Lecture 3: Neural Network Basics & Architecture Design

Xiangyu Zhang

Face++ Researcher

zhangxiangyu@megvii.com

Face++ 旷视

Visual Recognition

A fundamental task in computer vision

- Classification
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Key point Detection
- VQA

…

Face++ 旷视

Why Recognition Difficult?

Pose

Occlusion

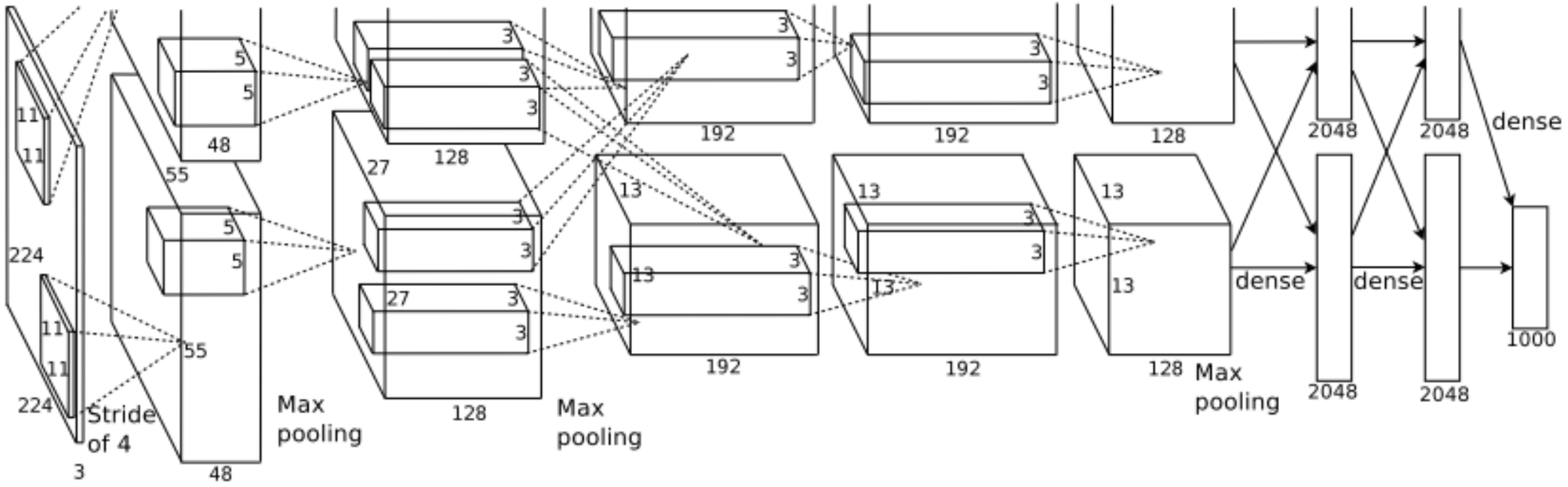
Multiple Objects

Inter-class Similarity

Face++ 旷视

Any Silver Bullet?

- Deep Neural Networks



Outline

- Neural Network Basics
- Architecture Design

PART 1: Neural Network Basics

- Motivation
- Deep neural networks
- Convolutional Neural Networks (CNNs)

Face++ 旷视

*** Special thanks Marc'Aurelio Ranzato for the tutorial "Large-Scale Visual Recognition With Deep Learning" in CVPR 2013. All pictures are owned by the authors.*

PART 1: Neural Network Basics

- Motivation
- Deep neural networks
- Convolutional Neural Networks (CNNs)

Features for Recognition

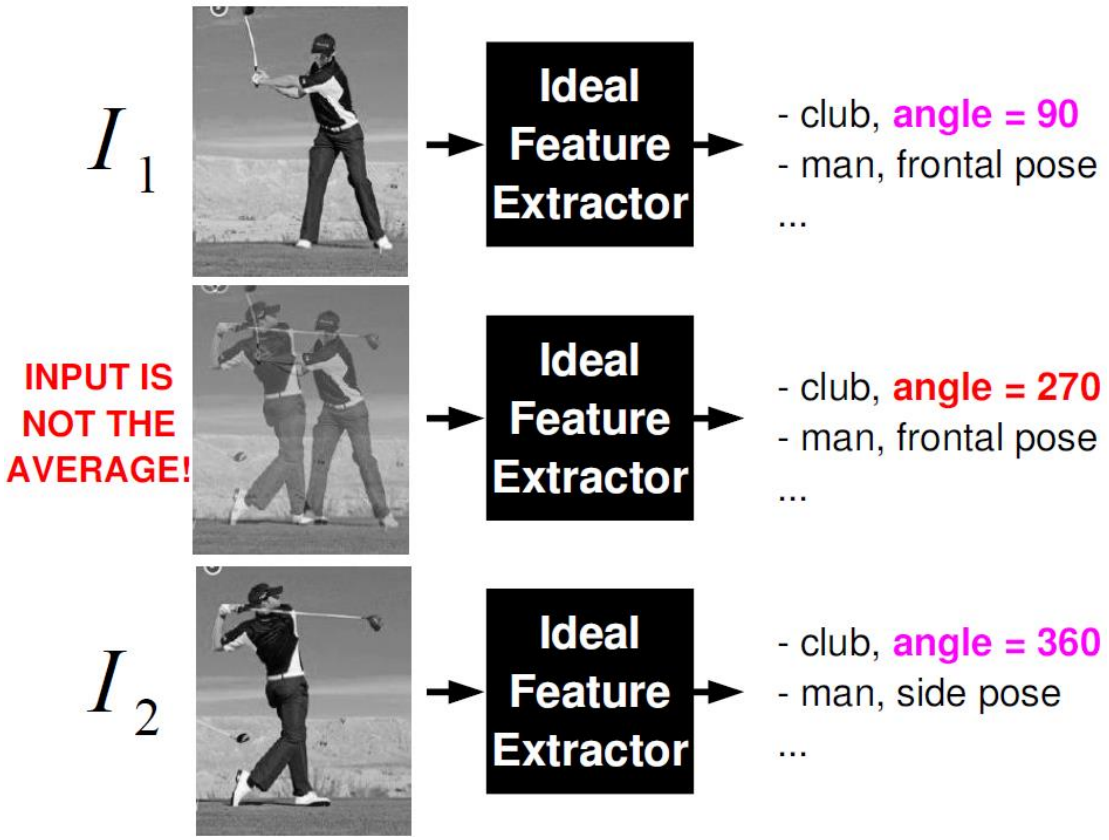
Ideal Feature Extractor

- window, top-left
- clock, top-middle
- shelf, left
- drawing,middle
- statue, bottom left
- …

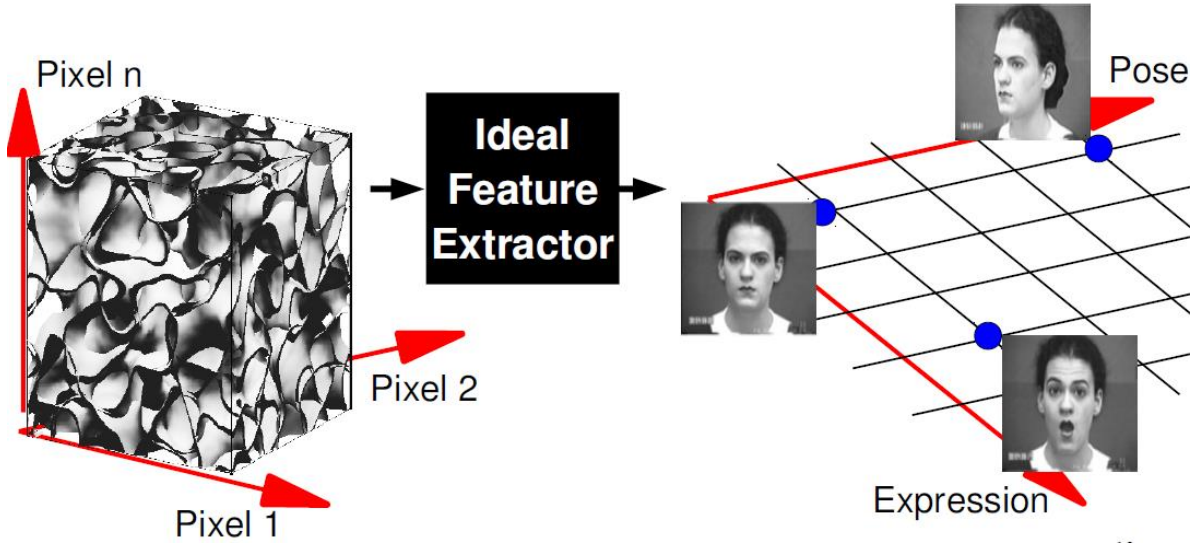
- hat, bottom right

Face++ 旷视

Nonlinear Features vs. Linear Classifiers



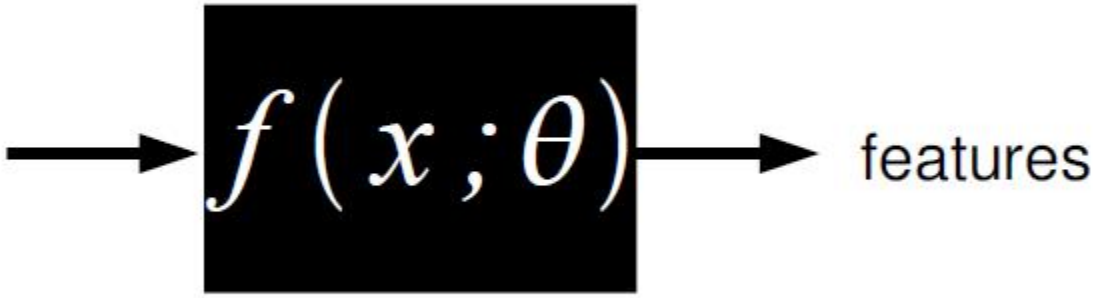
Feature extractor should be nonlinear!



Face++ 旷视

Learning Non-Linear Features

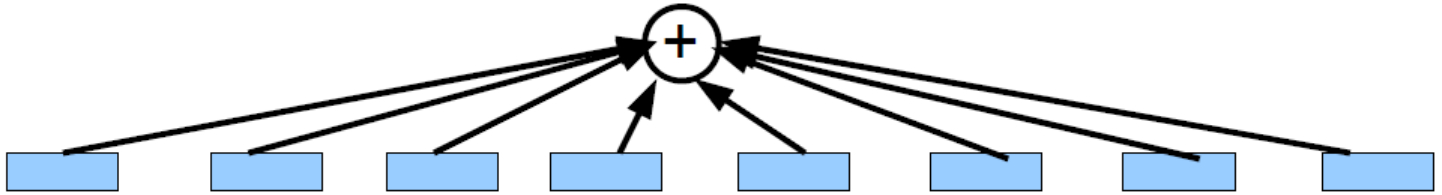
- Q: which class of non-linear functions shall we consider?



Shallow or Deep

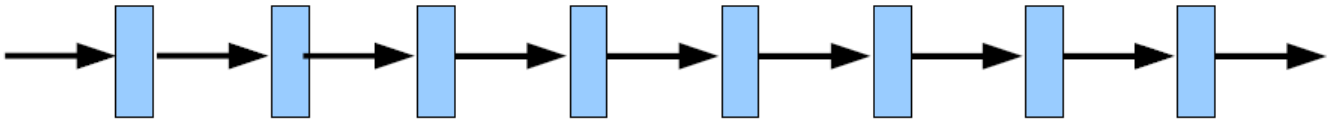
Given a dictionary of simple non-linear functions: $g_1, \ldots, g_n$

Proposal #1: linear combination $f(x) \approx \sum_j g_j$



Shallow

Proposal #2: composition $f(x) \approx g_1(g_2(\ldots g_n(x)\ldots))$

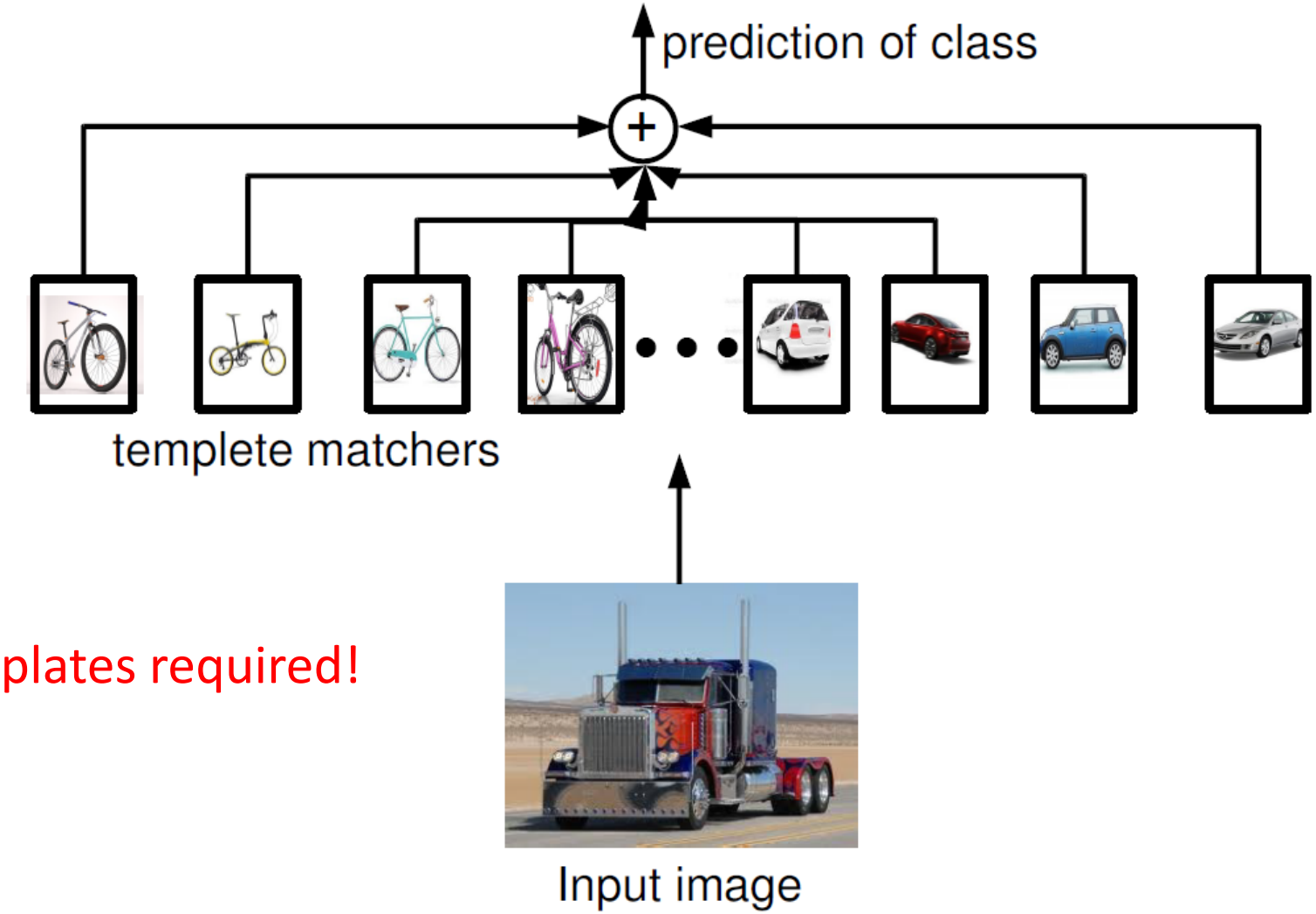


Deep

Face++ 旷视

Linear Combination

- Kernel learning
- Boosting
- …

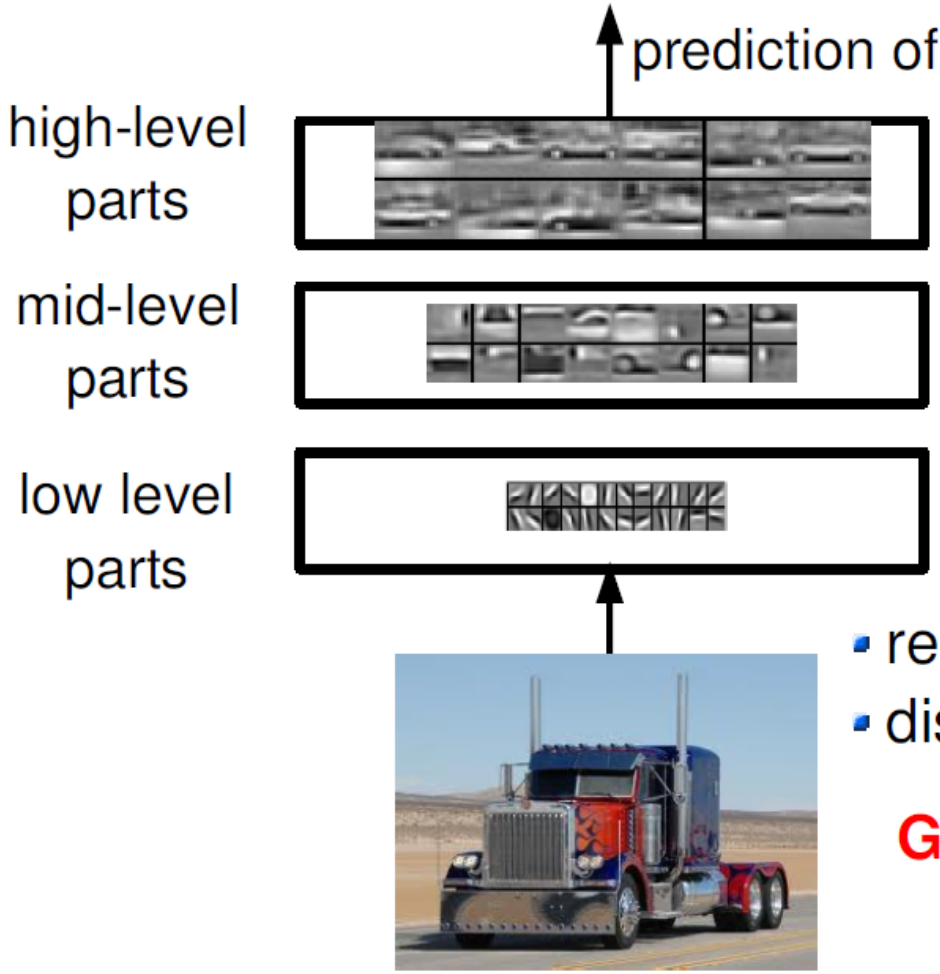


Drawbacks:

Exponential number of templates required!

Face++ 旷视

Composition



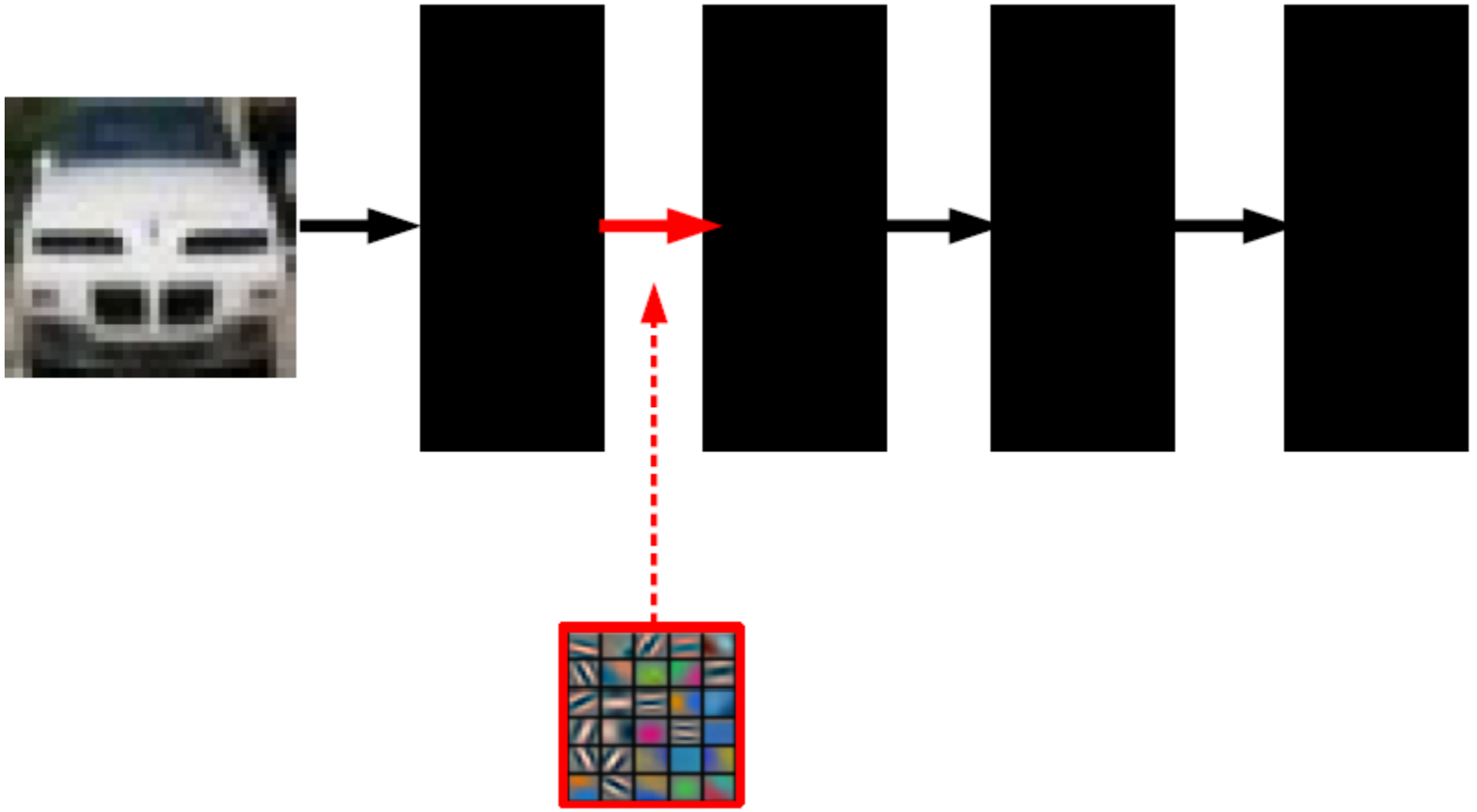
Main Idea of Deep Learning

- reuse of intermediate parts
- distributed representations

GOOD: (exponentially) more efficient

Face++ 旷视

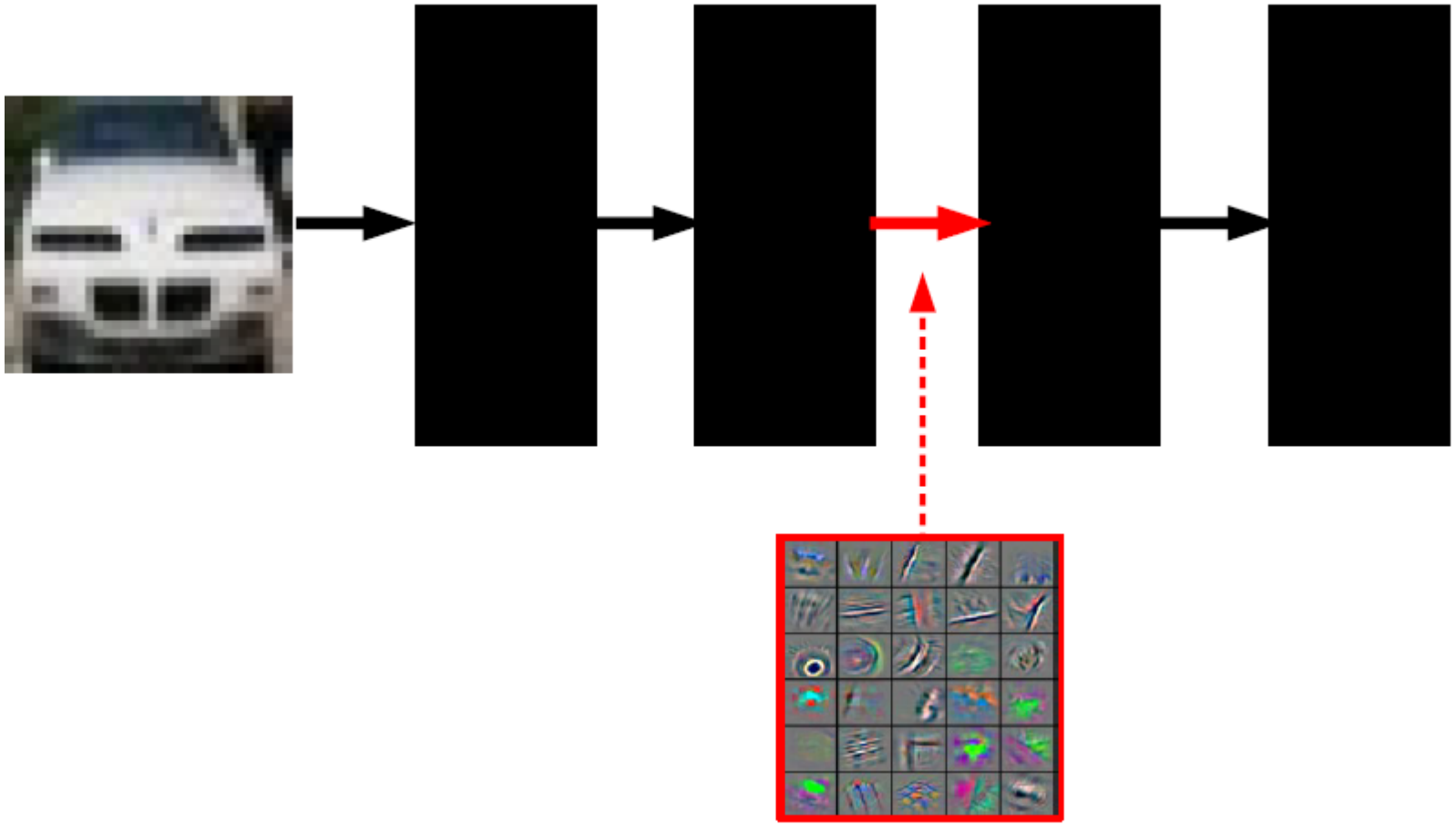
Concepts Reuse in Deep Learning



Face++ 旷视

Zeiler M D, Fergus R. Visualizing and understanding convolutional networks

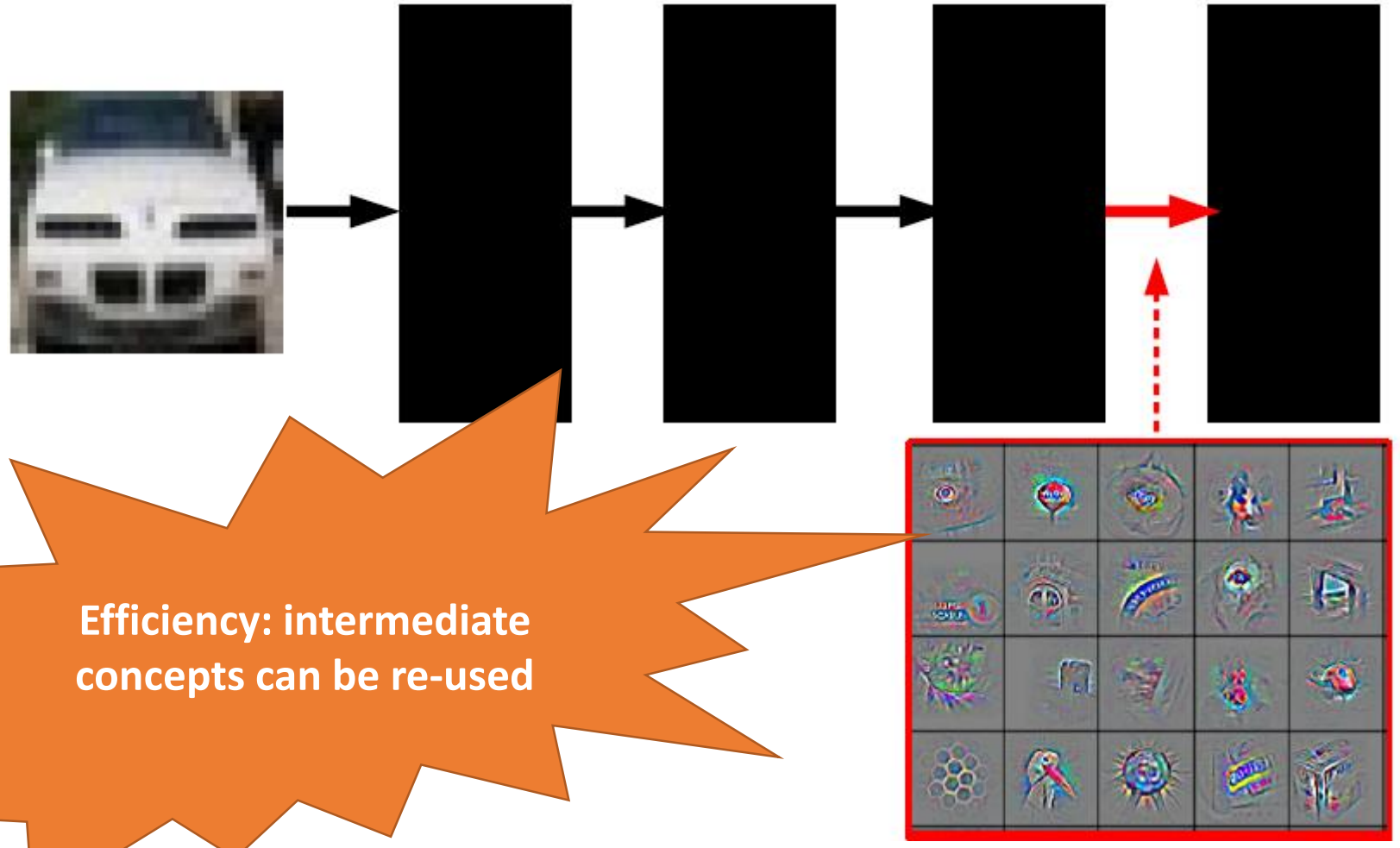
Concepts Reuse in Deep Learning (cont'd)



Face++ 旷视

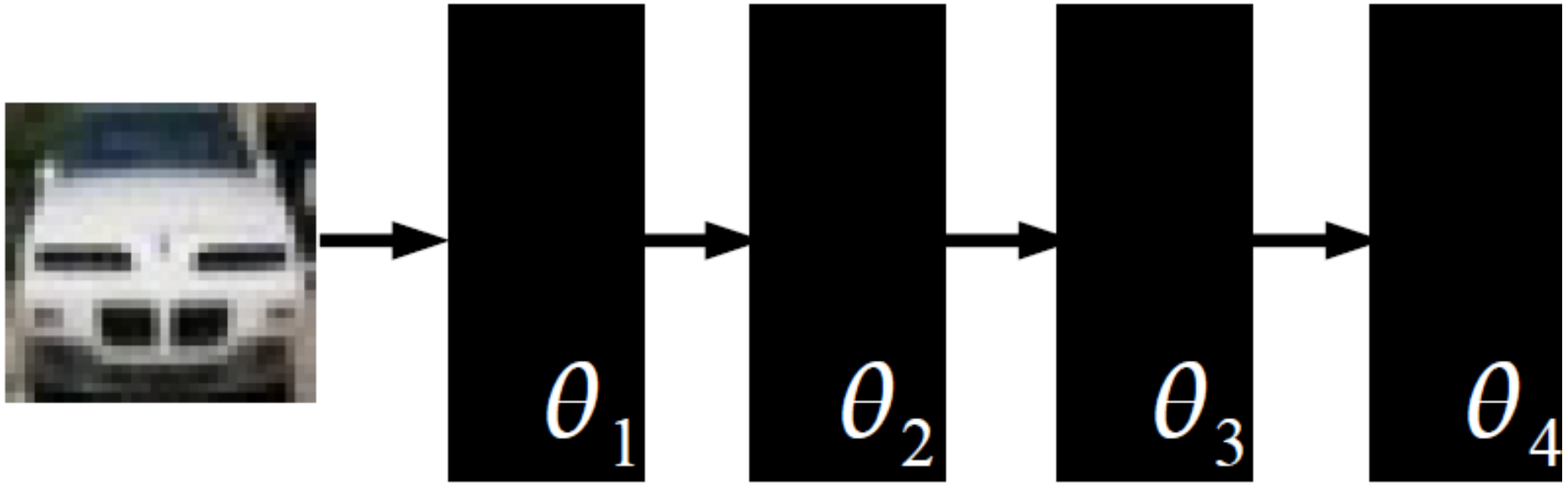
Zeiler M D, Fergus R. Visualizing and understanding convolutional networks

Concepts Reuse in Deep Learning (cont'd)



Face++ 旷视

Deep Learning Framework

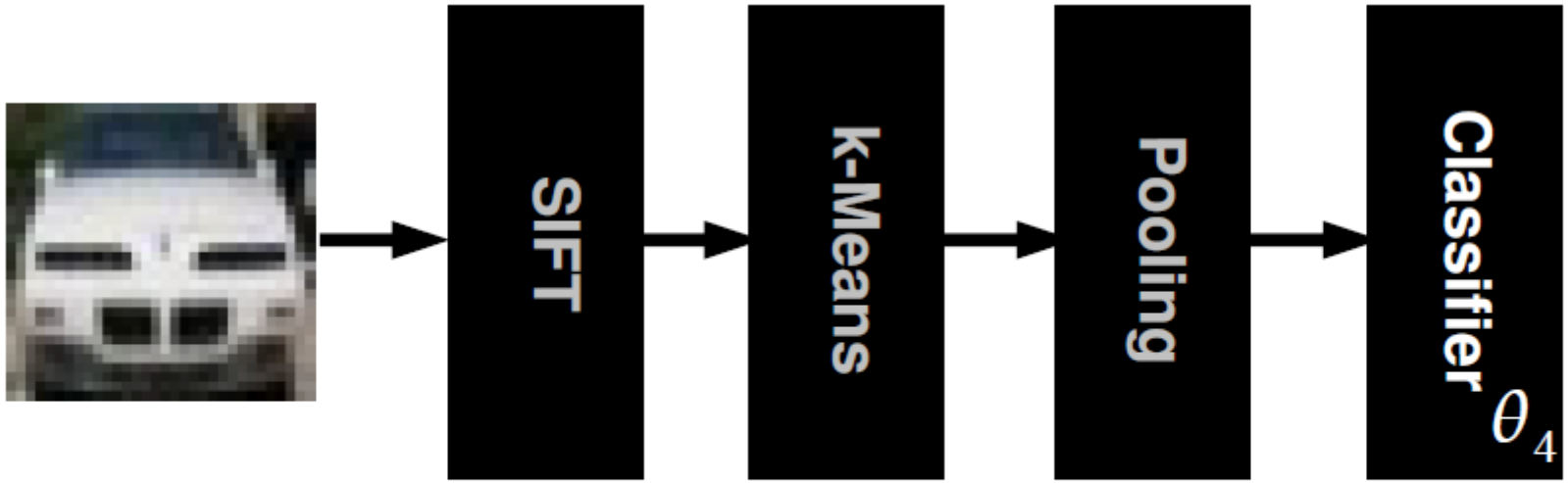


A problem:

Optimization is difficult: non-convex, non-linear system

Face++ 旷视

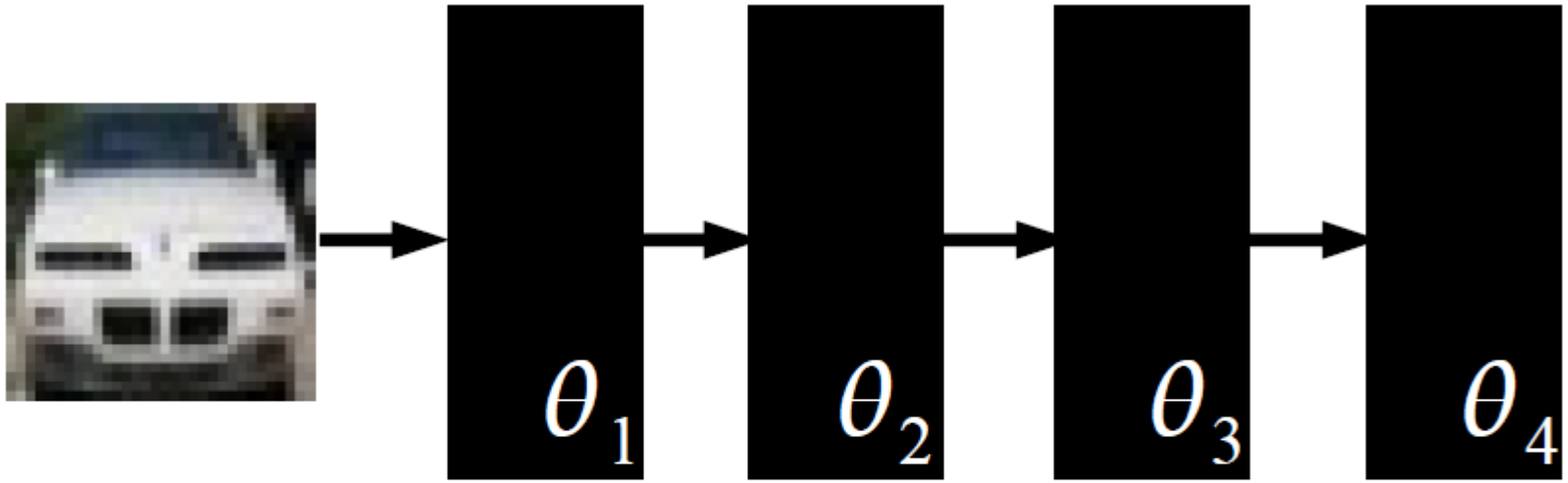
Deep Learning Framework (cont'd)



Solution #1: freeze first N-1 layer (engineer the features)

It makes it shallow!

Deep Learning Framework (cont'd)



Q: What's the feature extractor? And what's the classifier?

A: No distinction, end-to-end learning!

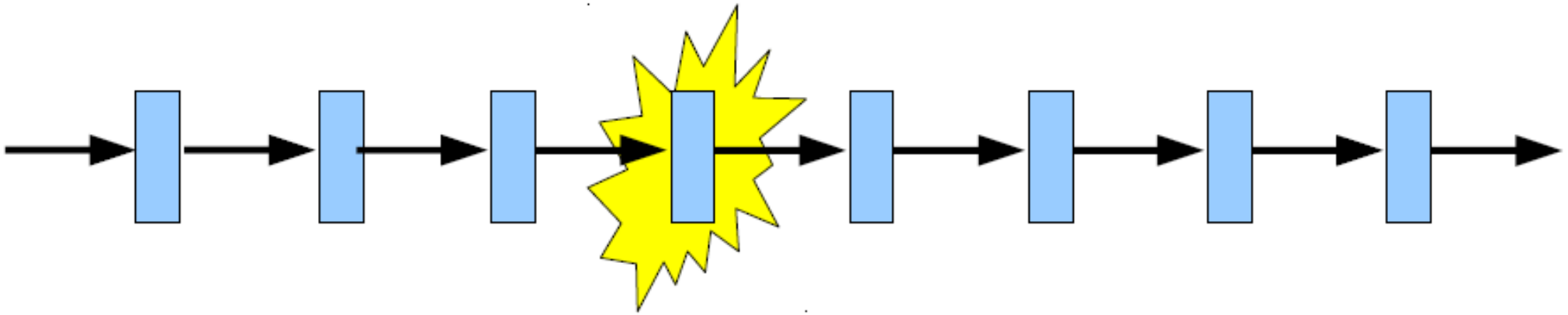
Summary: Key Ideas of Deep Learning

- We need nonlinear system
- We need to learn it from data
- Build feature hierarchies (function composition)
- End-to-end learning

PART 1: Neural Network Basics

- Motivation
- Deep neural networks
- Convolutional Neural Networks (CNNs)

How to Build Deep Network?



“Neuron” or “Layer” Design

Face++ 旷视

Shallow Cases

- Linear Case:
- SVM

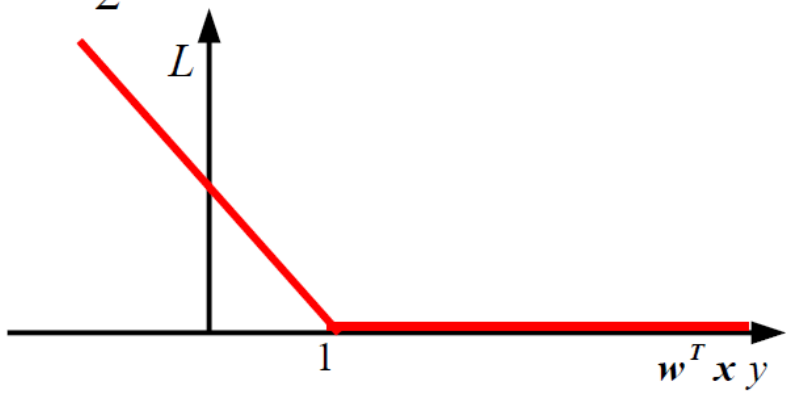
Input: $\boldsymbol{x} \in R^D$

Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $\boldsymbol{w}^T \boldsymbol{x}$

Loss: $L = \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda \, max\,[0, 1 - \boldsymbol{w}^T \boldsymbol{x}\, y]$



Face++ 旷视

Shallow Cases (cont'd)

- Linear Case:
- Logistic Regression

Input: $\boldsymbol{x} \in R^D$

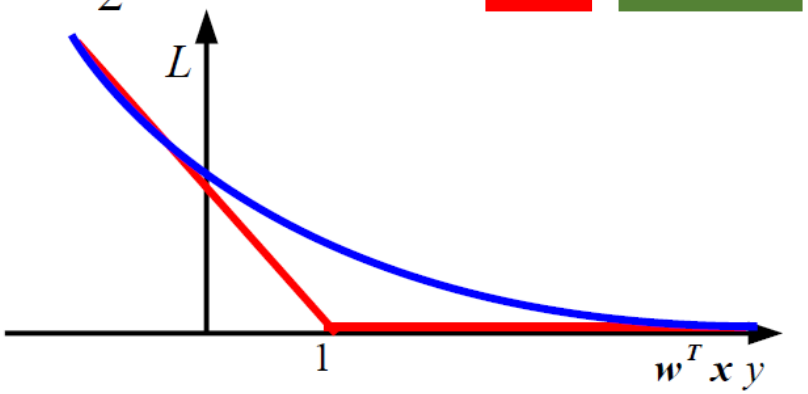
Binary label: $y \in \{-1, +1\}$

Parameters: $\boldsymbol{w} \in R^D$

Output prediction: $p(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}}}$

Loss: $L = \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda \log(1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}\, y))$

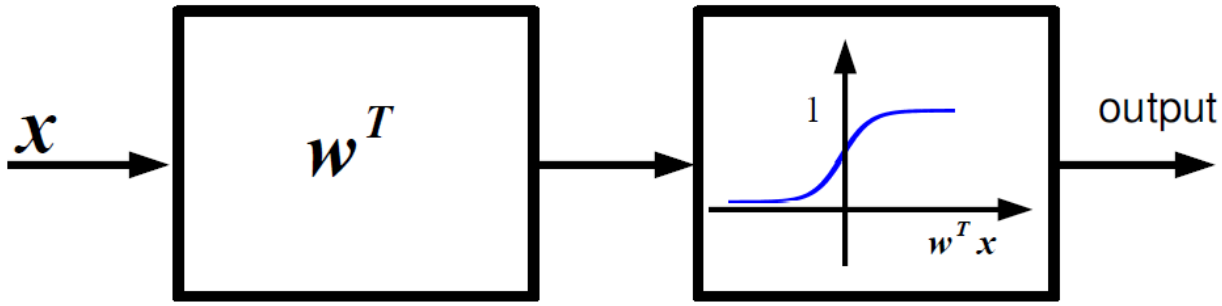
Linear transformation
+ nonlinear activation



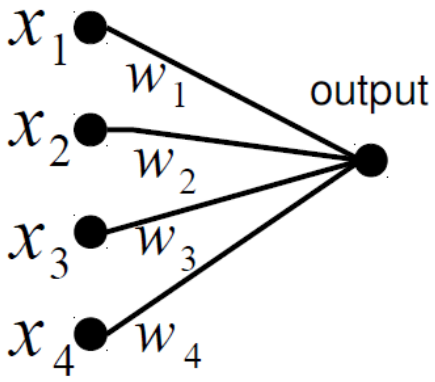
Face++ 旷视

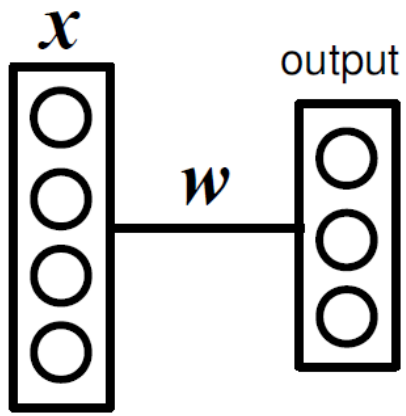
Neuron Design

Single Neuron:

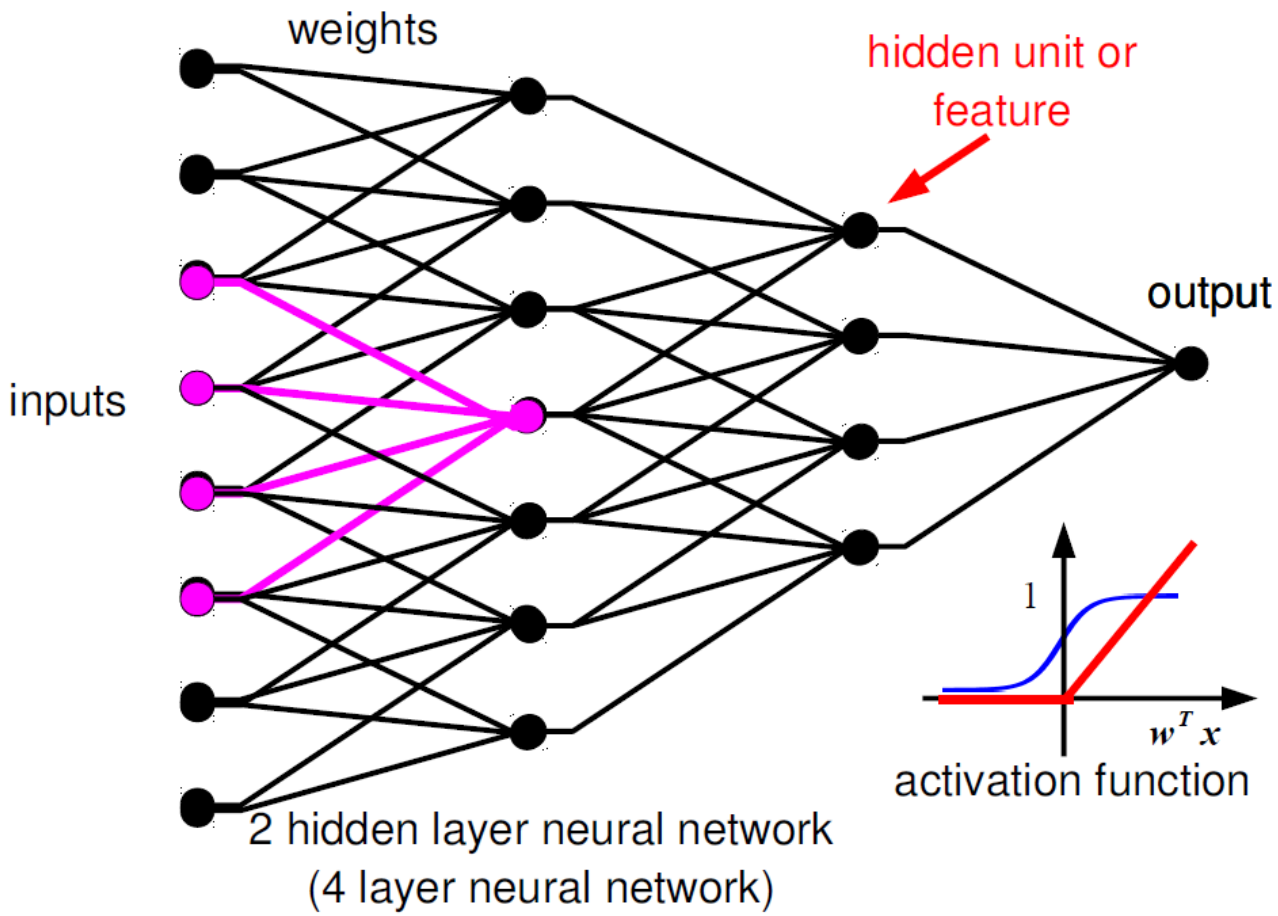


Linear Projection +
Nonlinear Activation

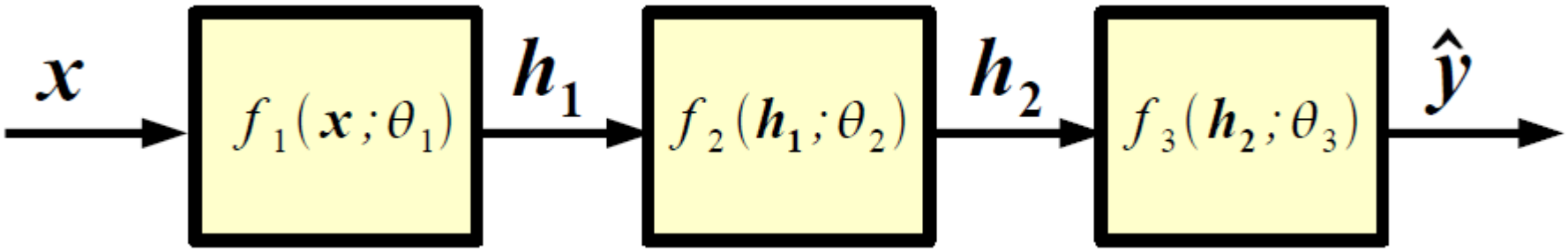




Deep Neuron Network



Deep Neural Network (cont'd)



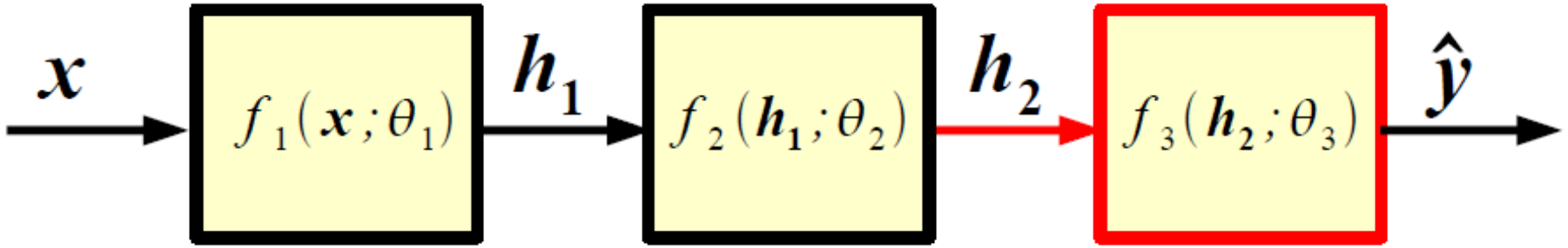
How to determine the parameters?

Gradient-based Training

- For each iteration:

    1. Forward Propagation
    2. Backward Propagation
    3. Update Parameters (Optimization)

Forward Propagation (FPROP)



1) Given x compute: $h_1 = f_1(x;\theta_1)$

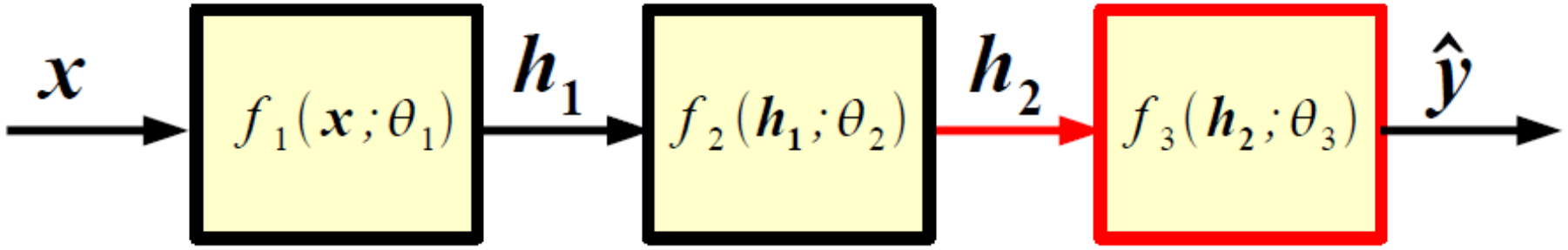
2) Given h_1 compute: $h_2 = f_2(h_1;\theta_2)$

3) Given h_2 compute: $\hat{y} = f_3(h_2;\theta_3)$

For instance,

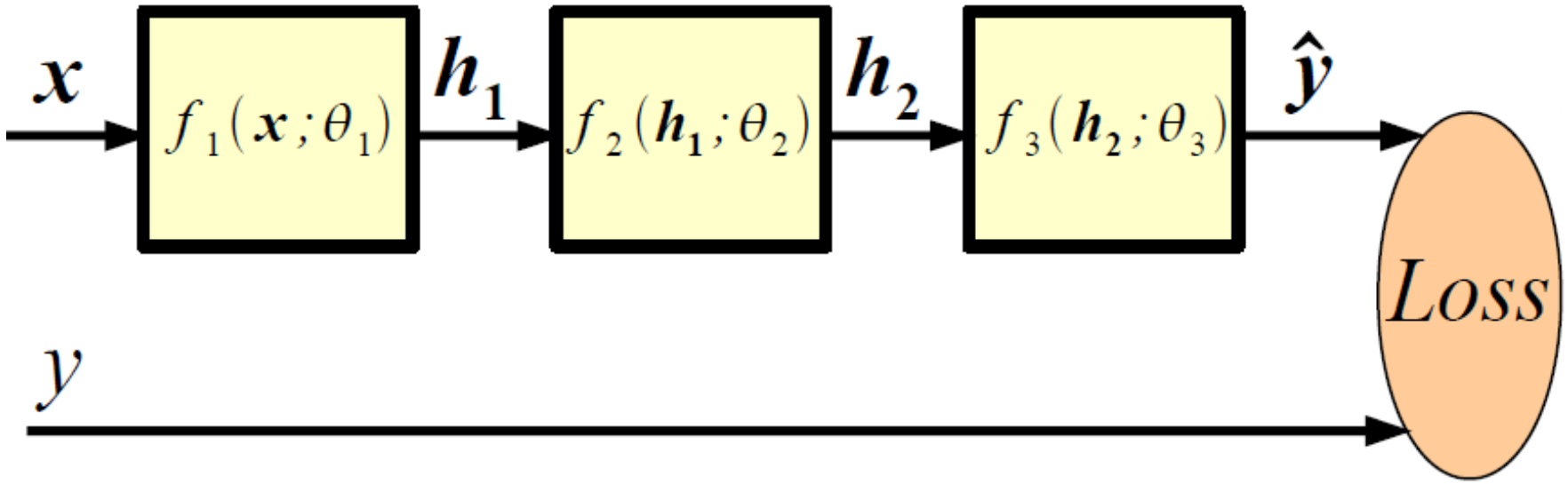
$$\hat{y}_i = p(class = i \mid x) = \frac{e^{W_{3i} h_2 + b_{3i}}}{\sum_k e^{W_{3k} h_2 + b_{3k}}}$$

Forward Propagation (FPROP)



This is the typical processing at test time. At training time, we need to compute an error measure and tune the parameters to decrease the error.

Loss Function

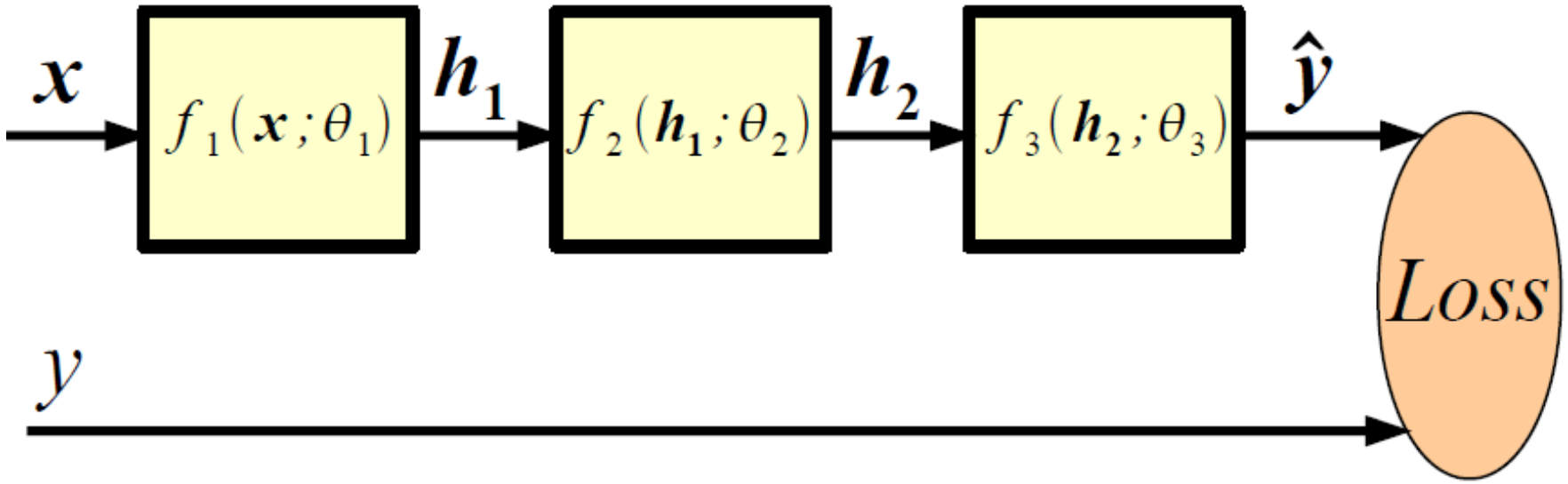


The measure of how well the model fits the training set is given by a suitable loss function: $L(x, y; \theta)$

For instance,

$$L(x, y=k; \theta) = -\log(p(class=k|x))$$

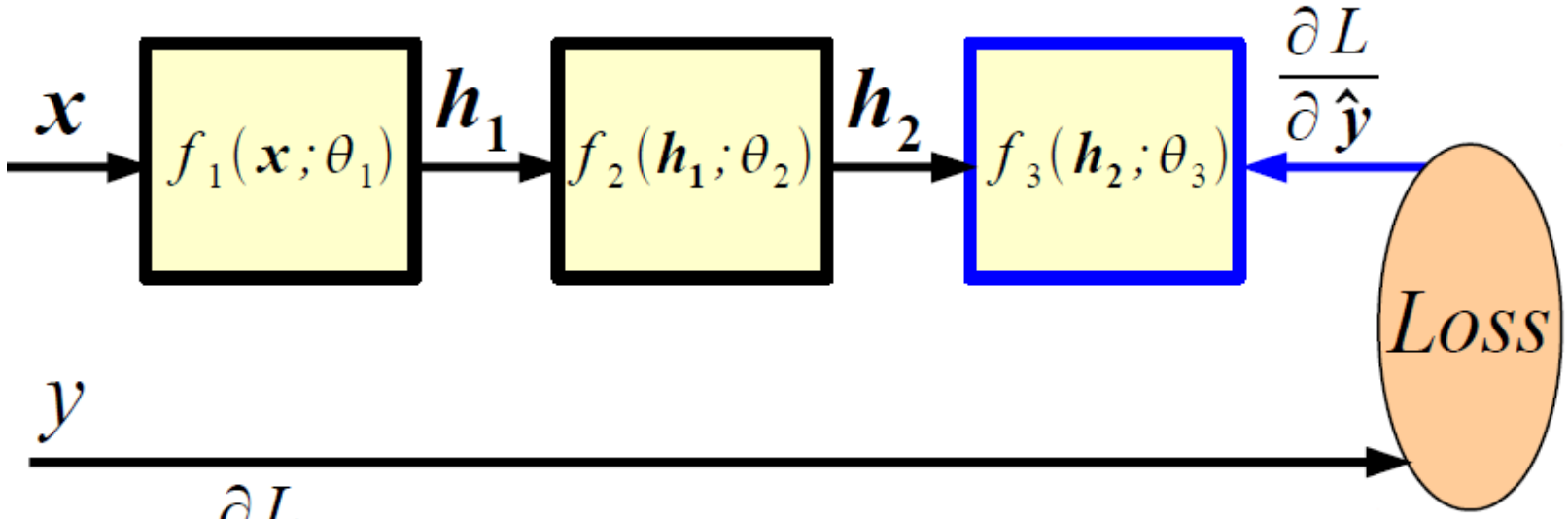
Loss Function



Q: how to tune the parameters to decrease the loss?
A: If loss is (a.e.) differentiable we can compute gradients. We can use chain-rule, a.k.a. **back-propagation**, to compute the gradients w.r.t. parameters at the lower layers.

Face++ 旷视

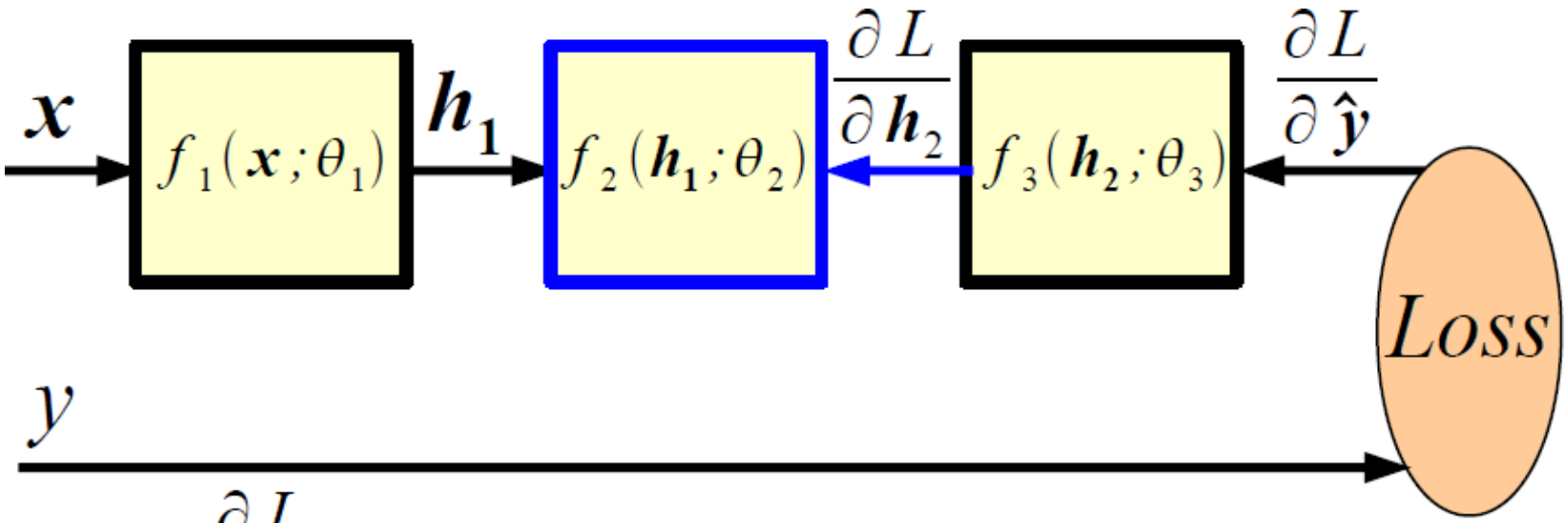
Backward Propagation (BPROP)



Given $\frac{\partial L}{\partial \hat{y}}$ and assumiing the Jacobian of each module is easy to compute, then we have:

$$\frac{\partial L}{\partial \theta_3} = (\hat{y} - y)\ h_2' \qquad \frac{\partial L}{\partial h_2} = (\hat{y} - y)\ \theta_3'$$

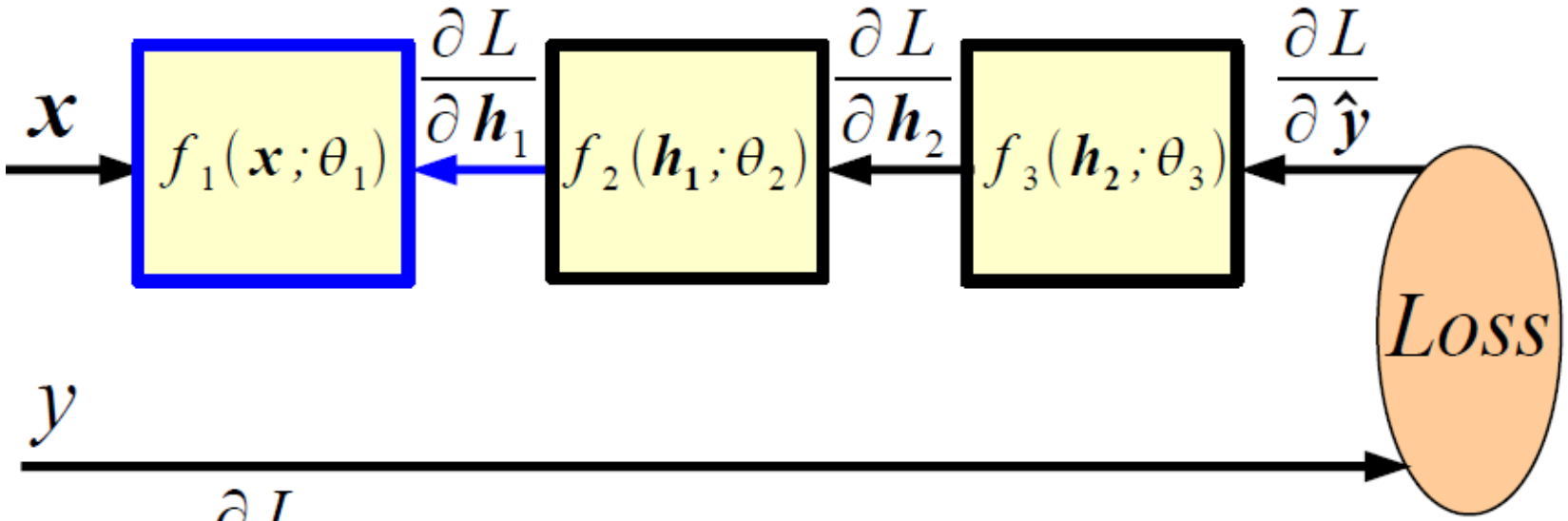
Backward Propagation (BPROP) (cont'd)



Given $\frac{\partial L}{\partial h_2}$ we can compute now:

$$\frac{\partial L}{\partial \theta_2} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial \theta_2} \qquad \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$

Backward Propagation (BPROP) (cont'd)



Given $\frac{\partial L}{\partial h_1}$ we can compute now:

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial \theta_1}$$

Optimization

- Stochastic Gradient Descent (on mini-batches):

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}, \eta \in R$$

- Stochastic Gradient Descent with Momentum:

$$\theta \leftarrow \theta - \eta \Delta$$
$$\Delta \leftarrow 0.9 \Delta + \frac{\partial L}{\partial \theta}$$

Summary: Key Ideas of Deep Neural Networks

- Neural Net = stack of feature detectors
- F-Prop / B-Prop
- Learning by SGD

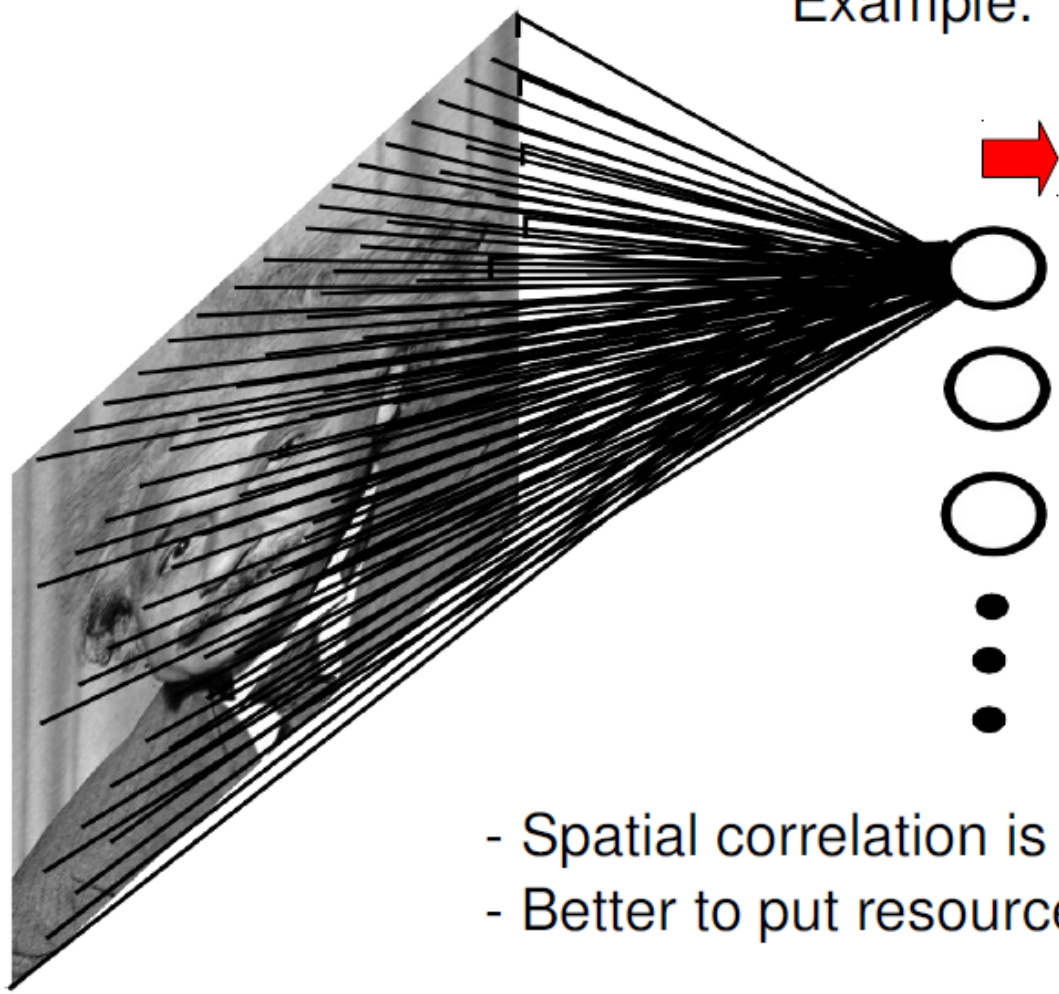
PART 1: Neural Network Basics

- Motivation
- Deep neural networks
- Convolutional Neural Networks (CNNs)

Deep Neural Networks on Images

- How to apply a neural network on 2D or 3D inputs?

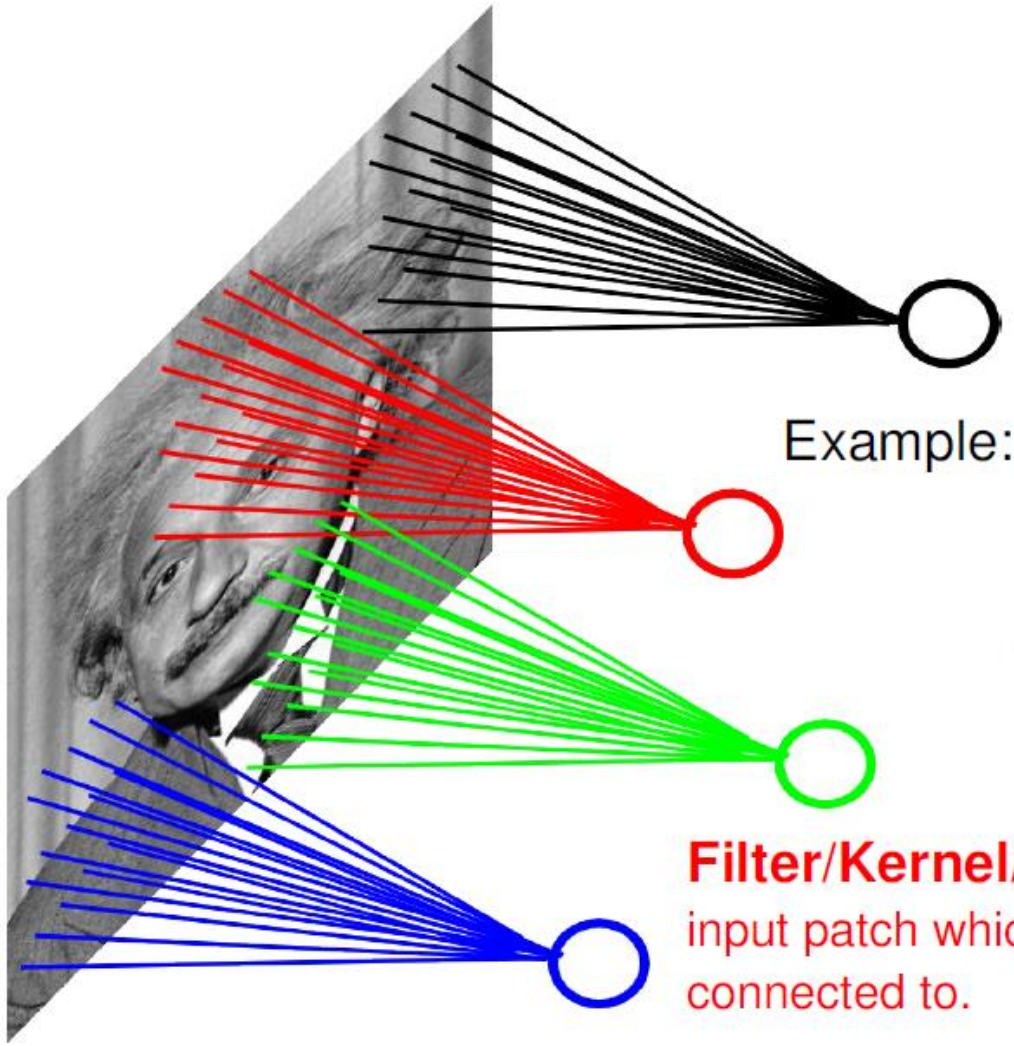
Fully-connected Net



Example: 1000x1000 image
1M hidden units
10^12 parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!

Locally-connected Net



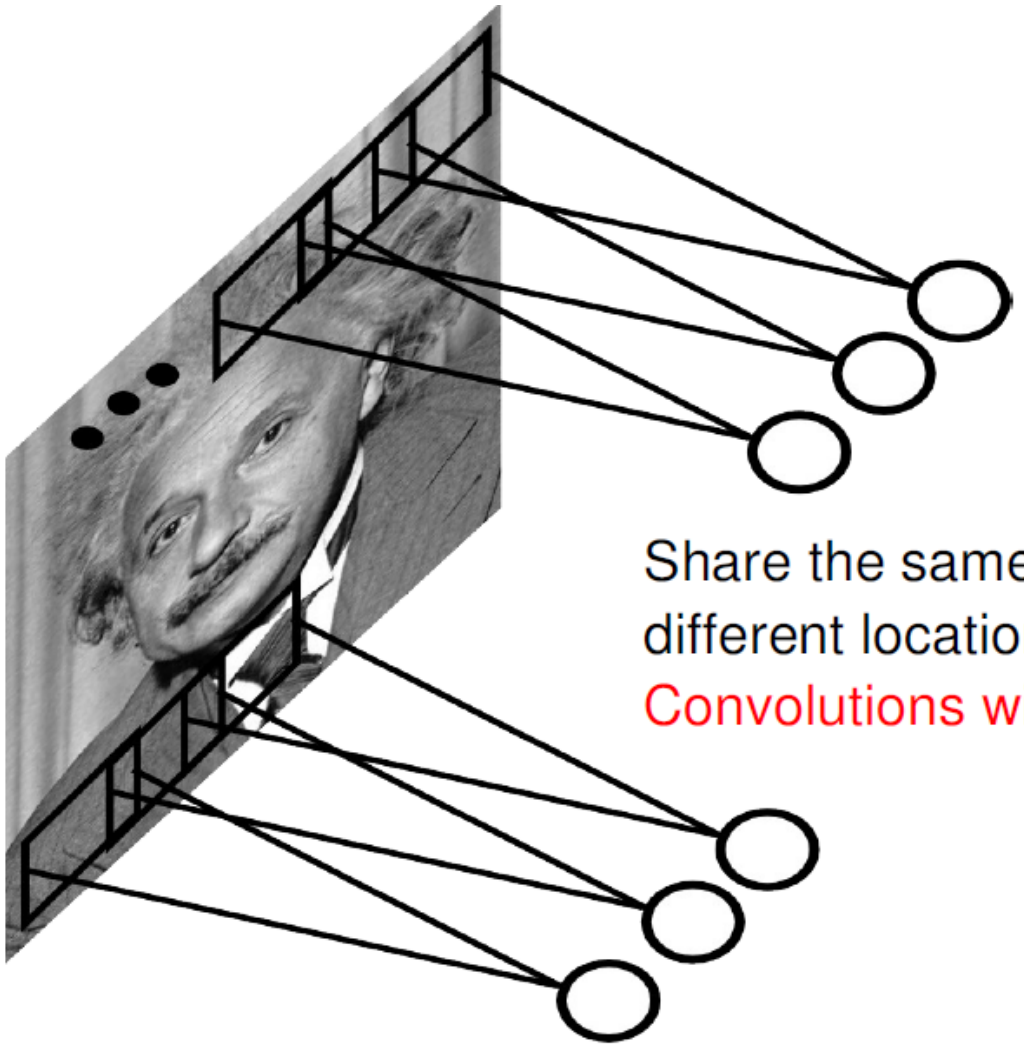
STATIONARITY?
Statistics are similar at different locations (translation invariance)

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Filter/Kernel/Receptive field:
input patch which the hidden unit is connected to.

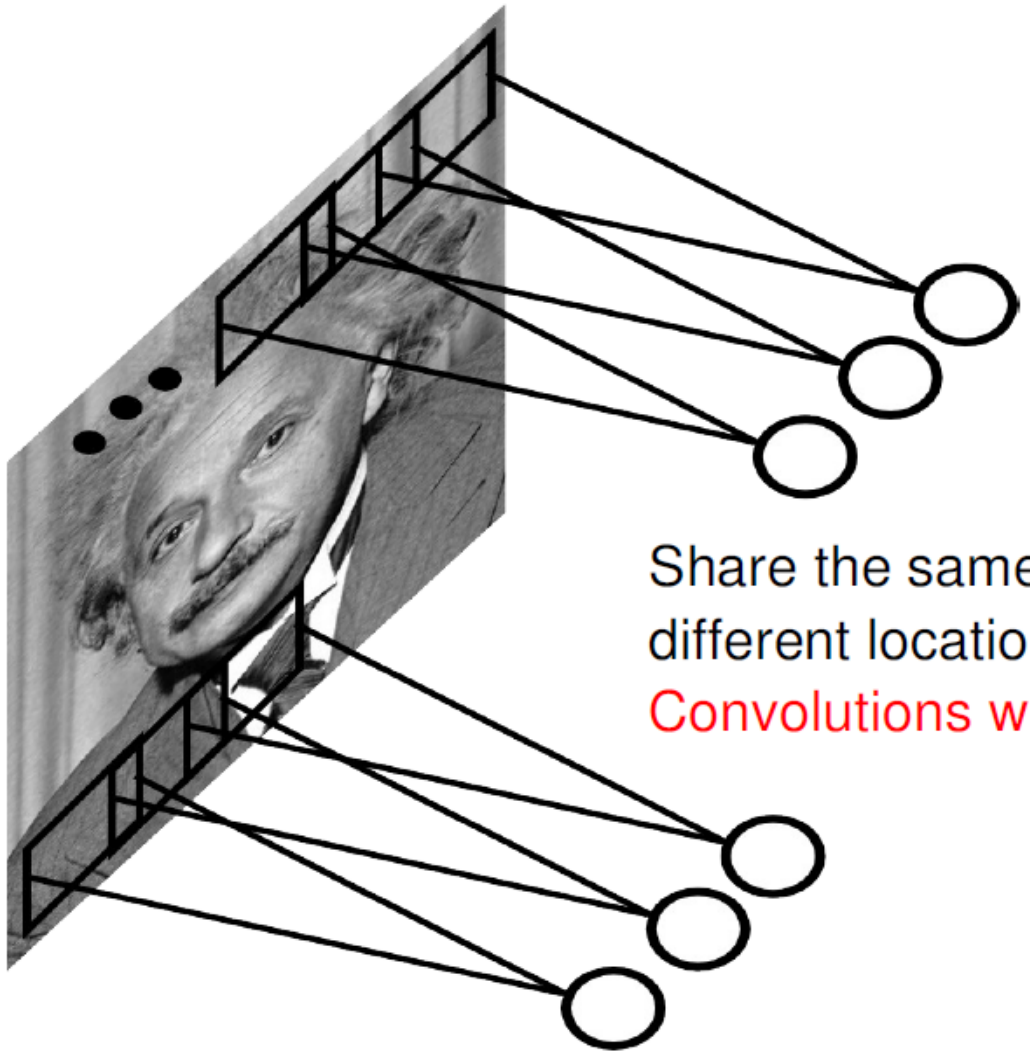
Face++ 旷视

Convolutional Net



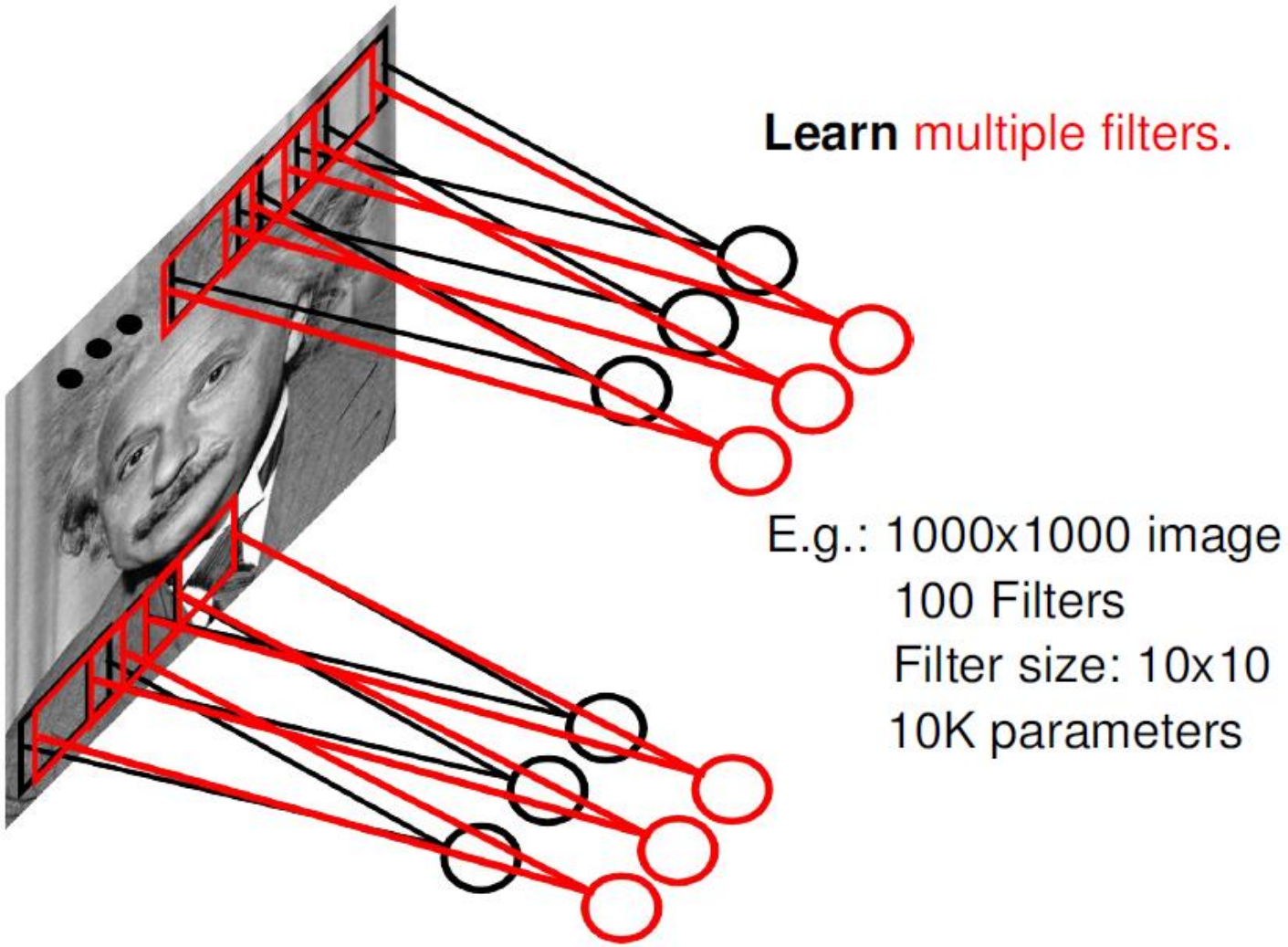
Share the same parameters across different locations:
Convolutions with learned kernels

Convolutional Net (cont'd)

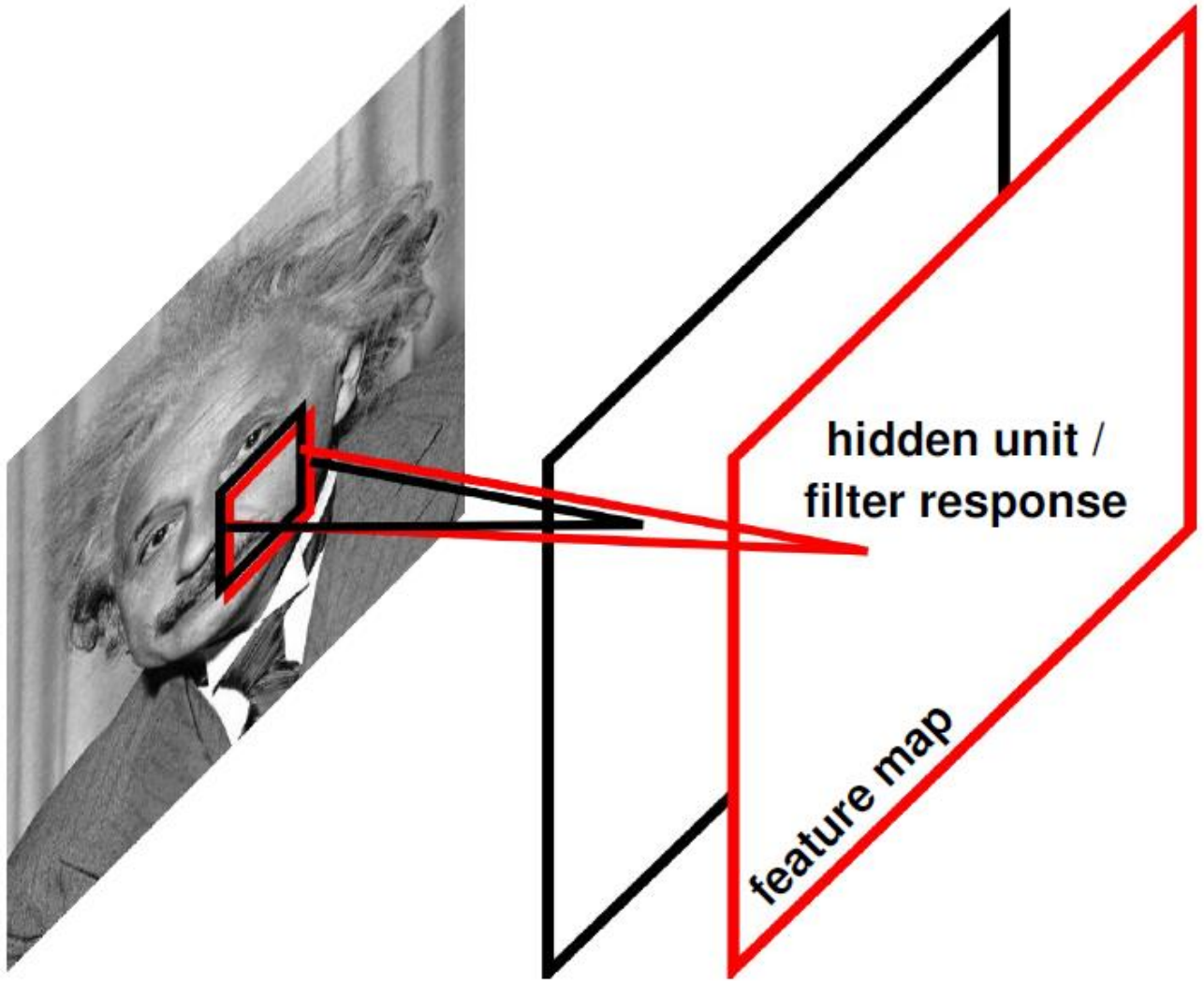


Share the same parameters across different locations:
Convolutions with learned kernels

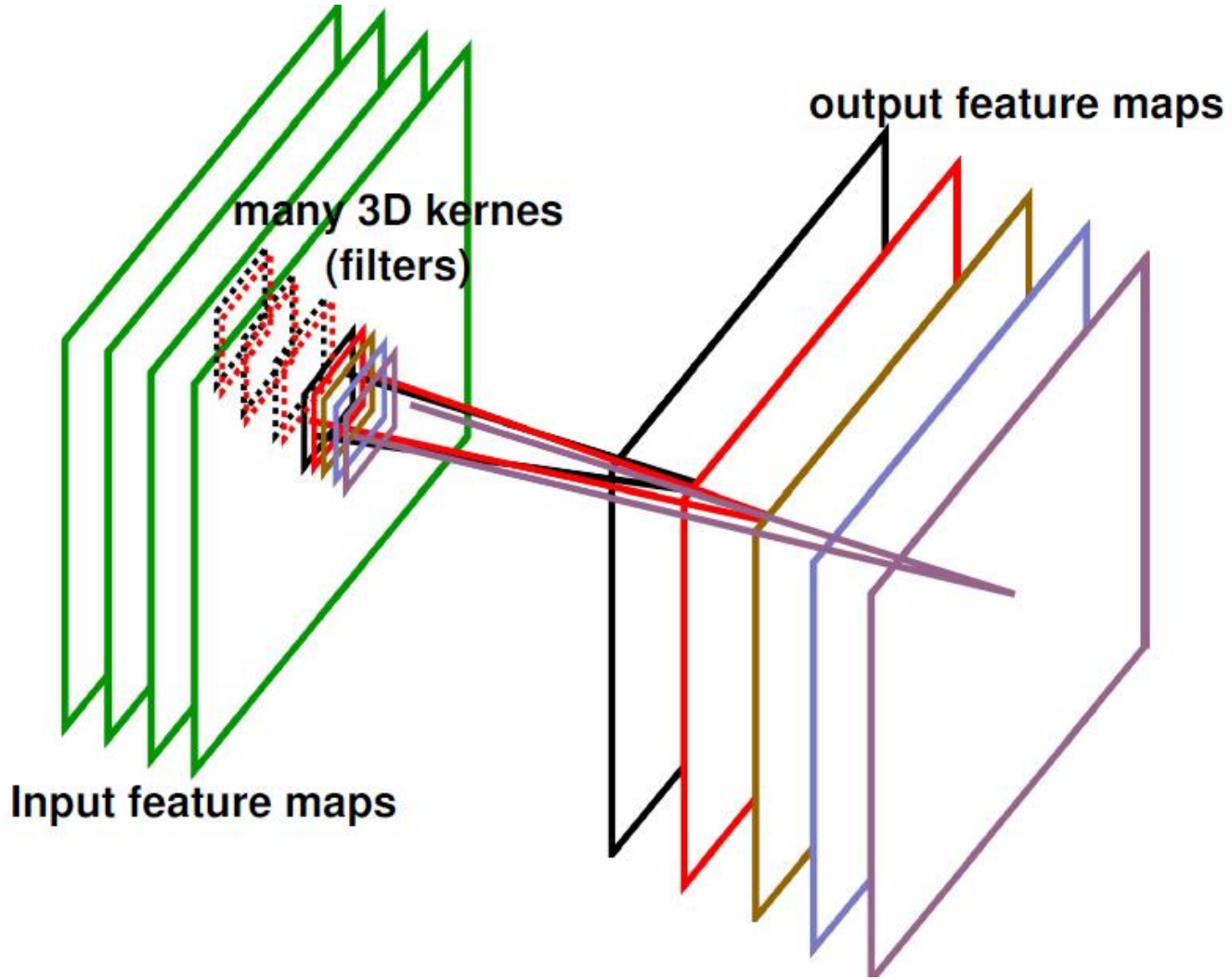
Convolutional Net (cont'd)



Convolutional Net (cont'd)

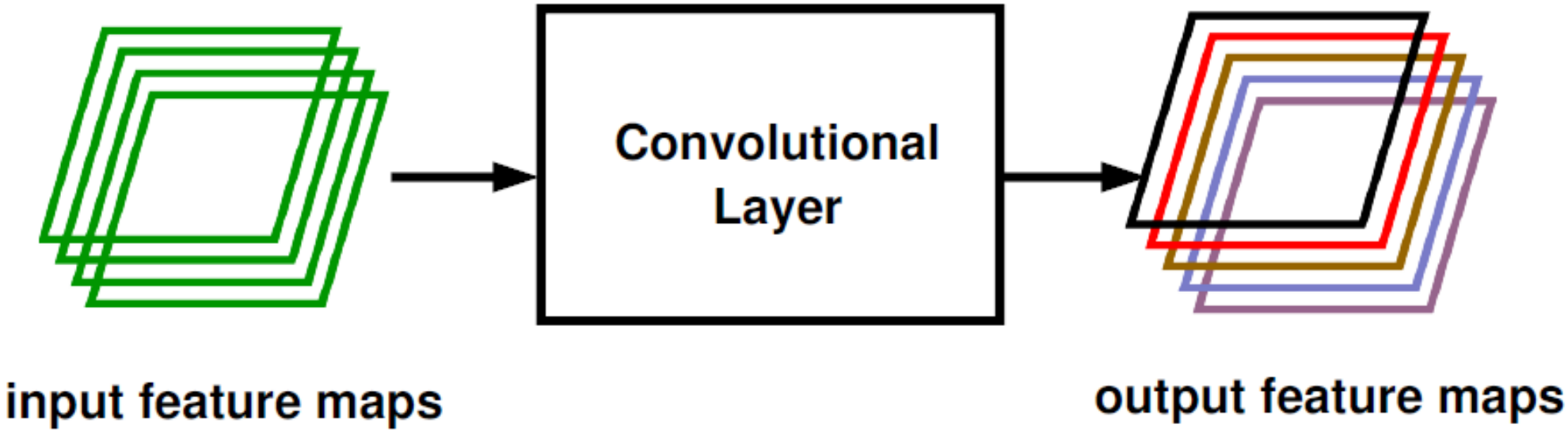


Convolutional Layer



Face++ 旷视

Convolutional Layer (cont'd)



Summary: Key Ideas of Convolutional Nets

- A standard neural net applied to images:
  - scales quadratically with the size of the input
  - does not leverage stationarity

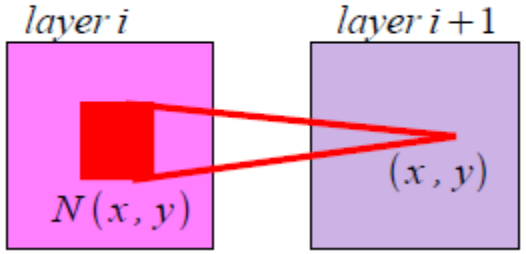
- Solution:
  - connect each hidden unit to a small patch of the input
  - share the weight across hidden units

- This is called: **convolutional network.**

Other Layers

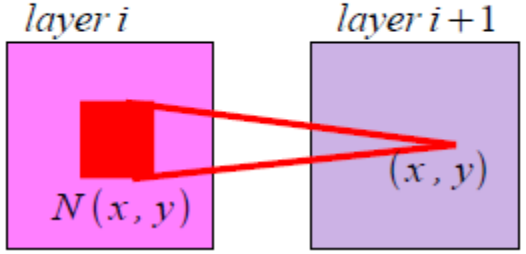
- Over the years, some new modules have proven to be very effective when plugged into conv-nets:

- **Pooling** (average, L2, max)



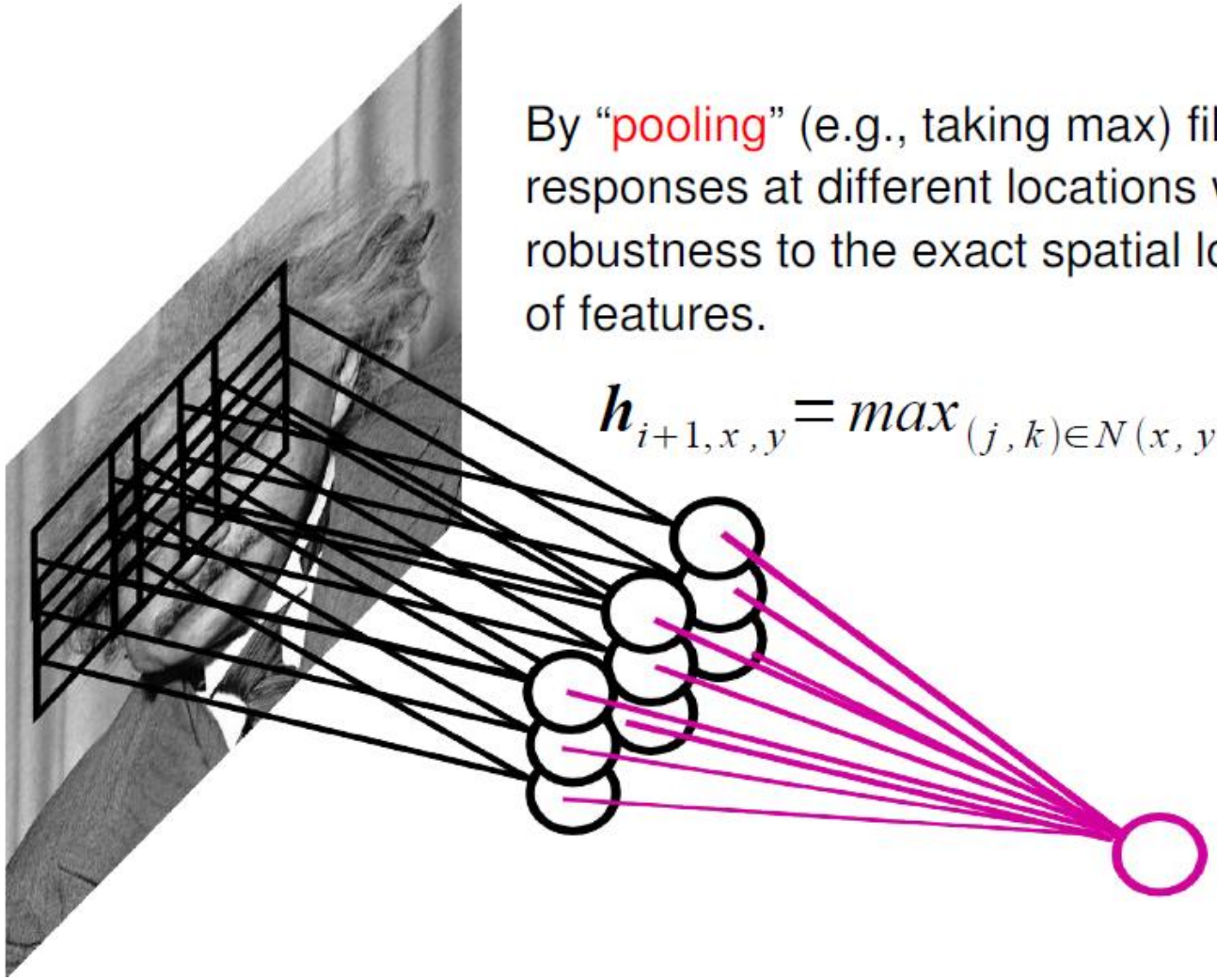
$$h_{i+1,x,y} = max_{(j,k) \in N(x,y)} h_{i,j,k}$$

- **Local Contrast Normalization** (over space / features)



$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,x,y}}{\sigma_{i,x,y}}$$

Pooling Layer

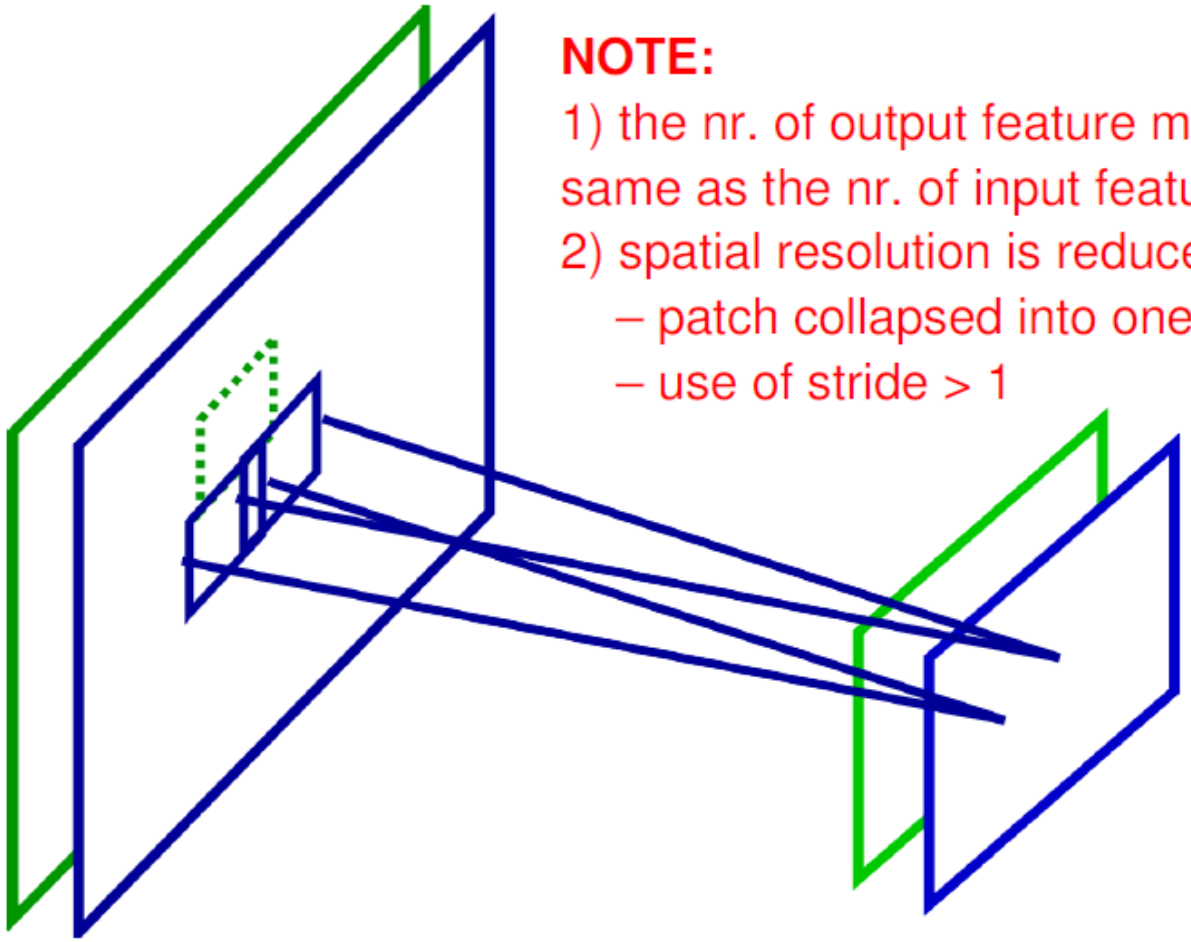


By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

$$\boldsymbol{h}_{i+1,x,y} = max_{(j,k)\in N(x,y)} \boldsymbol{h}_{i,j,k}$$

Face++ 旷视

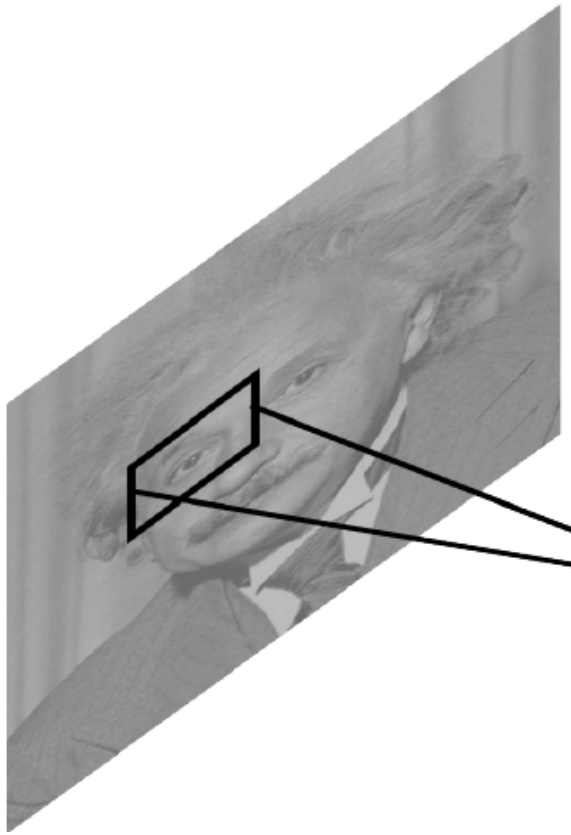
Pooling Layer



Face++ 旷视

Local Contrast Normalization Layer

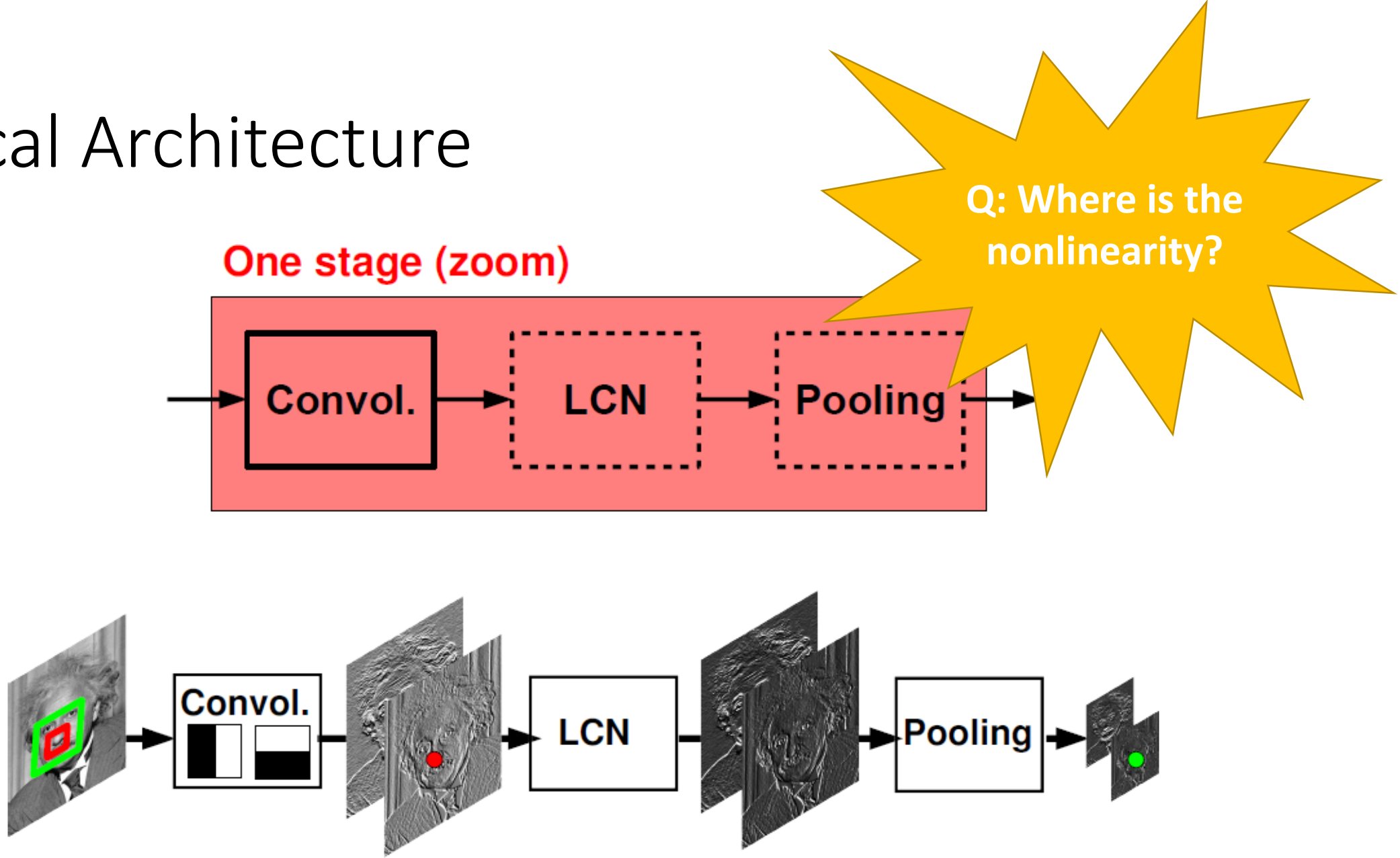
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



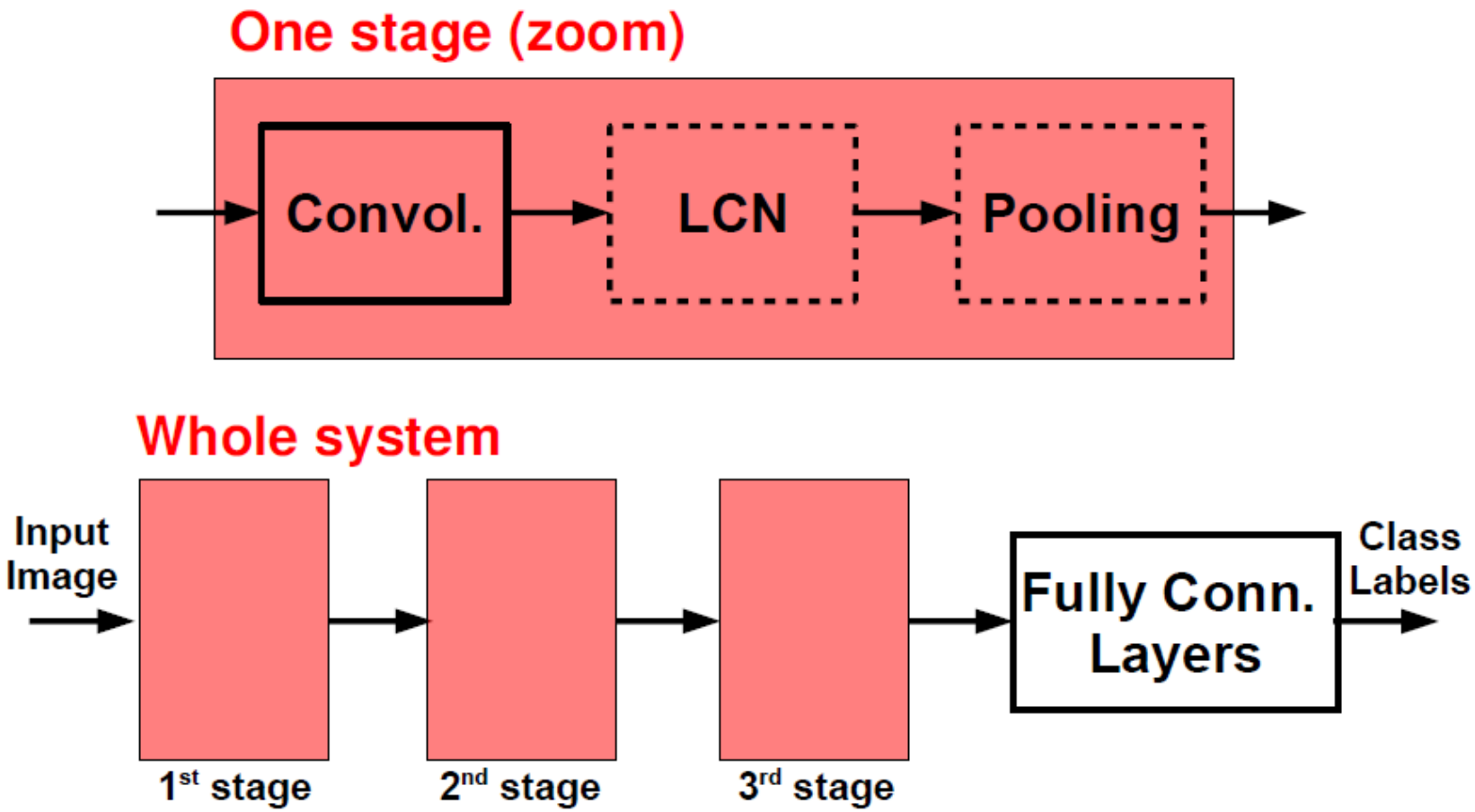
Performed also across features and in the higher layers.

Effects:
- improves invariance
- improves optimization
- increases sparsity

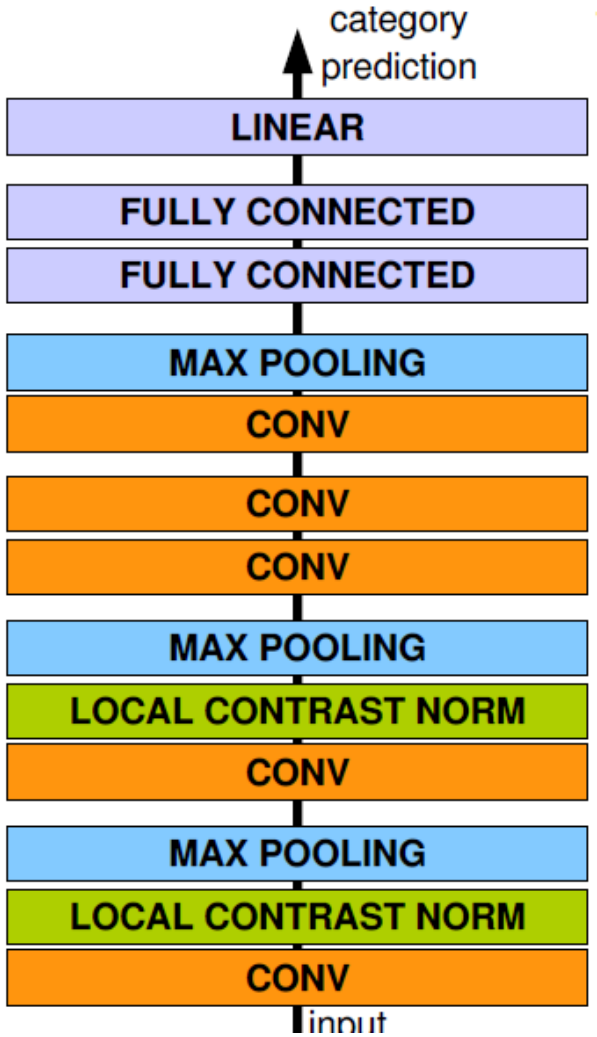
Typical Architecture



Typical Architecture (cont'd)



Conv Architecture Example (AlexNet)



Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Convolutional Nets: Training

- All layers are differentiable (a.e.). We can use standard back-propagation.

- Algorithm:

  Given a small mini-batch

  1. F-PROP
  2. B-PROP
  3. PARAMETER UPDATE

Summary: Key Ideas of Conv Nets

- Conv. Nets have special layers like:
  - – pooling, and
  - – local contrast normalization
  - Back-propagation can still be applied.

- These layers are useful to:
  - – reduce computational burden
  - – increase invariance
  - – ease the optimization

PART 2: Architecture Design

- Overview
- Structure design
- Layer design
- Architecture for special tasks

Face++ 旷视

PART 2: Architecture Design

- Overview
- Structure design
- Layer design
- Architecture for special tasks

Face++ 旷视

Architecture Design

- What?
  - Network topology
  - Layer functions
  - Hyper-parameters
  - Optimization algorithms

  …
- Why?
  - Difficult to determine the optimal structures
  - Requirements of different applications, datasets or limitations

Architecture Design (cont'd)

- How?
  - Manually
  - Automatically

- Objective
  - Representation capability
  - Robustness, anti-overfitting
  - Computation or parameter efficiency
  - Ease of optimization

  …

Face++ 旷视

PART 2: Architecture Design

- Overview
- Structure design
- Layer design
- Architecture for special tasks

Face++ 旷视

Benchmark: ImageNet Dataset

- 1K classes (for ILSVRC competition)
- 1.2M+ training images, 50K validation images, 100K test images
- ILSVRC competition

Difficulty

- Fine-grained classes
- Large variation
- Costly training

Benchmark: ImageNet Dataset

- 1K classes (for ILSVRC competition)
- 1.2M+ training images, 50K validation images, 100K test images
- ILSVRC competition

Difficulty

- Fine-grained classes
- Large variation
- Costly training

?

Walker hound

English foxhound

Beagle

Benchmark: ImageNet Dataset

- 1K classes (for ILSVRC competition)
- 1.2M+ training images, 50K validation images, 100K test images
- ILSVRC competition

Difficulty

- Fine-grained classes
- Large variation
- Costly training




Lemon

Benchmark: ImageNet Dataset

- 1K classes (for ILSVRC competition)
- 1.2M+ training images, 50K validation images, 100K test images
- ILSVRC competition

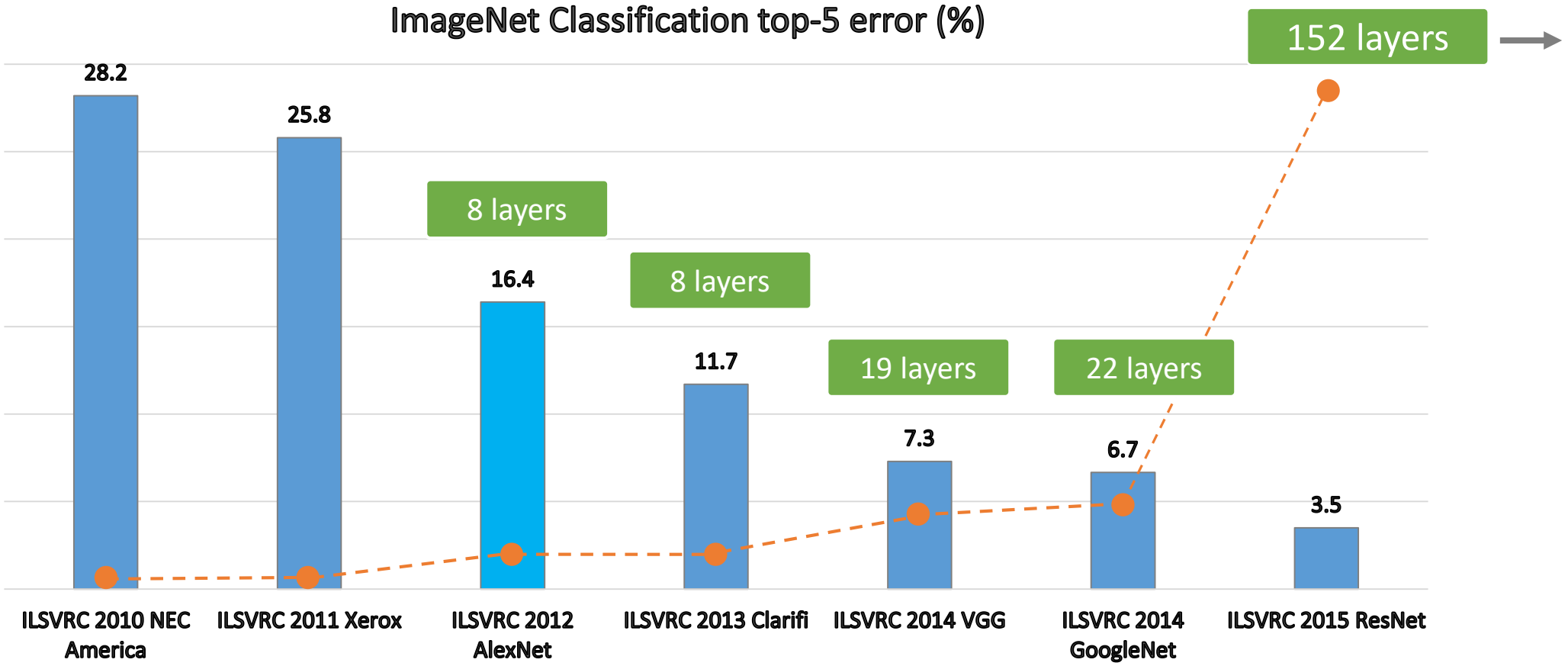
Difficulty

- Fine-grained classes
- Large variation
- Costly training

8x Nvidia K40 GPU

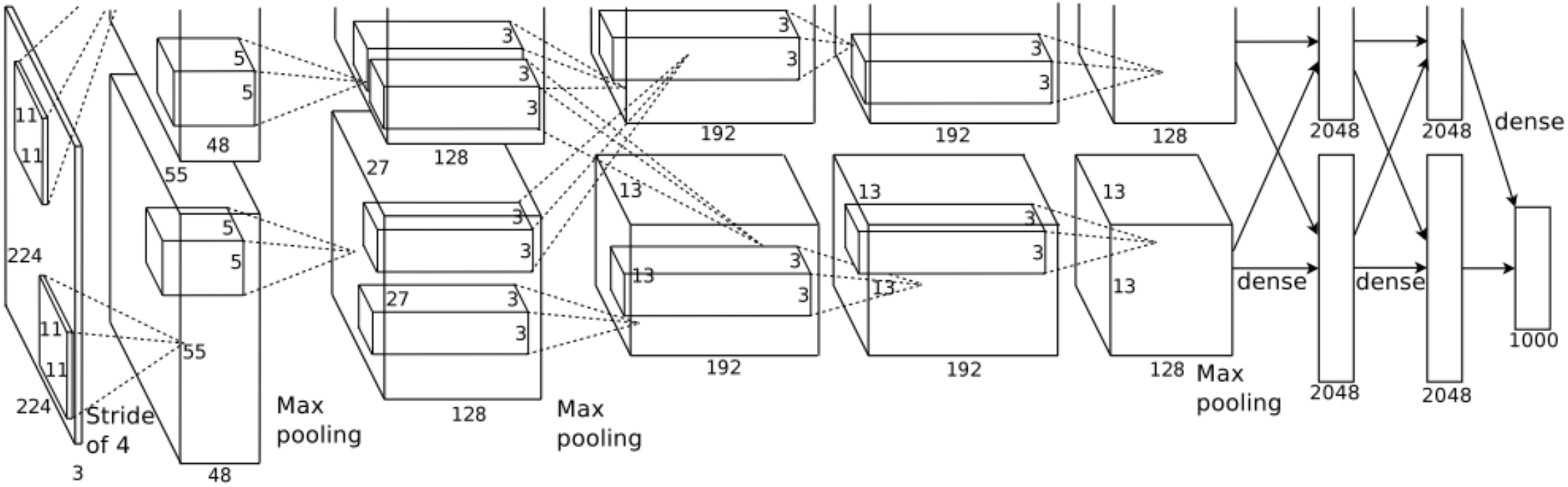
Still take 20~30 days!

Recent Nets – ImageNet Classification Scores



Face++ 旷视

AlexNet



Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks

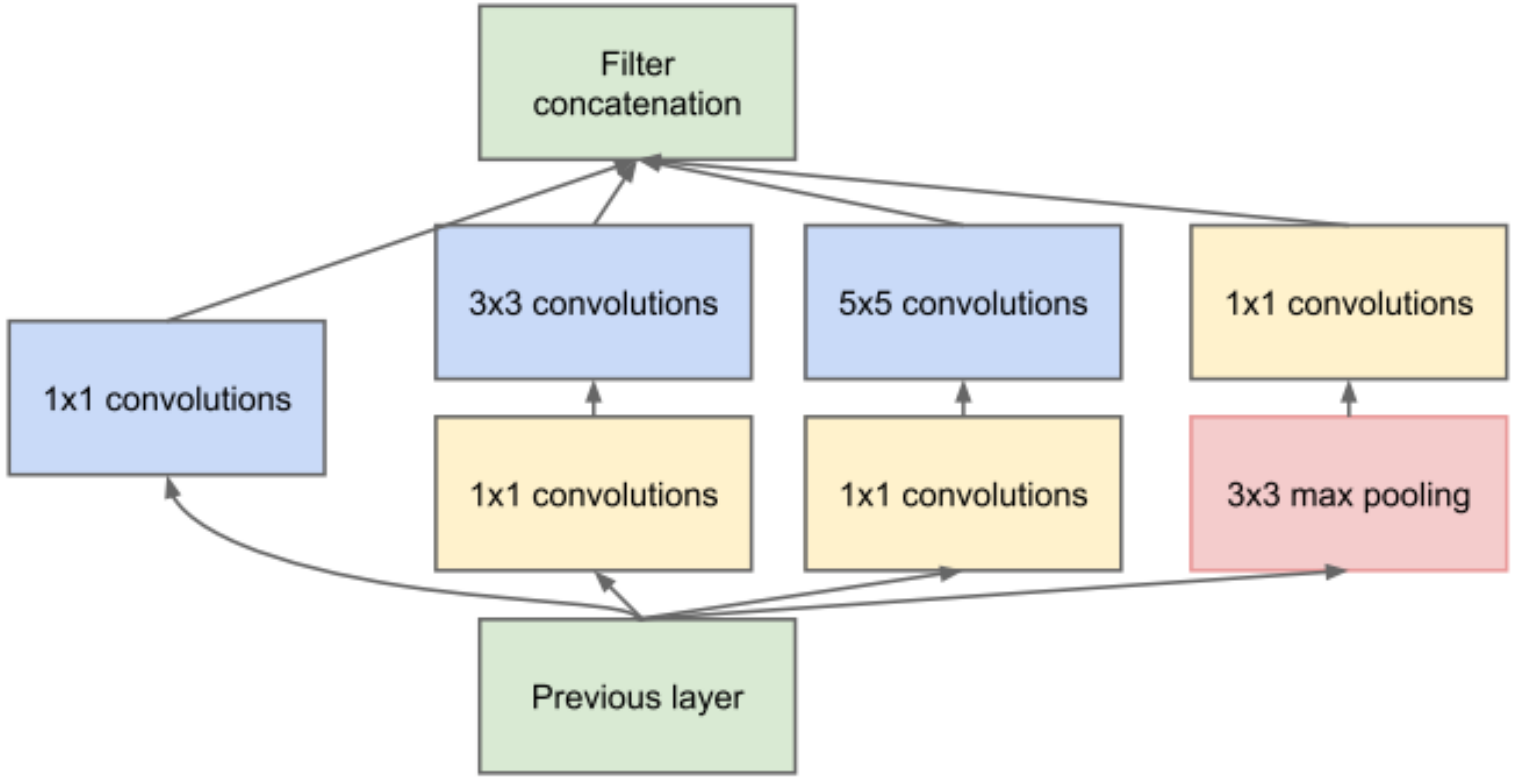
VGGNet

Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Face++ 旷视

GoogleNet



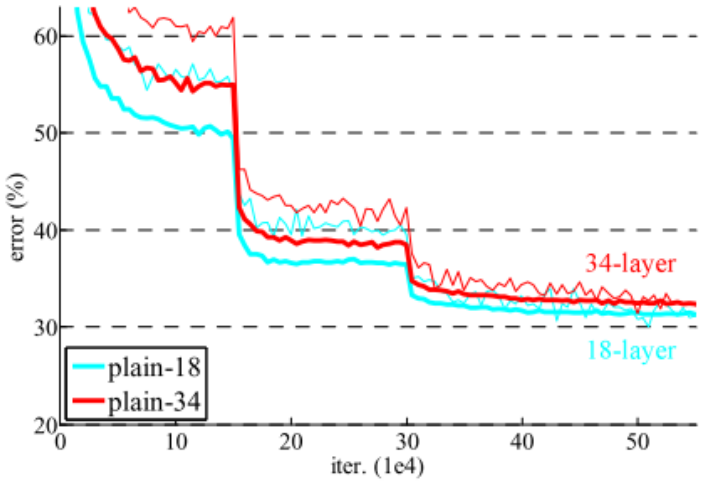
Face++ 旷视

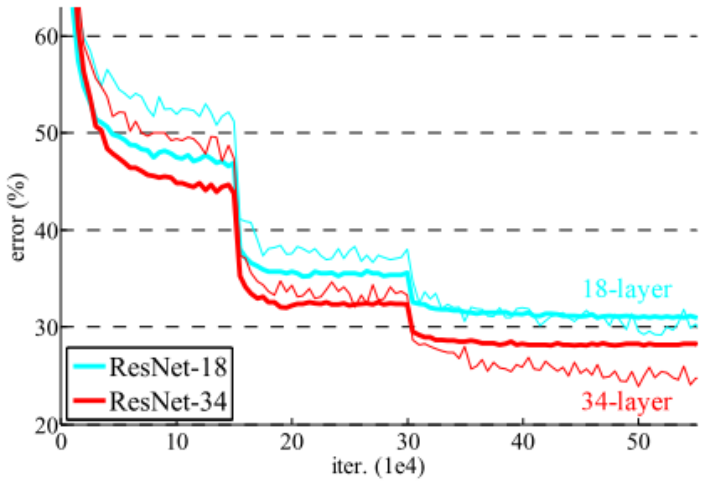
Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions

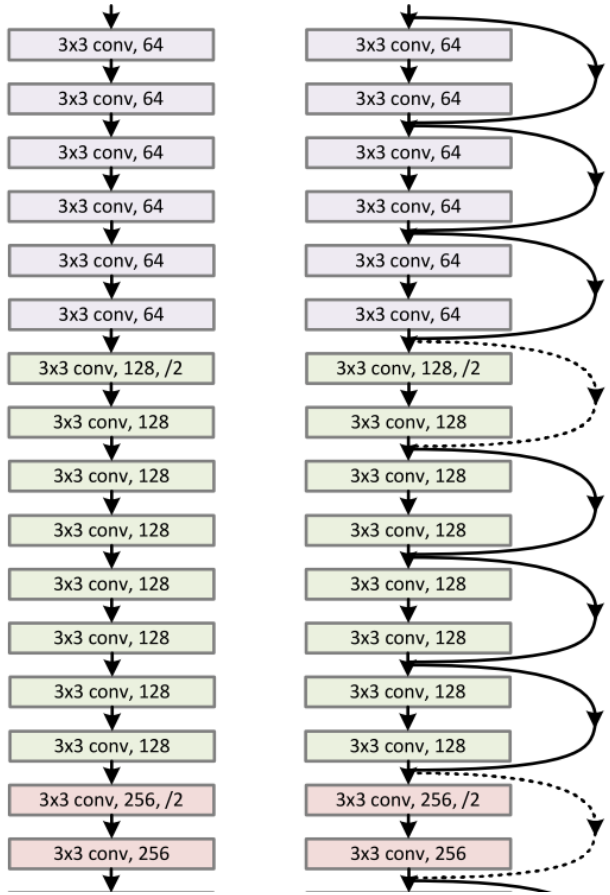
Deep Residual Network

- Easy to optimize
- Enable very deep structures

  -- Over 100 layers for ImageNet model



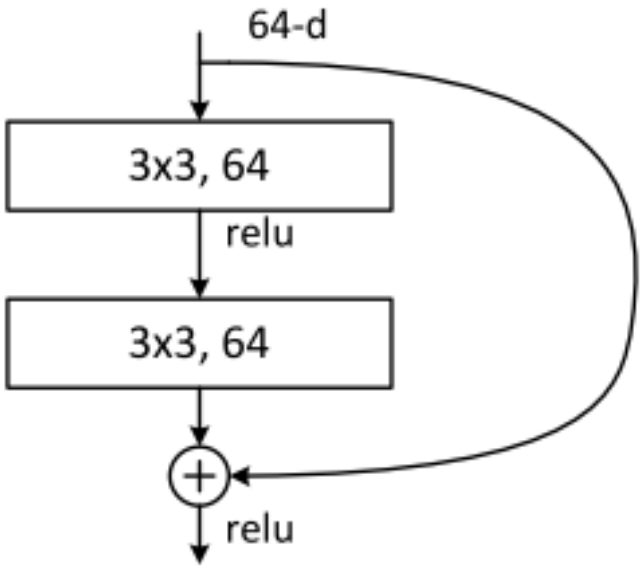


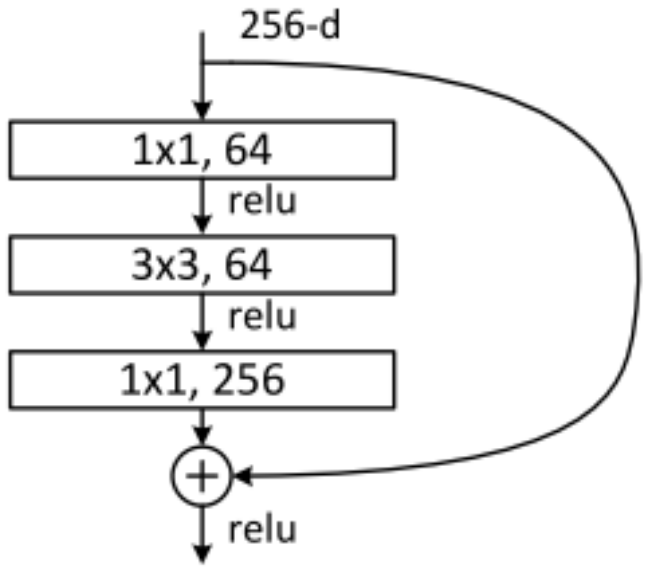


He K, Zhang X, Ren S, et al. Deep residual learning for image recognition

Deep Residual Network (cont'd)

- "Bottleneck" design
- Increasing depth, less complexity

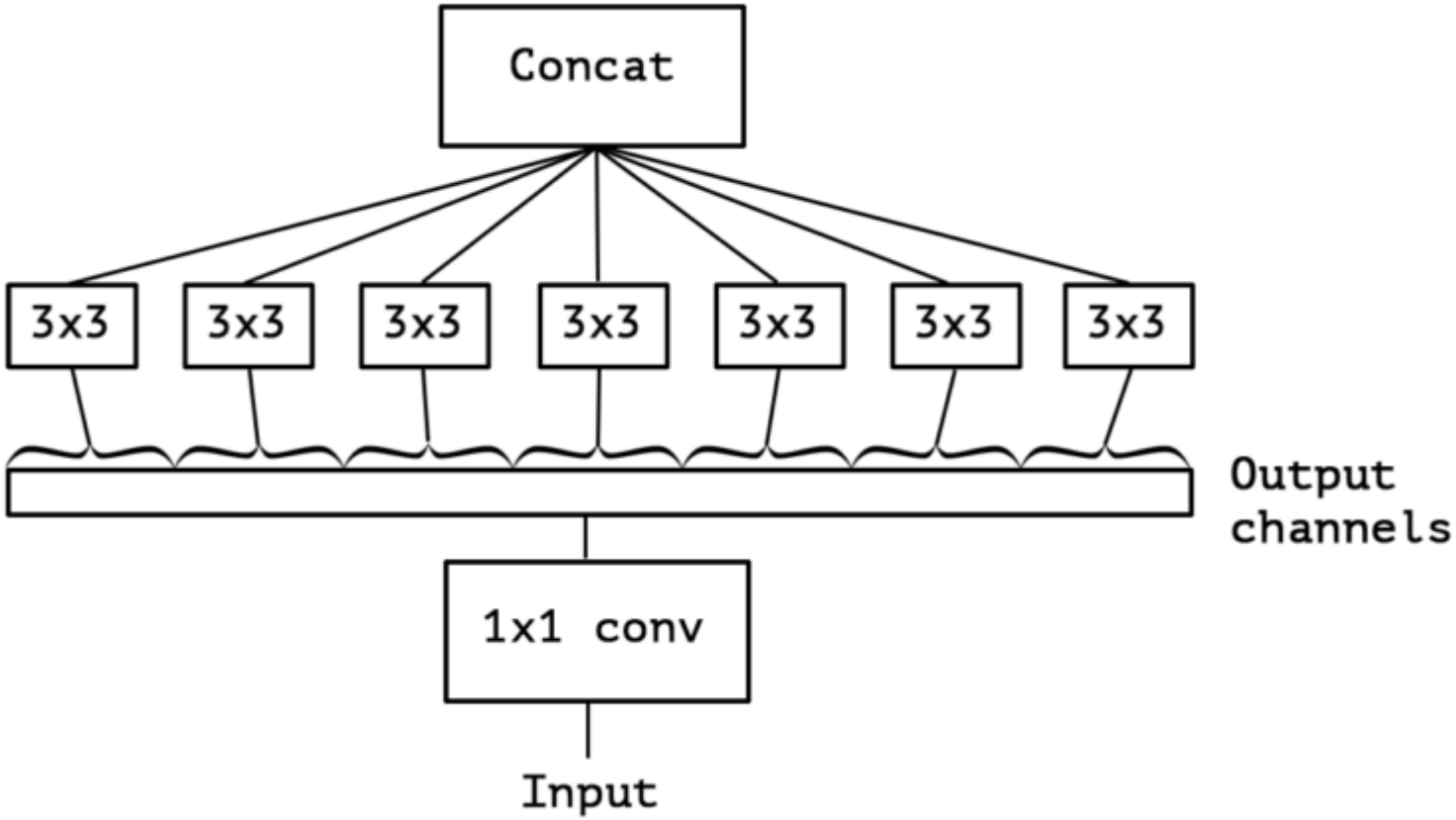




Face++ 旷视

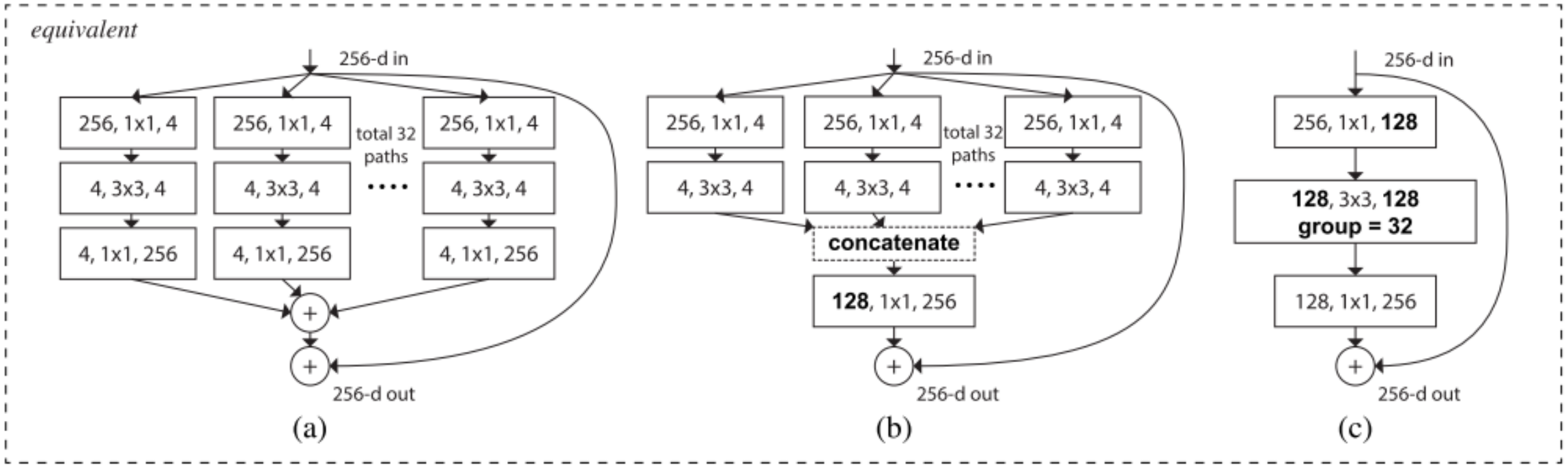
He K, Zhang X, Ren S, et al. Deep residual learning for image recognition

Xception



Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions

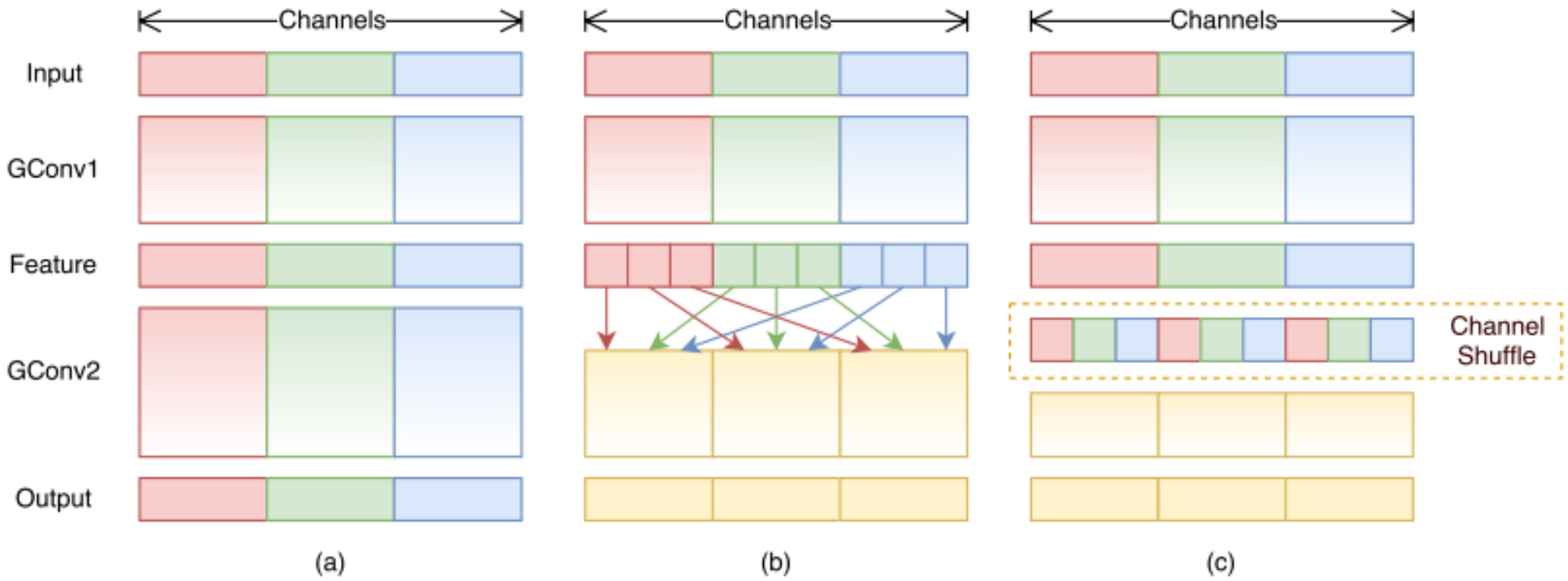
ResNeXt



Face++ 旷视

Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks

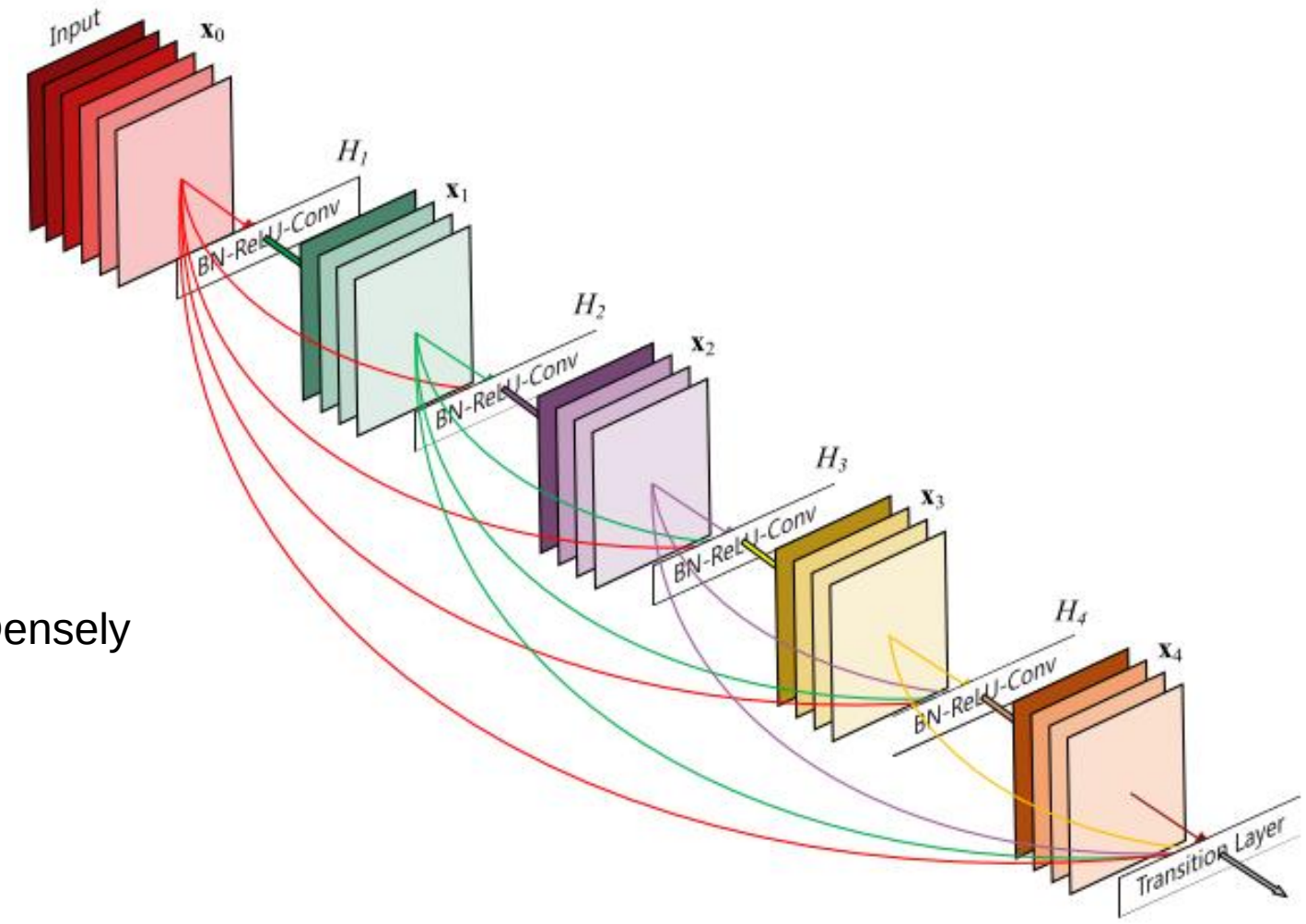
ShuffleNet



Face++ 旷视

Zhang X, Zhou X, Lin M, et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

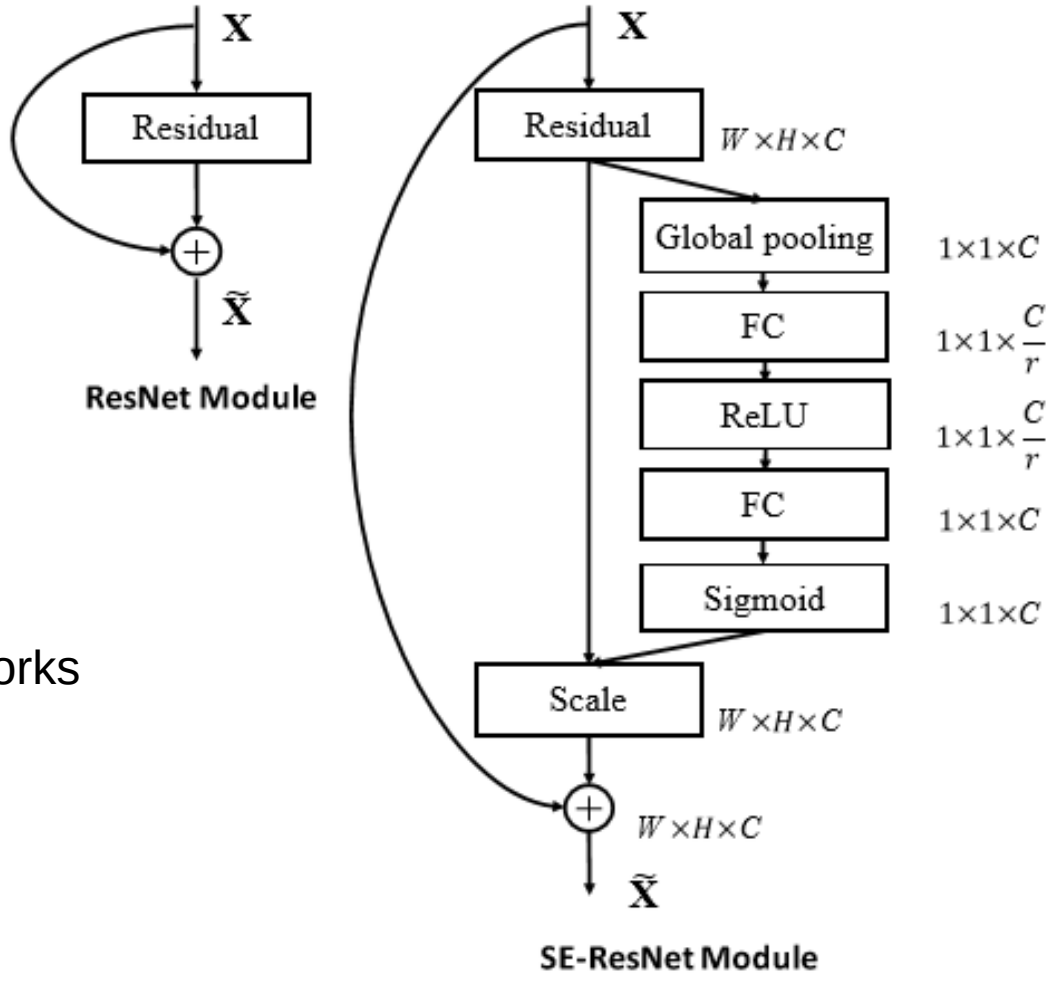
Densely Connected Convolutional Networks



Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks

Face++ 旷视

Squeeze-and-Excitation Networks



Hu J, Shen L, Sun G. Squeeze-and-Excitation Networks

Face++ 旷视

Summary: Ideas of Structure Design

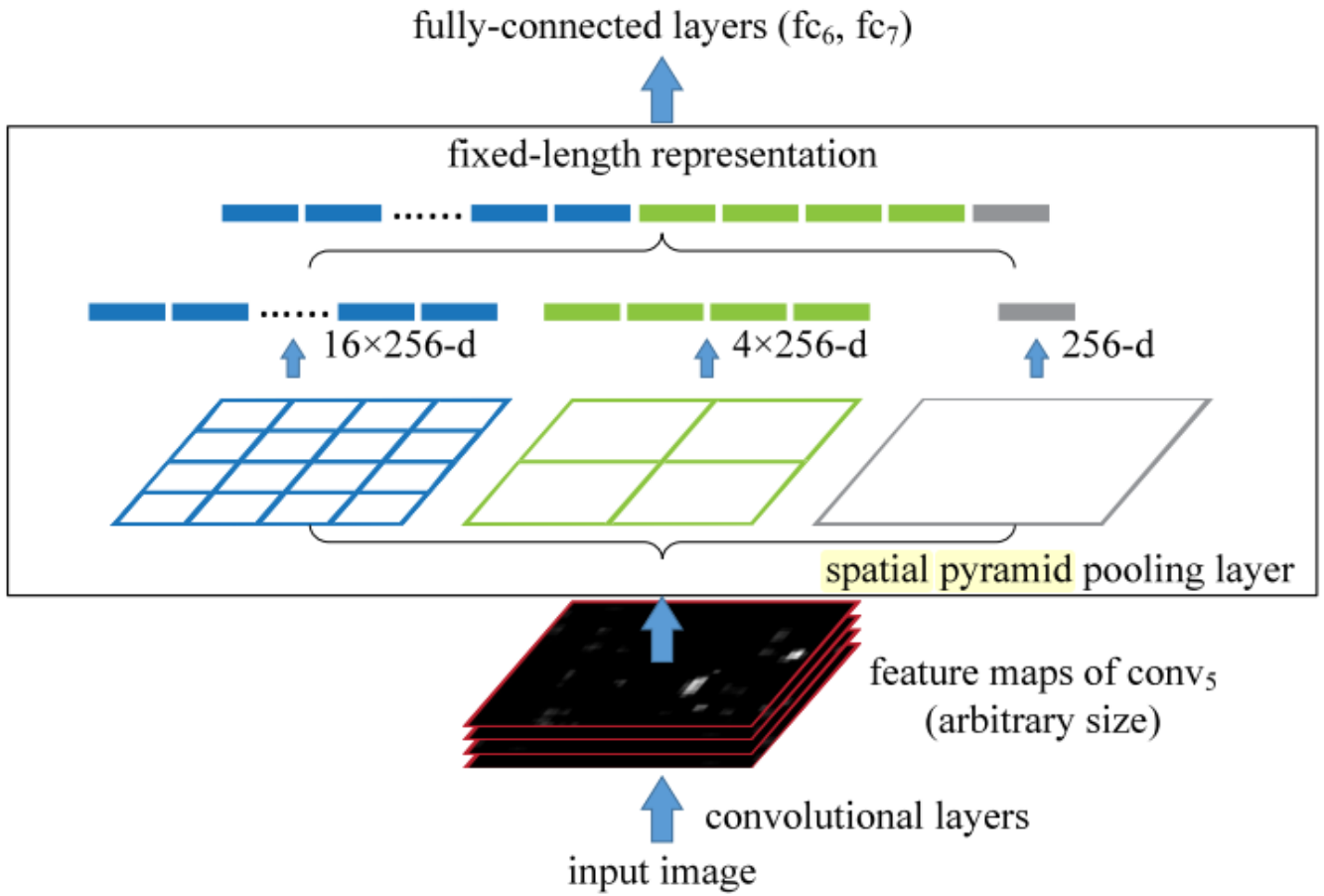
- Deeper and wider
- Ease of optimization
- Multi-path design
- Residual path
- Sparse connection

PART 2: Architecture Design

- Overview
- Structure design
- Layer design
- Architecture for special tasks

Face++ 旷视

Spatial Pyramid Pooling



He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

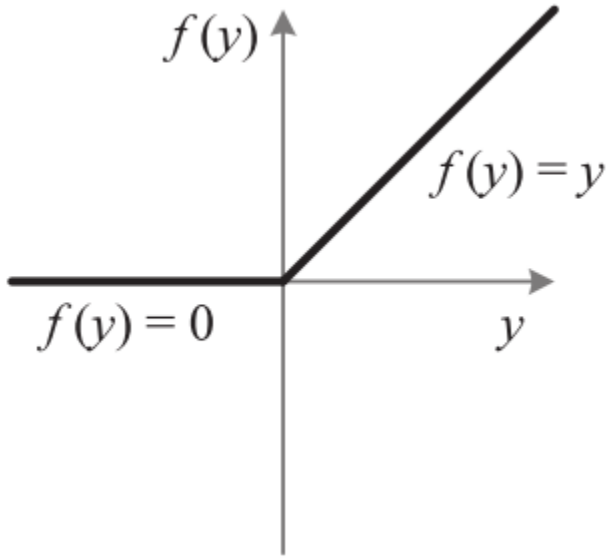
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

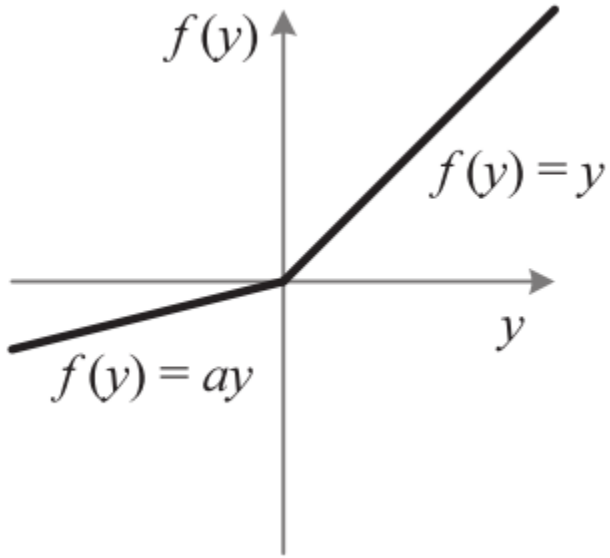
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Face++ 旷视

Batch normalization: Accelerating deep network training by reducing internal covariate shift

Parametric Rectifiers

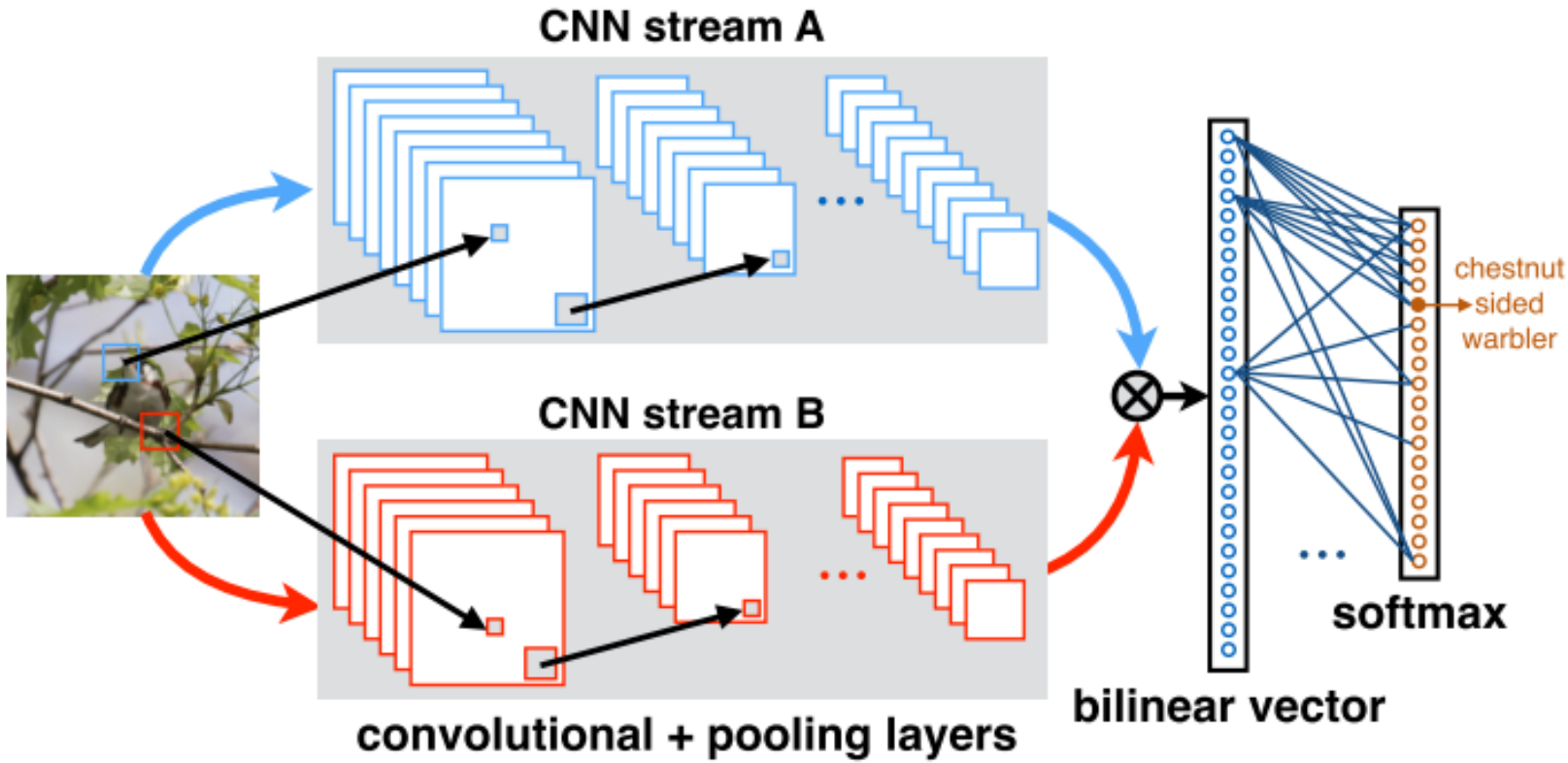




Face++ 旷视

He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification

Bilinear CNNs



Face++ 旷视

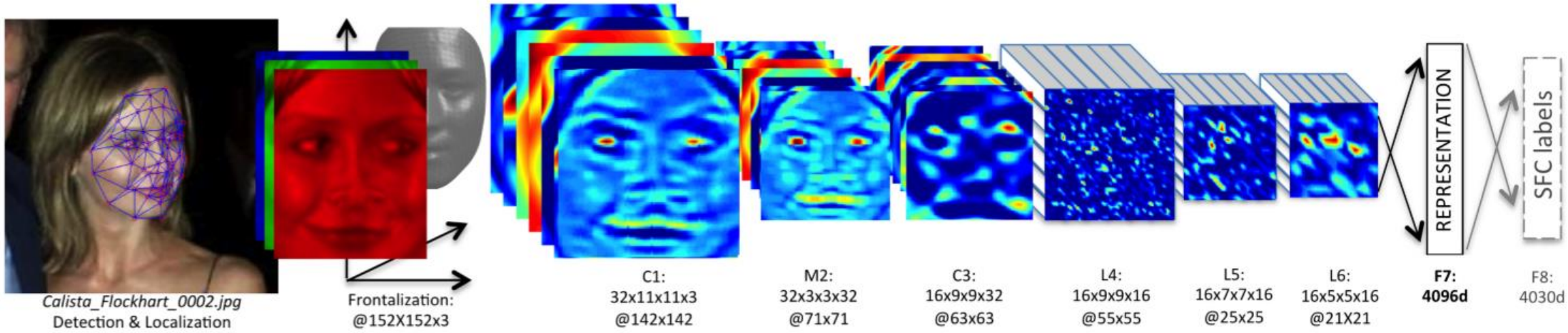
Lin T Y, RoyChowdhury A, Maji S. Bilinear cnn models for fine-grained visual recognition

PART 2: Architecture Design

- Overview
- Structure design
- Layer design
- Architecture for special tasks

Face++ 旷视

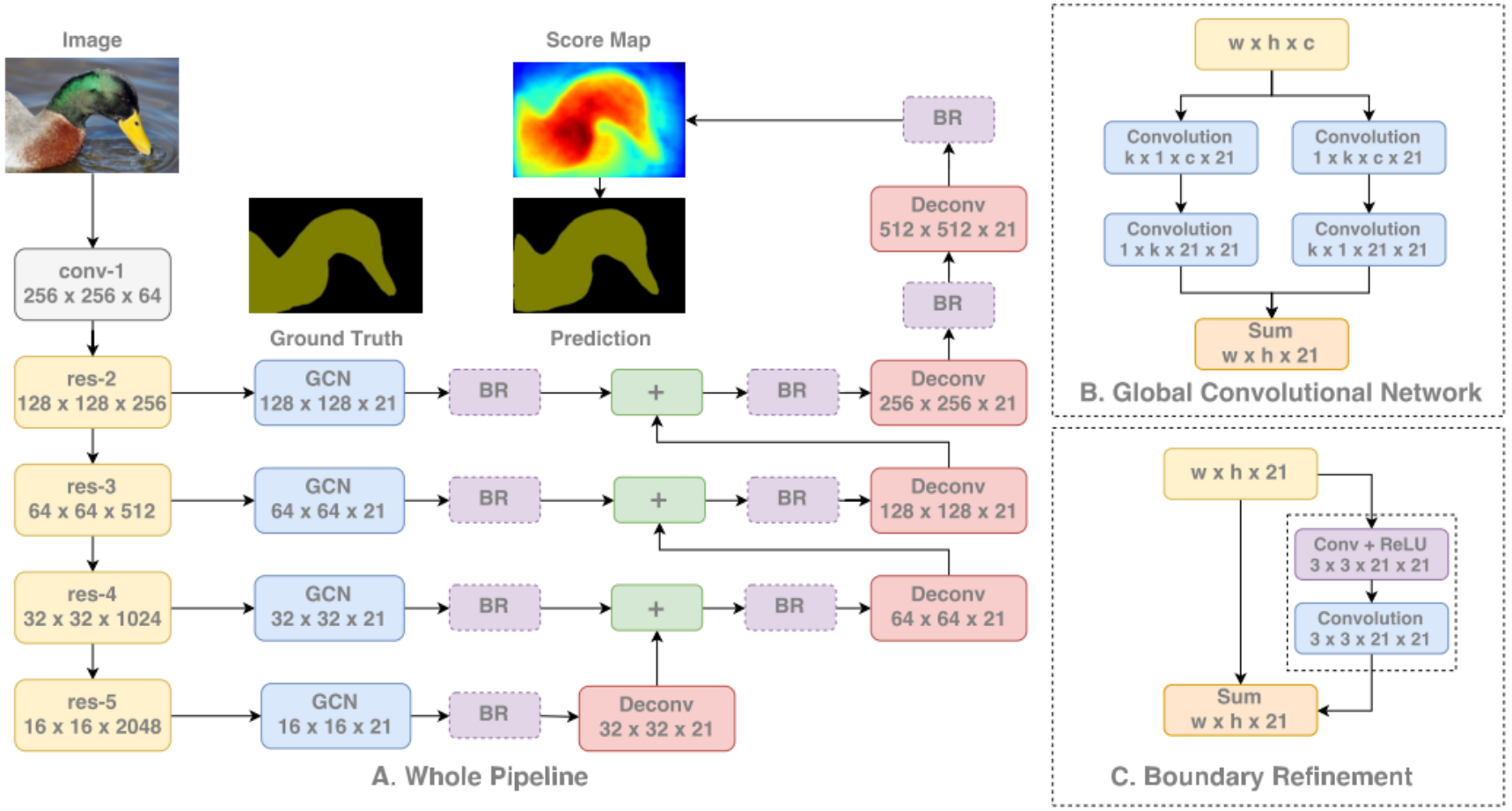
Deepface



Face++ 旷视

Taigman Y, Yang M, Ranzato M A, et al. Deepface: Closing the gap to human-level performance in face verification

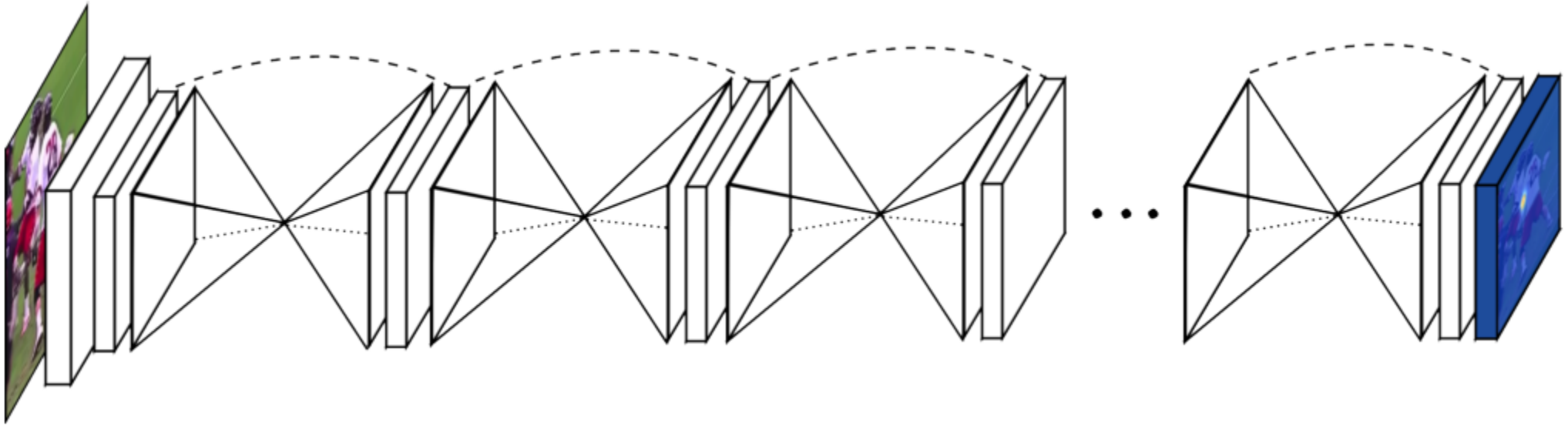
Global Convolutional Networks



Face++ 旷视

Peng C, Zhang X, Yu G, et al. Large Kernel Matters--Improve Semantic Segmentation by Global Convolutional Network

Hourglass Networks



Face++ 旷视

Newell A, Yang K, Deng J. Stacked hourglass networks for human pose estimation

Summary: Trends on Architecture Design

- Effectiveness and efficiency
- Task & data specific
- ML & optimization perspective
- Insight & motivation driven

Face++ 旷视

Thanks

Face++ 旷视