Robotic Navigation and Exploration

Week 9: Deep Learning for Computer Vision

Min-Chun Hu anitahu@cs.nthu.edu.tw
CS, NTHU / CSIE, NCKU

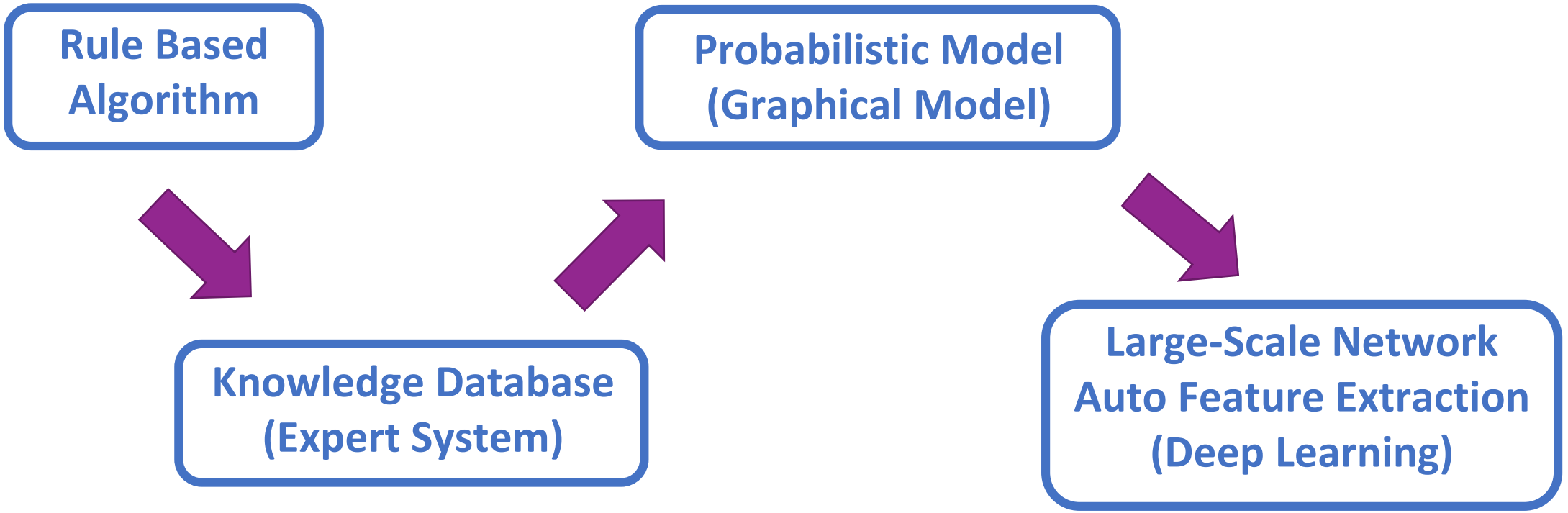
What is learning?

What is Learning ?

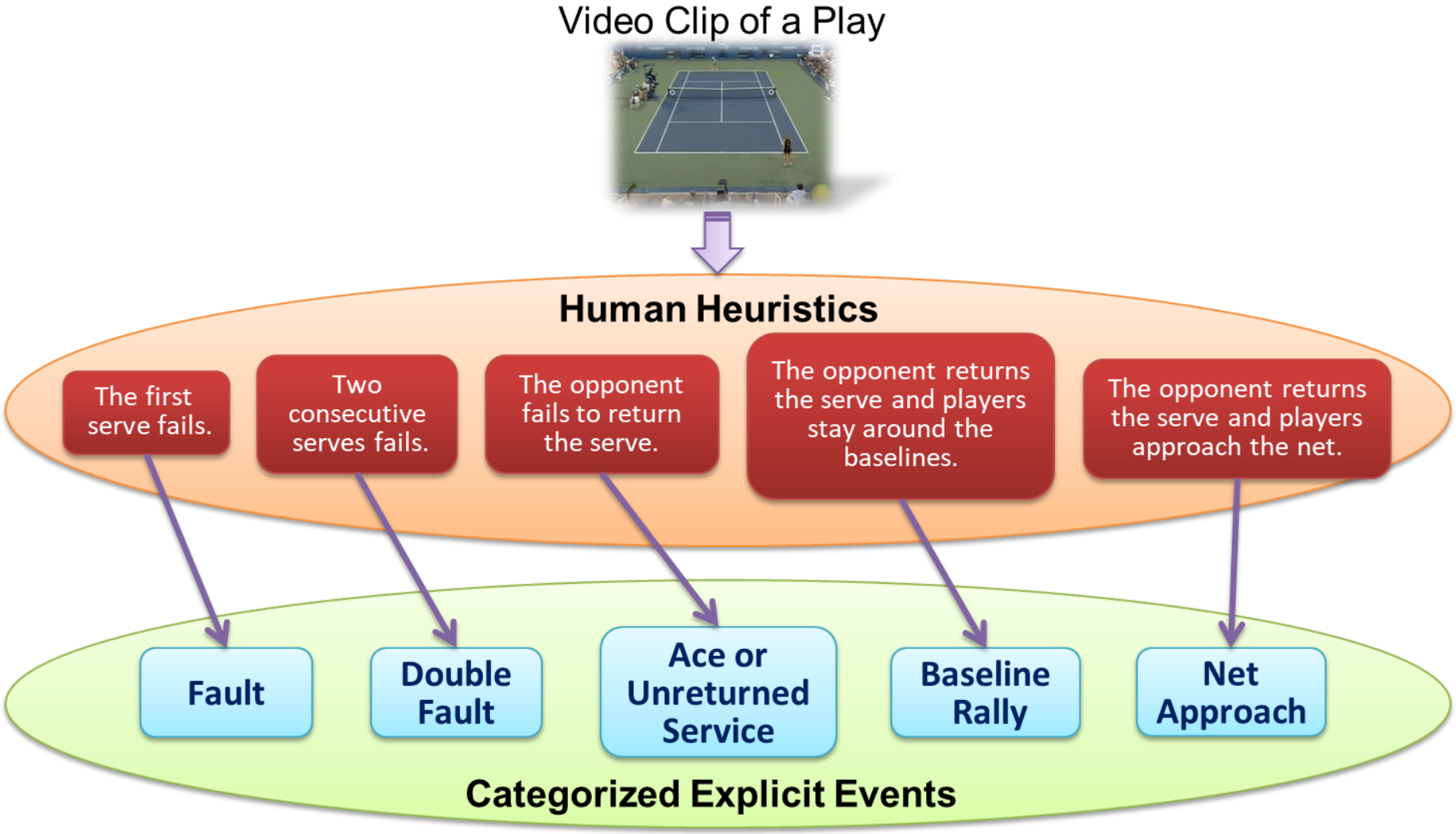
- Find *"simple"* ways to explain the complex data

- Compress data with some *"pattern"*
  - generated based on a *probability distribution*

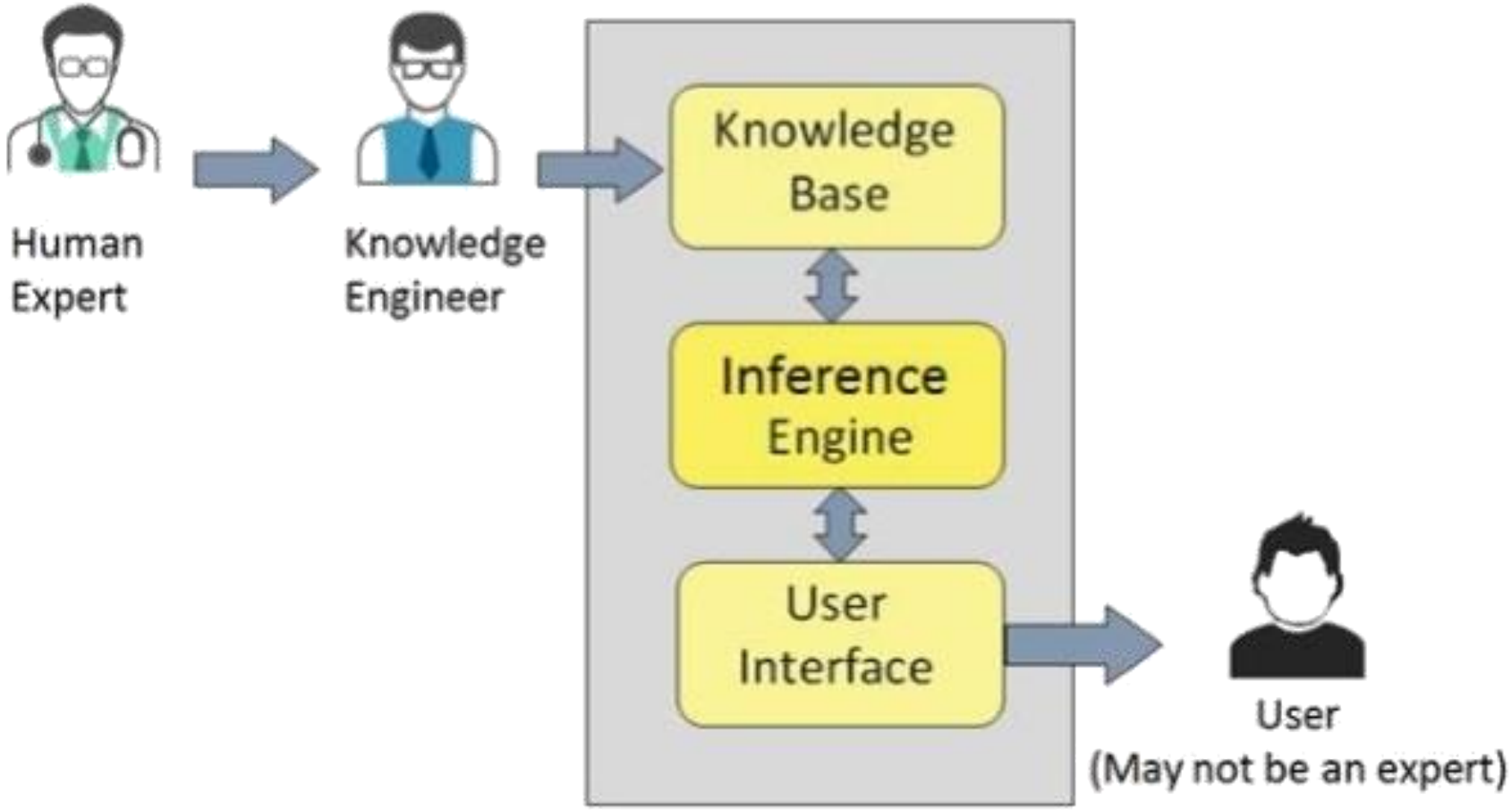
Evolution of Machine Learning



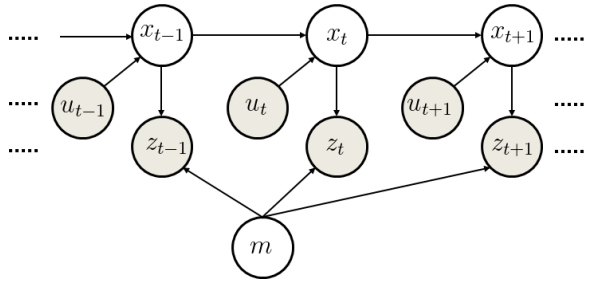
Rule Based Algorithm



Expert System



Probabilistic Graphical Model



SLAM

Body Pose Estimation

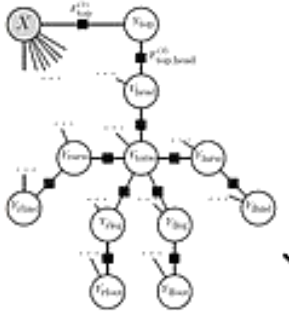
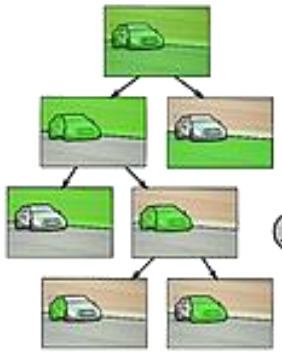
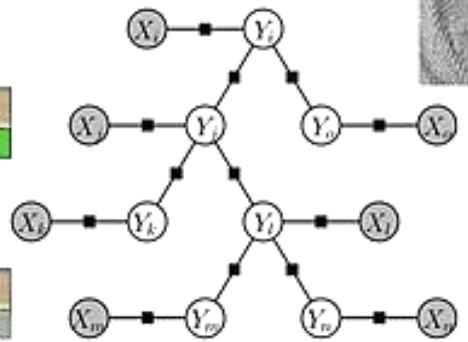


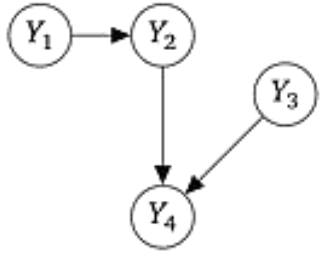
Image Denoising

Semantic Image Segmentation

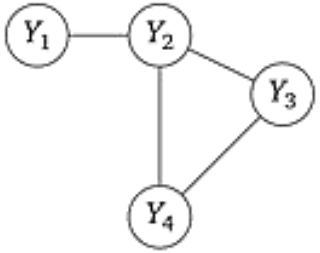




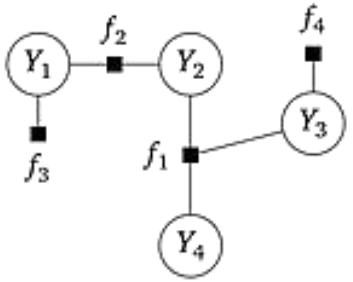
Graphical Models



(a) Bayesian Network

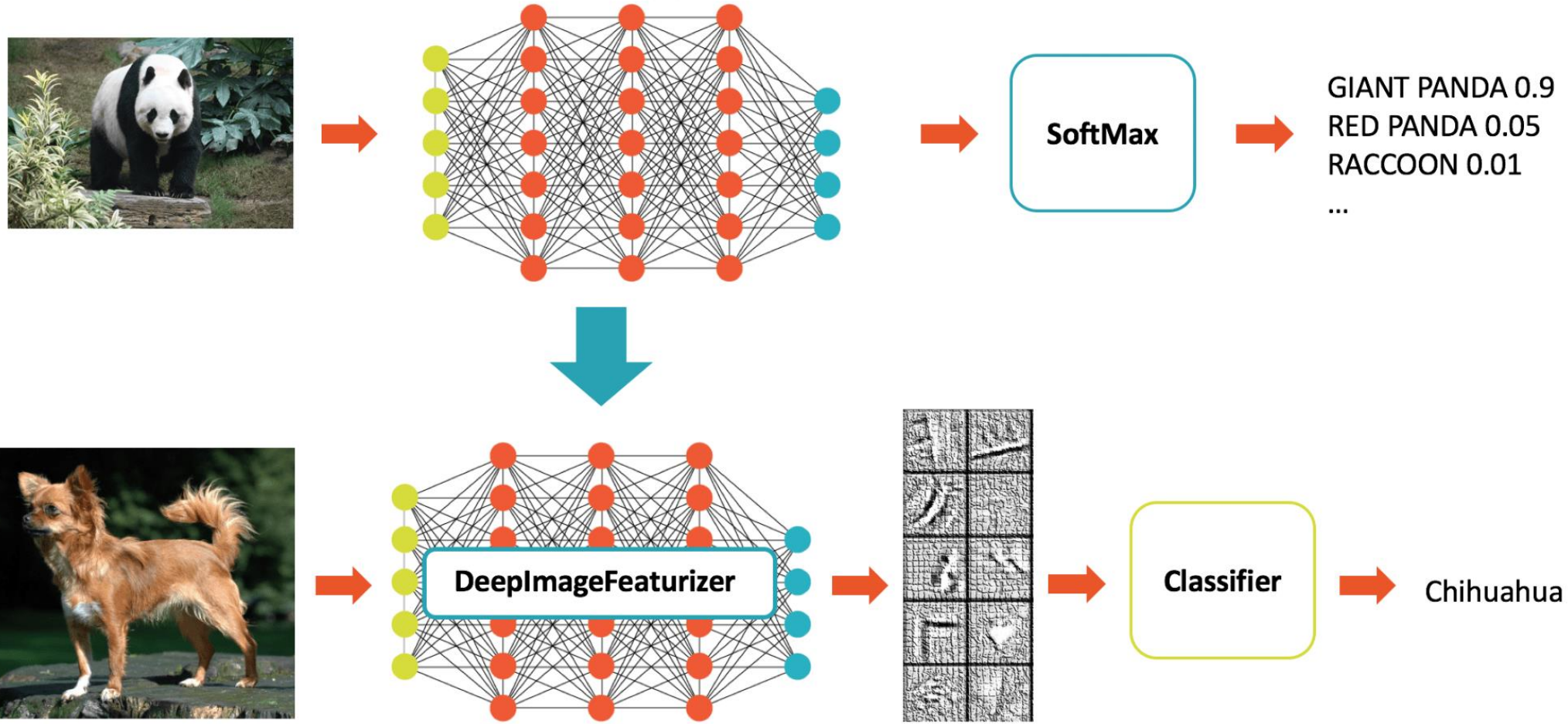


(b) Markov Random Field

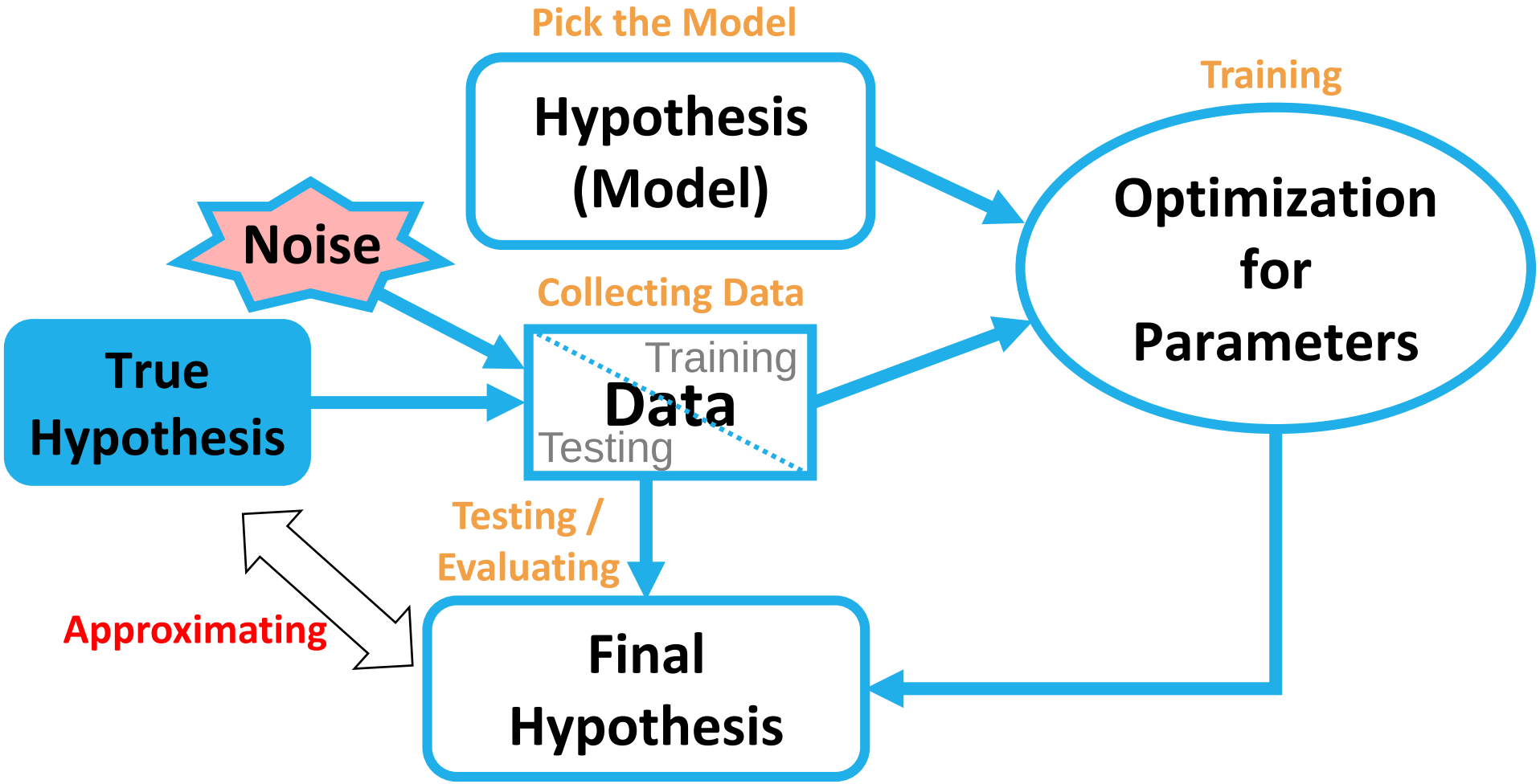


(c) Factor Graph

Deep Learning



Machine Learning Process



A Learning Example

A large set of data/feature vectors

x_1

x_2

.

.

.

x_N

a machine learning algorithm

y(x)

The target vectors

t_1 Cat (1)

t_2 Dog (2)

.

.

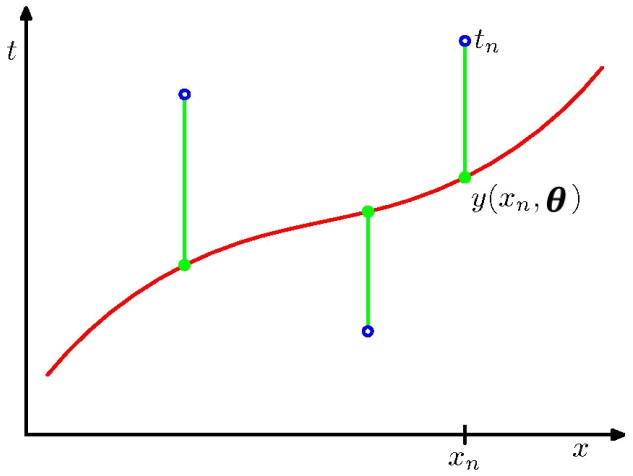
.

t_N Dog (2)

What is a well machine learning algorithm ?

$$\tilde{E}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \boldsymbol{\theta}) - t_n\}^2 \quad + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$$

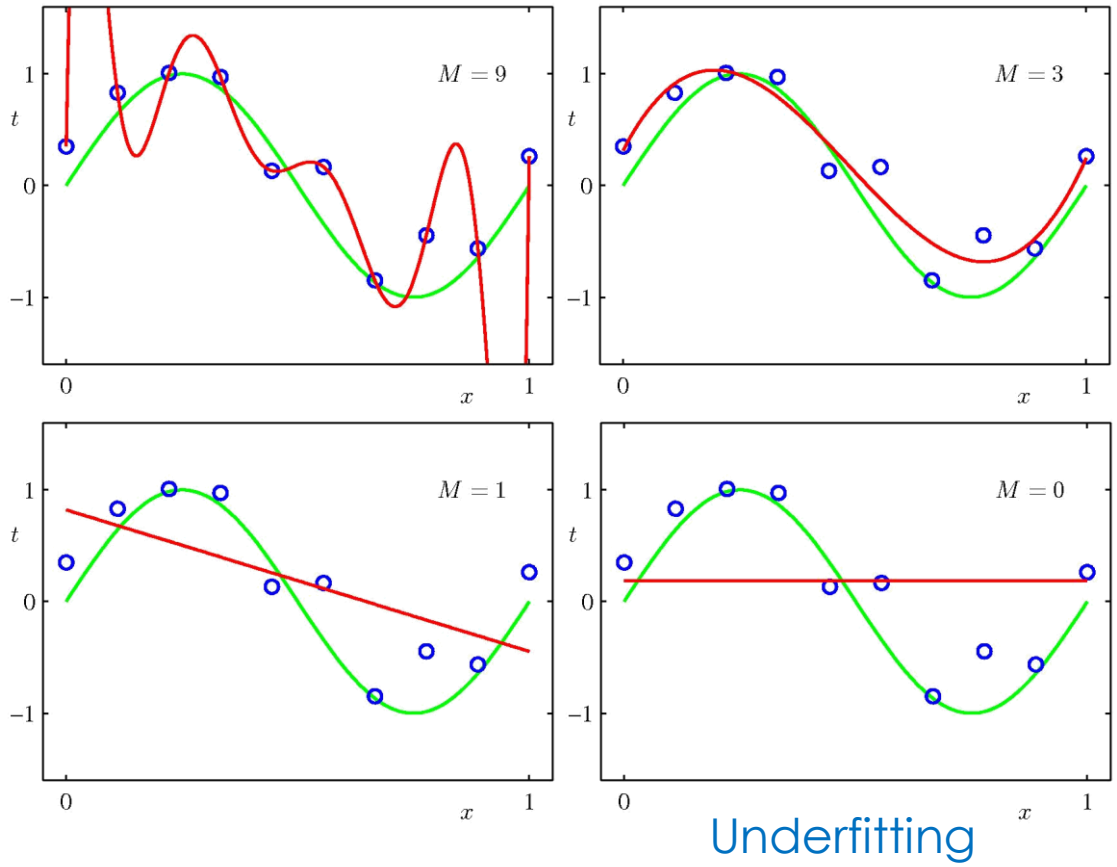
Overfitting & Underfitting



$$\tilde{E}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \boldsymbol{\theta}) - t_n\}^2$$

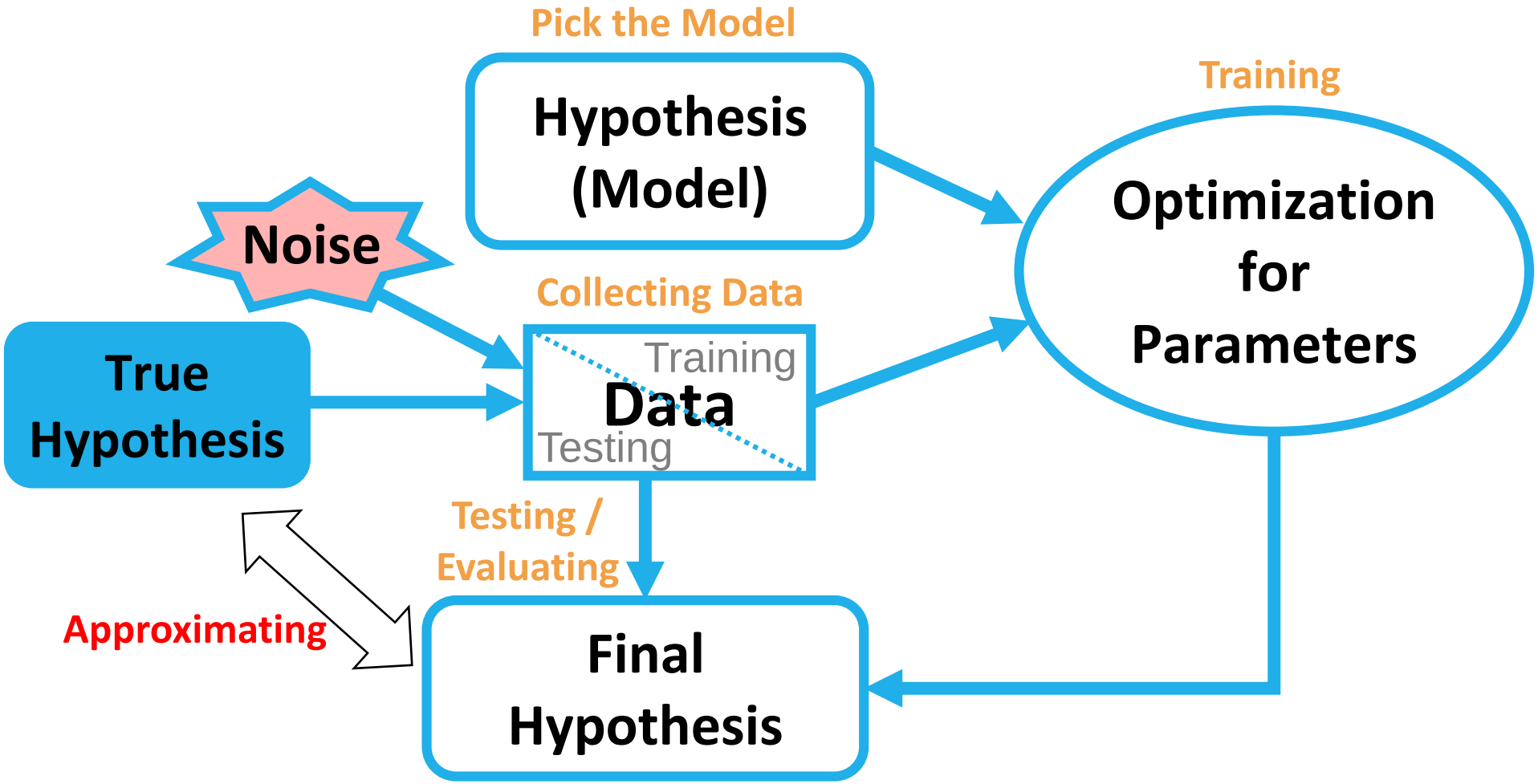
$$+\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$$

Overfitting



Underfitting

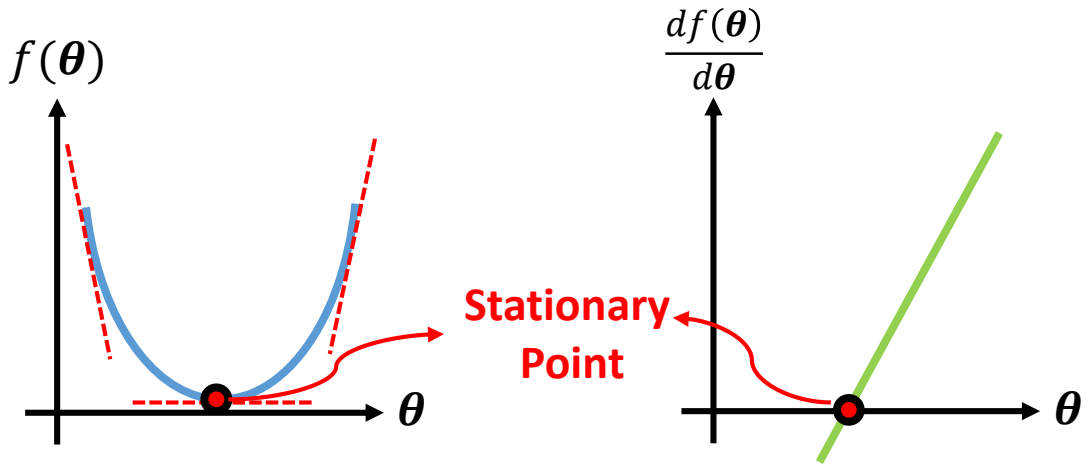
Machine Learning Process



Optimization Methods

- Given an objective function $f(\boldsymbol{\theta})$, find the optimal $\boldsymbol{\theta}_*$ to minimize or maximize the objective function.

$$\min_{\boldsymbol{\theta} \in R^N} f(\boldsymbol{\theta})$$

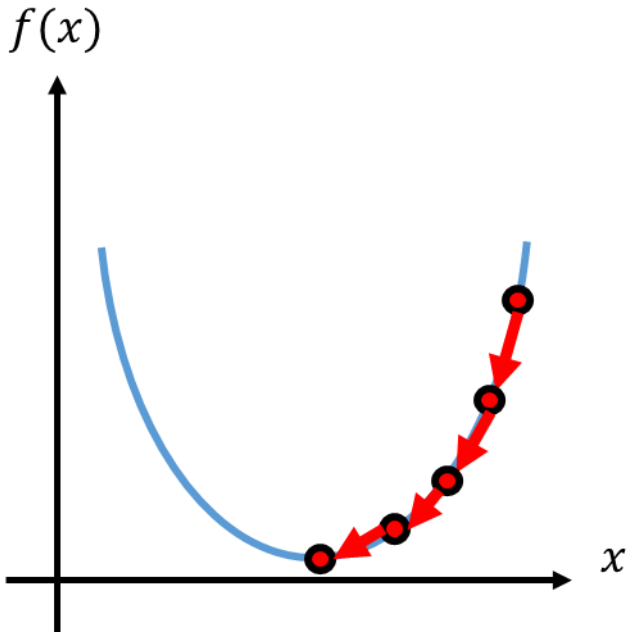


✓ Find the stationary point (first-order differential)

$$\nabla f(\boldsymbol{\theta}) = \mathbf{0}$$

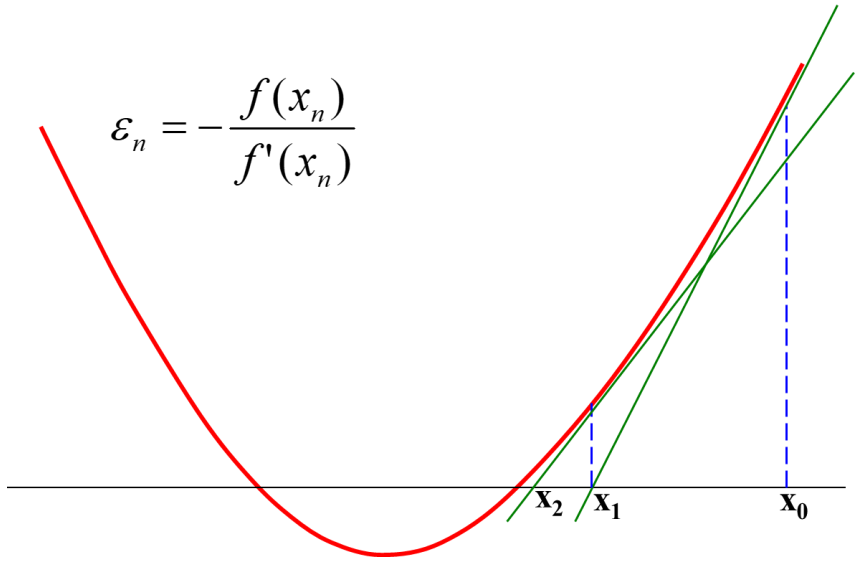
Optimization Methods

- Searching method:



First-order (Gradient Descent)

Fixed Step Length



Second-order (Newton's method)

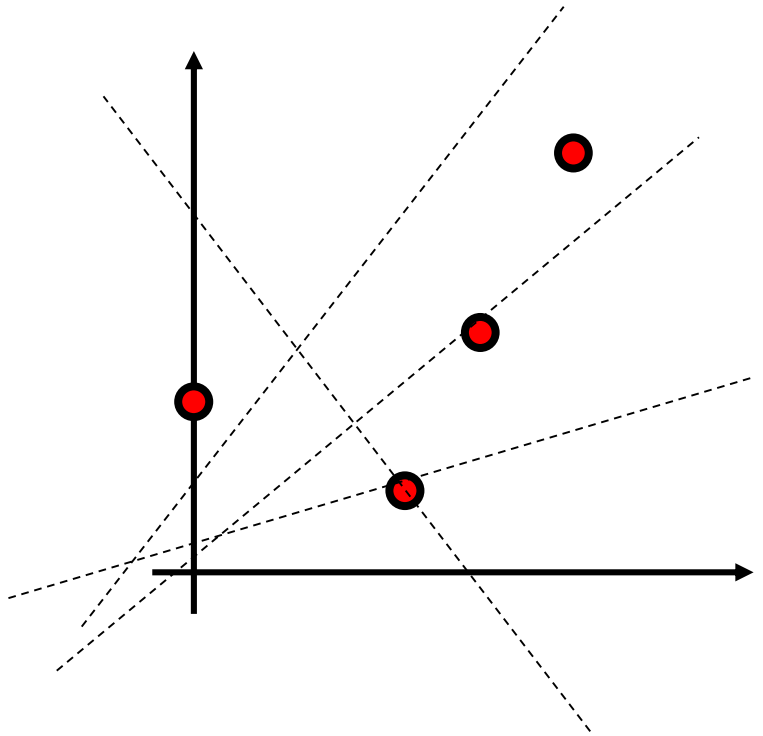
Dynamic Step Length

Example : Linear Regression

- Example: Linear Regression
  - Collecting data
  - Pick a model

$$f(x) = w_1 x + w_0$$

  - Find the suitable parameters w_0 and w_1



Example : Linear Regression

- Optimization

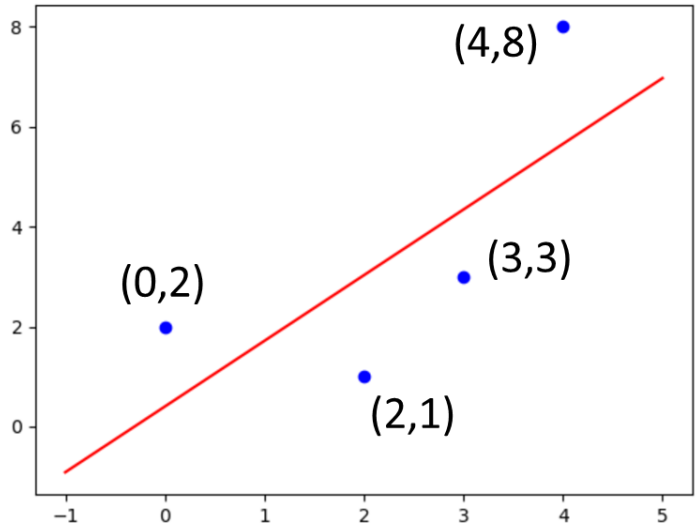
$$L(x,y) = \sum_i \frac{1}{2}(y_i - w_1 x_i - w_0)^2 \; (object\ function)$$

$$\frac{\partial L}{\partial w_0} = \sum_i (y_i - w_1 x_i - w_0)(-x_i) = 0 \rightarrow \sum_i (-x_i y_i + w_1 x_i^2 + w_0 x_i) = 0$$

$$\frac{\partial L}{\partial w_1} = \sum_i (y_i - w_1 x_i - w_0)(-1) = 0 \rightarrow \sum_i (-y_i + w_1 x_i + w_0) = 0$$

Example : Linear Regression

$$\begin{cases} \sum_i (-x_i y_i + w_1 x_i^2 + w_0 x_i) = 0 \\ \sum_i (-y_i + w_1 x_i + w_0) = 0 \end{cases}$$



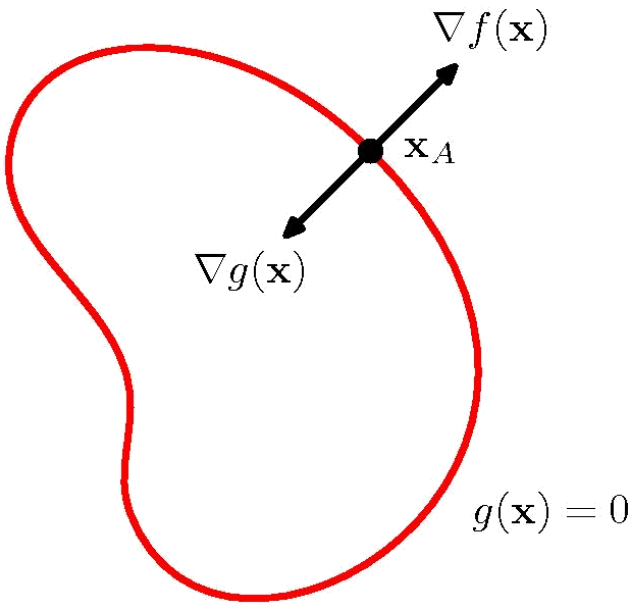
$$\begin{cases} (0) + (-2 + 4w_1 + 2w_0) + (-9 + 9w_1 + 3w_0) + (-32 + 16w_1 + 4w_0) = 0 \\ (-2 + w_0) + (-1 + 2w_1 + w_0) + (-3 + 3w_1 + w_0) + (-8 + 4w_1 + w_0) = 0 \end{cases}$$

$$\begin{cases} 29w_1 + 9w_0 - 43 = 0 \\ 9w_1 + 4w_0 - 14 = 0 \end{cases} \qquad w_1 = \frac{46}{35},\ w_0 = \frac{14}{35}$$

Lagrange Multiplier

- Given an object function $f(\boldsymbol{\theta})$ with constraint $g(\boldsymbol{\theta}) = 0$, find the optimal $\boldsymbol{\theta}_*$ to minimize/maximize the objective function.

$$\max_{\boldsymbol{\theta} \in R^N} f(\boldsymbol{\theta}) \text{, subject to } g(\boldsymbol{\theta}) = 0$$



Lagrange Multiplier

$$f(\boldsymbol{\theta}) + \lambda g(\boldsymbol{\theta}) = 0$$

Naive Optimization

- Finding the maximum of a function $f(x_1, x_2)$ subject to a constraint relating x_1 and x_2 : $g(x_1, x_2) = 0$
- Use $g(x_1, x_2) = 0$ to express x_2 as a function of x_1:

$$x_2 = h(x_1)$$?

- Substitute x_2 into $f(x_1, x_2)$:

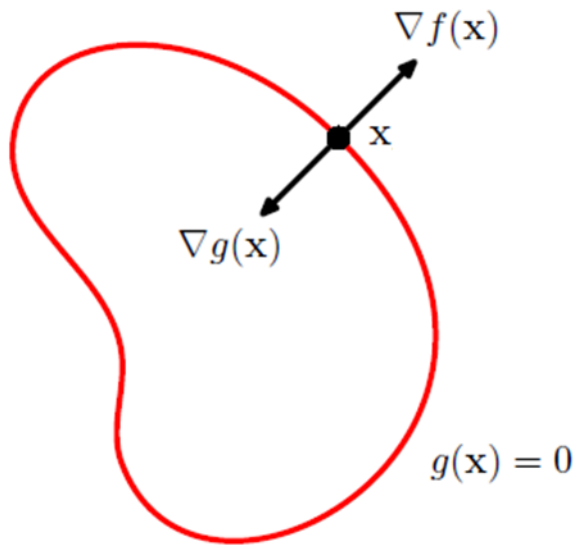
$$f(x_1, x_2) = f(x_1, h(x_1))$$

- The maximum with respect to x_1 could be obtained by differentiation.
  - x_1* is the stationary value and corresponding
  - x_2* is obtained by $h(x_1*)$

Lagrange Multiplier

Maximize $f(\mathbf{x})$

constrained on $g(\mathbf{x}) = 0$



$\mathbf{x} + \boldsymbol{\epsilon}$: a nearby point that also lie on the constraint surface

λ : Lagrange Multiplier

By Taylor expansion around $\mathbf{x}$:

$g(\mathbf{x} + \boldsymbol{\epsilon}) \simeq g(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla g(\mathbf{x})$

$\because$ both $\mathbf{x} + \boldsymbol{\epsilon}$ and $\mathbf{x}$ lie on the constraint surface

$\therefore g(\mathbf{x} + \boldsymbol{\epsilon}) = g(\mathbf{x}) = 0$

$\rightarrow \boldsymbol{\epsilon}^T \nabla g(\mathbf{x}) \simeq 0$

$\rightarrow \boldsymbol{\epsilon} \perp \nabla g(\mathbf{x})$

$\because \boldsymbol{\epsilon} \parallel g(\mathbf{x}) = 0$

$\therefore \nabla g(\mathbf{x})$ is normal to the surface $g(\mathbf{x}) = 0$

Taylor expansion of $f(\mathbf{x})$:

$f(\mathbf{x} + \boldsymbol{\epsilon}) \simeq f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x})$

If $f(\mathbf{x})$ is the maximum, Why?

$\nabla f(\mathbf{x})$ is also normal to the surface $g(\mathbf{x}) = 0$

$\therefore \nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are parallel or anti-parallel

$\rightarrow \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = \mathbf{0}$, where $\lambda \neq \mathbf{0}$

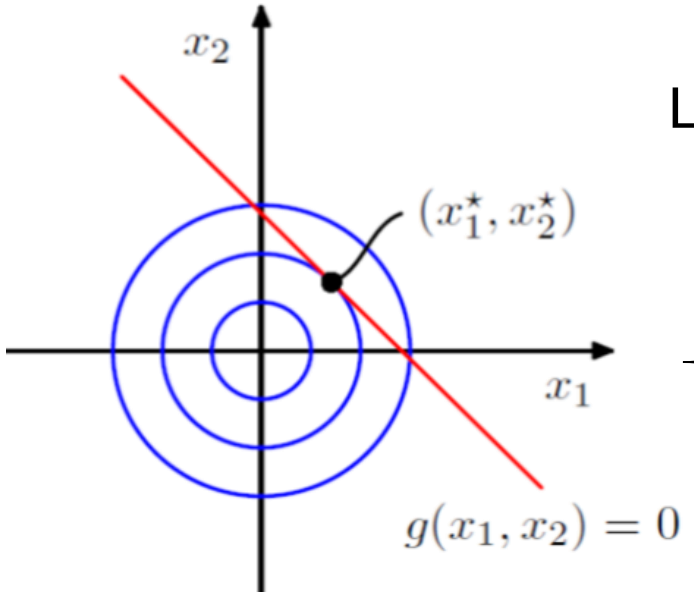
(constrained stationary condition)

Define **Lagrange function:** $\mathrm{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$

- Constraint stationary is obtained by: $\nabla_{\mathbf{x}} L = 0$
- $\frac{\partial L}{\partial \lambda} = 0$ leads to the constraint $g(\mathbf{x}) = 0$

Example

$f(x_1, x_2) = 1 - x_1^2 - x_2^2$ subject to $g(x_1, x_2) = x_1 + x_2 - 1 = 0$



Let $L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$

$$\begin{cases} \nabla_{\mathbf{x}} L = 0 \\ \dfrac{\partial L}{\partial \lambda} = 0 \end{cases} \Rightarrow \begin{cases} -2x_1 + \lambda = 0 \\ -2x_2 + \lambda = 0 \\ x_1 + x_2 - 1 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} (x_1^\star, x_2^\star) = (\frac{1}{2}, \frac{1}{2}) \\ \lambda = 1 \end{cases}$$

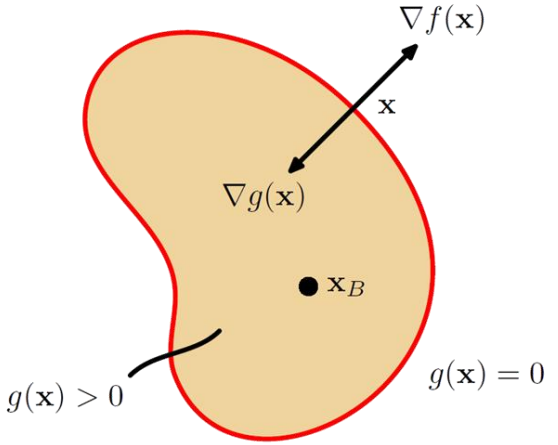
Lagrange Multipliers with Inequality Constraint

Maximize f(x) constrained on $g(\mathbf{x}) \geq 0$

➔Optimize $\mathrm{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$
with respect to $\mathbf{x}$ and λ subject to

$$\begin{aligned} g(\mathbf{x}) &\geqslant 0 \\ \lambda &\geqslant 0 \\ \lambda g(\mathbf{x}) &= 0 \end{aligned}$$

Karush-Kuhn-Tucker (KKT) conditions



[Case 1] The stationary point lies in $g(\mathbf{x}) > 0$,
→ The constraint $g(\mathbf{x}) = 0$ is **inactive** :
The stationary condition is simply

$$\nabla f(\mathbf{x}) = 0$$

➔**Lagrange function:**
$\mathrm{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$ with $\lambda = 0$
$\because \lambda = 0$
$\therefore \lambda g(\mathbf{x}) = 0$

[Case 2] The stationary point lies on $g(\mathbf{x}) = 0$,
→ The constraint $g(\mathbf{x}) = 0$ is **active** :
The constrained stationary condition is

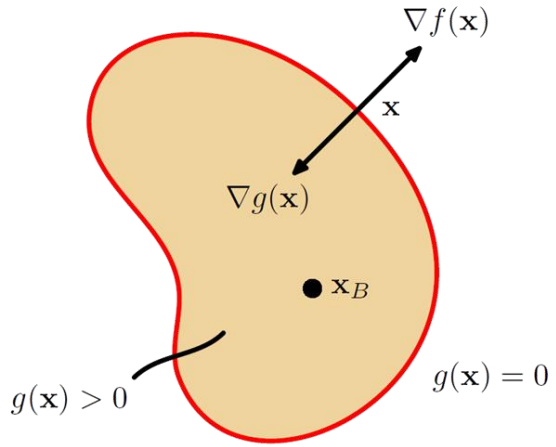
$$\nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0$$

➔**Lagrange function:**
$\mathrm{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$ with $\lambda \neq 0$
$\because g(\mathbf{x}) = 0$
$\therefore \lambda g(\mathbf{x}) = 0$
$\because f(\mathbf{x})$ will be at a maximum only if $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ have different direction.
Moreover, $\nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0$
$\therefore \lambda > 0$

Lagrange Multipliers

Maximize $f(\mathbf{x})$

constrained on $g(\mathbf{x}) > 0$



$\mathbf{x} + \epsilon$: a nearby point that also lie on the constraint surface

By Taylor expansion around $\mathbf{x}$:

$g(\mathbf{x} + \epsilon) \simeq g(\mathbf{x}) + \epsilon^T \nabla g(\mathbf{x})$

$\because$ both $\mathbf{x} + \epsilon$ and $\mathbf{x}$ lie on the constraint surface

$\therefore g(\mathbf{x} + \epsilon) = g(\mathbf{x}) = 0$

$\rightarrow \epsilon^T \nabla g(\mathbf{x}) = 0$

$\rightarrow \epsilon \perp \nabla g(\mathbf{x})$

$\because \epsilon \parallel g(\mathbf{x}) = 0$

$\therefore \nabla g(\mathbf{x})$ is normal to the surface $g(\mathbf{x}) = 0$

Taylor expansion of $f(\mathbf{x})$:

$f(\mathbf{x} + \boldsymbol{\epsilon}) \simeq f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x})$

If $f(\mathbf{x})$ is the maximum,

$\nabla f(\mathbf{x})$ is normal to the surface $g(\mathbf{x}) = 0$

$\therefore \nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are parallel or anti-parallel

$\rightarrow \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = \mathbf{0}$, where $\lambda \neq 0$

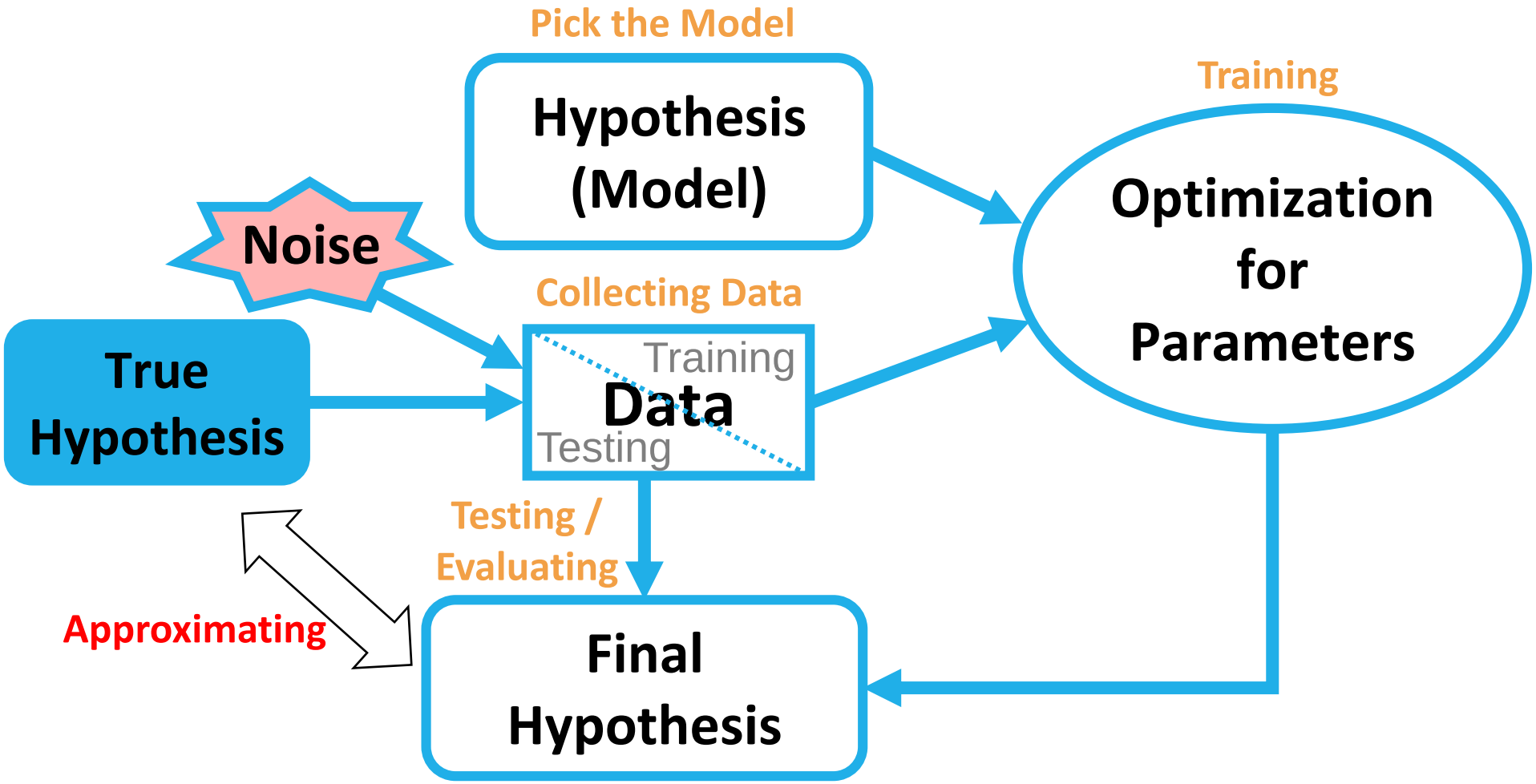
(constrained stationary condition)

$\lambda = 0$

Define **Lagrange function:** $\mathrm{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$

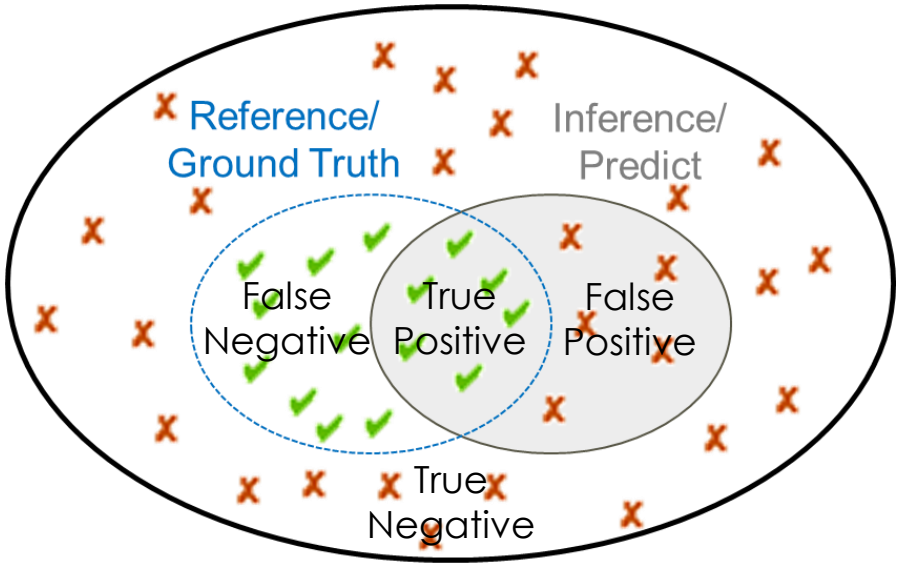
- Constraint stationary is obtained by: $\nabla_{\mathbf{x}} L = 0$
- $\frac{\partial L}{\partial \lambda} = 0$ leads to the constraint $g(\mathbf{x}) = 0$

Machine Learning Process



Evaluation

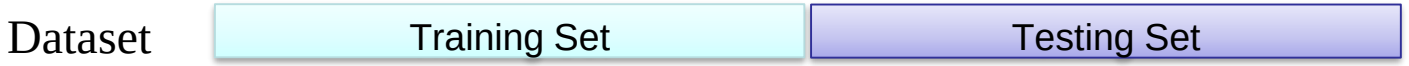
- Accuracy
  - $\frac{\#Correct}{\#Total\ Sample} = \frac{\#TP+\#TN}{\#TP+\#FN+\#FP+\#TN}$
- Precision
  - $P = \frac{\#TP}{\#TP+\#FP}$
- Recall (Sensitivity)
  - $R = \frac{\#TP}{\#TP+\#FN}$
- F1-Measure
  - $F1 = \frac{2PR}{P+R}$
- AP, MAP



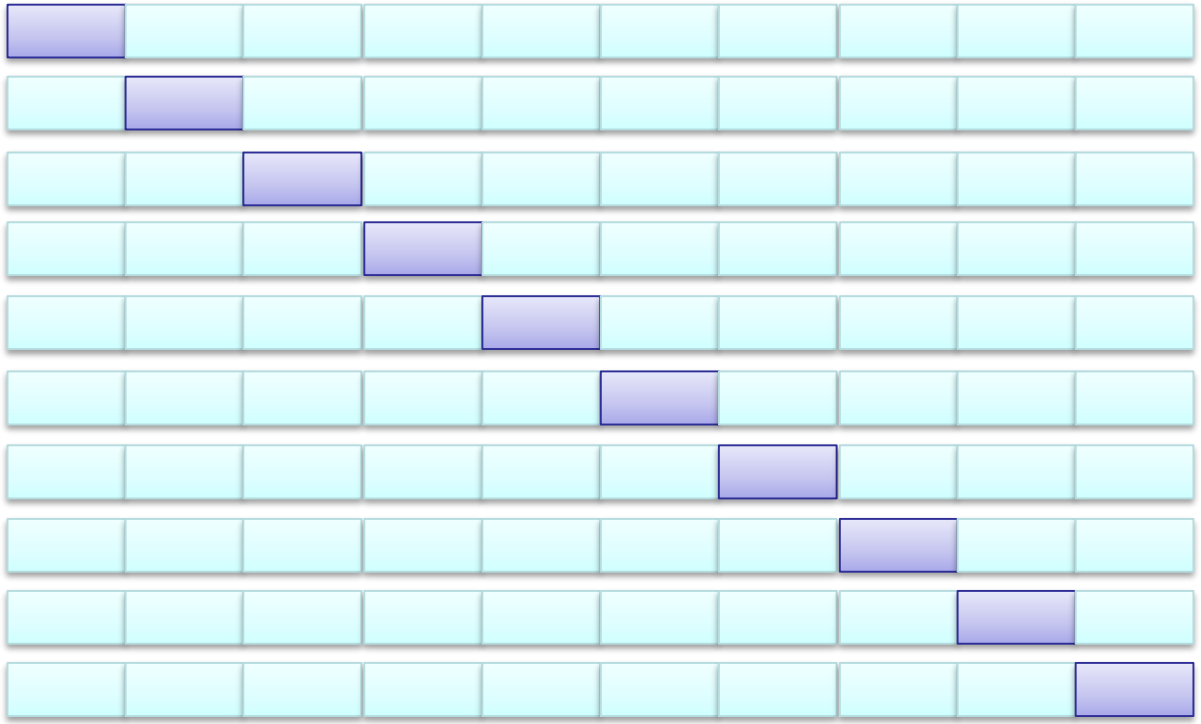
Inference/
Predict

Reference/ Ground Truth		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Validation



- K-fold cross validation
- Leave One Out



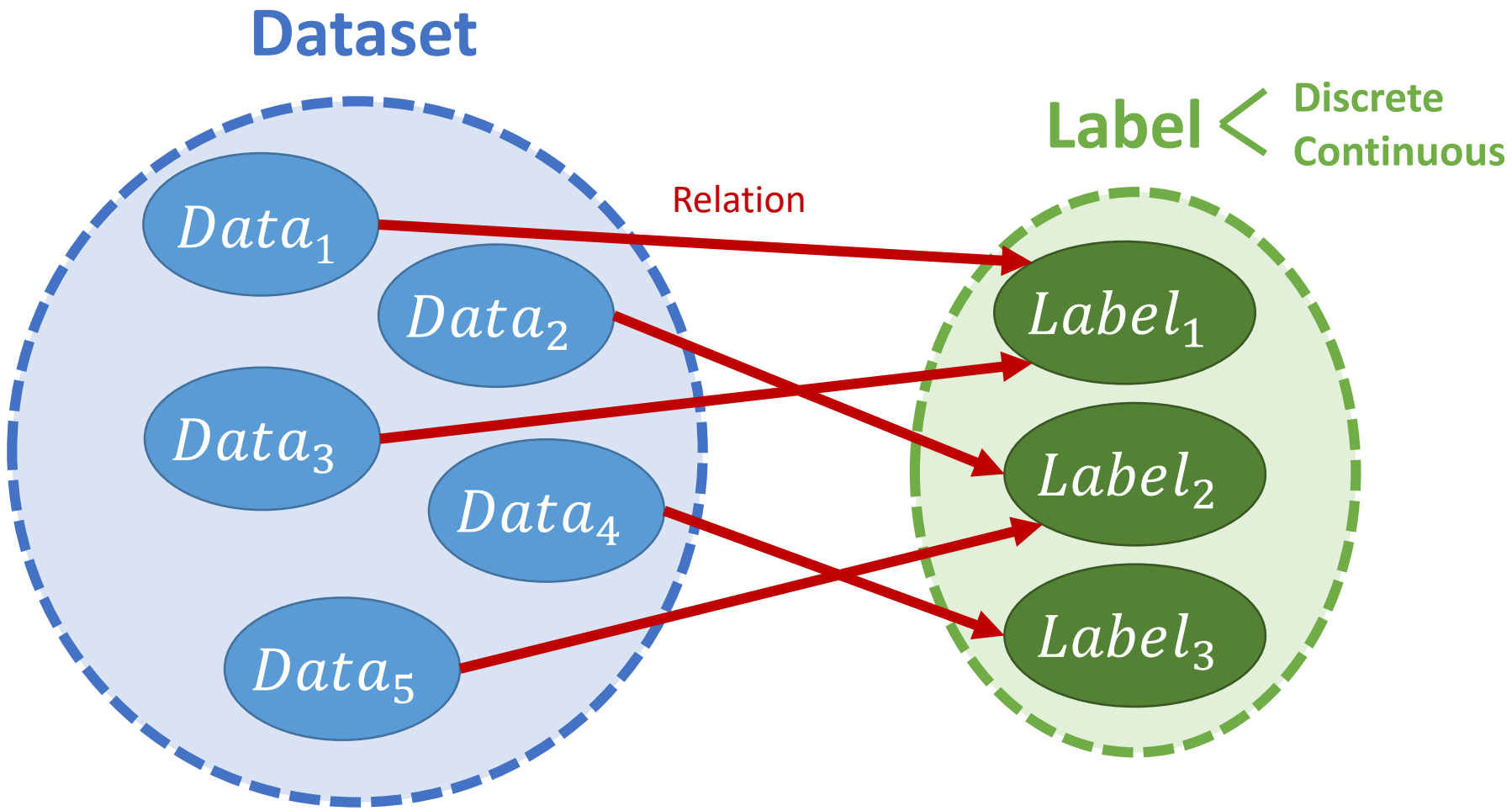
$M(\theta_1) \rightarrow Acc_1$

$M(\theta_2) \rightarrow Acc_2$

$\vdots$

$M(\theta_K) \rightarrow Acc_K$

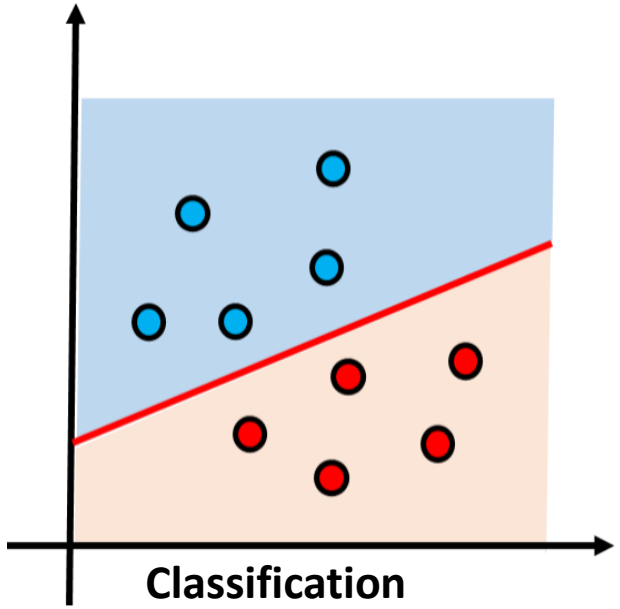
Supervised Learning



Find the relation between data and label.

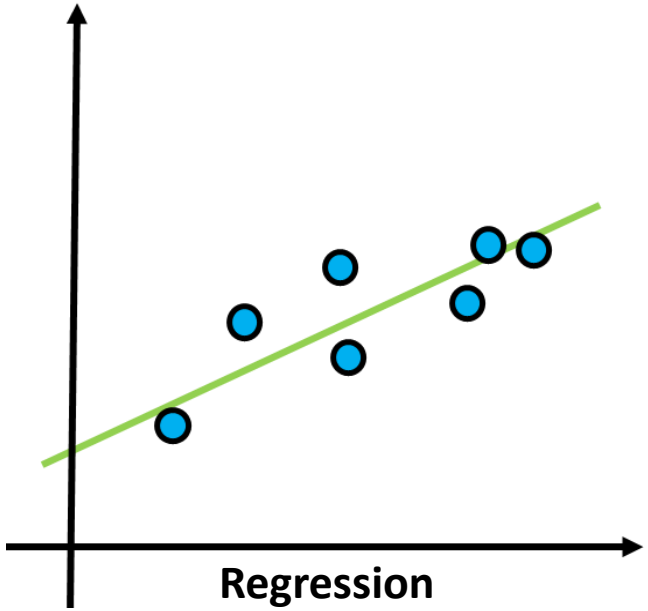
Classification vs. Regression

Discrete Label



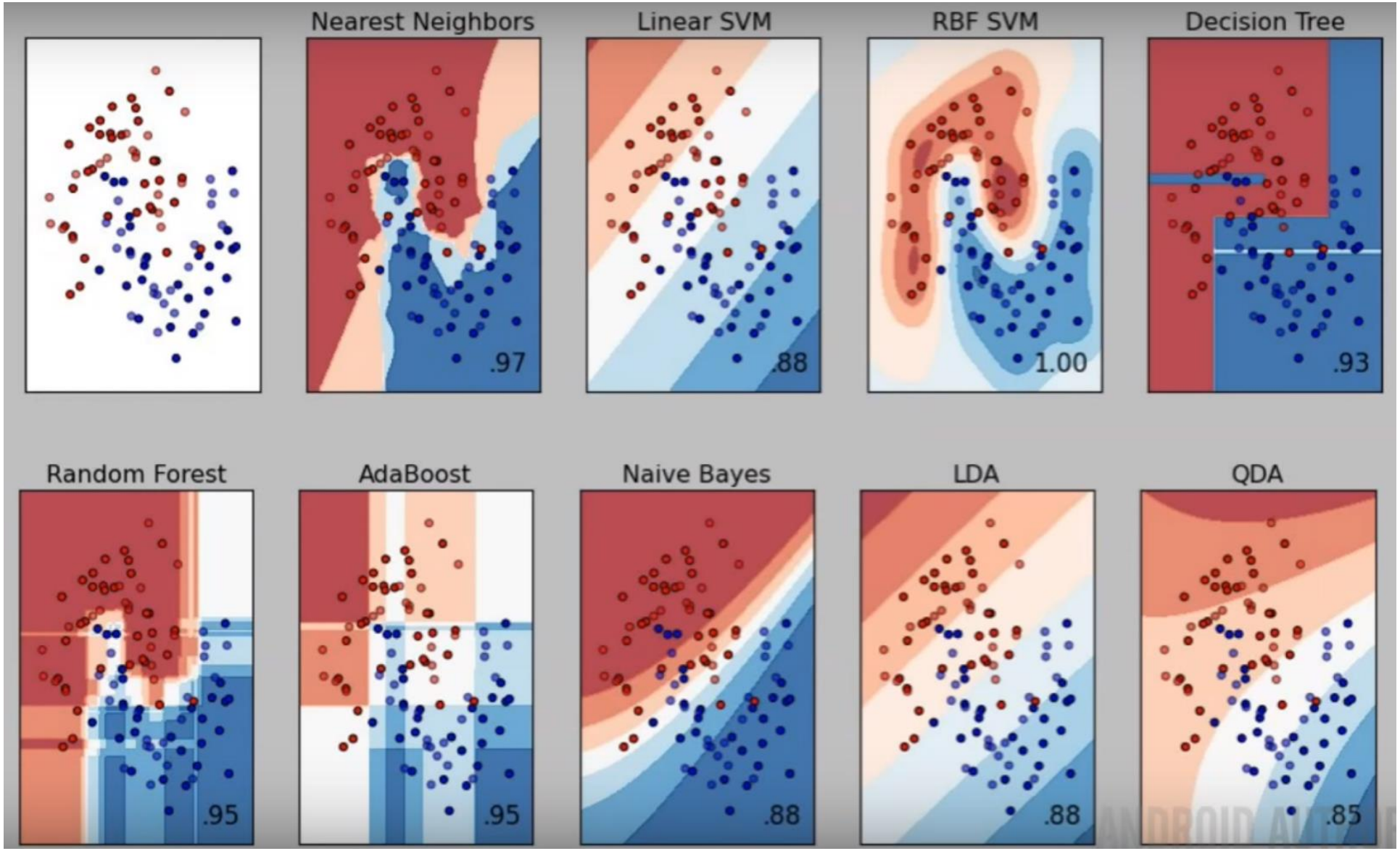
- ✓ Logistic Regression
- ✓ Support Vector Machine (SVM)
- ✓ Neural Network (NN)

Continuous Label

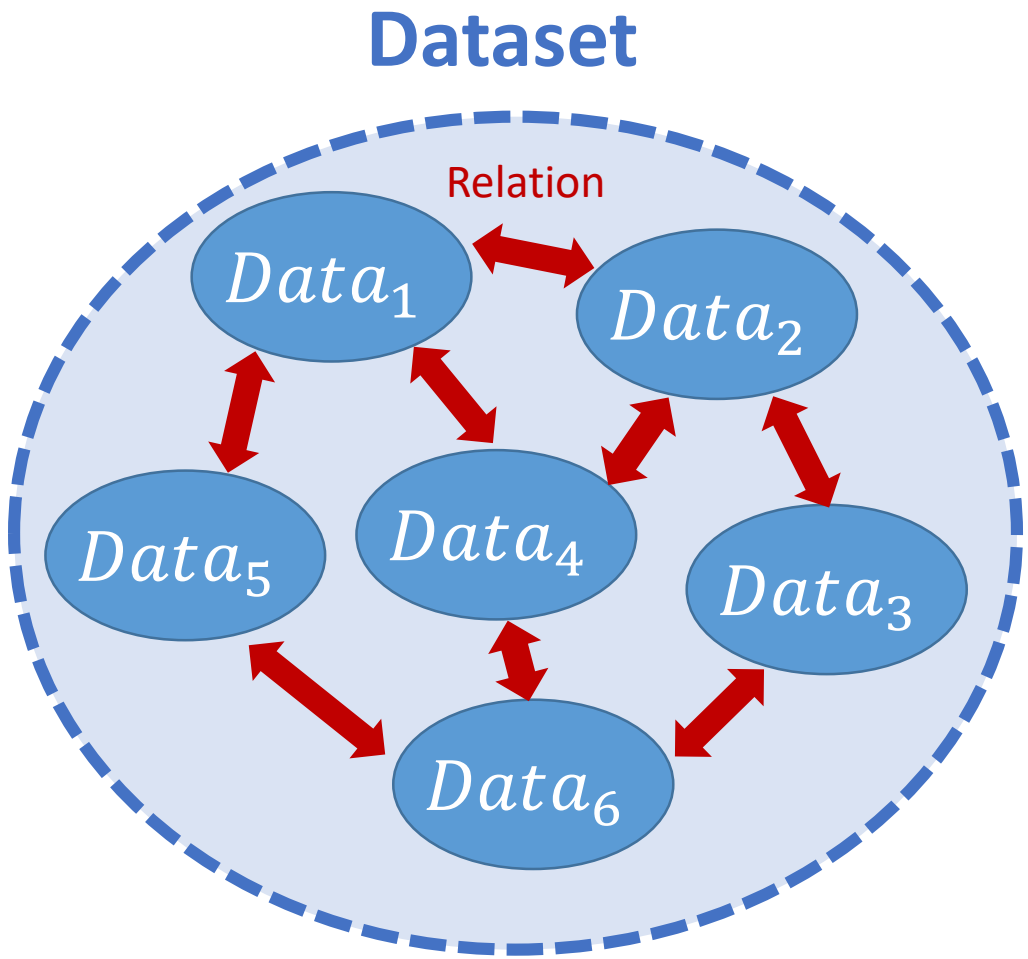


- ✓ Linear Regression
- ✓ Support Vector Regression (SVR)
- ✓ Neural Network (NN)

Visualization of Classification Methods



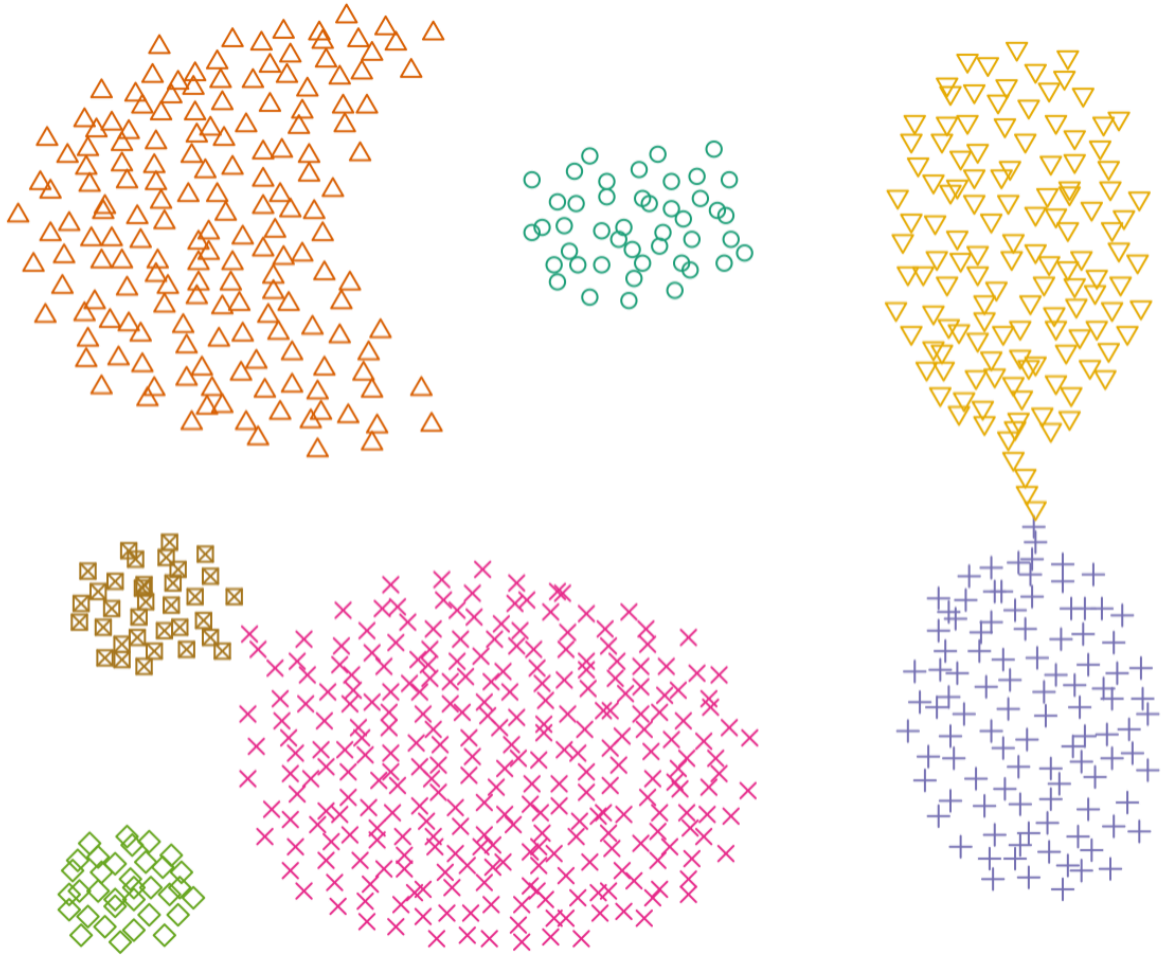
Unsupervised Learning



Find the relation between data points.

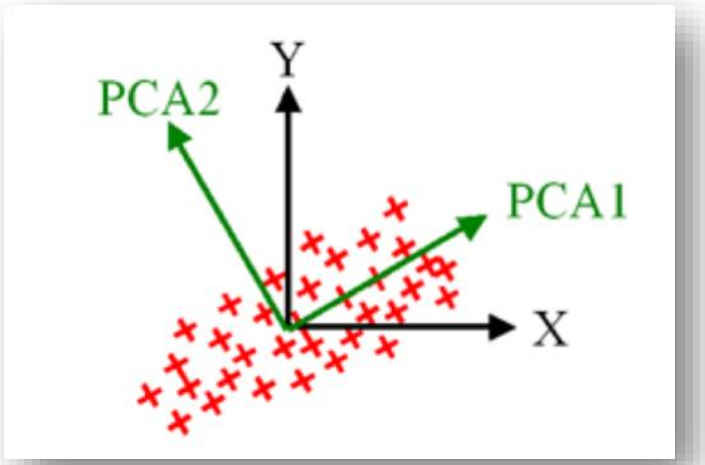
Unsupervised Learning

- Clustering
  - K-means
  - Affinity Propagation
  - Spectral Clustering
  - NMF

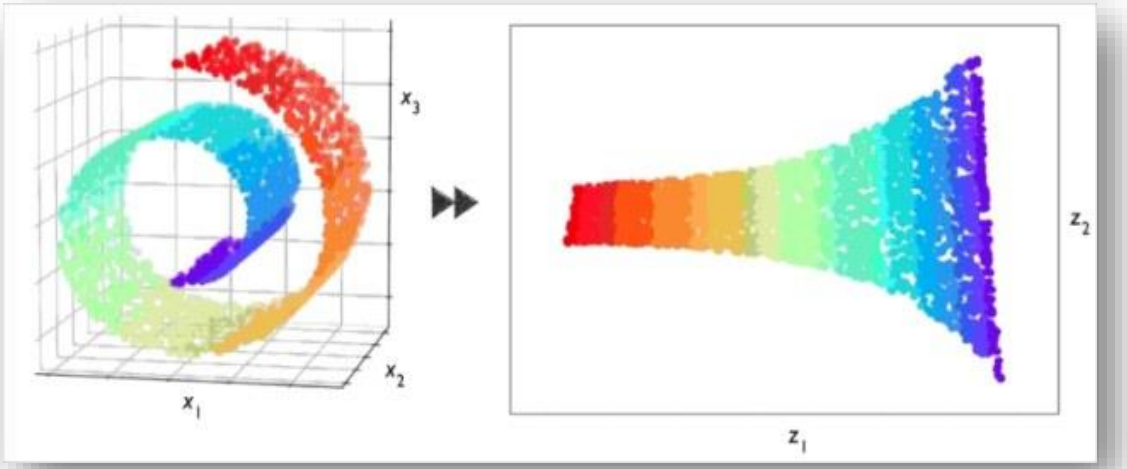


Unsupervised Learning

- Dimension Reduction
  - PCA/LDA
  - LLE/TSNE/AutoEncoder



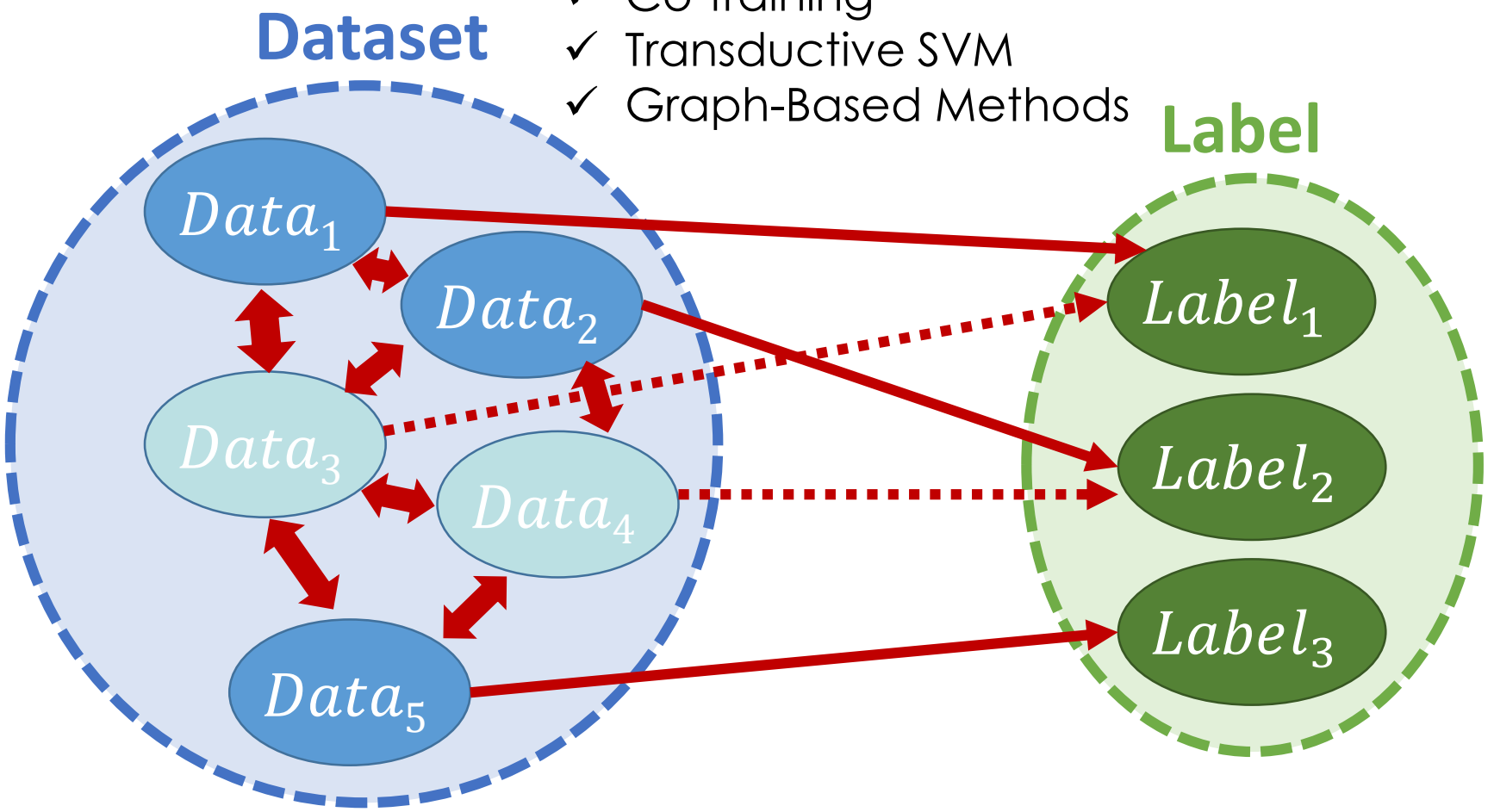
Linear



Non-Linear

Semi-supervised Learning

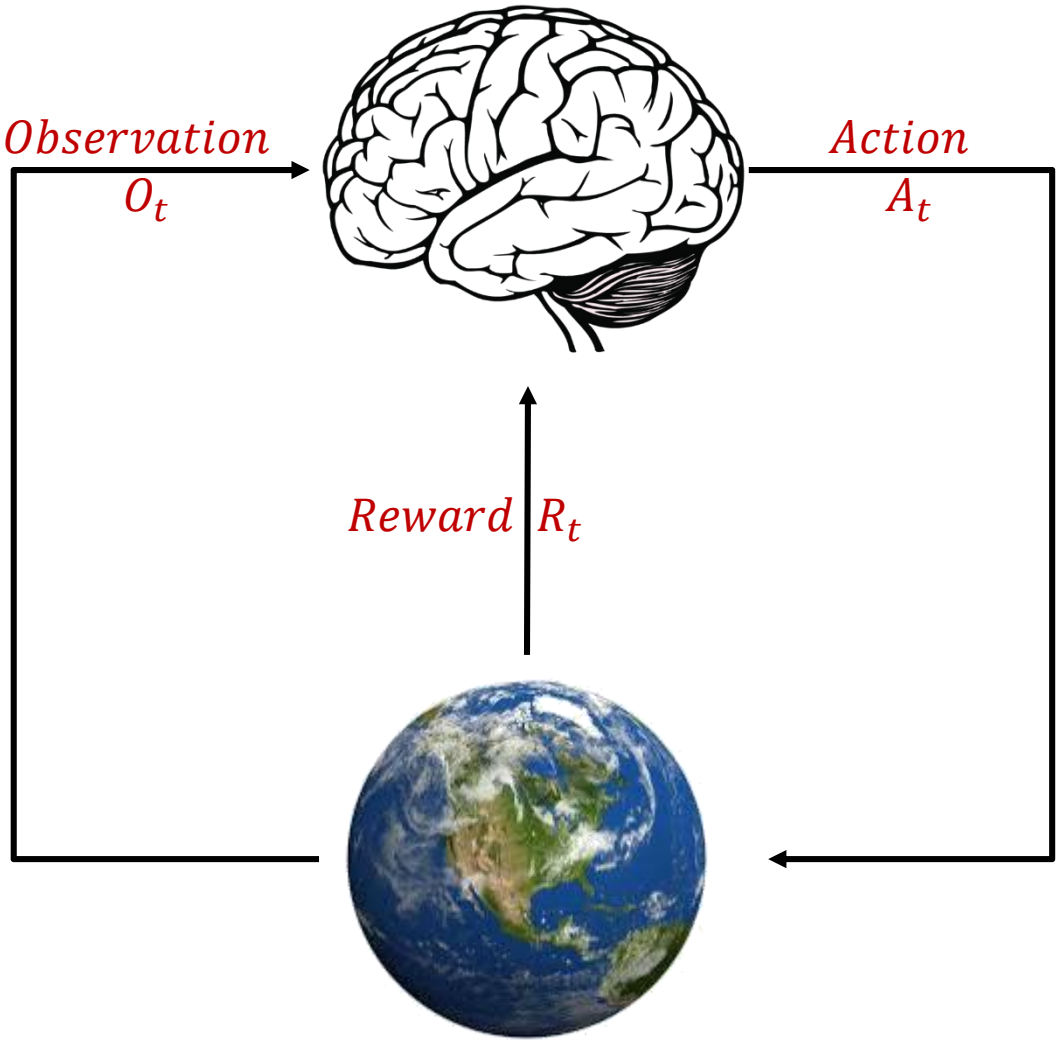
- ✓ EM with generative mixture models
- ✓ Self Training
- ✓ Co-training
- ✓ Transductive SVM
- ✓ Graph-Based Methods



Reinforcement Learning

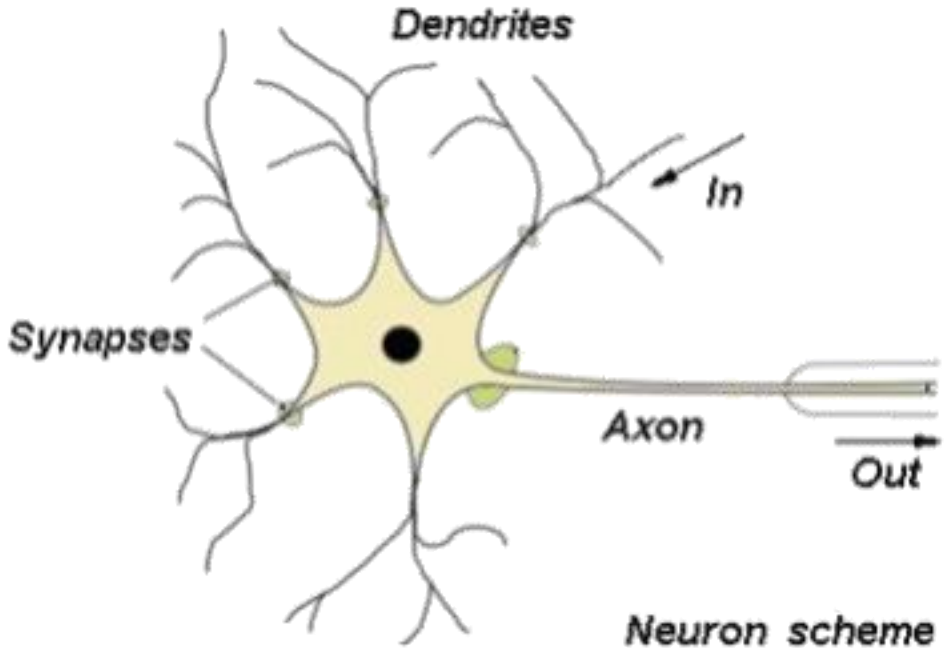
At each step t

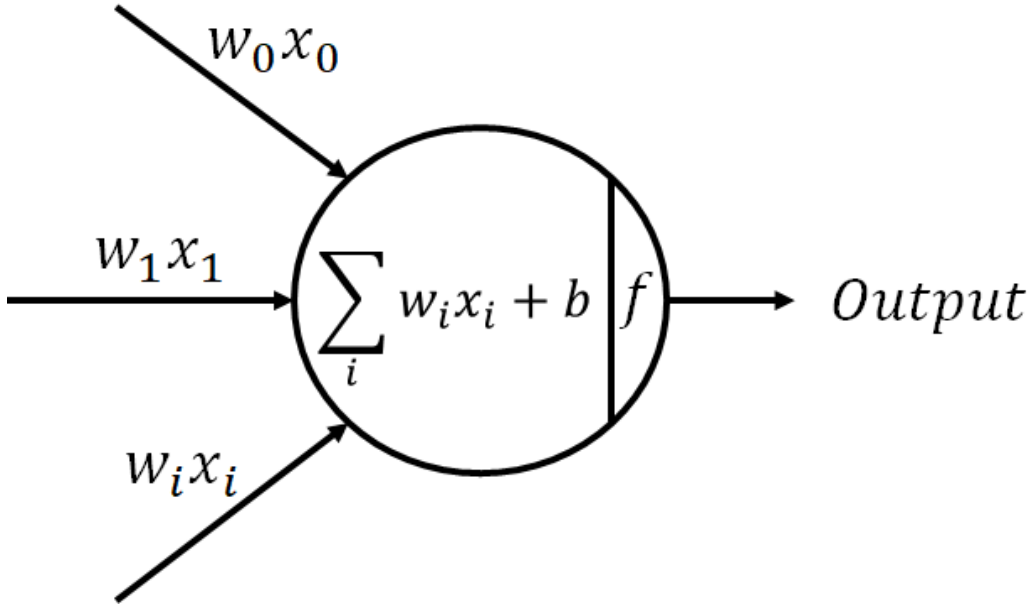
- Agent
  - Receives $observation\ O_t$
  - Receives $reward\ R_t$
  - Executes $action\ A_t$
- Environment
  - Receives $action\ A_t$
  - Emits $observation\ O_{t+1}$
  - Emits $reward\ R_{t+1}$



Deep Learning

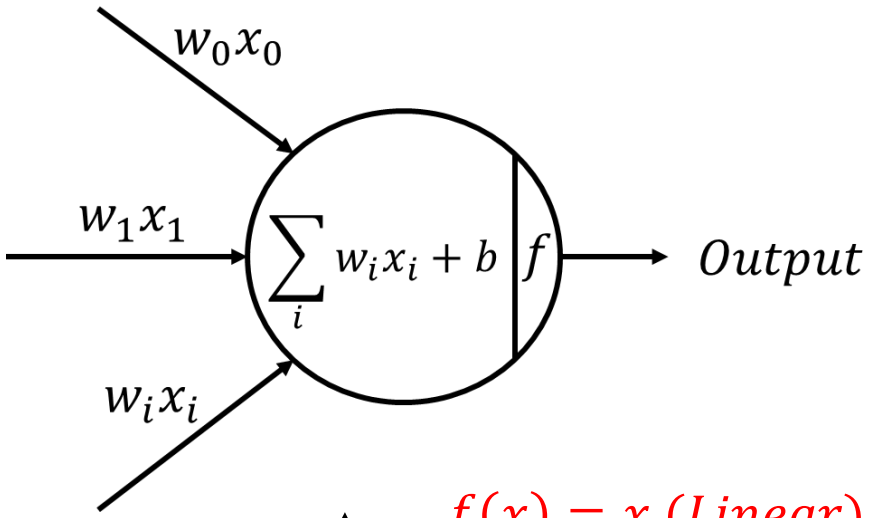
Neuron

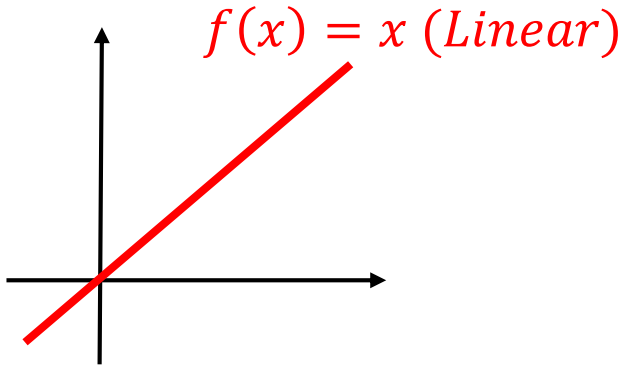

Dendrites
In
Synapses
Axon
Out
Neuron scheme


$w_0 x_0$
$w_1 x_1$
$w_i x_i$
$\sum_i w_i x_i + b$
f
$Output$

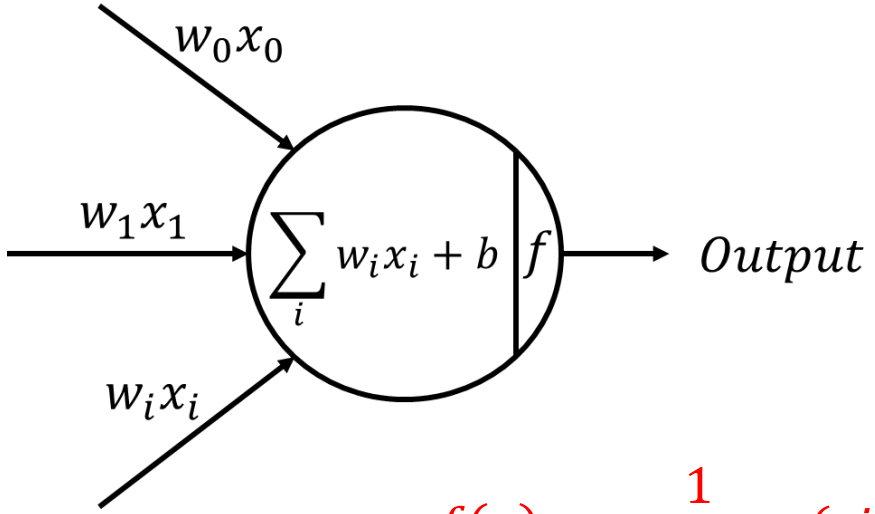
Linear/Logistic Regression

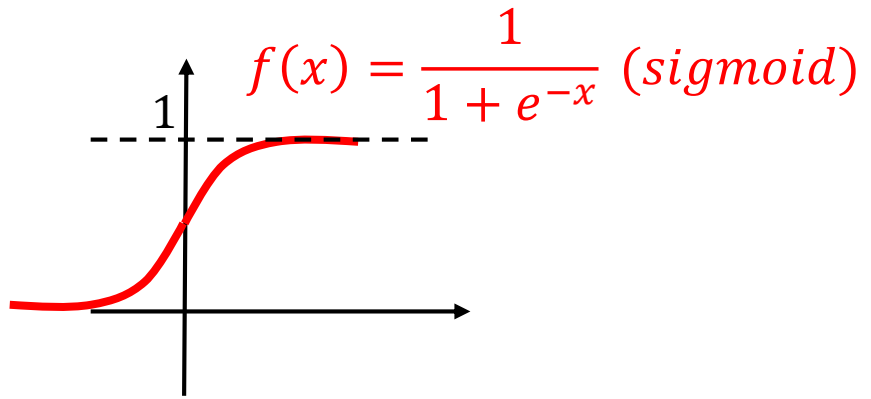
Linear Regression



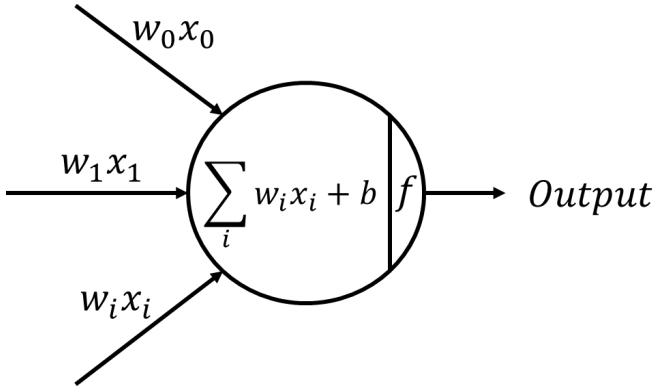


Logistic Regression

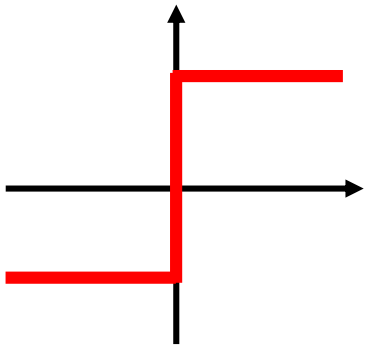




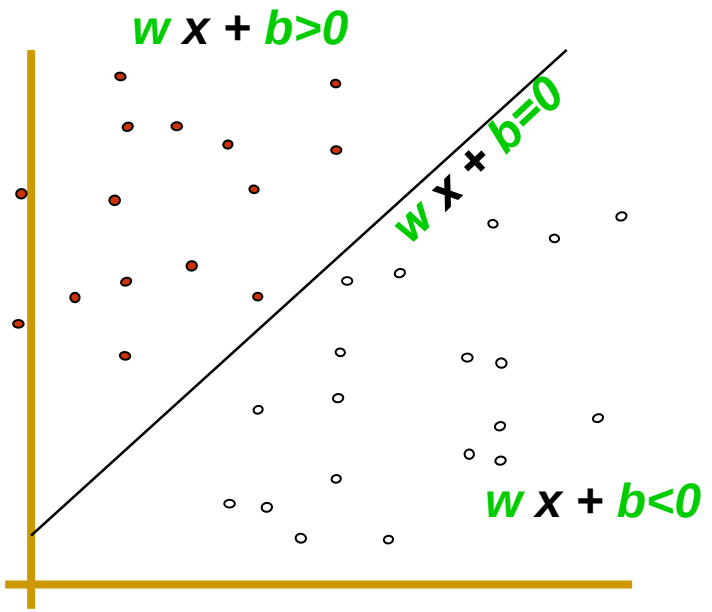
Perceptron



$$f(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \end{cases}$$



$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}\,\mathbf{x} + b)$$



The Perceptron Learning Algorithm (PLA)

$$y(\mathbf{x}) = f(\boldsymbol{w}^{\boldsymbol{T}}\mathbf{x}) \qquad f(a) = \begin{cases} +1, a \geq 0 \\ -1, a < 0 \end{cases}$$

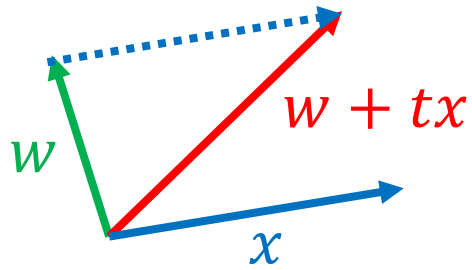
Perceptron criterion:

$$E_P(\boldsymbol{w}) = -\sum_i \boldsymbol{w}^T \mathbf{x}_\mathrm{i} t_i$$

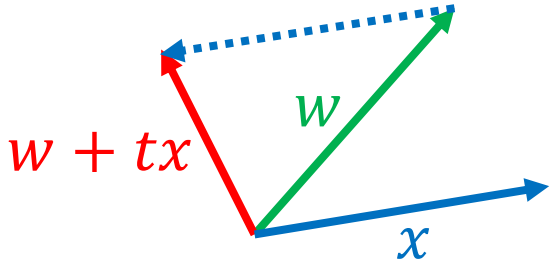
$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_P(w) = w^{(\tau)} + \eta x_i t_i$$

η: Learning Rate

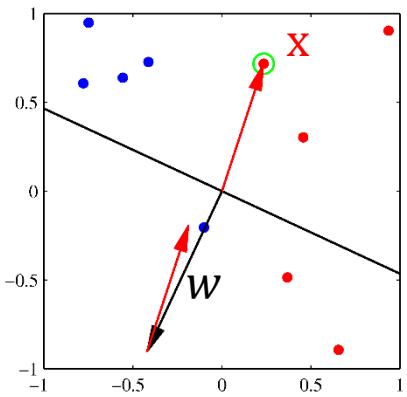
$t = +1$



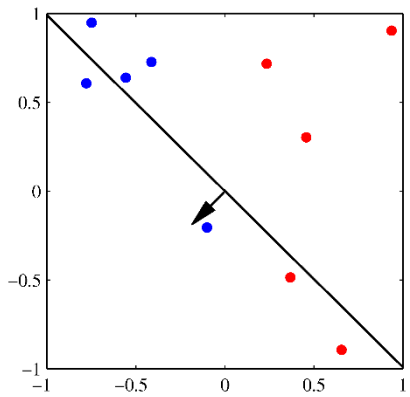
$t = -1$

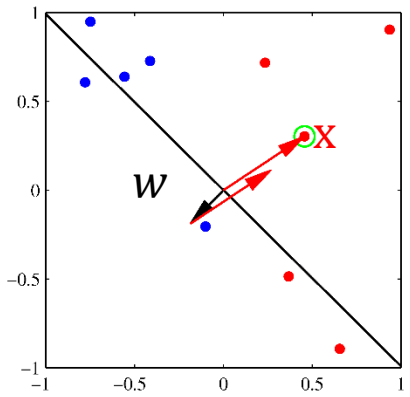


The Perceptron Learning Algorithm (PLA)

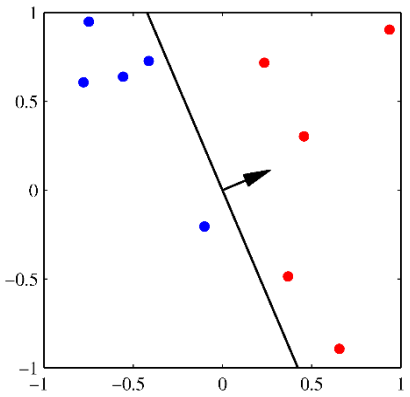


$t = +1$

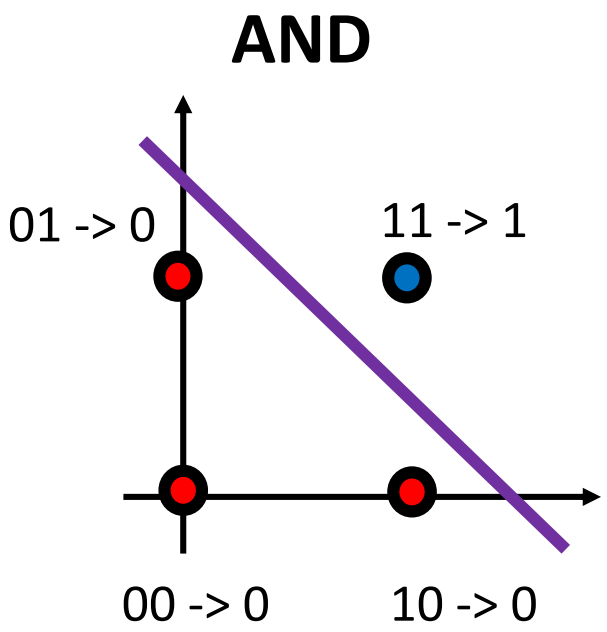


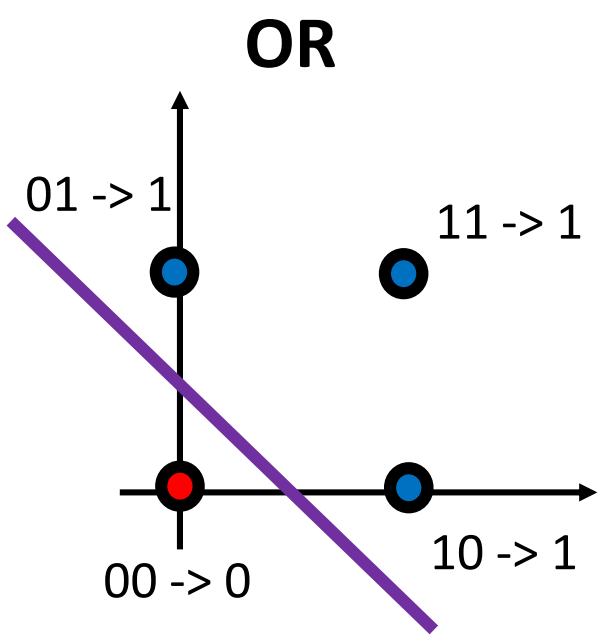


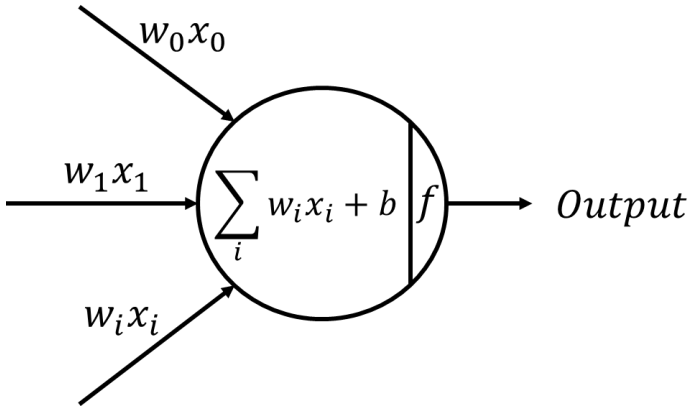
$t = +1$



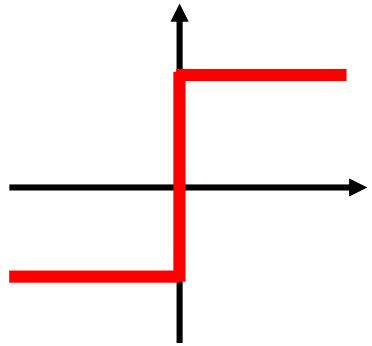
Perceptron



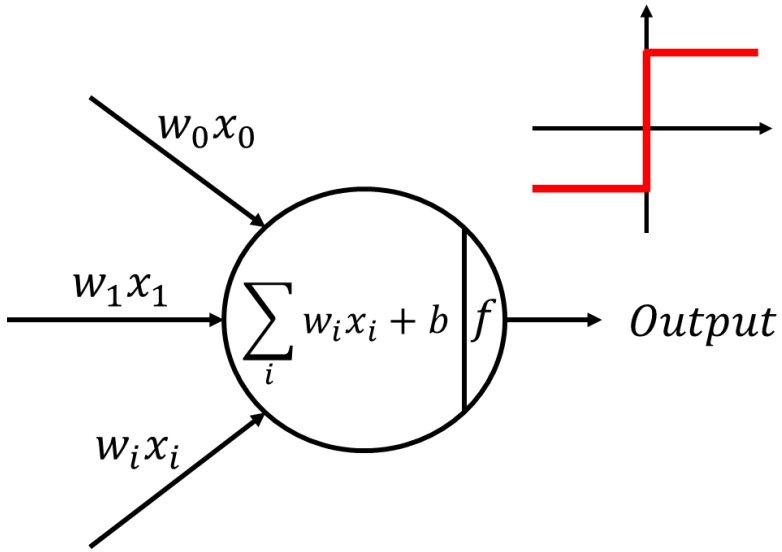


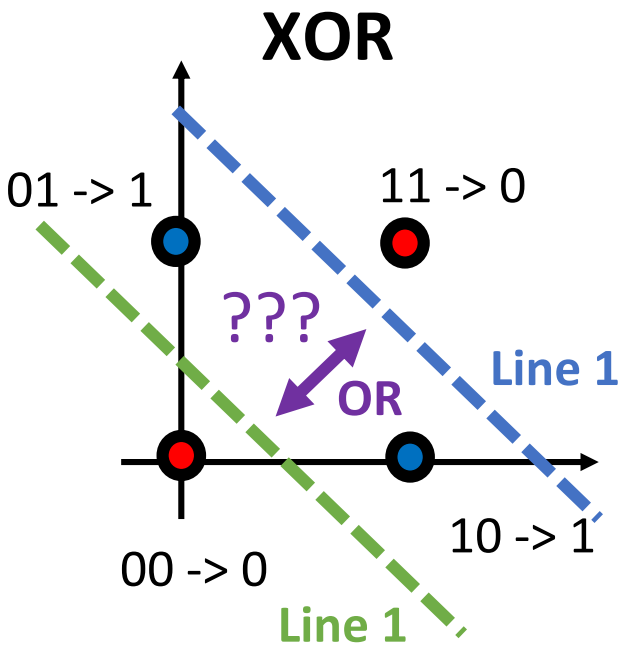


$$f(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \end{cases}$$

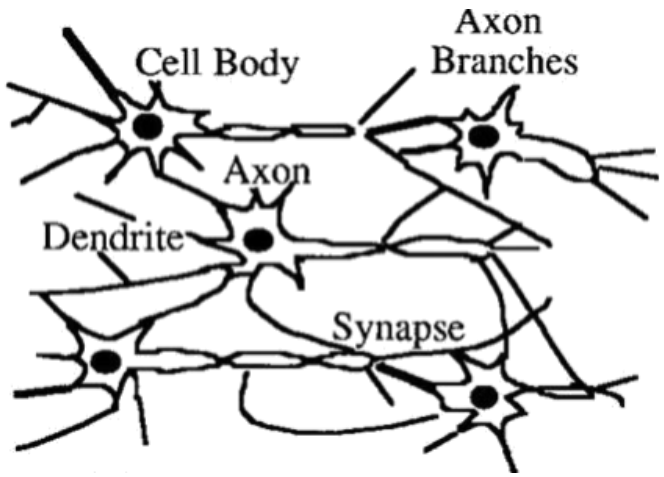


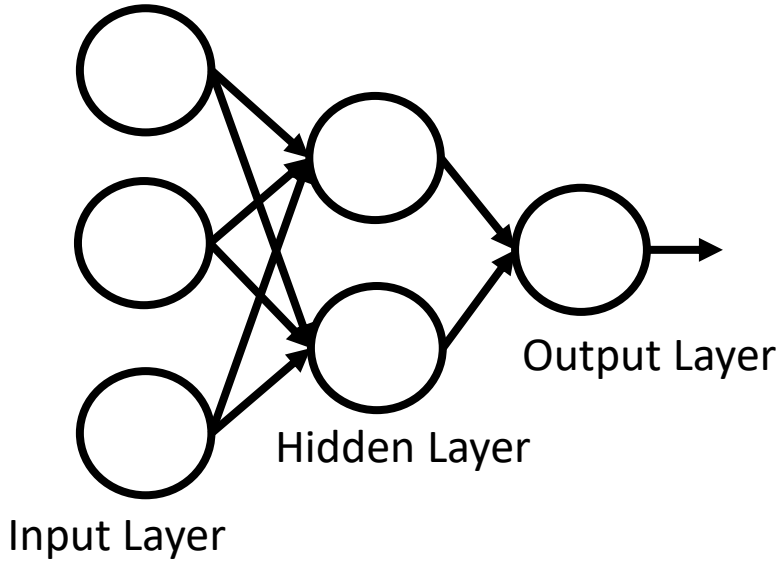
XOR Problem

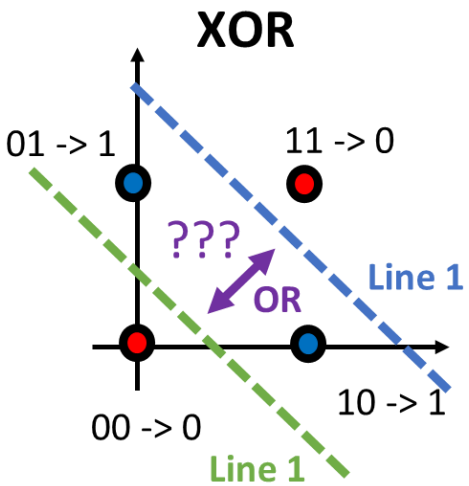

w_0x_0
w_1x_1
w_ix_i
$\sum_i w_ix_i + b$
f
$Output$

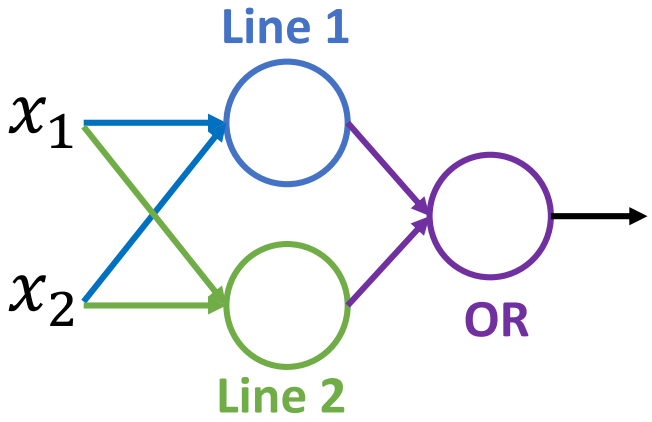

XOR
01 -> 1
11 -> 0
???
OR
Line 1
10 -> 1
00 -> 0
Line 1

Connect the Neurons



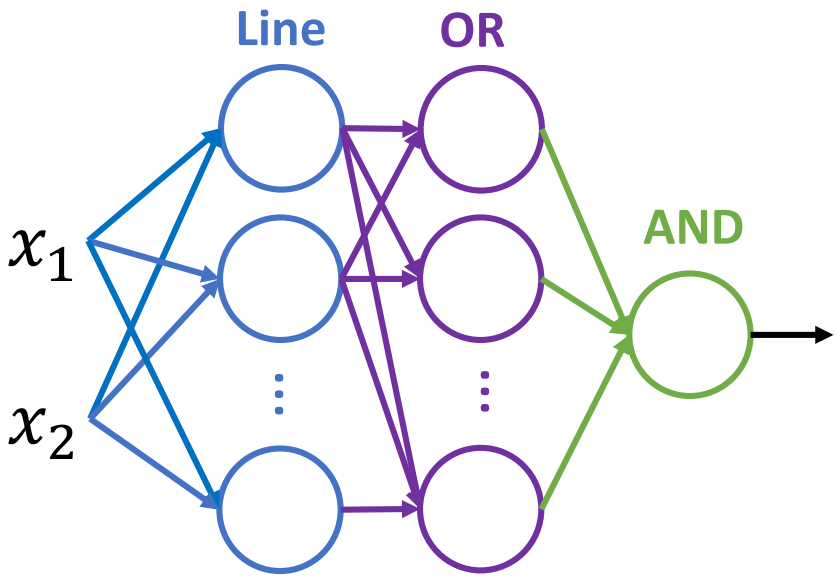


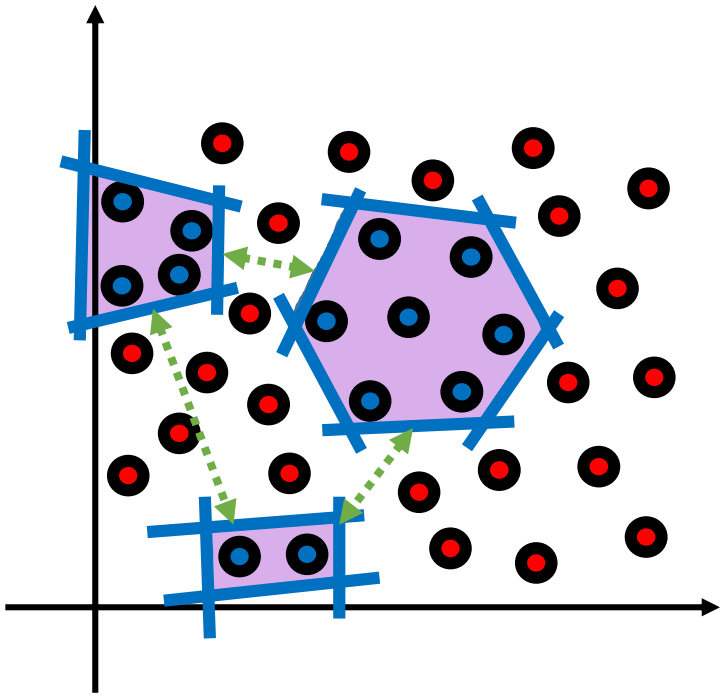




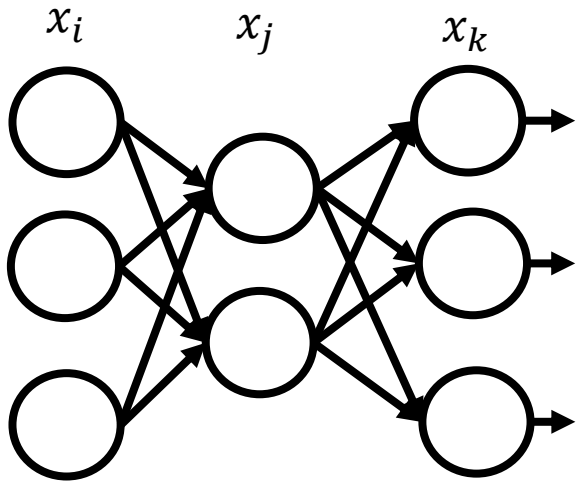
Remark

The neural network with 3-layers can fit any finite training data.





Formulate the Multi-Layer NN

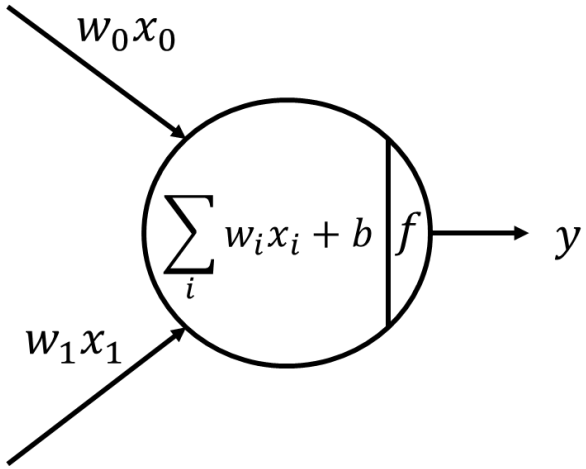


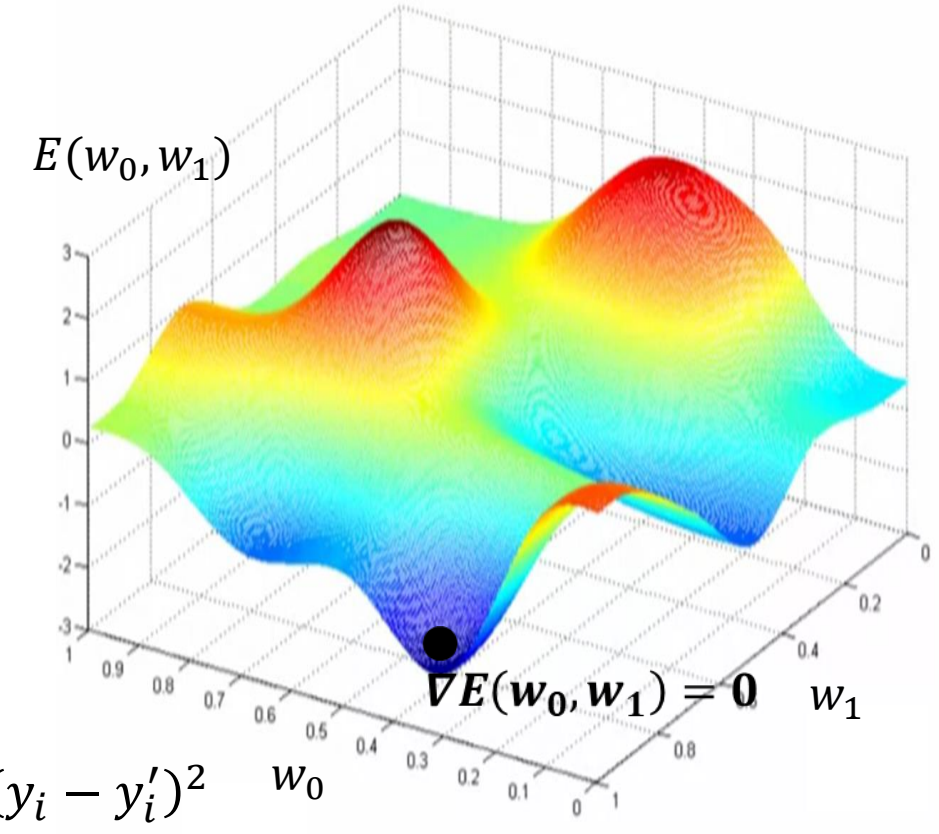
$$f(\sum_j w_{jk}^1 f\left(\sum_i w_{ij}^0 x_i + b_j^0\right) + b_k^1) = y_k$$

$$f(\begin{bmatrix} w_{00}^1 & w_{10}^1 \\ w_{01}^1 & w_{11}^1 \\ w_{02}^1 & w_{12}^1 \end{bmatrix} f\left(\begin{bmatrix} w_{00}^0 & w_{10}^0 & w_{20}^0 \\ w_{01}^0 & w_{11}^0 & w_{21}^0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_0^0 \\ b_1^0 \end{bmatrix}\right) + \begin{bmatrix} b_0^1 \\ b_1^1 \\ b_2^1 \end{bmatrix}) = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

$$f\left(W^1 f\left(W^0 X + b^0\right) + b^1\right) = Y$$

Find the Optimal Parameter

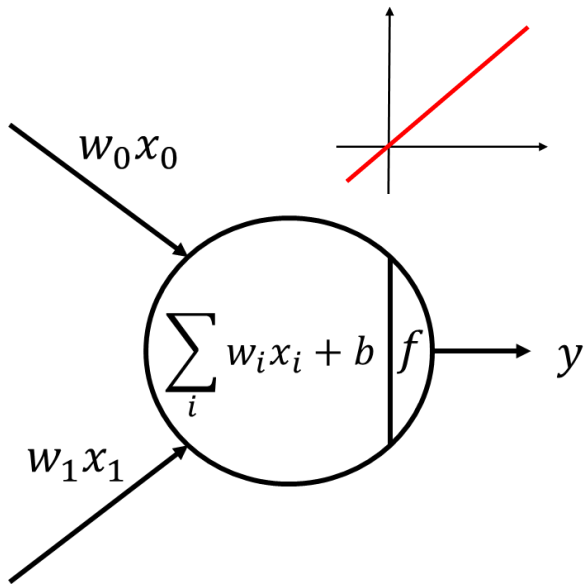


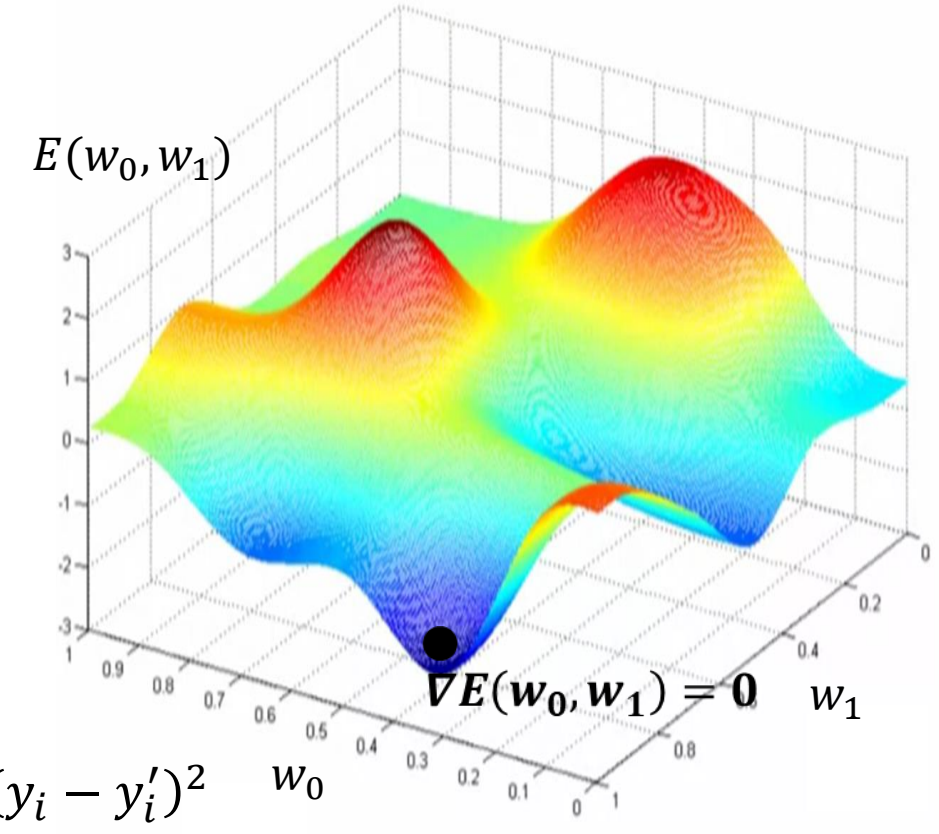


$$E(w_0, w_1) = \frac{1}{2}\sum_i (y_i - y_i')^2$$

$$Calculate\ \boldsymbol{gradient}\colon \nabla E(w_0, w_1) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}\right)$$

Find the Optimal Parameter



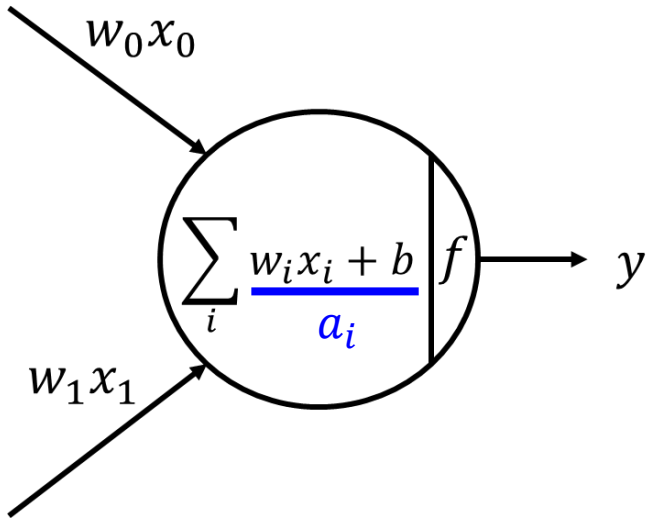


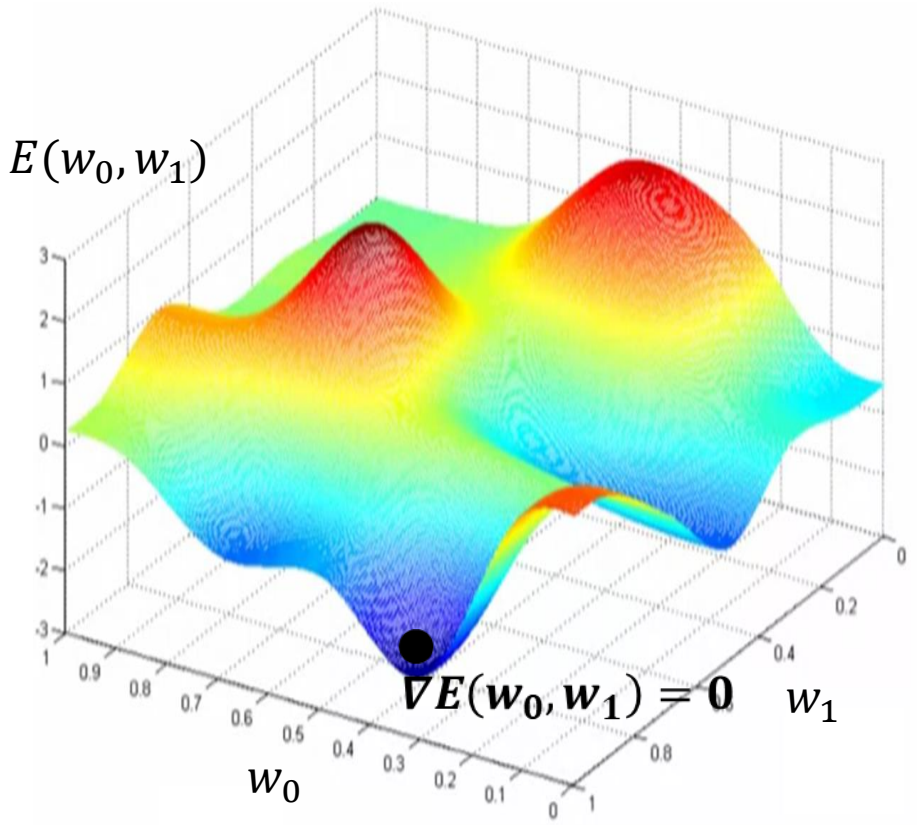
$$E(w_0, w_1) = \frac{1}{2}\sum_i (y_i - y_i')^2$$

$if\ f(x) = x,$

$$\frac{\partial E}{\partial w_i} = \frac{\partial \frac{1}{2}(\sum_i w_i x_i + b - y')^2}{\partial w_i} = \frac{\partial \frac{1}{2}(w_i x_i + b - y')^2}{\partial w_i} = (w_i x_i + b - y')x_i$$

Chain Rules & Back Propagation





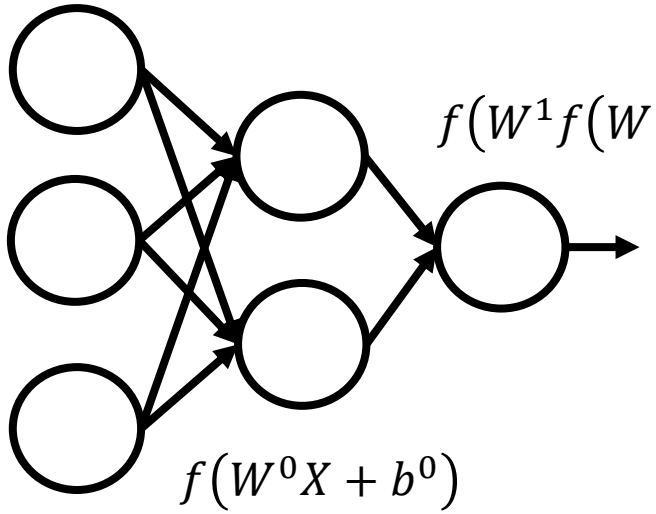
Chain Rule

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial f}\frac{\partial f}{\partial a_i}\frac{\partial a_i}{\partial w_i}$$

f is sigmoid :

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}, \qquad \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Chain Rules & Back Propagation

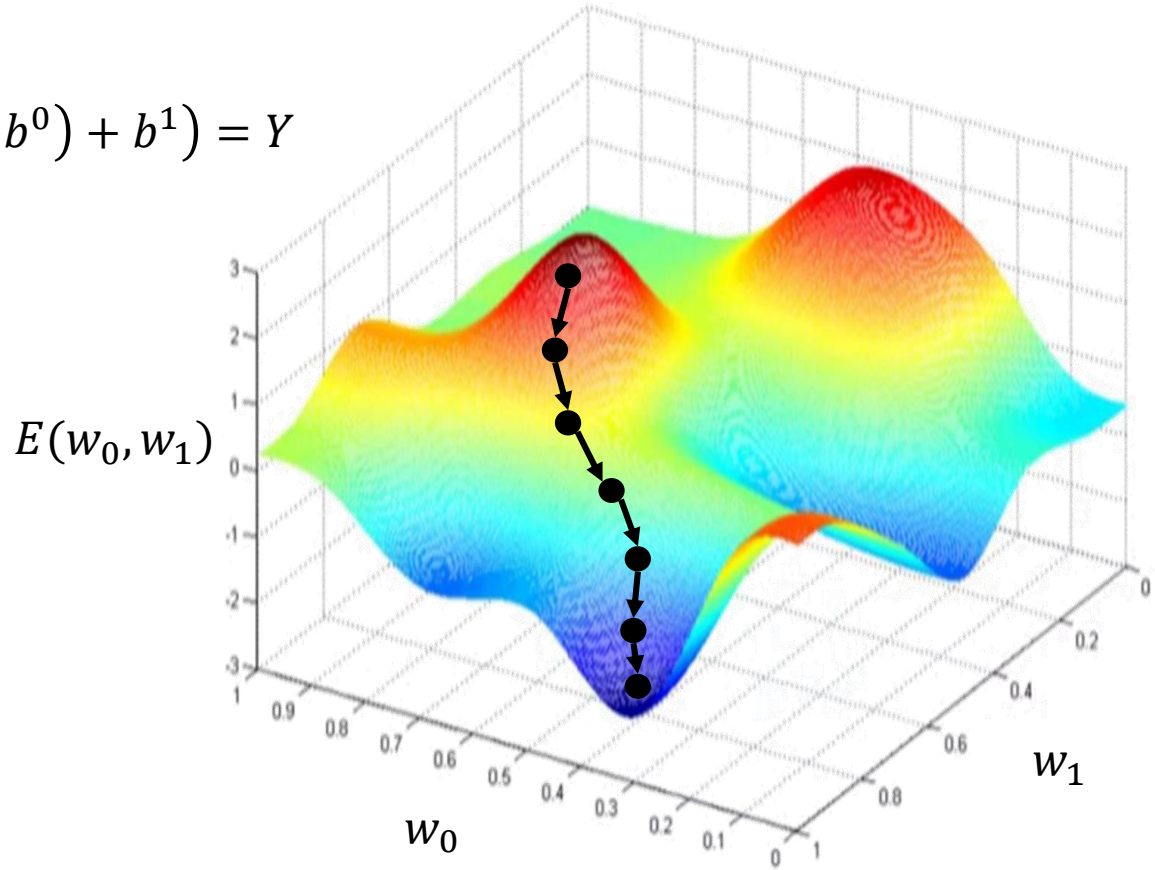


$$f\left(W^1 f\left(W^0 X + b^0\right) + b^1\right) = Y$$

$$f\left(W^0 X + b^0\right)$$

$$\frac{\partial E}{\partial w_i^1} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_i^1}$$

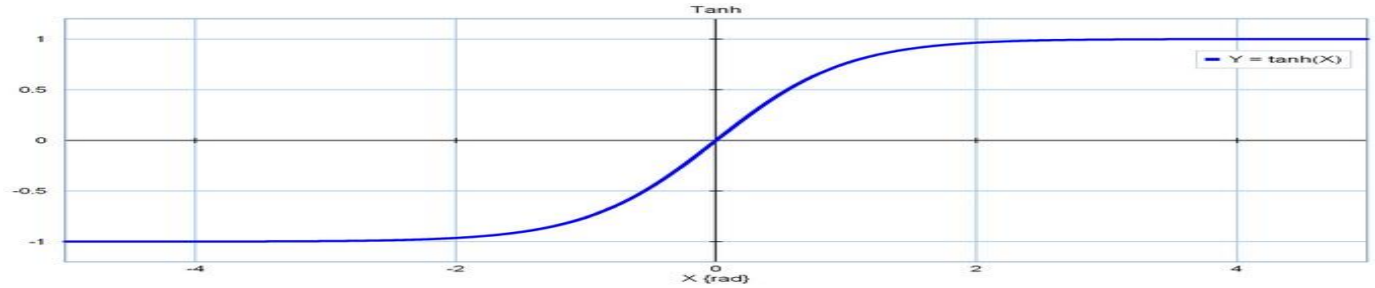
$$\frac{\partial E}{\partial w_i^0} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial x_i^1}\frac{\partial x_i^1}{\partial w_i^0}$$



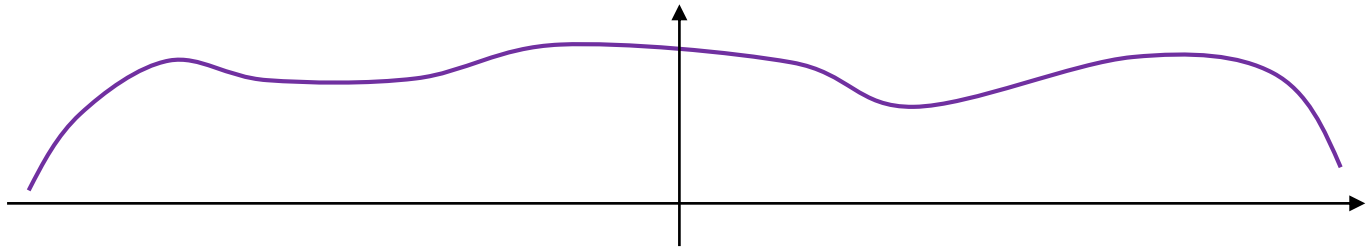
Normalization Problem

- The distribution of the input tensor can affect the performance of the training process dramatically.

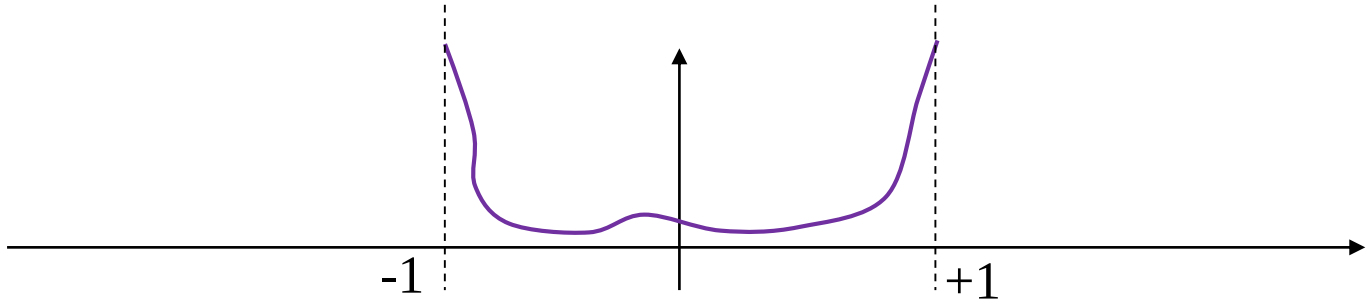
Activation



Input Distribution



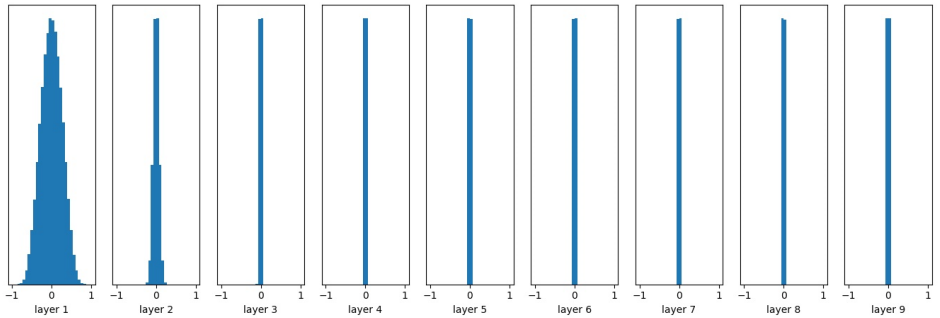
Output Distribution



Normalization Problem

- If the variance of the input is small, after the activation function, the distribution will aggregate to zero

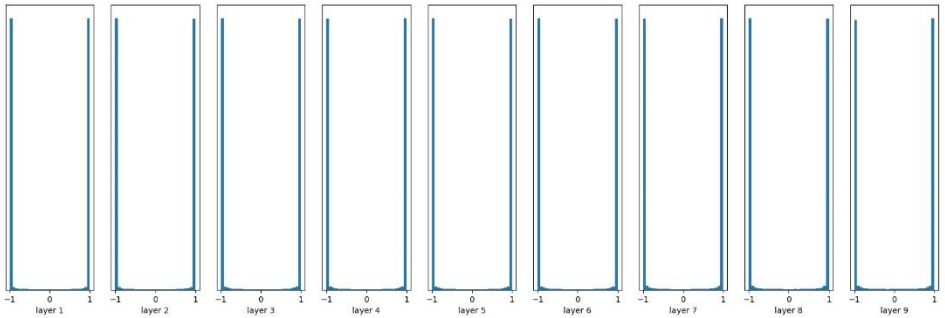
$Small\ Variance$



Tanh Activation

- If the variance of the input is big, after the activation function, the distribution will aggregate to -1 and +1.

$Big\ Variance$

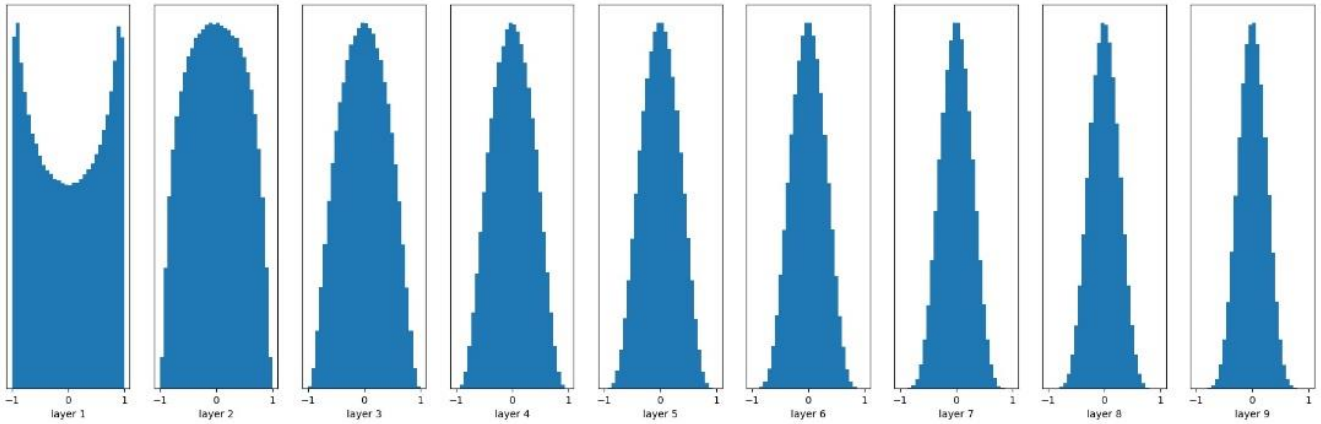


Tanh Activation

Initialization of Parameters

- Xavier initialization sets the variance of weighting according to input dimension and output dimension, which avoids the output value approaching to zero or one.

$$Var(W_i) = \frac{1}{n_{in} + n_{out}} \quad or \quad Var(W_i) = \frac{1}{n_{in}}$$



Tanh Activation

Batch Normalization

- Batch Normalization normalizes the distribution for each layer then follow by a trainable scale and shift parameter.

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

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

Calculate in training time.
Fixed in testing time

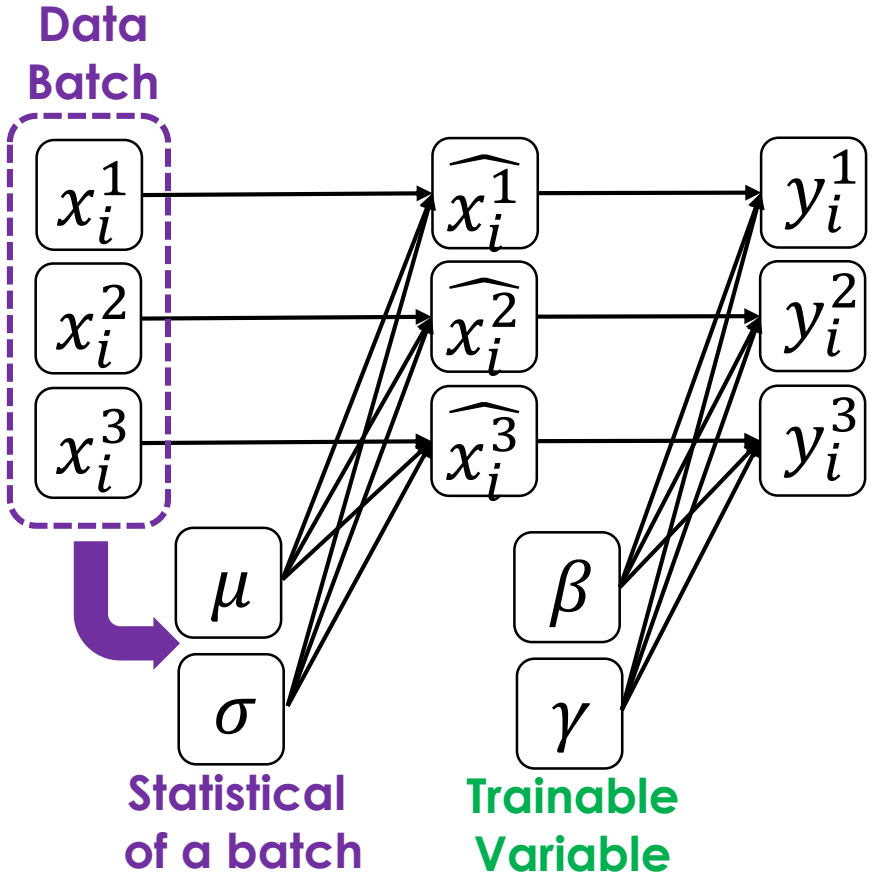
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

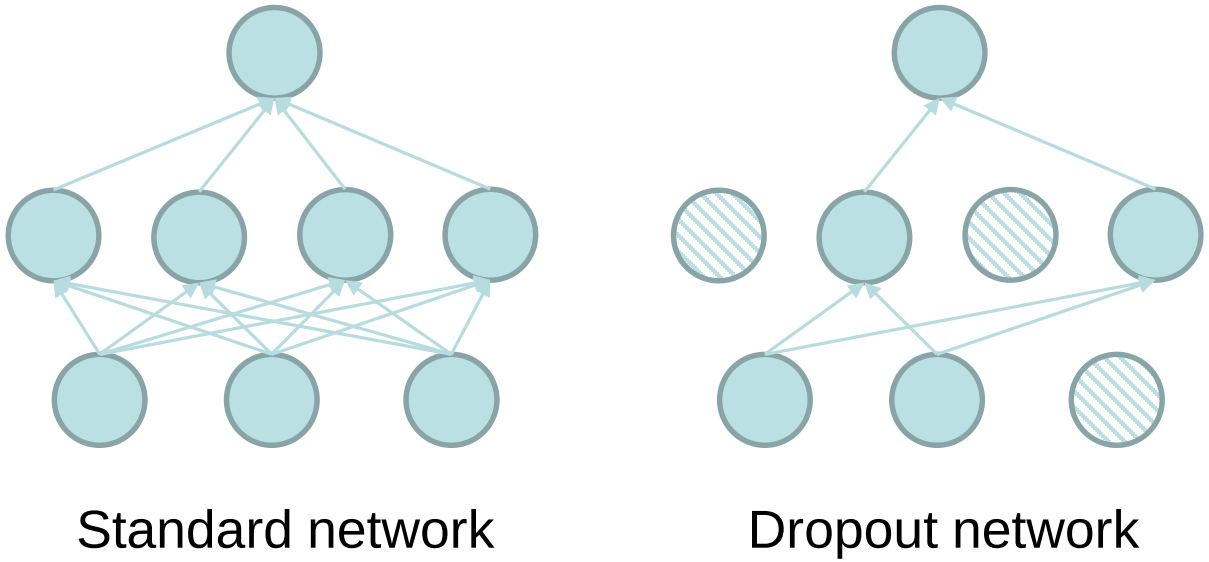
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



Dropout

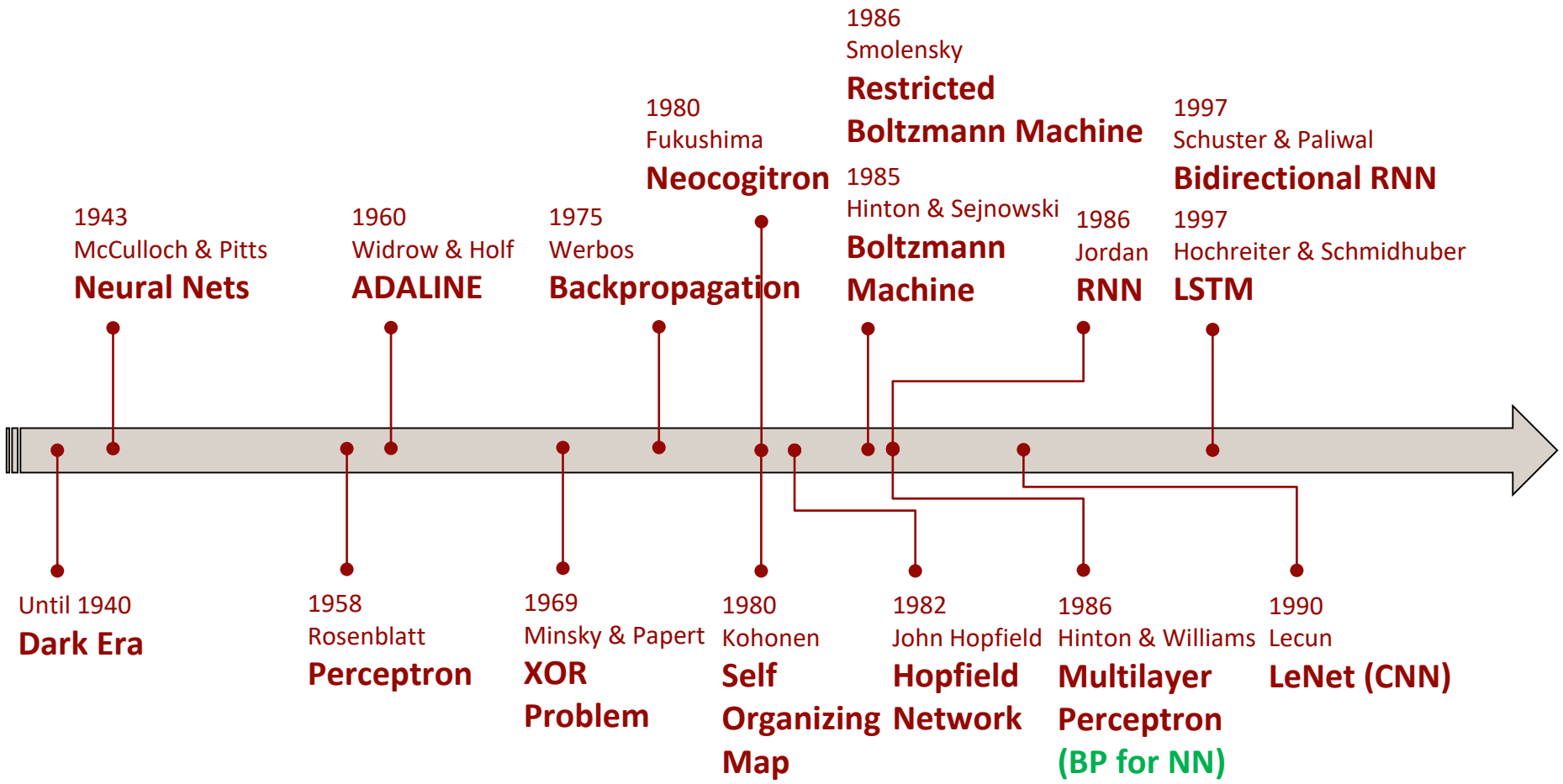
- In training phase, drop some visible and hidden units to make learned features be more general
  - Avoid that some units may bias learning
- At test phase, use weight to mimic the dropout in training
  - By averaging the outputs of several dropout networks, it would approximate the output of the traditional neural network



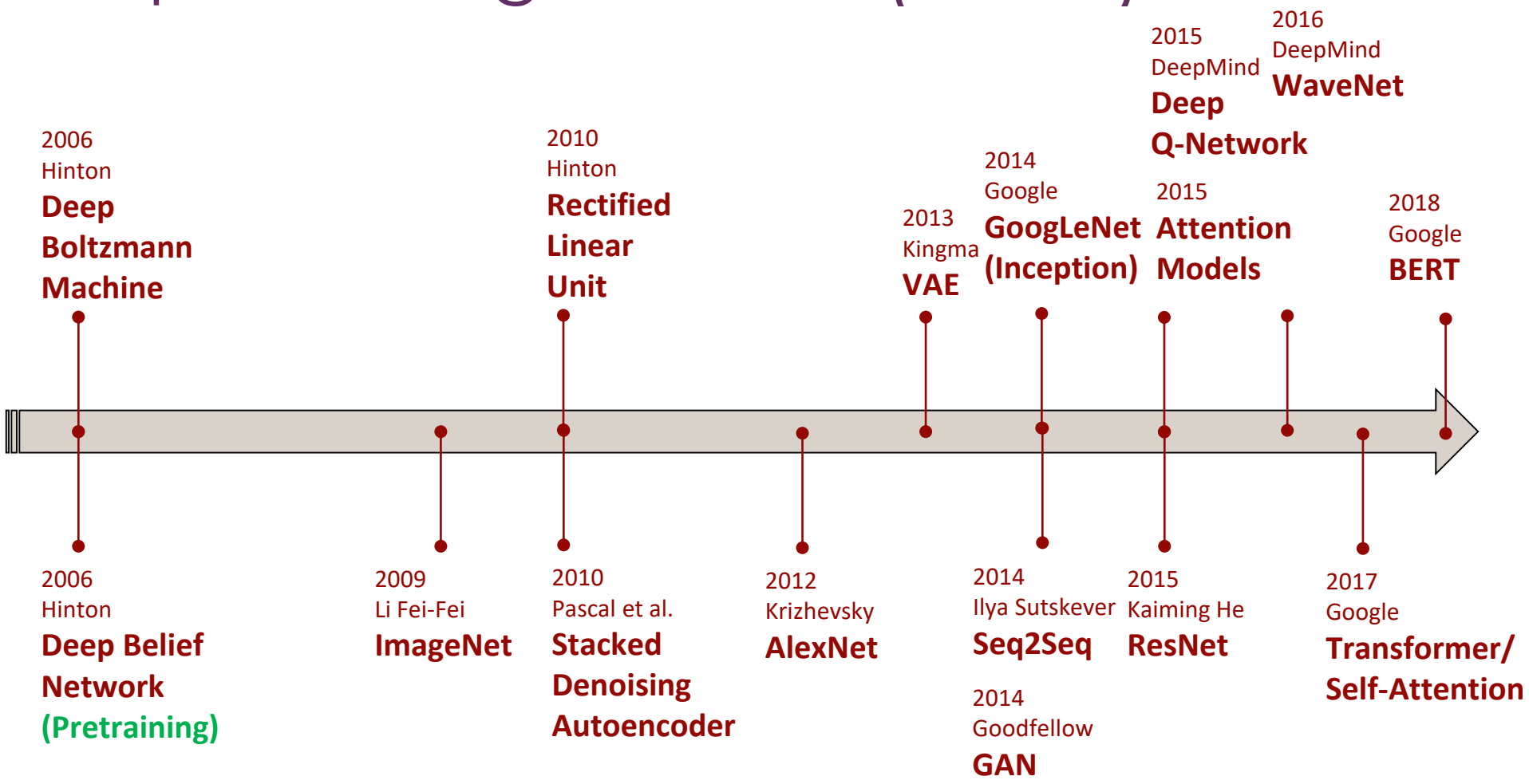
Standard network

Dropout network

Deep Learning Timeline

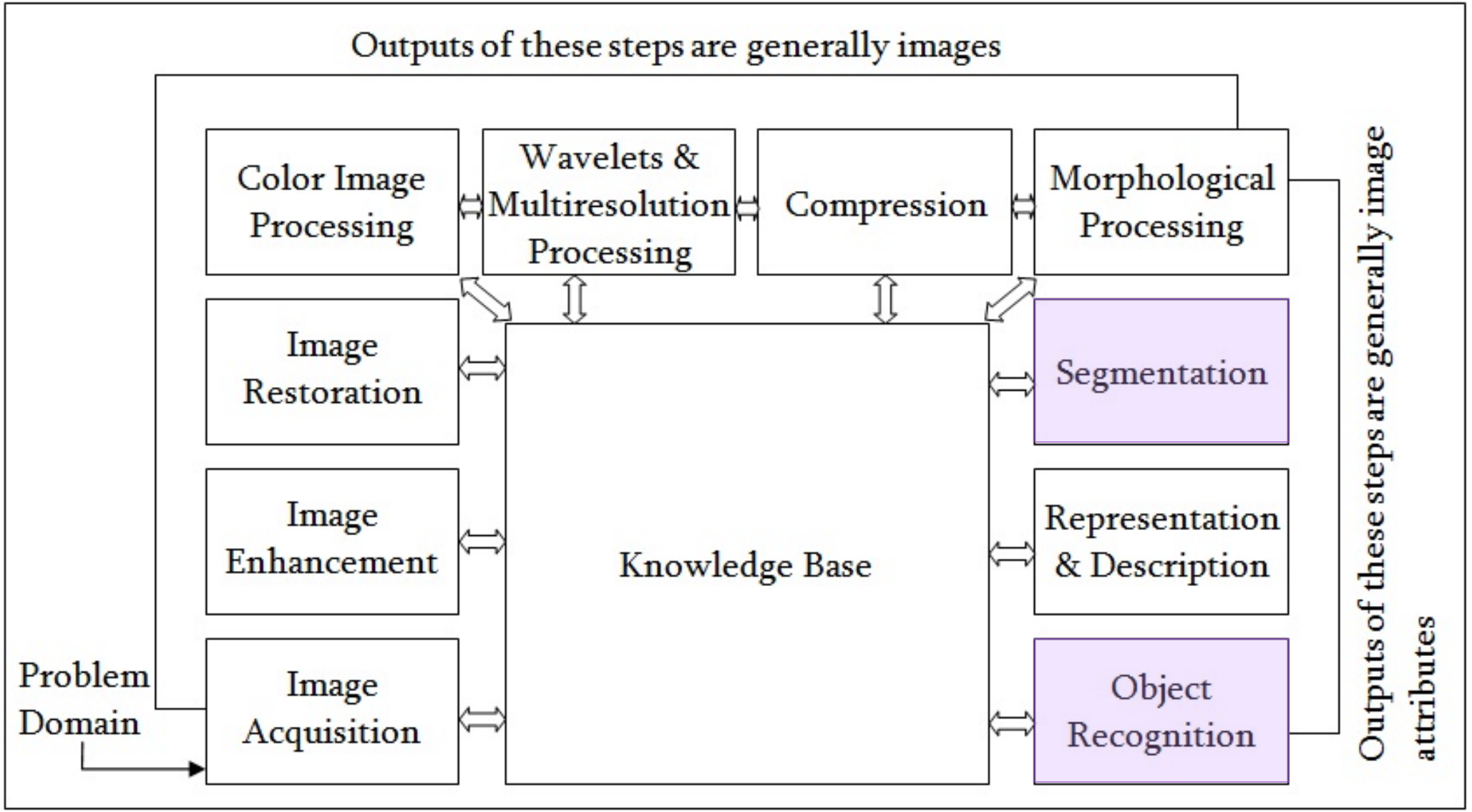


Deep Learning Timeline (Cont.)



Application to Images

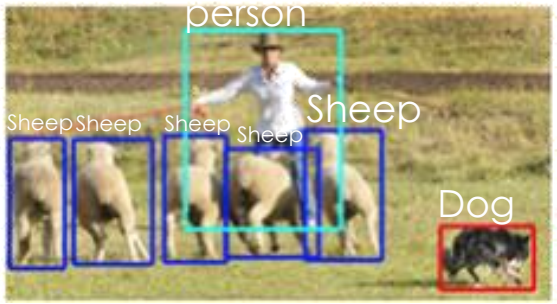
Fundamental Steps in DIP



Computer Vision Problems

- **Object Classification**
  - **Is a target object in the image?**

- **Object Detection/Recognition**
  - **Locate the bounding box of the target object(s).**

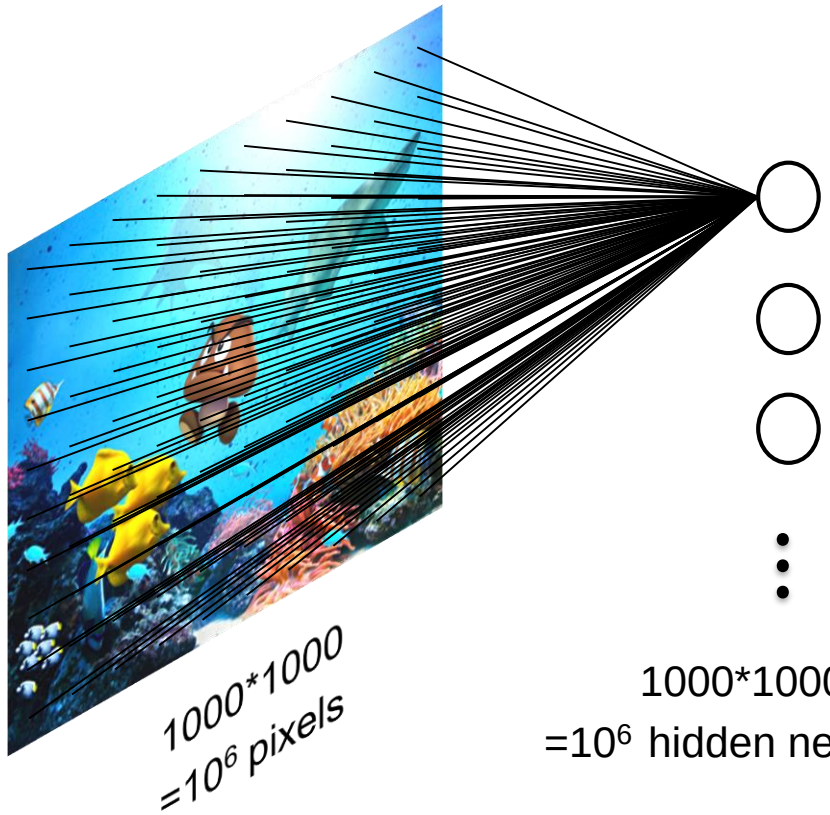


- **Semantic Segmentation**
  - **Segment each kind of object pixel-by-pixel**

- **Instance Segmentation**
  - **Segment each object instance pixel-by-pixel**

Convolutional Neural Network (CNN)

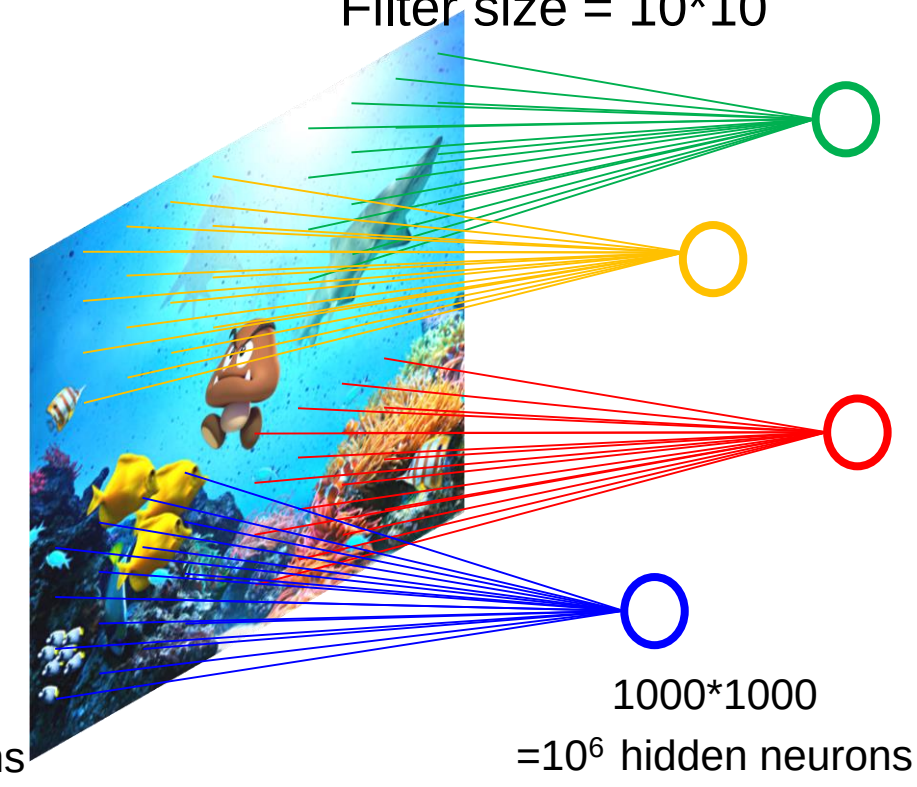
Fully Connected Neural Net



$10^6 \times 10^6 = 10^{12}$ parameters

Locally Connected Neural Net

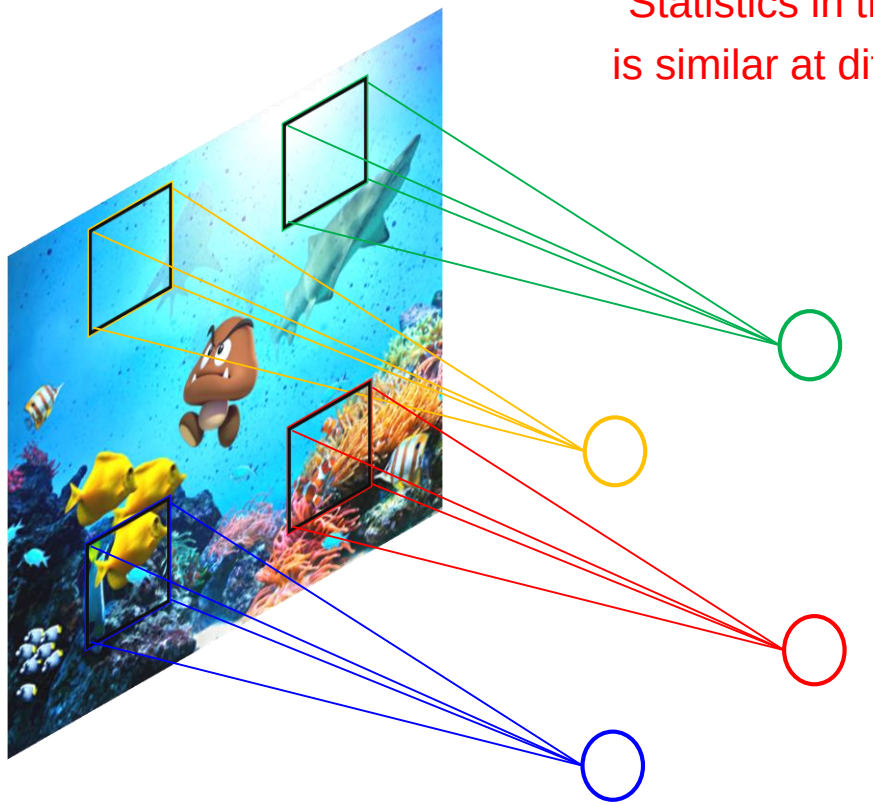
Filter size = 10*10



$(10 \times 10) \times 10^6 = 10^8$ parameters

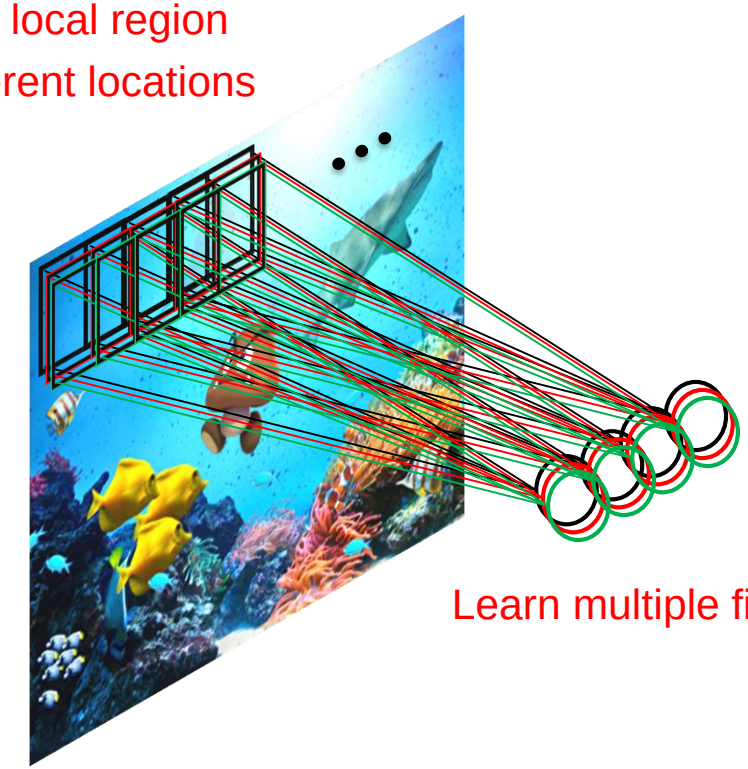
Convolutional Neural Network (CNN)

Locally Connected Neural Net



Convolutional Net

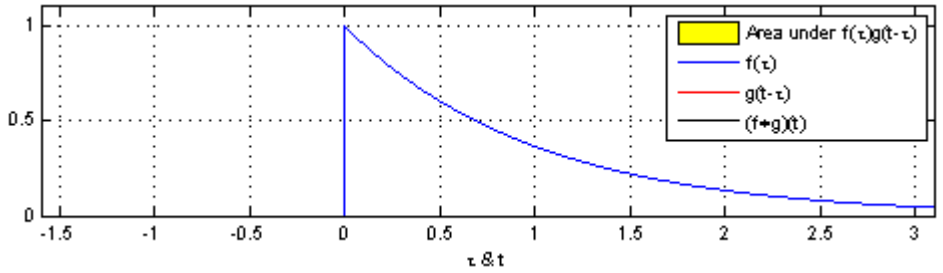
Statistics in the local region
is similar at different locations

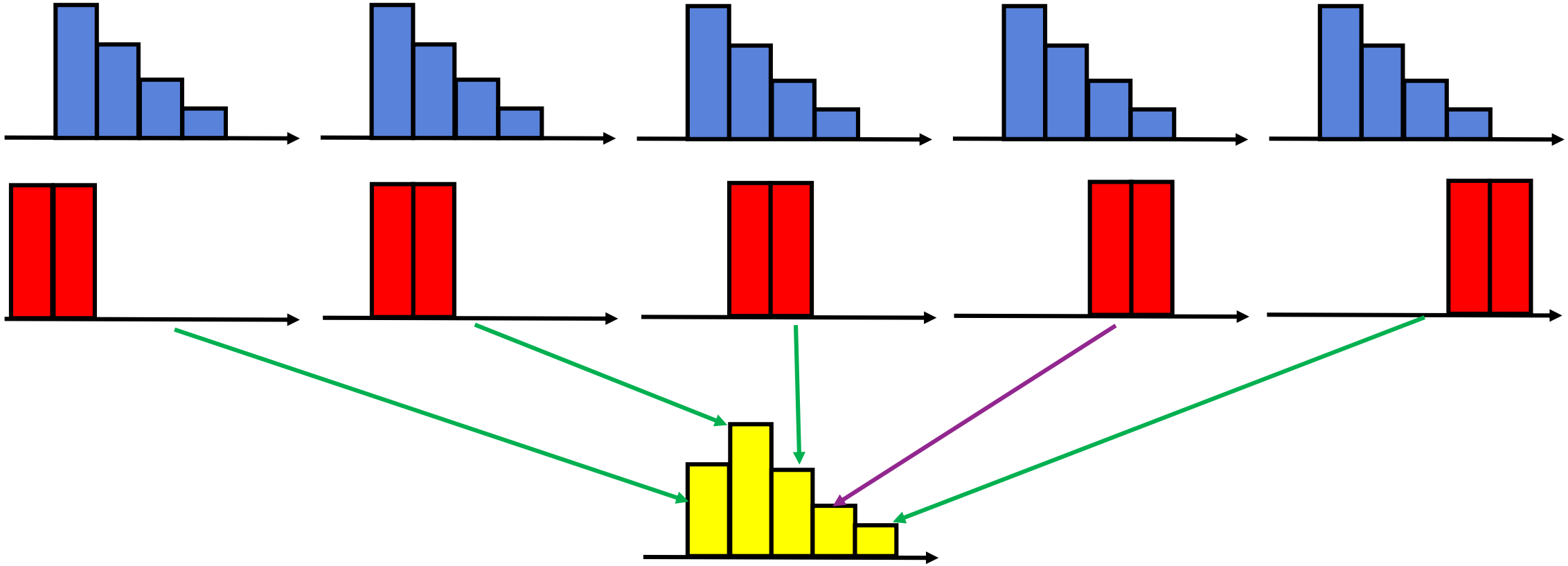


Learn multiple filters

Continuous-Time Convolution

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)\,\mathrm{d}\tau$$





Convolutional Operation

1	3	2	5
7	4	3	2
2	3	1	6
7	6	5	4

Input

0.1	0.5	0.3
0.3	0.4	0.7
0.6	0.2	0.8

Kernel

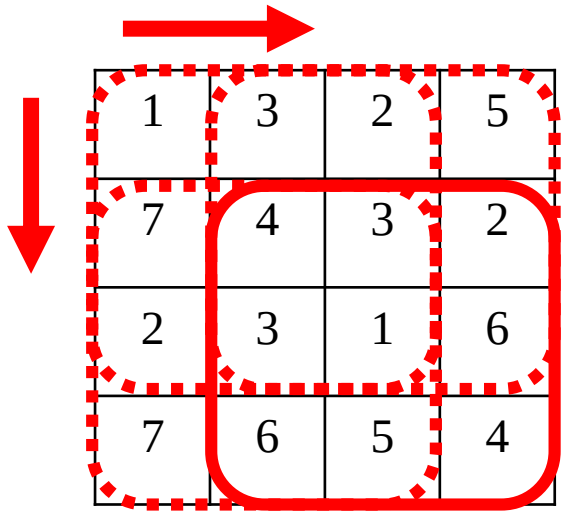
1	3	2
7	4	3
2	3	1

$\odot$

0.1	0.5	0.3
0.3	0.4	0.7
0.6	0.2	0.8

$$
\begin{aligned}
= \quad & 1 \times 0.1 \\
& +3 \times 0.5 \\
& +2 \times 0.3 \\
& +7 \times 0.3 \\
& +4 \times 0.4 \\
& +3 \times 0.7 \\
& +2 \times 0.6 \\
& +3 \times 0.2 \\
& +1 \times 0.8
\end{aligned}
= 10.6
$$

Convolutional Operation



Input

$\odot$

0.1	0.5	0.3
0.3	0.4	0.7
0.6	0.2	0.8

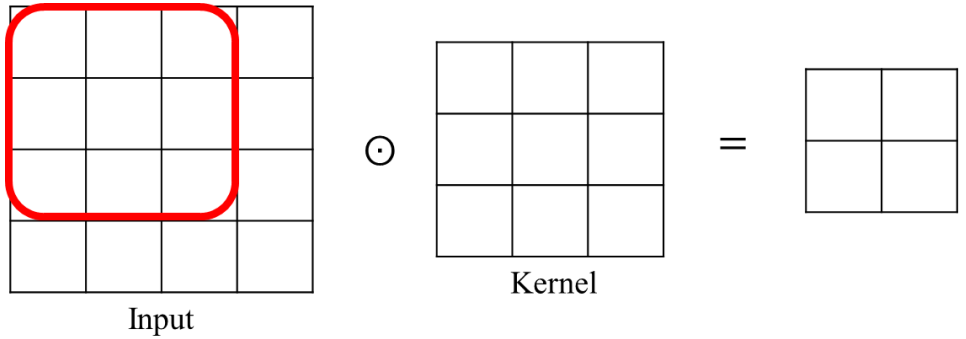
Kernel

$=$

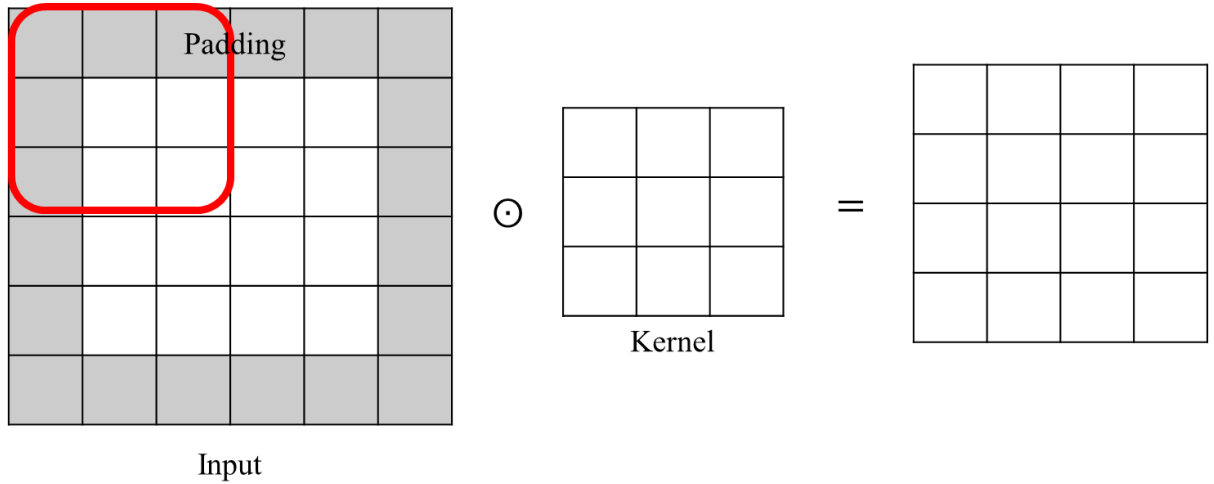
10.6	13.4
15.5	15.8

Convolutional Operation

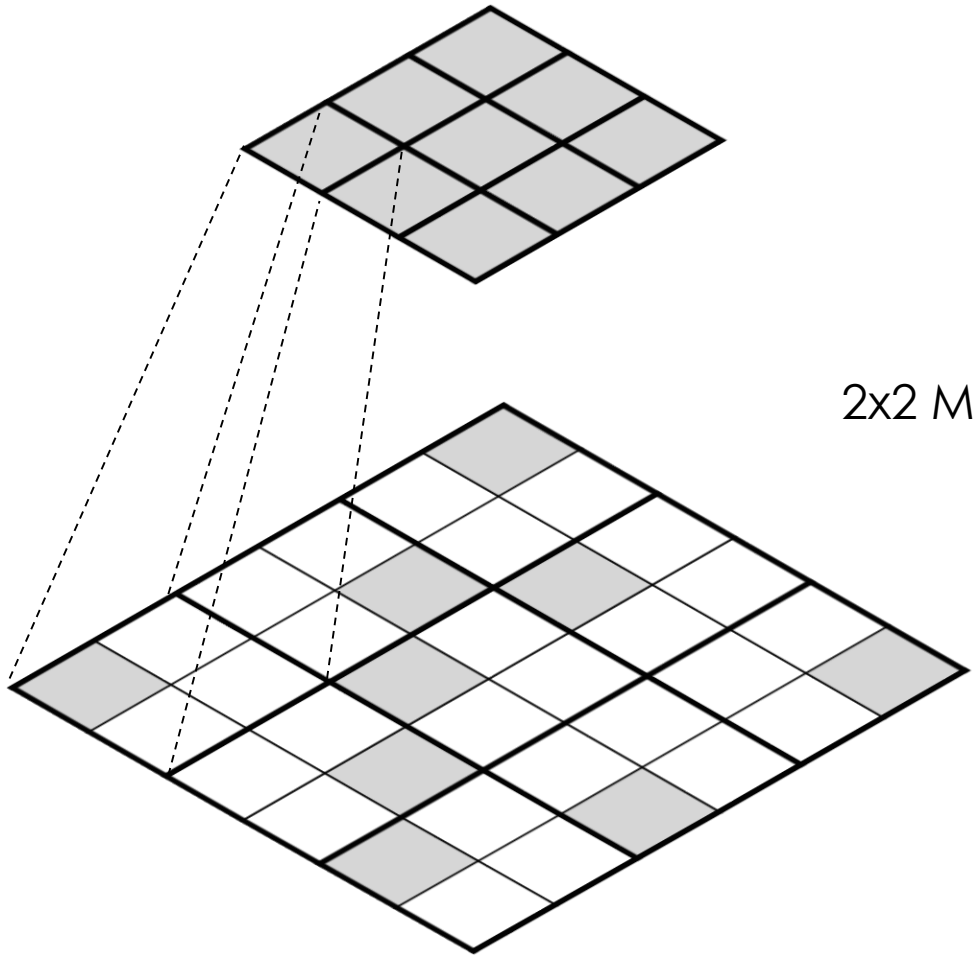
- Standard Convolution (In TensorFlow, padding='VALID')



- Padding (In TensorFlow, padding='SAME')

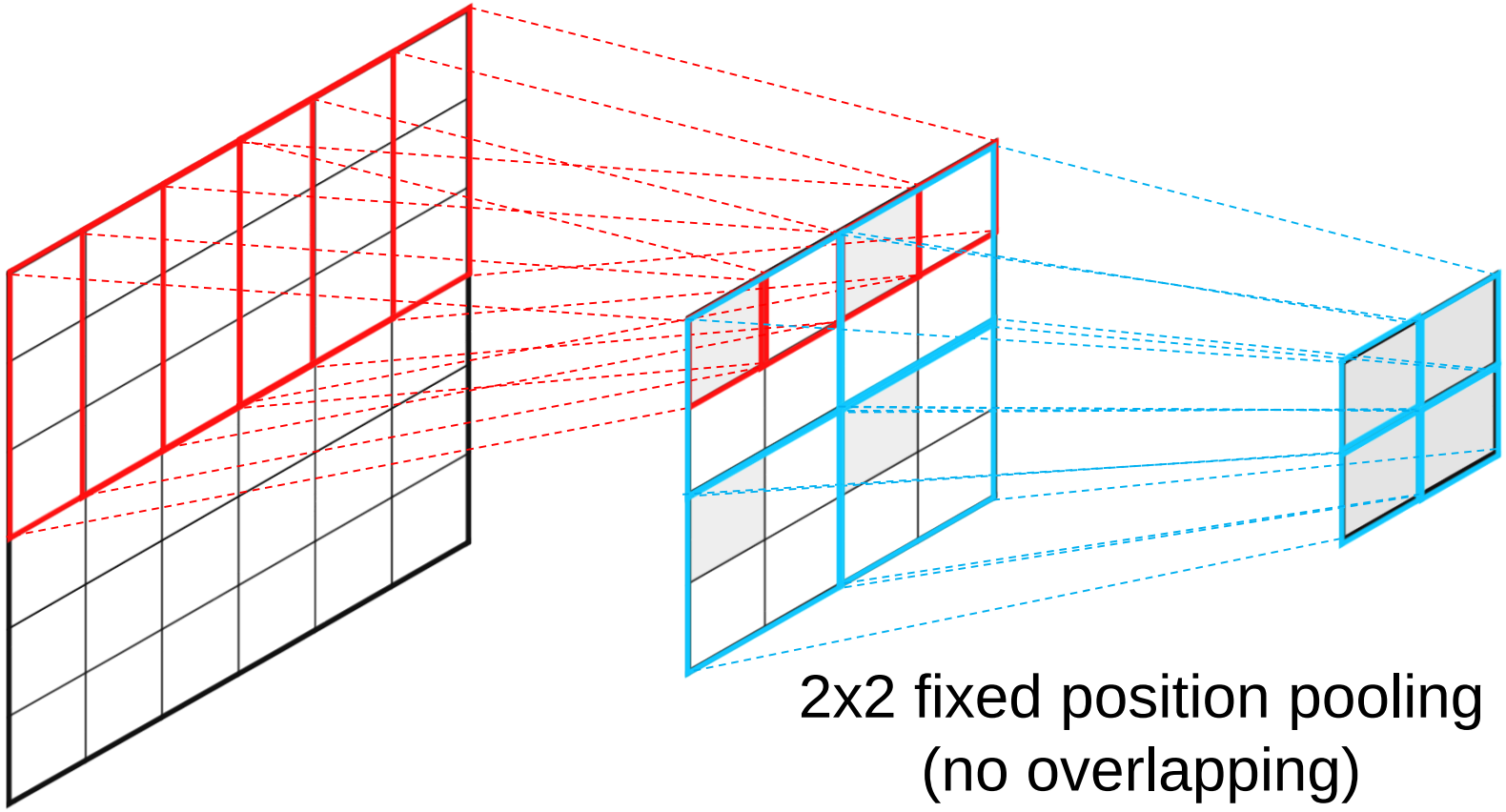


Pooling



2x2 Max pooling

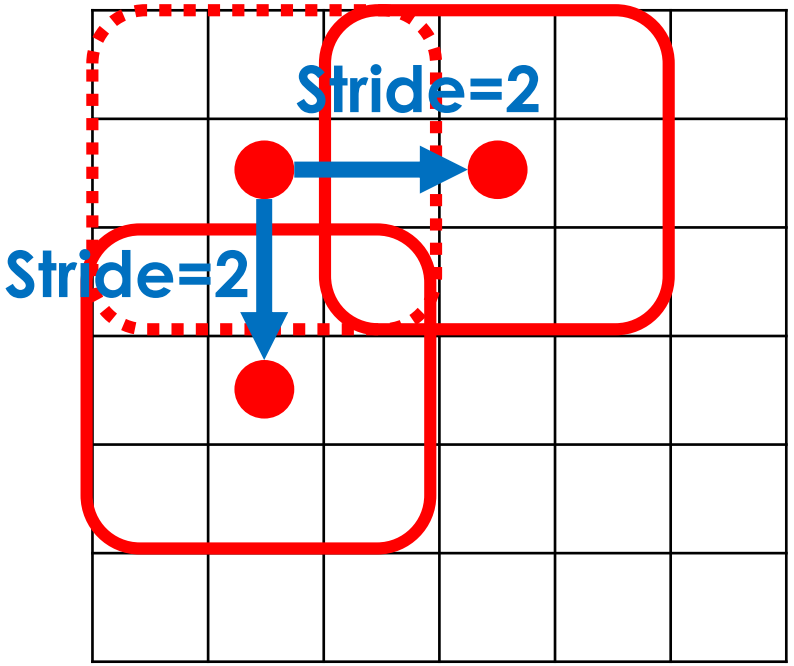
Convolution + Pooling



2x2 fixed position pooling
(no overlapping)

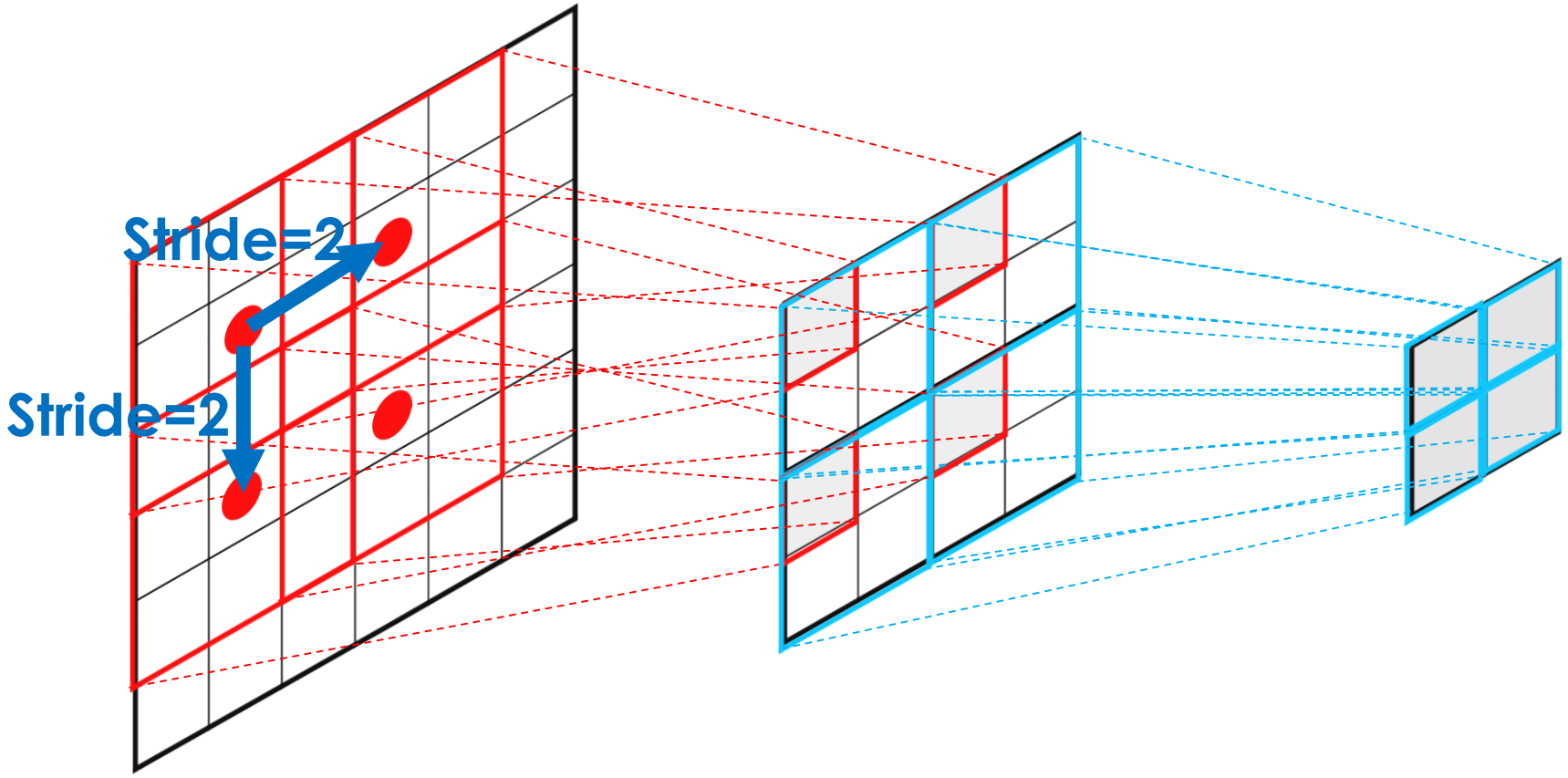
Convolution with stride=1

Stride Convolution



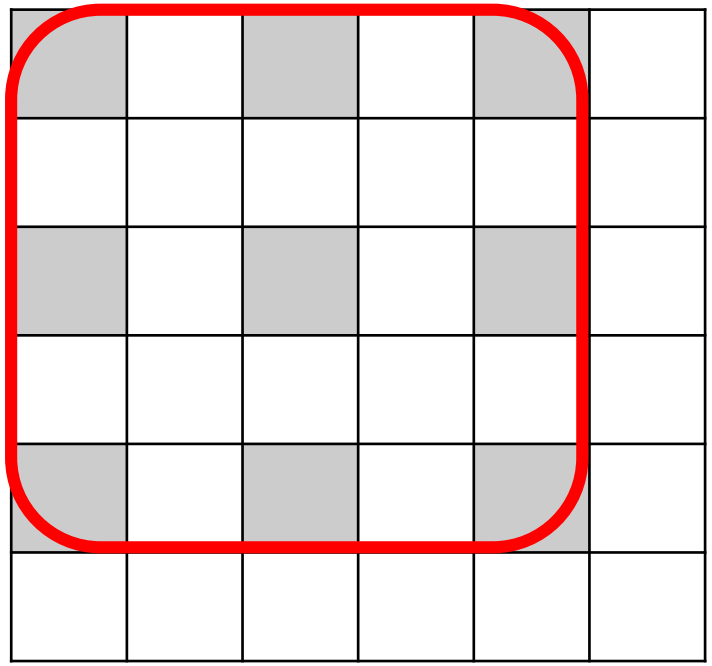
Can replace the pooling operation
(Usually be used in generative model)

Convolution+ Pooling

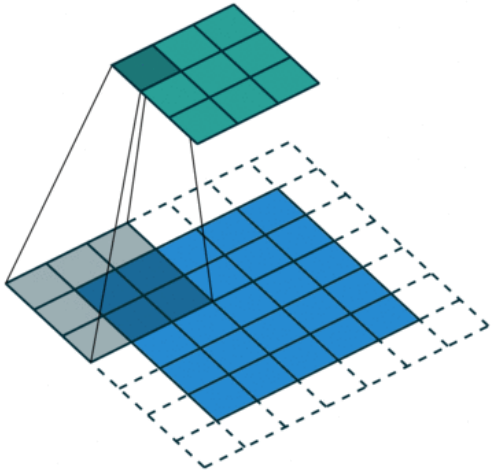


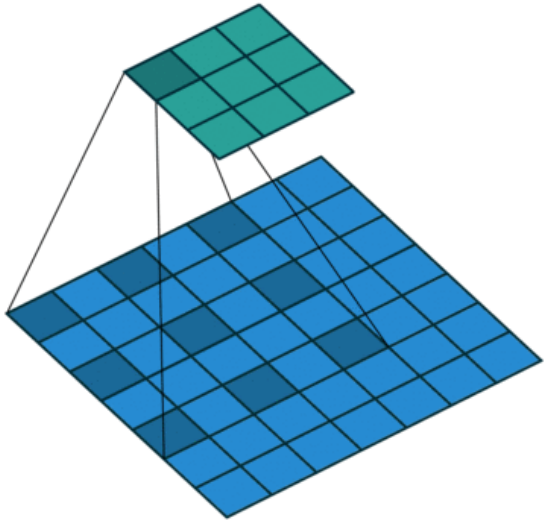
Equal to Convolution with stride=1 + 2x2 fixed position pooling (no overlapping)

Dilated Convolution

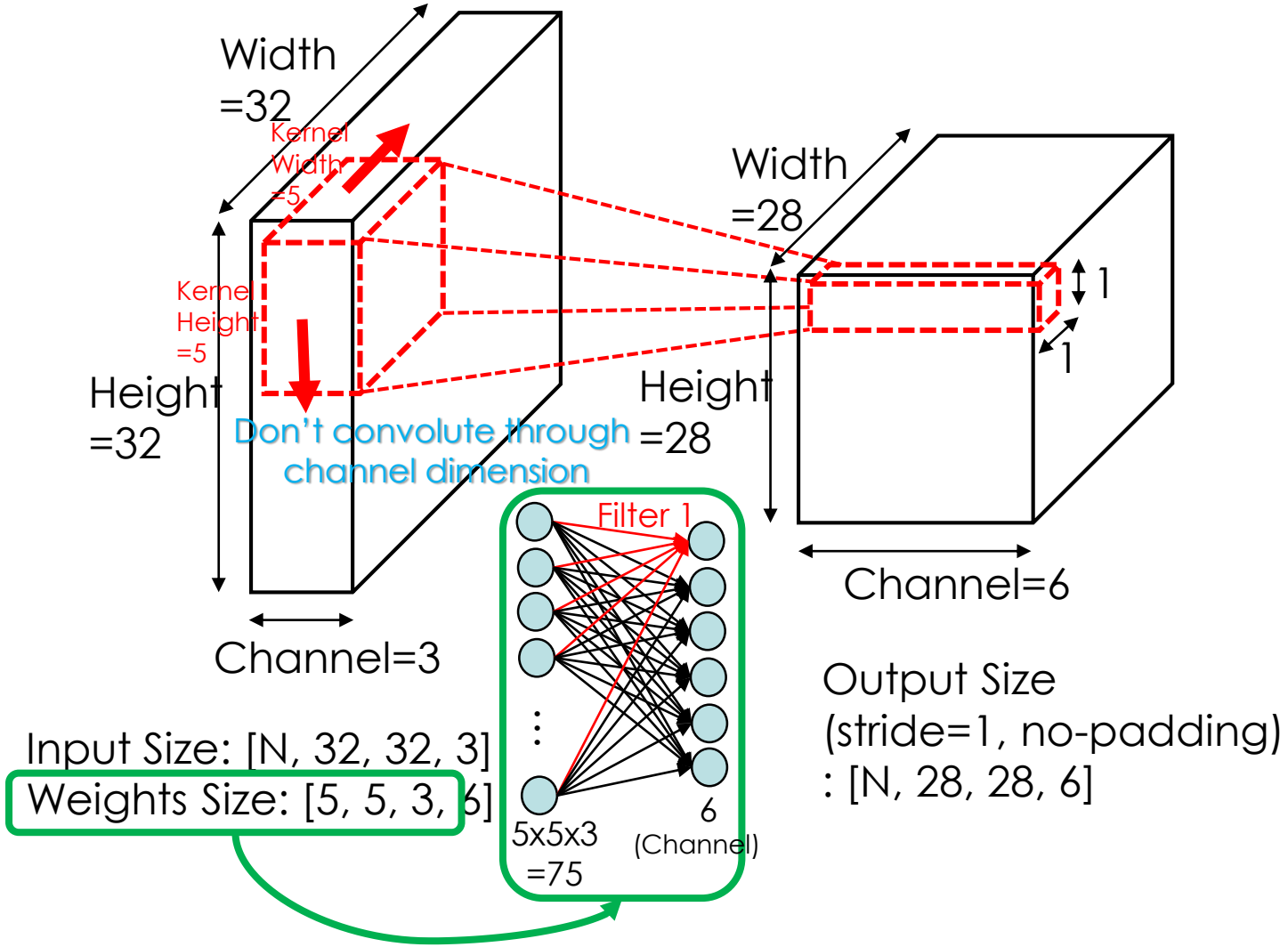


Increase the Receptive Field
(3x3 Kernel with dilation rate 2)

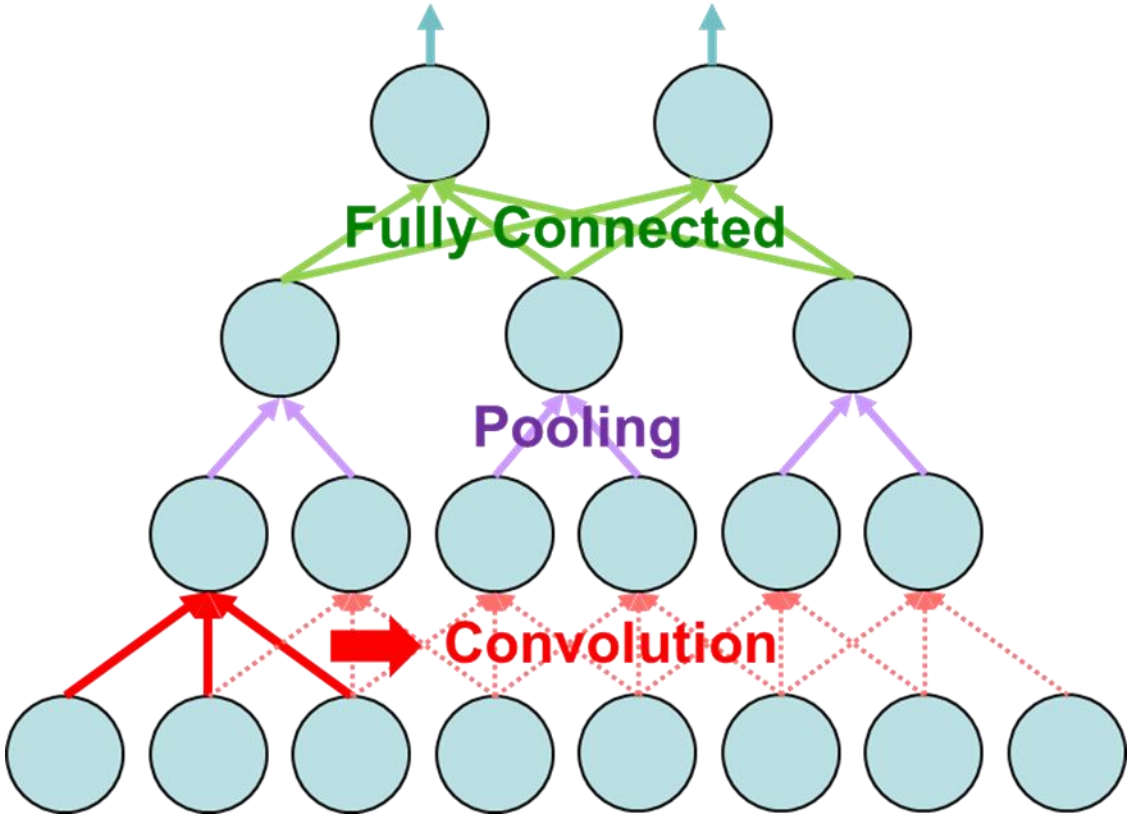




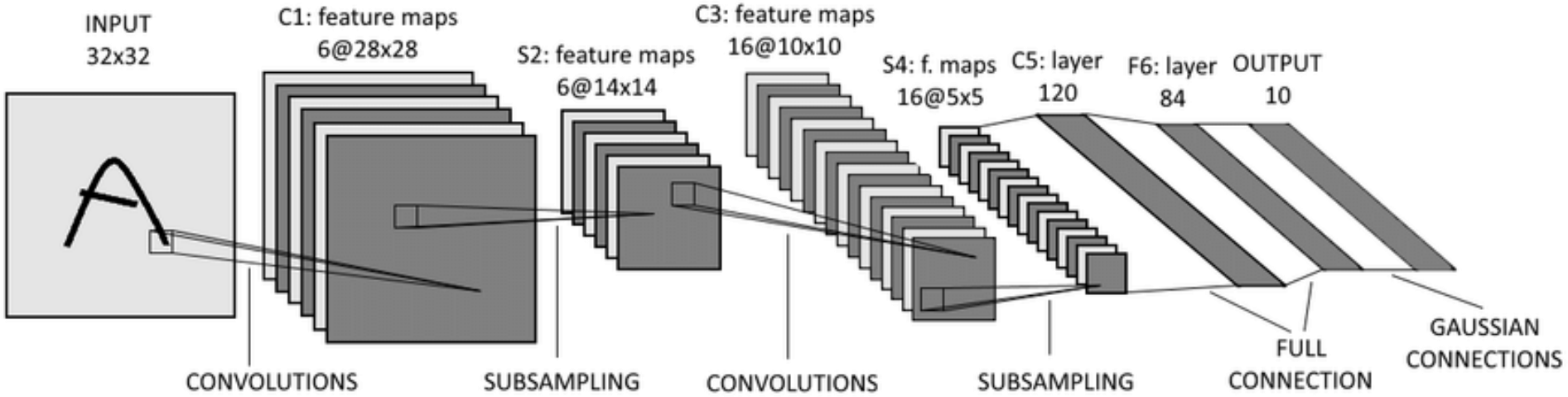
Take a Deep Look at Convolutional Layer

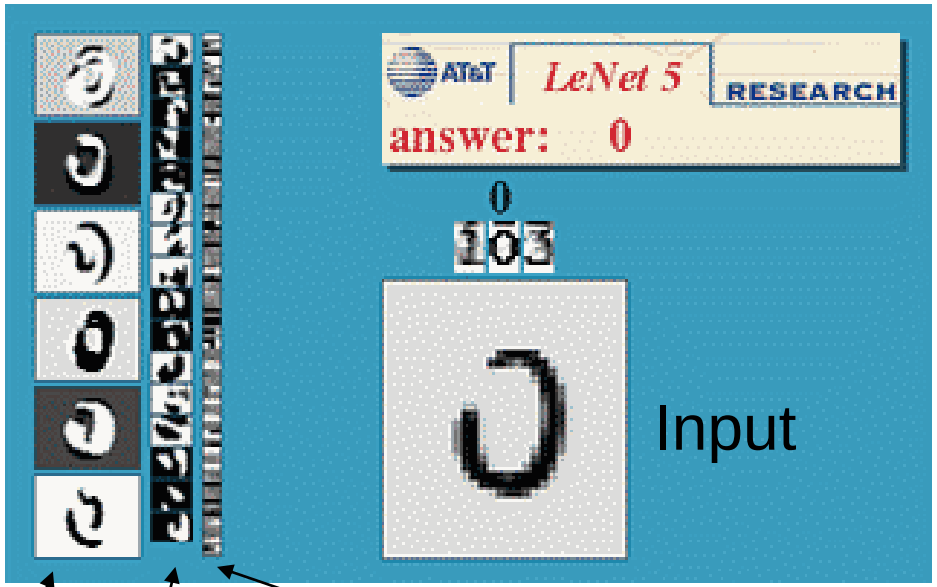


Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN): LeNet-5





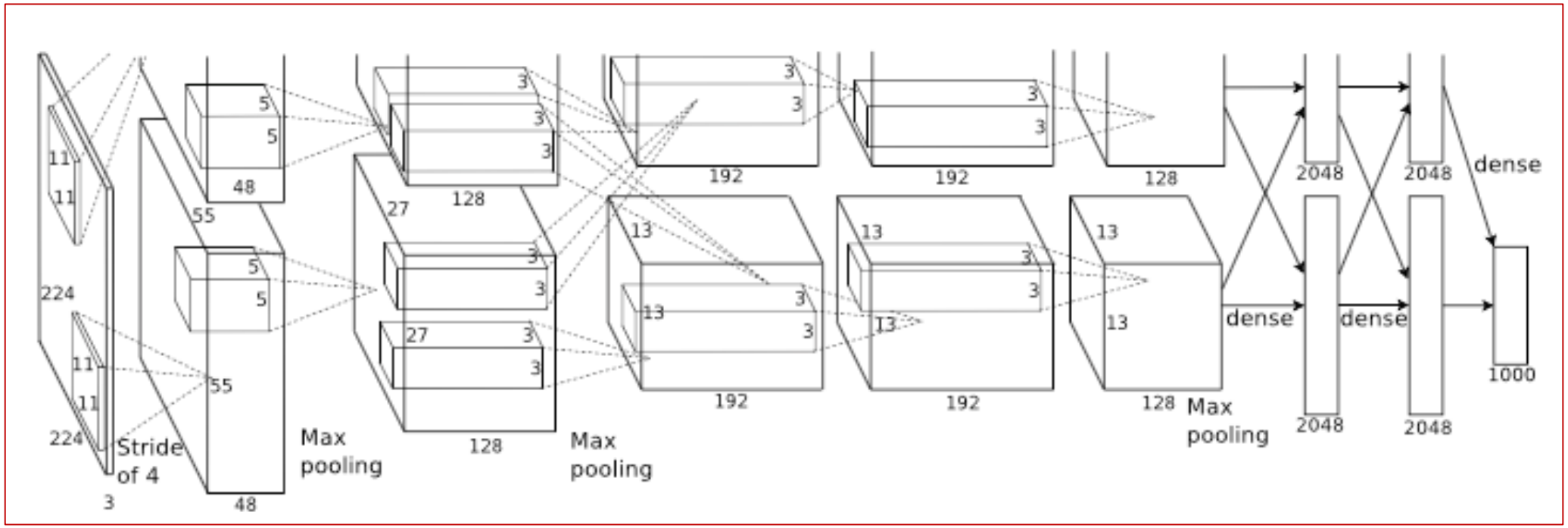
[Ref] http://yann.lecun.com/exdb/lenet/index.html

ImageNet Dataset

- Over 14,000,000 images
- Over 20,000 classes
- Over 1,000,000 images with bounding boxes
- ILSVRC competition start from 2010 (1000 class category)
- In 2012, AlexNet achieve less than **16%** error rate

AlexNet

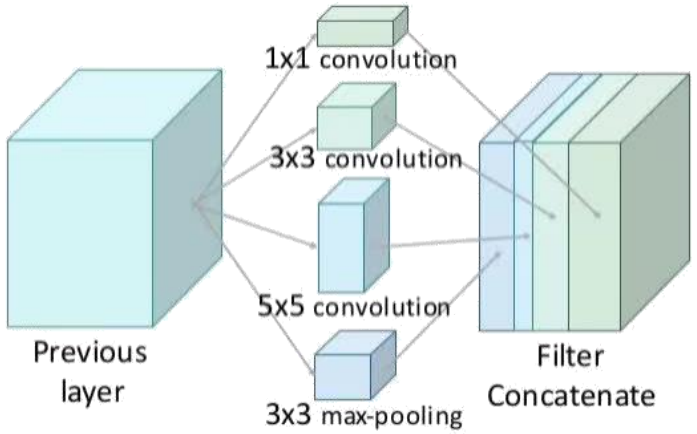
- Alex net (Alex, Hinton, 2012)
  - 5 convolution layers (with max-pooling) and 3 fully-connected neural nets
  - Use dropout and LRN (Local Response Normalization)
  - GPU Acceleration

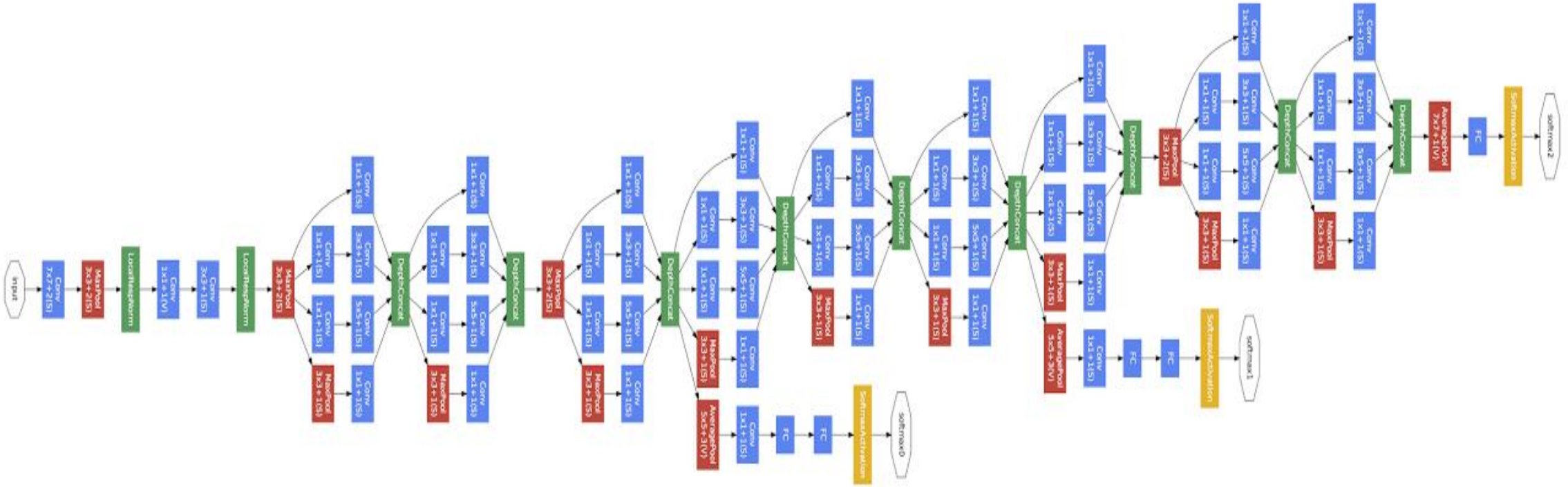


GoogLeNet

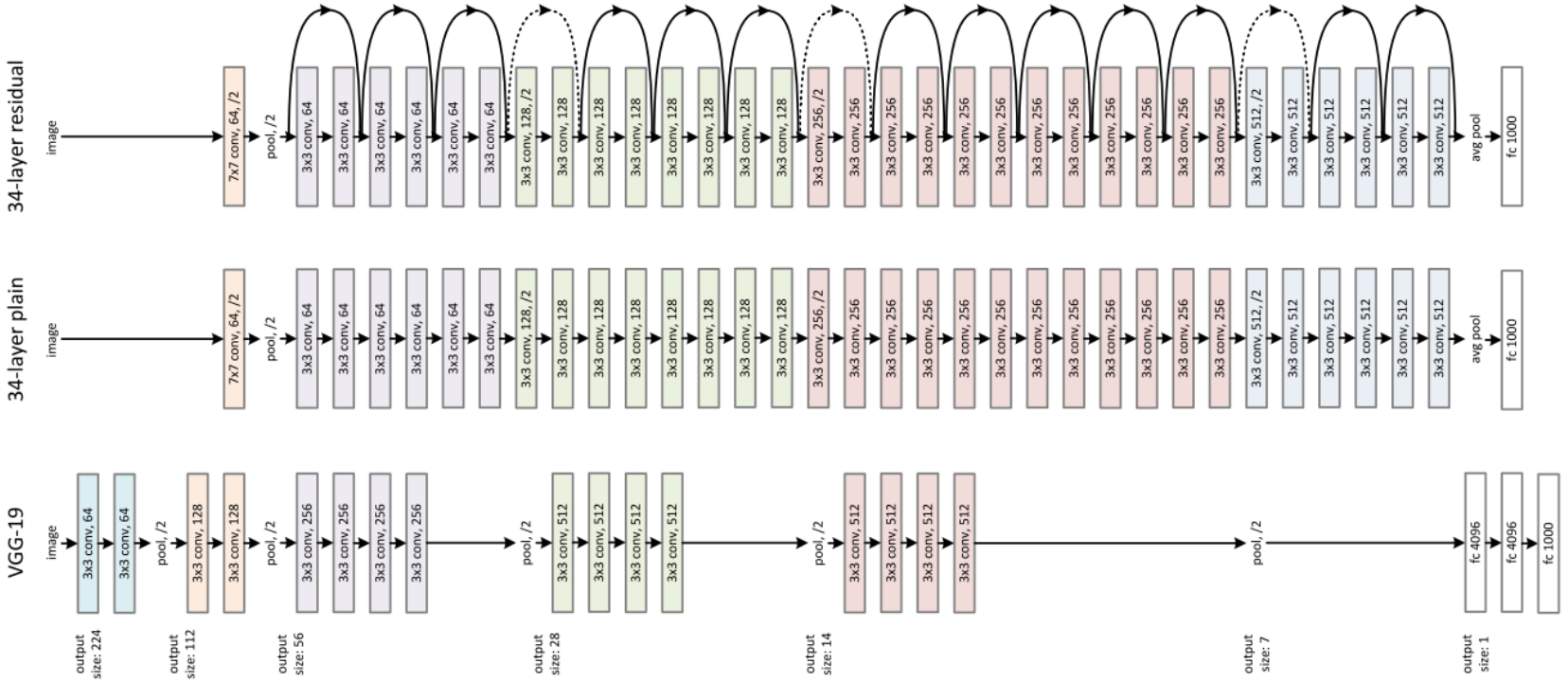
- GoogLeNet (Google Inc. 2014)
  - 22 layers
  - 9 inception modules (includes 2 layers)

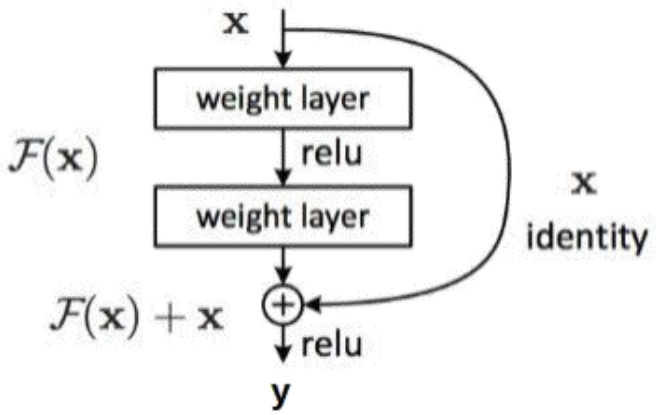
Inception Model





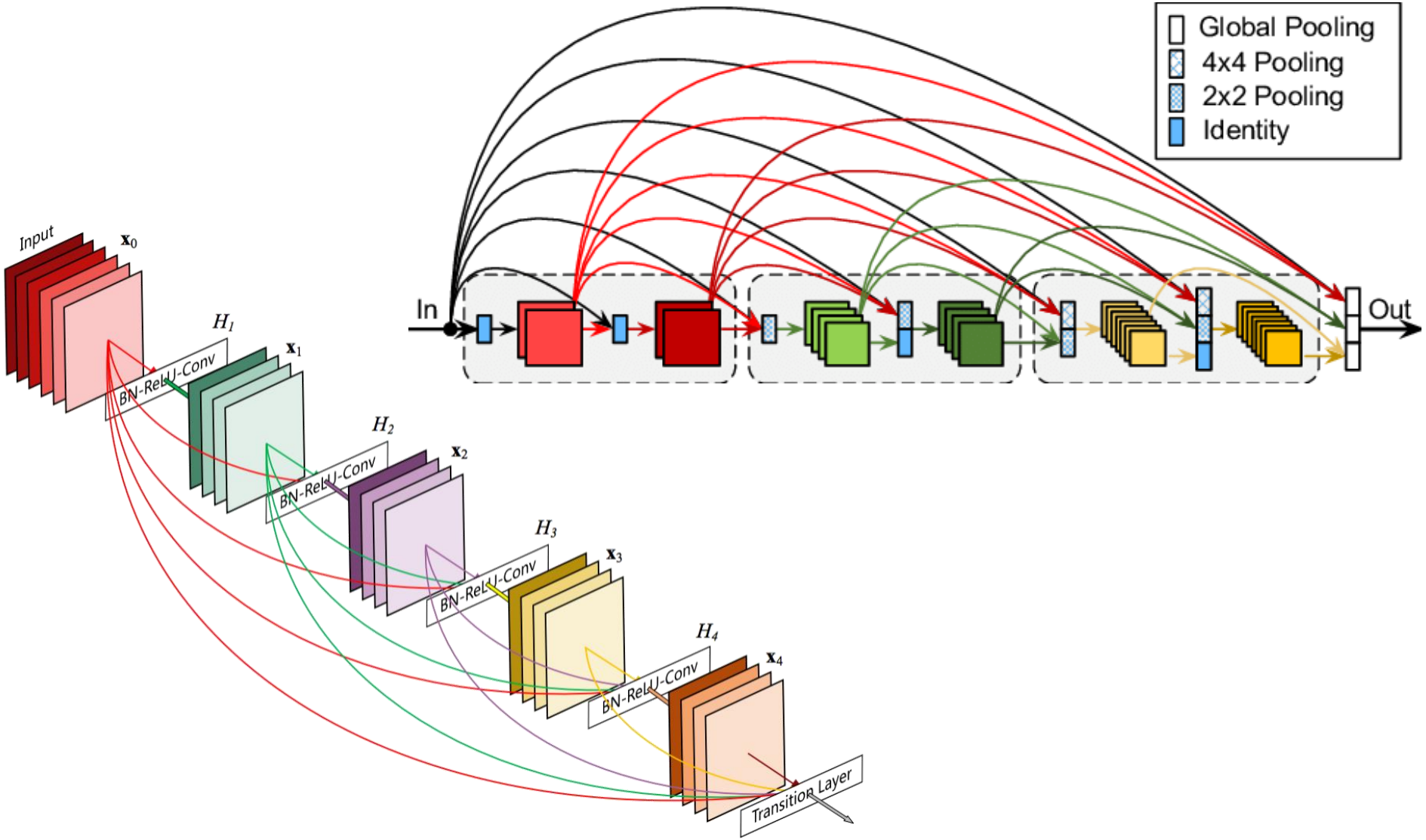
ResNet



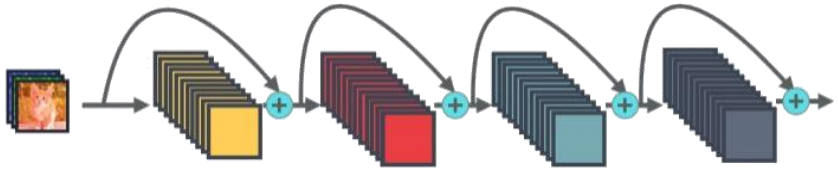


$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i, W_i)\right)$$

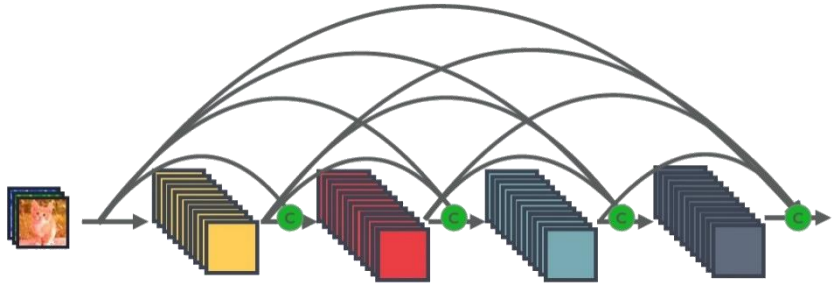
DenseNet


Global Pooling
4x4 Pooling
2x2 Pooling
Identity
In
Out
Input
$\mathbf{x}_0$
H_1
$\mathbf{x}_1$
H_2
$\mathbf{x}_2$
H_3
$\mathbf{x}_3$
H_4
$\mathbf{x}_4$
BN-ReLU-Conv
Transition Layer

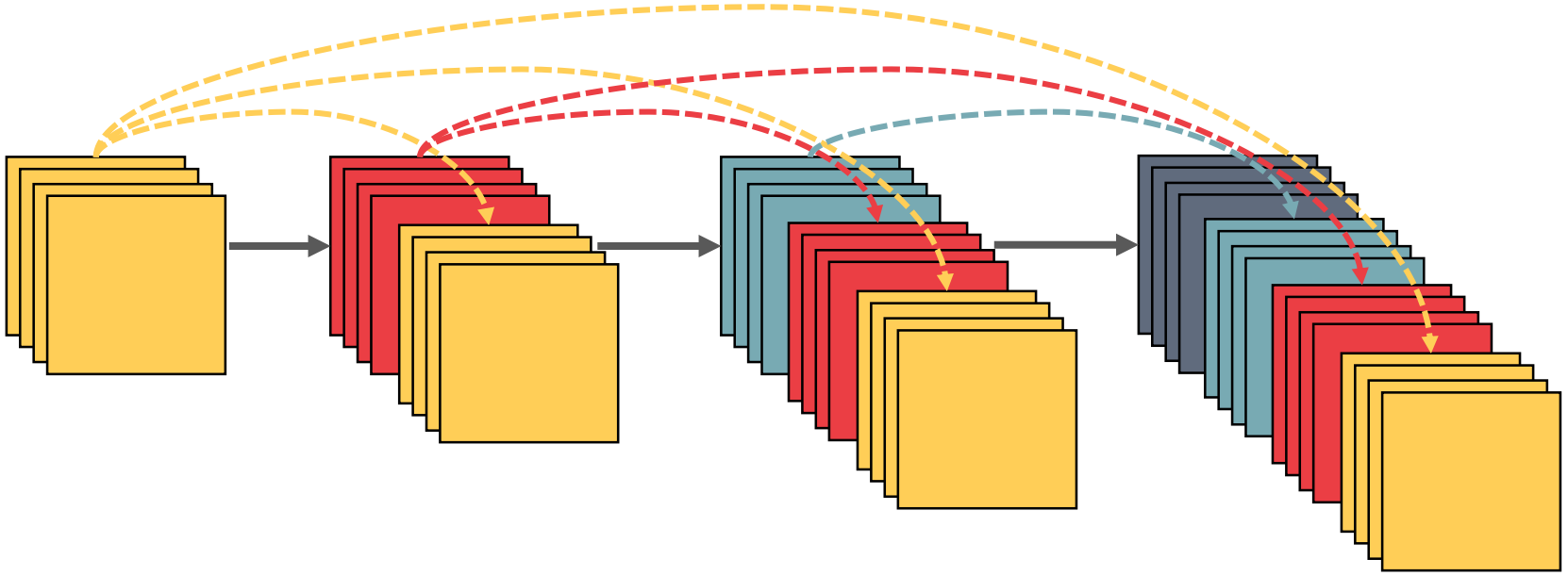
DenseNet



ResNet: Element-wise Addition

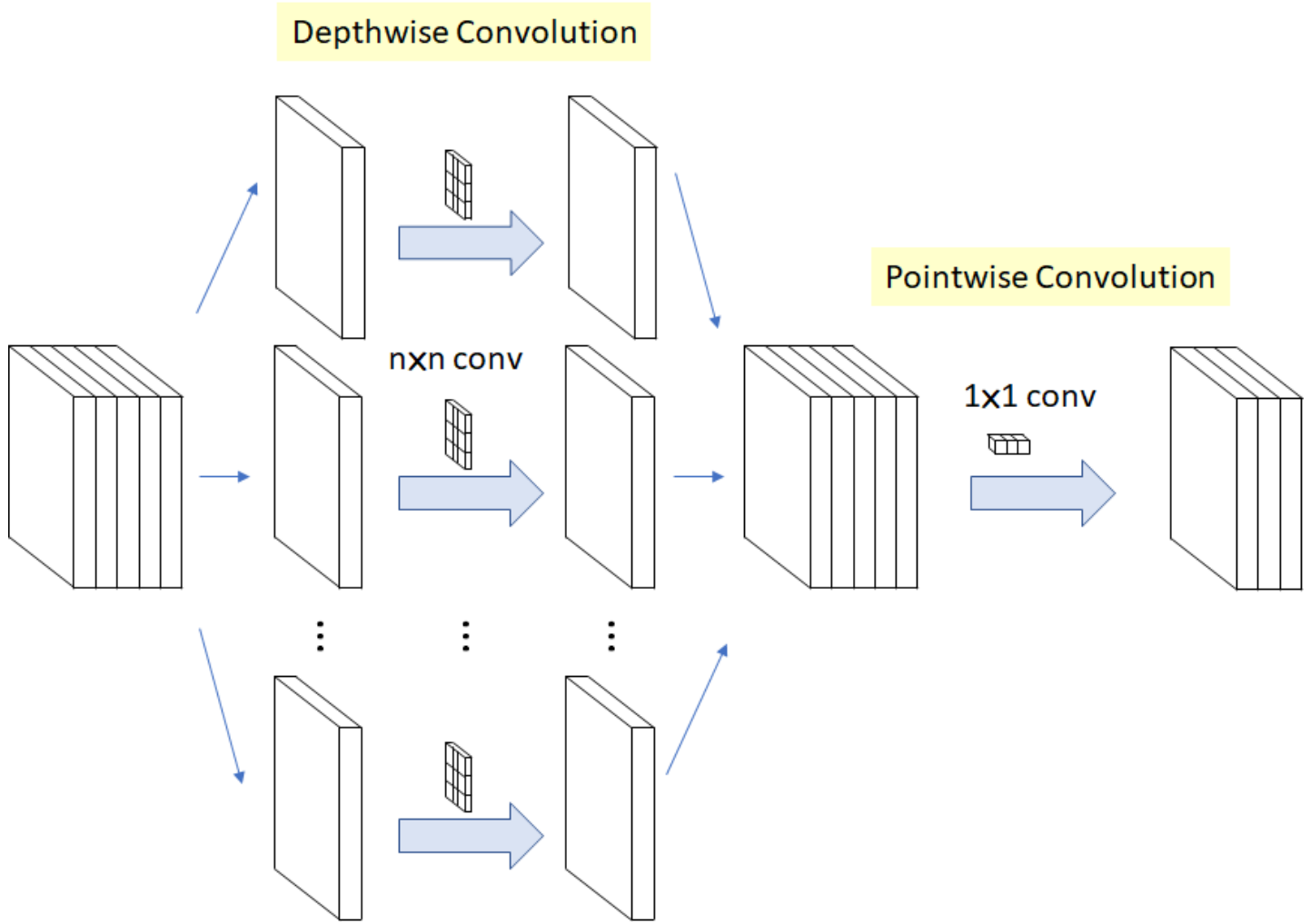


DenseNet: Channel-wise Concatenation



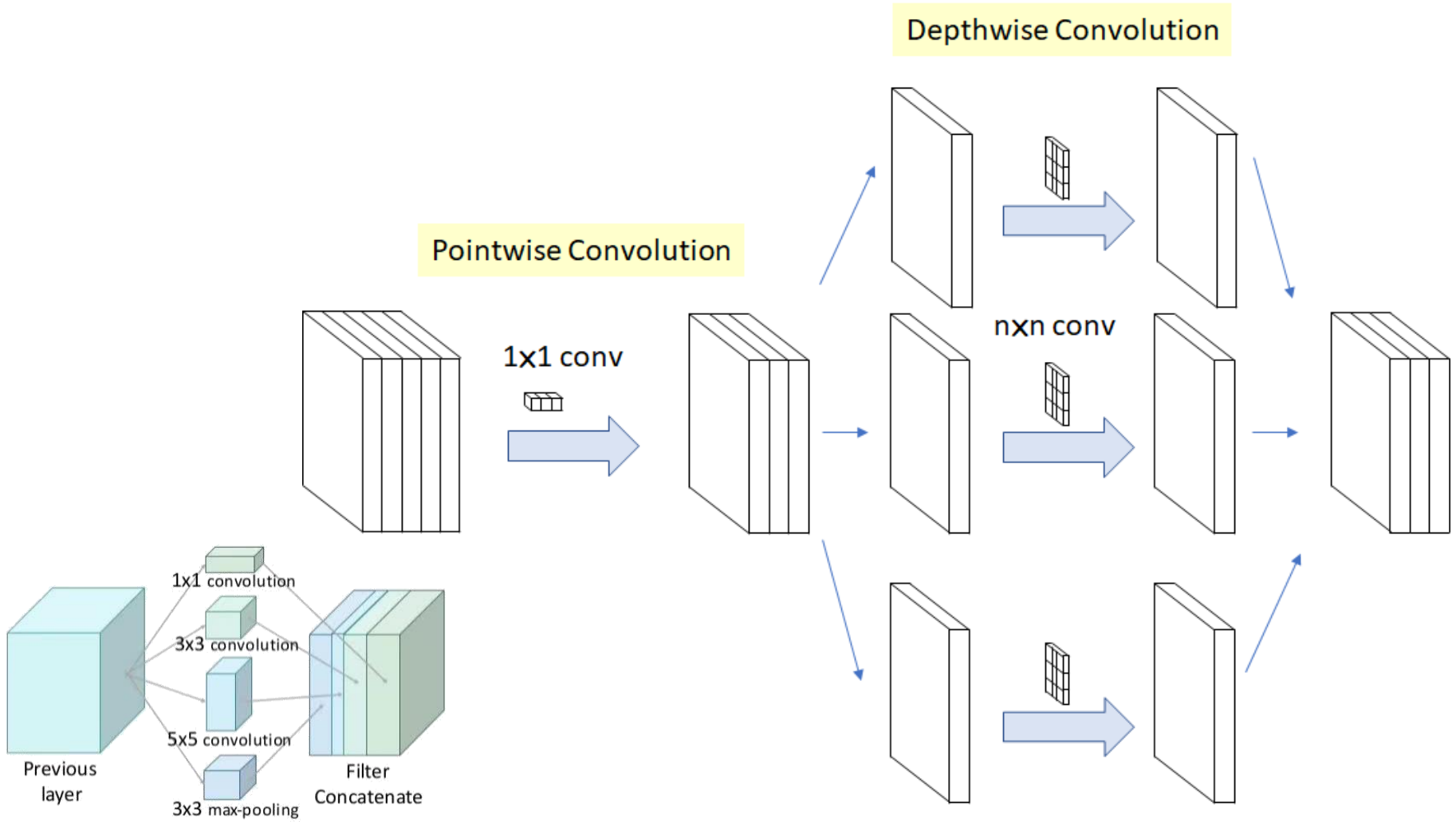
MobileNet

- Depthwise Separable Convolution



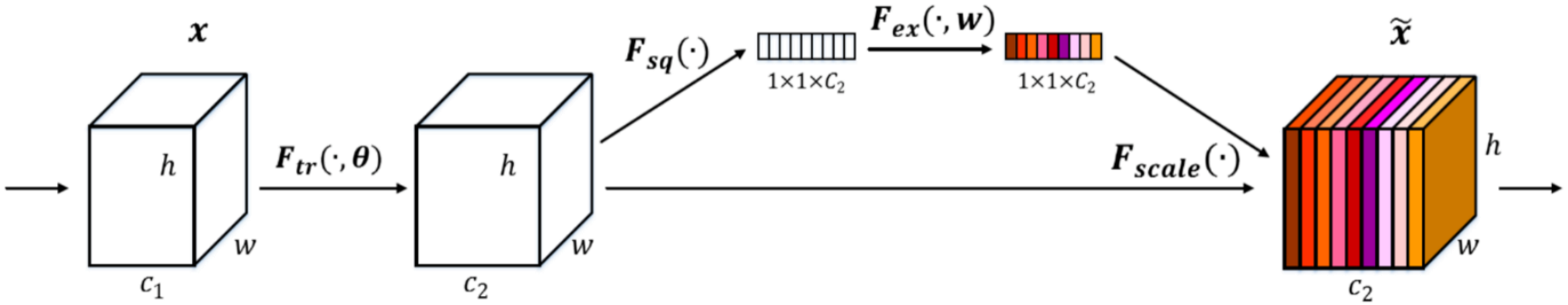
Xception

- Modified Depthwise Separable Convolution



SENet

- Squeeze-and-Excitation Networks



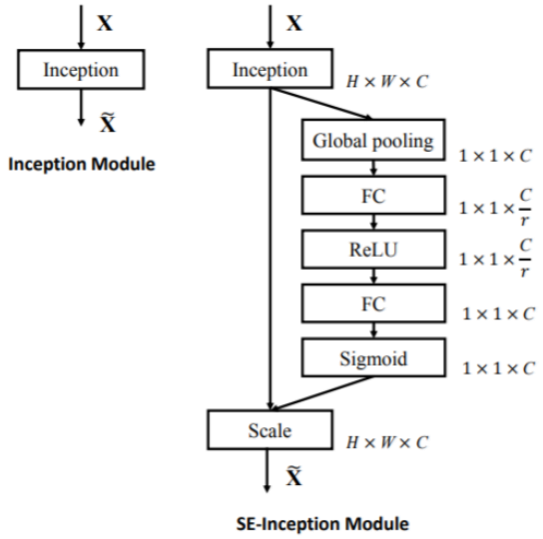


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

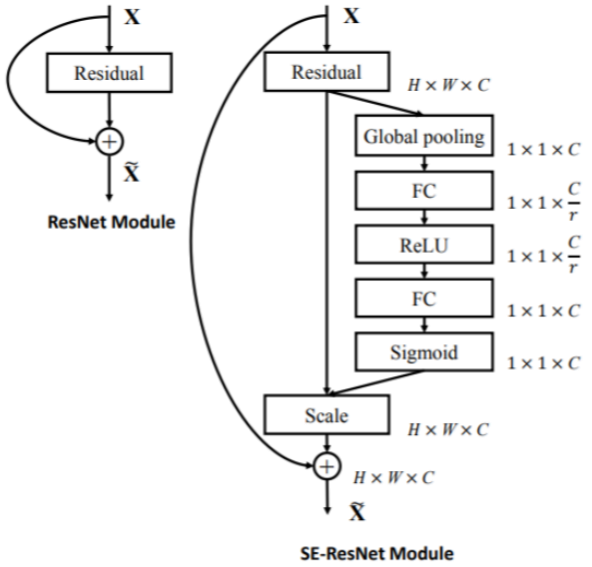
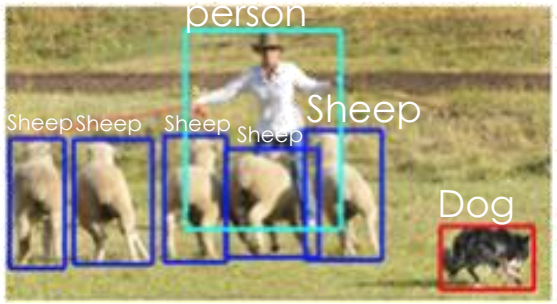


Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

Computer Vision Problems

- **Object Classification**
  - **Is a target object in the image?**

- **Object Detection/Recognition**
  - **Locate the bounding box of the target object(s).**

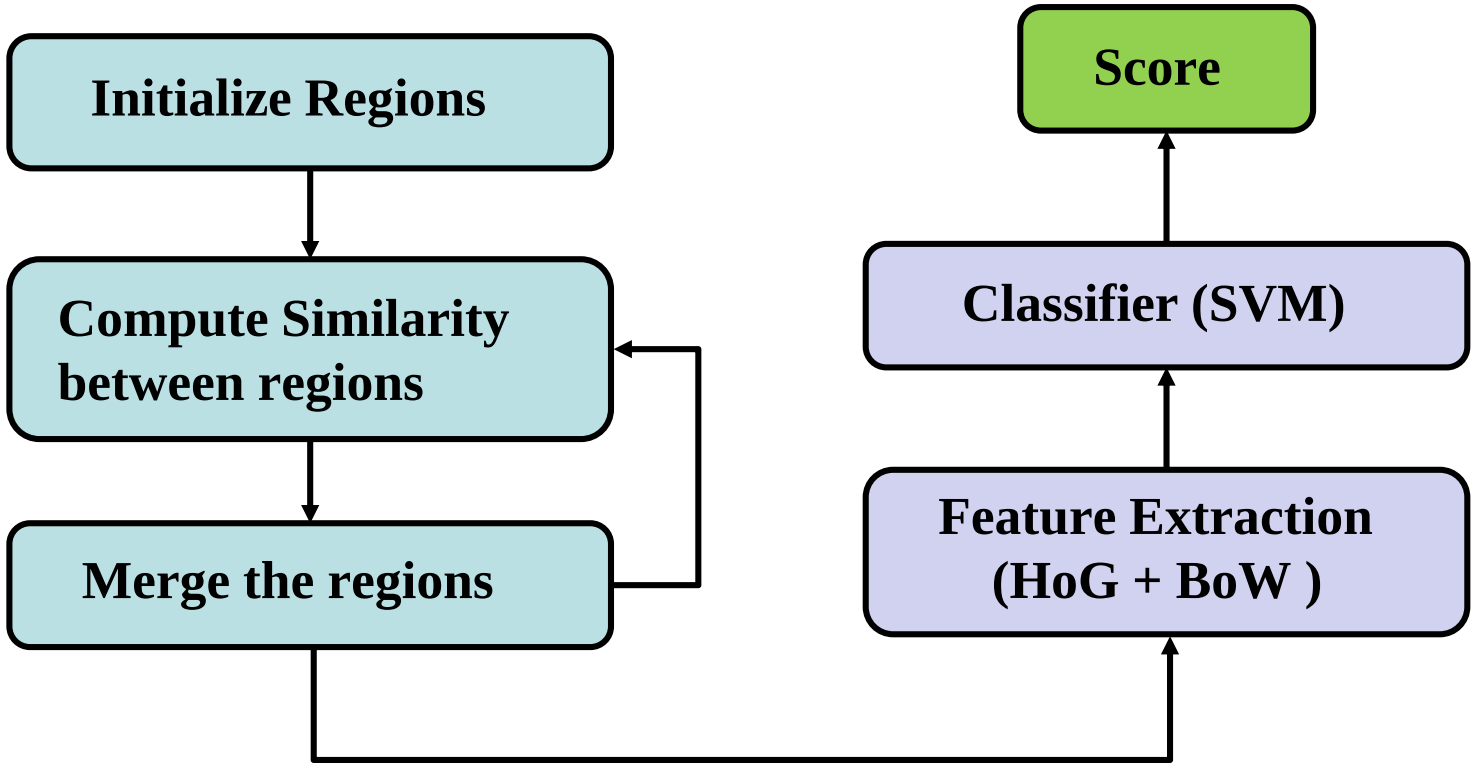


- **Semantic Segmentation**
  - **Segment each kind of object pixel-by-pixel**

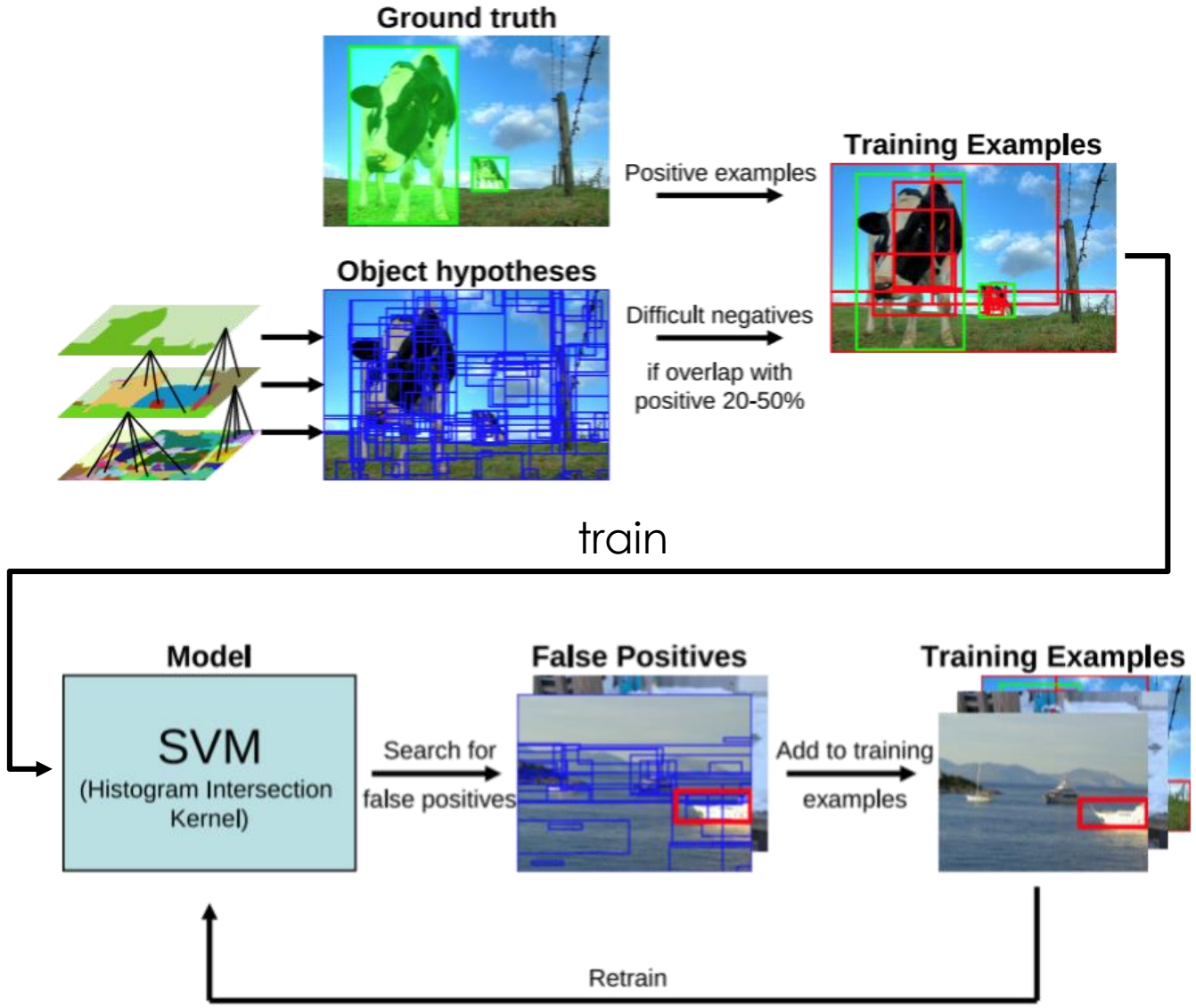
- **Instance Segmentation**
  - **Segment each object instance pixel-by-pixel**

Two-stage Object Detection

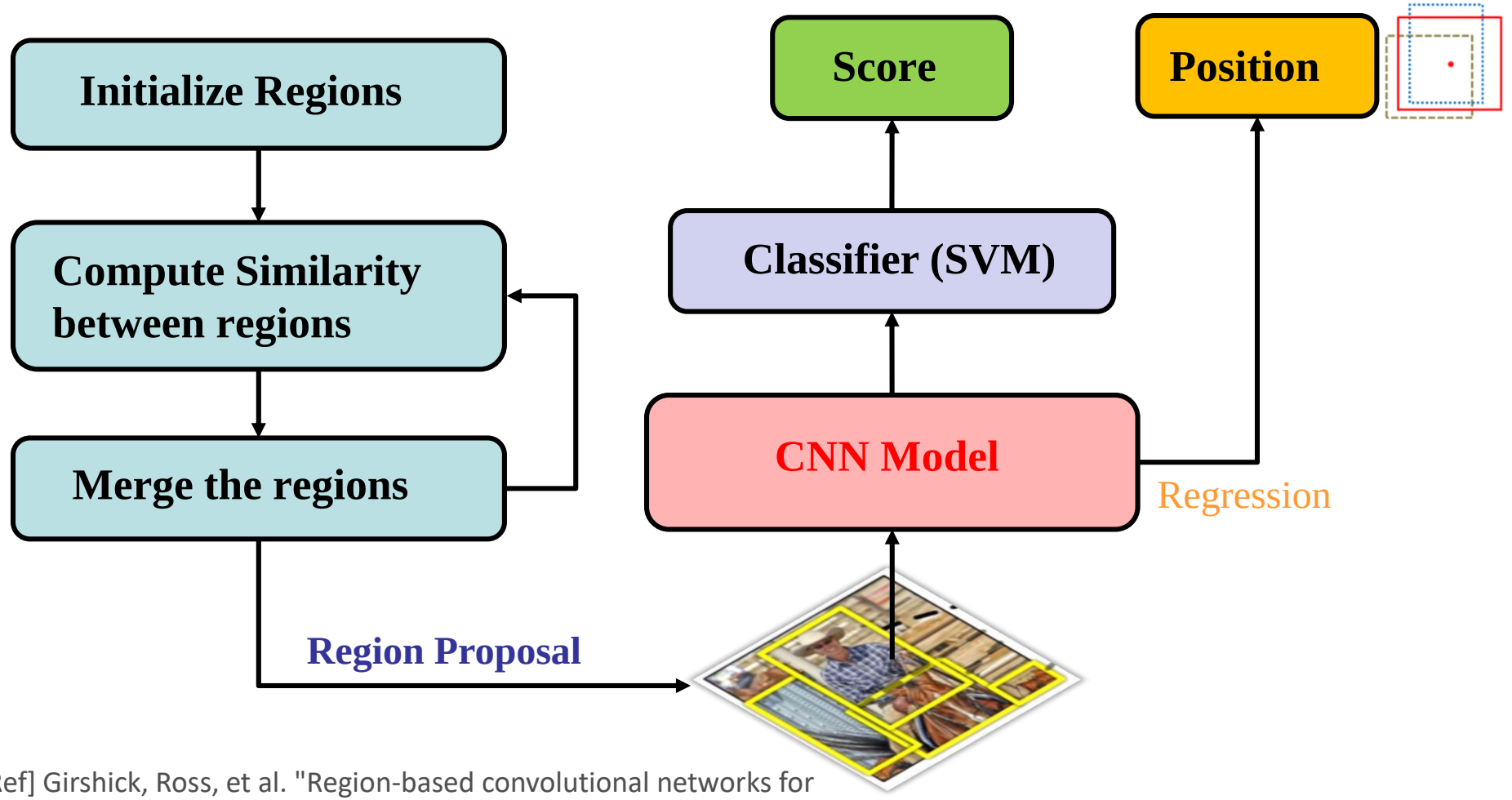
Selective Search



Selective Search

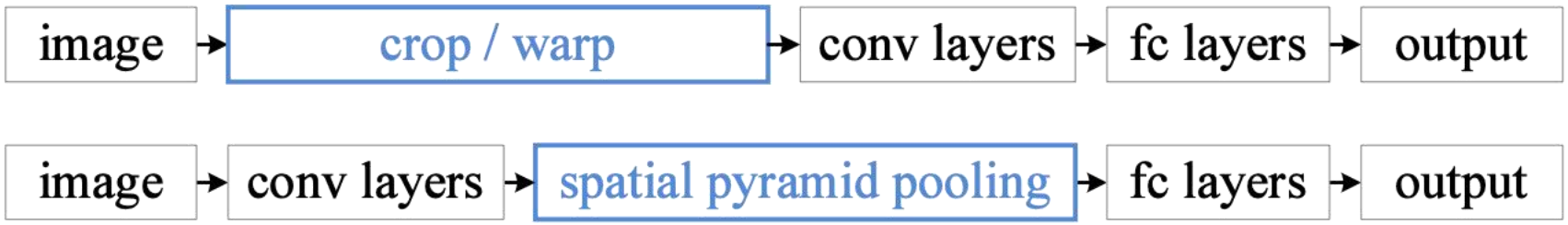


Region-based CNN (R-CNN)



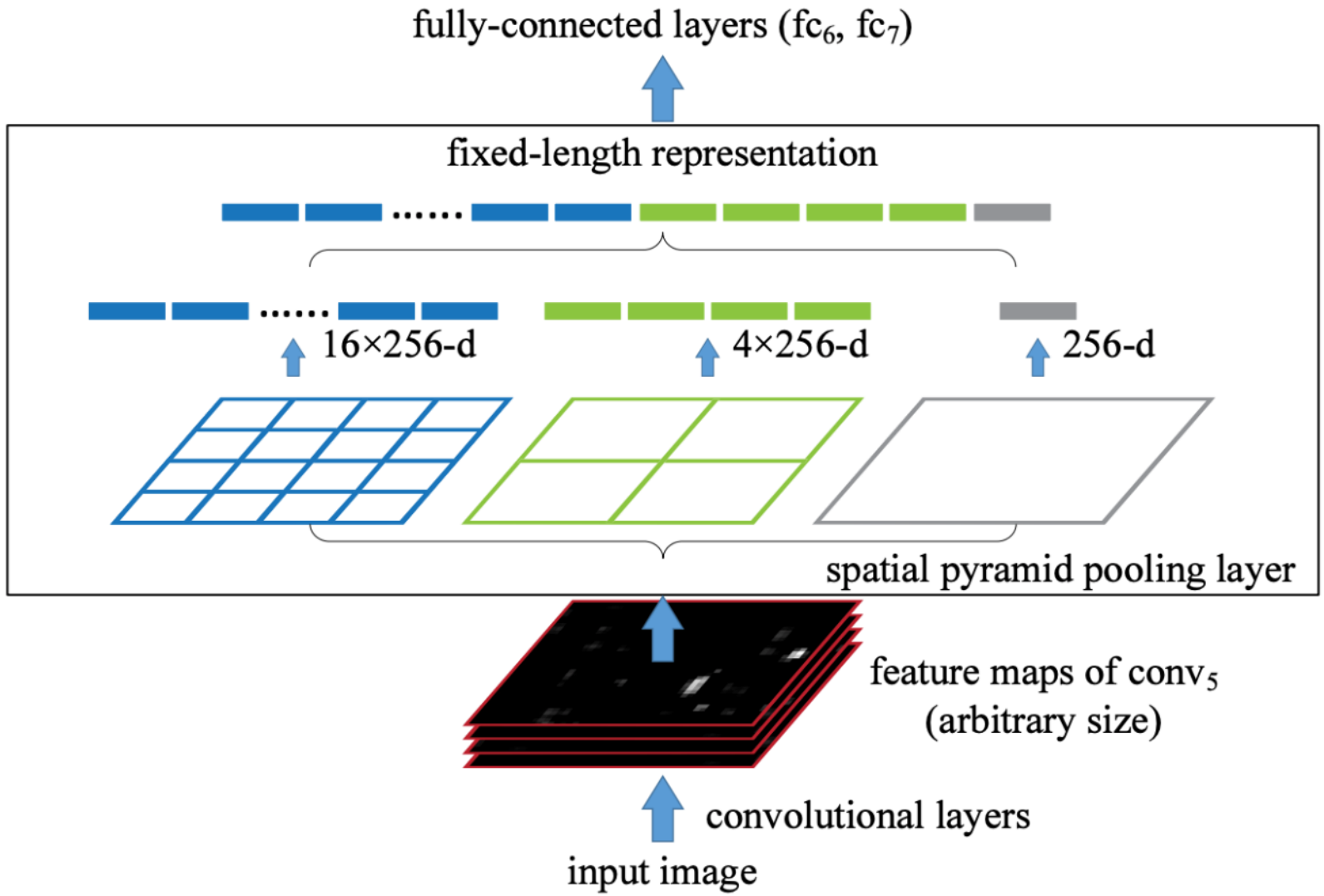
[Ref] Girshick, Ross, et al. "Region-based convolutional networks for accurate object detection and segmentation." IEEE transactions on pattern analysis and machine intelligence 38.1 (2016): 142-158.

SPP-Net

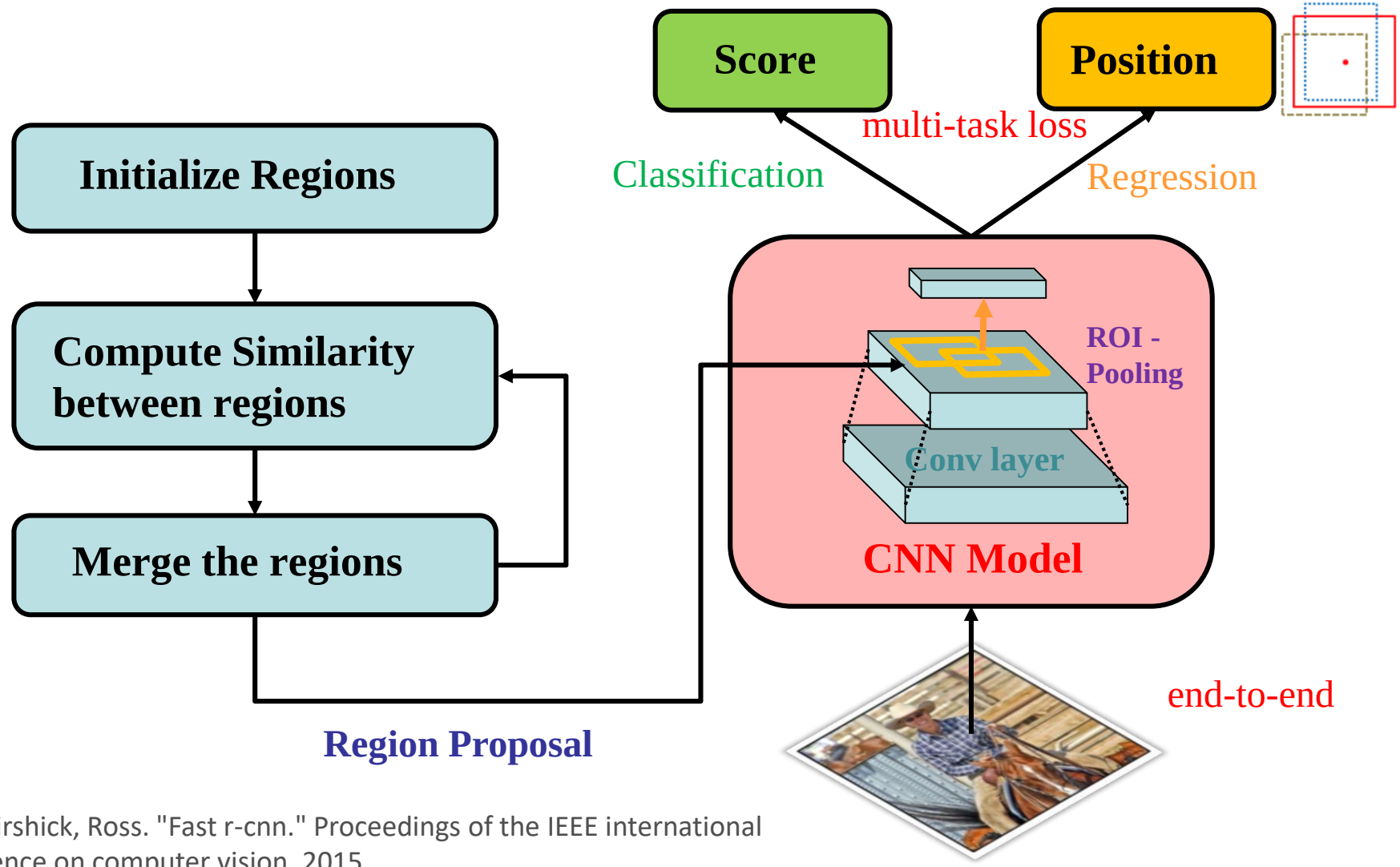


[Ref] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." European conference on computer vision. Springer, Cham, 2014.

Spatial Pyramid Pooling

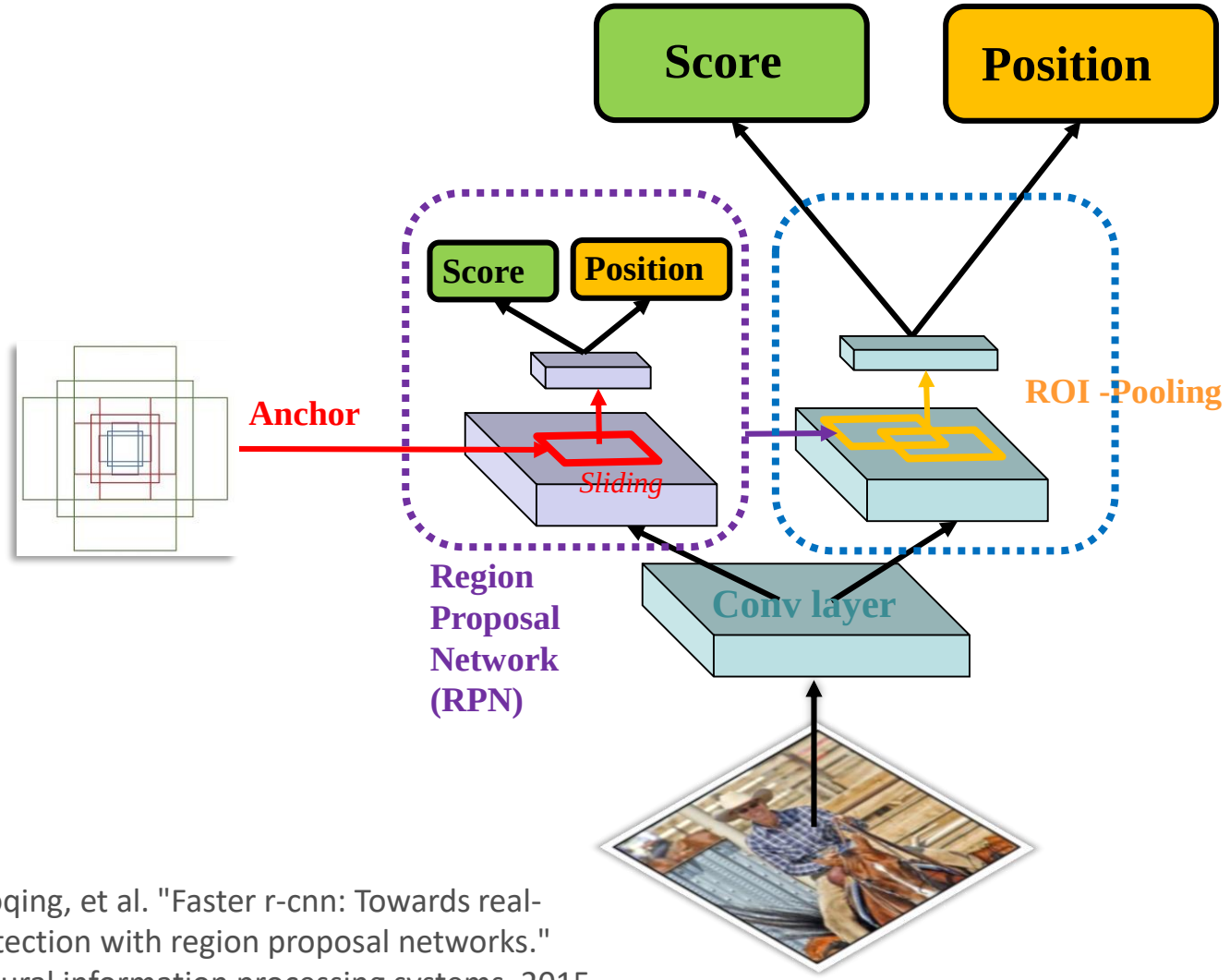


Fast R-CNN



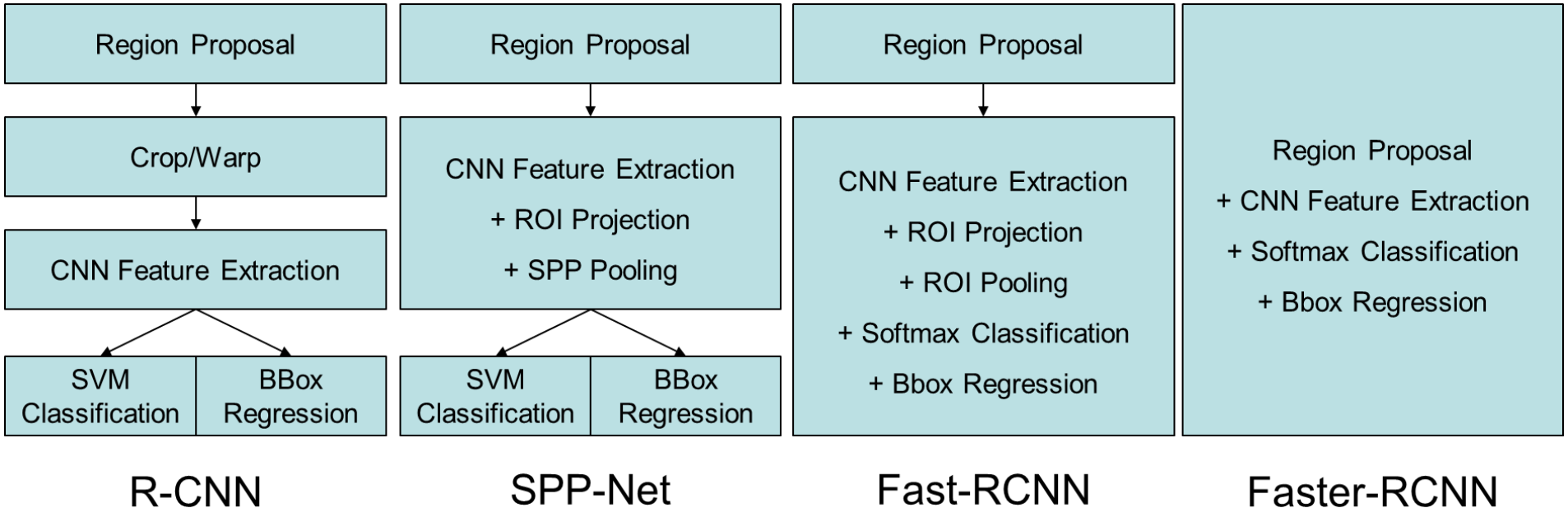
[Ref] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

Faster R-CNN



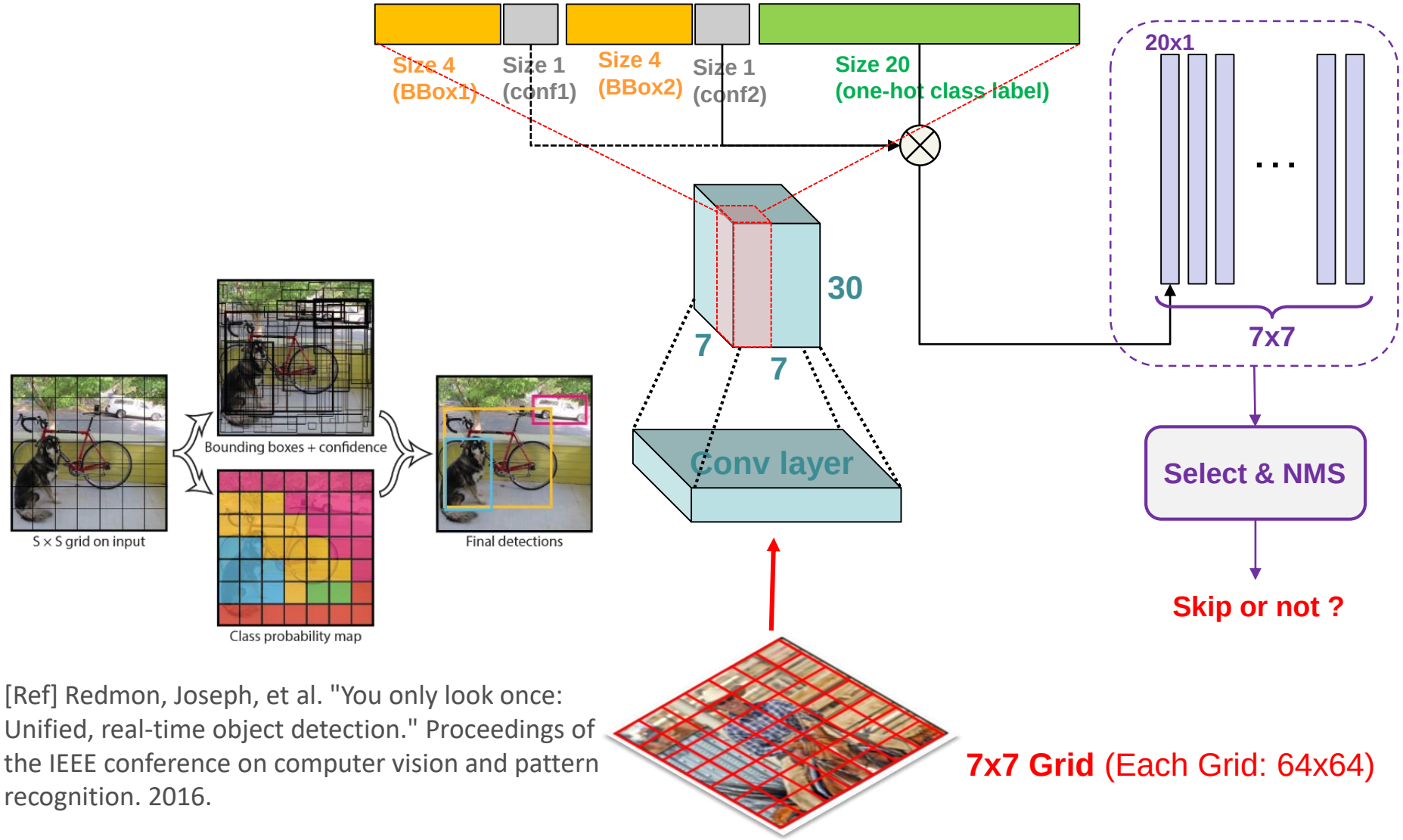
[Ref] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

Revolution of R-CNN



One-stage Object Detection

You Only Look Once (Yolo)



[Ref] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

You Only Look Once (Yolo)

- Loss Function

$$\text{loss}_{YOLO} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2 \right]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

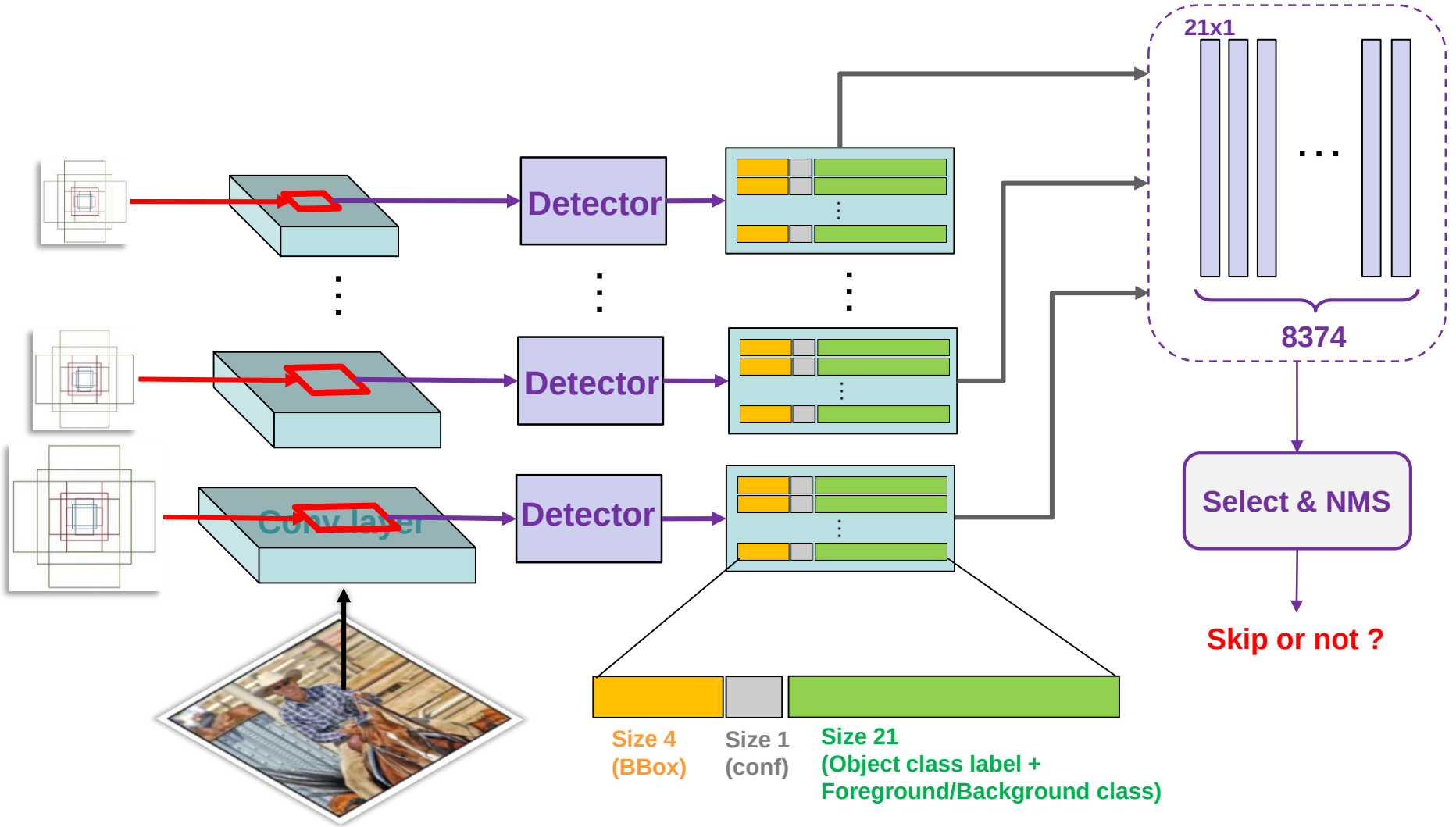
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

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

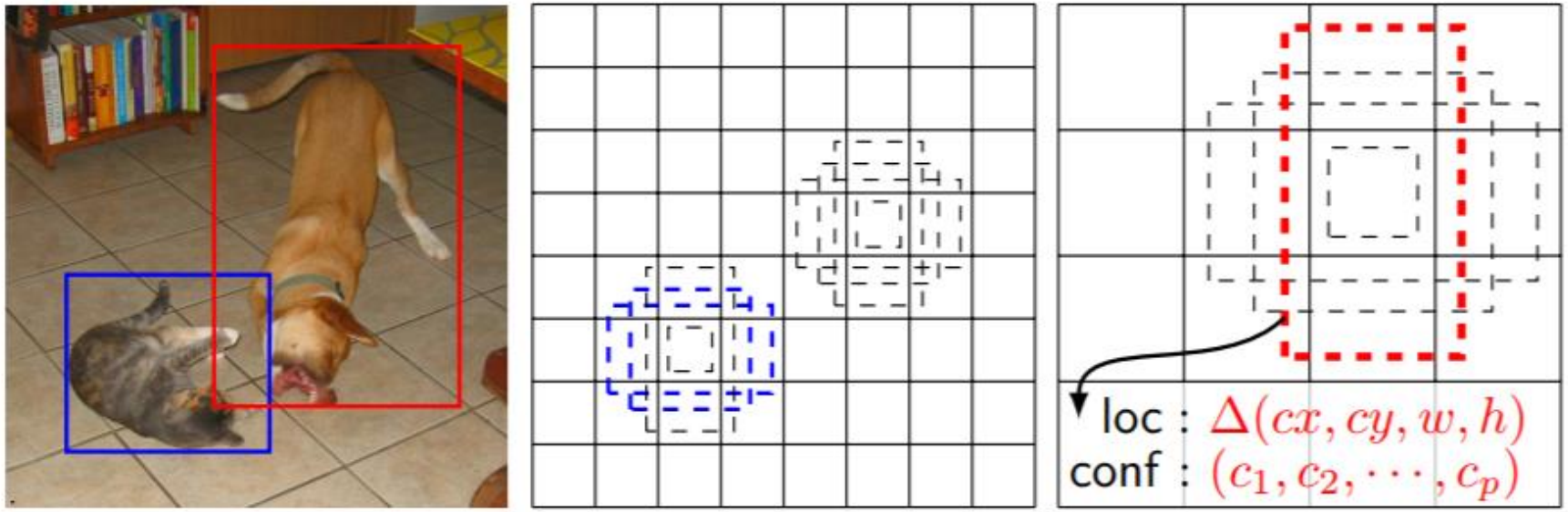
1_i^{obj}: 是物件有出現在 grid cell i。

1_{ij}^{obj}:是在第 i 個 grid cell 的第 j 個 Bounding Box 負責做預測。

Single Shot MultiBox Detection (SSD)

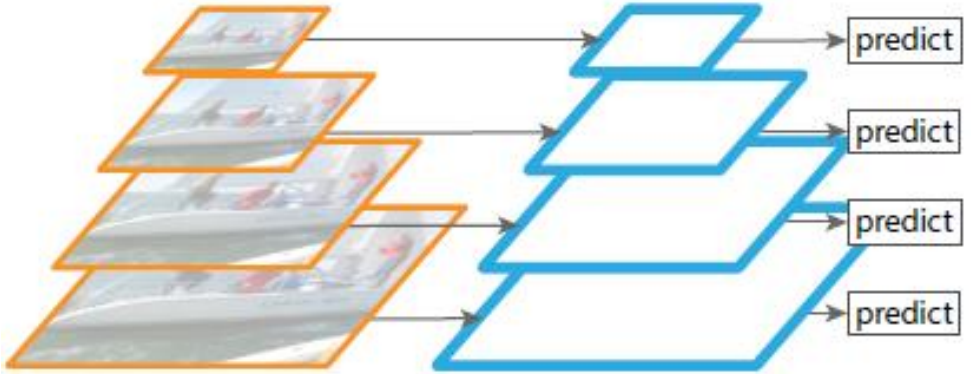


Single Shot MultiBox Detection (SSD)

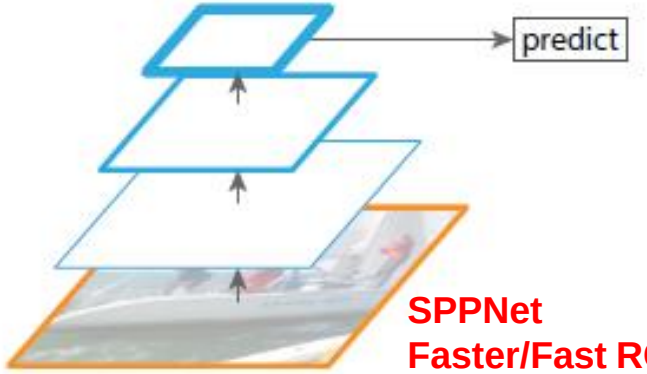


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Feature Pyramid Network

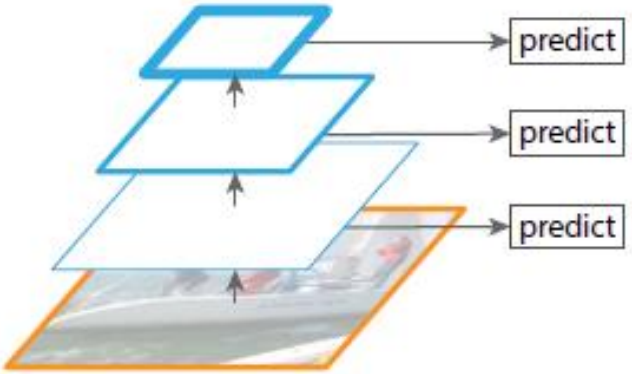


(a) Featurized image pyramid



(b) Single feature map

SPPNet
Faster/Fast RCNN



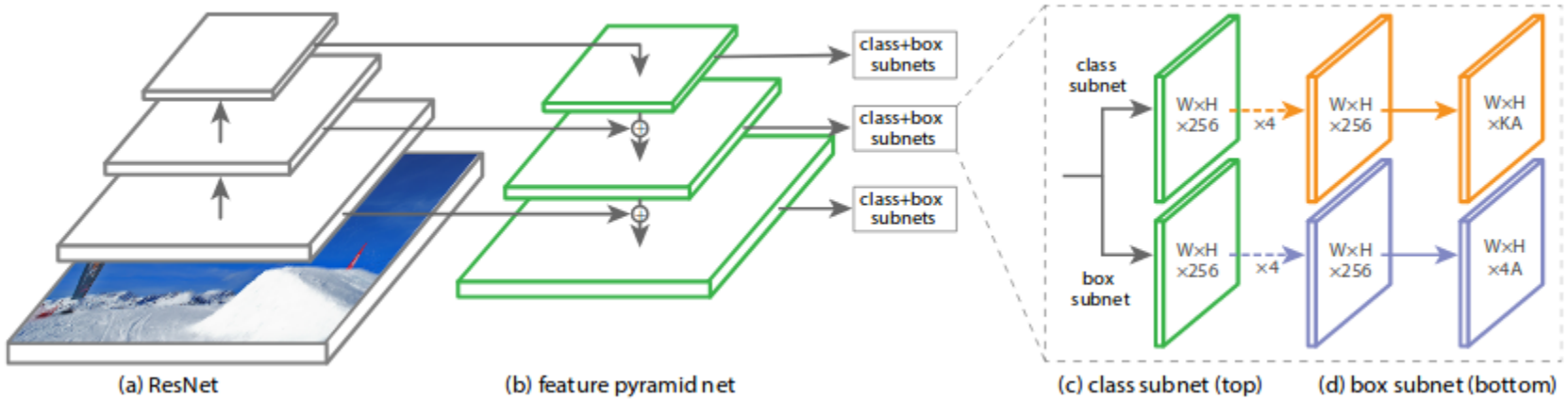
(c) Pyramidal feature hierarchy

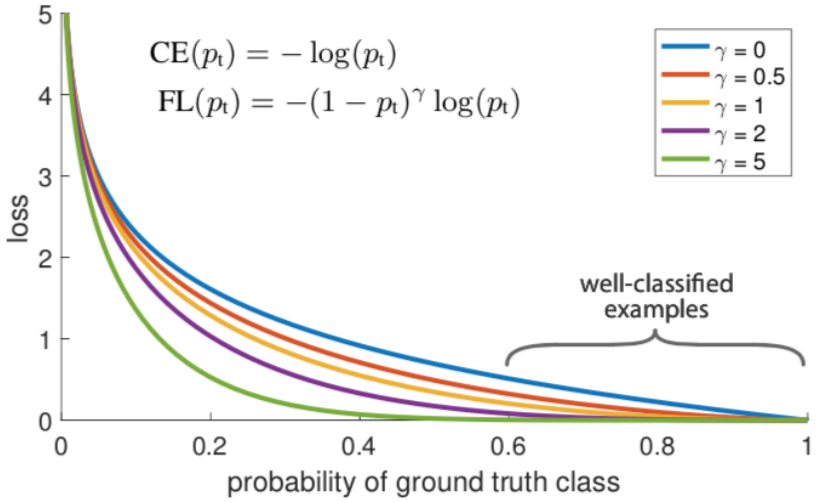
SSD

(d) Feature Pyramid Network

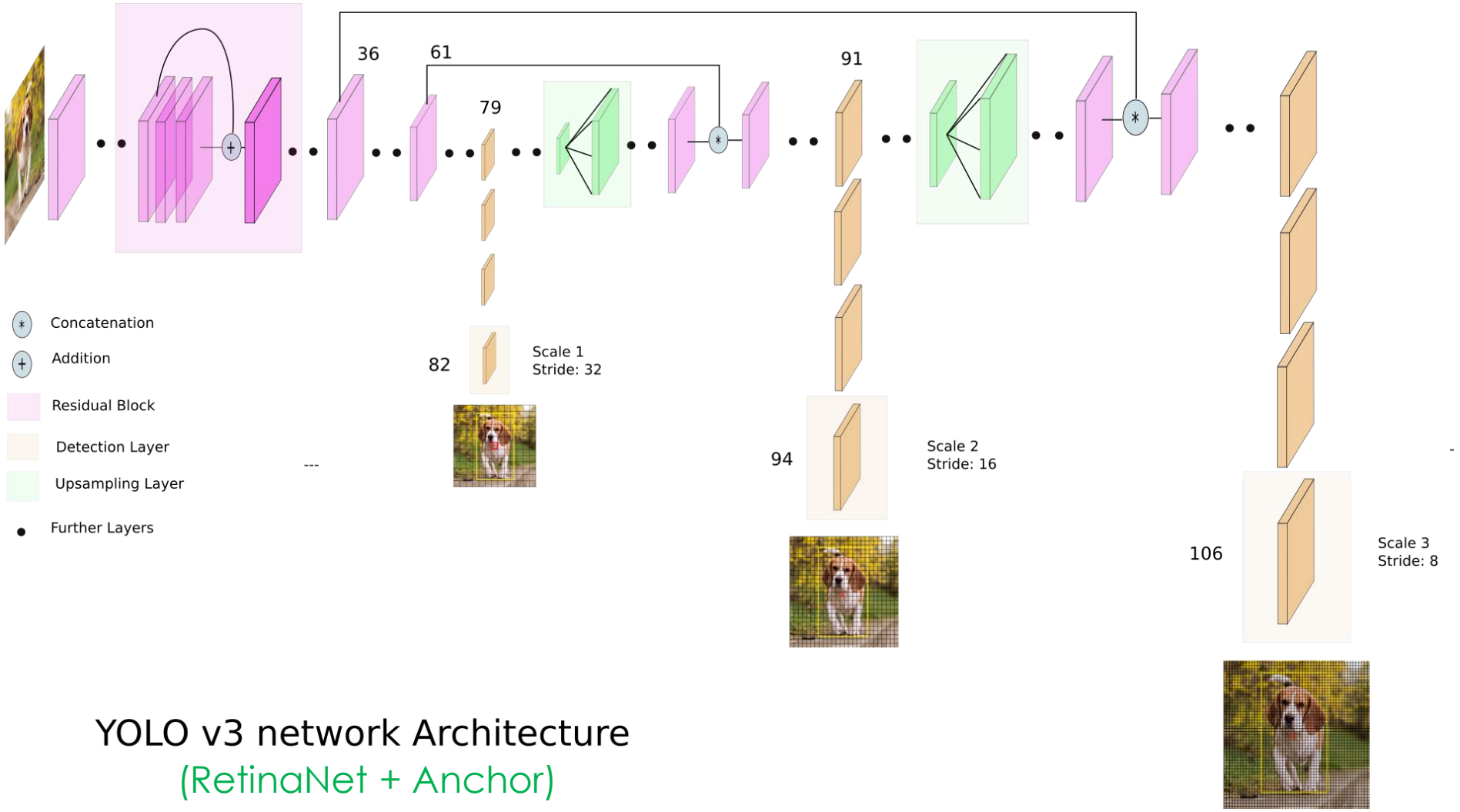
RetinaNet

- Focal Loss





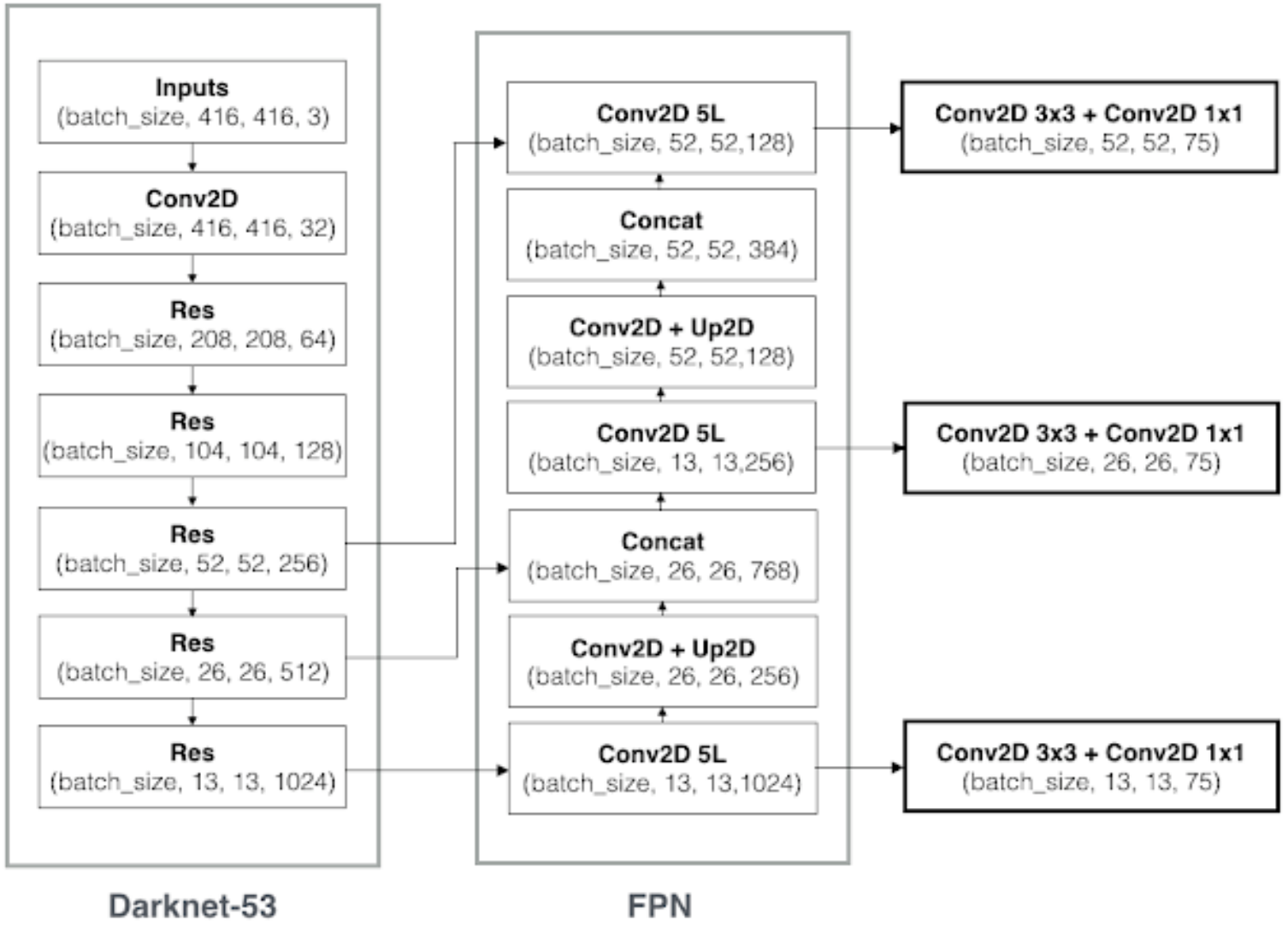
Yolo-v3



YOLO v3 network Architecture

(RetinaNet + Anchor)

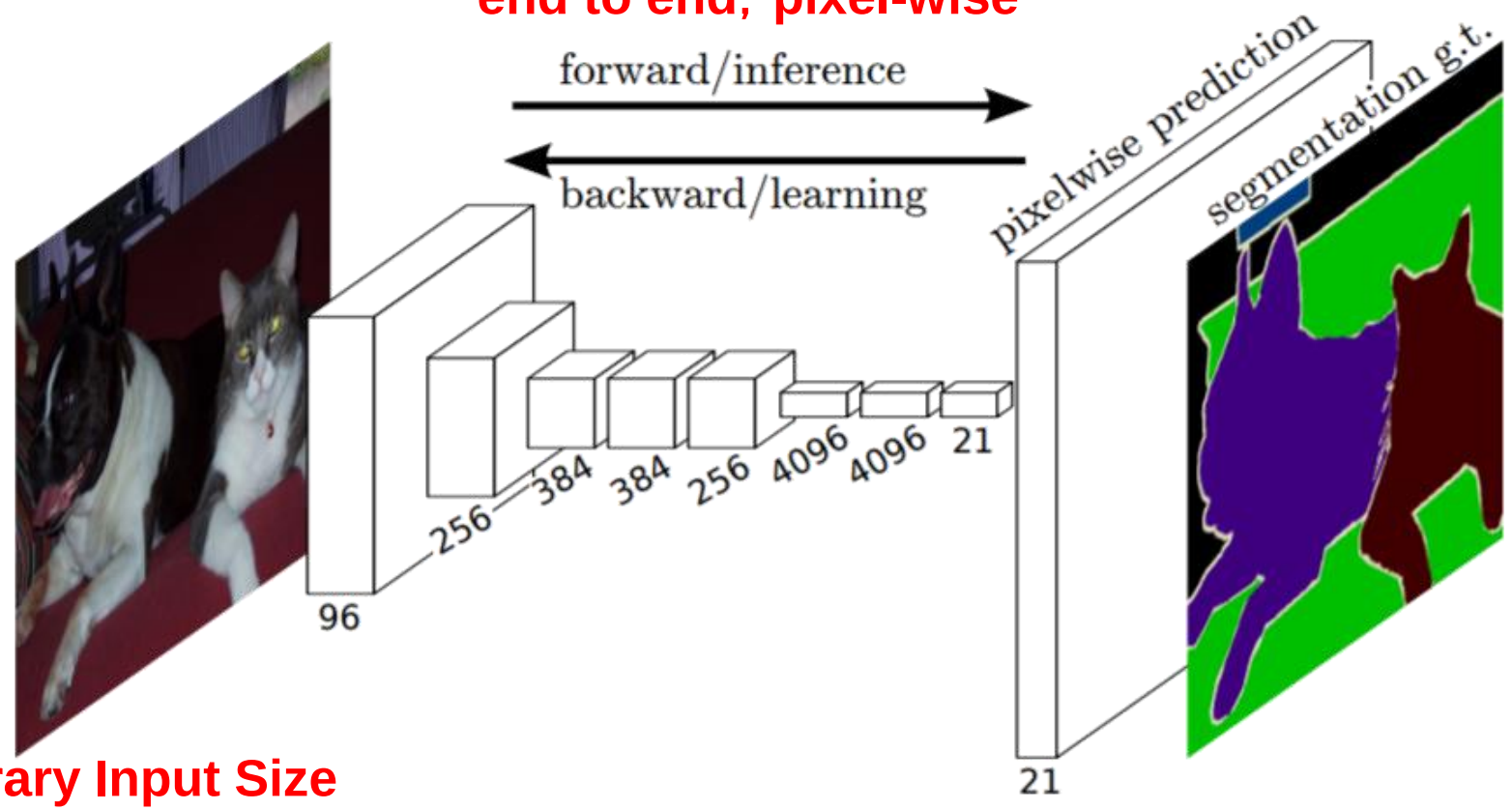
Yolo-v3


Inputs
(batch_size, 416, 416, 3)
Conv2D
(batch_size, 416, 416, 32)
Res
(batch_size, 208, 208, 64)
Res
(batch_size, 104, 104, 128)
Res
(batch_size, 52, 52, 256)
Res
(batch_size, 26, 26, 512)
Res
(batch_size, 13, 13, 1024)
Darknet-53
Conv2D 5L
(batch_size, 52, 52,128)
Concat
(batch_size, 52, 52, 384)
Conv2D + Up2D
(batch_size, 52, 52,128)
Conv2D 5L
(batch_size, 13, 13,256)
Concat
(batch_size, 26, 26, 768)
Conv2D + Up2D
(batch_size, 26, 26, 256)
Conv2D 5L
(batch_size, 13, 13,1024)
FPN
Conv2D 3x3 + Conv2D 1x1
(batch_size, 52, 52, 75)
Conv2D 3x3 + Conv2D 1x1
(batch_size, 26, 26, 75)
Conv2D 3x3 + Conv2D 1x1
(batch_size, 13, 13, 75)

Segmentation

Fully Convolution Network

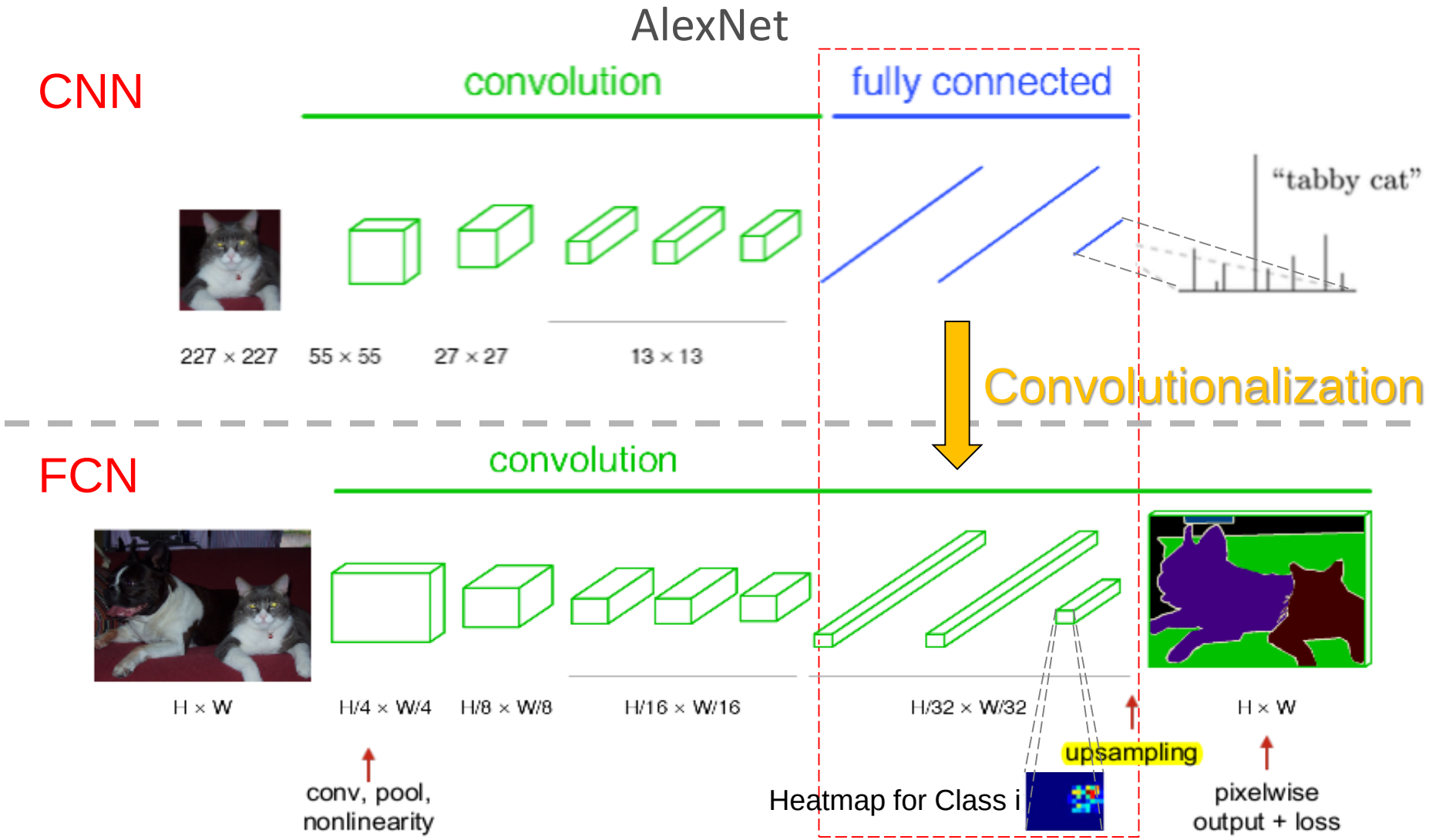
end to end, pixel-wise



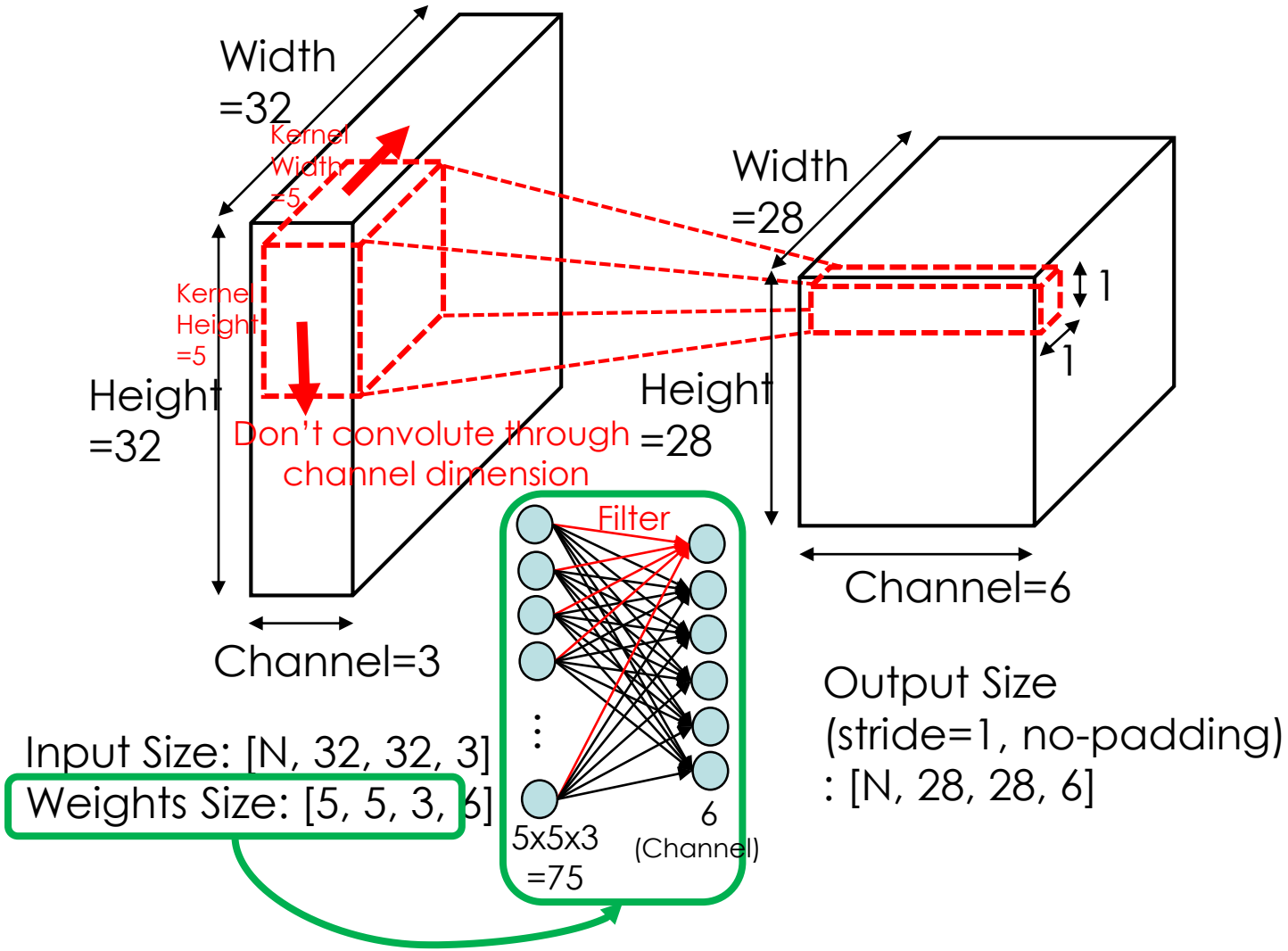
Arbitrary Input Size
(Similar Object Scale)

[Ref] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

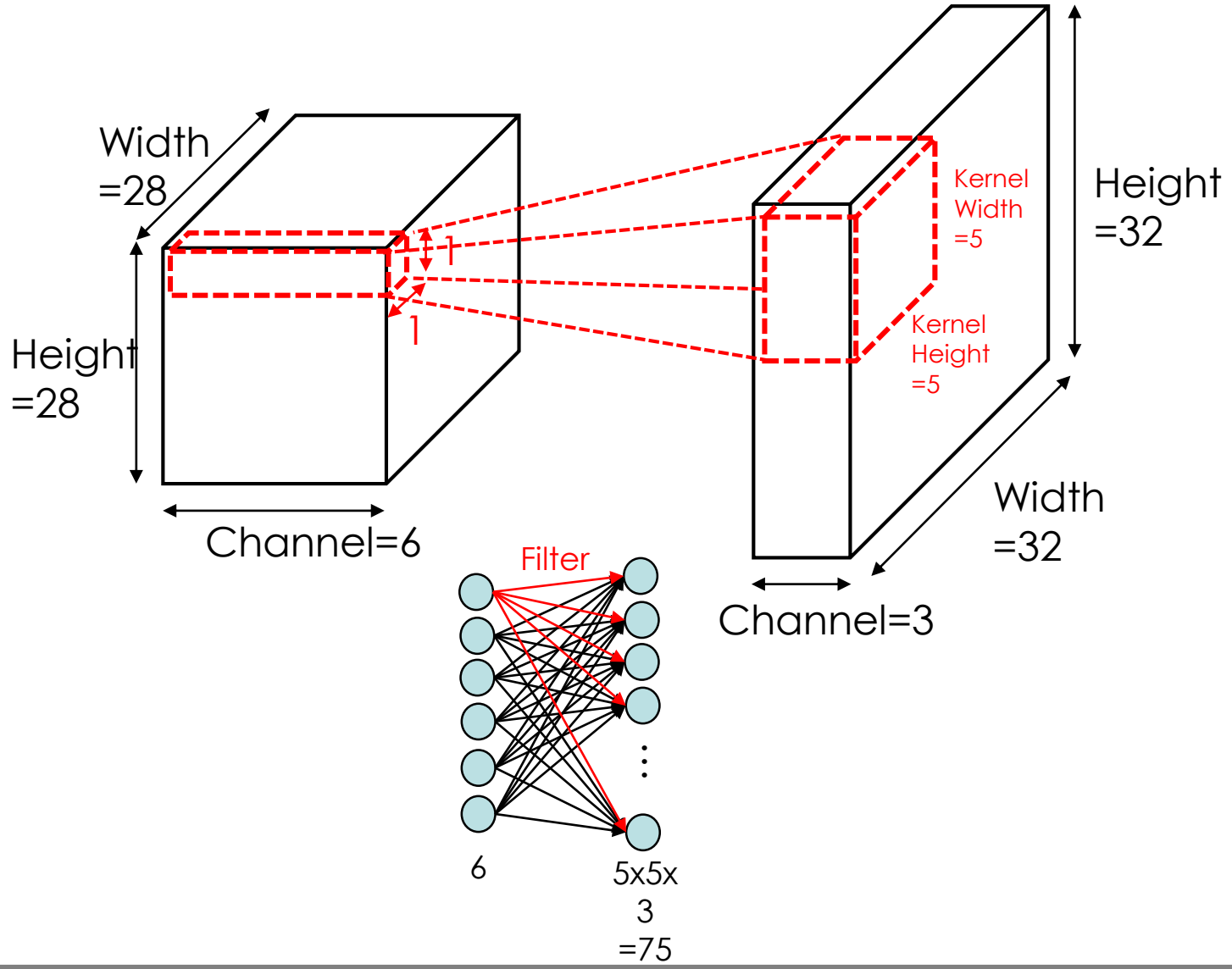
CNN vs FCN


AlexNet
CNN
convolution
fully connected
"tabby cat"
227 × 227
55 × 55
27 × 27
13 × 13
Convolutionalization
FCN
convolution
H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
H × W
upsampling
conv, pool,
nonlinearity
Heatmap for Class i
pixelwise
output + loss

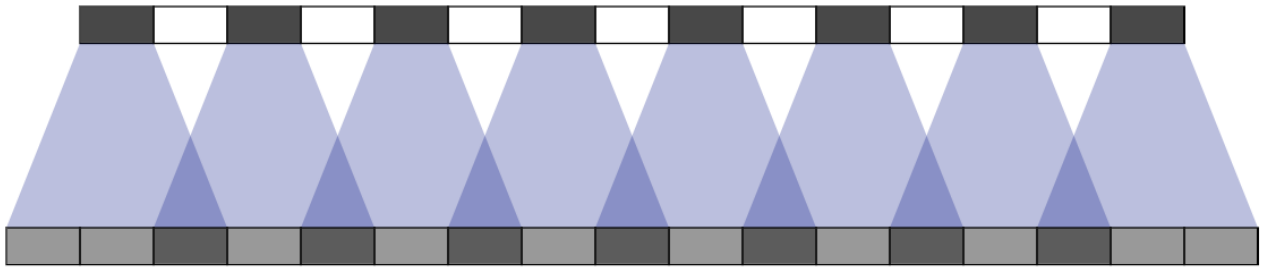
Convolutional Layer

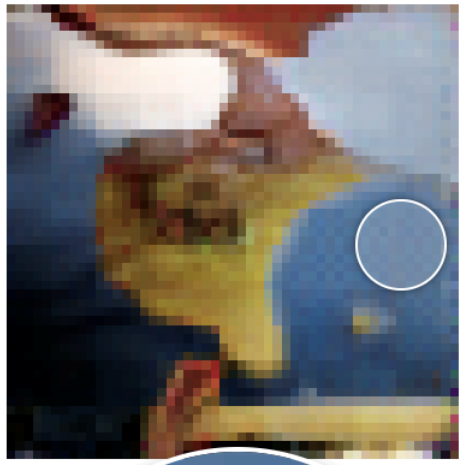


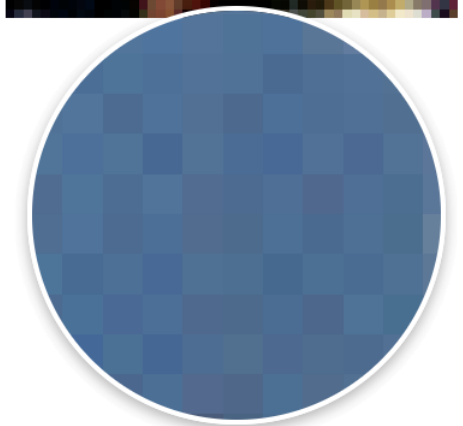
De-convolutional Layer



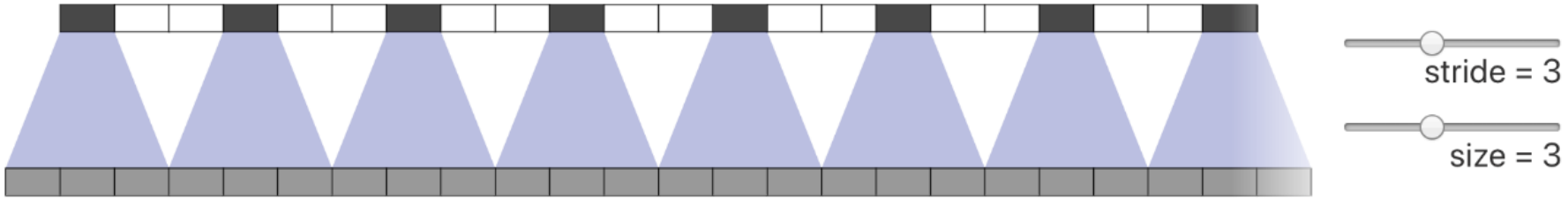
Checkerboard Artifacts







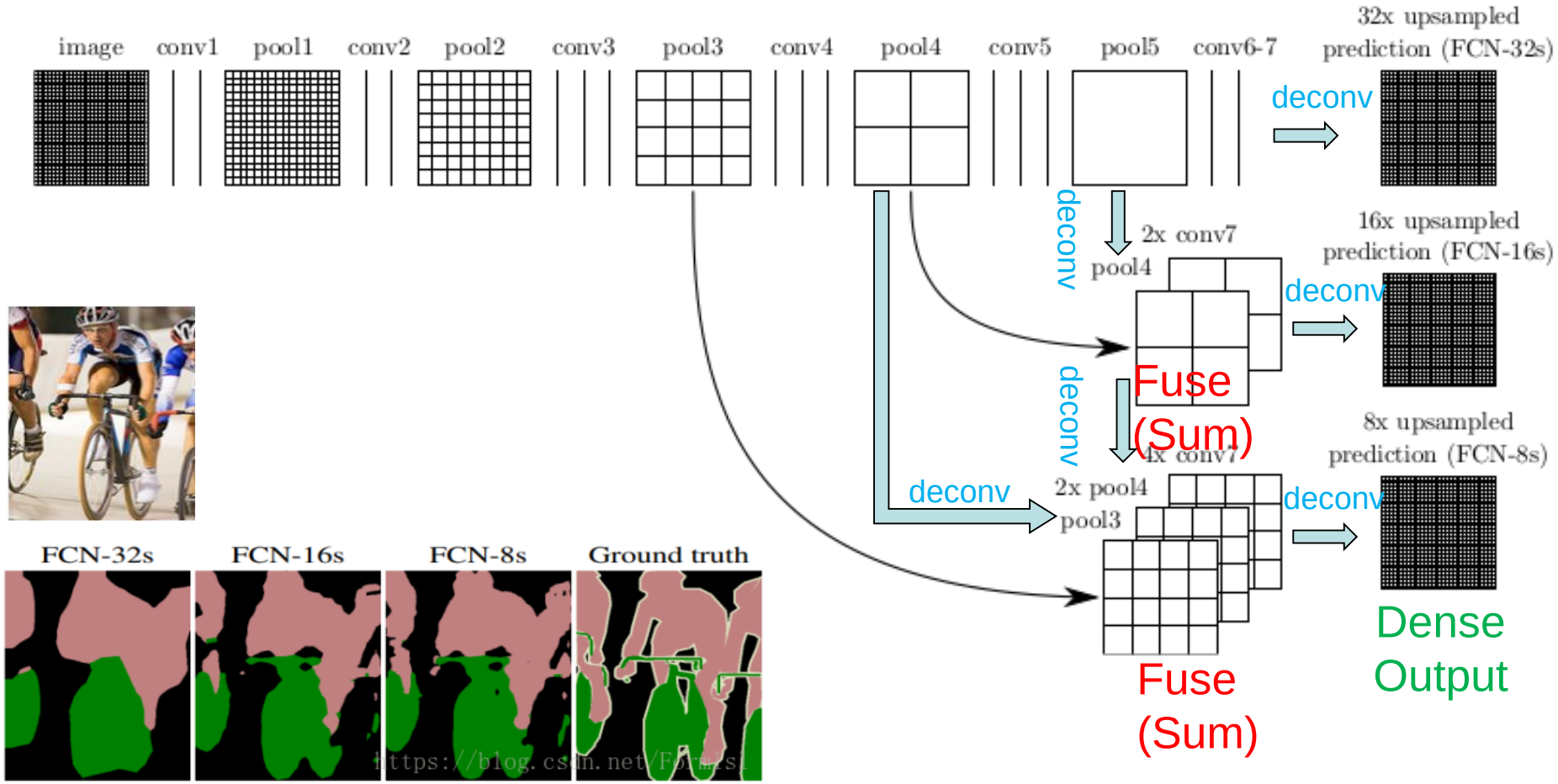
Checkerboard Artifacts



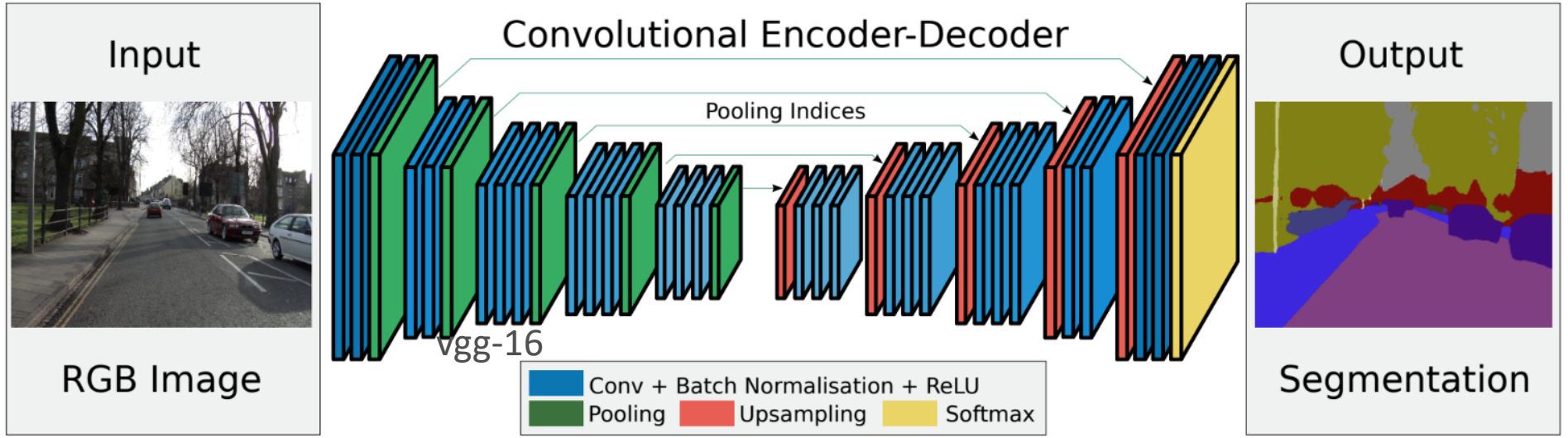
Solution 1:
Adjust stride and filter size to non-overlapping deconvolution

Solution 2:
Bilinear/Max Upsampling
+ 1 stride convolution

Upsample in FCN

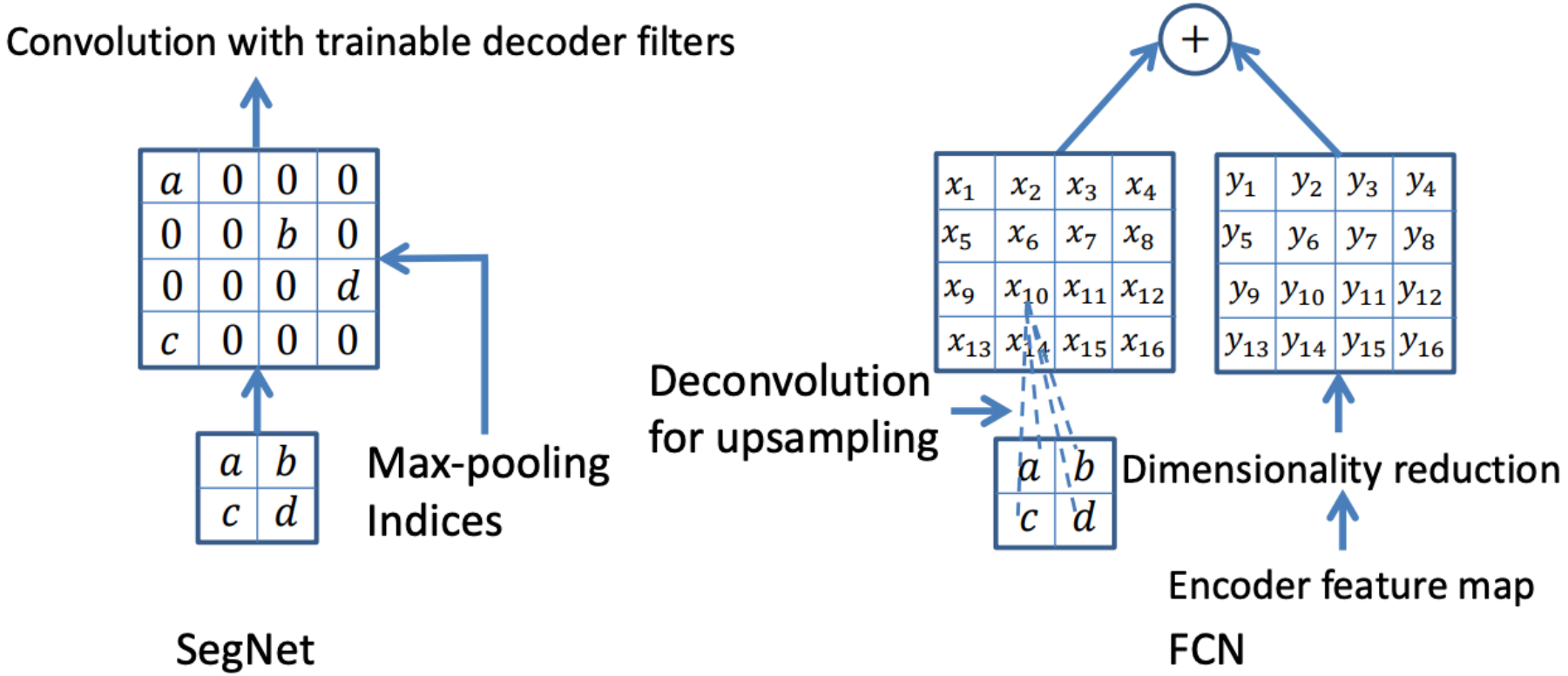


SegNet

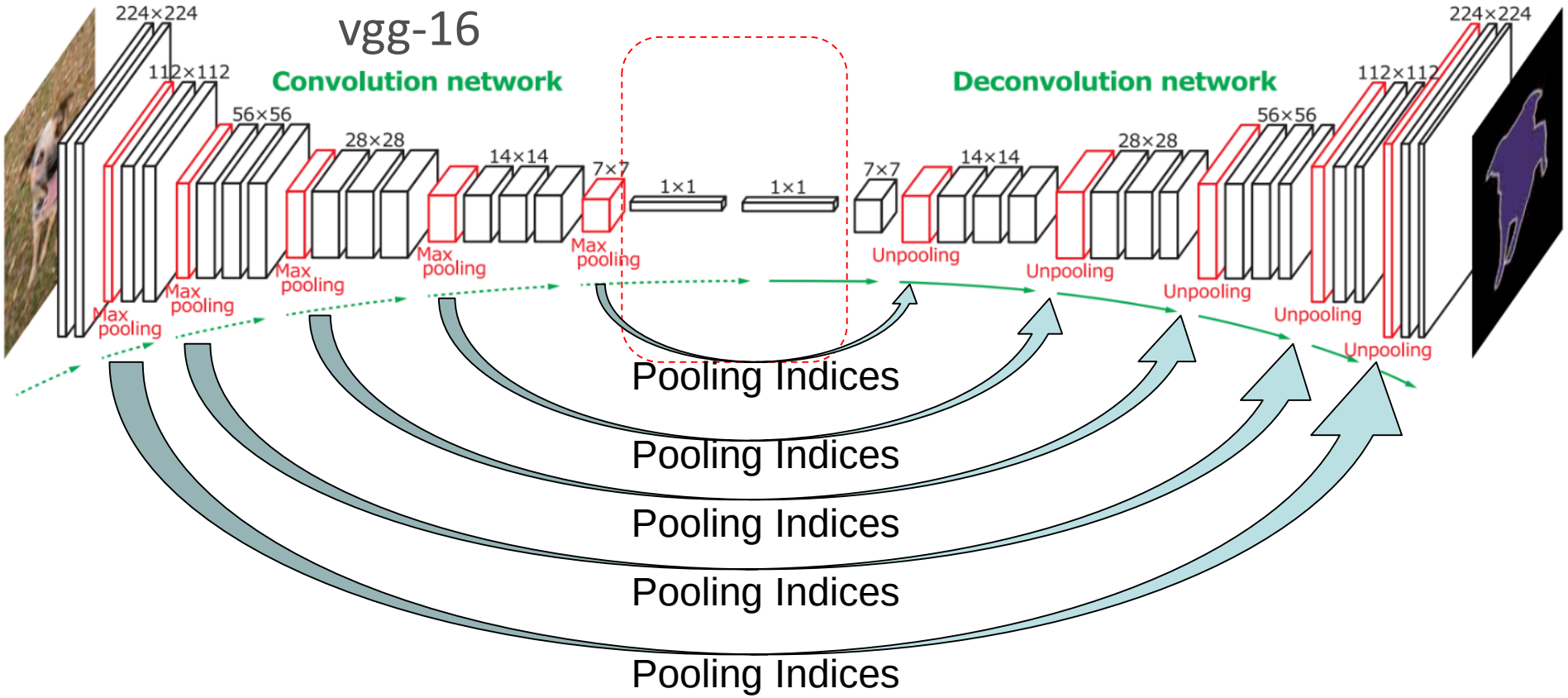


[Ref] Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv preprint arXiv:1511.00561 (2015).

SegNet vs FCN

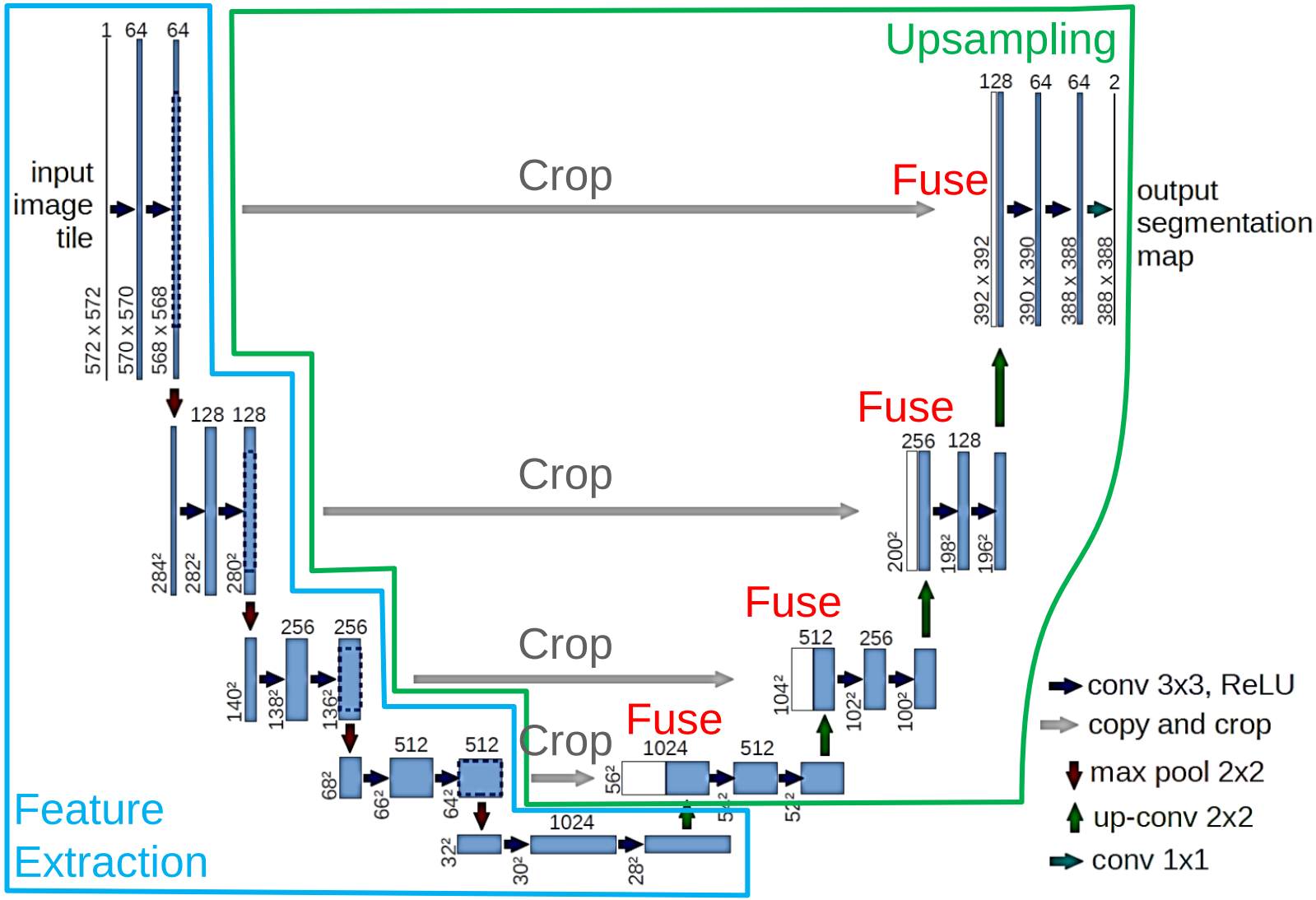


DeconvNet



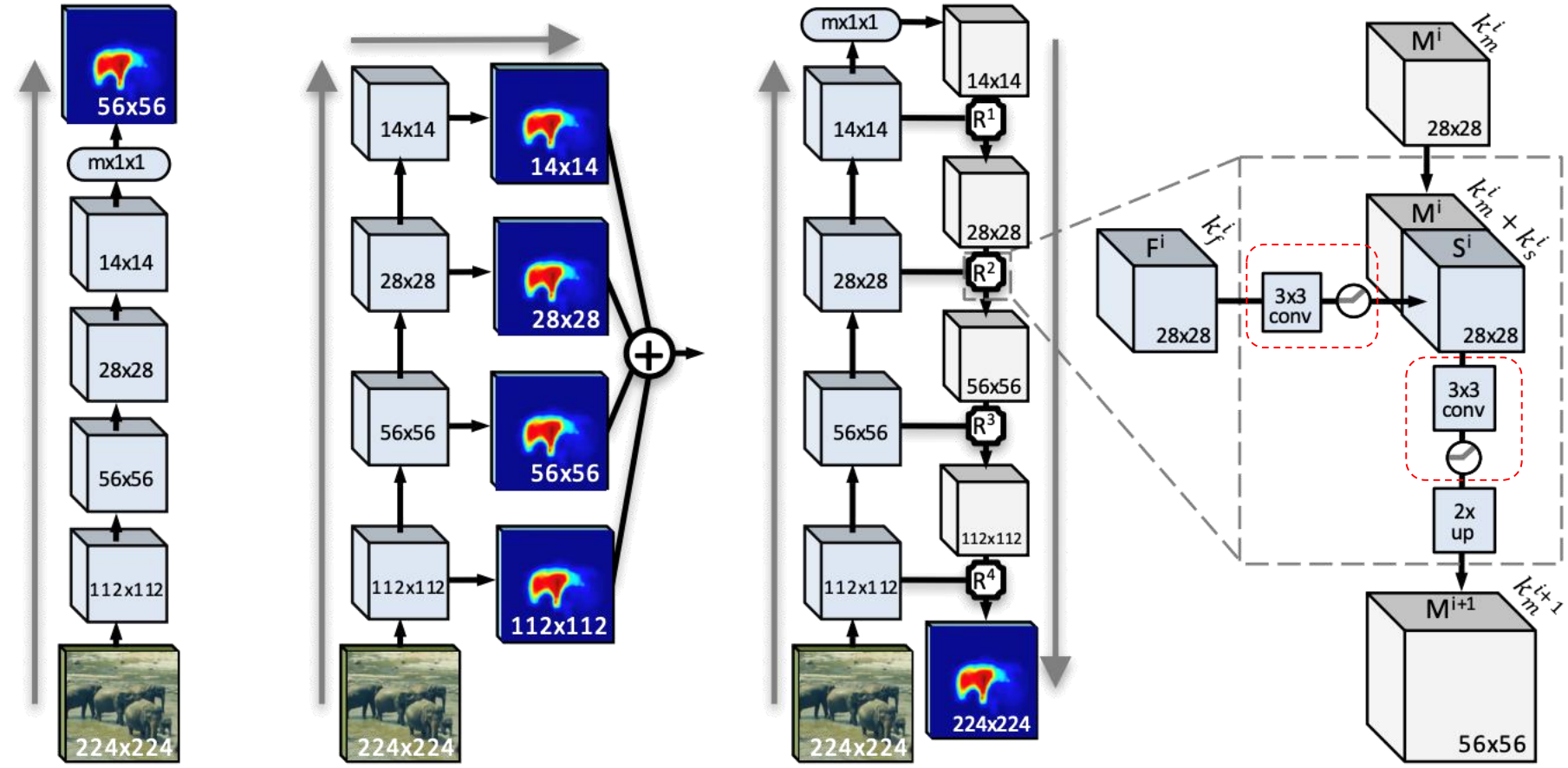
[Ref] Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." Proceedings of the IEEE international conference on computer vision. 2015.

U-Net



[Ref] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

SharpMask



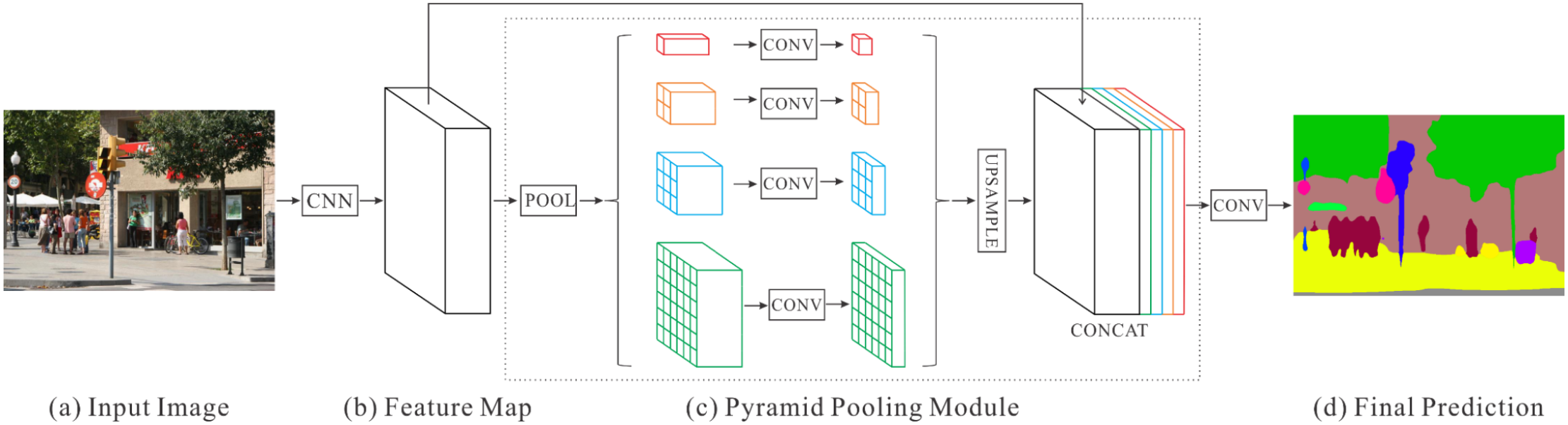
(a) feedforward (b) feedforward + skip (c) proposed network (d) refinement module

DeepMask

Similar to U-Net

[Ref] Pinheiro, Pedro O., et al. "Learning to refine object segments." European Conference on Computer Vision. Springer, Cham, 2016. Facebook AI Research (FAIR)

Pyramid Scene Parsing Network(PSPNet)



DeepLab v3

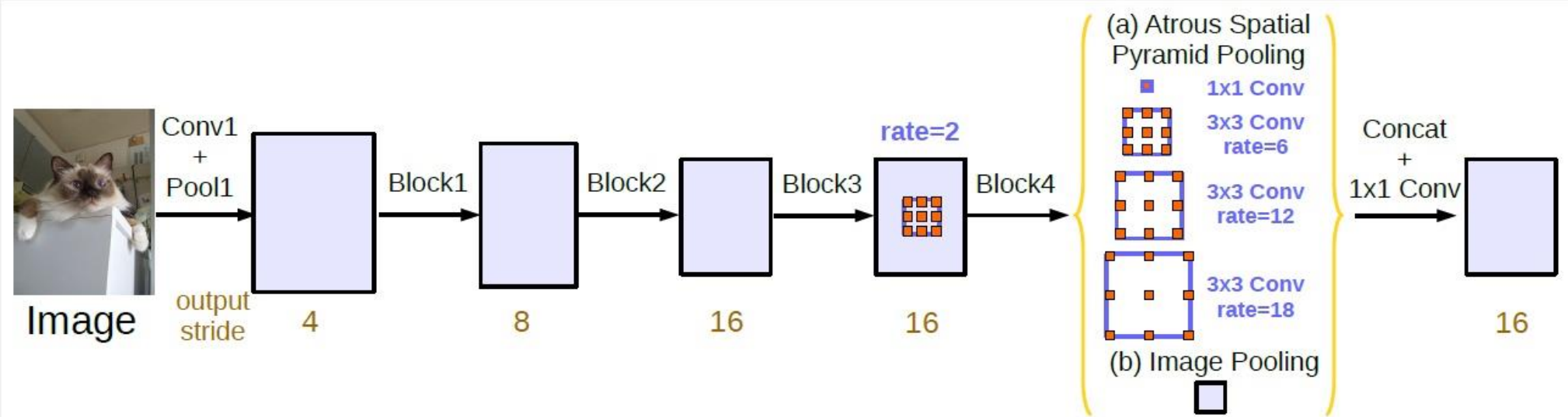
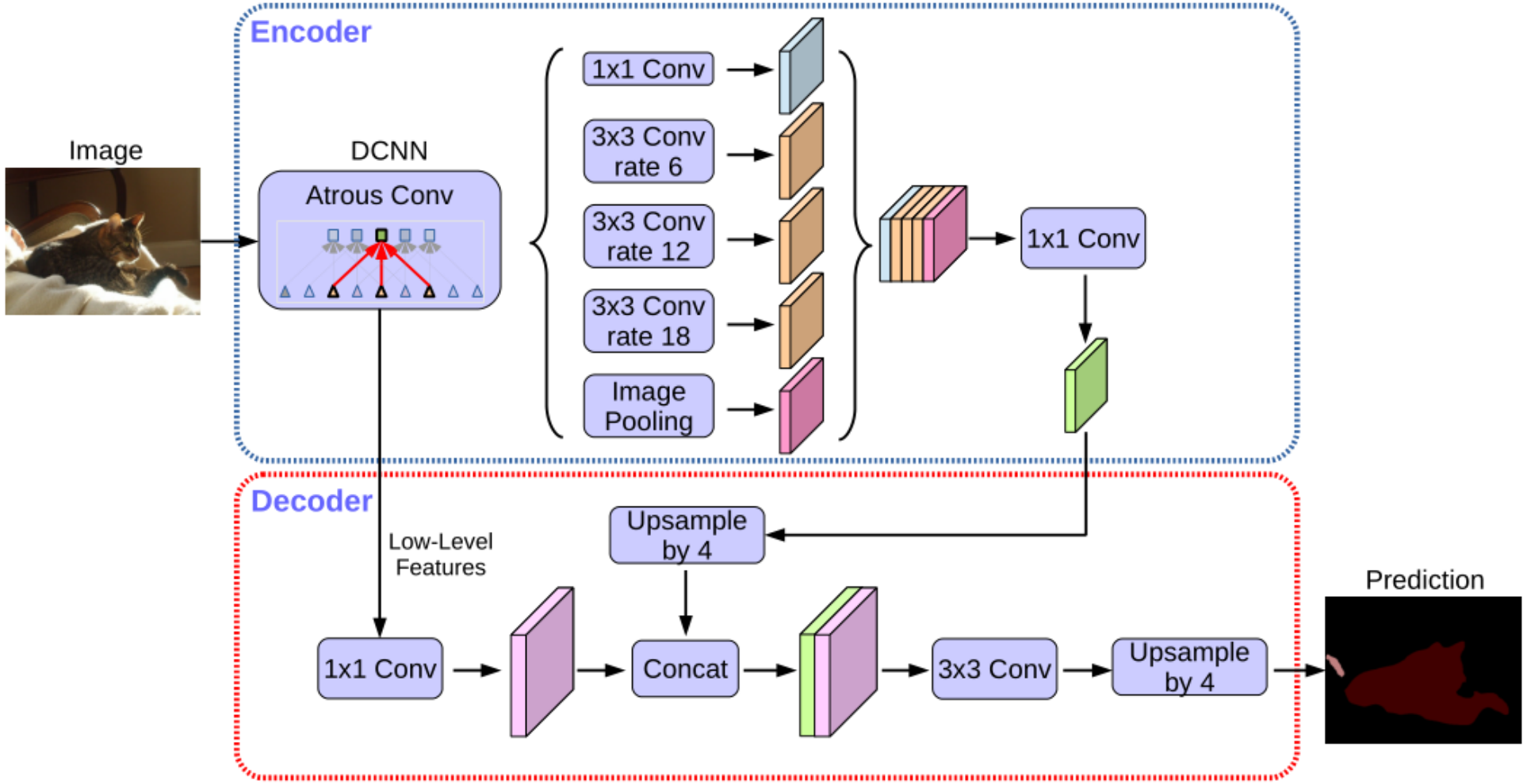


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

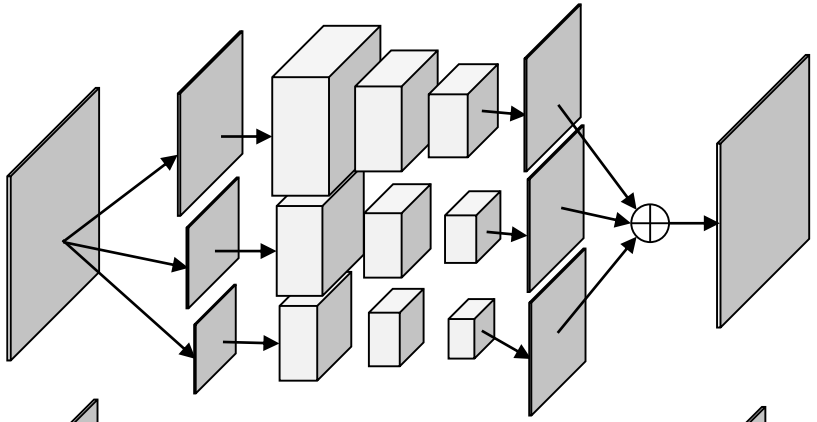
DeepLab v3+


Encoder
Image
DCNN
Atrous Conv
1x1 Conv
3x3 Conv rate 6
3x3 Conv rate 12
3x3 Conv rate 18
Image Pooling
1x1 Conv
Decoder
Low-Level Features
Upsample by 4
1x1 Conv
Concat
3x3 Conv
Upsample by 4
Prediction

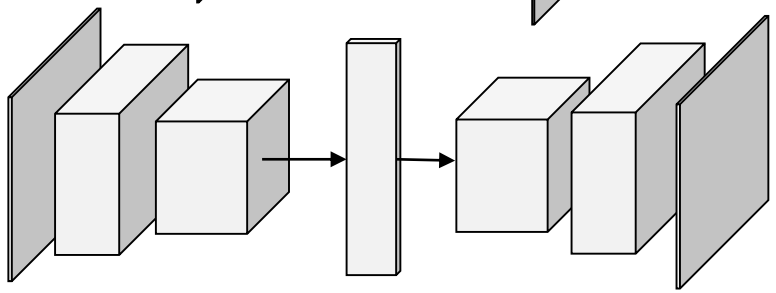
Different Segmentation Structures

- Capture the multiscale information of the image

Image Pyramid Structure

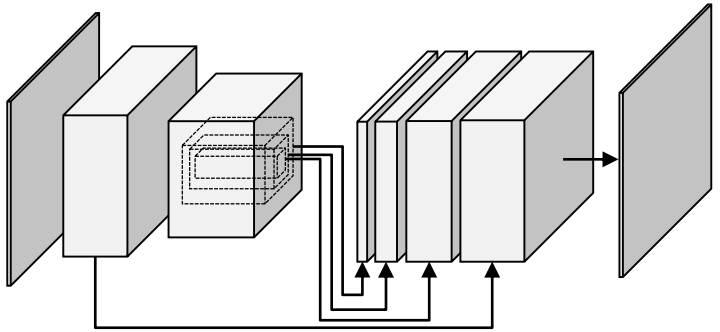


Encoder-Decoder Structure



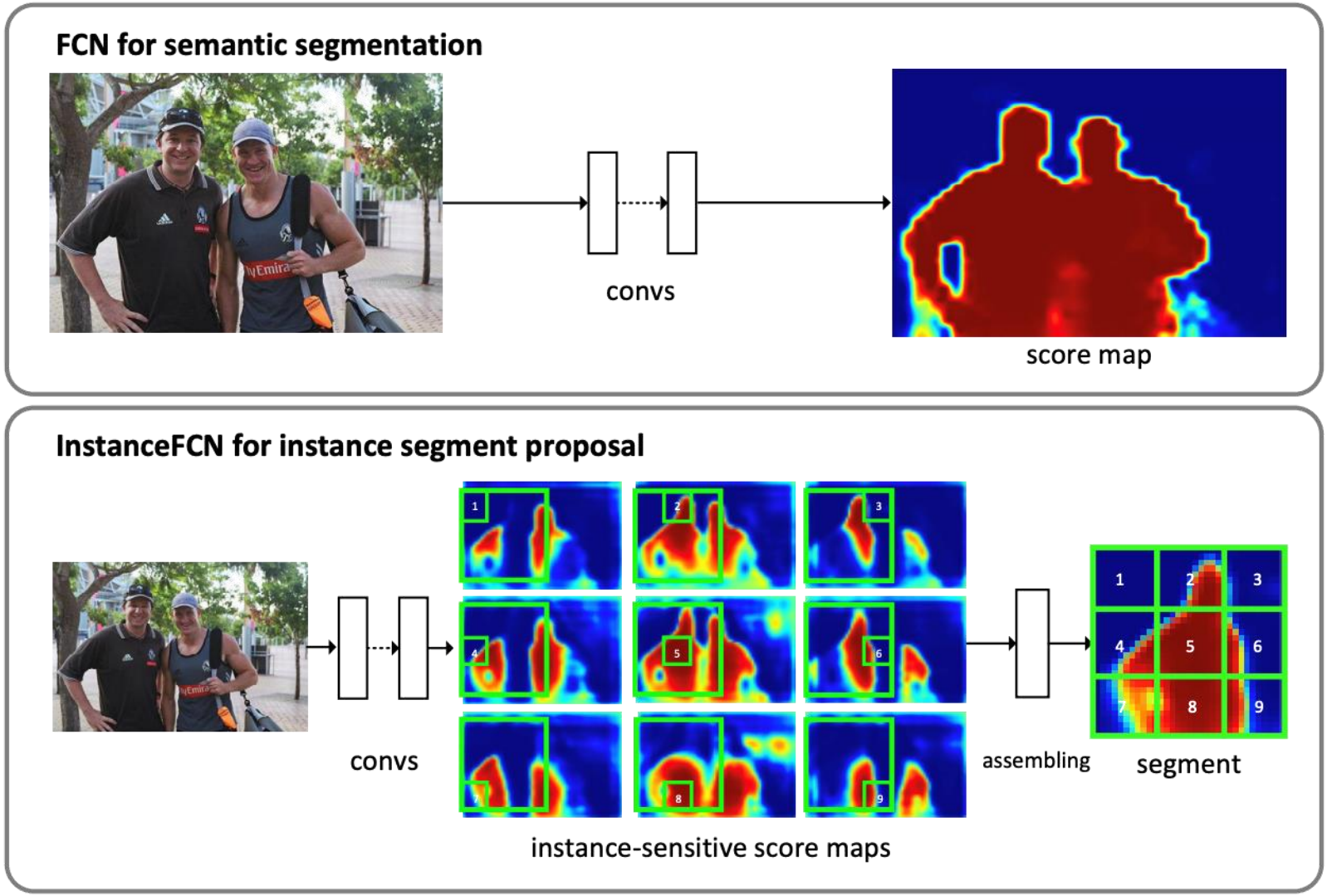
FCN
U-Net

Pyramid Pooling Structure



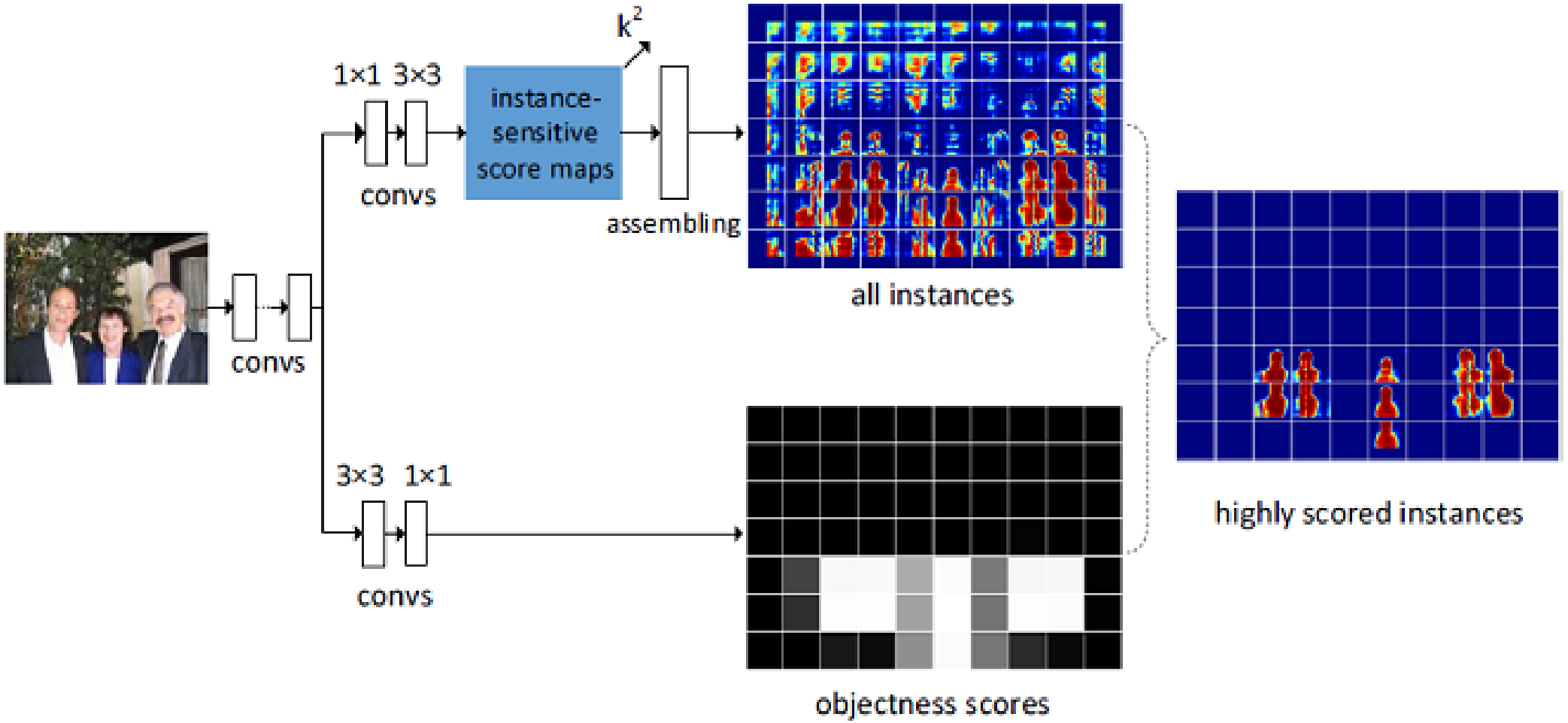
PSPNet
DeepLab

InstanceFCN



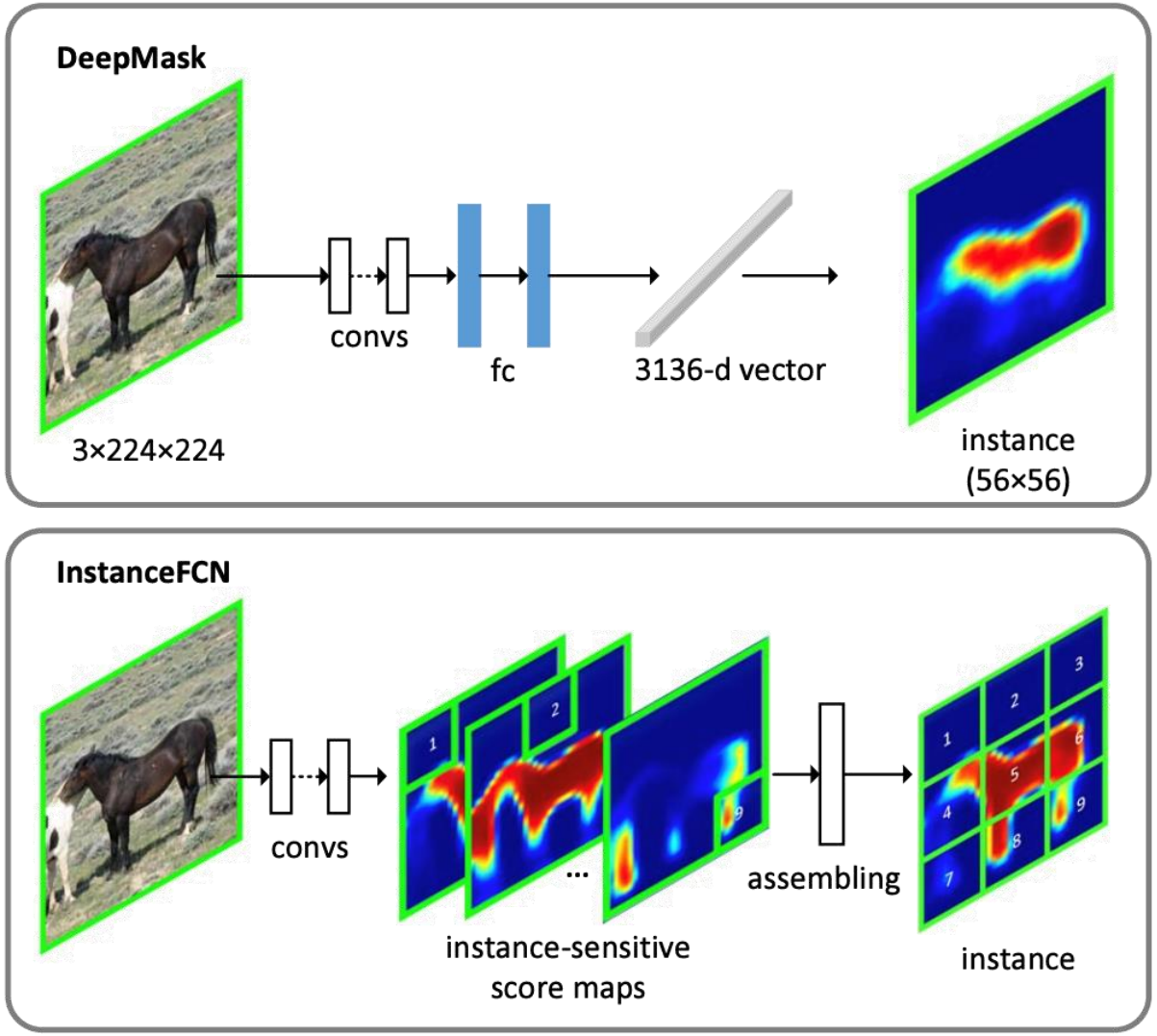
[Ref] Dai, Jifeng, et al. "Instance-sensitive fully convolutional networks." European Conference on Computer Vision. Springer, Cham, 2016.

InstanceFCN

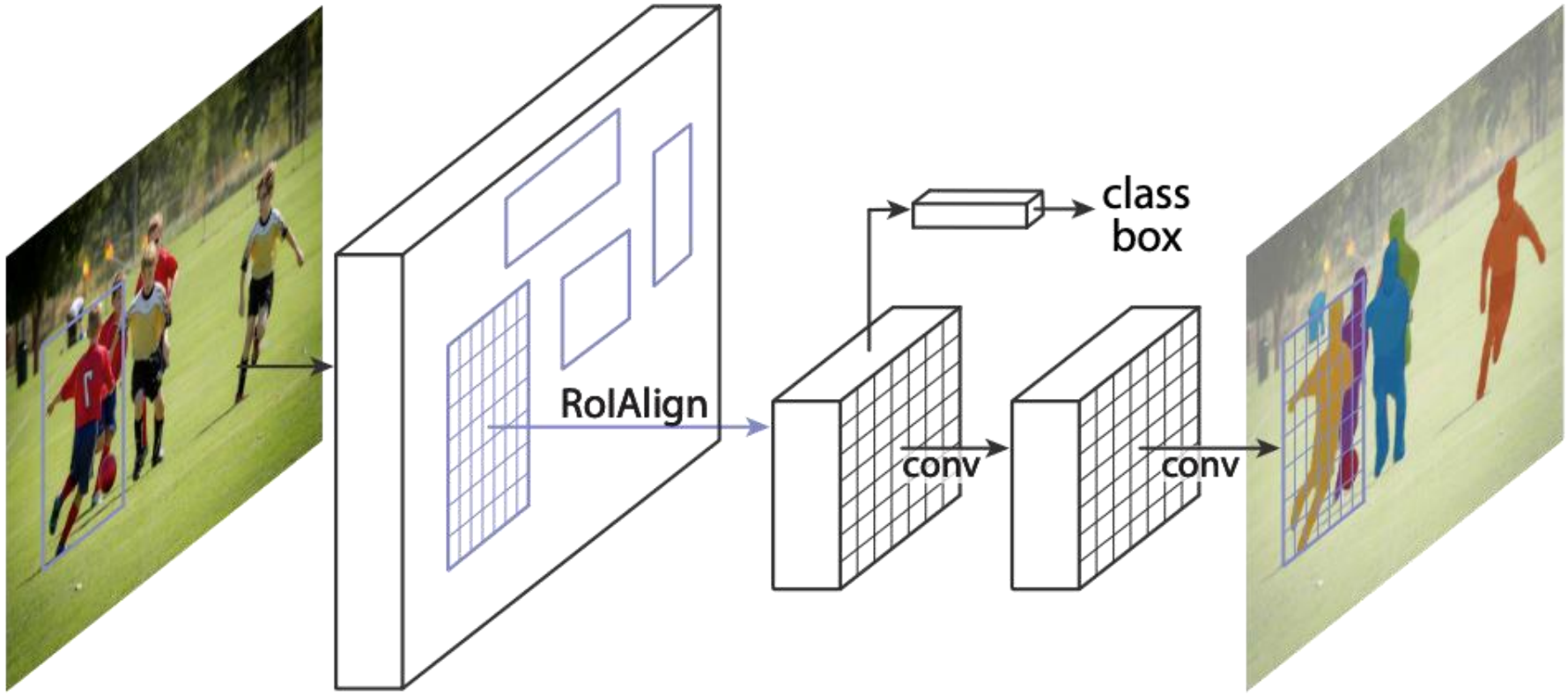


[Ref] Dai, Jifeng, et al. "Instance-sensitive fully convolutional networks." European Conference on Computer Vision. Springer, Cham, 2016.

InstanceFCN vs DeepMask



Mask R-CNN



[Ref] He, Kaiming, et al. "Mask r-cnn." Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017.

Q & A