Lecture 6
Neural Networks and CNN

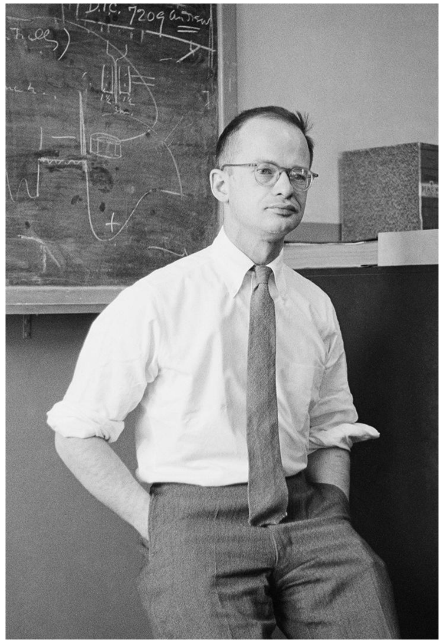
Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2024

Outline

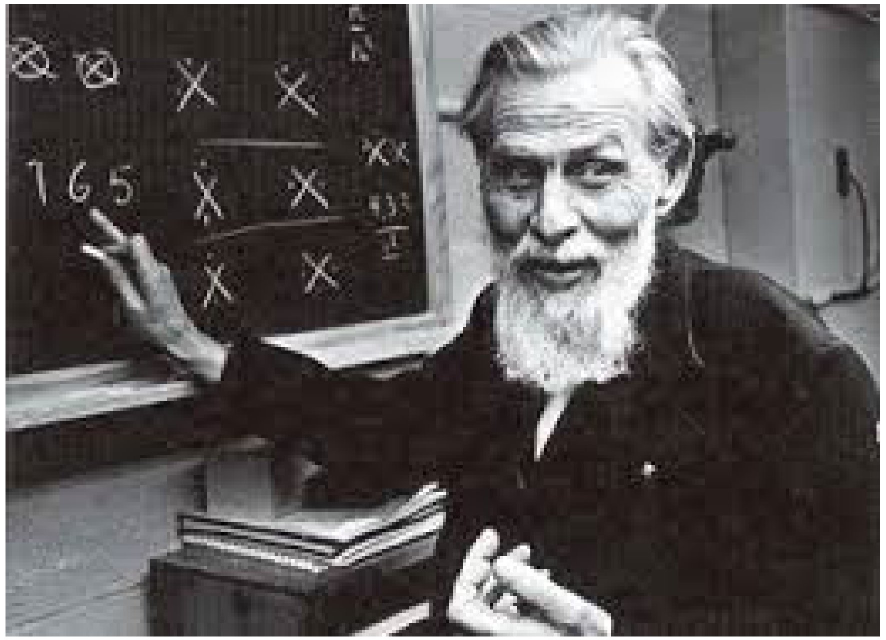
- Neural network
- Convolutional neural network (CNN)
- Modern CNN architectures
- DCNN for object detection

Neural networks

- Neural networks are proposed by Walter Harry Pitts and Warren Sturgis McCulloch in 1943



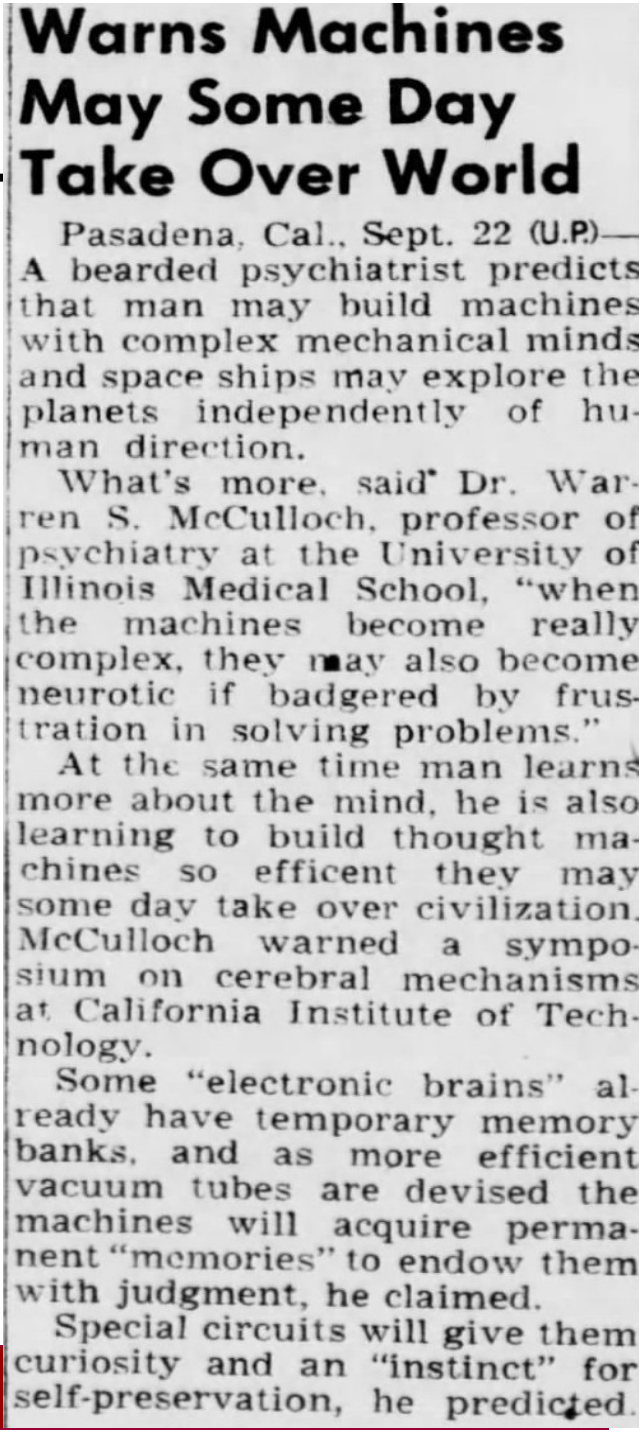
Walter H. Pitts, 23 April 1923 – 14 May 1969, USA



Warren Sturgis McCulloch, Nov. 16, 1898 – Sep. 24, 1969, USA

Neural networks

- In 1948, McCulloch concerned that eventually AI might rule humankind, a topic of much concern in these increasingly automated times. The below article from the September 22, 1948 Brooklyn Daily Eagle records his clarion call about the future



Warns Machines May Some Day Take Over World

Pasadena, Cal., Sept. 22 (U.P.)—A bearded psychiatrist predicts that man may build machines with complex mechanical minds and space ships may explore the planets independently of human direction.

What's more, said Dr. Warren S. McCulloch, professor of psychiatry at the University of Illinois Medical School, "when the machines become really complex, they may also become neurotic if badgered by frustration in solving problems."

At the same time man learns more about the mind, he is also learning to build thought machines so efficent they may some day take over civilization, McCulloch warned a symposium on cerebral mechanisms at California Institute of Technology.

Some "electronic brains" already have temporary memory banks, and as more efficient vacuum tubes are devised the machines will acquire permanent "memories" to endow them with judgment, he claimed.

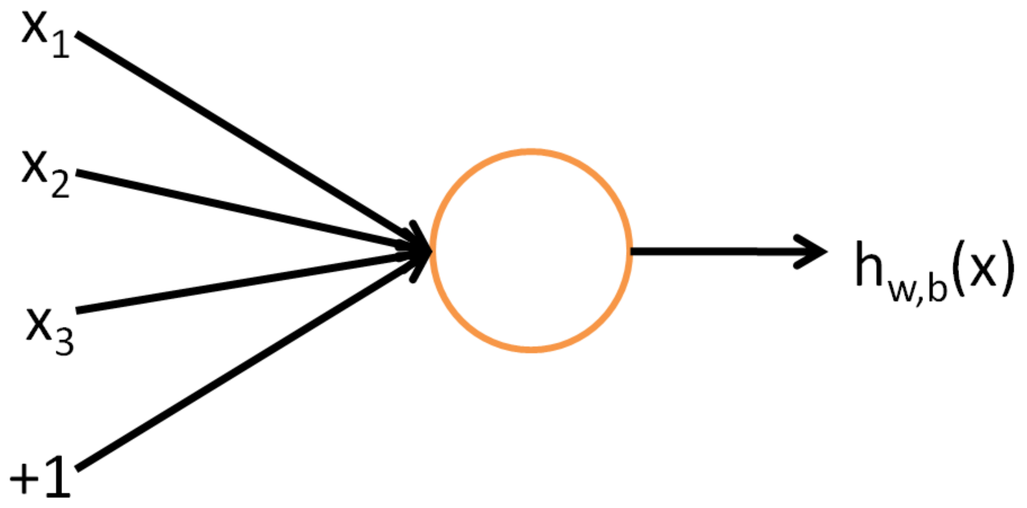
Special circuits will give them curiosity and an "instinct" for self-preservation, he predicted.

Neural networks

- It is one way to solve a supervised learning problem given labeled training examples $\{\boldsymbol{x}_i, y_i\}(i=1,...,m)$
- Neural networks give a way of defining a complex, non-linear form of hypothesis $h_{W,b}(\boldsymbol{x})$, where W and b are the parameters we need to learn from training samples

Neural networks

- A single neuron
  - x_1, x_2, and x_3 are the inputs, +1 is the intercept term, $h_{W,b}(\boldsymbol{x})$ is the output of this neuron



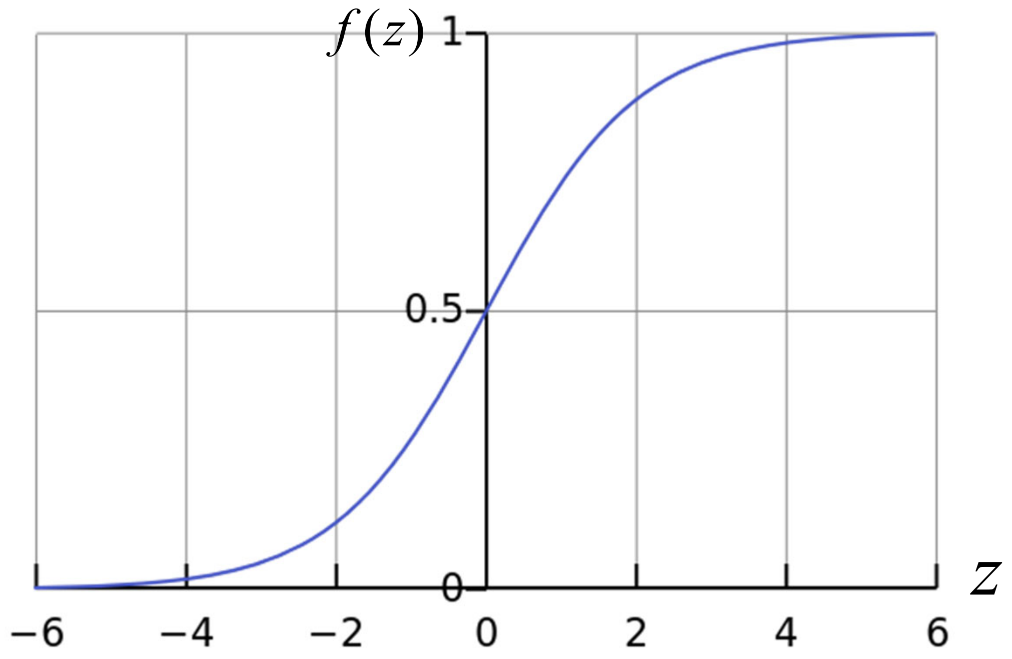
$$h_{W,b}(x) = f\left(W^T \boldsymbol{x}\right) = f\left(\sum_{i=1}^{3} W_i x_i + b\right)$$

where $f(\cdot)$ is the activation function

Neural networks

- Commonly used activation functions
  - Sigmoid function

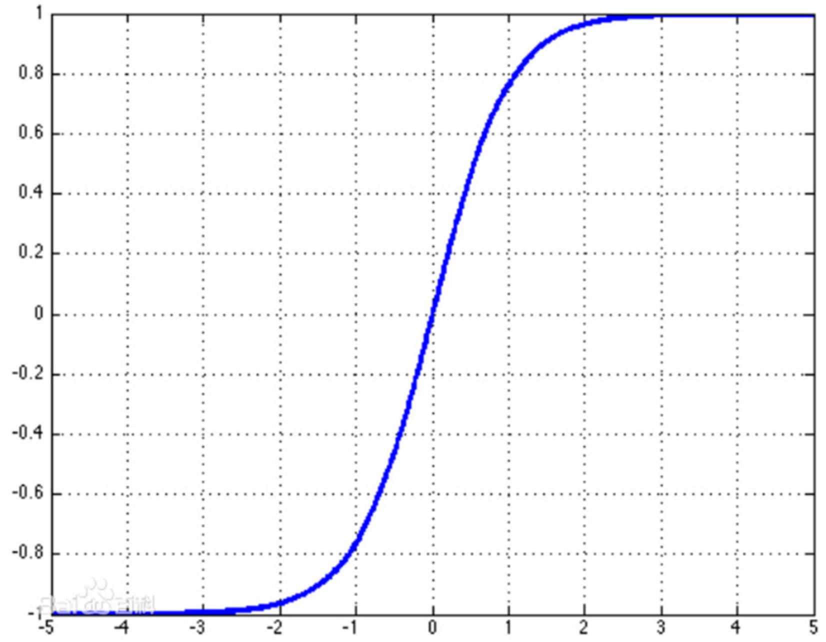
$$f(z) = \frac{1}{1+\exp(-z)}$$



Neural networks

- Commonly used activation functions
  - Tanh function

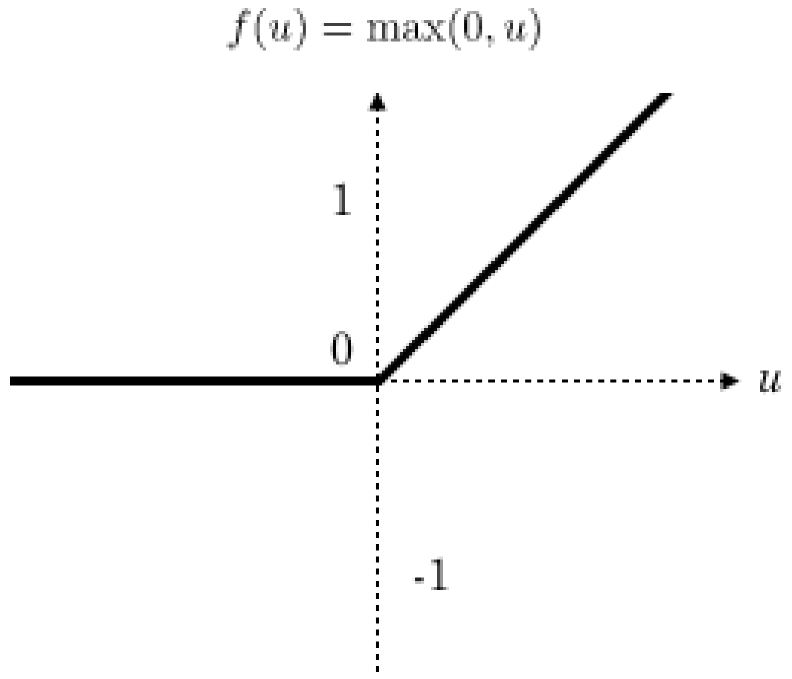
$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



Neural networks

- Commonly used activation functions
  - Rectified linear unit (ReLU)

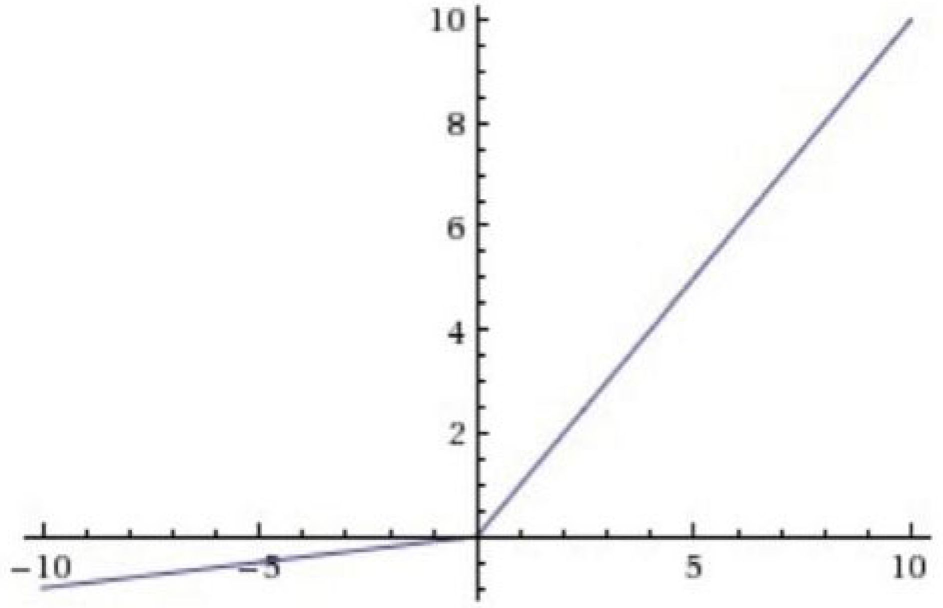
$$f(z) = \max(0, z)$$



Neural networks

- Commonly used activation functions
  - Leaky Rectified linear unit (ReLU)

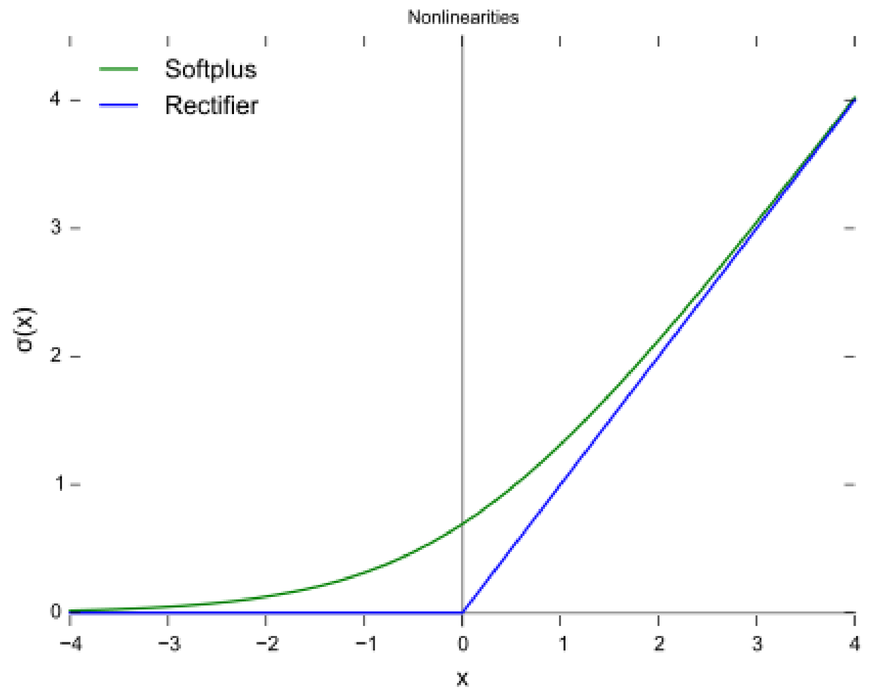
$$f(z)=\begin{cases} z, if\ \ z>0 \\ 0.01z, otherwise \end{cases}$$



Neural networks

- Commonly used activation functions
  - Softplus (can be regarded as a smooth approximation to ReLU)

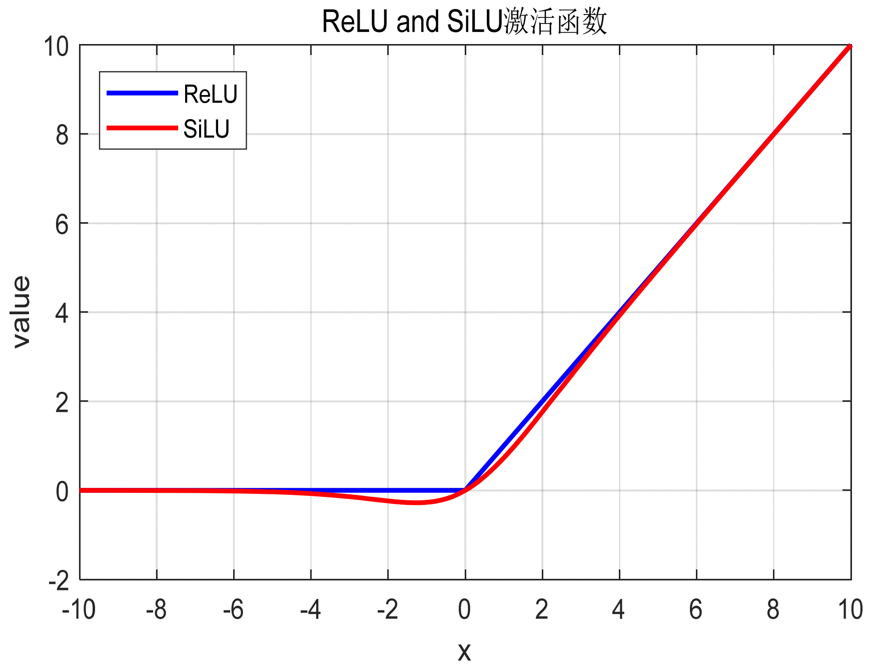
$$f(z) = \ln\left(1 + e^{z}\right)$$



Neural networks

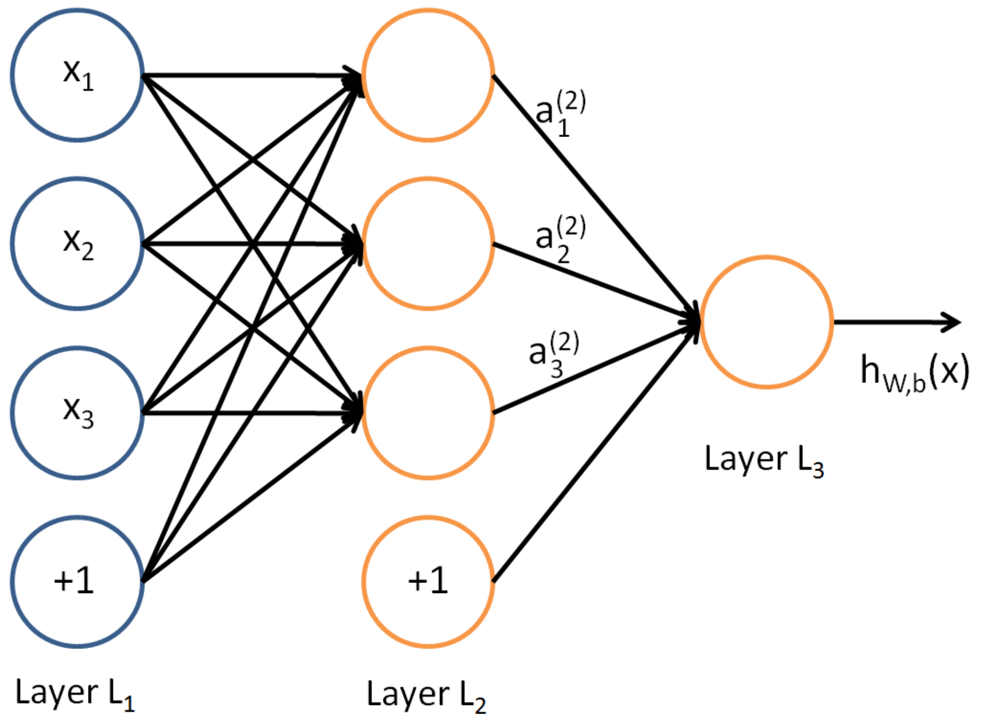
- Commonly used activation functions
  - SiLU, Sigmoid linear unit

$$\mathrm{SiLU}(x) = x \cdot \sigma(x) = x \cdot \frac{1}{1+e^{-x}}$$



Neural networks

- A neural network is composed by hooking together many simple neurons
- The output of a neuron can be the input of another
- Example, a three layers neural network,



Neural networks

- Terminologies about the neural network
  - The circle labeled +1 are called **bias units**
  - The leftmost layer is called the **input layer**
  - The rightmost layer is the **output layer**
  - The middle layer of nodes is called the **hidden layer**
    - » In our example, there are 3 input units, 3 hidden units, and 1 output unit
  - We denote the activation (output value) of unit i in lay l by $a_i^{(l)}$

Neural networks

$$a_1^{(2)} = f\left(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}\right)$$

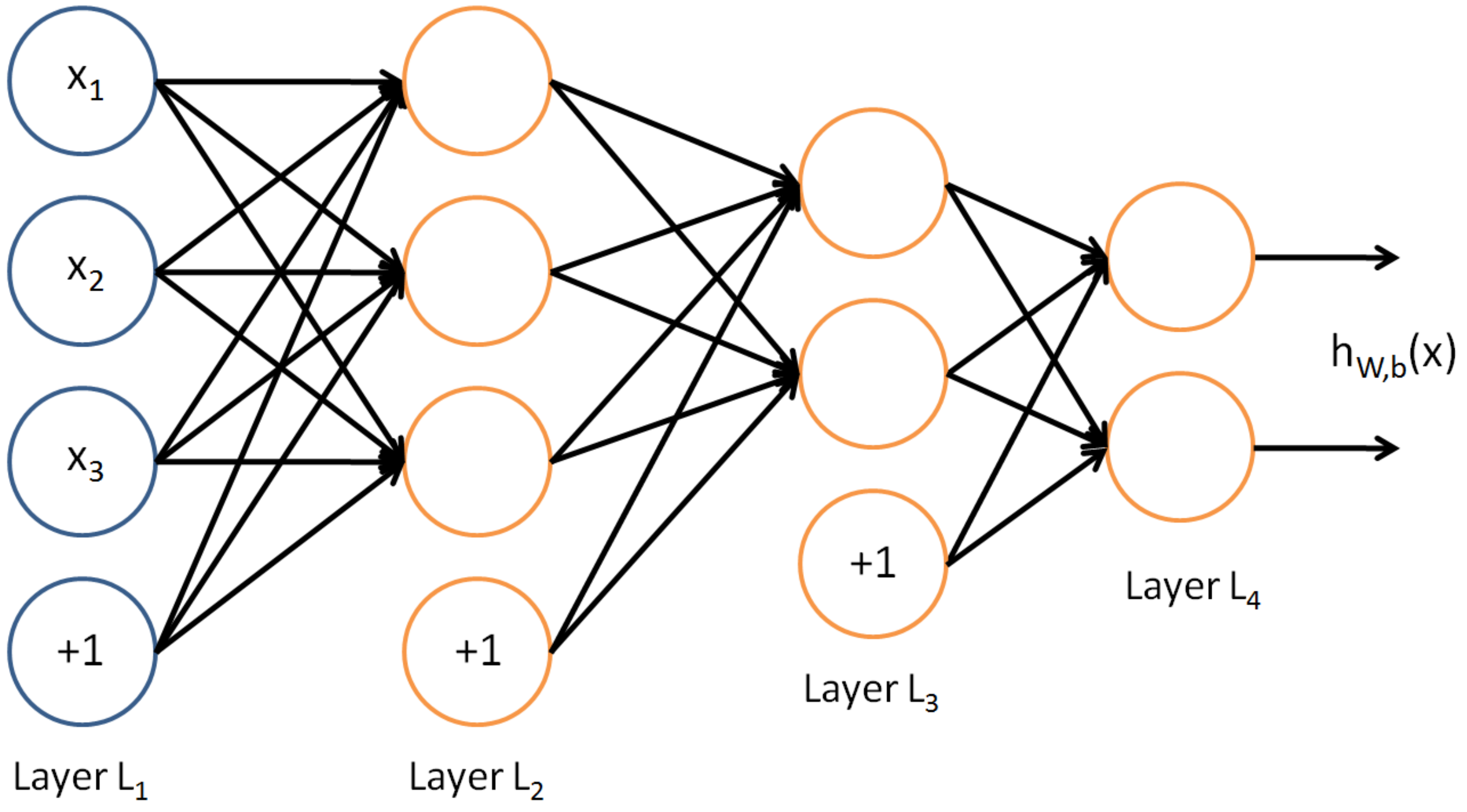
$$a_2^{(2)} = f\left(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}\right)$$

$$a_3^{(2)} = f\left(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}\right)$$

$$h_{W,b}(\boldsymbol{x}) = a_1^{(3)} = f\left(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{1}a_3^{(2)} + b_1^{(2)}\right)$$

Neural networks

- Neural networks can have multiple outputs
- Usually, we can add a softmax layer as the output layer to perform multiclass classification



Neural networks

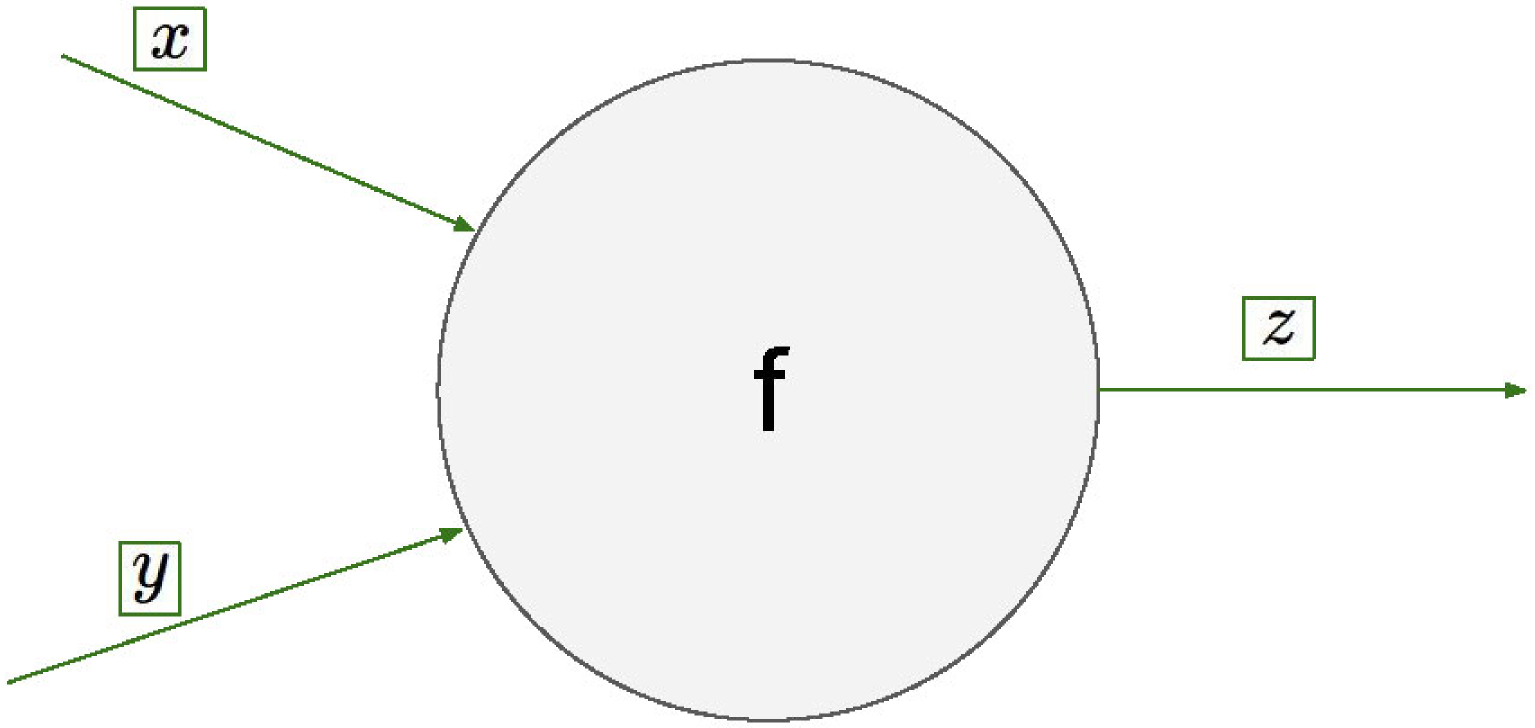
- At the testing stage, given a test input $\boldsymbol{x}$, it is straightforward to evaluate its output
- At the training stage, given a set of training samples, we need to train W and b
  - The key problem is how to compute the gradient
  - Backpropagation algorithm

Neural networks

- Backpropagation
  - A common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent
  - Its purpose is to compute the partial derivative of the loss to each parameter (weights)
  - neural nets will be very large: impractical to write down gradient formula by hand for all parameters
  - recursive application of the chain rule along a computational graph to compute the gradients of all parameters

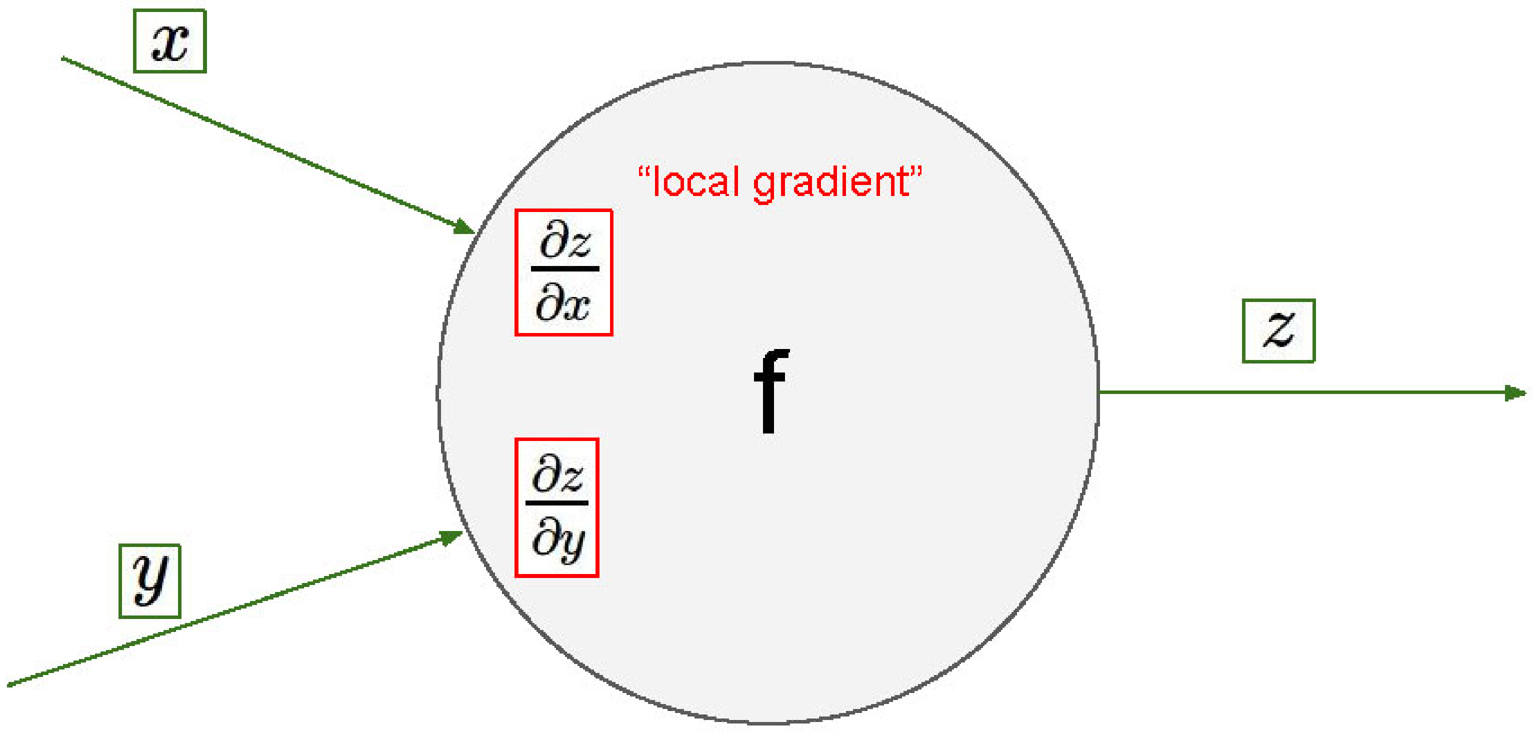
Neural networks

- Backpropagation



Neural networks

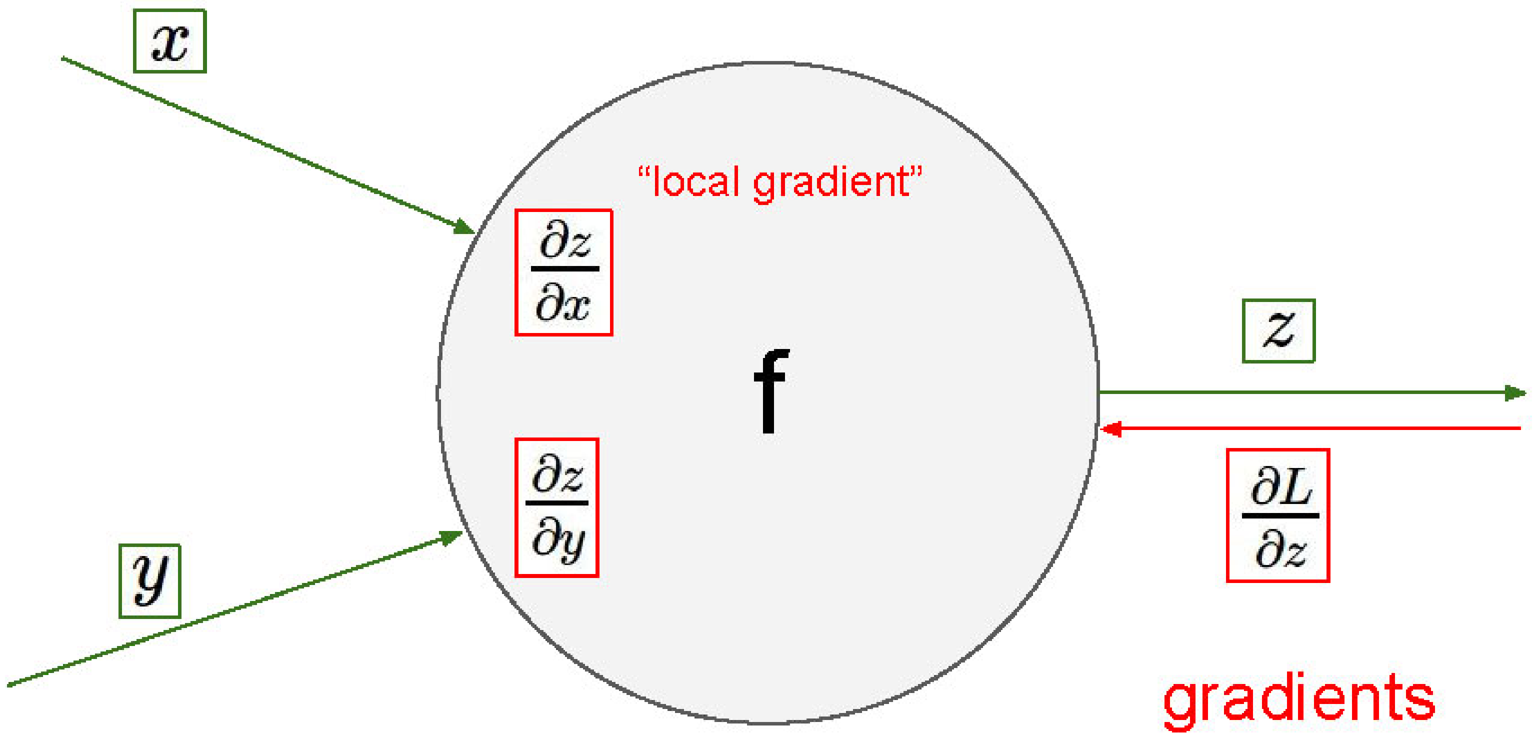
- Backpropagation



x
y
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z

Neural networks

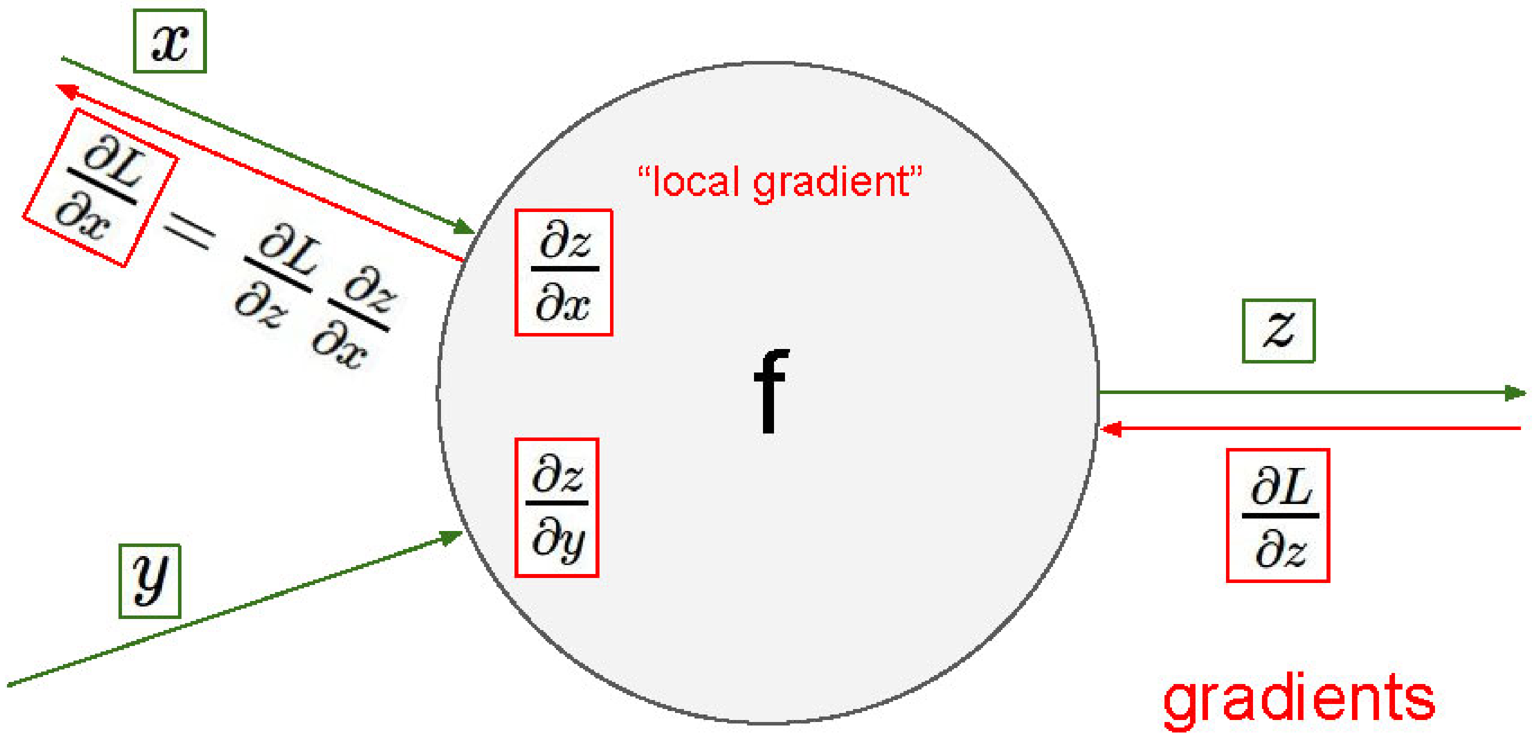
- Backpropagation



x
y
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
gradients

Neural networks

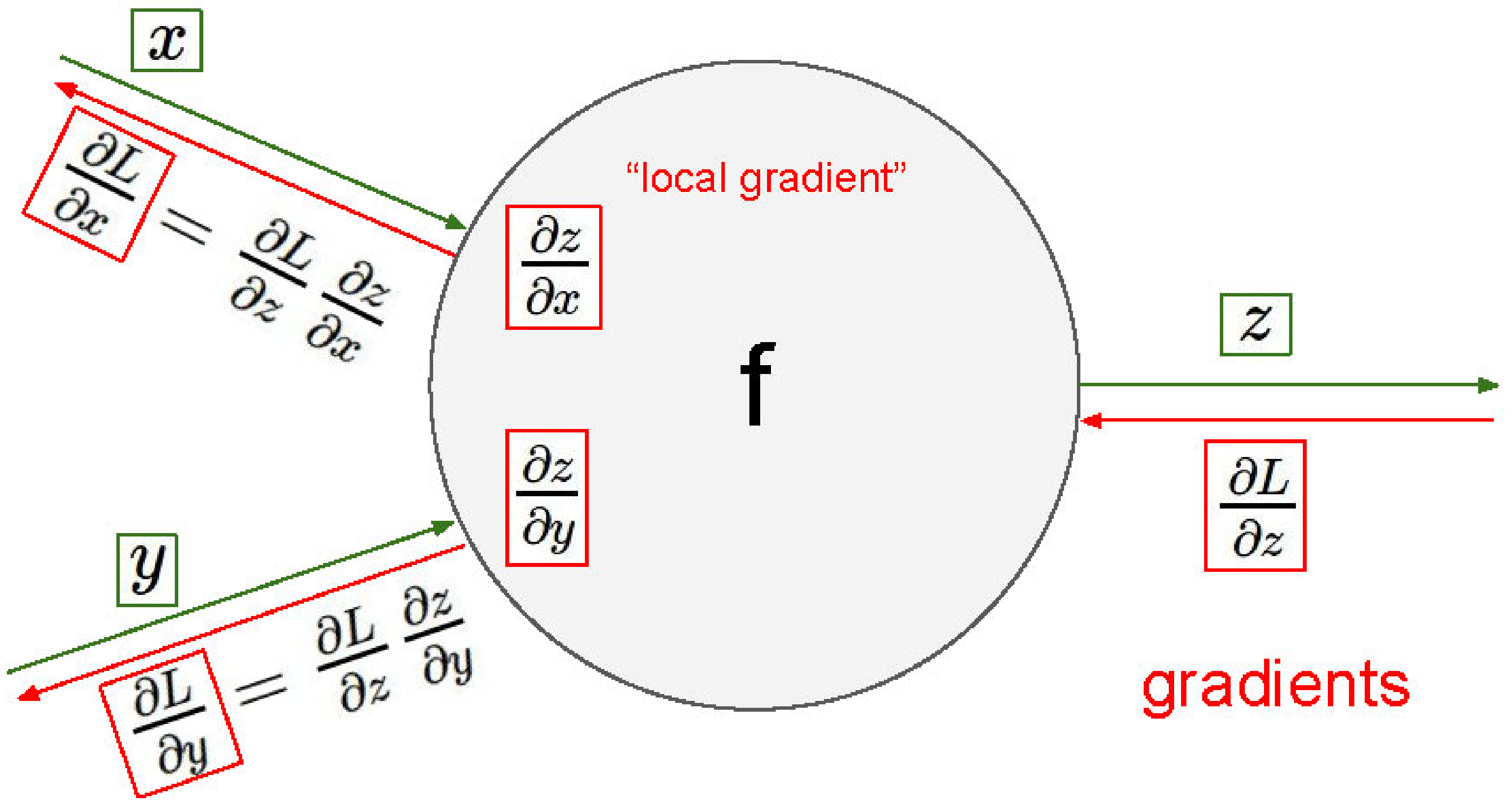
- Backpropagation



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
y
gradients

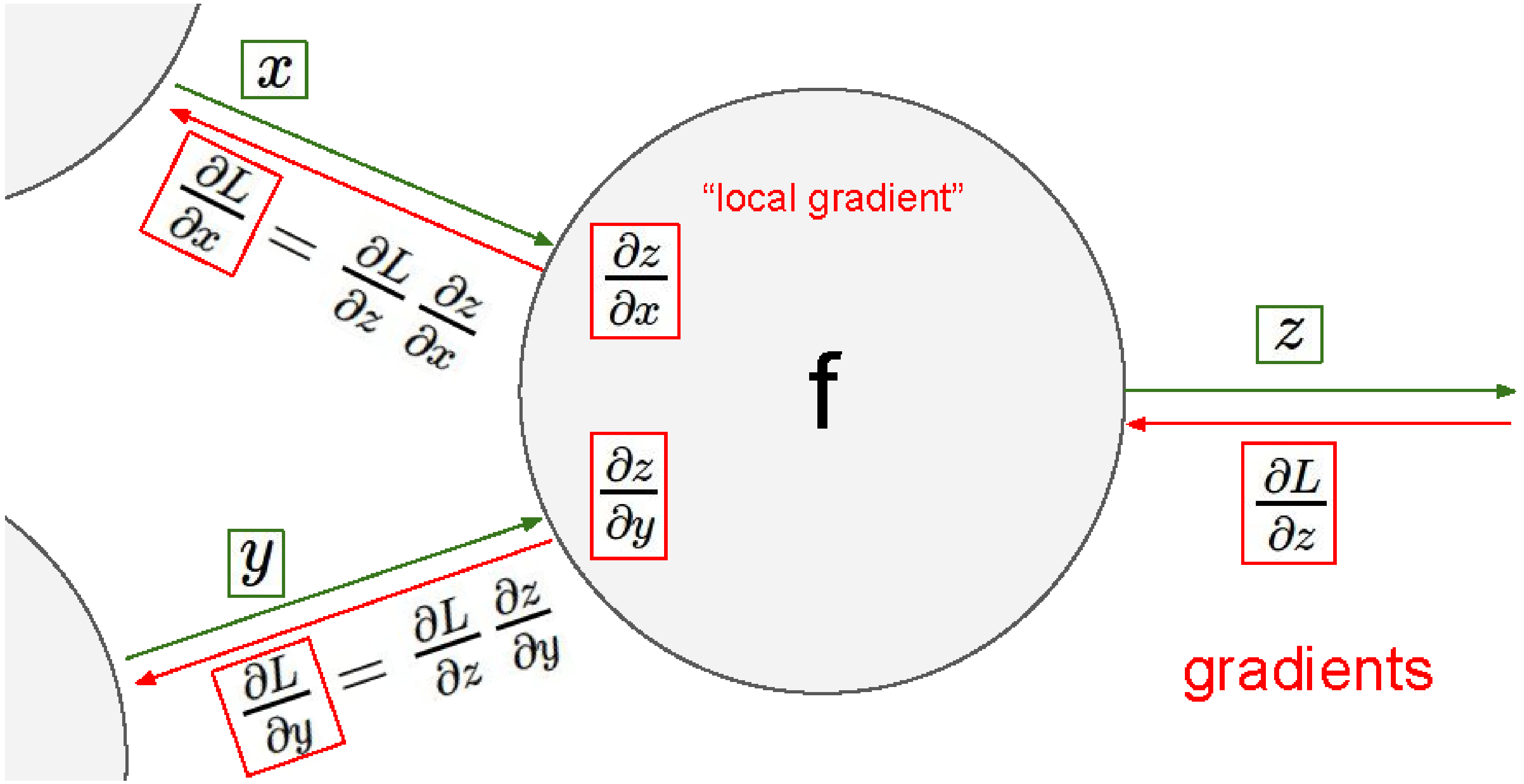
Neural networks

- Backpropagation


x
$\boxed{\frac{\partial L}{\partial x}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\boxed{\frac{\partial L}{\partial y}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Neural networks

- Backpropagation



x
y
z
f
"local gradient"
gradients
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$

Some techniques for better training neural networks

Neural network is trained by iteratively updating the network parameters $\boldsymbol{\theta}$ using the following basic rule,

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \alpha \mathbf{g}_n\left(\boldsymbol{\theta}_{n-1}\right)$$

where $\mathbf{g}_n\left(\boldsymbol{\theta}_{n-1}\right) \in \mathbb{R}^{|\boldsymbol{\theta}|}$ is the gradient of the loss function at nth iteration

- Since the problem of neural network training is extremely large-scale, it is not efficient to find the exact optimal α at each iteration step; instead, we need to use some heuristic strategies to appropriately adjust α
- Another issue is the effective estimation of gradients. The training of neural networks generally adopts SGD or mini-batch SGD, which leads to a small number of samples selected each time, resulting in the estimated gradient of each iteration not being consistent with the optimal gradient on the entire training set and having a certain degree of randomness. We can make full use of the average value of the gradients used in the last few iterations to correct the random gradient calculated in the current iteration. This is beneficial for accelerating the convergence of training

Some techniques for better training neural networks

- Heuristic strategies to appropriately adjust α
  - Learning rate decay

Inverse time decay, $\alpha_t = \alpha_0 \frac{1}{1+\beta \times t}, \beta > 0$

Exponential decay, $\alpha_t = \alpha_0 \beta^t, 0 < \beta < 1$

Natural Exponential decay, $\alpha_t = \alpha_0 \exp(-\beta \times t), \beta > 0$

where α_0 is the initial learning rate, t is the iteration index, and β is the decay rate

Some techniques for better training neural networks

- Heuristic strategies to appropriately adjust α
  - Learning rate decay
  - Parameter-adaptive learning rate adjusting policies
    - » AdaGrad[1]

At n-th iteration, first compute the cumulative square of each parameter's partial derivatives,

$$\mathbf{G}_n = \sum_{\tau=1}^{n} \mathbf{g}_\tau(\boldsymbol{\theta}_{\tau-1}) \odot \mathbf{g}_\tau(\boldsymbol{\theta}_{\tau-1})$$

where $\mathbf{g}_\tau \in \mathbb{R}^{|\boldsymbol{\theta}|}$ is the gradient at τ-th iteration, $\odot$ is the elementwise product

Then, $\boldsymbol{\theta}$ is updated as,

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \frac{\alpha_0}{\sqrt{\mathbf{G}_n + \varepsilon}} \odot \mathbf{g}_n(\boldsymbol{\theta}_{n-1})$$

Note: all the operations here are elementwise

[1] J. Duchi *et al.*, Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7), 2011

Some techniques for better training neural networks

- Heuristic strategies to appropriately adjust α
  - Learning rate decay
  - Parameter-adaptive learning rate adjusting policies
    - » AdaGrad[1]

Properties:

① At this iteration, for a parameter, if its cumulative square values of partial derivates is large, the associated learning rate will be small, otherwise large

② The learning rate for each parameter will continuously decrease as the number of iterations increases; sometimes it is difficult for it to find the optimal solution since the learning rates have already been too small

[1] J. Duchi *et al.*, Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7), 2011

Some techniques for better training neural networks

- Heuristic strategies to appropriately adjust α
  - Learning rate decay
  - Parameter-adaptive learning rate adjusting policies
    - » AdaGrad[1]
    - » RMSprop[2] (Root Mean Squared Propagation)

At n-th iteration, first compute the moving average square of each parameter's partial derivatives,

$$\mathbf{G}_n = \beta \mathbf{G}_{n-1} + (1-\beta)\mathbf{g}_n(\boldsymbol{\theta}_{n-1}) \odot \mathbf{g}_n(\boldsymbol{\theta}_{n-1}) = (1-\beta)\sum_{\tau=1}^{n} \beta^{n-\tau} \mathbf{g}_\tau(\boldsymbol{\theta}_{\tau-1}) \odot \mathbf{g}_\tau(\boldsymbol{\theta}_{\tau-1})$$

where β is usually set as 0.9

Then, $\boldsymbol{\theta}$ is updated as,

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \frac{\alpha_0}{\sqrt{\mathbf{G}_n + \varepsilon}} \odot \mathbf{g}_n(\boldsymbol{\theta}_{n-1})$$

[2] T. Tieleman and G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning 4 (2), 26-31, 2012

Some techniques for better training neural networks

- Heuristic strategies to appropriately adjust α
  - Learning rate decay
  - Parameter-adaptive learning rate adjusting policies
    - » AdaGrad[1]
    - » RMSprop[2] (Root Mean Squared Propagation)

Properties:

① At this iteration, for a parameter, if its associate element in $\mathbf{G}_n$ is large, the associated learning rate will be small, otherwise large

② Different from AdaGrad, the learning rate for each parameter does not monotonically decrease as the number of iterations increases

[2] T. Tieleman and G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning 4 (2), 26-31, 2012

Some techniques for better training neural networks

- Correction for gradient estimation
  - Momentum method

$$\Delta\boldsymbol{\theta}_n = \rho\Delta\boldsymbol{\theta}_{n-1} - \alpha\mathbf{g}_n\left(\boldsymbol{\theta}_{n-1}\right) = -\alpha\sum_{\tau=1}^{n}\rho^{n-\tau}\mathbf{g}_\tau\left(\boldsymbol{\theta}_{\tau-1}\right)$$

where ρ is usually set as 0.9

($\boldsymbol{\theta}_n$ is updated as $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \Delta\boldsymbol{\theta}_n$)

Properties:

① The gradient used in this iteration is the weighted average of the past gradients

Some techniques for better training neural networks

- Correction for gradient estimation
  - Momentum method
  - NAG (Nesterov Accelerated Gradient)[3]

Moment method

The parameter update is completed as,

$$\begin{cases} \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{n-1} + \rho \Delta \boldsymbol{\theta}_{n-1} \\ \boldsymbol{\theta}_n = \hat{\boldsymbol{\theta}} - \alpha \mathbf{g}_n \left(\boldsymbol{\theta}_{n-1} \right) \end{cases}$$

It's more valid if we take gradient at $\hat{\boldsymbol{\theta}}$

$\xrightarrow{\text{improved}}$

NAG

The parameter update is completed as,

$$\begin{cases} \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{n-1} + \rho \Delta \boldsymbol{\theta}_{n-1} \\ \boldsymbol{\theta}_n = \hat{\boldsymbol{\theta}} - \alpha \mathbf{g}_n \left(\hat{\boldsymbol{\theta}} \right) = \hat{\boldsymbol{\theta}} - \alpha \mathbf{g}_n \left(\boldsymbol{\theta}_{n-1} + \rho \Delta \boldsymbol{\theta}_{n-1} \right) \end{cases}$$

$\Big\downarrow$ with NAG

$$\Delta \boldsymbol{\theta}_n = \rho \Delta \boldsymbol{\theta}_{n-1} - \alpha \mathbf{g}_n \left(\boldsymbol{\theta}_{n-1} + \rho \Delta \boldsymbol{\theta}_{n-1} \right)$$

[3] Y. Nesterov, Gradient methods for minimizing composite functions, Mathematical Programming, 140 (1), 125-161, 2013

Some techniques for better training neural networks

- Correction for gradient estimation
  - Momentum method
  - NAG (Nesterov Accelerated Gradient)[3]
  - Adam (Adaptive Moment Estimation Algorithm) [4]

$$\boldsymbol{M}_n = \beta_1 \boldsymbol{M}_{n-1} + (1-\beta_1)\mathbf{g}_n(\boldsymbol{\theta}_{n-1})$$ (similar as the moment method)

$$\mathbf{G}_n = \beta_2 \mathbf{G}_{n-1} + (1-\beta_2)\mathbf{g}_n(\boldsymbol{\theta}_{n-1}) \odot \mathbf{g}_n(\boldsymbol{\theta}_{n-1})$$ (similar as RMSprop)

(β_1 and β_2 are usually set as 0.9 and 0.99, respectively)

When n is small, M_n and G_n are not accurate, so they need to be corrected as,

$$\widehat{\mathbf{M}}_n = \frac{\mathbf{M}_n}{1-\beta_1^n} \qquad \widehat{\mathbf{G}}_n = \frac{\mathbf{G}_n}{1-\beta_2^n}$$

Update is completed as, $$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \frac{\alpha_0}{\sqrt{\widehat{\mathbf{G}}_n + \varepsilon}} \odot \widehat{\mathbf{M}}_n$$

Adam can be deemed as a combination of RMSprop and moment method

[4] D. Kingma and J. Ba, Adam: A method for stochastic optimization, ICLR, 2015

Outline

- Neural network
- Convolutional neural network (CNN)
- Modern CNN architectures
- CNN for object detection

A little history about deep learning

深度学习

2006年，Hinton和他的学生Salakhutdinov在《科学》上发表了一篇文章，开启了深度学习在学术界和工业界的浪潮。

Reducing the Dimensionality of Data with Neural Networks

文章提出了深层网络训练中梯度消失问题的解决方案：**无监督预训练对权值进行初始化+有监督训练微调**

2012年，Hinton课题组为了证明深度学习的潜力，首次参加ImageNet图像识别比赛，其通过构建的CNN网络AlexNet一举夺得冠军，且碾压第二名（SVM方法）的分类性能。也正是由于该比赛，CNN吸引到了众多研究者的注意

A little history about deep learning

Yann LeCun

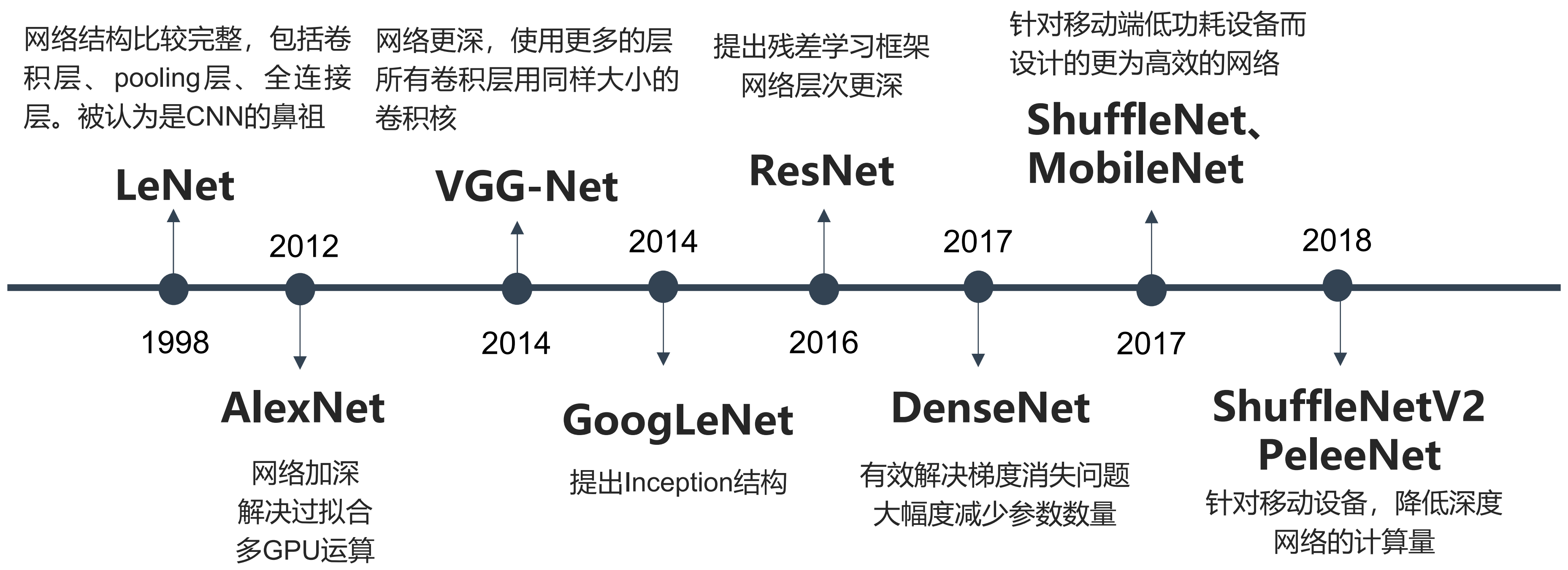
Geoffrey Hinton

Yoshua Bengio

2018 ACM A.M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing

A little history about deep learning

深度学习的发展



网络结构比较完整，包括卷积层、 pooling层、 全连接层。被认为是CNN的鼻祖

LeNet

1998

2012

AlexNet

网络加深
解决过拟合
多GPU运算

网络更深，使用更多的层
所有卷积层用同样大小的卷积核

VGG-Net

2014

2014

GoogLeNet

提出Inception结构

提出残差学习框架
网络层次更深

ResNet

2016

2017

DenseNet

有效解决梯度消失问题
大幅度减少参数数量

针对移动端低功耗设备而设计的更为高效的网络

ShuffleNet、MobileNet

2017

2018

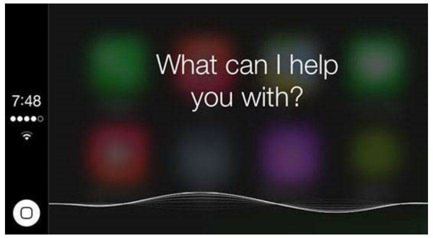
ShuffleNetV2 PeleeNet

针对移动设备，降低深度网络的计算量

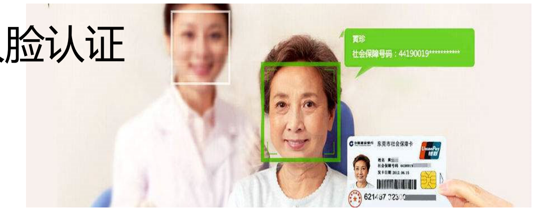
A little history about deep learning

深度学习的应用领域

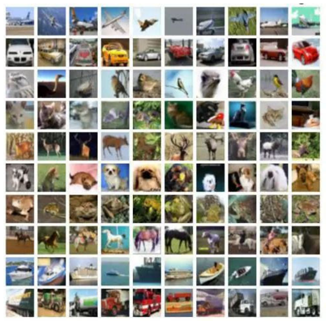
语音识别

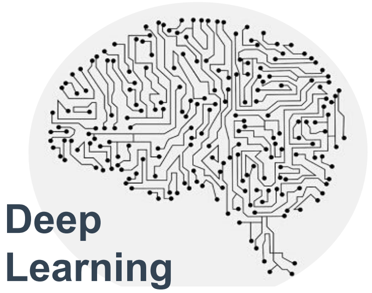


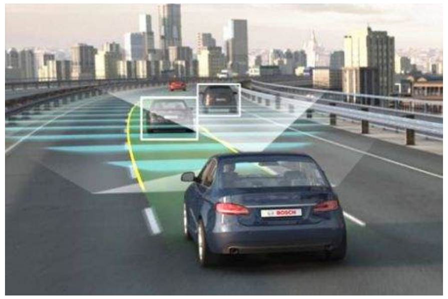
人脸认证



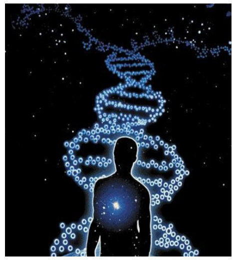
图像分类







自动驾驶



生命科学

游戏博弈

语言翻译

A little history about deep learning

深度学习为什么有效

开源的计算平台

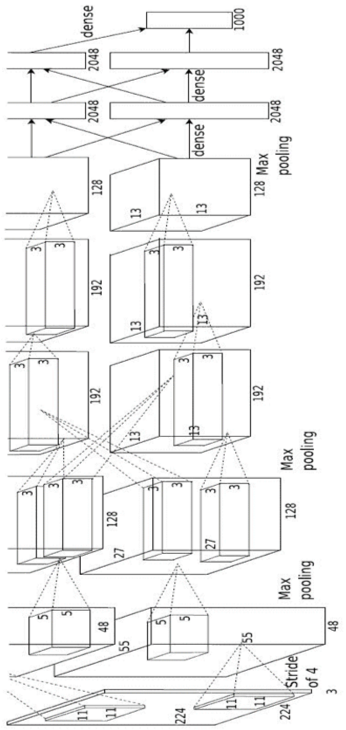
Caffe、Tensorflow、Pytorch 等

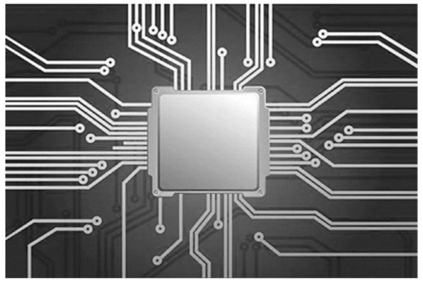
计算力是引擎

GPU服务器、集群

数据就是燃料

如ImageNet图像分类数据集等百万张图片



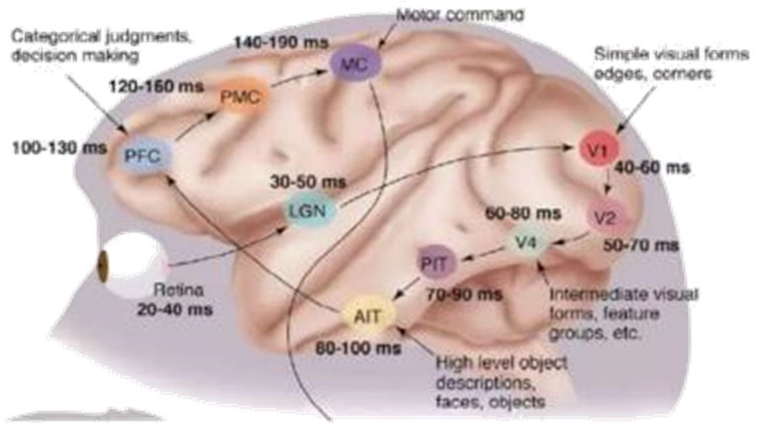


A little history about deep learning

深度学习助力人工智能

深度学习的“层次化”学习思想与人类视觉认知机理高度适应

1958 年， 约翰霍普金斯大学 David Hubel和Torsten Wiesel发现人的视觉系统的信息处理是分级的，人对物品的识别可能是一个不断迭代不断抽象的过程。

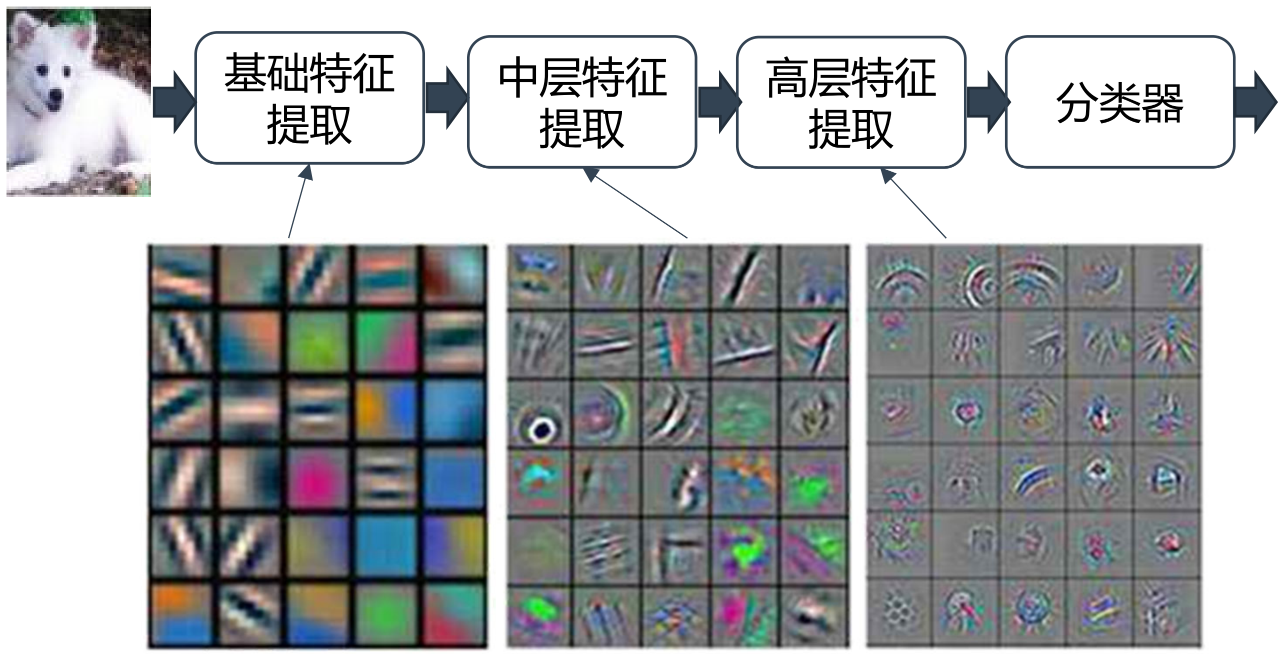


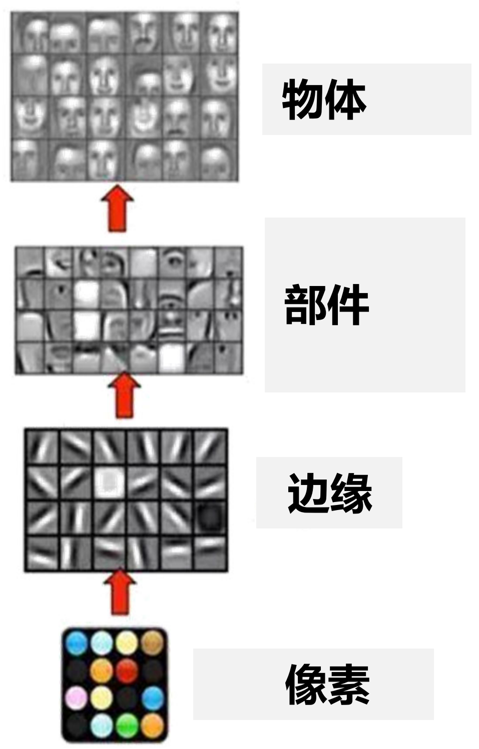
Motor command
Categorical judgments, decision making
140-190 ms
MC
120-160 ms
PMC
Simple visual forms edges, corners
100-130 ms
PFC
V1
40-60 ms
30-50 ms
LGN
60-80 ms
V2
V4
50-70 ms
PIT
Retina
20-40 ms
70-90 ms
Intermediate visual forms, feature groups, etc.
AIT
80-100 ms
High level object descriptions, faces, objects

这一发现是神经科学与认知领域的重大突破，促进了人工智能领域以后五十年的发展

A little history about deep learning

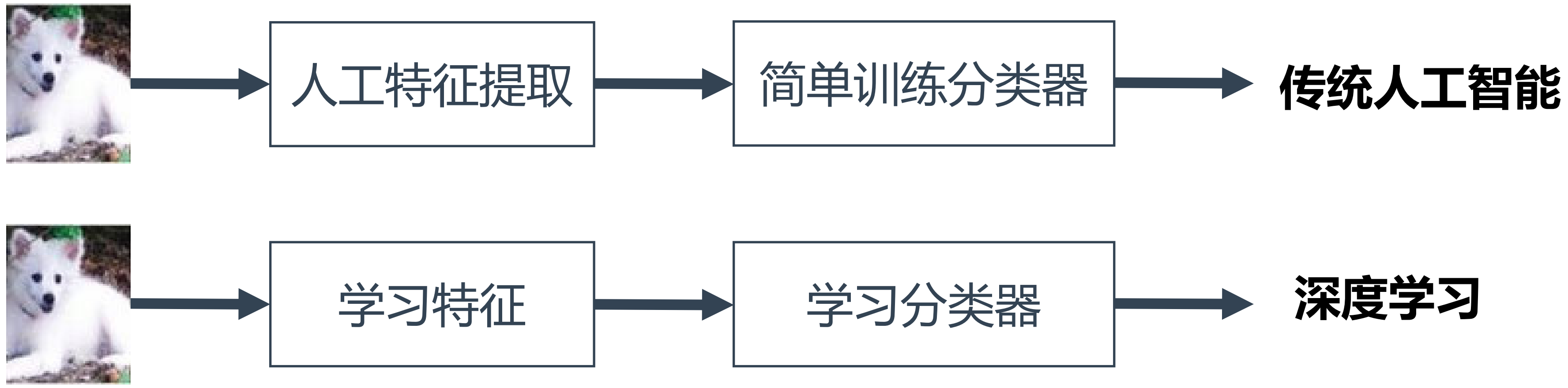
深度学习助力人工智能





A little history about deep learning

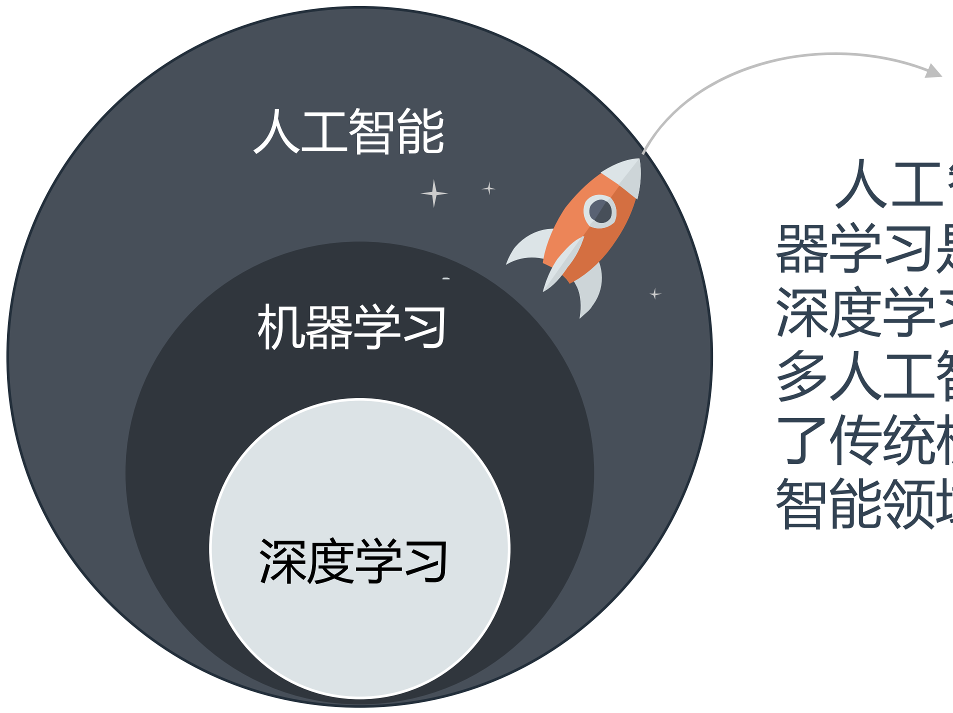
深度学习助力人工智能



与传统人工智能相比，深度学习的最大特点就在于能够自动学习与任务相适应的特征

A little history about deep learning

深度学习与人工智能



人工智能是一个非常广泛的问题，机器学习是解决这类问题的一个重要手段。深度学习则是机器学习的一个分支。在很多人工智能问题上，深度学习的方法突破了传统机器学习方法的瓶颈，推动了人工智能领域的发展

When you know these terminologies, you are an expert of DCNN

Conv layer
Pooling
Padding
Stride
ReLU
Leaky ReLU
SiLU
Res Unit
CBL (conv+BN+L-ReLU)
SPPF
(Spatial Pyramid Pooling Fast)
Fully connection
Skip connection (shortcut)
Concat
Batch normalization
Softmax
l_2-loss
Cross entropy loss
BCE (binary cross entroy)
Backbone+Head
Anchor box (point)

LeNet
AlexNet
NIN
GoogLeNet
VGGNet
ResNet
DenseNet
MobileNet
ShuffleNet
PeleeNet
ShuffleNetV2
EfficientNet

RCNN
Fast-RCNN
Faster-RCNN
SPPNet
MaskNet
SSD
Yolo
YoloV2
YoloV3
….
YoloV8
Pelee-SSD
EfficientDet

Caffe
Caffe2
TensorFlow
MatConvNet
Theano
Torch
PyTorch
Keras
MXNet
DIGITS
TensorRT

When you know these terminologies, you are an expert of DCNN

GTX980
GTX1080
TitanX
TitanXP
Tesla K40
Tesla K80
Jetson TX1
Jetson TX2
RTX 3080
RTX 3090
RTX 4080
RTX 4090

Minibatch SGD
Batch size
Iteration
Epoch
Learning rate
AdaGrad
RMSprop
Momentum
Nesterov
Adam
Finetuning
Pre-training
Early stop
Data augmentation

Gradient vanishing
Adversarial Samples
Nonconvex optimization
Saddle points

SYNTHIA
Virtual Kitti
URSA
VIPER
synMT
Grand Theft Auto V

Geoffrey Hinton
Alex Krizhevsky
Yoshua Bengio
Yann LeCun
Ian Goodfellow
Kaiming He
Ross Girshick
Joseph Redmon

ImageNet
VOC2007
VOC2012
MS COCO
Kitti

Convolutional neural network

- Specially designed for data with grid-like structures (LeCun et al. 98)
  - 1D grid: sequential data
  - 2D grid: image
  - 3D grid: video, 3D image volume
- Beat all the existing computer vision technologies on object recognition on ImageNet challenge with a large margin in 2012

Convolutional neural network

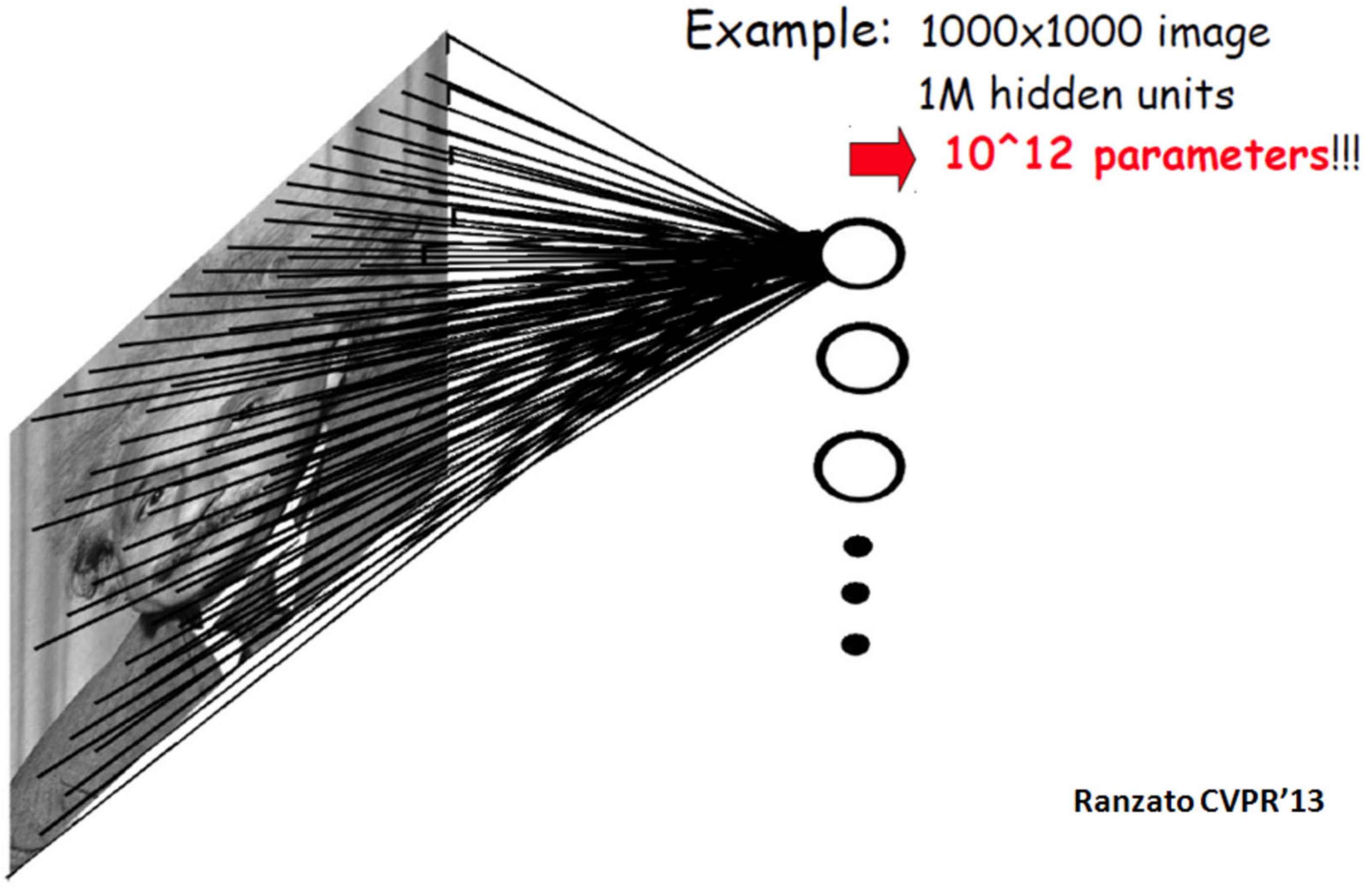
- Something you need to know about DCNN
  - Traditional model for PR: fixed/engineered features + trainable classifier
  - For DCNN: it is usually an **end-to-end** architecture; learning data representation and classifier together
  - The learned features from big datasets are **transferable**
  - For training a DCNN, usually we use a **fine-tuning** scheme
  - For training a DCNN, to avoid overfitting, **data augmentation** can be performed

Convolutional neural network

- Problems of fully connected networks
  - Every output unit interacts with every input unit
  - The number of weights grows largely with the size of the input image
  - Pixels in distance are less correlated

Convolutional neural network

- Problems of fully connected networks

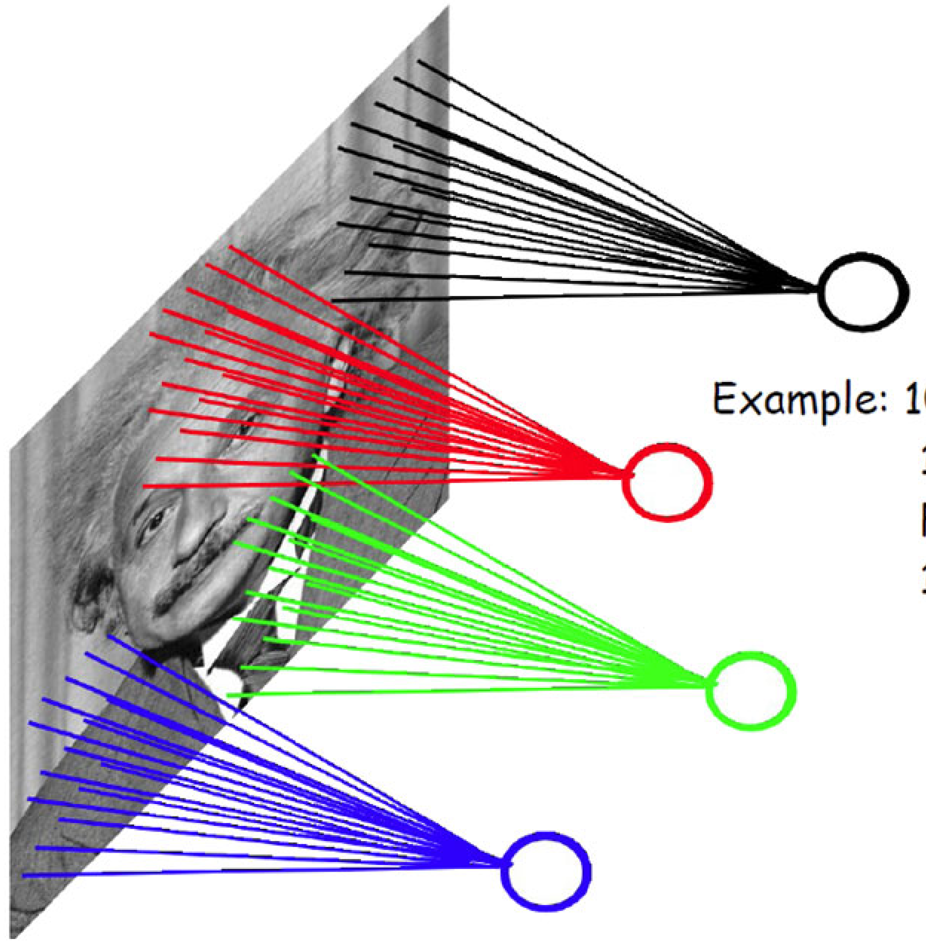


Convolutional neural network

- One simple solution is locally connected neural networks
  - Sparse connectivity: a hidden unit is only connected to a local patch (weights connected to the patch are called filter or kernel)
  - It is inspired by biological systems, where a cell is sensitive to a small sub-region of the input space, called a receptive field; Many cells are tiled to cover the entire visual field

Convolutional neural network

- One simple solution is locally connected neural networks

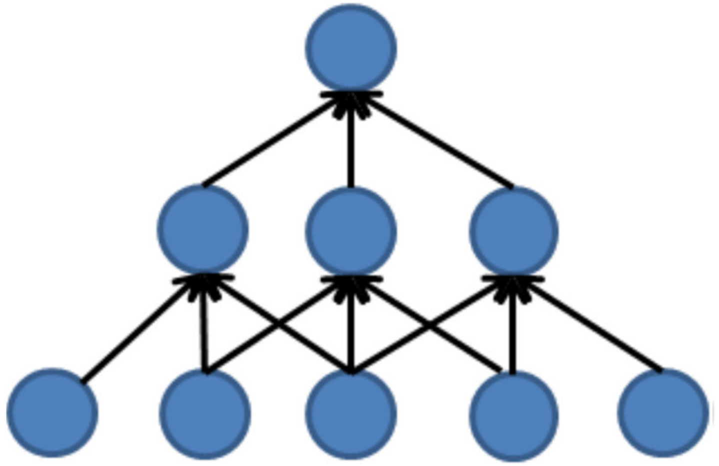


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Ranzato CVPR'13

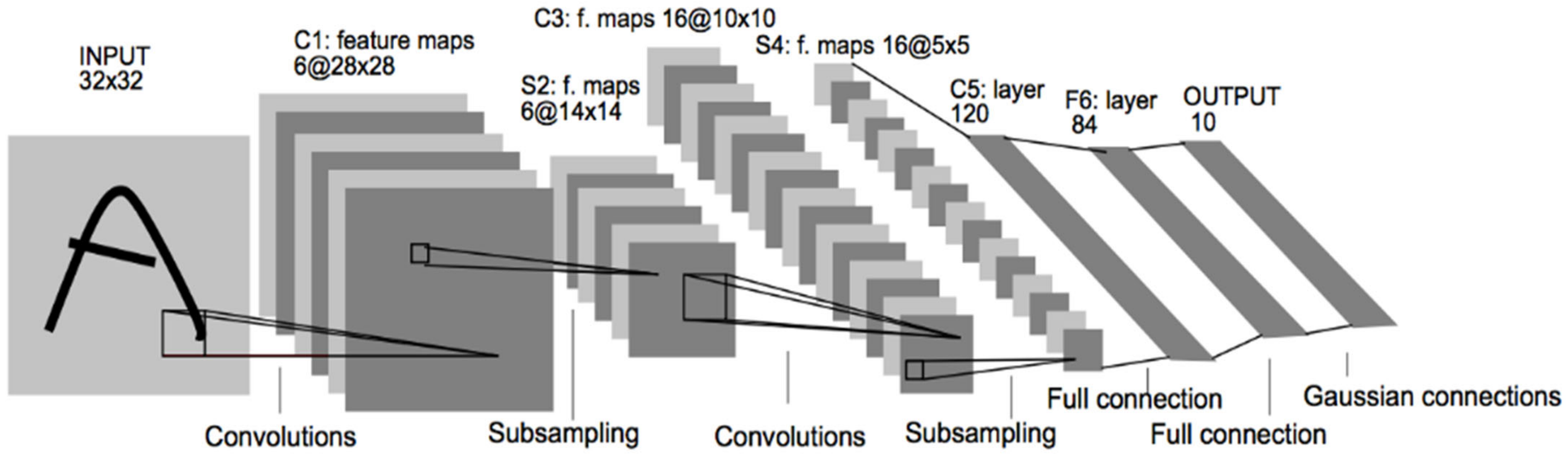
Convolutional neural network

- One simple solution is locally connected neural networks
  - The learned filter is a spatially local pattern
  - A hidden node at a higher layer has a larger receptive field in the input
  - Stacking many such layers leads to “filters” (not anymore linear) which become increasingly “global”



Convolutional neural network

- The first CNN
  - LeNet[5]



CNN called LeNet by Yann LeCun (1998)

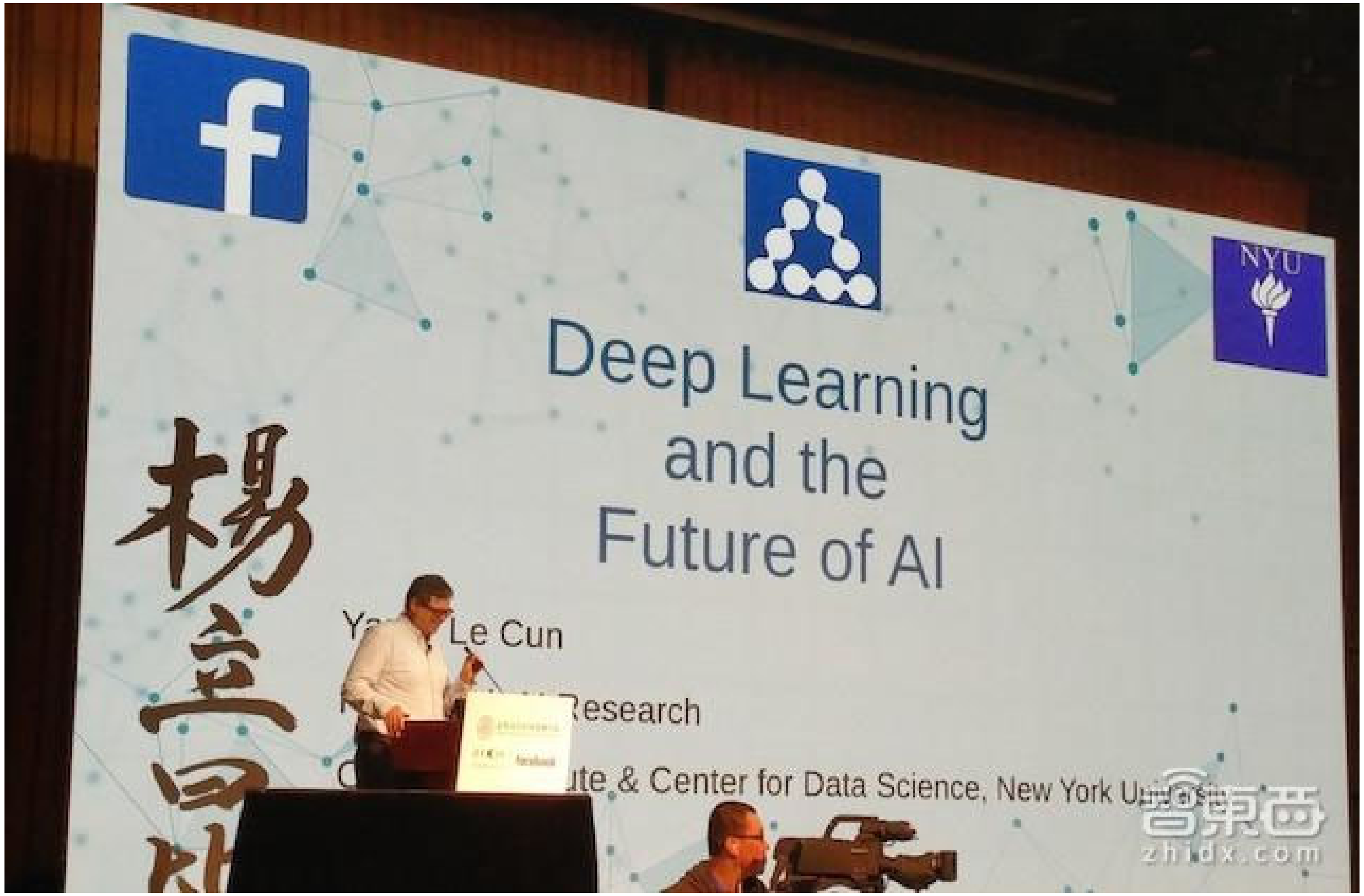
[5] Y. LeCun et al., Gradient-based Learning Applied to Document Recognition, Proceedings of the IEEE, Vol. 86, pp. 2278-2324, 1998

Convolutional neural network

Yann LeCun (1960-)
New York University
Facebook Artificial Intelligence Research
A founding father of convolutional nets

(His name is a little difficult to pronounce)

Convolutional neural network

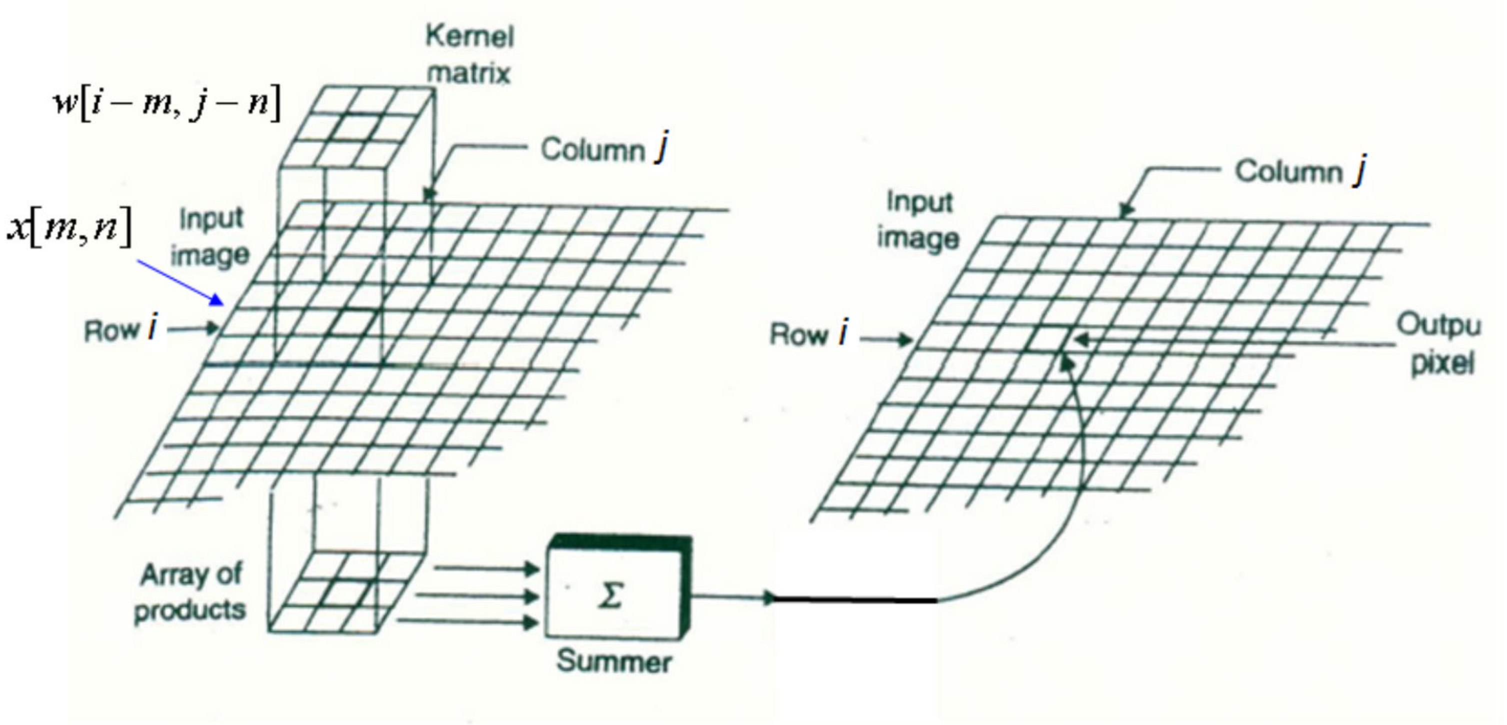


He gives himself a Chinese name, 杨立昆 (Mar. 2017)

Convolutional neural network

- Convolution
  - Computing the responses at hidden nodes is equivalent to convoluting the input image x with a learned filter w

$$net[i,j] = (x * w)[i,j] = \sum_m \sum_n x[m,n]\,w[i-m, j-n]$$



Convolutional neural network

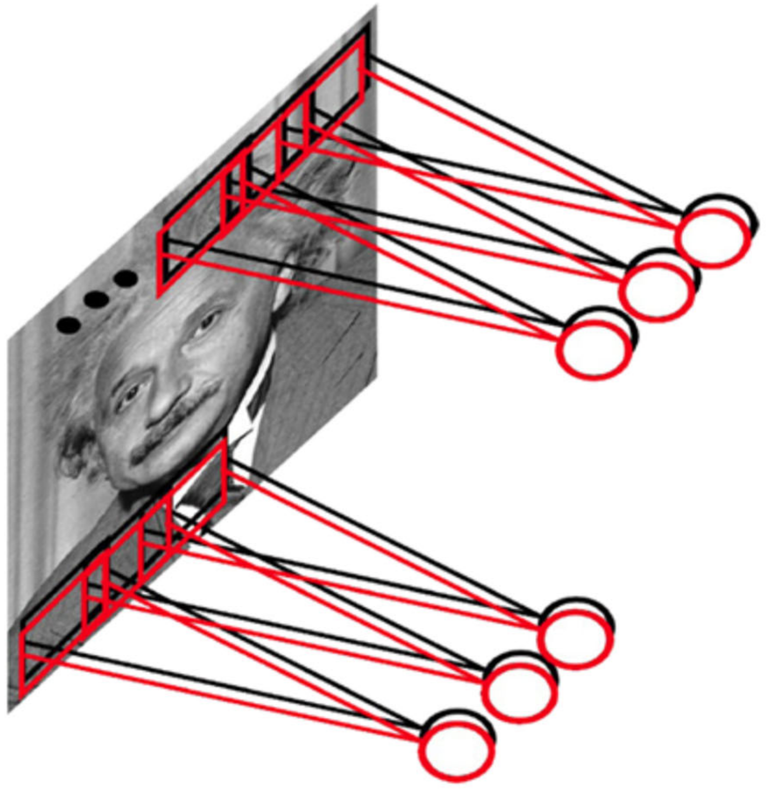
- Downsampled convolution layer (optional)
  - To reduce computational cost, we may want to skip some positions of the filter and sample only every s pixels in each direction. A downsampled convolution function is defined as

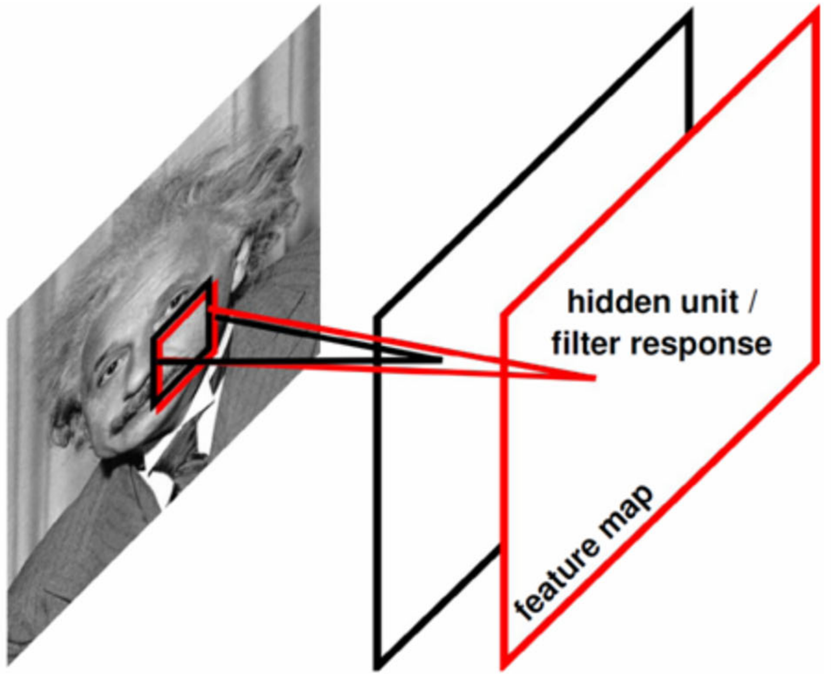
$$net(i, j) = (\mathbf{x} * \mathbf{w})[i \times s, j \times s]$$

  - s is referred as the stride of this downsampled convolution
  - Also called as strided convolution

Convolutional neural network

- Multiple filters
  - Multiple filters generate multiple feature maps
  - Detect the spatial distributions of multiple visual patterns

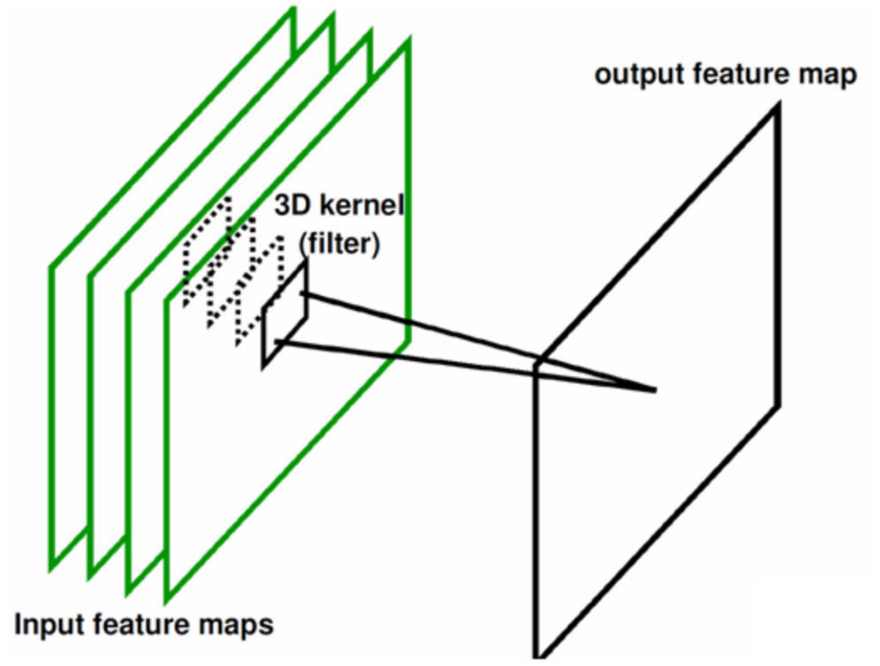


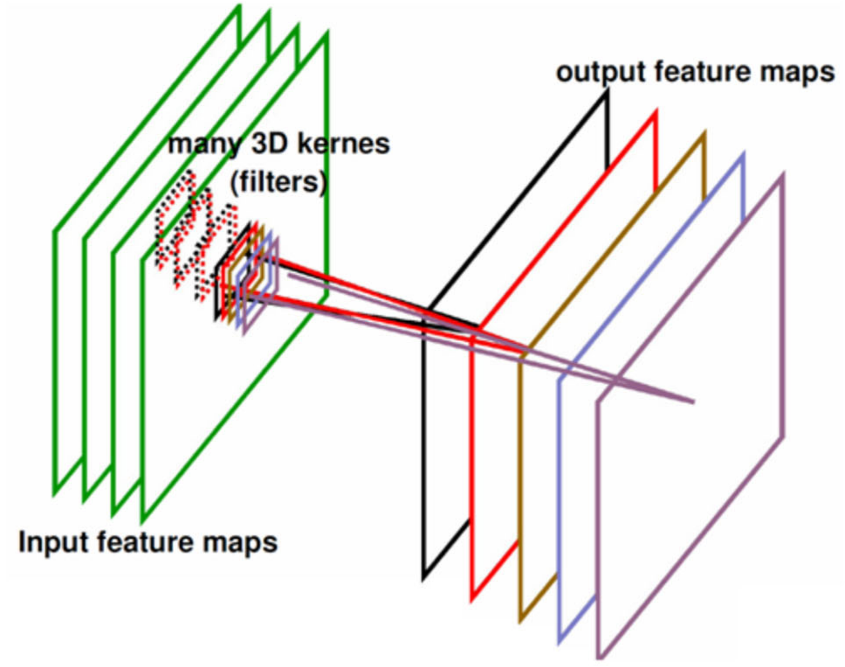


Ranzato CVPR'13

Convolutional neural network

- 3D filtering when input has multiple feature maps

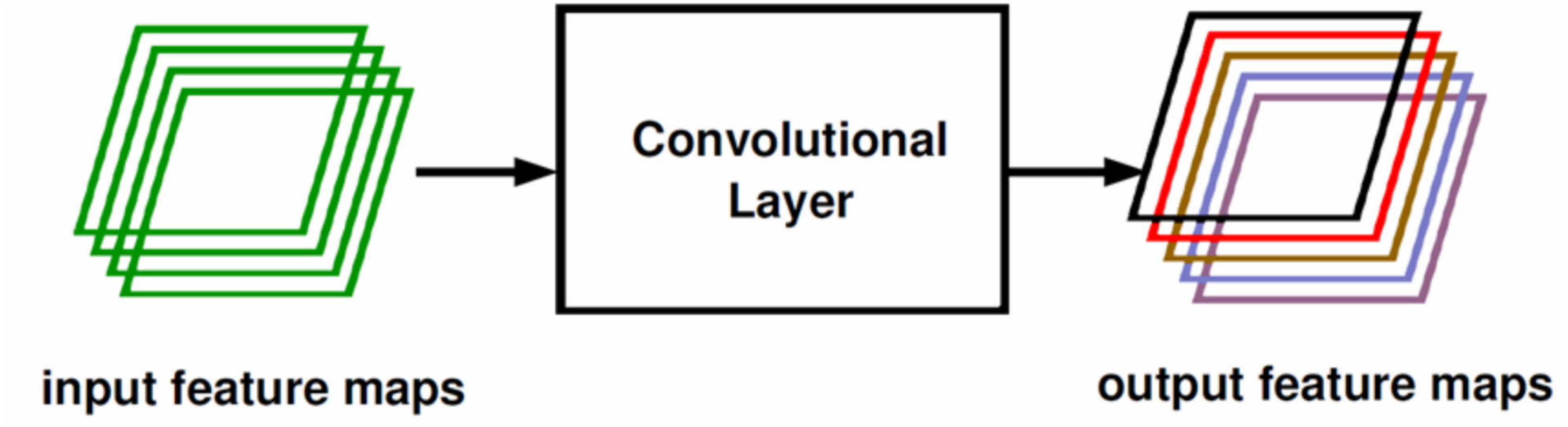




Ranzato CVPR'13

Convolutional neural network

- Convolutional layer



Ranzato CVPR'13

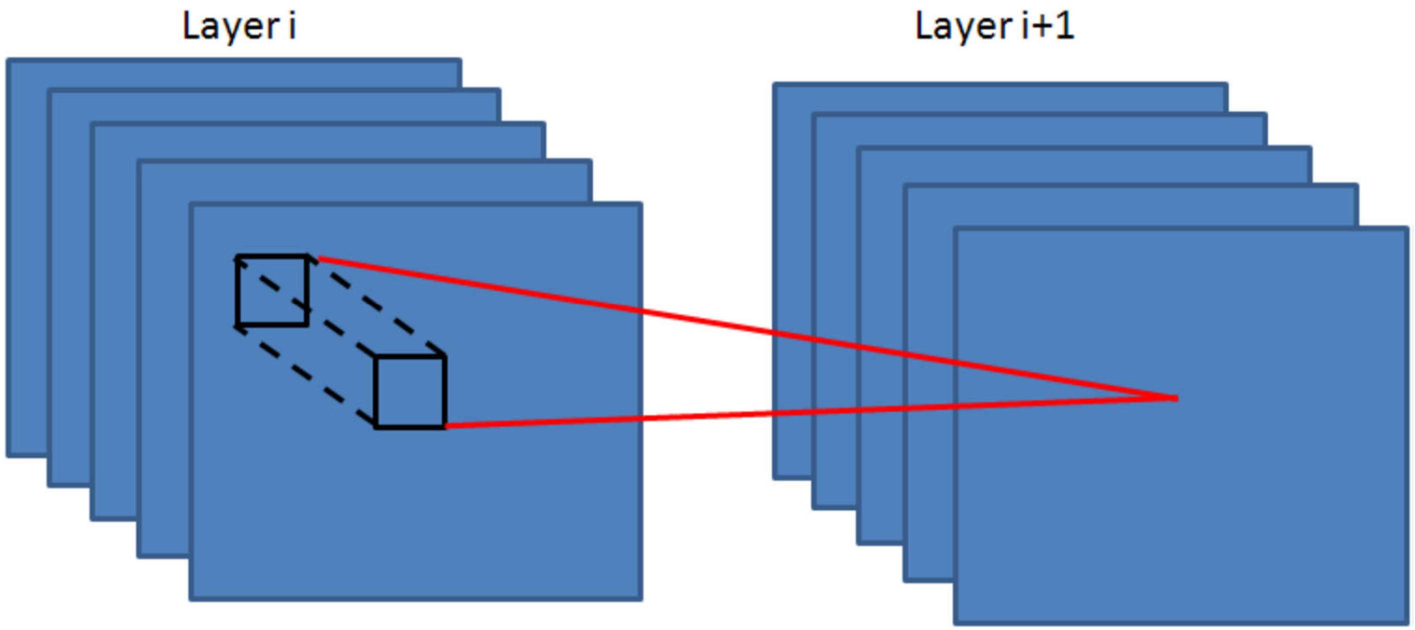
Convolutional neural network

- To the convolution responses, we then perform nonlinear activation
  - ReLU
  - Tanh
  - Sigmoid
  - Leaky ReLU
  - Softplus
  - SiLU

Convolutional neural network

- Local contrast normalization (optional)
  - Normalization can be done within a neighborhood along both spatial and feature dimensions

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

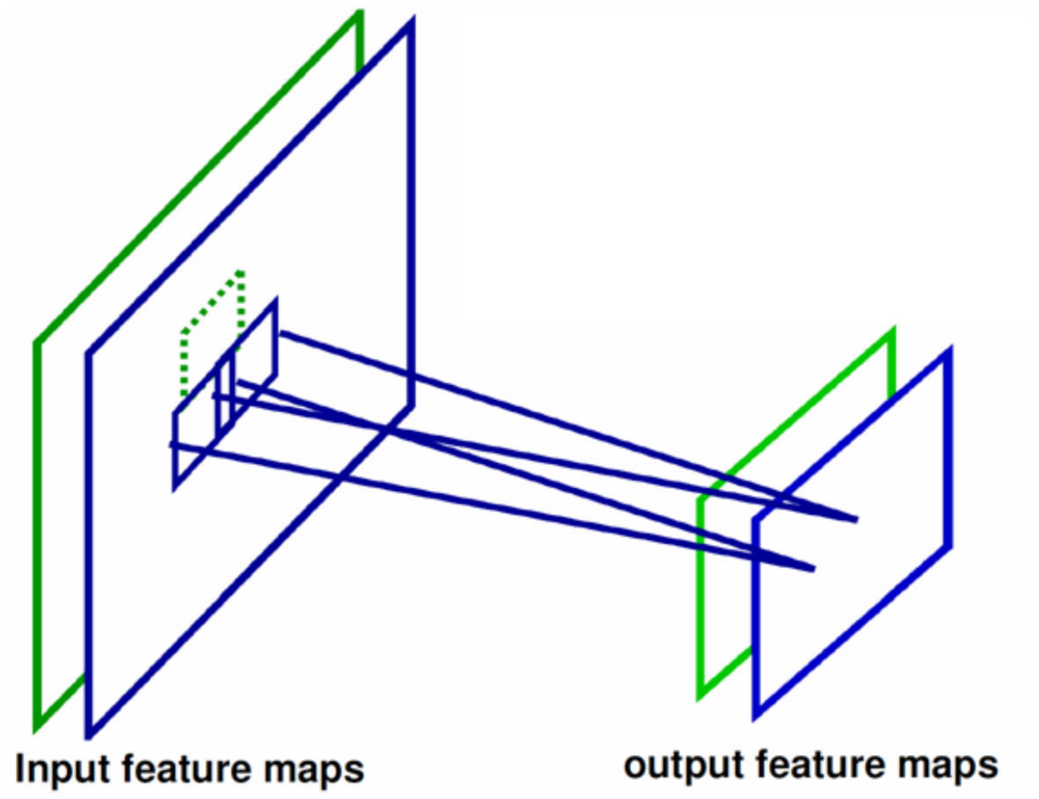


Convolutional neural network

- Then, we perform pooling
  - Max-pooling partitions the input image into a set of rectangles, and for each sub-region, outputs the maximum value
  - Non-linear down-sampling
  - The number of output maps is the same as the number of input maps, but the resolution is reduced
  - Reduce the computational complexity for upper layers and provide a form of translation invariance
  - Average pooling can also be used

Convolutional neural network

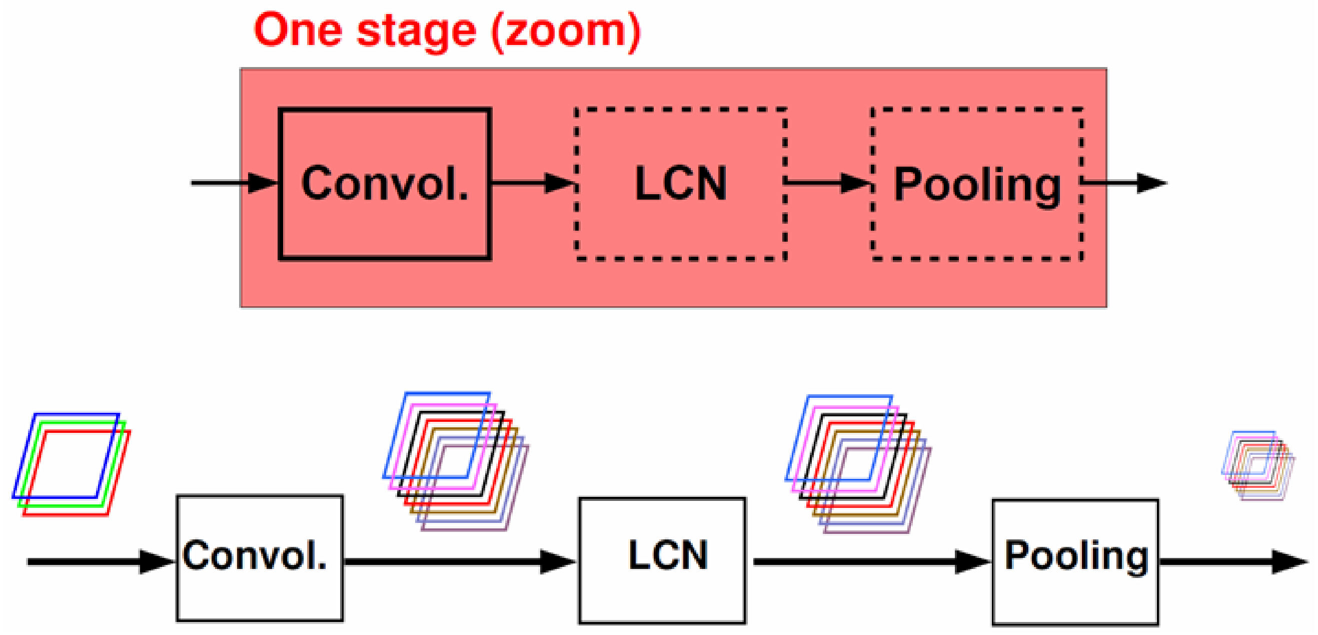
- Then, we perform pooling



Ranzato CVPR'13

Convolutional neural network

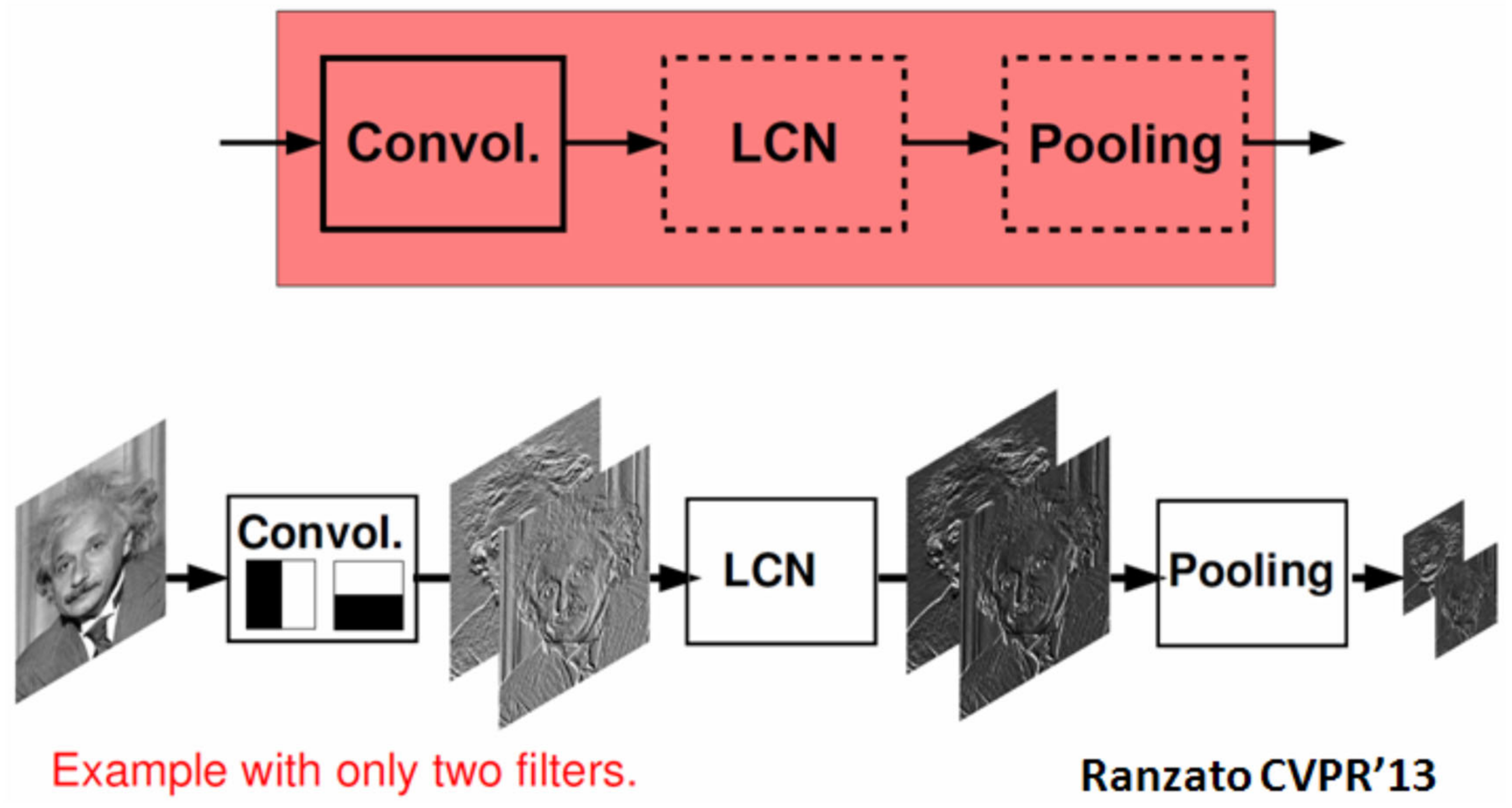
- Typical architecture of CNN
  - Convolutional layer increases the number of feature maps
  - Pooling layer decreases spatial resolution
  - LCN and pooling are optional at each stage



Ranzato CVPR'13

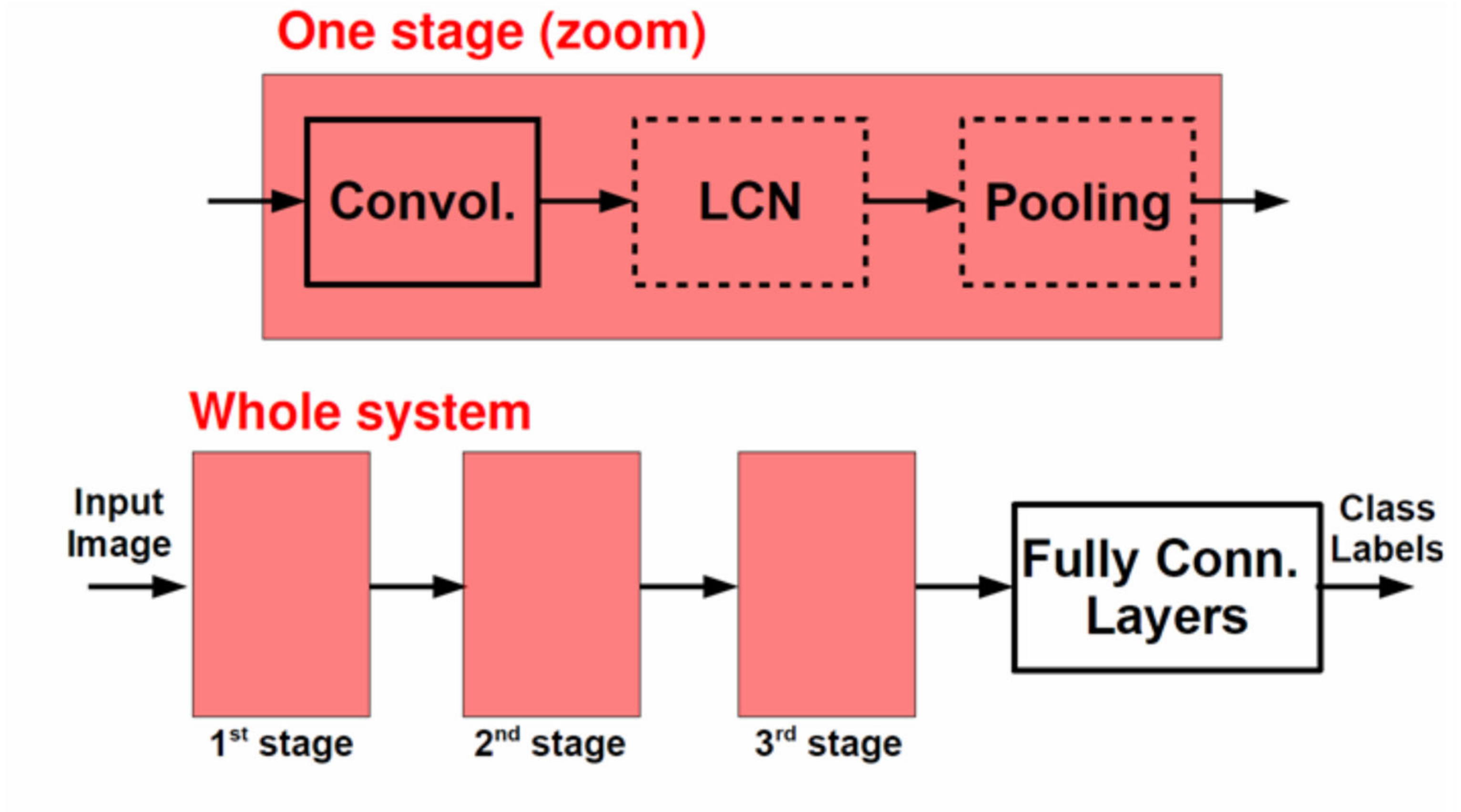
Convolutional neural network

- Typical architecture of CNN



Convolutional neural network

- Typical architecture of CNN

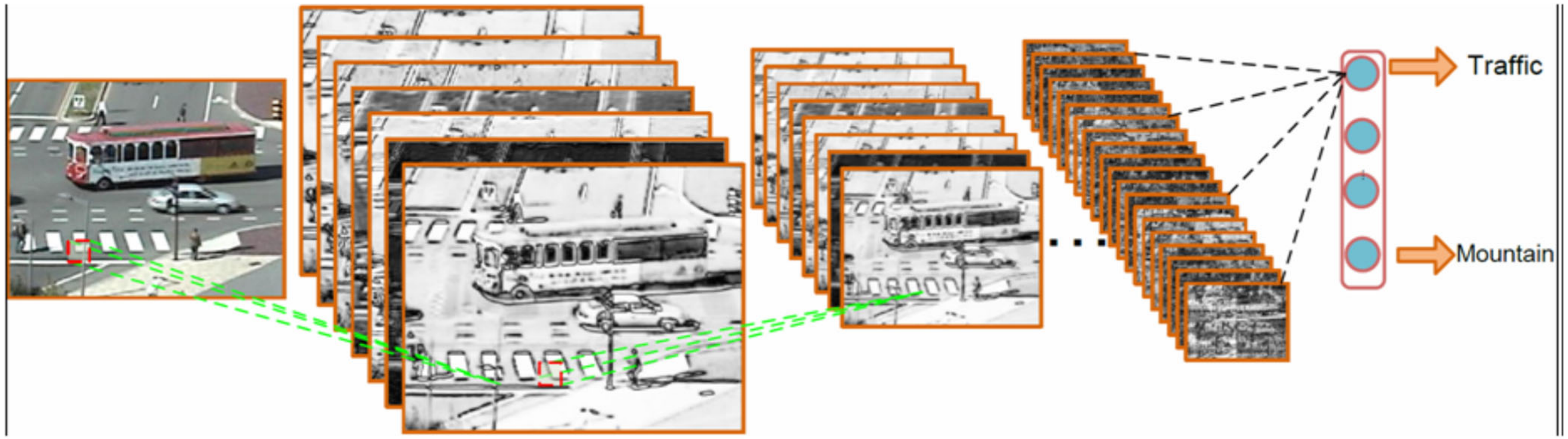


After a few stages, residual spatial resolution is very small.
We have learned a descriptor for the whole image.

Ranzato CVPR'13

Convolutional neural network

- Typical architecture of CNN



Convolution Pooling

Convolutional neural network

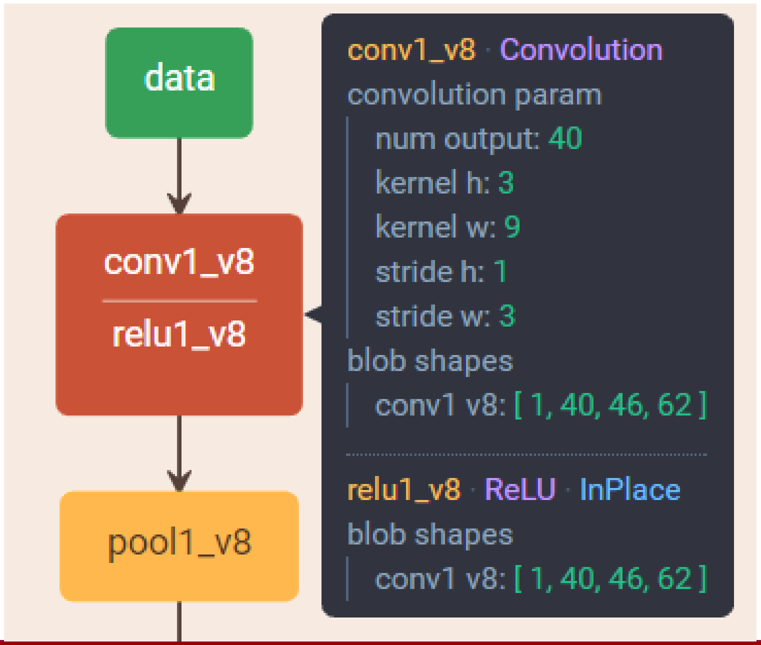
- Some notes about the CNN layers in most recent net architectures
  - Spatial pooling (such as max pooling) is not recommended now. It is usually replaced by a strided convolution, allowing the network to learn its own spatial downsampling
  - Fully connected layers are not recommended now; instead, the last layer is replaced by global average pooling (for classification problems, the number of feature map channels of the last layer should be the same as the number of classes

Convolutional neural network

- Opensource platforms for CNN
  - CAFFE official, http://caffe.berkeleyvision.org/
  - Tensorflow, https://www.tensorflow.org/
  - Pytorch, www.pytorch.org/
  - Theano, http://deeplearning.net/software/theano/

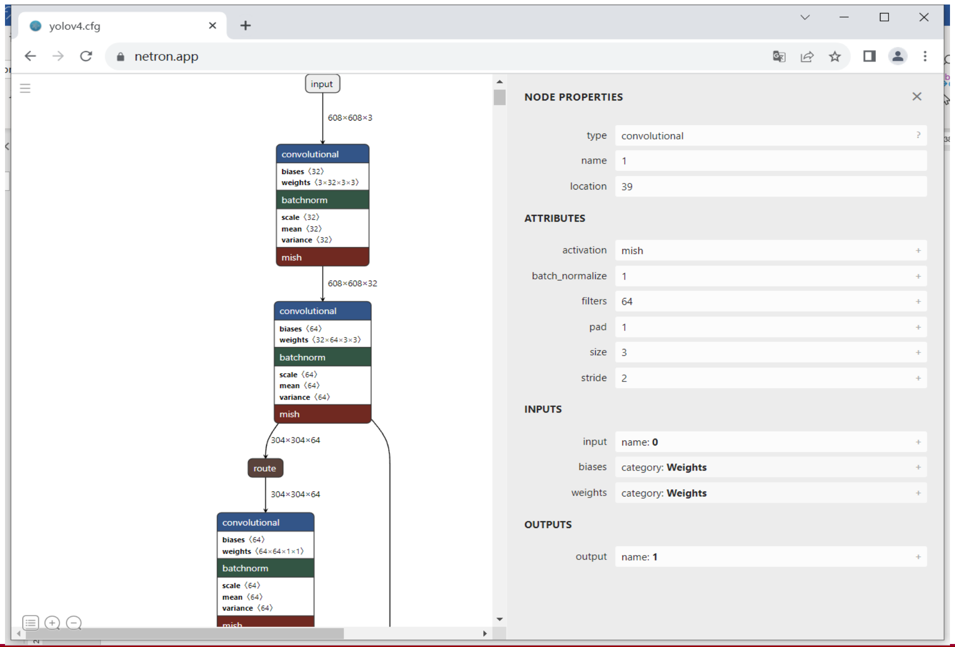
Convolutional neural network

- Online tools for network architecture visualization
  - http://ethereon.github.io/netscope/quickstart.html
    - » Network architecture conforms to the CAFFE prototxt format
    - » The parameter settings and the output dimension of each layer can be conveniently observed
  - https://netron.app/
    - » Support CAFFE model, darknet models, ONNX formats etc



conv1_v8 · Convolution
convolution param
num output: 40
kernel h: 3
kernel w: 9
stride h: 1
stride w: 3
blob shapes
conv1 v8: [1, 40, 46, 62]

relu1_v8 · ReLU · InPlace
blob shapes
conv1 v8: [1, 40, 46, 62]



NODE PROPERTIES

type	convolutional
name	1
location	39

ATTRIBUTES

activation	mish
batch_normalize	1
filters	64
pad	1
size	3
stride	2

INPUTS

input	name: **0**
biases	category: **Weights**
weights	category: **Weights**

OUTPUTS

output	name: **1**

Outline

- Neural network
- Convolutional neural network (CNN)
- Modern CNN architectures
  - AlexNet
  - NIN
  - GoogLeNet
  - ResNet
  - DenseNet
  - PeleeNet
- CNN for object detection

AlexNet (NIPS 2012)

- AlexNet: CNN for object recognition on ImageNet challenge
  - Trained on one million images of 1000 categories collected from the web with two GPU. 2GB RAM on each GPU. 5GB of system memory
  - Training lasts for one week
  - Google and Baidu announced their new visual search engines with the same technology six months after that
  - Google observed that the accuracy of their visual search engine was doubled

[6] A. Krizhevsky *et al*., ImageNet classification with deep convolutional neural networks, in Proc. NIPS, 2012

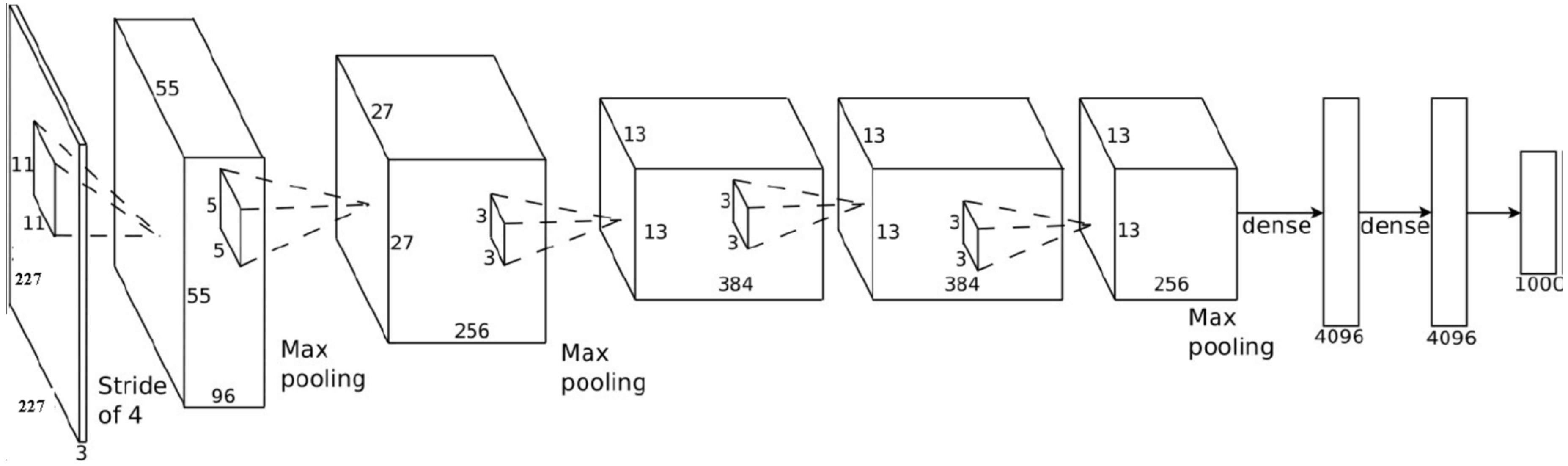
AlexNet (NIPS 2012)

- ImageNet
  - http://www.image-net.org/

1000 object classes that we recognize
poster created by Fengjun Lv using VIPBase
images courtesy of ImageNet (http://www.image-net.org/challenges/LSVRC/2010/index)

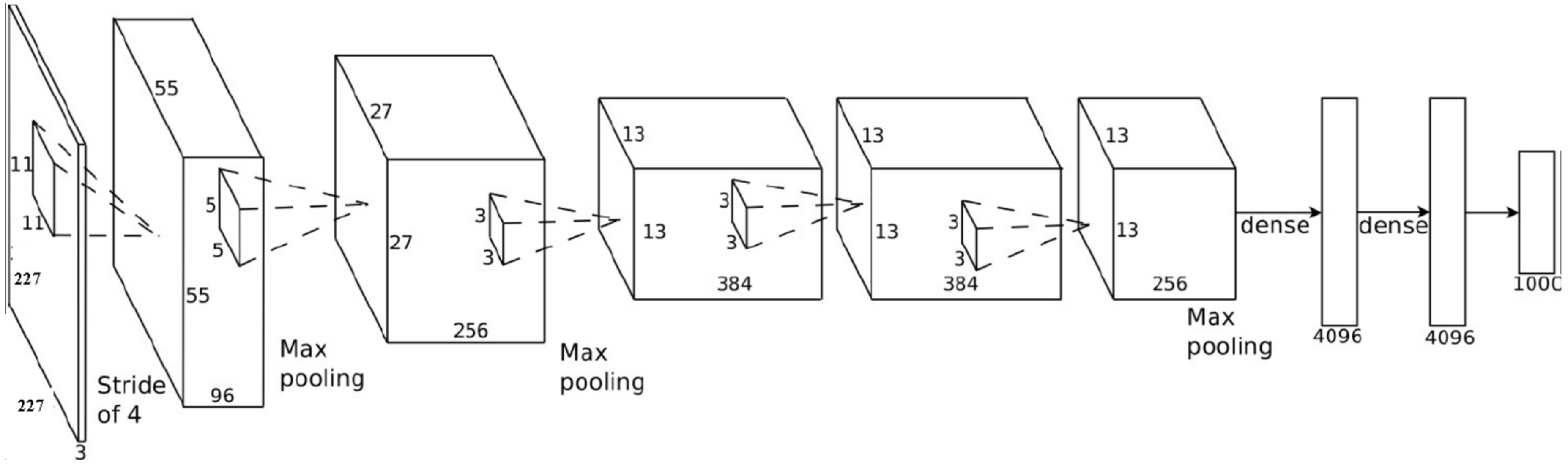
AlexNet (NIPS 2012)

- Architecture of AlexNet
  - 5 convolutional layers and 2 fully connected layers for learning features
  - Max-pooling layers follow first, second, and fifth convolutional layers



AlexNet (NIPS 2012)

- Architecture of AlexNet
  - The first time deep model is shown to be effective on large scale computer vision task
  - The first time a very large scale deep model is adopted
  - GPU is shown to be very effective on this large deep model

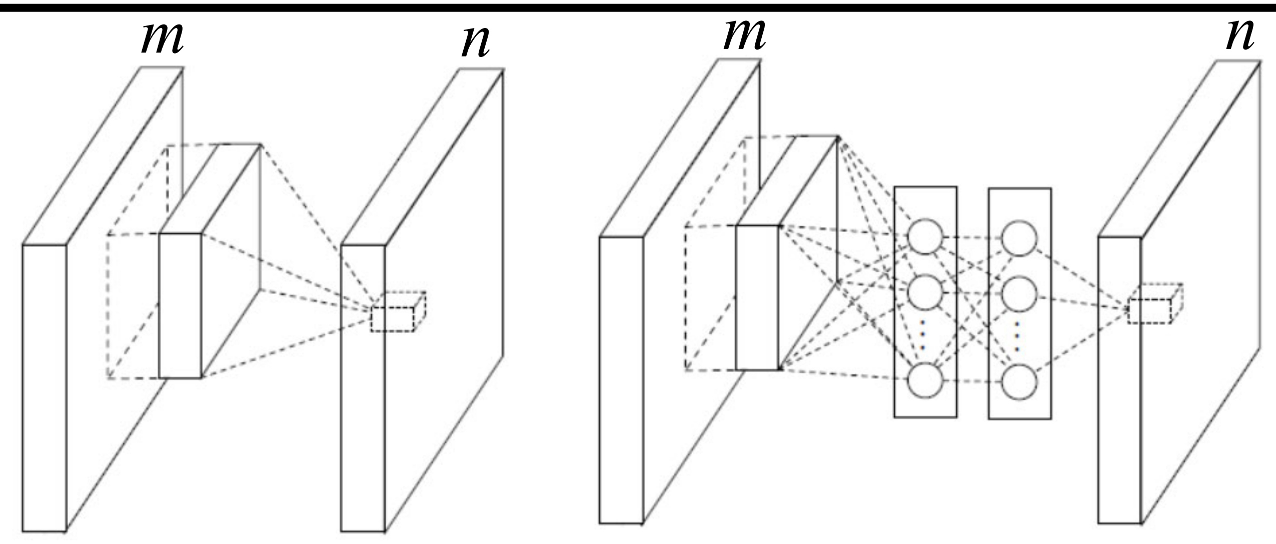


Network In Network (NIN, ICLR 2014)

- Main idea of NIN
  - Conventional convolutional layers uses linear filters followed by a nonlinear activation function to abstract the information within a receptive field
  - Instead, NIN uses micro neural networks with more complex structures to abstract the data within the receptive field
  - The feature maps are obtained by sliding the micro network over the input in a similar manner as CNN
  - Moreover, they use global average pooling over feature maps in the classification layer, which is easier to interpret and less prone to overfitting than traditional fully connected layers

[7] M. Liu *et al*., Network in network, in Proc. ICLR, 2014

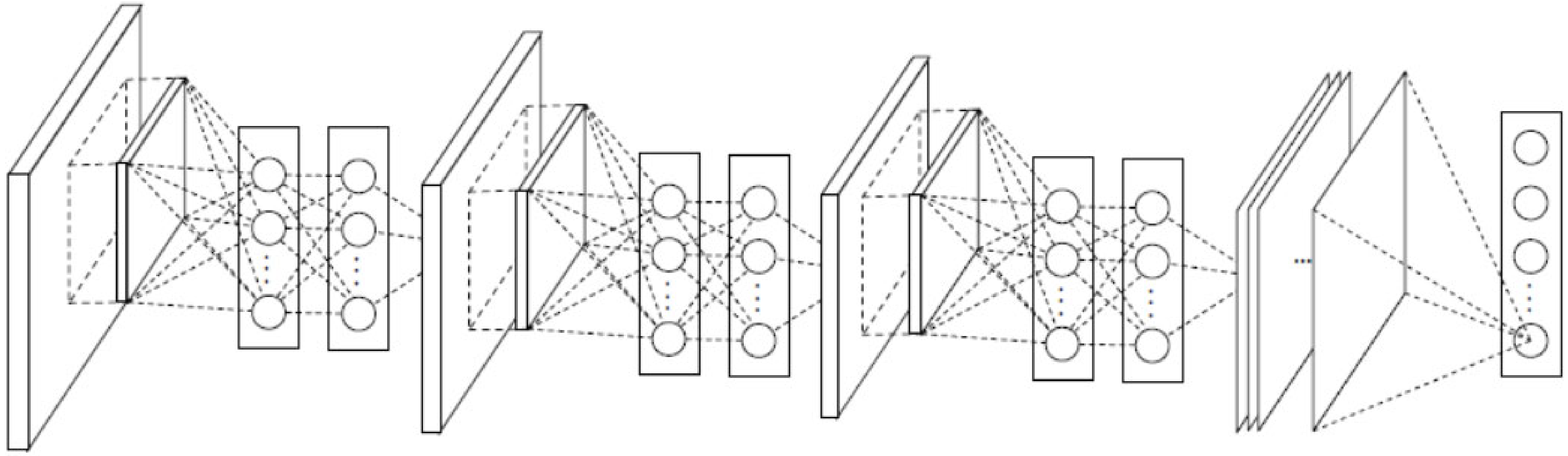
Network In Network (NIN, ICLR 2014)



(a) Linear convolution layer (b) Mlpconv layer

- Comparison of linear convolution layer and mlpconv layer
  - Both the layers map the local receptive field to an output feature vector
  - The mlpconv layer maps the input local patch to the output feature vector with a multilayer perceptron (MLP) consisting of multiple fully connected layers with nonlinear activation functions

Network In Network (NIN, ICLR 2014)



The overall structure of NIN. The last layer is the global average pooling

- More about global average pooling
  - Fully connected layers are prone to overfitting
  - If there are c classes, the last MLP layer should output c feature maps, one feature map for each corresponding category of the classification task
  - Take the average of each feature map to get a c dimensional vector for softmax classification

Network In Network (NIN, ICLR 2014)

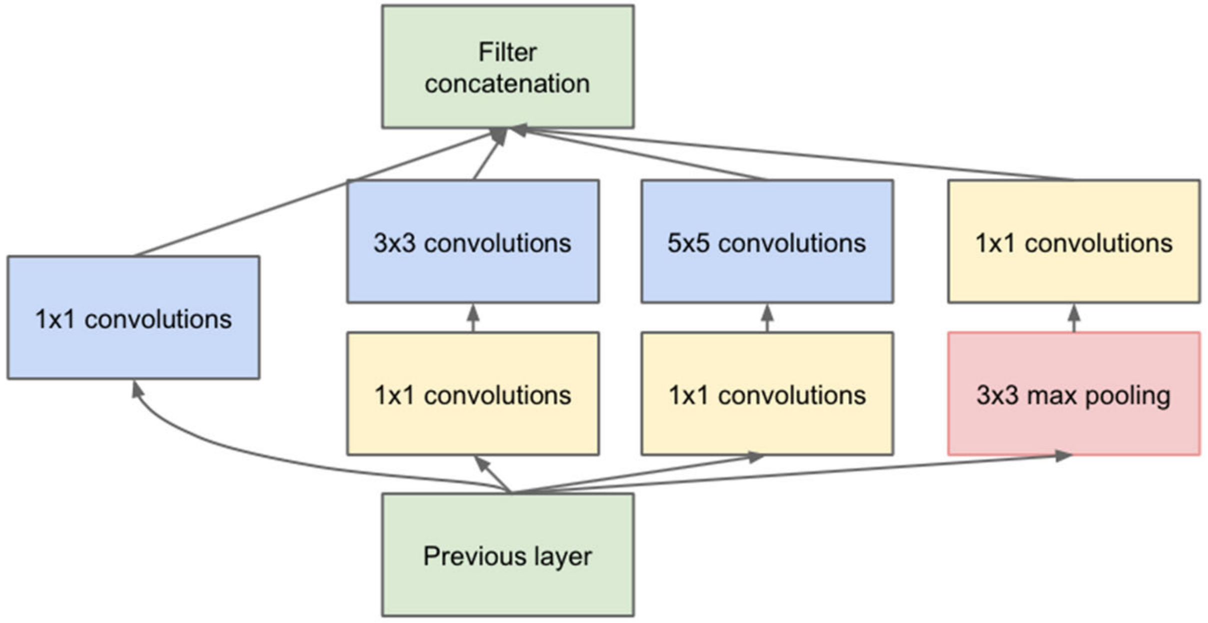
- NIN can be implemented with conventional convolutional layers

For a mlpconv layer, suppose that the input feature map is of the size $m \times m \times 32$, the expected output feature map is of the size $m \times m \times 64$, the receptive field is 5×5; the mlpconv layer has 2 hidden layers, whose node numbers are 16 and 32, respectively.

How to implement this mlpconv layer with convolutional layers?

GoogLeNet (CVPR 2015)

- Main idea: make the network deeper and wider, while keeping the number of parameters
- Inception module



[8] C. Szegedy *et al*., Going deeper with convolutions, in Proc. CVPR, 2015

GoogLeNet (CVPR 2015)

- Many Inception modules can stack together to form a very deep network
- GoogLeNet refers to the version the authors submitted for the ILSVRC 2014 competition
  - This network consists 27 layers (including pooling layers)

ResNet (CVPR 2016 Best Paper)

- What is the problem of stacking more layers using conventional CNNs?
  - Vanishing gradient, which can hamper the convergence
  - Accuracy get saturated, and then degraded

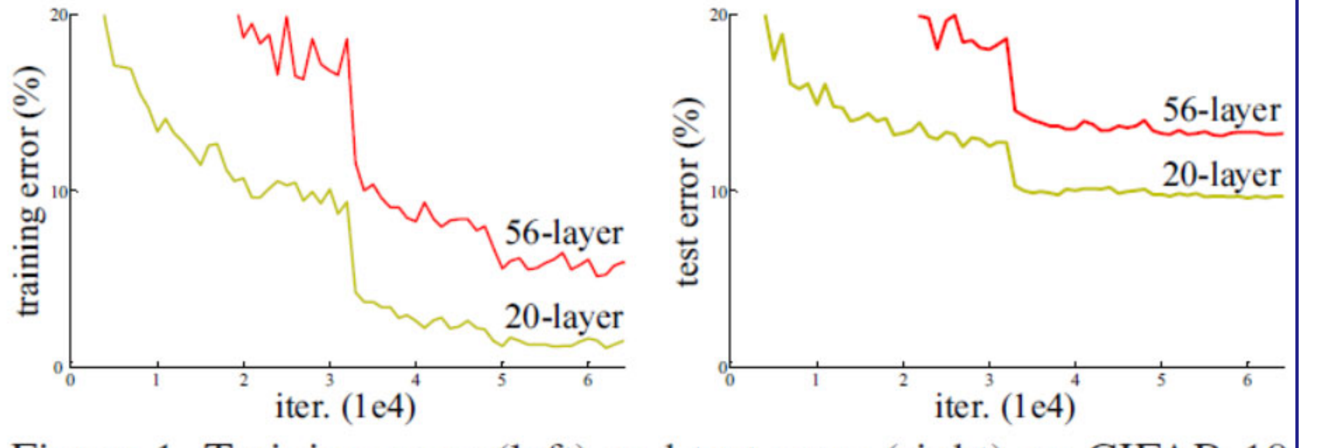


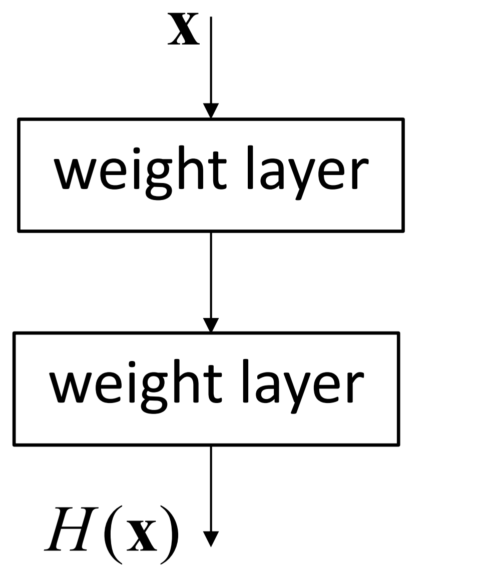
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

[9] K. He *et al*., Deep residual learning for image recognition, in Proc. CVPR, 2016

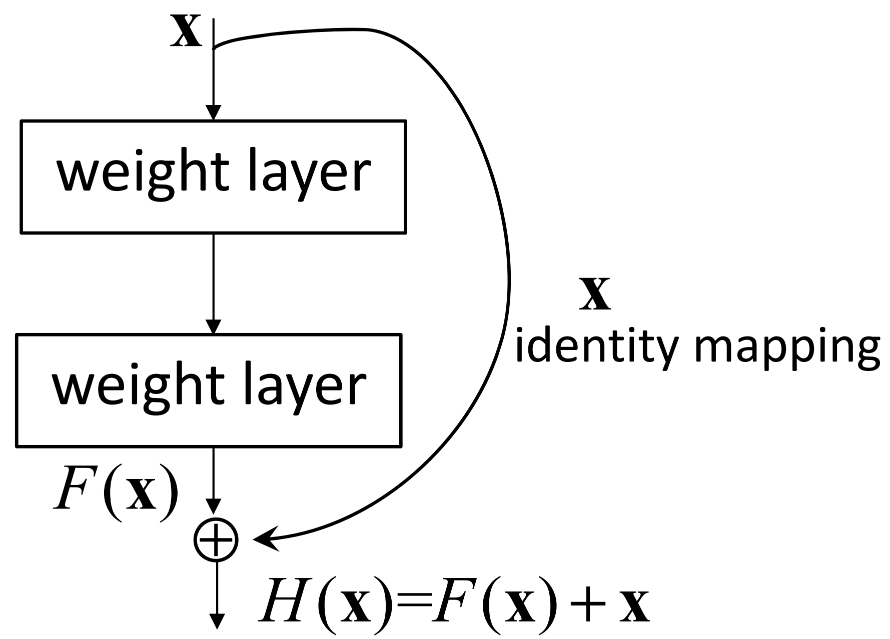
ResNet (CVPR 2016 Best Paper)

Is there any better way to design deeper networks?

Answer: Residual learning



Conventional CNN



Residual block

ResNet (CVPR 2016 Best Paper)

- It is easier to optimize the residual mapping ($F(\mathbf{x})$) than to optimize the original mapping ($H(\mathbf{x})$)
- Identity mapping is implemented by shortcut
- A residual learning block is defined as,

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers $F+\mathbf{x}$ is performed by a shortcut connection and element-wise addition

Note: If the dimensions of F and $\mathbf{x}$ are not equal (usually caused by changing the numbers of input and output channels), a linear projection W_s (implemented with 1*1 convolution) is performed on $\mathbf{x}$ to match the dimensions,

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$

ResNet (CVPR 2016 Best Paper)

- It is easier to optimize the residual mapping ($F(\mathbf{x})$) than to optimize the original mapping ($H(\mathbf{x})$)
- Identity mapping is implemented by shortcut
- A residual learning block is defined as,

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers
$F+\mathbf{x}$ is performed by a shortcut connection and element-wise addition

I highly recommend you to take a look the prototxt file of ResNet *(https://github.com/KaimingHe/deep-residual-networks)*

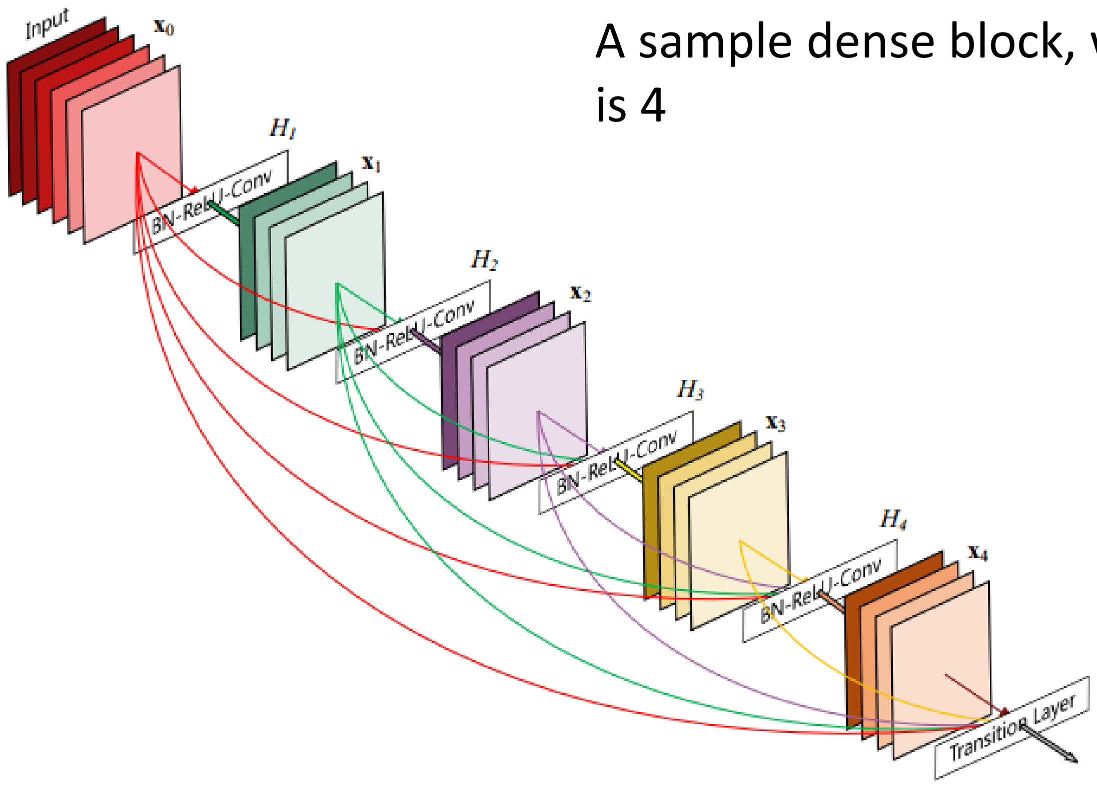
DenseNet (CVPR 2017 Best Paper)

- Highly motivated by ResNet
- A DenseNet comprises “dense blocks” and transition layers
  - Within a dense block, connect all layers with each other in a feed-forward fashion
  - In contrast to ResNet, DenseNet combine features by concatenating them
  - The number of output feature maps of each layer is set as a constant within a dense block and is called as “growth rate”
  - Between two blocks, there is a transition layer, consisting of batch normalization, 1×1 convolution, and average pooling

[10] G. Huang *et al.*, Densely connected convolutional networks, in Proc. CVPR, 2017

DenseNet (CVPR 2017 Best Paper)

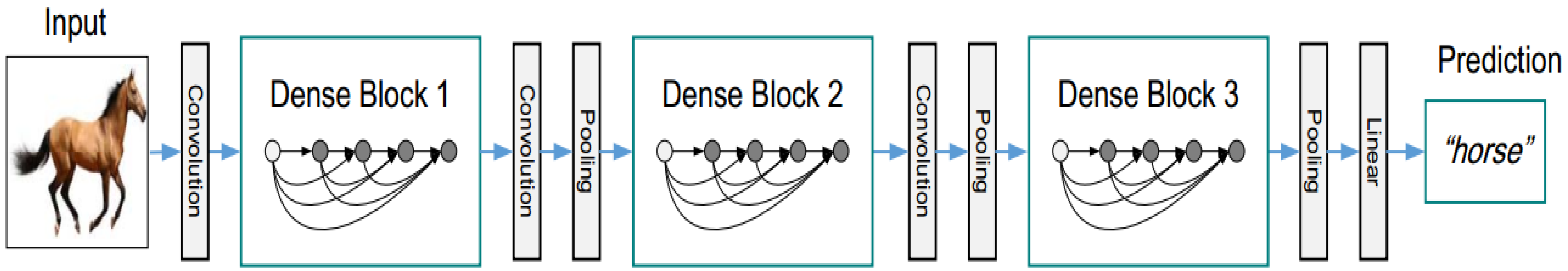
- Highly motivated by ResNet
- A DenseNet comprises “dense blocks” and transition layers



A sample dense block, whose growth rate is 4

DenseNet (CVPR 2017 Best Paper)

- Highly motivated by ResNet
- A DenseNet comprises “dense blocks” and transition layers



A sample DenseNet with three dense blocks

DenseNet (CVPR 2017 Best Paper)

- Highly motivated by ResNet
- A DenseNet comprises "dense blocks" and transition layers
- More details about DenseNet design
  - Bottleneck layers. A 1×1 convolution layer can be introduced as bottleneck layer before each 3×3 convolution to reduce the number of input feature maps, and thus to improve computational efficiency
  - Compression. If a dense block contains m feature maps, we let the following transition layer generate θm output feature maps where $0<\theta\leq1$ is referred to as the compression factor

PeleeNet (ICLR Workshop 2018)

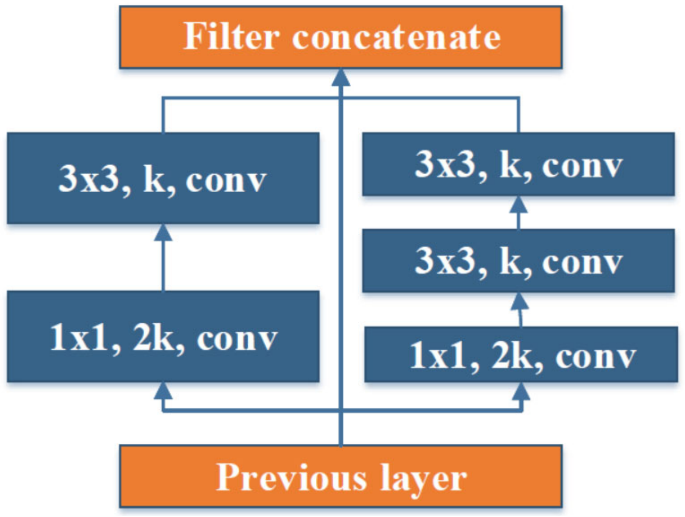
- MobileNet and ShuffleNet are designed to run on mobile devices with limited computing power and memory resource
- However, they both depend on depthwise separable convolution which lacks efficient implementation in most deep learning frameworks
- PeleeNet
  - based on conventional convolution instead
  - follows the innovate connectivity pattern and some of key design principals of DenseNet
  - achieves a higher accuracy by 0.6% and 11% lower computational cost than MobileNet
  - PeleeNet is only 66% of the model size of MobileNet

[11] J. Wang *et al*., Pelee: A Real-Time Object Detection System on Mobile Devices, in Proc. ICLR Workshop, 2018

PeleeNet (ICLR Workshop 2018)

- PeleeNet's design
  - Two-Way Dense Layer

    Motivated by GoogLeNet, use a 2-way dense layer to get different scales of receptive fields. One way of the layer uses a small kernel size (3x3), which is good enough to capture small-size objects. The other way of the layer uses two stacked 3x3 convolution to learn visual patterns for large objects

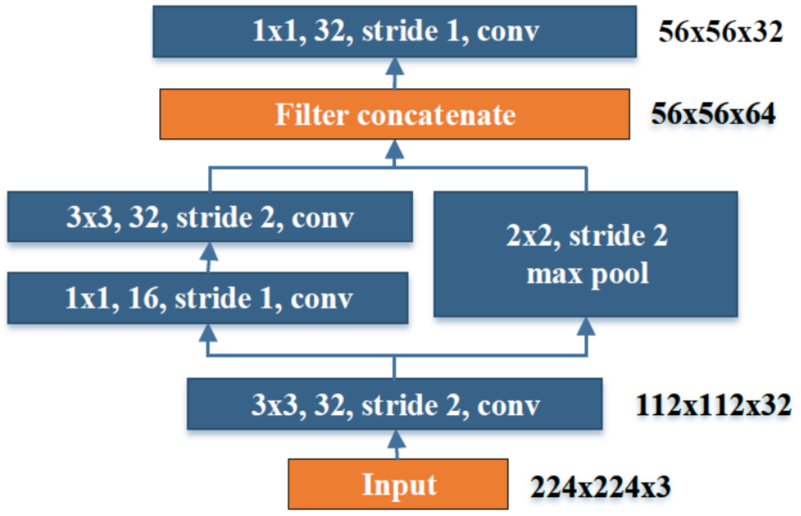


2-way dense layer

PeleeNet (ICLR Workshop 2018)

- PeleeNet's design
  - Two-Way Dense Layer
  - Stem block

  The stem block can effectively improve the feature expression ability without adding computational cost too much - better than other more expensive methods, e.g., increasing channels of the first convolution layer



stem block

PeleeNet (ICLR Workshop 2018)

- PeleeNet’s design
  - Two-Way dense layer
  - Stem block
  - Dynamic number of channels in bottleneck layer
  - Transition layer without compression
  - Composite Function

  To improve actual speed, they use the conventional wisdom of post activation (Convolution - Batch Normalization - Relu) as their composite function instead of pre-activation used in DenseNet

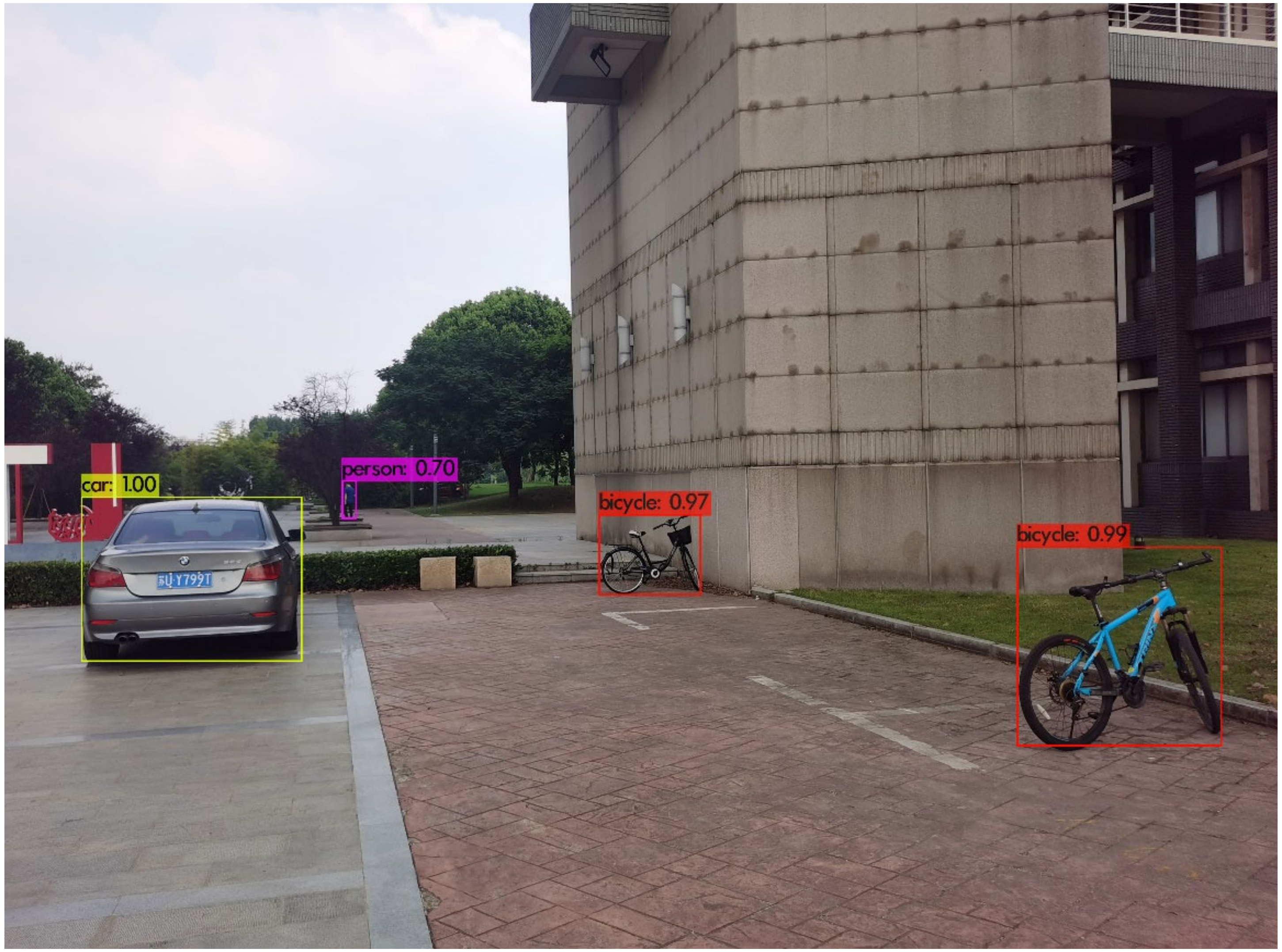
Outline

- Basic concepts
- Linear model
- Neural network
- Convolutional neural network (CNN)
- Modern CNN architectures
- CNN for object detection

Background

- Detection is different from classification
  - An image classification problem is predicting the label of an image among the predefined labels; It assumes that there is **single** object of interest in the image and it covers a significant portion of image
  - Detection is about not only finding the class of object but also localizing the extent of an object in the image; the object can be lying anywhere in the image and can be of any size (scale)

Background



Multiple objects detection

Background

- Traditional methods of detection involved using a block-wise orientation histogram (SIFT or HOG) feature which could not achieve high accuracy in standard datasets such as PASCAL VOC; these methods encode a very low level characteristics of the objects and therefore are not able to distinguish well among the different labels
- Deep learning based methods have become the state-of-the-art in object detection in image; they construct a representation in a hierarchical manner with increasing order of abstraction from lower to higher levels of neural network

Background

- Recent developments of CNN based object detectors
  - R-CNN series
    - R-CNN (CVPR 2014), FastRCNN (ICCV 2015), FasterRCNN (NIPS 2015), MaskRCNN (ICCV 2017)
  - SSD (single-shot multibox detector) (ECCV 2016)
  - Pelee-SSD (ICLR Workshop 2018)
  - YOLO series
    - YOLOv1 (CVPR 2016), YOLOv2 (CVPR 2017), …., YOLOv8 (Jan. 2023)
  - EfficientDet (CVPR 2020)

R-CNN

- Brute-force idea
  - One could perform detection by carrying out a classification on different sub-windows or patches or regions extracted from the image. The patch with high probability will not only the class of that region but also implicitly gives its location too in the image
  - One brute force method is to run classification on all the sub-windows formed by sliding different sized patches (to cover each and every location and scale) all through the image

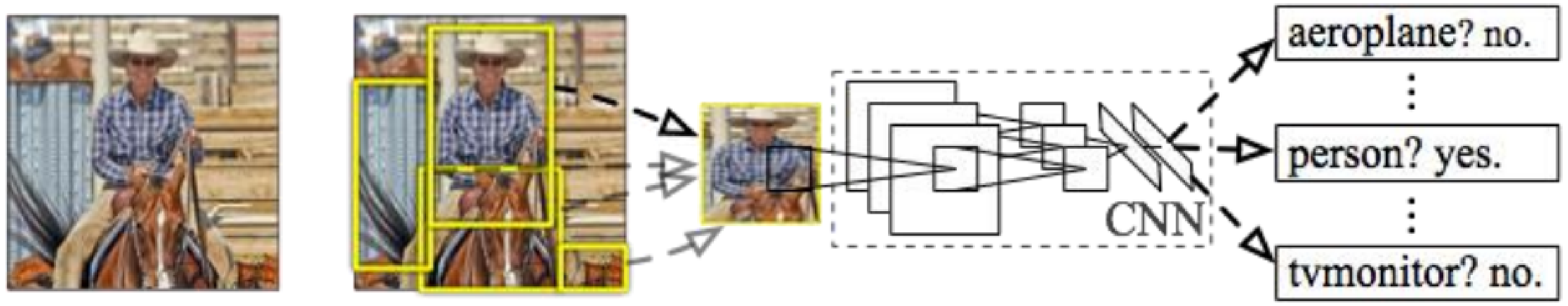
Quite Slow!!!

R-CNN

- R-CNN therefore uses an object proposal algorithm (selective search) in its pipeline which gives out a number (~2000) of TENTATIVE object locations
- These object regions are warped to fixed sized (227X227) regions and are fed to a classification convolutional network which gives the individual probability of the region belonging to background and classes

R-CNN

Region-based Convolution Networks (R-CNNs)



Input image

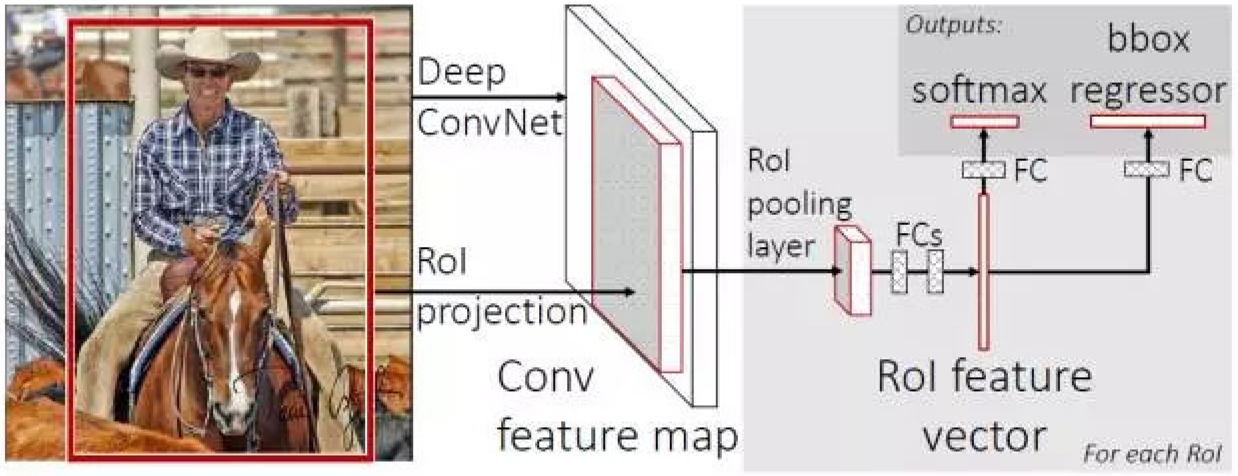
Extract region proposals (~2k / image)
e.g., selective search
[van de Sande, Uijlings et al.]

Compute CNN features on regions

Classify and refine regions

Fast-RCNN

- Compared to RCNN
  - A major change is a single network with two loss branches pertaining to soft-max classification and bounding box regression
  - This multitask objective is a salient feature of Fast-RCNN as it no longer requires training of the network independently for classification and localization

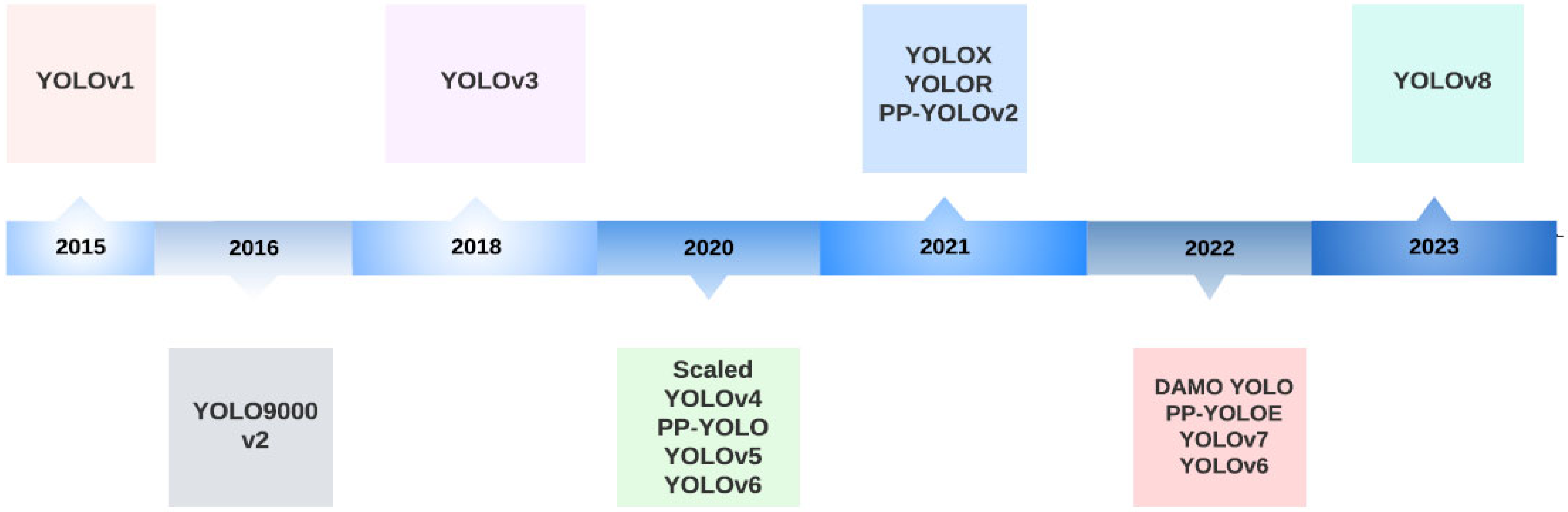


Faster-RCNN

- Compared to Fast-RCNN
  - Faster-RCNN replaces Selective Search with CNN itself for generating the region proposals (called RPN-region proposal network) which gives out tentative regions at almost negligible amount of time

YOLO series

- YOLO (You Only Look Once)
  - The major exceptional idea is that it tackles the object detection problem as a regression problem
  - A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation
  - The whole detection pipeline is a single network
  - It is extremely fast

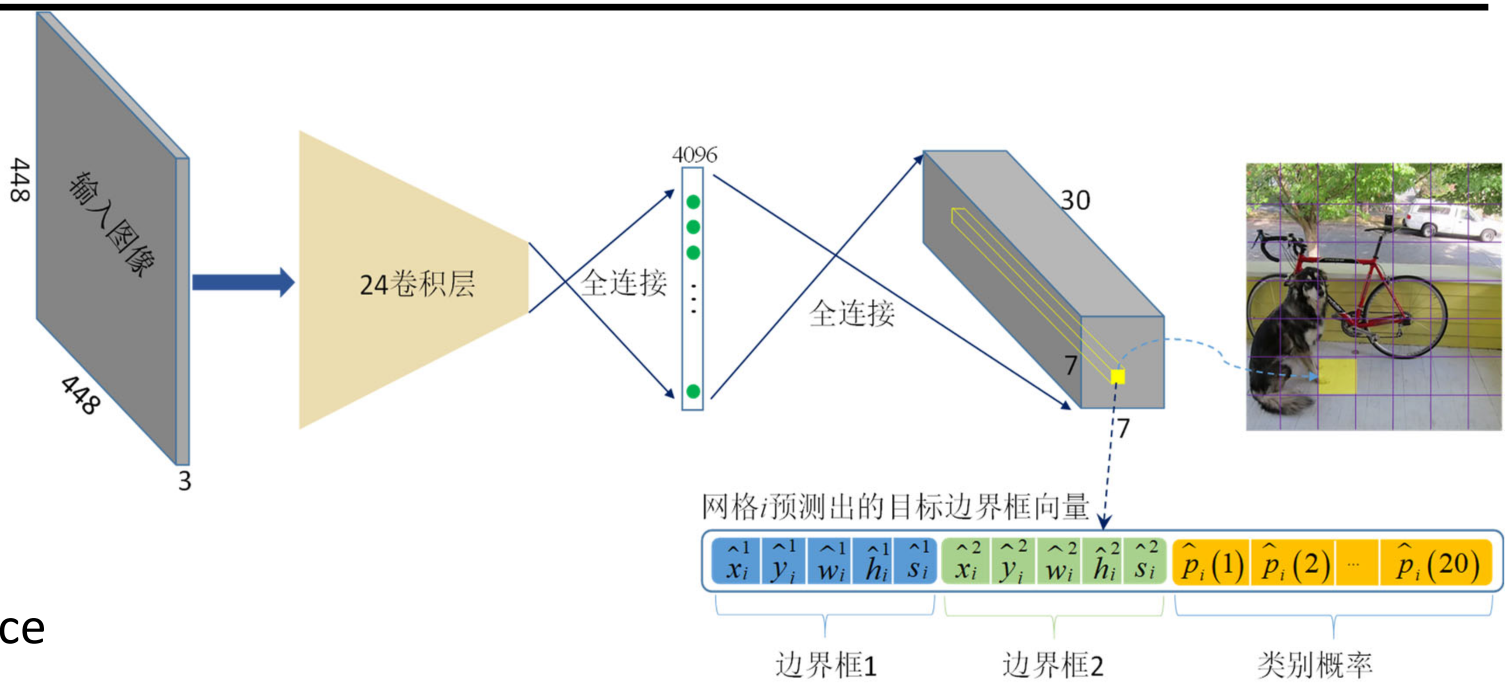


Legends about YOLO

- The original idea of YOLO was proposed by Joseph Redmon in 2016[12]
- Then he improved it as YOLOv2 and YOLOv3 in 2017 and 2018
- However, on Feb. 21, 2020, Redmon announced that he would stop doing research work in CV (*I stopped doing CV research because I saw the impact my work was having. I loved the work but the military applications and privacy concerns eventually became impossible to ignore*), so all the following YOLOs actually do not have "Redmon" in the author lists
- Now from YOLOv1~YOLOv4, the models are trained and inferenced on darknet (a C language based DCNN platform), which is now mainly maintained by a Russian researcher Alexey Bochkovskiy; From YOLOv5, the models are developed by Python and PyTorch
- YOLOv5 was developed by a company, namely Ultralytics, founded by Glenn R. Jocher and on Jan. 2023, they announced YOLOv8

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of CVPR*, pp. 779–788, 2016

YOLOv1



448
输入图像
448
3
24卷积层
全连接
4096
全连接
30
7
7
网格i预测出的目标边界框向量
$\hat{x}_i^1$ $\hat{y}_i^1$ $\hat{w}_i^1$ $\hat{h}_i^1$ $\hat{s}_i^1$ $\hat{x}_i^2$ $\hat{y}_i^2$ $\hat{w}_i^2$ $\hat{h}_i^2$ $\hat{s}_i^2$ $\hat{p}_i(1)$ $\hat{p}_i(2)$ … $\hat{p}_i(20)$
边界框1
边界框2
类别概率

For inference

- The results are derived from the $7\times7\times30$ output matrix
- Conceptually, the input image is divided in to 7×7 cells and each cell predicts a 30*d* vector
  - Each output vector contains two bounding-boxes (whose centers are relative to the cell's top-left corner and the sizes are relative to the size of the image resolution) and classification probabilities of 20 classes (the default model was trained on VOC which has 20 classes)
  - The final confidence score of the 1st bounding-box in cell i is

$$\hat{s}_i^1 \cdot \hat{p}_i(cls) \text{ where } cls = \arg\max_c \left\{ \hat{p}_i(c) :| c \in \{20\ classes\} \right\}$$

- Finally, for each class, perform NMS for the bounding-boxes belonging to that class

YOLOv1

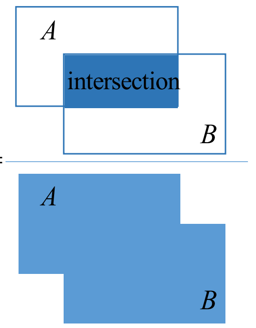
算法 15-1：目标边界框初始集合 $\mathcal{B}$ 的非极大值抑制

输入：目标边界框初始集合$\mathcal{B}$，分类检测置信度集合$\mathcal{S}$，IoU 阈值 τ，分类检测置信度阈值 T

输出：经过 NMS 操作之后的目标边界框集合$\mathcal{F}$

```
1)  $\mathcal{F} \leftarrow \phi$
2)  Filter the bounding boxes: $\mathcal{B} \leftarrow \{b \in \mathcal{B} \mid S(b) \geq T\}$
3)  Sort the bounding boxes in $\mathcal{B}$ by their confidence scores in descending order
4)  while $\mathcal{B} \neq \phi$ do
5)      Select the bounding box $b$ from $\mathcal{B}$ with the highest confidence score
6)      Add $b$ to the set of final bounding boxes $\mathcal{F}$: $\mathcal{F} \leftarrow \mathcal{F} \cup \{b\}$
7)      Remove $b$ from $\mathcal{B}$: $\mathcal{B} \leftarrow \mathcal{B} - \{b\}$
8)      for every remaining bounding box $r$ in $\mathcal{B}$ do
9)          Calculate the IoU between $b$ and $r$: $iou \leftarrow IoU(b, r)$
10)         if $iou \geq \tau$
11)             Remove $r$ from $\mathcal{B}$: $\mathcal{B} \leftarrow \mathcal{B} - \{r\}$
12)         end if
13)     end for
14) end while
```

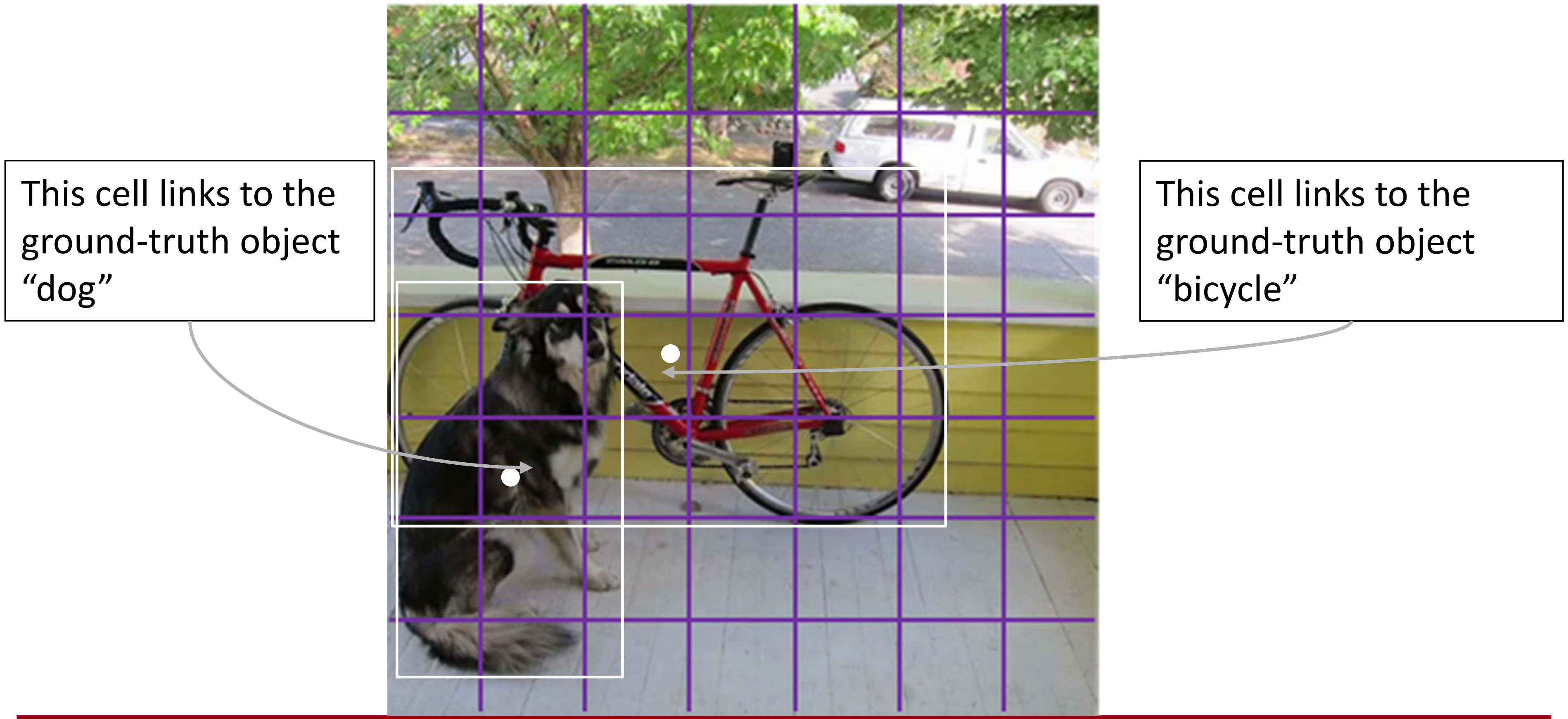
$$IoU(A,B) = \frac{\textit{intersection area between } A \textit{ and } B}{\textit{union area of } A \textit{ and } B} =$$



YOLOv1

For training

- Conceptually, the input image is divided in to 7×7 cells
- For cell i, if one ground-truth object's center falls in i, we say cell i links to a ground-truth object and is responsible for the loss computation related to that ground-truth object


This cell links to the ground-truth object "dog"
This cell links to the ground-truth object "bicycle"

YOLOv1

For training

- Conceptually, the input image is divided in to 7×7 cells
- For cell i, if one ground-truth object's center falls in i, we say cell i links to a ground-truth object and is responsible for the loss computation related to that ground-truth object
- For a cell linking to a ground-truth object, though each time two bounding-boxes are predicted, only the one with the larger IoU with the ground-truth will participate in the loss calculation raised by this ground-truth object
- The loss is composed of three kinds of terms, related to position, objectiveness, and classification

cell i links to a ground-truth object and box j is responsible for predicting it

position

$$\lambda_{coord}\sum_{i=1}^{49}\sum_{j=1}^{2}\mathbf{1}_{ij}^{obj}\left[\left(x_i-\hat{x}_i\right)^2+\left(y_i-\hat{y}_i\right)^2+\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)\right]$$

objectiveness

$$\left\{\begin{array}{l}+\sum_{i=1}^{49}\sum_{j=1}^{2}\mathbf{1}_{ij}^{obj}\left(s_i-\hat{s}_i\right)^2\\ +\lambda_{noobj}\sum_{i=1}^{49}\sum_{j=1}^{2}\mathbf{1}_{ij}^{noobj}\left(0-\hat{s}_i^{j}\right)^2\end{array}\right.$$

The ground-truth value for the objectiveness is the IoU of this predicted bb with the ground-truth

box j of cell i is not responsible for predicting any ground-truth

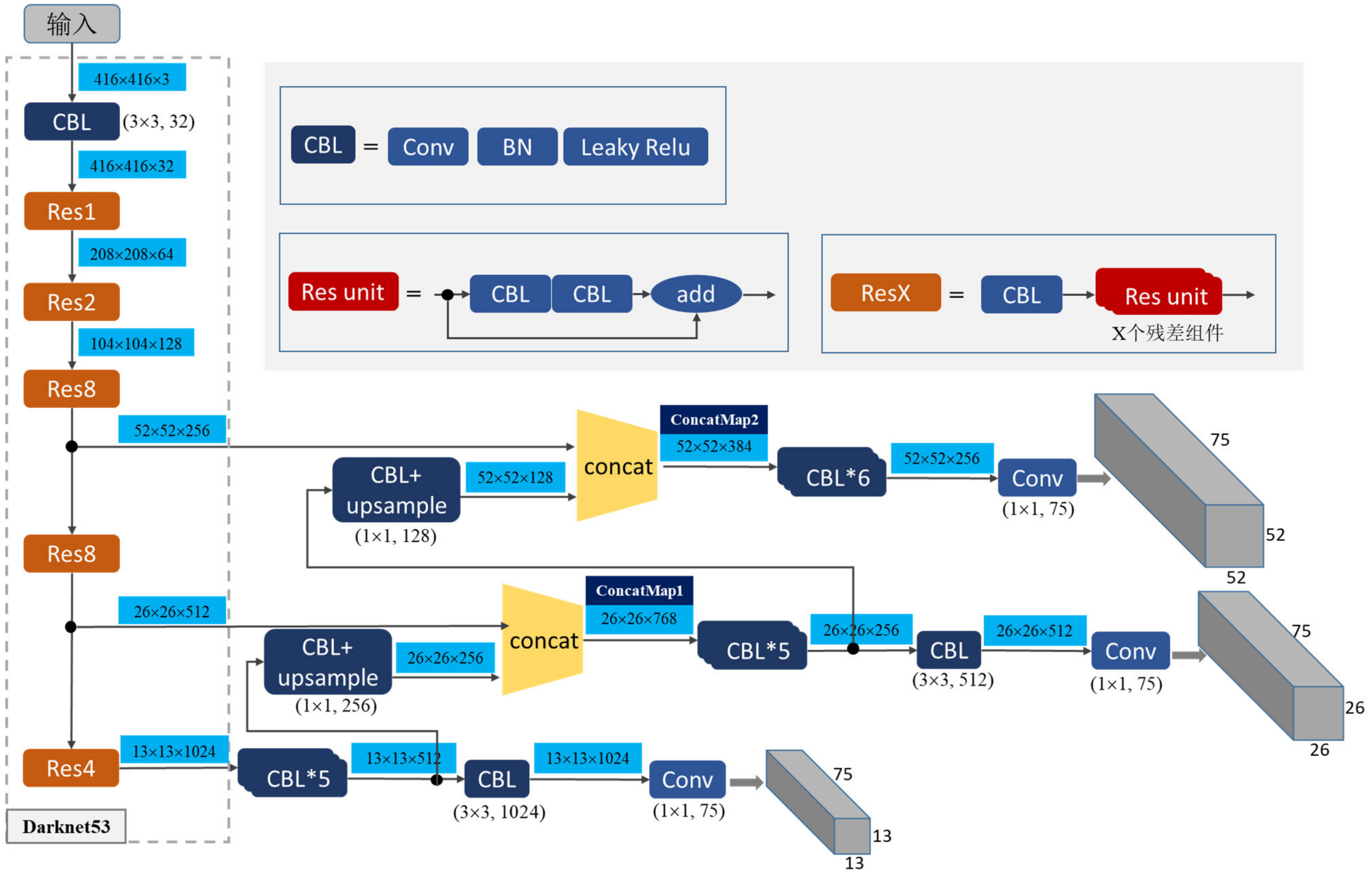
classification

$$+\sum_{i=1}^{49}\left(\mathbf{1}_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2\right)$$

cell i links to a ground-truth

The values with ^ are predicted

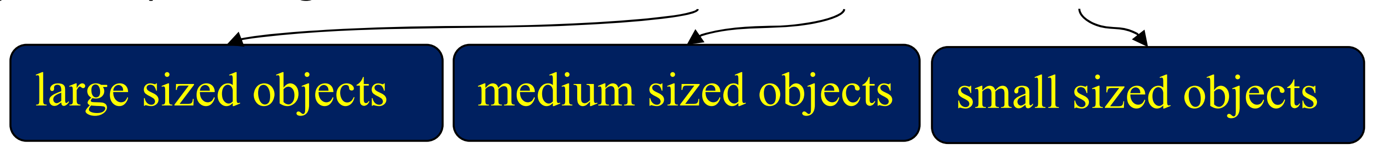
YOLOv3



输入
416×416×3
CBL
(3×3, 32)
416×416×32
Res1
208×208×64
Res2
104×104×128
Res8
52×52×256
Res8
26×26×512
Res4
13×13×1024
Darknet53
CBL = Conv BN Leaky Relu
Res unit = CBL CBL add
ResX = CBL Res unit
X个残差组件
CBL+ upsample
52×52×128
(1×1, 128)
concat
ConcatMap2
52×52×384
CBL*6
52×52×256
Conv
(1×1, 75)
75
52
52
CBL+ upsample
26×26×256
(1×1, 256)
concat
ConcatMap1
26×26×768
CBL*5
26×26×256
CBL
(3×3, 512)
26×26×512
Conv
(1×1, 75)
75
26
26
CBL*5
13×13×512
CBL
(3×3, 1024)
13×13×1024
Conv
(1×1, 75)
75
13
13

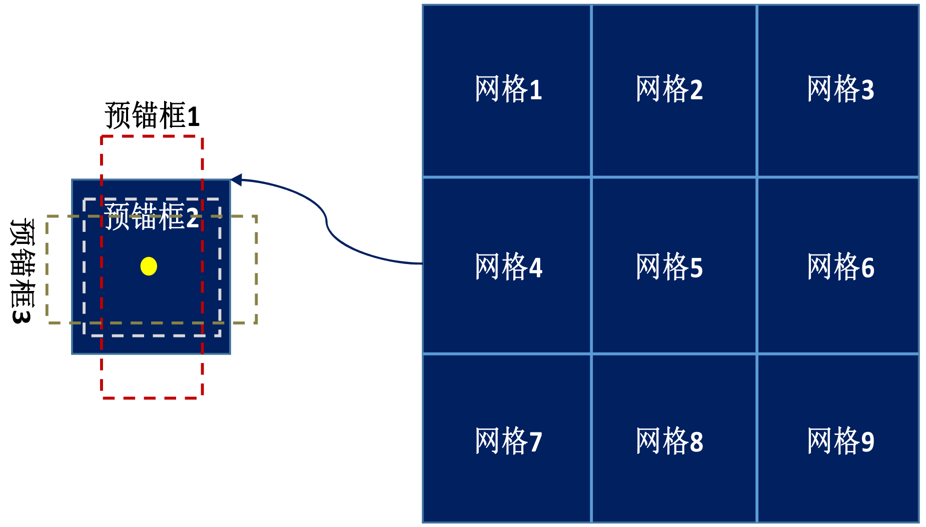
YOLOv3

- It incorporates a multi-scale detection scheme that can better detect objects with various sizes
  - It outputs three matrices with different resolutions, 13×13, 26×26 and 52×52
  - Conceptually, the input image is divided into 13×13, 26×26, and 52×52 cells



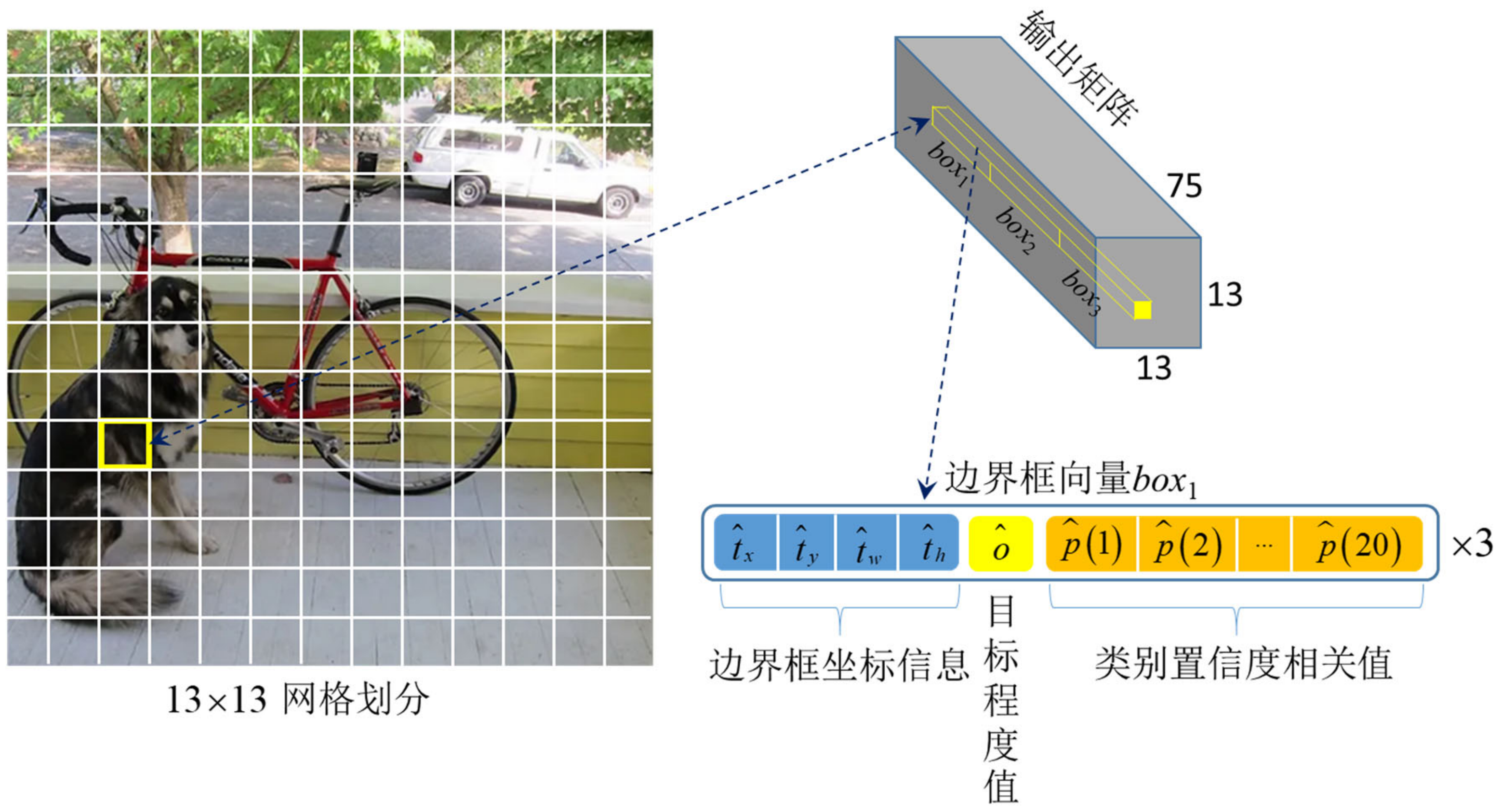
YOLOv3

- It incorporates a multi-scale detection scheme that can better detect objects with various sizes
- It uses the anchor-box mechanism to more accurately predict objects with various aspect ratios
  - An anchor-box is a rectangle with pre-defined size
  - Using anchor-boxes, the predicted box's size related values are relative to the corresponding anchor-box
  - In YOLOv3, three anchor-boxes are used for each cell
  - For the three different output resolutions, three different groups of anchor-boxes are used



output resolution	Anchor-boxes
13×13	(116, 90), (156, 198), (373, 326)
26×26	(30, 61), (62, 45), (59, 119)
52×52	(10, 13), (16, 30), (33, 23)

YOLOv3

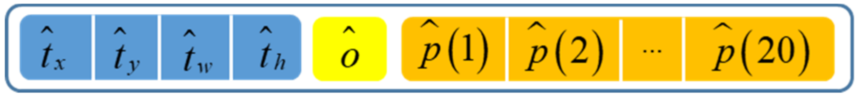


The detection results are derived from the three output matrices with different resolutions

YOLOv3

For inference

- The results are derived from the three output matrices
- Take the feature map with the resolution 13×13 as an example
  - ✓ Each cell predicts a 75d vector, which is actually composed of three predicted bounding-boxes, each for one anchor-box
  - ✓ For cell i, suppose one of its predicted boxes is



This bounding-box's center is $\begin{cases} b_x = \sigma(\hat{t}_x) + c_x \\ b_y = \sigma(\hat{t}_y) + c_y \end{cases}$, where (c_x, c_y) is cell i's position (values are 0~12)

This bounding-box's size is $\begin{cases} b_w = p_w e^{\hat{t}_w} \\ b_h = p_h e^{\hat{t}_h} \end{cases}$, where (p_w, p_h) is the associated anchor-box's size

This bounding-box's class-dependent confidence score is $\sigma(\hat{o}) \cdot \sigma(\hat{p}(j))$

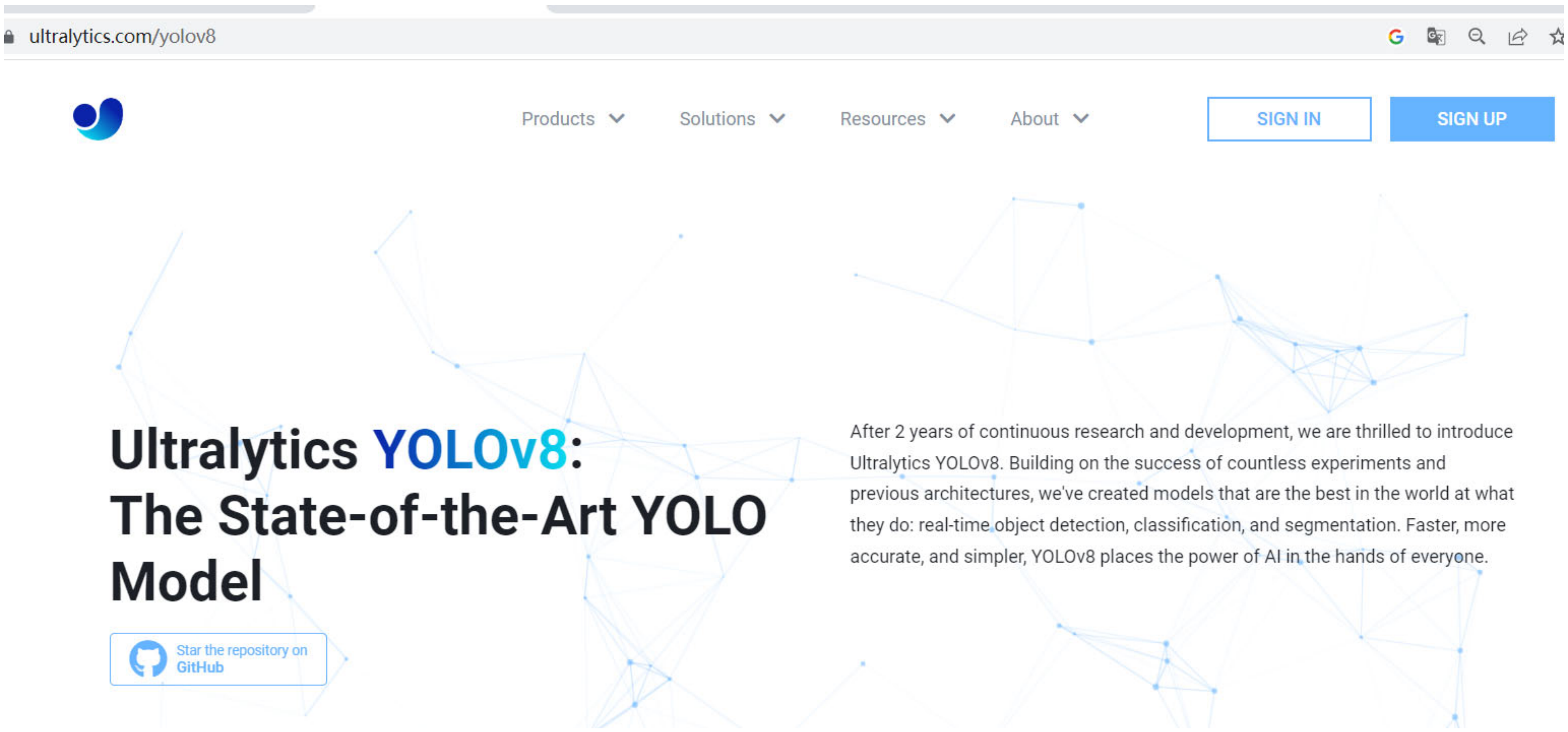
- Finally, for each class, perform NMS for the bounding-boxes based on their class-dependent confidence scores

YOLOv3

For training

- For the ground-truth object $\mathbf{b}$, it is assigned to an anchor-box according to the IoUs between $\mathbf{b}$ and all the anchor-boxes
- For every anchor-box, it falls in one of the following three cases,
  - It is linked to a ground-truth object; for such an anchor-box, its predicted bounding box will participate in the loss calculation related to position, classification, and objectness
  - It is not linked to any ground-truth object, but the IoU between it and some ground-truth object is over 0.5, then this anchor-box will be “ignored”; such an anchor-box will not participate in any loss term calculation (since in this case, the supervisory information provided by this anchor-box is ambiguous)
  - Others (neither linked to any ground-truth object nor be ignored); for such an anchor-box, its predicted bounding-box only participates in the loss term calculation related to objectness (for this case, the ground-truth for objectness is 0)
- The final loss is a weighted sum of the loss terms related to position, classification, and objectness

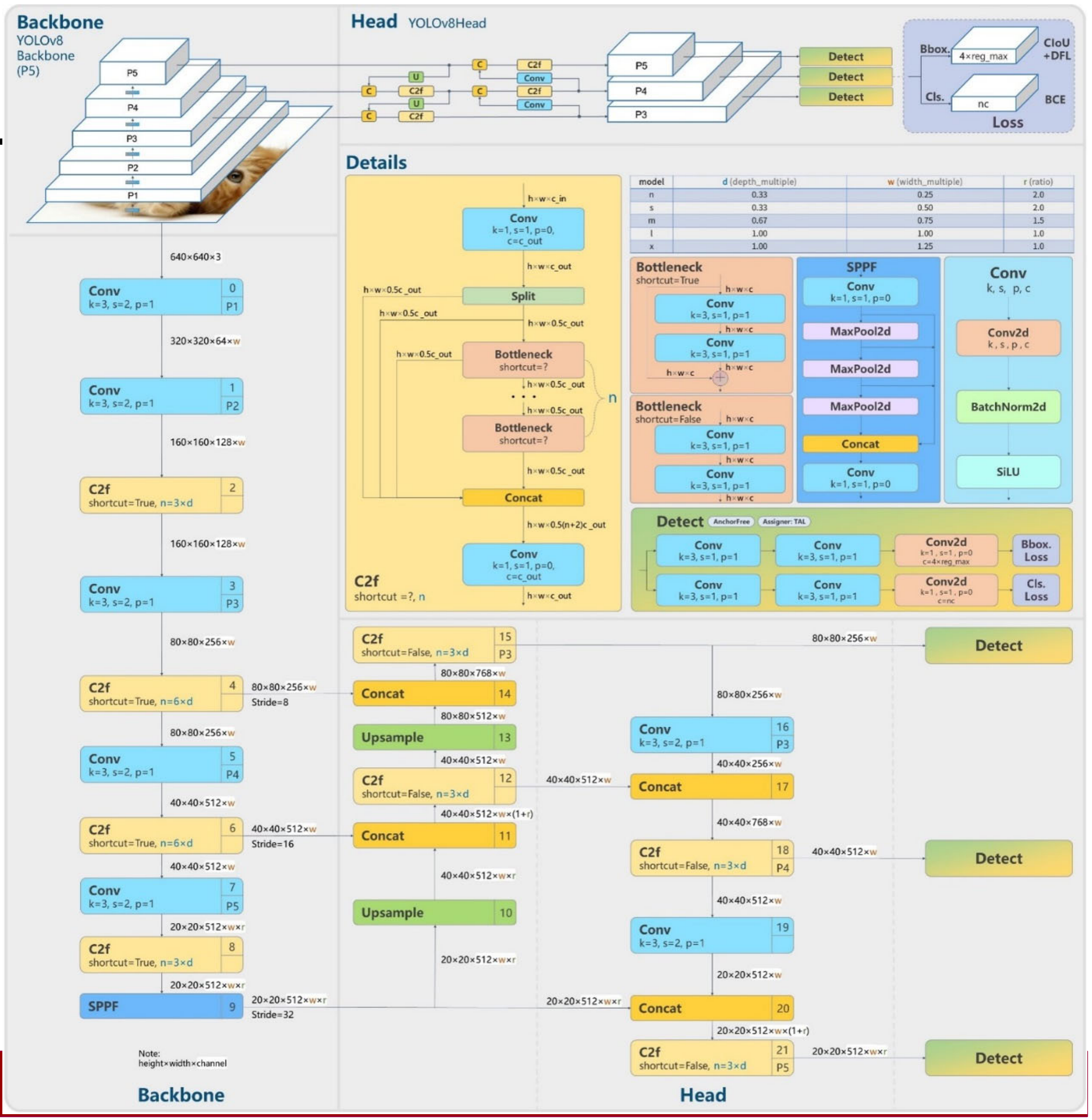
YOLOv8



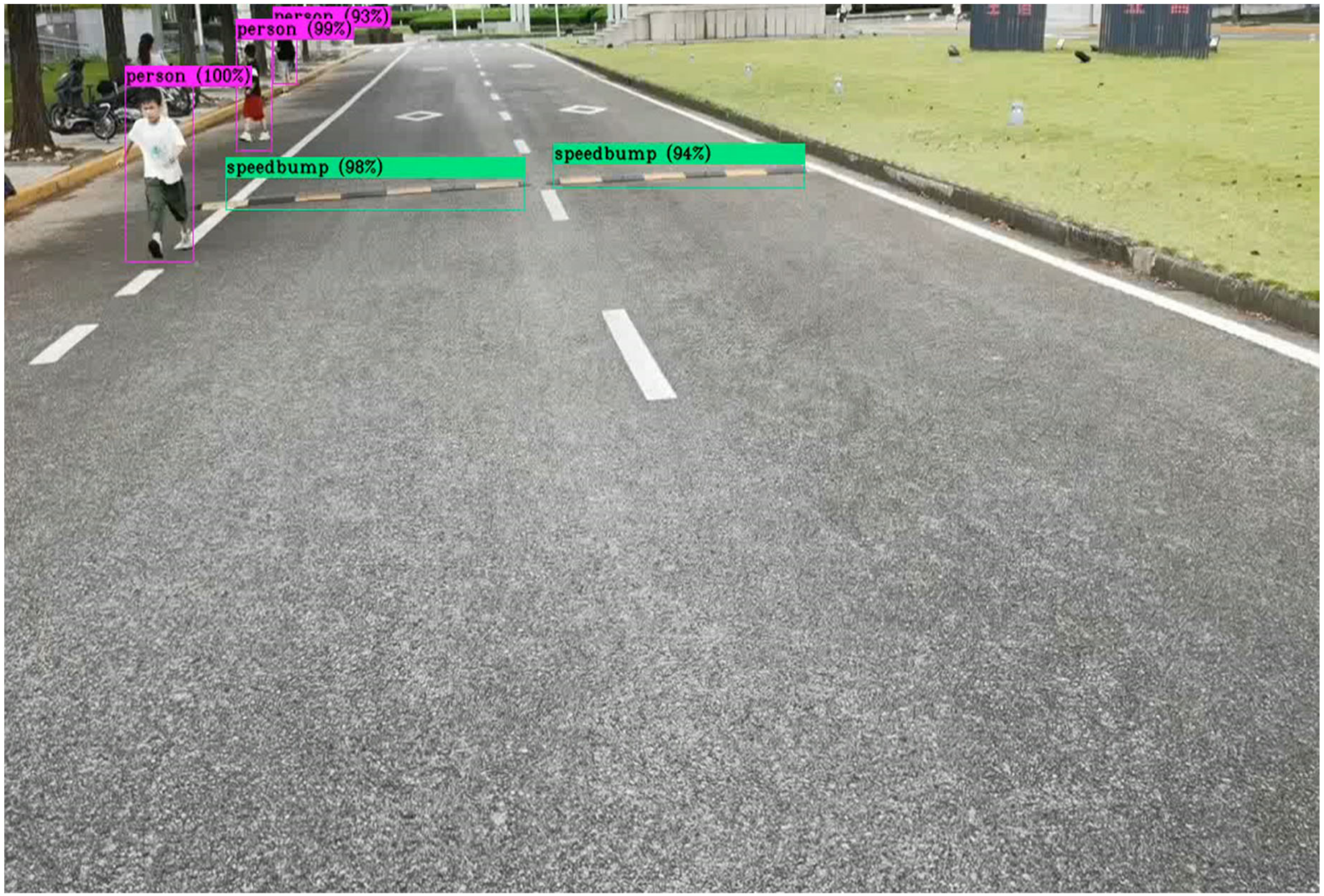
Ultralytics YOLOv8:
The State-of-the-Art YOLO Model

After 2 years of continuous research and development, we are thrilled to introduce Ultralytics YOLOv8. Building on the success of countless experiments and previous architectures, we've created models that are the best in the world at what they do: real-time object detection, classification, and segmentation. Faster, more accurate, and simpler, YOLOv8 places the power of AI in the hands of everyone.

YOLOv8



model	d (depth_multiple)	w (width_multiple)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0


person (93%)
person (99%)
person (100%)
speedbump (98%)
speedbump (94%)

Q+A