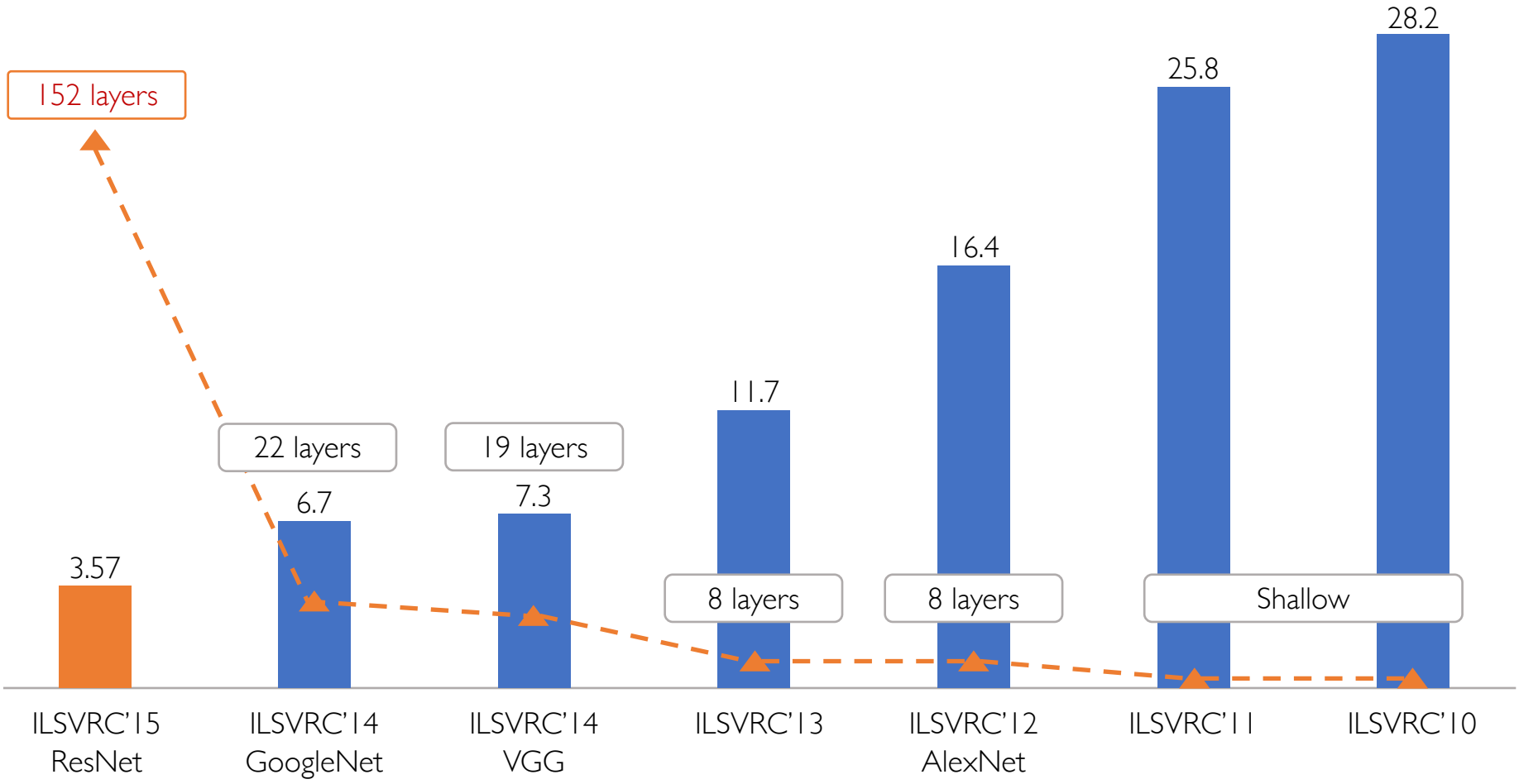
Label shuffle

Noise

Outline

- Convolutional Neural Network
- Practical Training Strategies
  - Weight Initialization
  - Batch Normalization
  - Data Augmentation
- **Architecture Revolution**

ImageNet ILSVRC

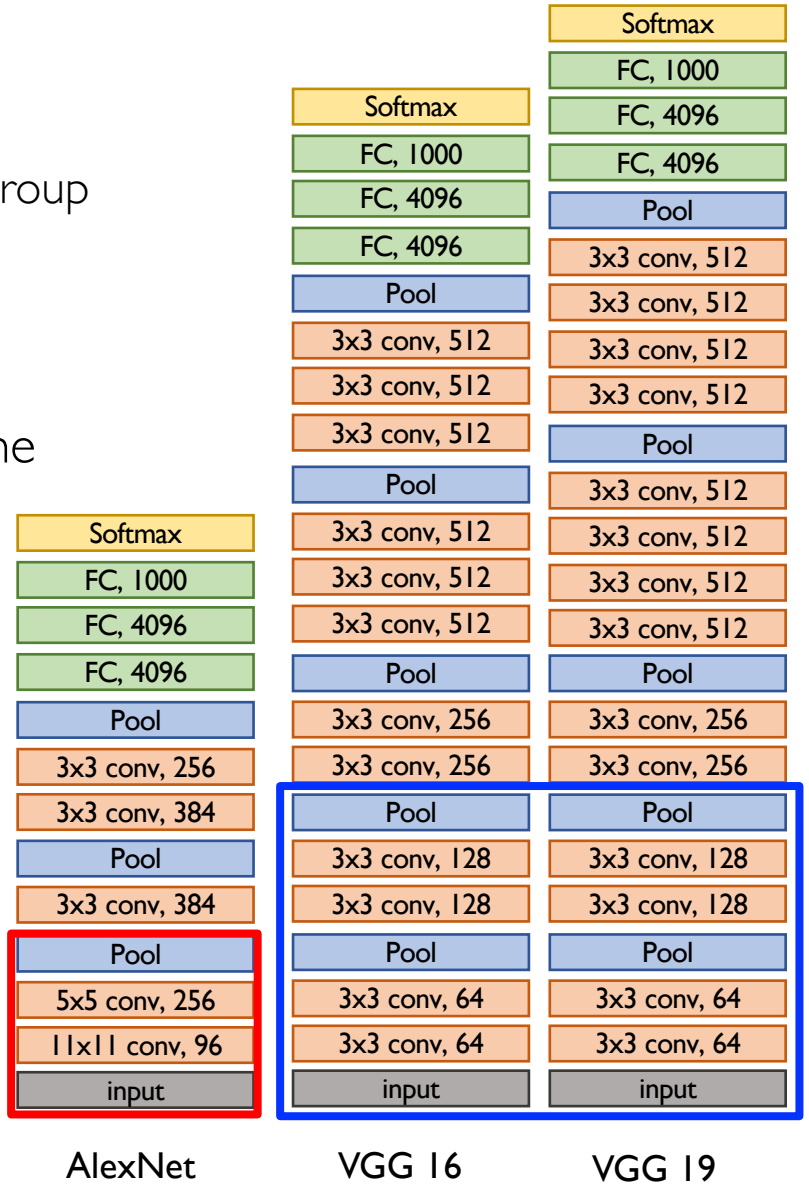


VGG

Visual Geometry Group

Smaller filters
Deeper networks

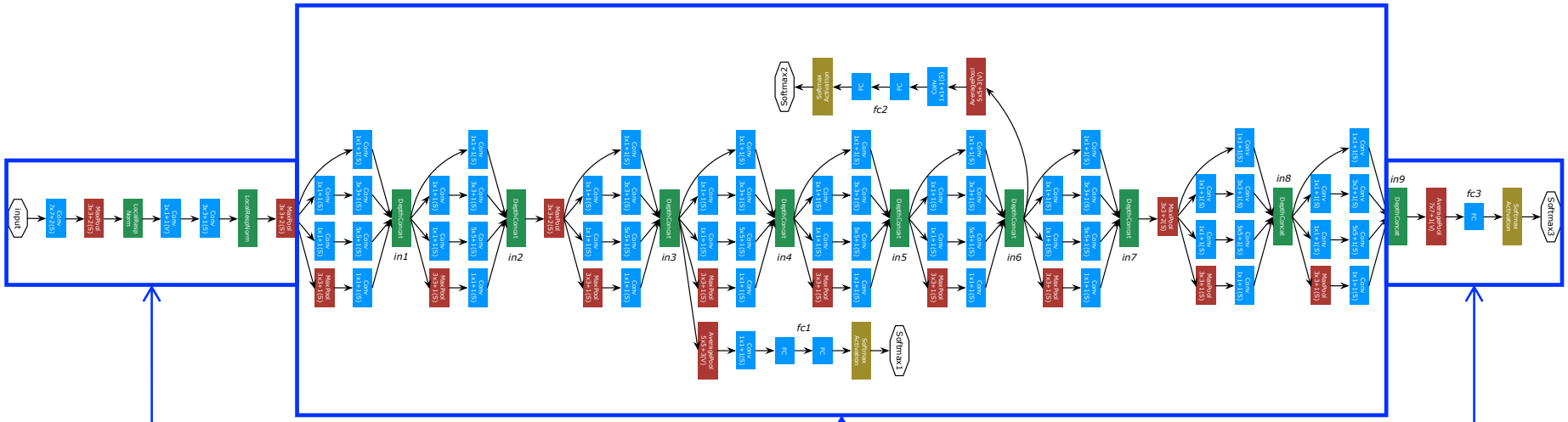
- Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer
  - But deeper, more nonlinearities
  - Pre-training required for deeper networks
- No Local Response Normalization (LRN)
  - Use ensemble for best results
  - Should focus on one-model accuracy
- FC features generalize well to other tasks
  - 2nd in classification, 1st in localization



K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

Inception-v1 (GoogLeNet)

Deeper networks, with computational efficiency

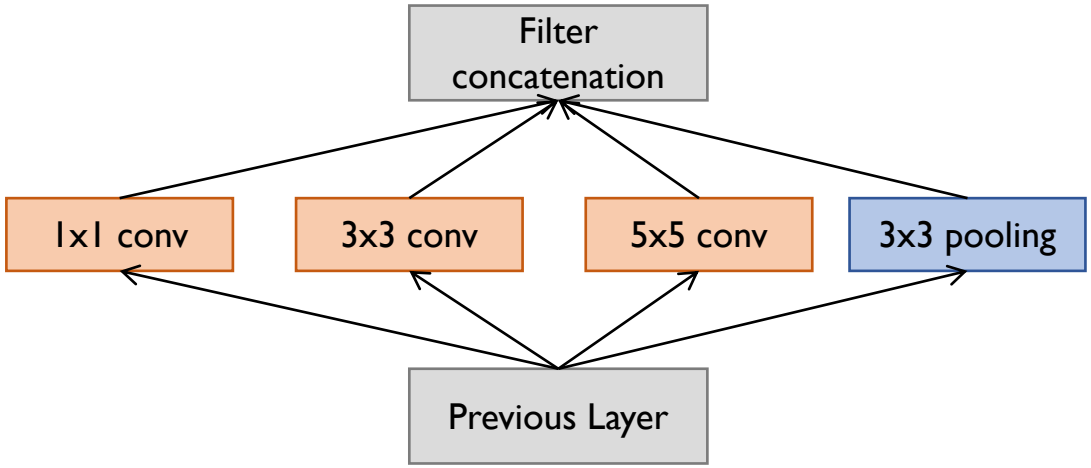


- 22 layers
- Efficient Inception modules
- The first one without FC layers
- Only 5 million parameters! 12x less than AlexNet
- ILSVRC'14 classification winner (6.7% top 5 error)

C Szegedy et al. Going Deeper with Convolutions. **CVPR** 2015.

Inception-v1 (GoogLeNet)

- **Inception module:** a local network topology (network in network) and then stack these modules on top of each other



Naive Inception module

Apply parallel filter operations upon previous layer:

- Multiple receptive fields for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

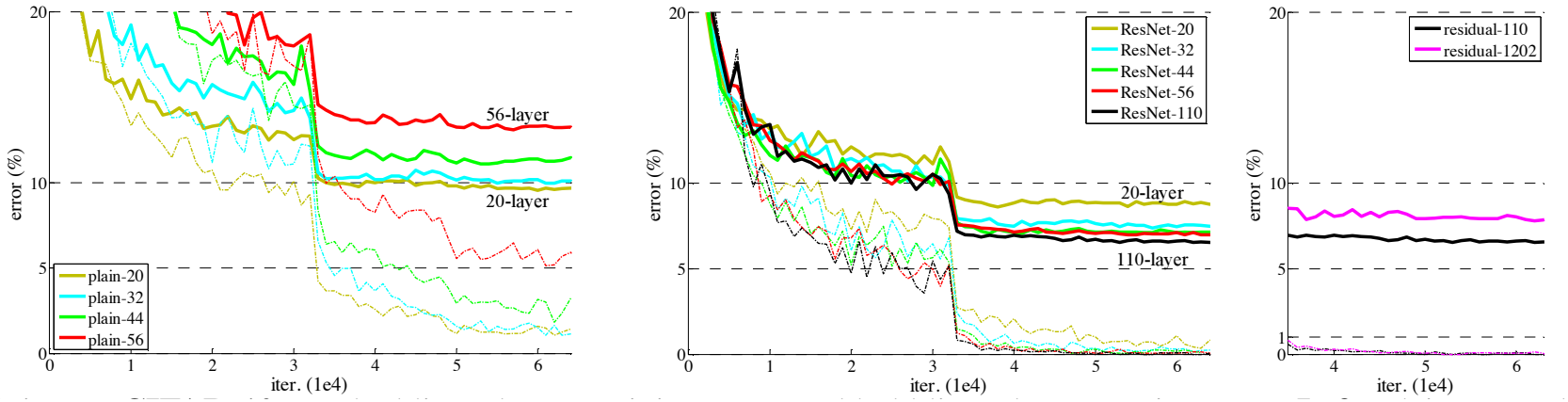
Concatenate all filter outputs together along channels

Apply padding when necessary

C Szegedy et al. Going Deeper with Convolutions. CVPR 2015.

Residual Network (ResNet)

- What happens when we continue stacking deeper layers on a plain convolutional neural network?

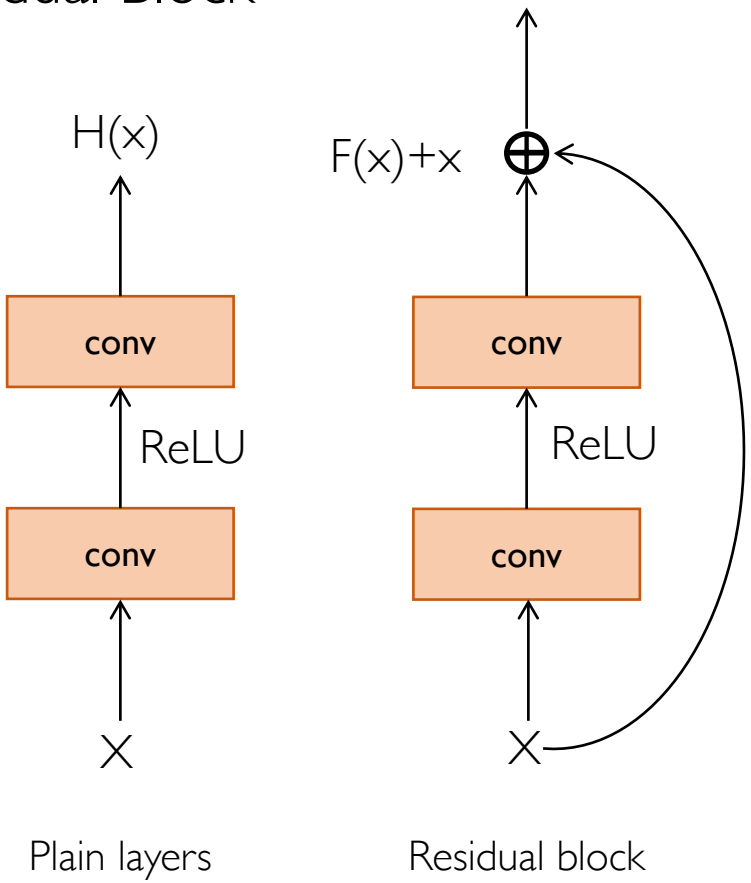


- 56-layer model performs worse on both training and test errors
  - The deeper model performs worse, but not caused by overfitting!
- Deeper models are harder to optimize. It is an optimization issue.

K. He et al. Deep Residual Learning for Image Recognition. CVPR 2016. (Best Paper)

ResNet

- Residual Block



How to analyze the problem?

- Copying the learned layers from the shallower model
- Setting additional layers to identity mapping.

Solution:

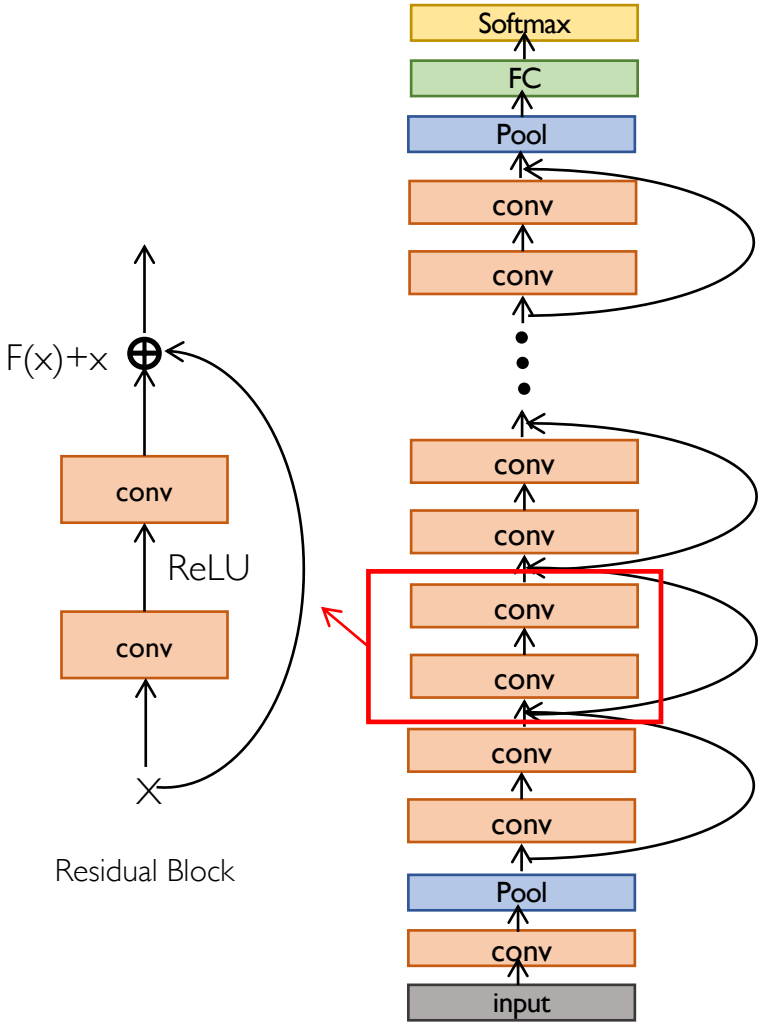
Learning $F(x) = H(x) - x$,

instead of learning $H(x)$ directly

K. He et al. Deep Residual Learning for Image Recognition. CVPR 2016. (Best Paper)

ResNet

- ResNet architecture:
  - Stack residual blocks
  - Every residual block has two 3x3 conv layers
  - Periodically, double #filters and downsample spatially using stride 2 (/2 in each dimension)
  - Additional conv layer at the beginning
  - Global average pooling at the end (FC layer only to output classes)

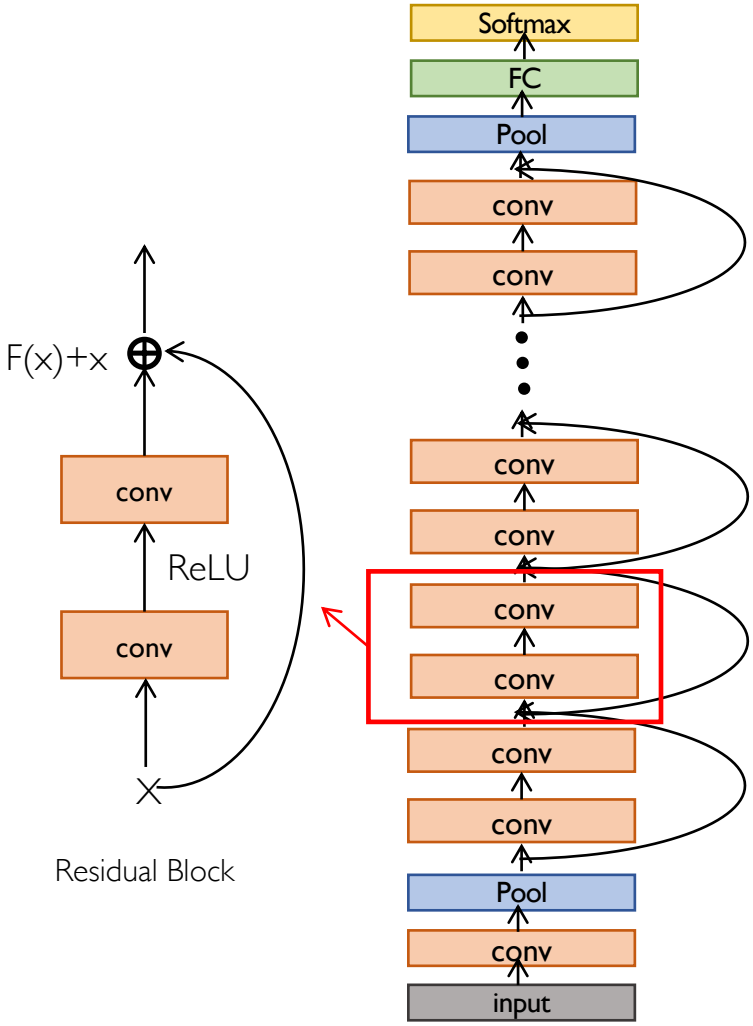


K. He et al. Deep Residual Learning for Image Recognition. CVPR 2016. (Best Paper)

ResNet

- Experimental Results
  - Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on CIFAR)
  - Deeper networks achieve lower training error as expected

- 1st place in all ILSVRC (3.6% top 5 error, surpassing human performance) and COCO 2015 competitions



K. He et al. Deep Residual Learning for Image Recognition. CVPR 2016. (Best Paper)