

Robotic Navigation and Exploration

Week 10: Deep Learning for Computer Vision

Min-Chun Hu anitahu@cs.nthu.edu.tw

CS, NTHU / CSIE, NCKU

What is learning?

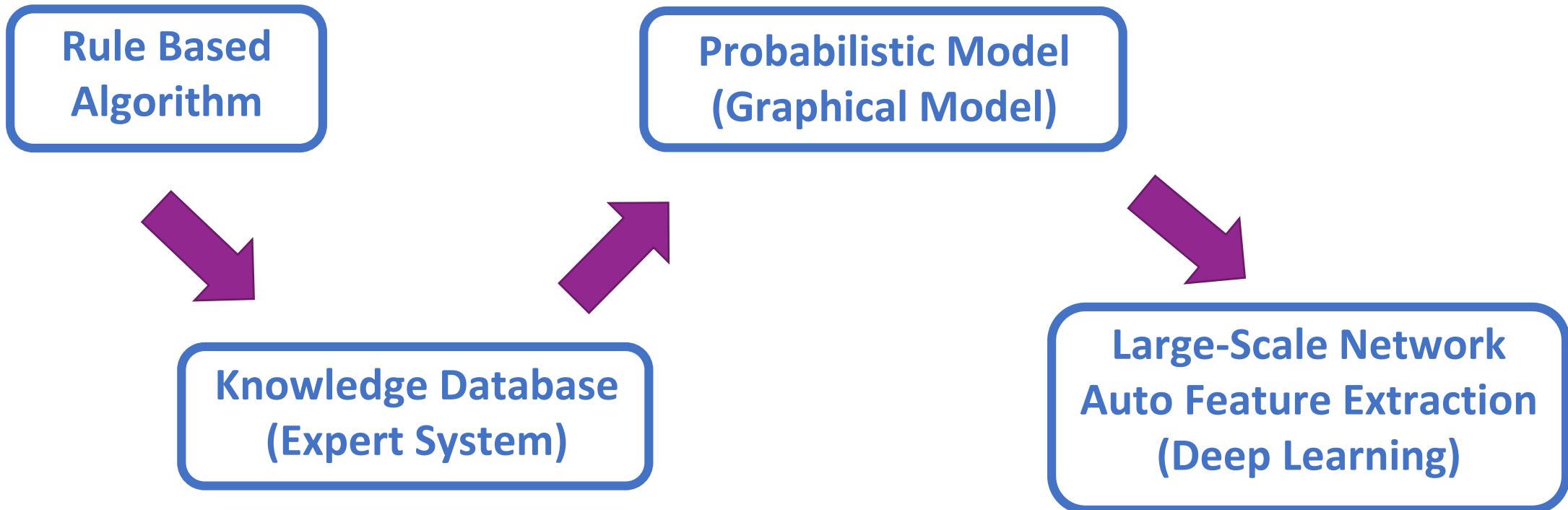
What is Learning ?

- 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

Video Clip of a Play



Human Heuristics

The first serve fails.

Two consecutive serves fails.

The opponent fails to return the serve.

The opponent returns the serve and players stay around the baselines.

The opponent returns the serve and players approach the net.

Fault

Double Fault

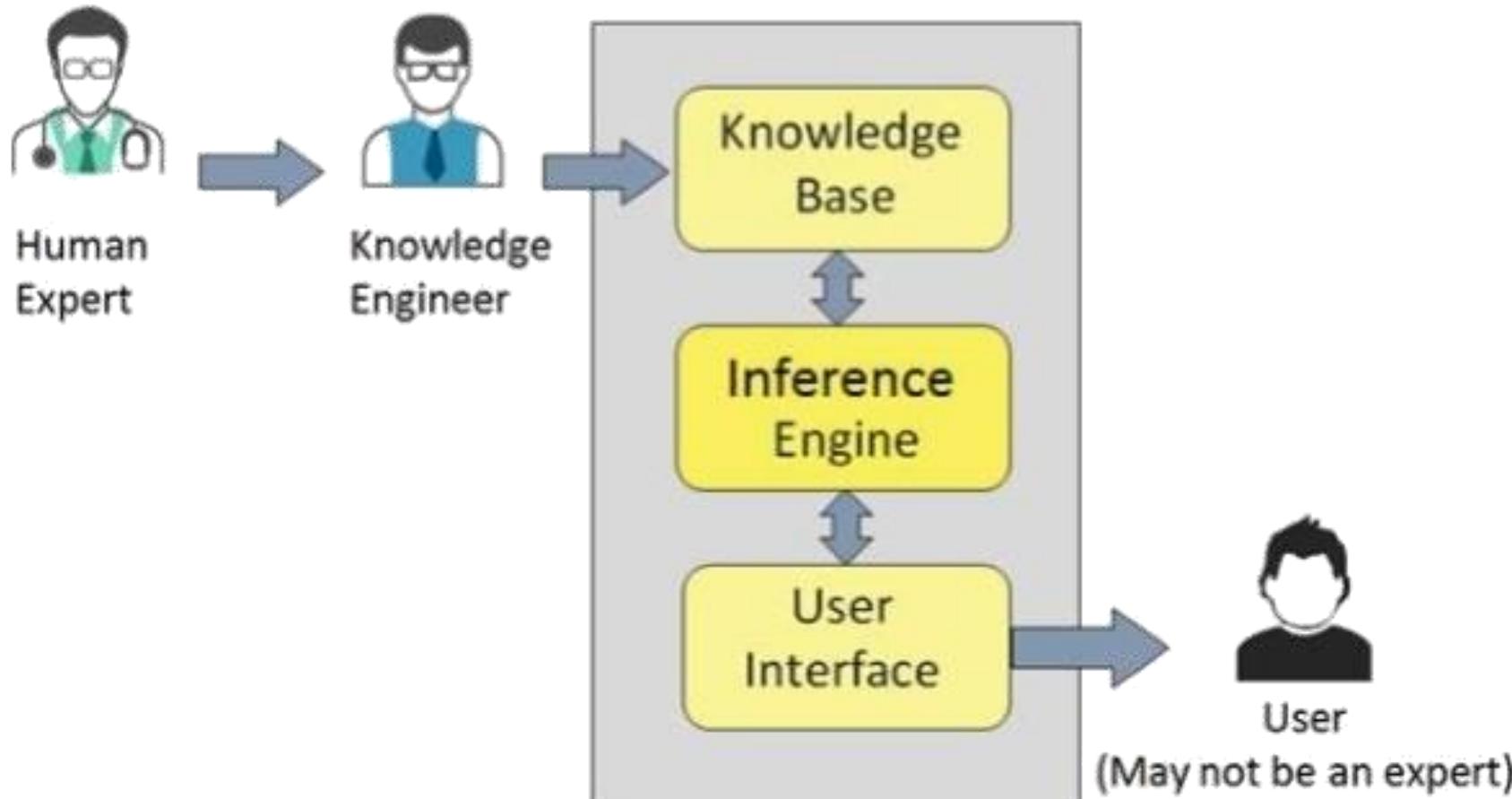
Ace or Unreturned Service

Baseline Rally

Net Approach

Categorized Explicit Events

Expert System



Probabilistic Graphical Model

Body Pose Estimation

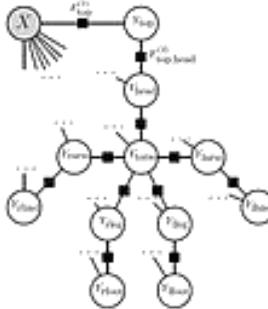
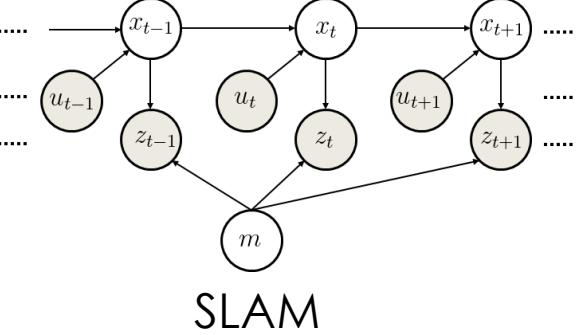
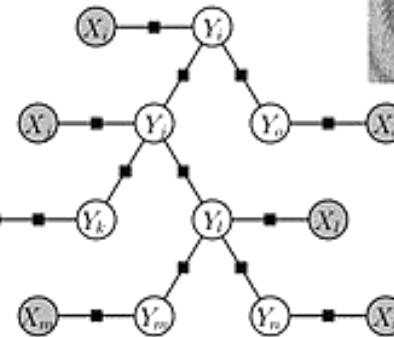
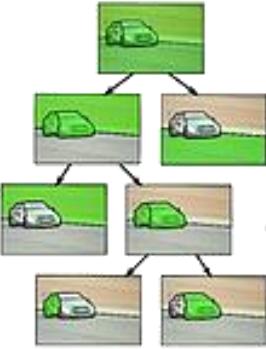


Image Denoising

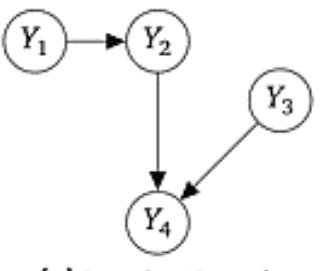


SLAM

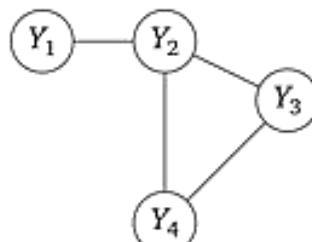
Semantic Image Segmentation



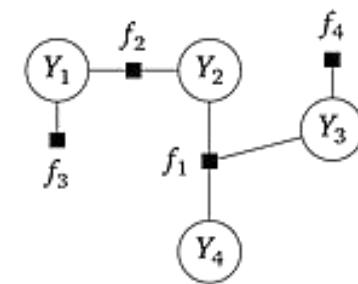
Graphical Models



(a) Bayesian Network

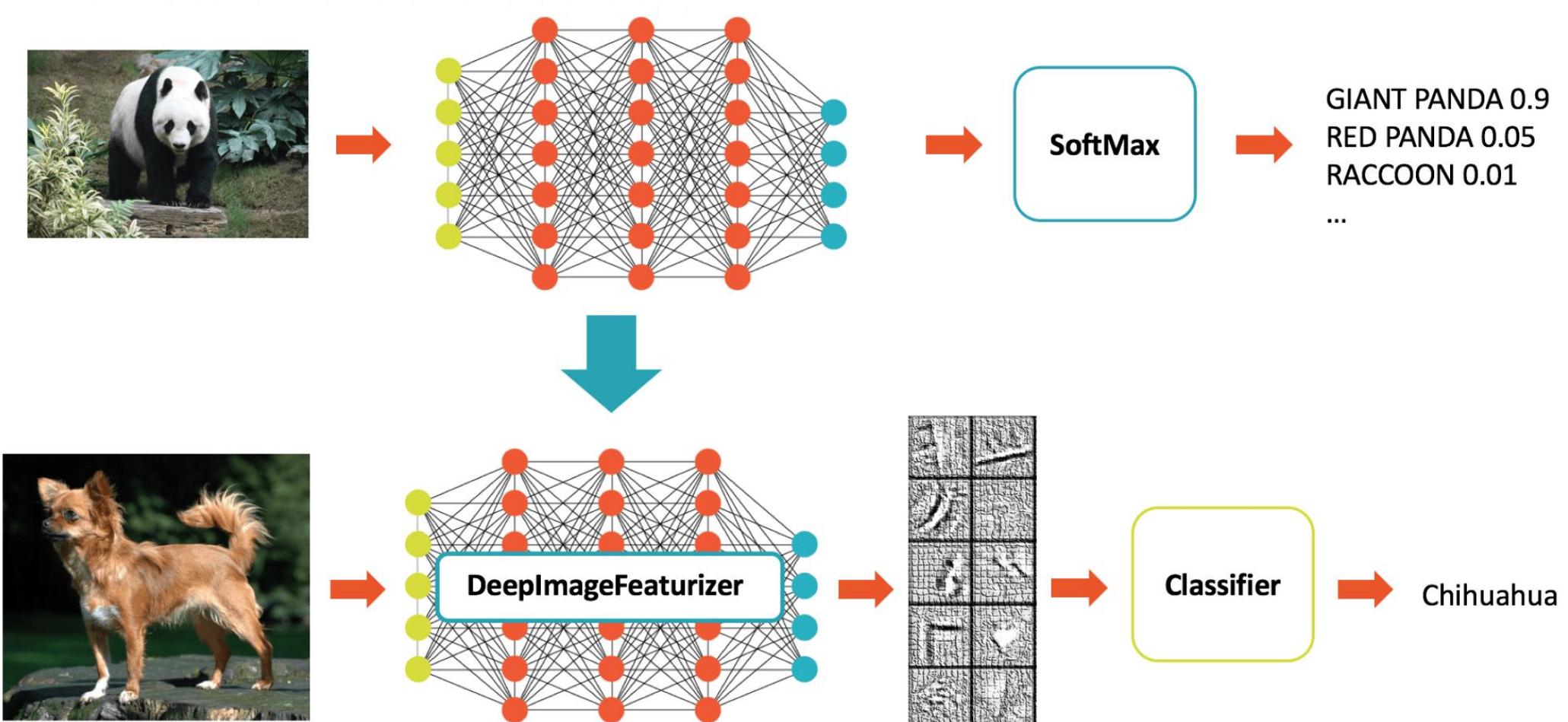


(b) Markov Random Field

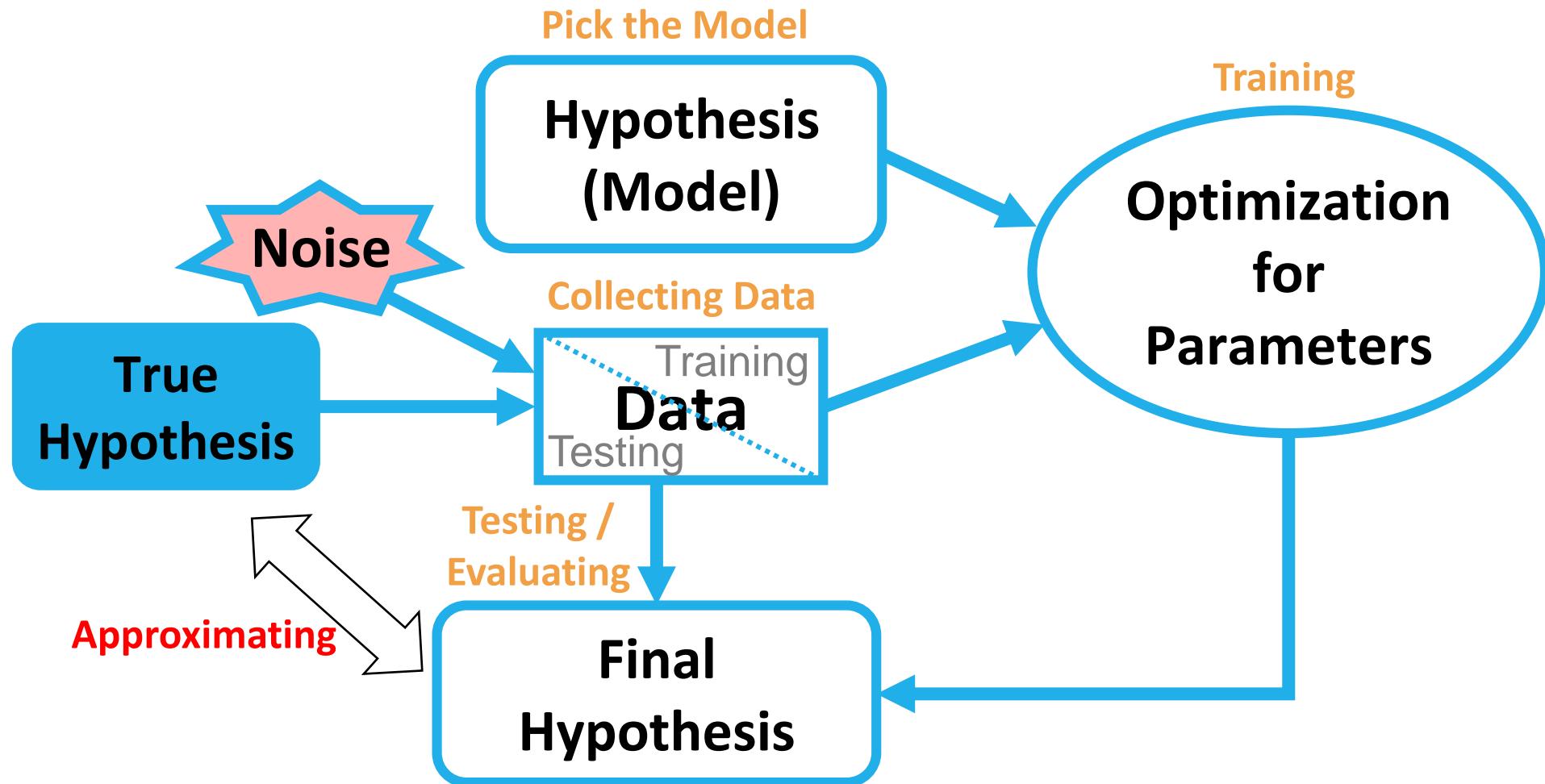


(c) Factor Graph

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)

.

.

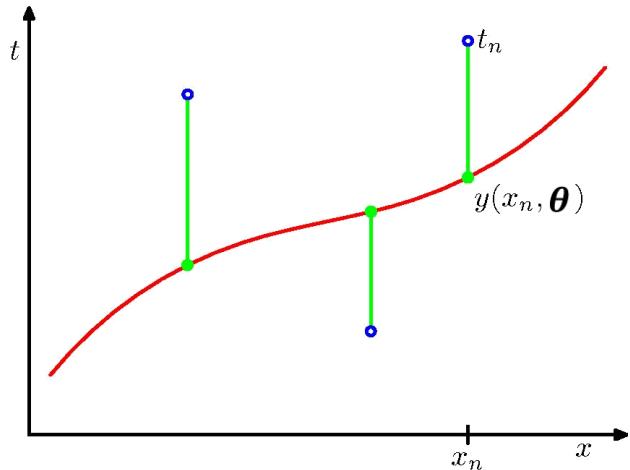
.

t_N Dog (2)

What is a well machine learning algorithm ?

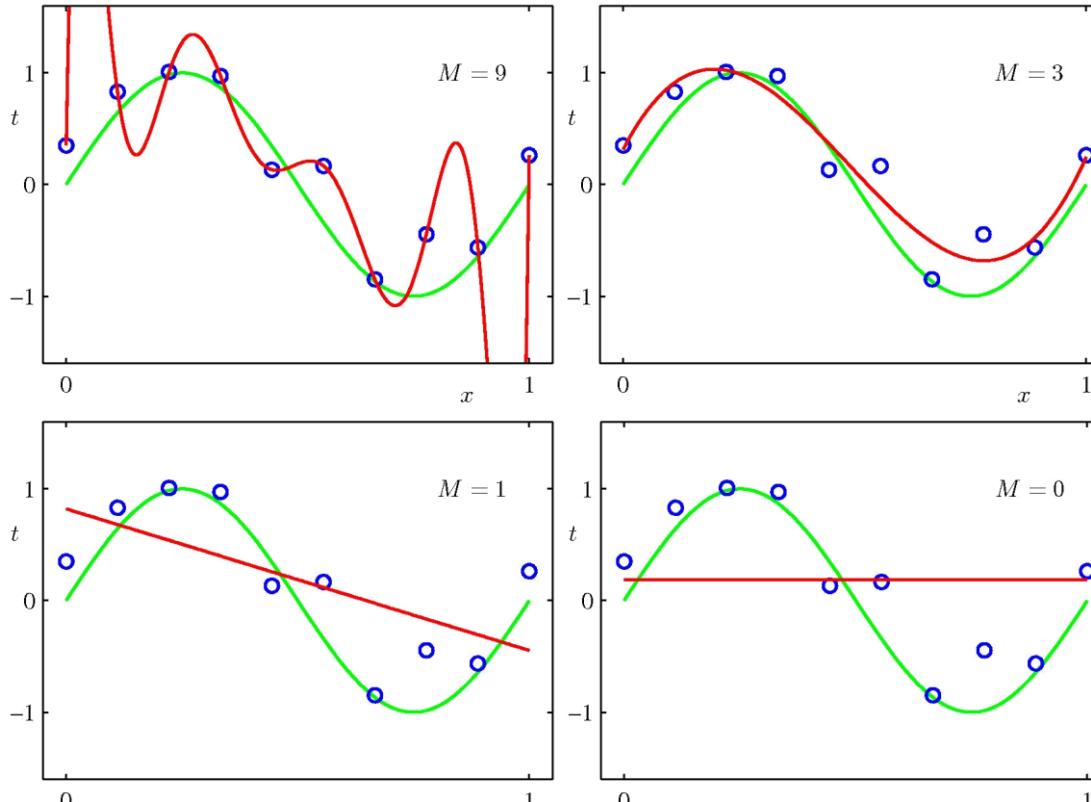
$$\tilde{E}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \boldsymbol{\theta}) - t_n\}^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2$$

Overfitting & Underfitting



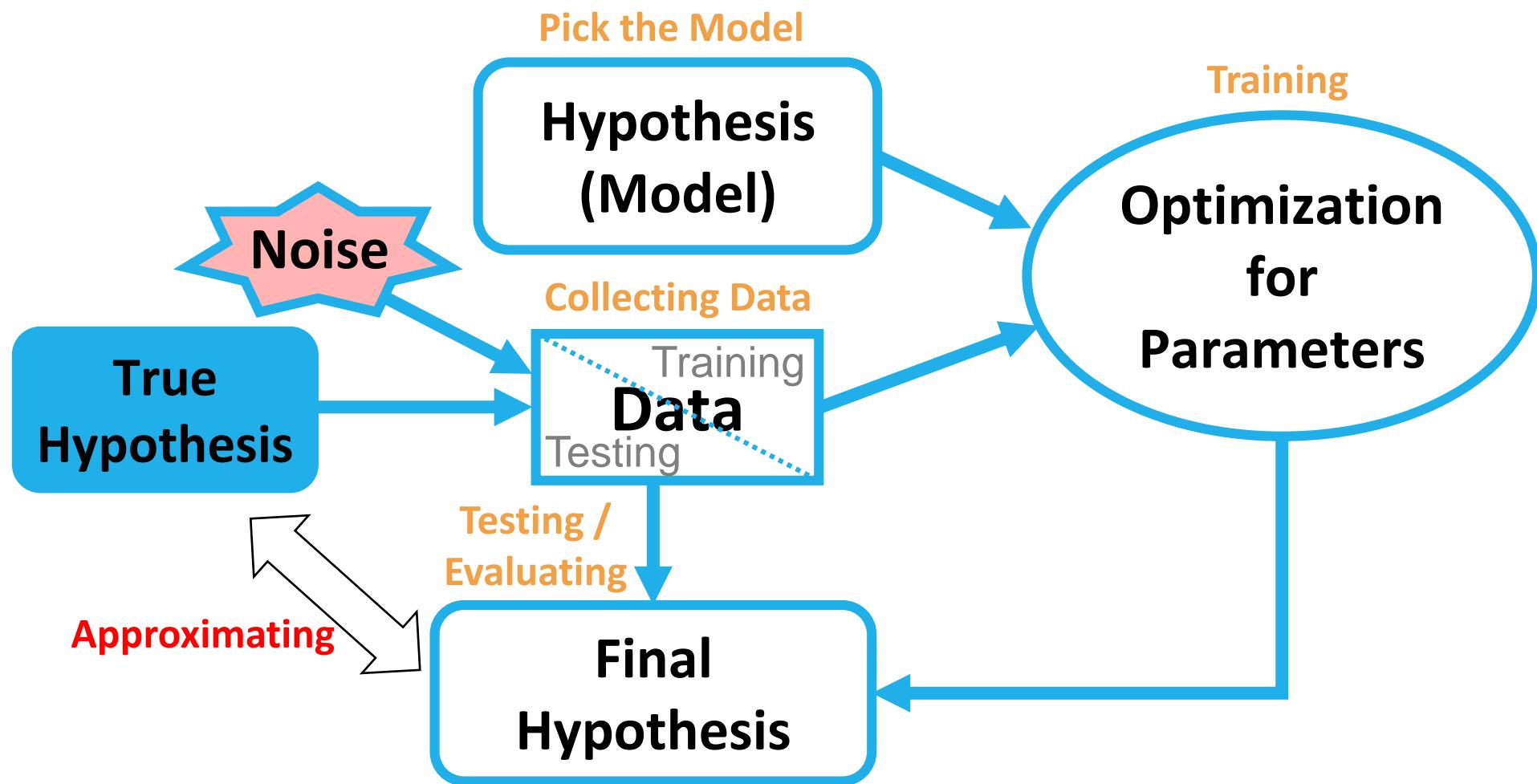
$$\tilde{E}(\theta) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \theta) - t_n\}^2 + \frac{\lambda}{2} \|\theta\|^2$$

Overfitting



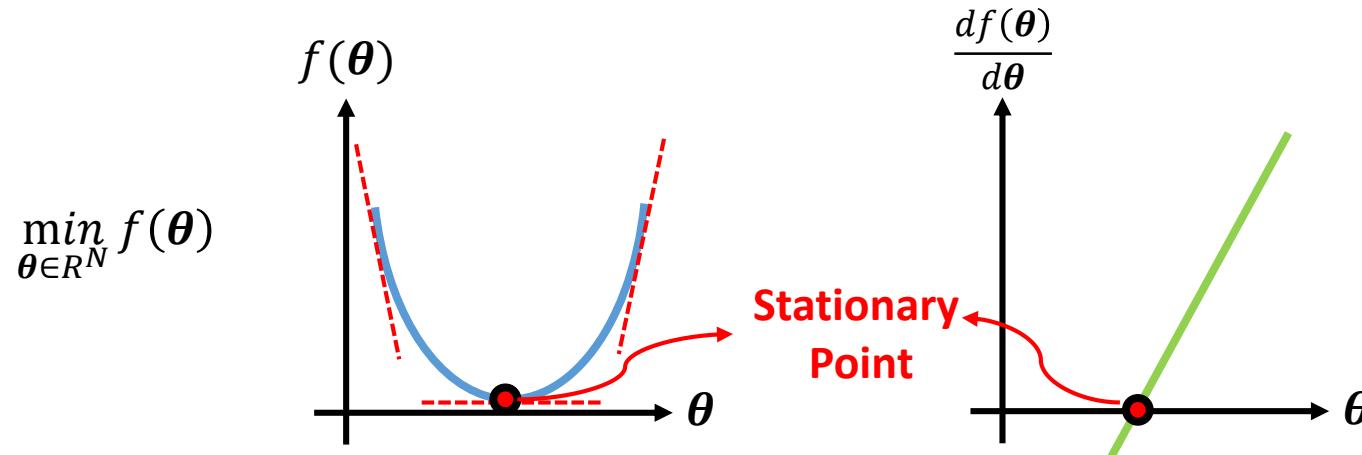
Underfitting

Machine Learning Process



Optimization Methods

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

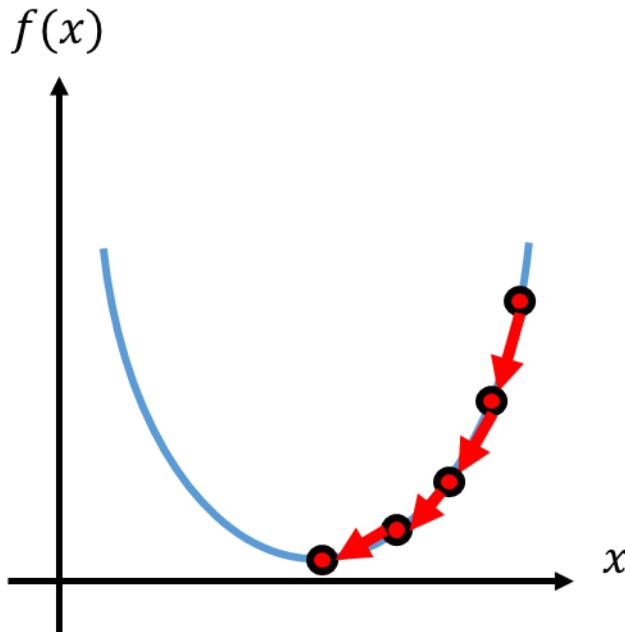


✓ Find the stationary point (first-order differential)

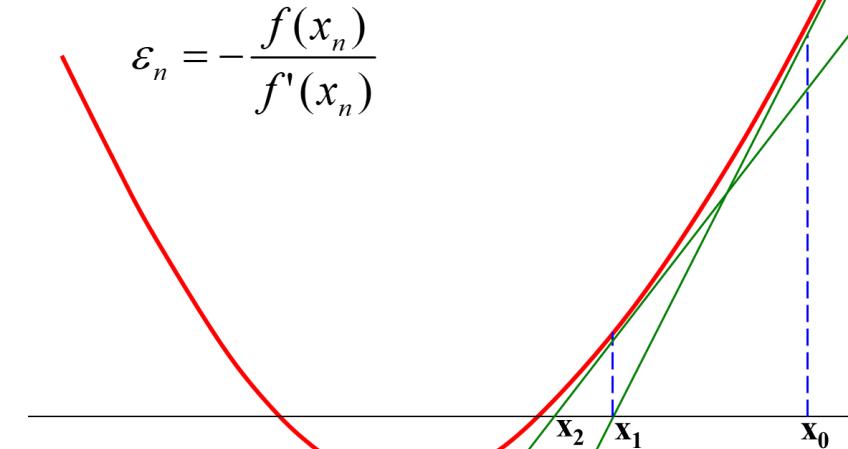
$$\nabla f(\theta) = 0$$

Optimization Methods

- Searching method:



First-order (Gradient Descent)
Fixed Step Length



Second-order (Newton's method)
Dynamic Step Length

Example : Linear Regression

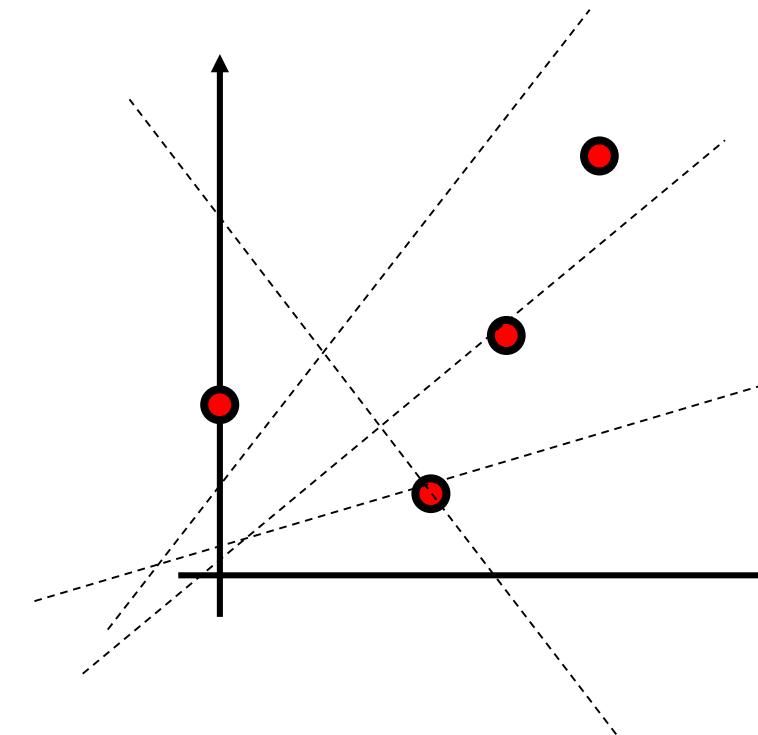
- Example: Linear Regression

- Collecting data

- Pick a model

$$f(x) = w_1x + w_0$$

- Find the suitable parameters w_0 and w_1



Example : Linear Regression

- Optimization

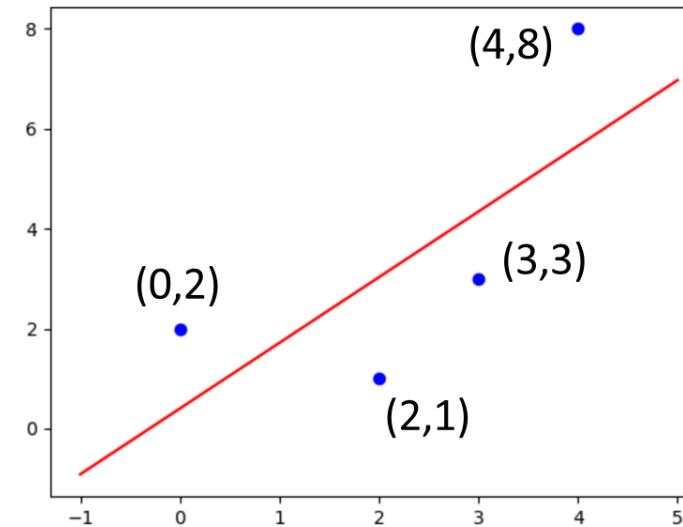
$$L(x, y) = \sum_i \frac{1}{2} (y_i - w_1 x_i - w_0)^2 \text{ (object function)}$$

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

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

Example : Linear Regression

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



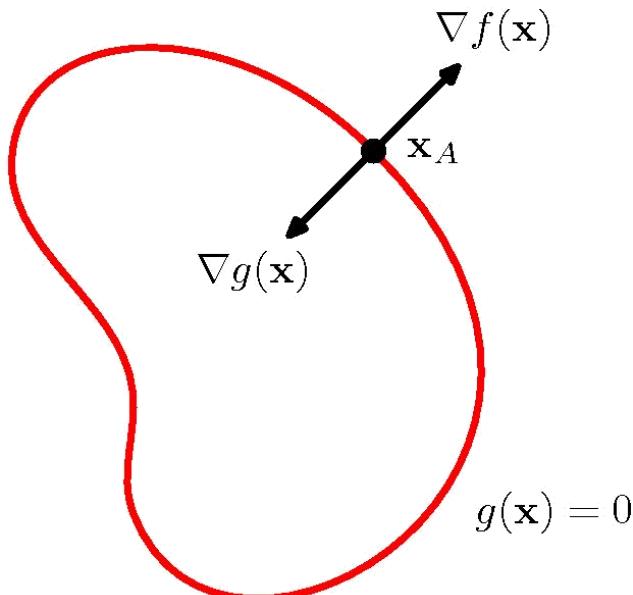
$$\begin{cases} (0) + (-2 + 4w_1 + 2w_0) + (-9 + 9w_1 + 3w_0) + (-32 + 16w_1 + 4w_0) = 0 \\ (-2 + w_0) + (-1 + 2w_1 + w_0) + (-3 + 3w_1 + w_0) + (-8 + 4w_1 + w_0) = 0 \end{cases}$$

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

Lagrange Multiplier

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

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



Lagrange Multiplier

$$f(\theta) + \lambda g(\theta) = 0$$

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 :

$$x_2 = h(x_1) \quad ?$$

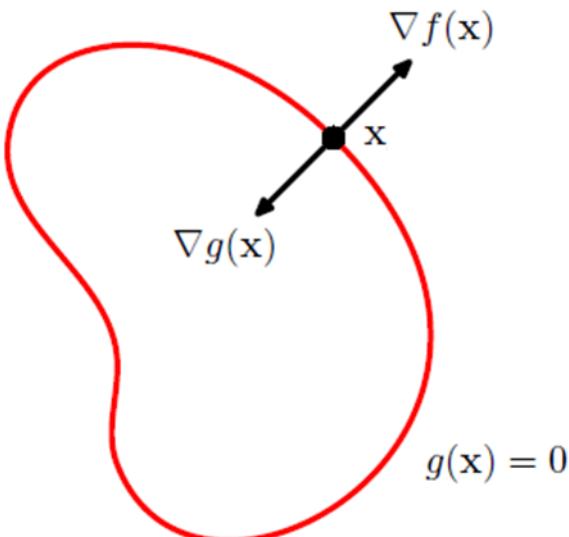
- Substitute x_2 into $f(x_1, x_2)$:

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

- 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 $g(\mathbf{x}) = 0$



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

λ : Lagrange Multiplier

By Taylor expansion around \mathbf{x} :

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

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

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

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

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

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

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

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

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

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

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

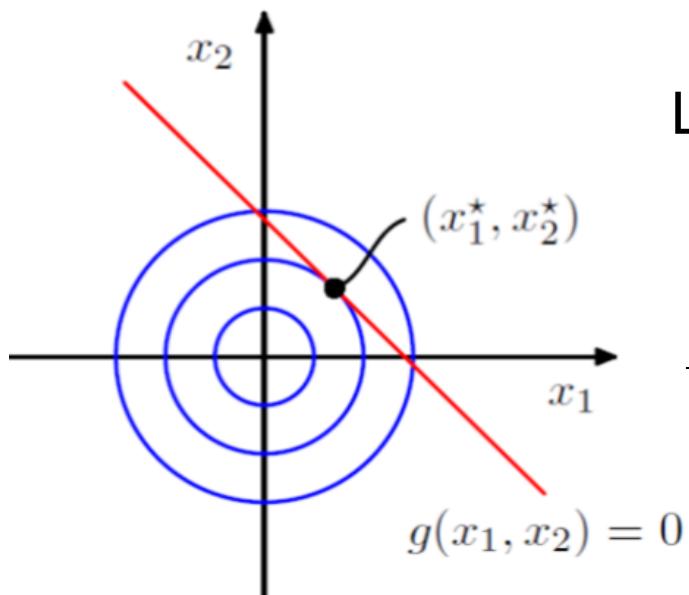
(constrained stationary condition)

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

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

Example

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



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

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

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

Lagrange Multipliers with Inequality Constraint

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

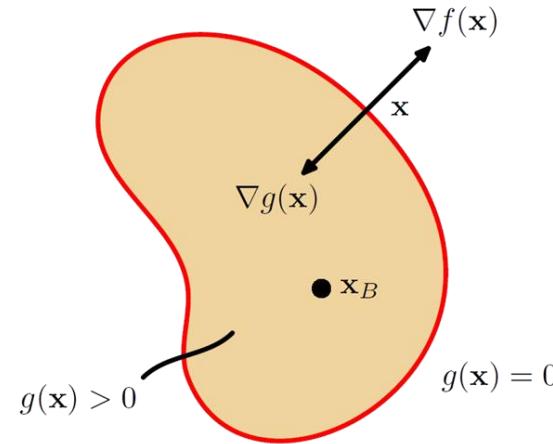
→ Optimize $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$
with respect to \mathbf{x} and λ subject to

$$g(\mathbf{x}) \geq 0$$

$$\lambda \geq 0$$

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

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:

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}) \text{ with } \lambda = 0$$

$$\therefore \lambda = 0$$

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

[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:

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}) \text{ with } \lambda \neq 0$$

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

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

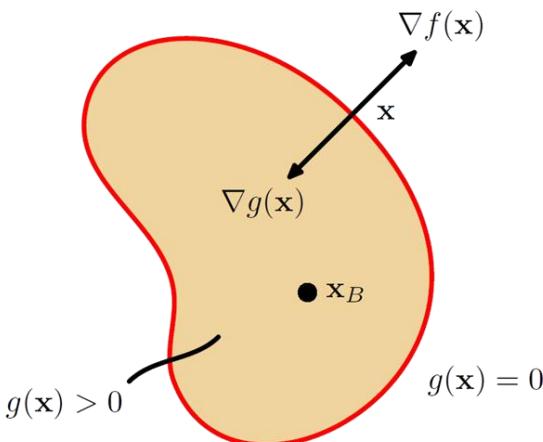
∴ $f(\mathbf{x})$ will be at a maximum only if $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ have different direction.

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} :

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

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

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

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

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

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

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

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

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

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

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

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

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

(constrained stationary condition)

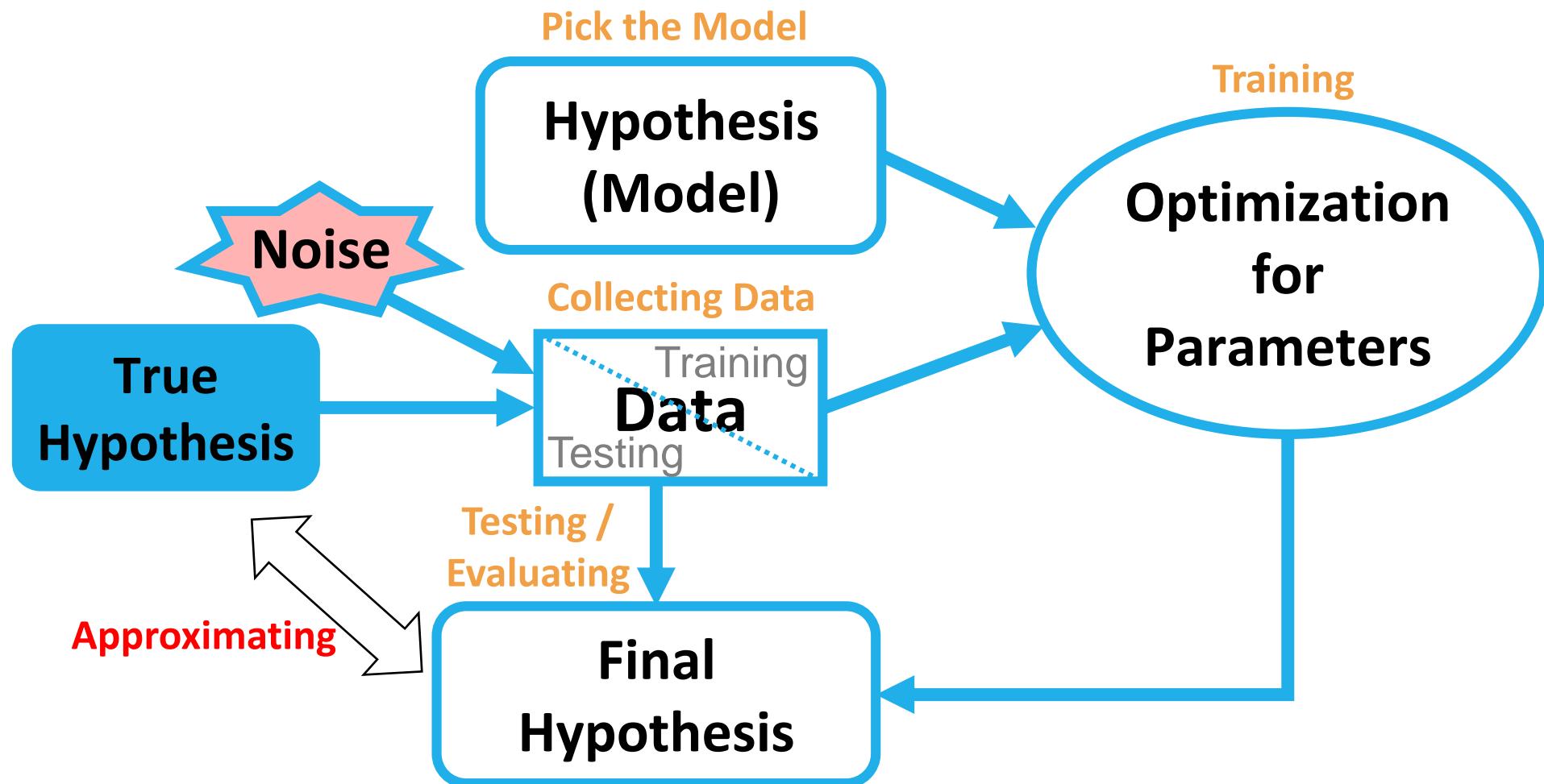
$$\lambda = 0$$

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

- 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

$$-\frac{\# \text{Correct}}{\# \text{Total Sample}} = \frac{\# \text{TP} + \# \text{TN}}{\# \text{TP} + \# \text{FN} + \# \text{FP} + \# \text{TN}}$$

- Precision

$$-P = \frac{\# \text{TP}}{\# \text{TP} + \# \text{FP}}$$

- Recall (Sensitivity)

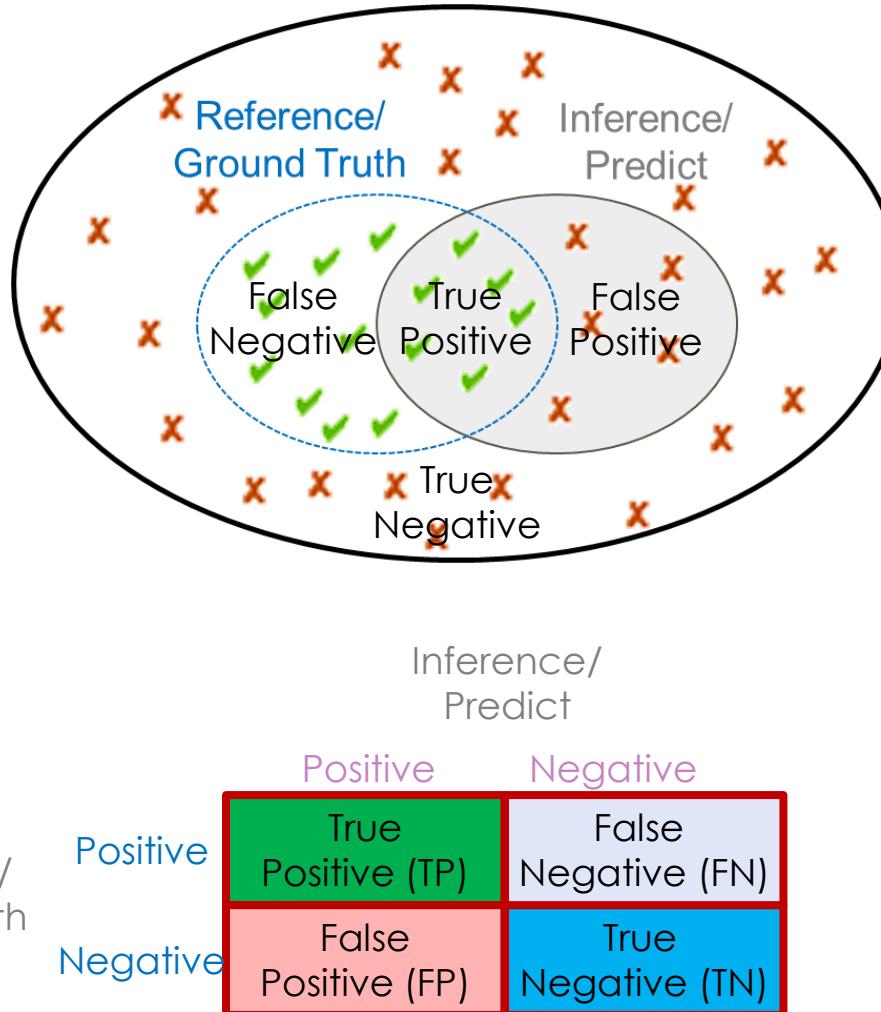
$$-R = \frac{\# \text{TP}}{\# \text{TP} + \# \text{FN}}$$

- F1-Measure

$$-F1 = \frac{2PR}{P+R}$$

- AP, MAP

Reference/
Ground Truth



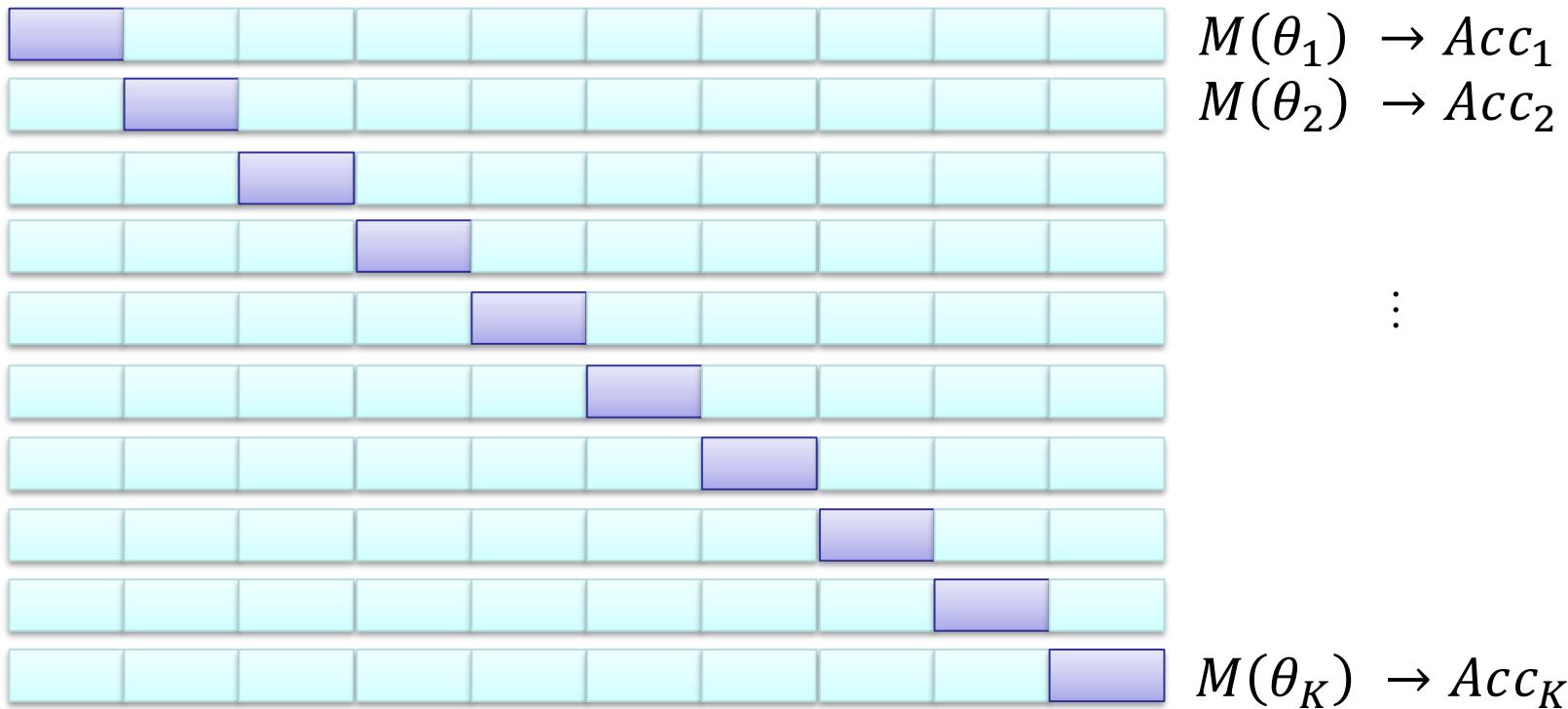
Validation

Dataset

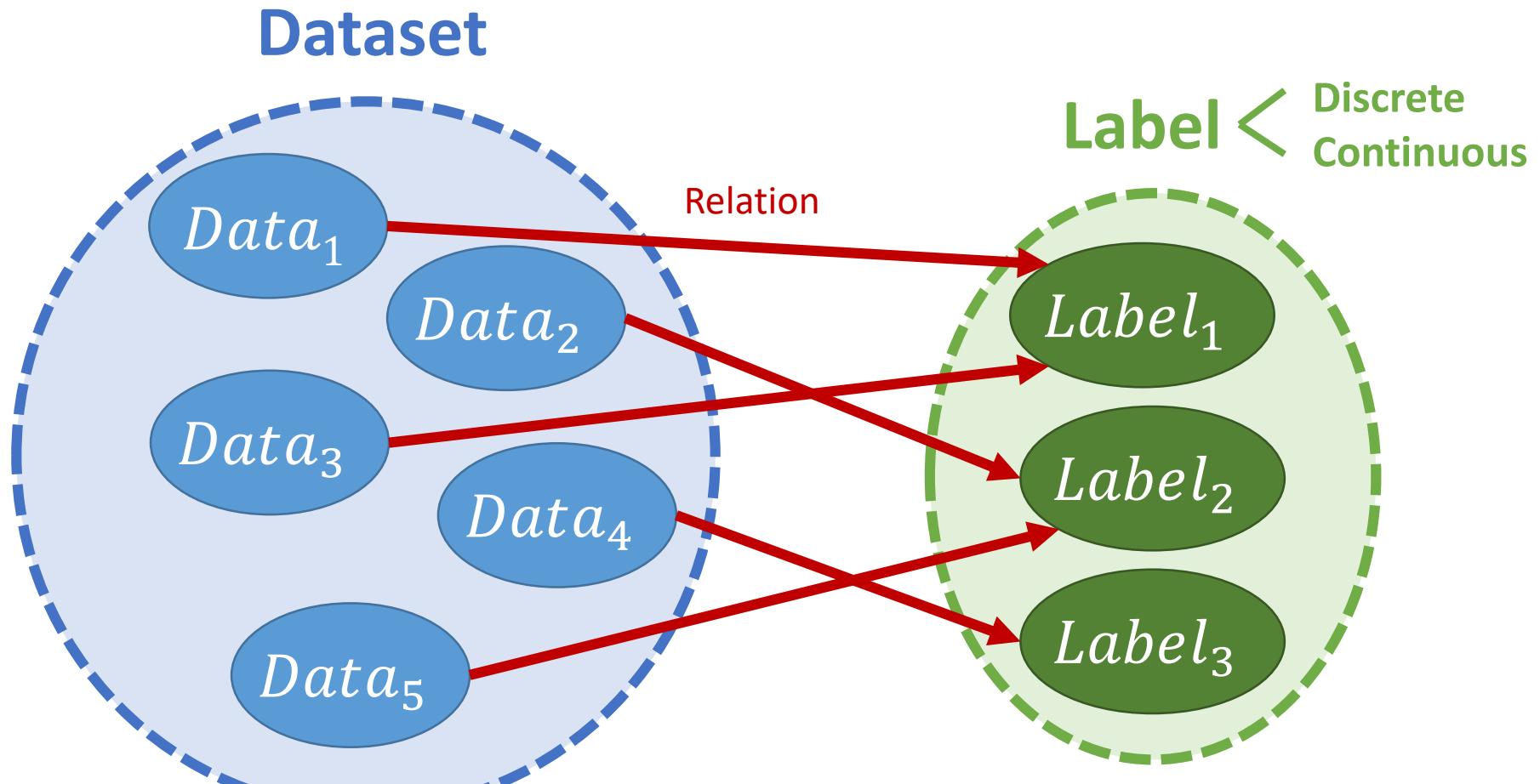
Training Set

Testing Set

- K-fold cross validation
- Leave One Out



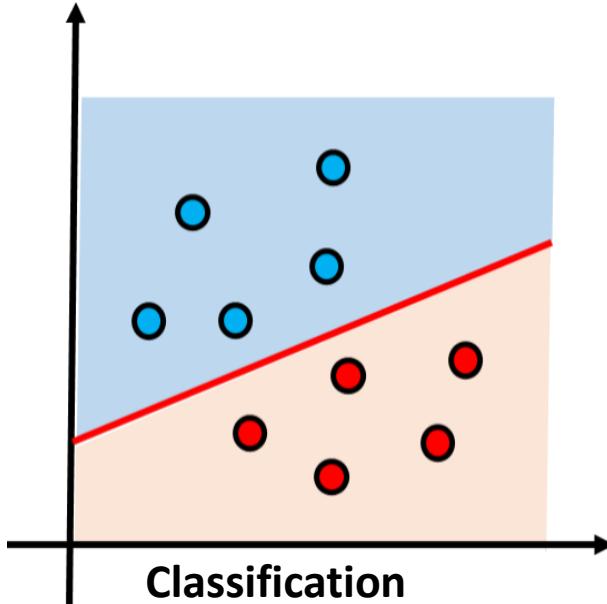
Supervised Learning



Find the relation between data and label.

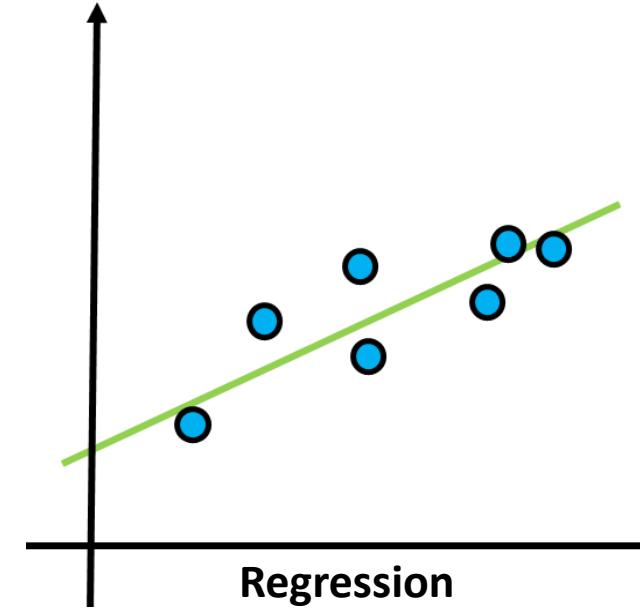
Classification vs. Regression

Discrete Label



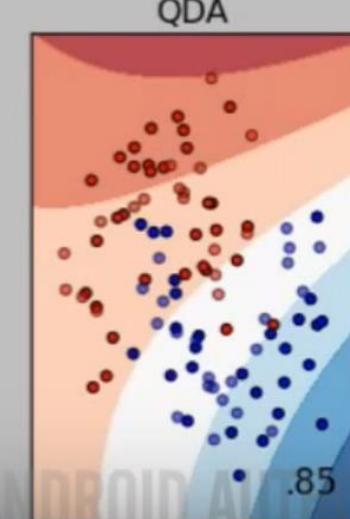
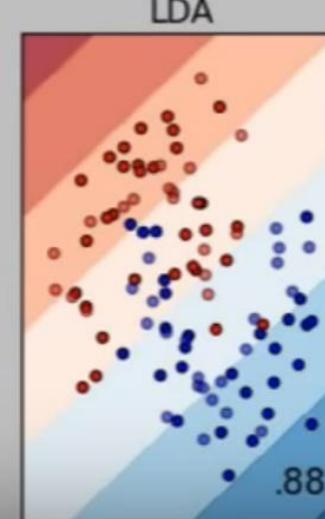
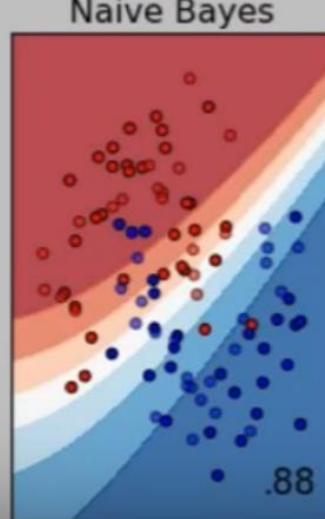
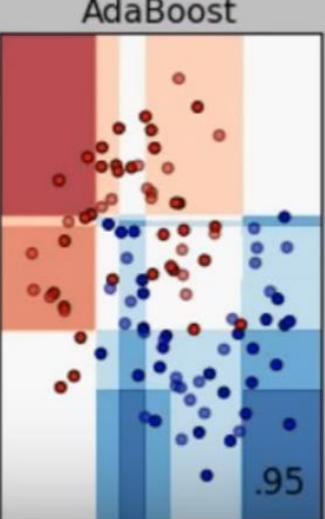
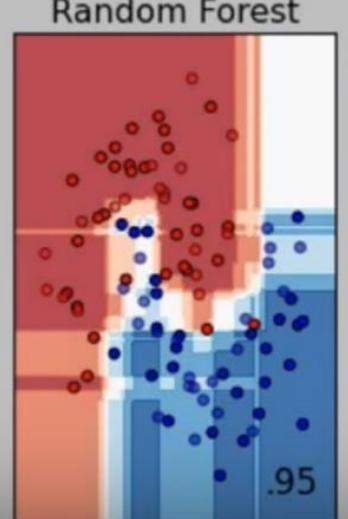
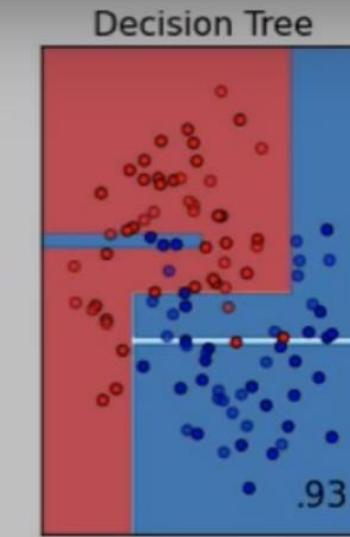
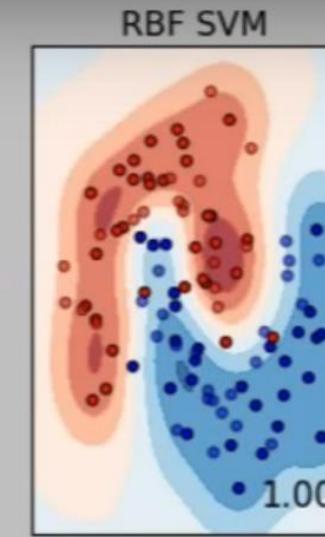
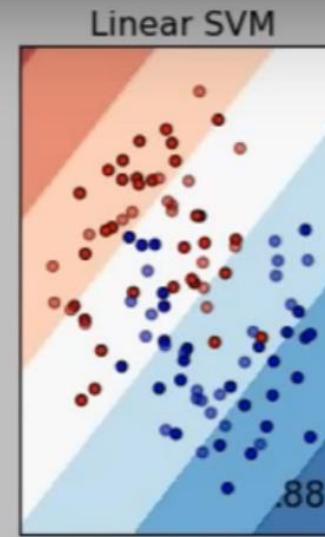
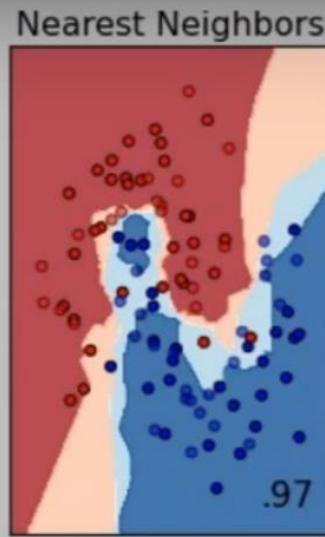
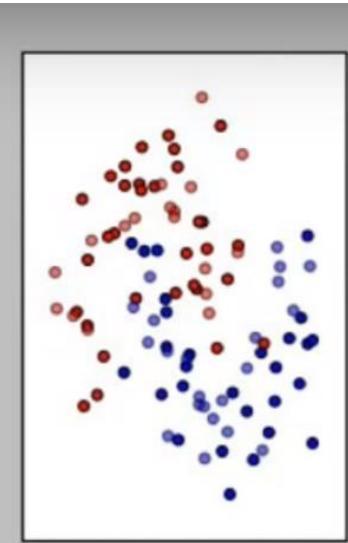
- ✓ Logistic Regression
- ✓ Support Vector Machine (SVM)
- ✓ Neural Network (NN)

Continuous Label

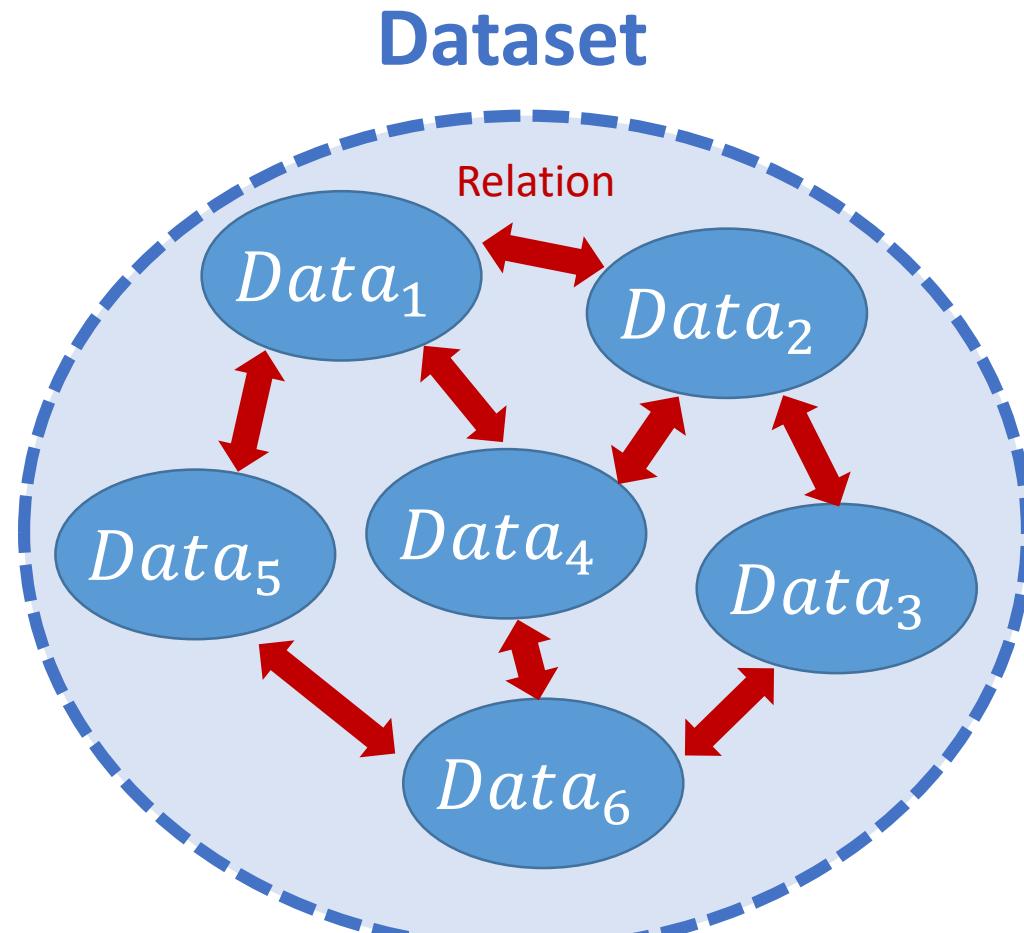


- ✓ Linear Regression
- ✓ Support Vector Regression (SVR)
- ✓ Neural Network (NN)

Visualization of Classification Methods



Unsupervised Learning

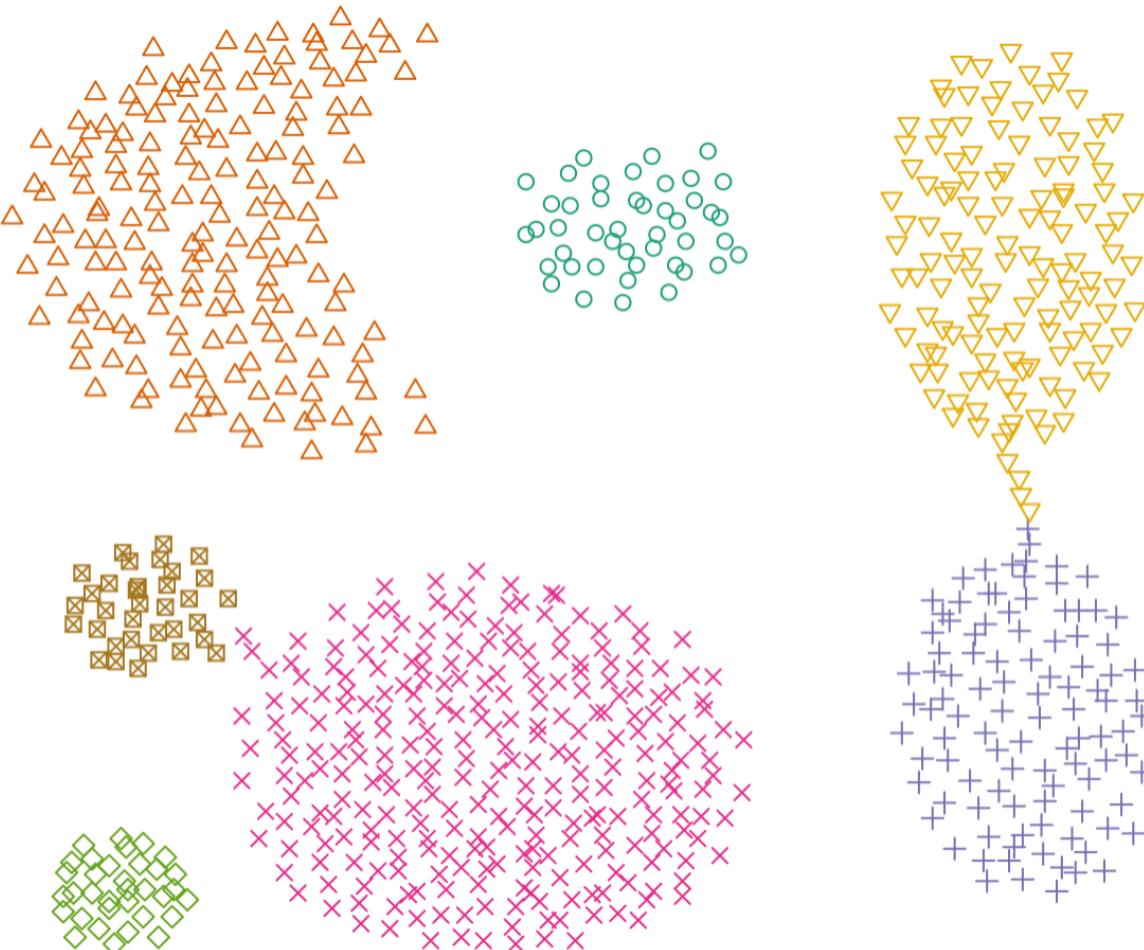


Find the relation between data points.

Unsupervised Learning

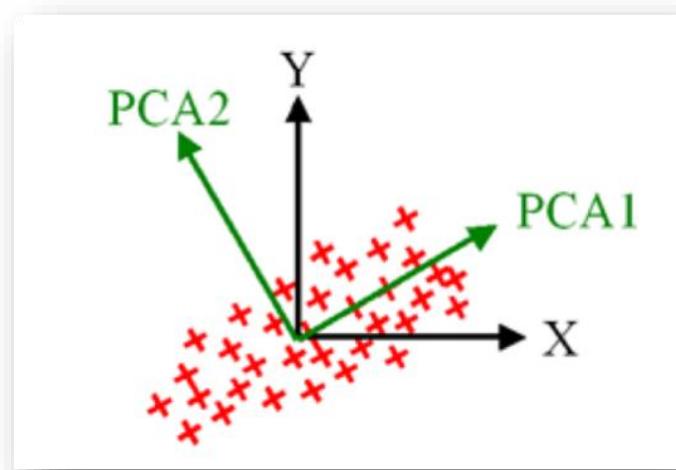
- Clustering

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

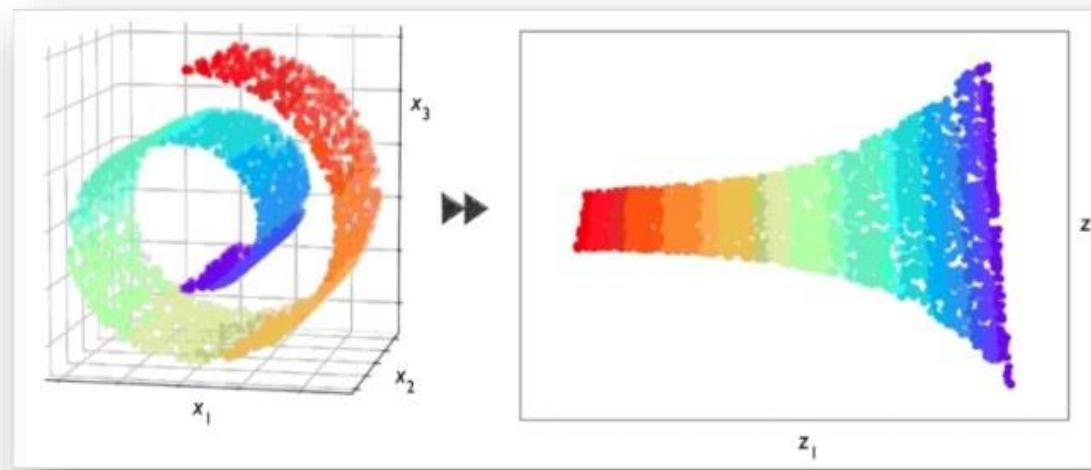


Unsupervised Learning

- Dimension Reduction
 - PCA
 - LLE/TSNE/AutoEncoder



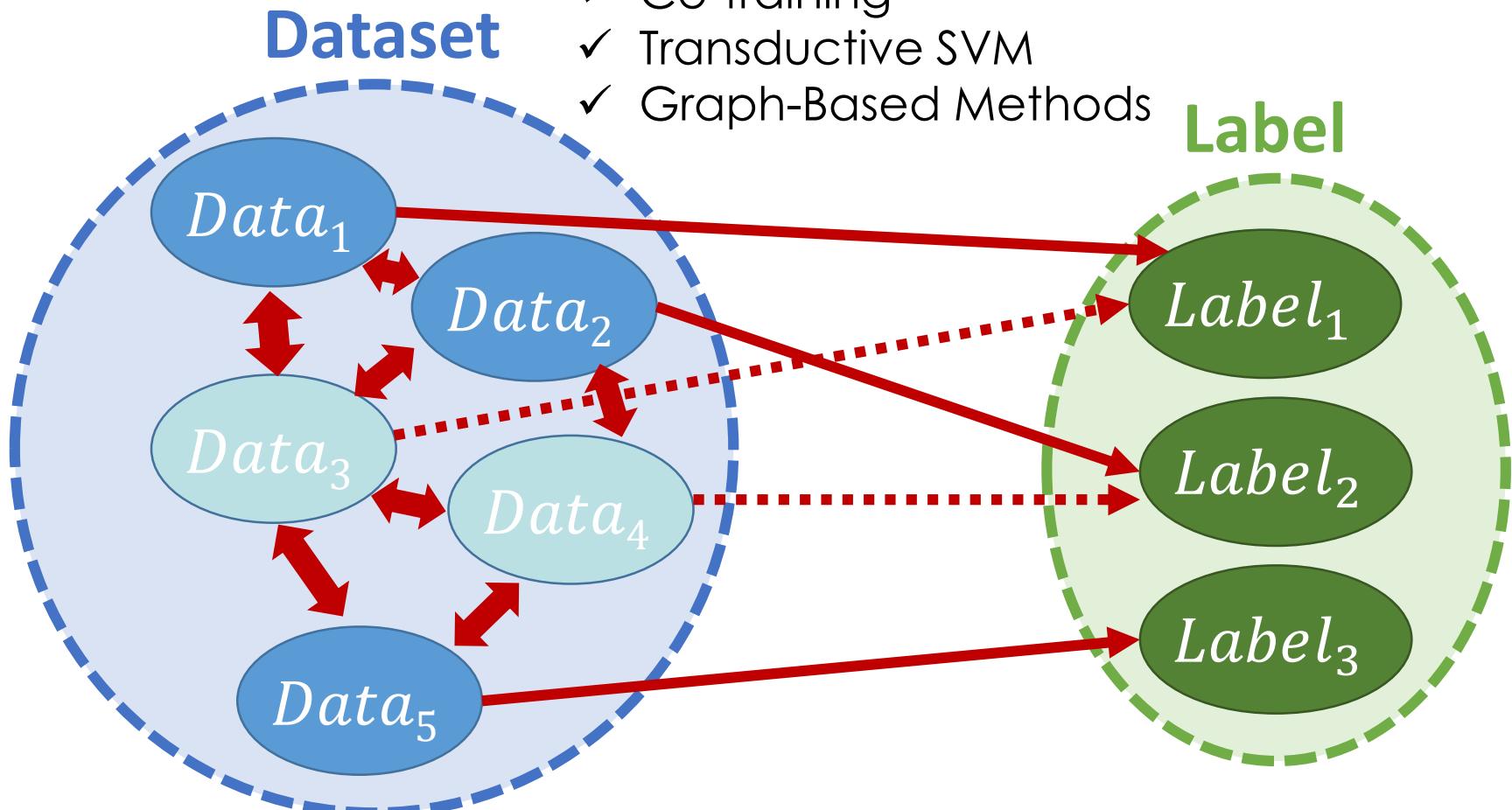
Linear



Non-Linear

Semi-supervised Learning

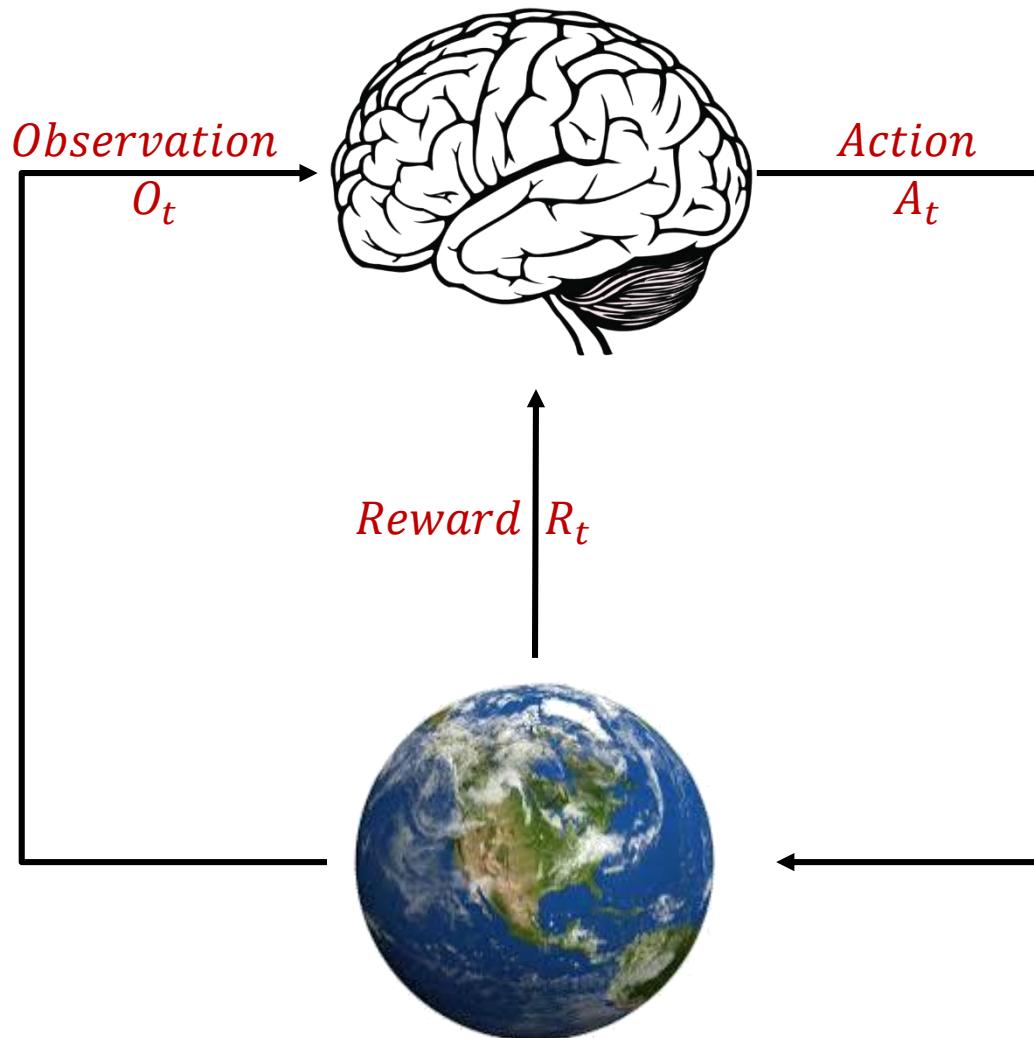
- ✓ EM with generative mixture models
- ✓ Self Training
- ✓ Co-training
- ✓ Transductive SVM
- ✓ Graph-Based Methods



Reinforcement Learning

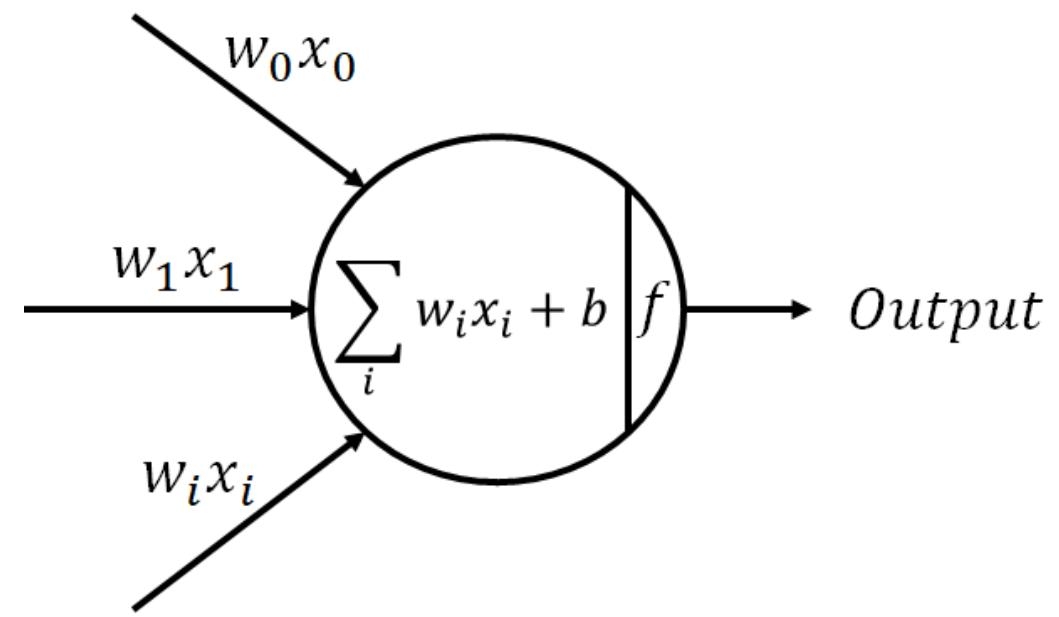
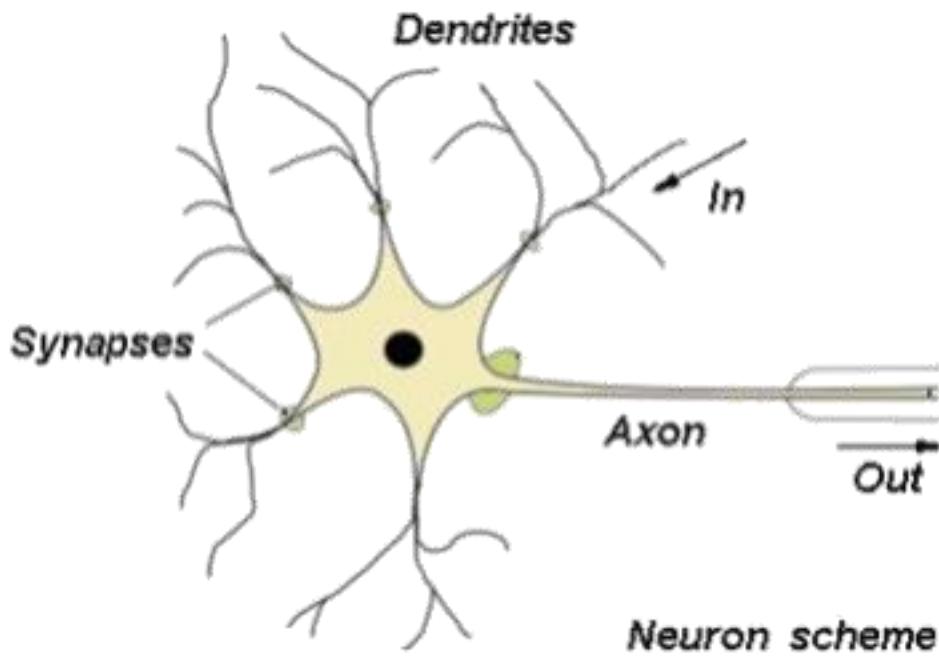
At each step t

- 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}



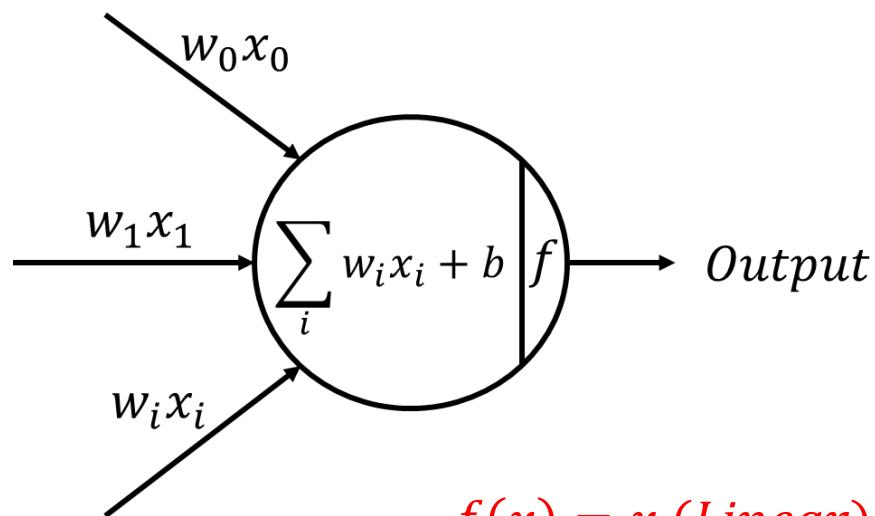
Deep Learning

Neuron

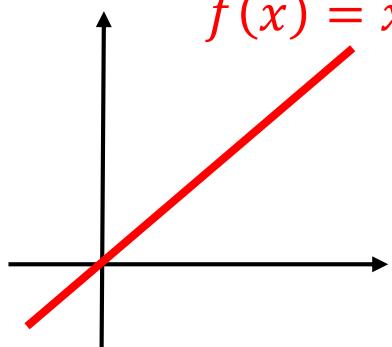


Linear/Logistic Regression

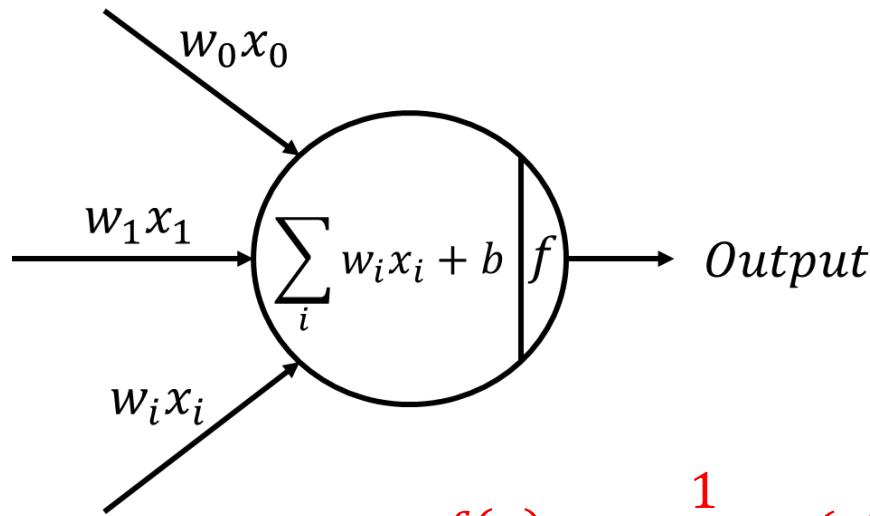
Linear Regression



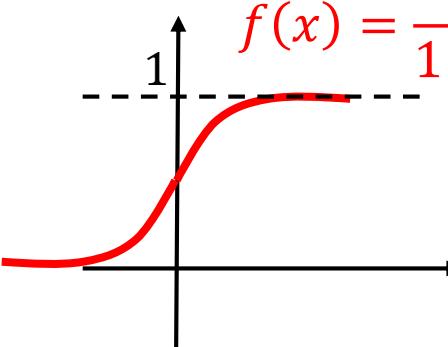
$$f(x) = x \text{ (Linear)}$$



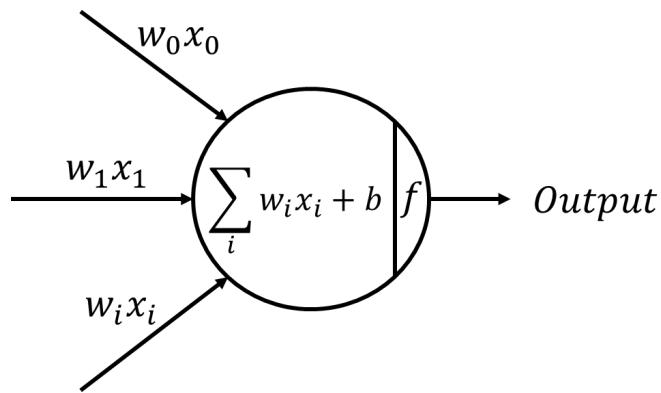
Logistic Regression



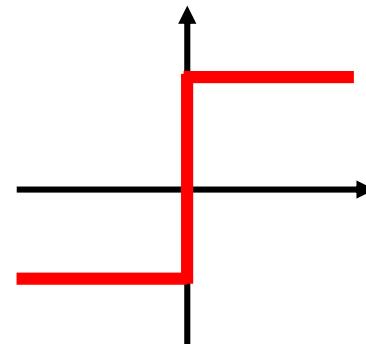
$$f(x) = \frac{1}{1 + e^{-x}} \text{ (sigmoid)}$$



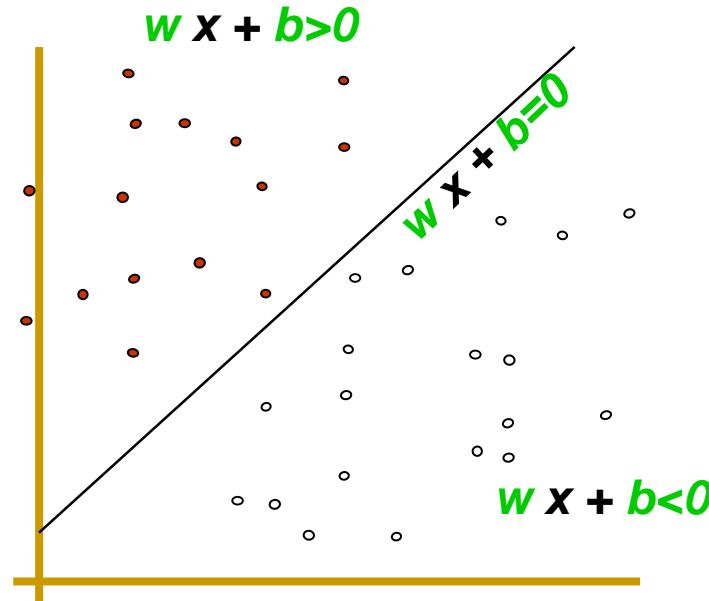
Perceptron



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



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



The Perceptron Learning Algorithm (PLA)

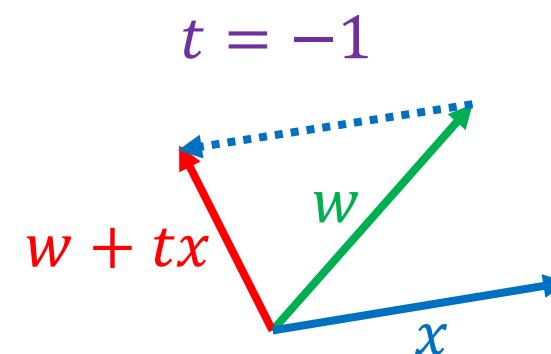
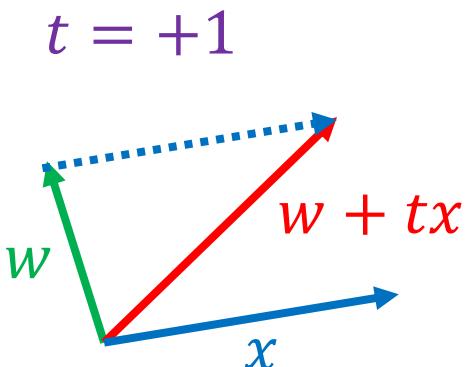
$$y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x}) \quad f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

Perceptron criterion:

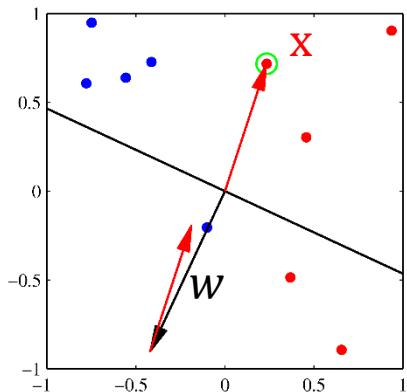
$$E_P(\mathbf{w}) = - \sum_i \mathbf{w}^T \mathbf{x}_i t_i$$

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

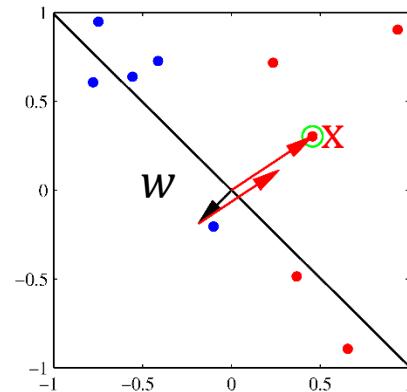
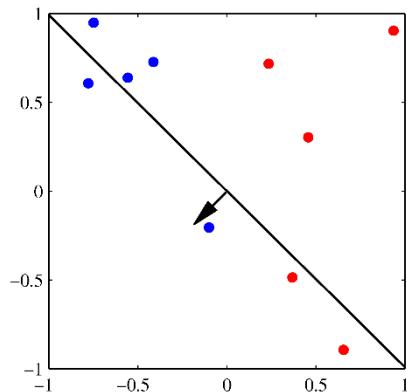
η : Learning Rate



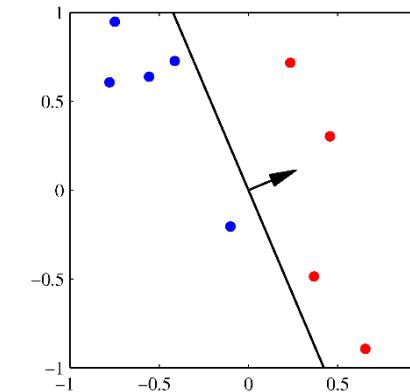
The Perceptron Learning Algorithm (PLA)



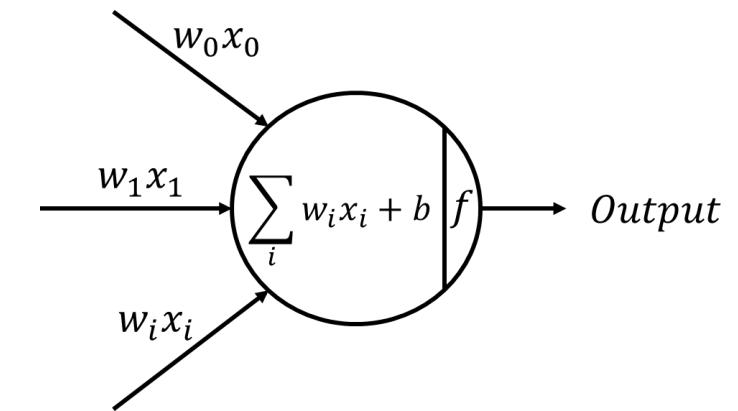
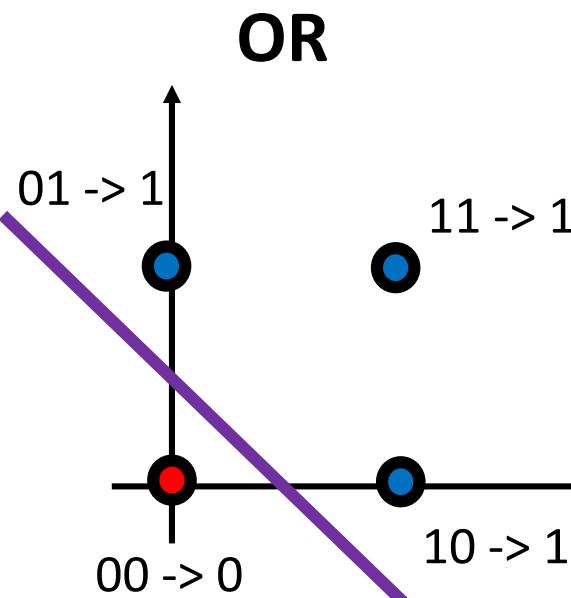
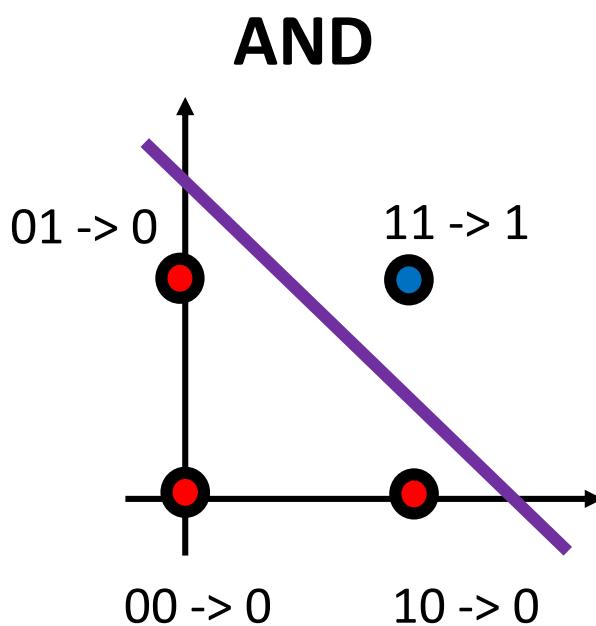
$t = +1$



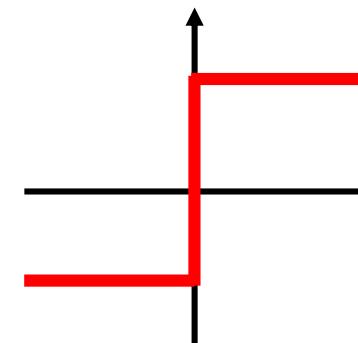
$t = +1$



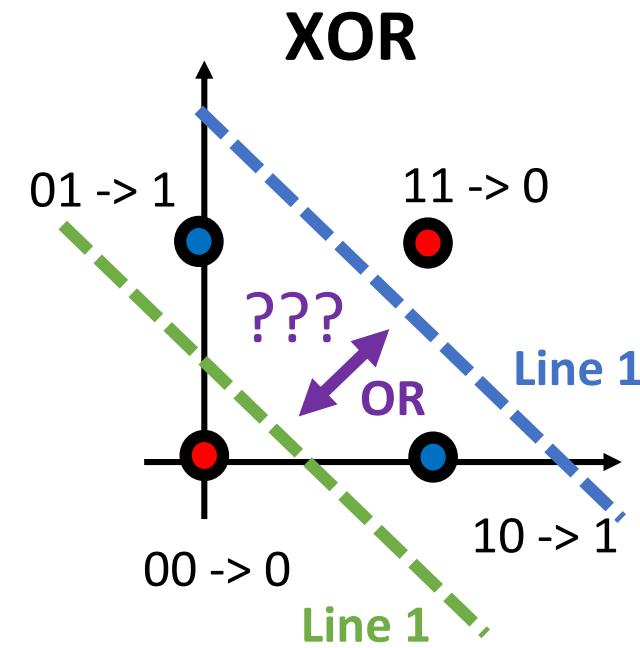
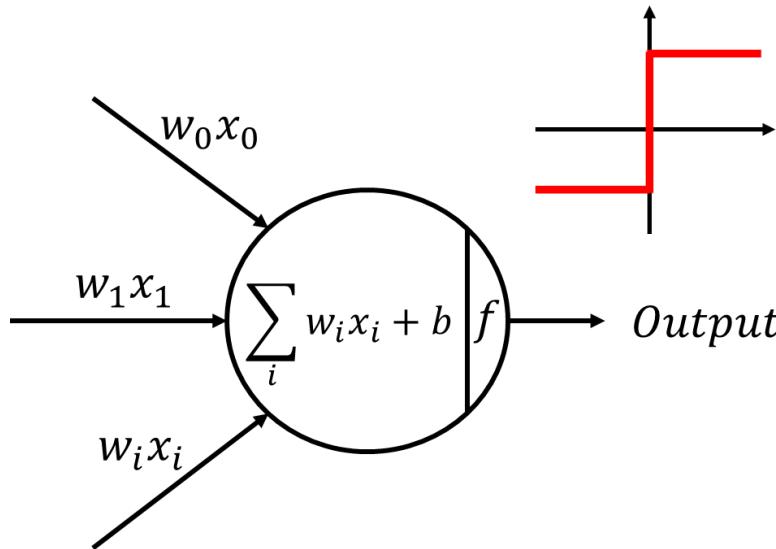
Perceptron



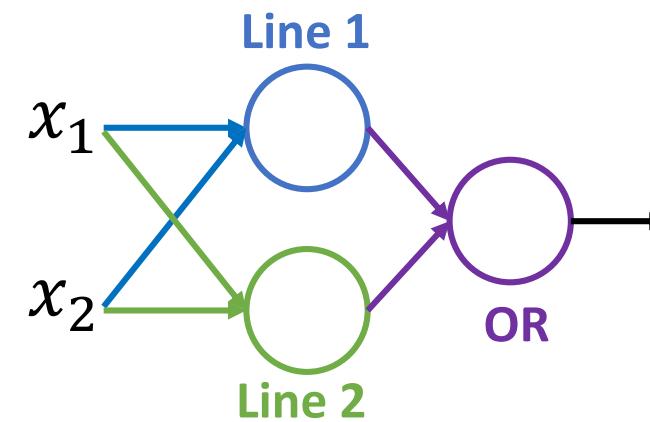
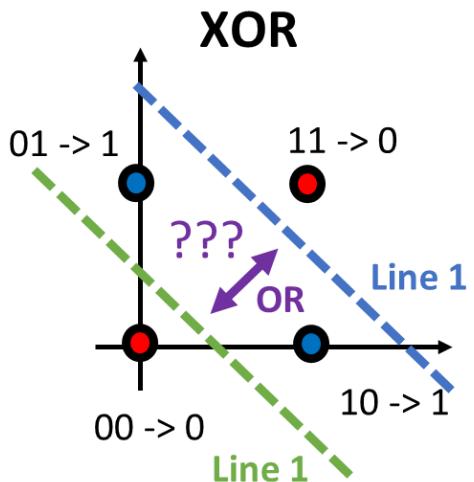
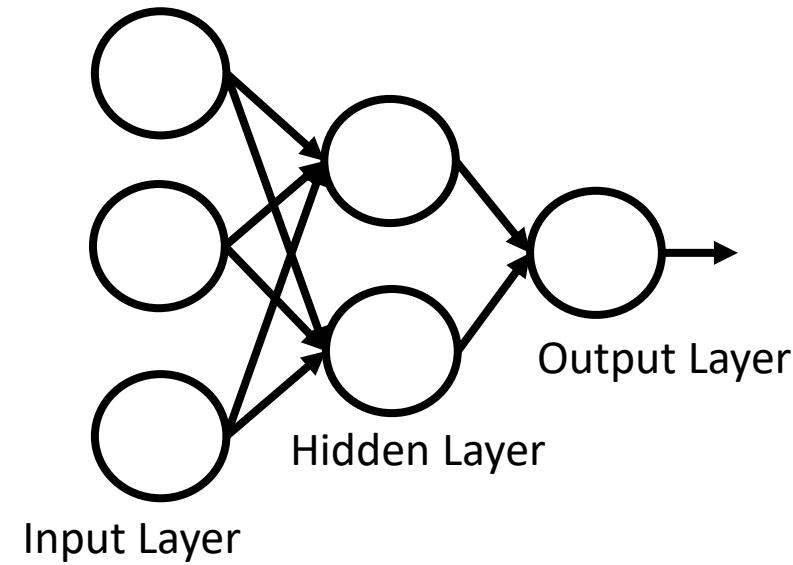
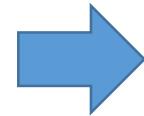
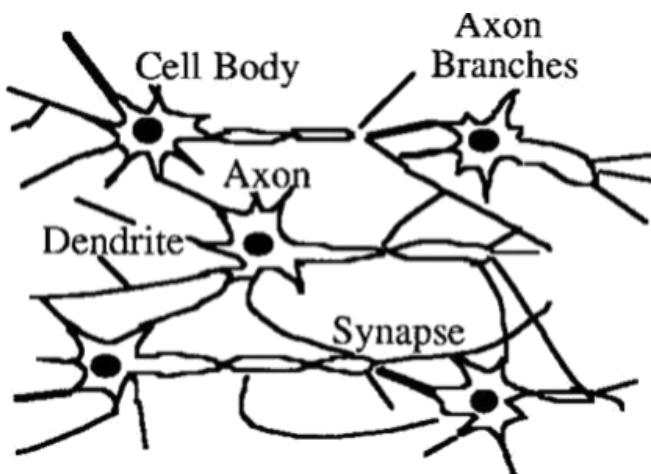
$$f(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases}$$



XOR Problem

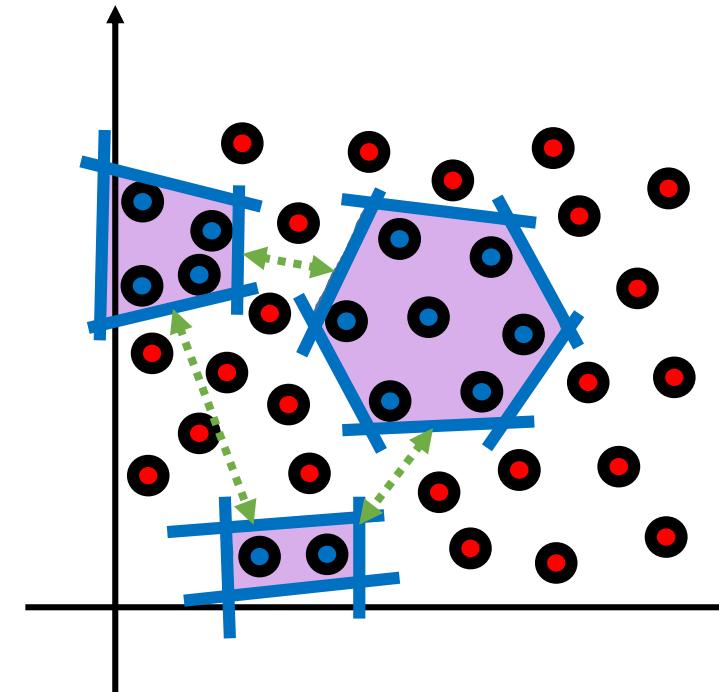
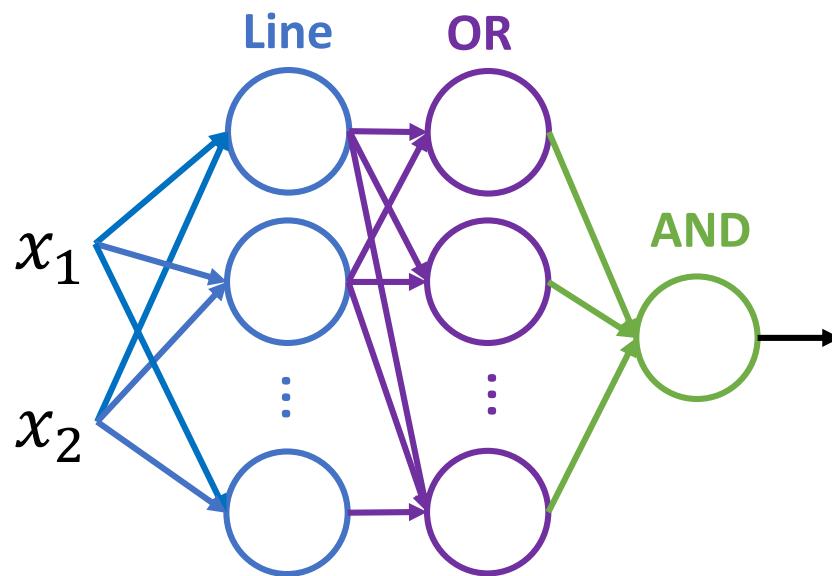


Connect the Neurons



Remark

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



Formulate the Multi-Layer NN

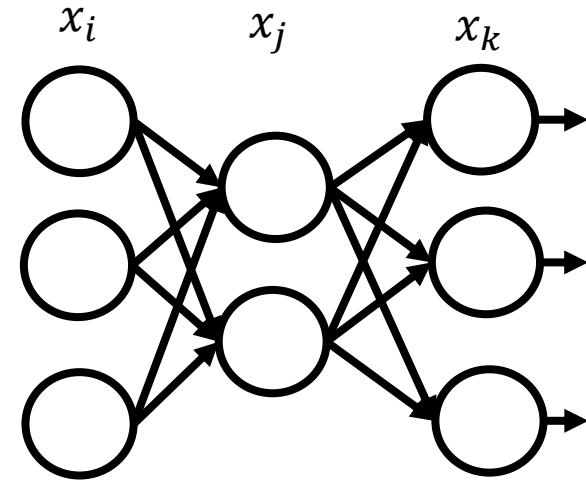
$$f\left(\sum_j w_{jk}^1 f\left(\sum_i w_{ij}^0 x_i + b_j^0\right) + b_k^1\right) = y_k$$



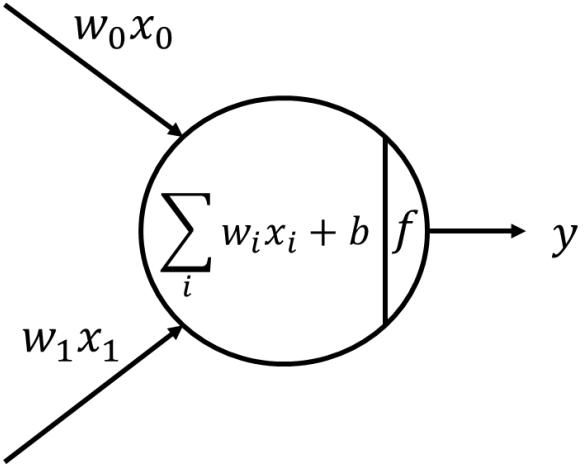
$$f\begin{pmatrix} w_{00}^1 & w_{10}^1 \\ w_{01}^1 & w_{11}^1 \\ w_{02}^1 & w_{12}^1 \end{pmatrix} f\left(\begin{bmatrix} w_{00}^0 & w_{10}^0 & w_{20}^0 \\ w_{01}^0 & w_{11}^0 & w_{21}^0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_0^0 \\ b_1^0 \\ b_2^0 \end{bmatrix}\right)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



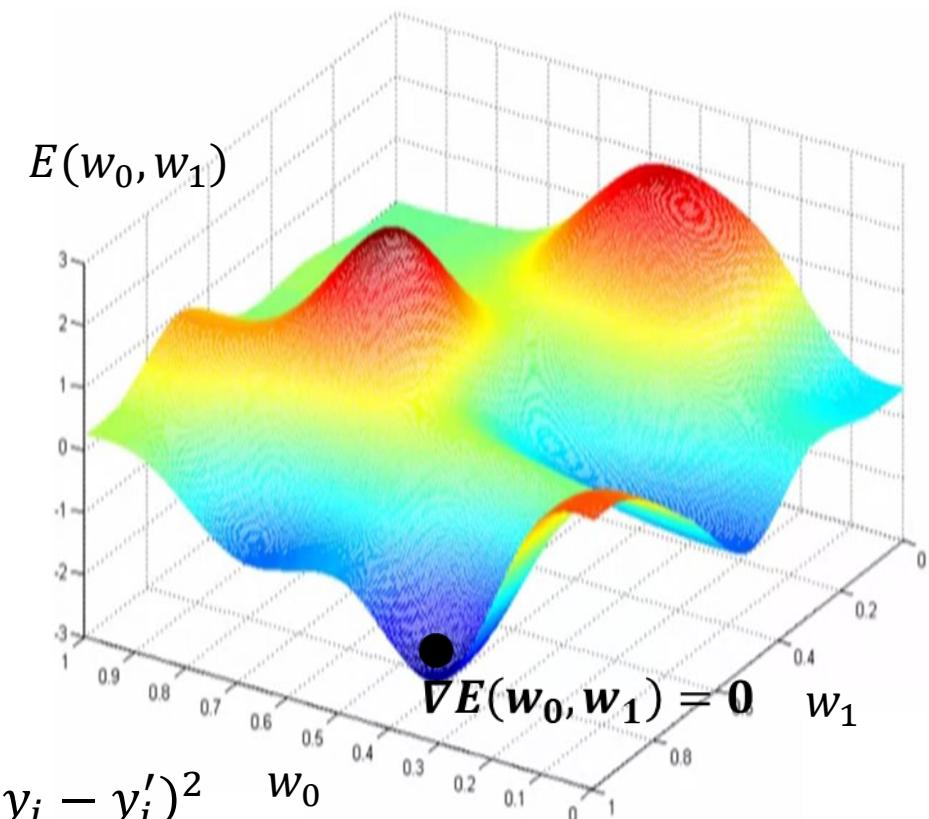
$$f(W^1 f(W^0 X + b^0) + b^1) = Y$$



Find the Optimal Parameter

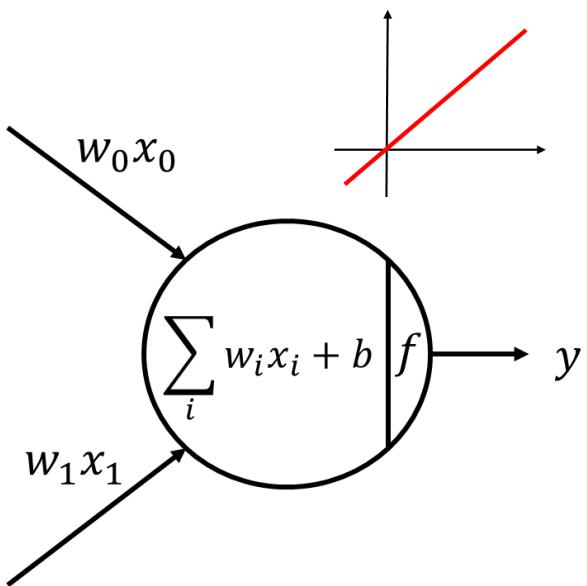


$$E(w_0, w_1) = \frac{1}{2} \sum_i (y_i - y'_i)^2$$



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

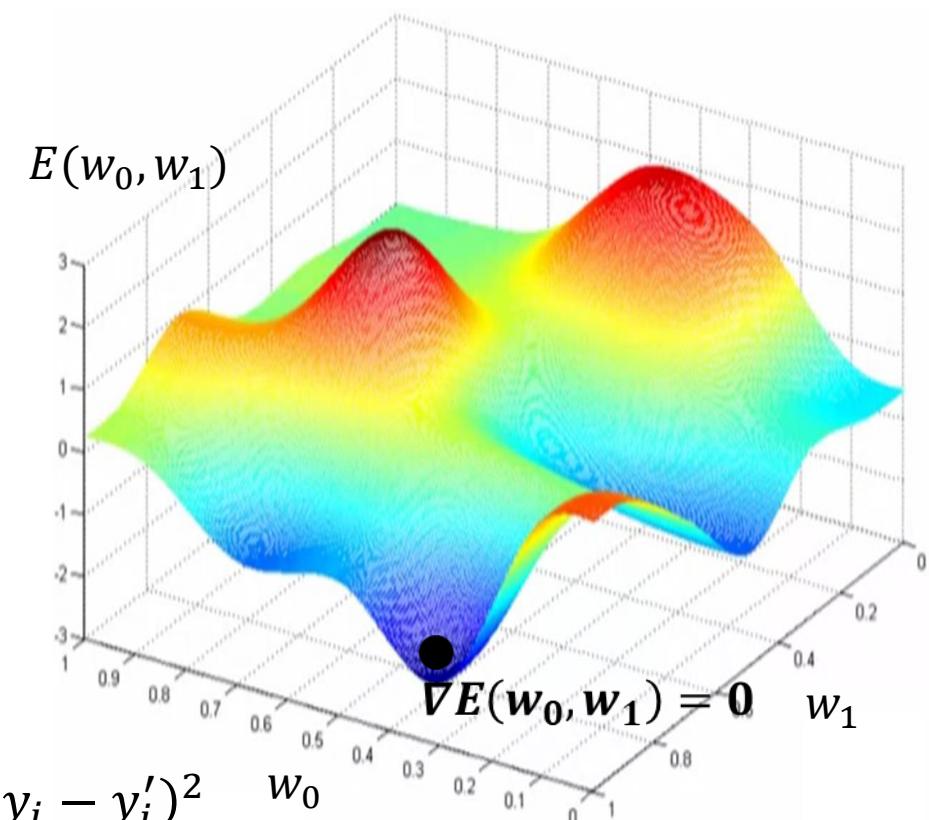
Find the Optimal Parameter



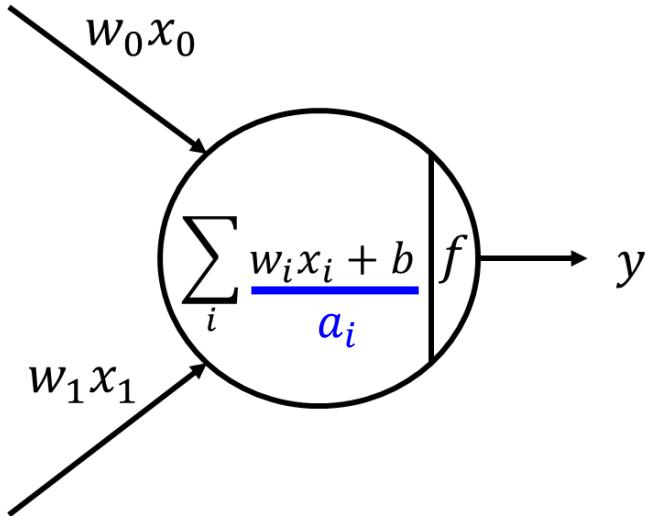
if $f(x) = x$,

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

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



Chain Rules & Back Propagation

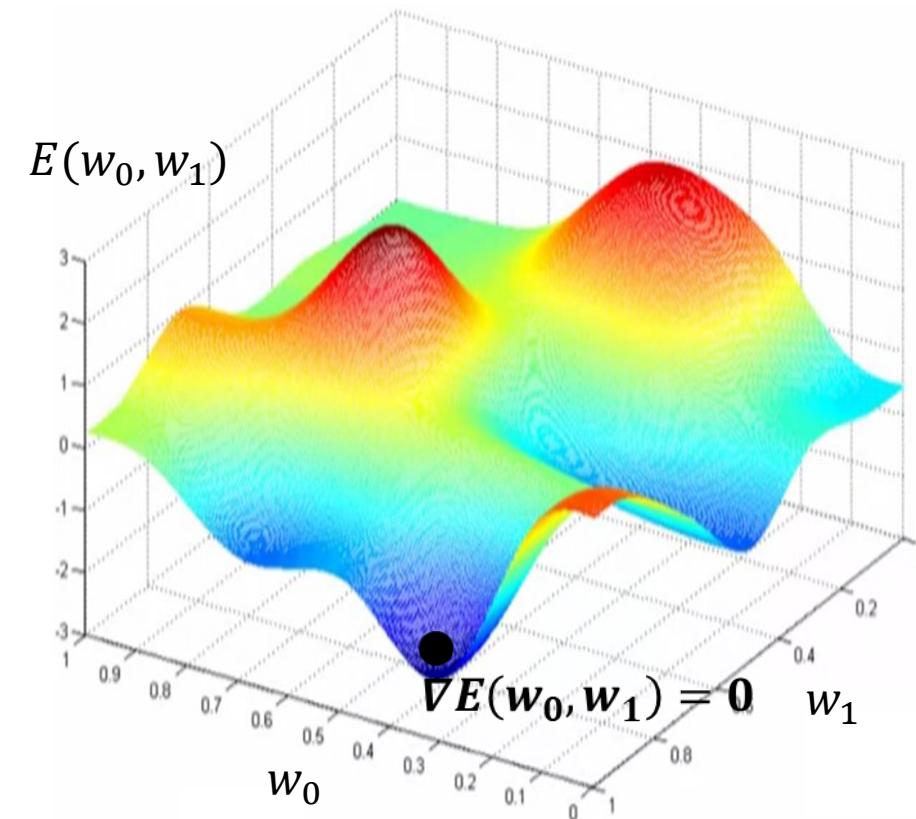


Chain Rule

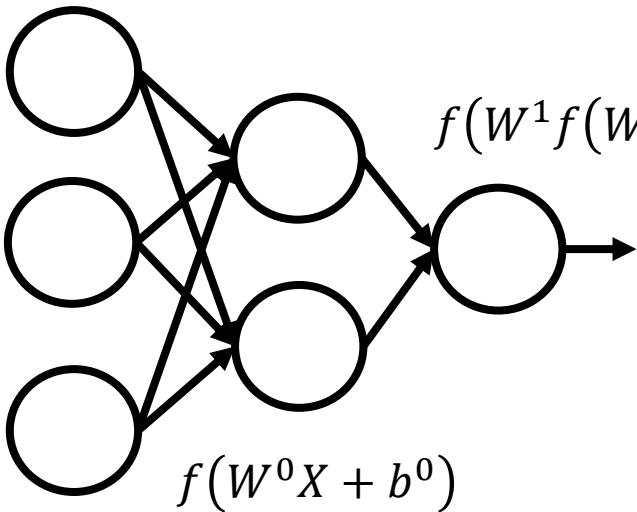
$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial f} \frac{\partial f}{\partial a_i} \frac{\partial a_i}{\partial w_i}$$

f is sigmoid :

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

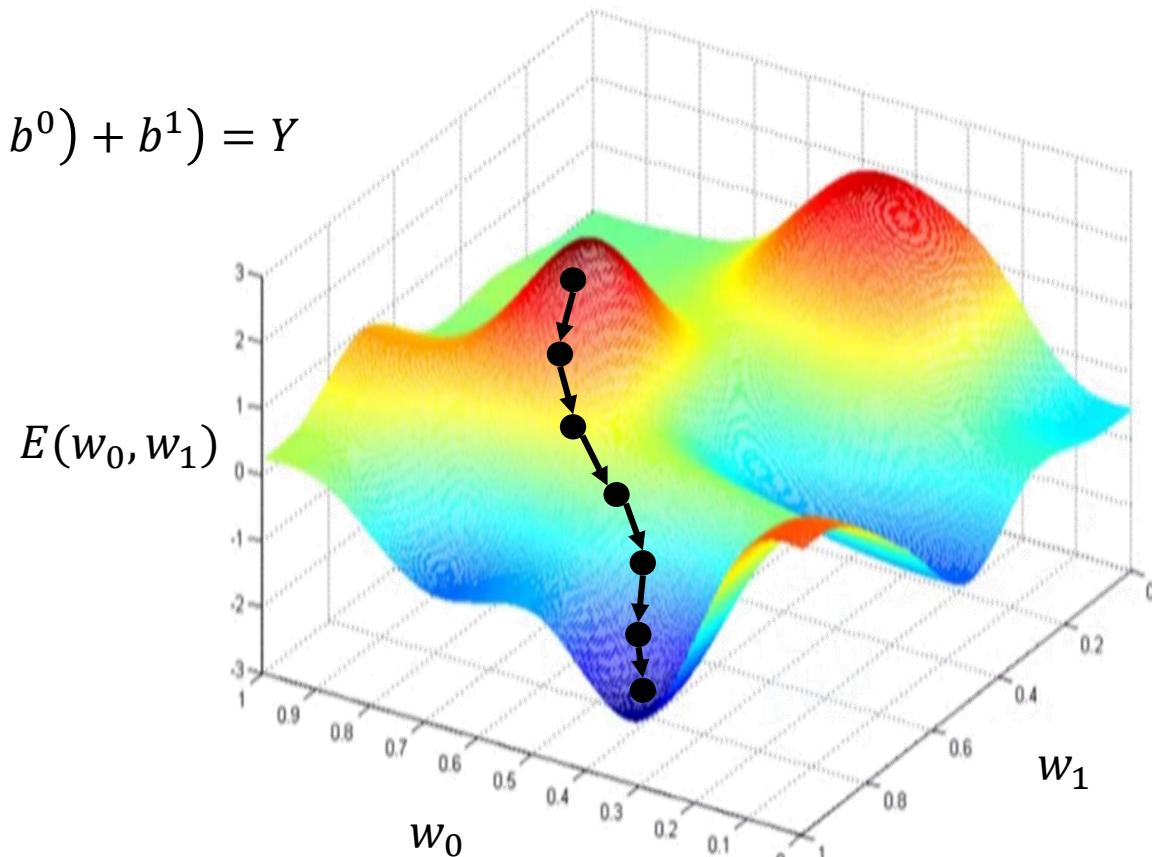


Chain Rules & Back Propagation



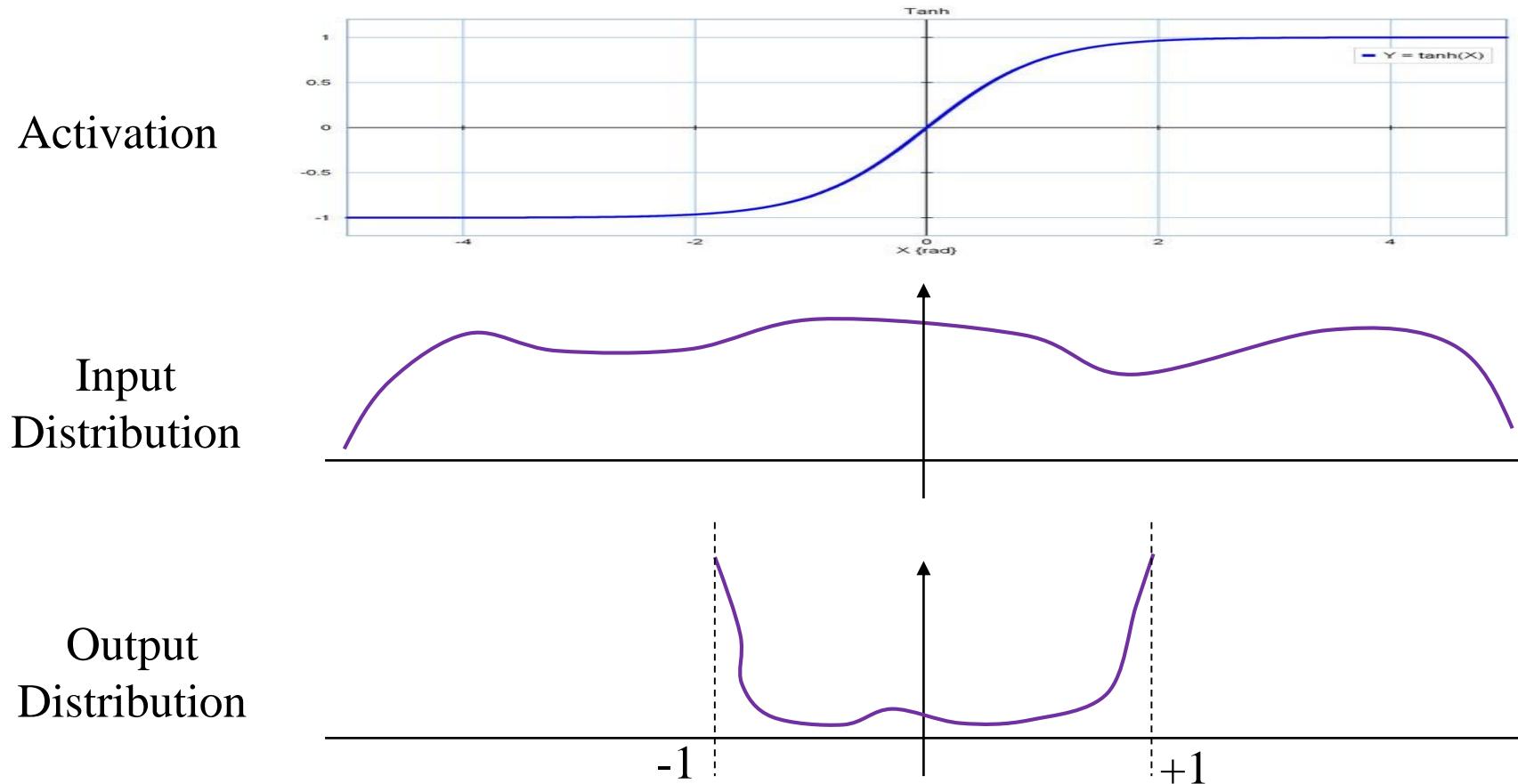
$$\frac{\partial E}{\partial w_i^1} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_i^1}$$

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



Normalization Problem

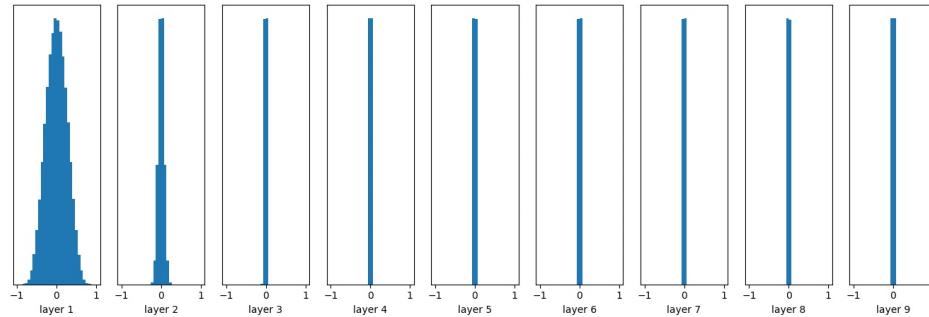
- The distribution of the input tensor can affect the performance of the training process dramatically.



Normalization Problem

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

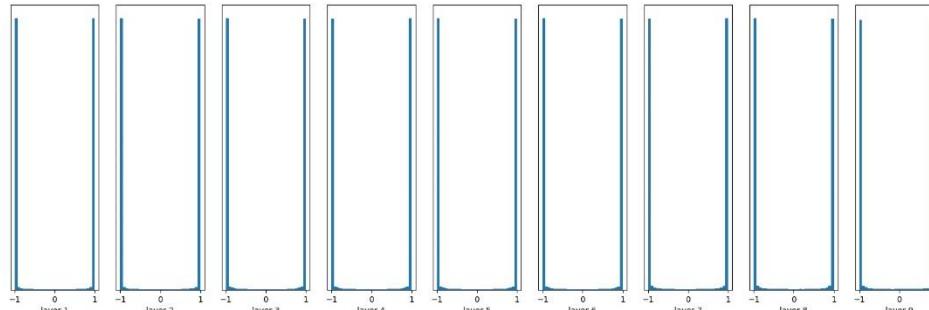
Small Variance



Tanh
Activation

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

Big Variance

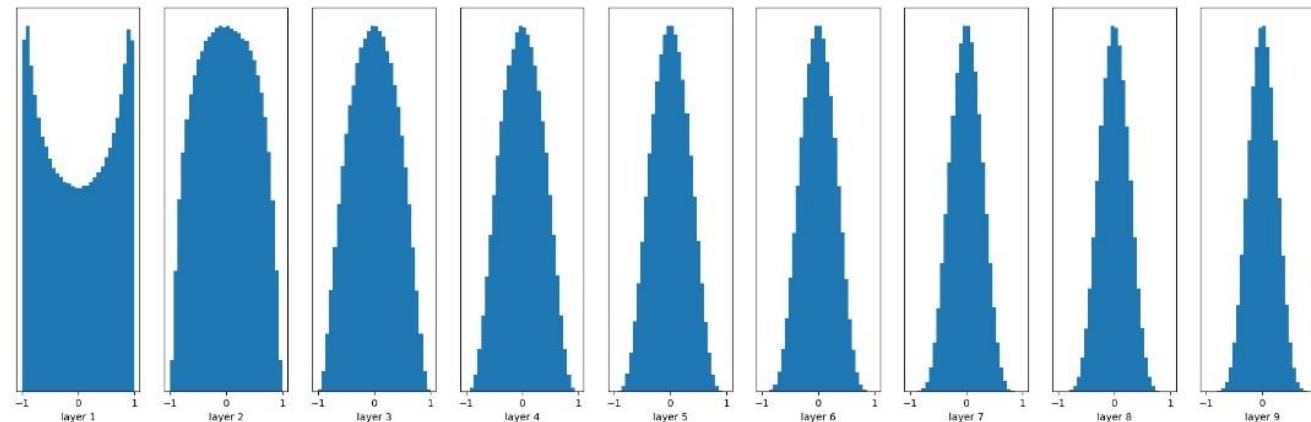


Tanh
Activation

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.

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



Tanh
Activation

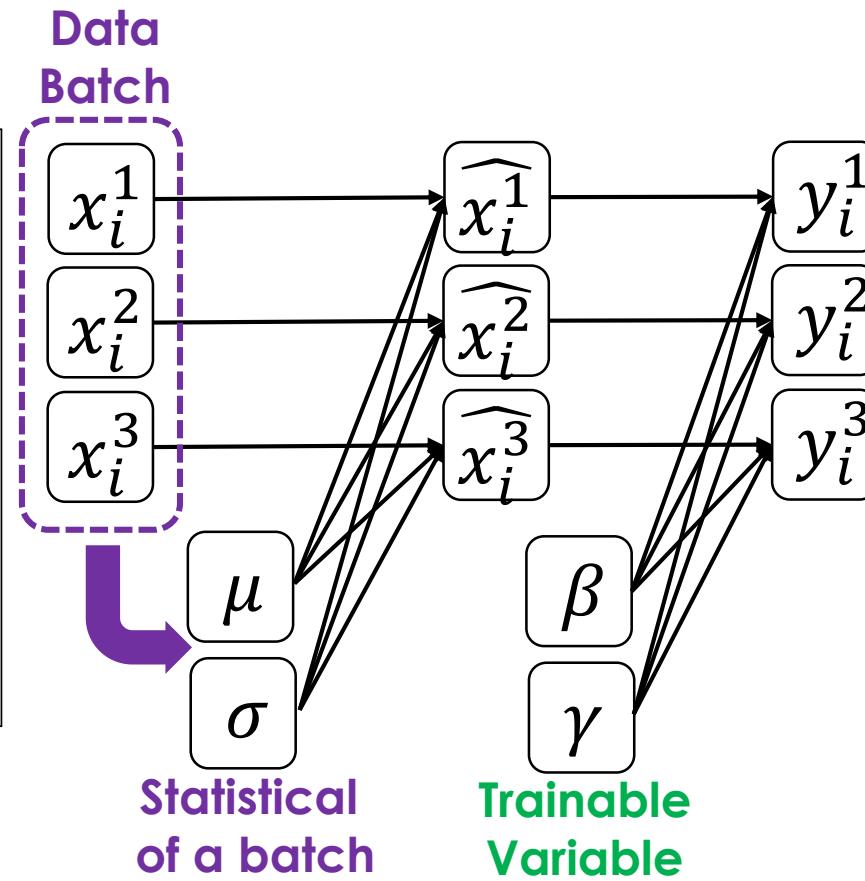
Batch Normalization

- 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: $\mathcal{B} = \{x_1 \dots m\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

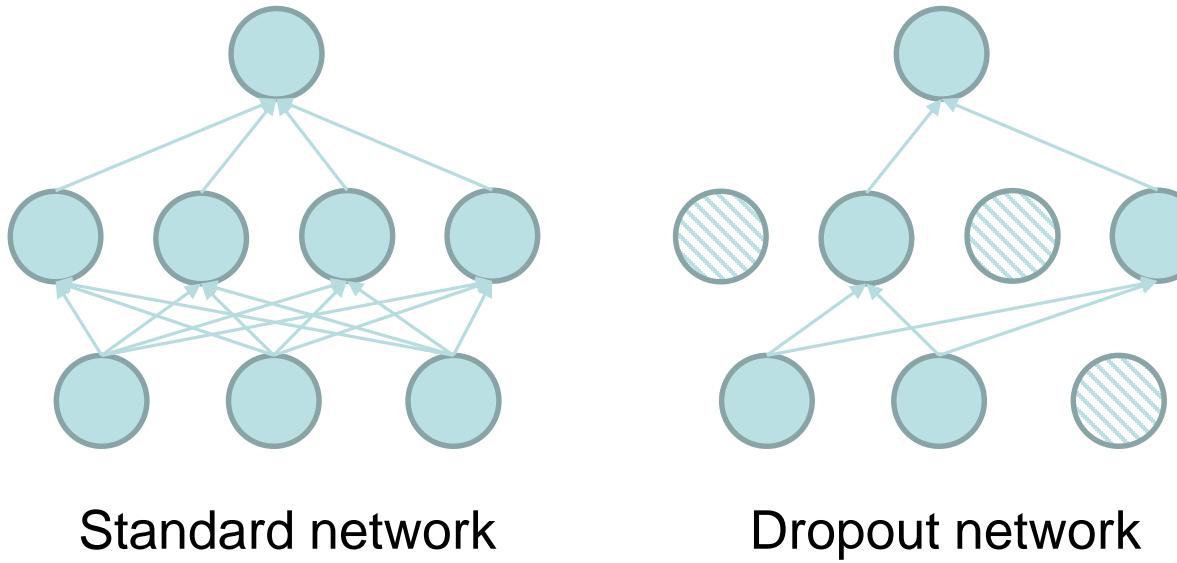
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$ // mini-batch mean
 $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$ // mini-batch variance
 $\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$ // normalize
 $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$ // scale and shift

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

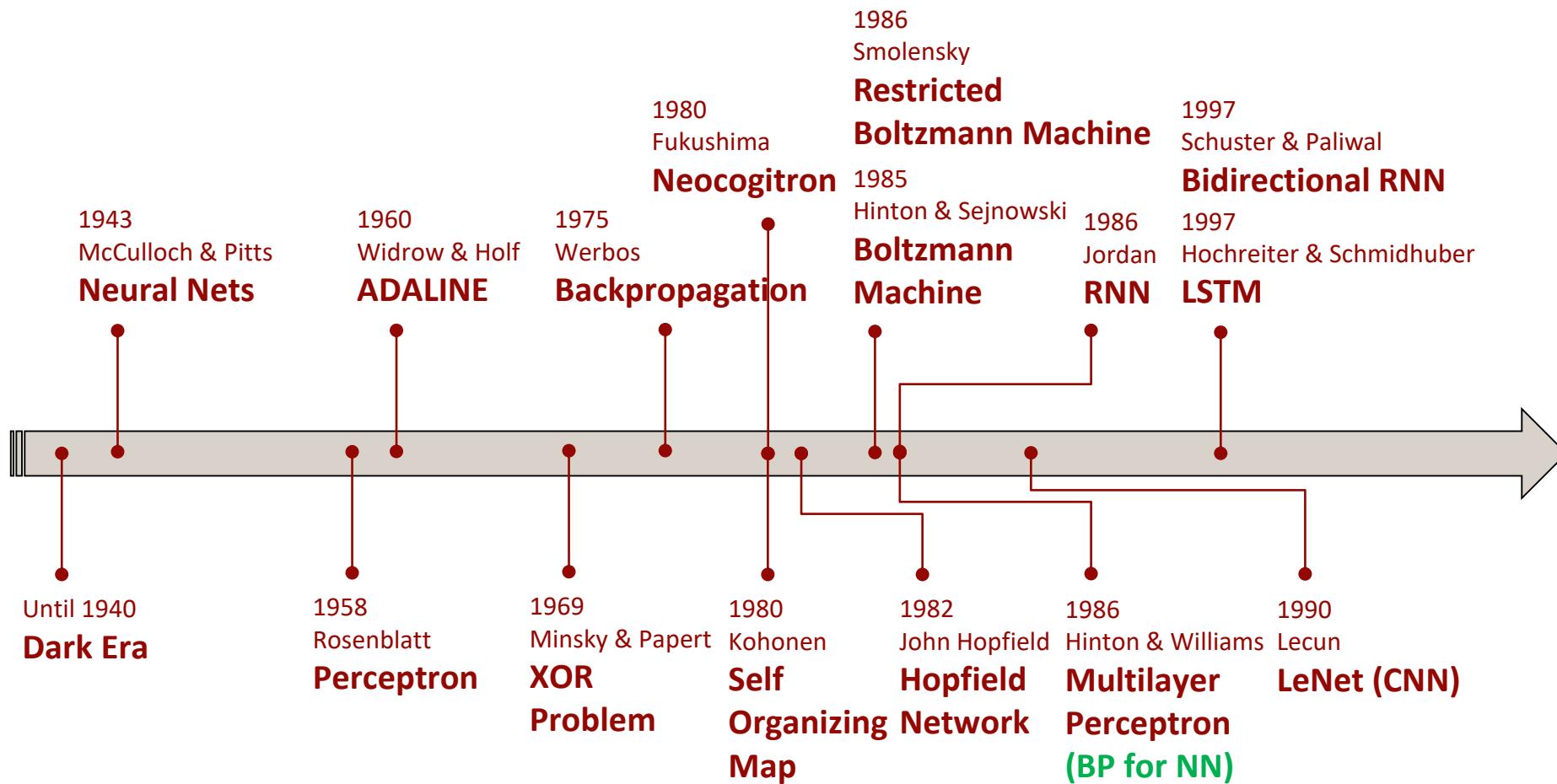


Dropout

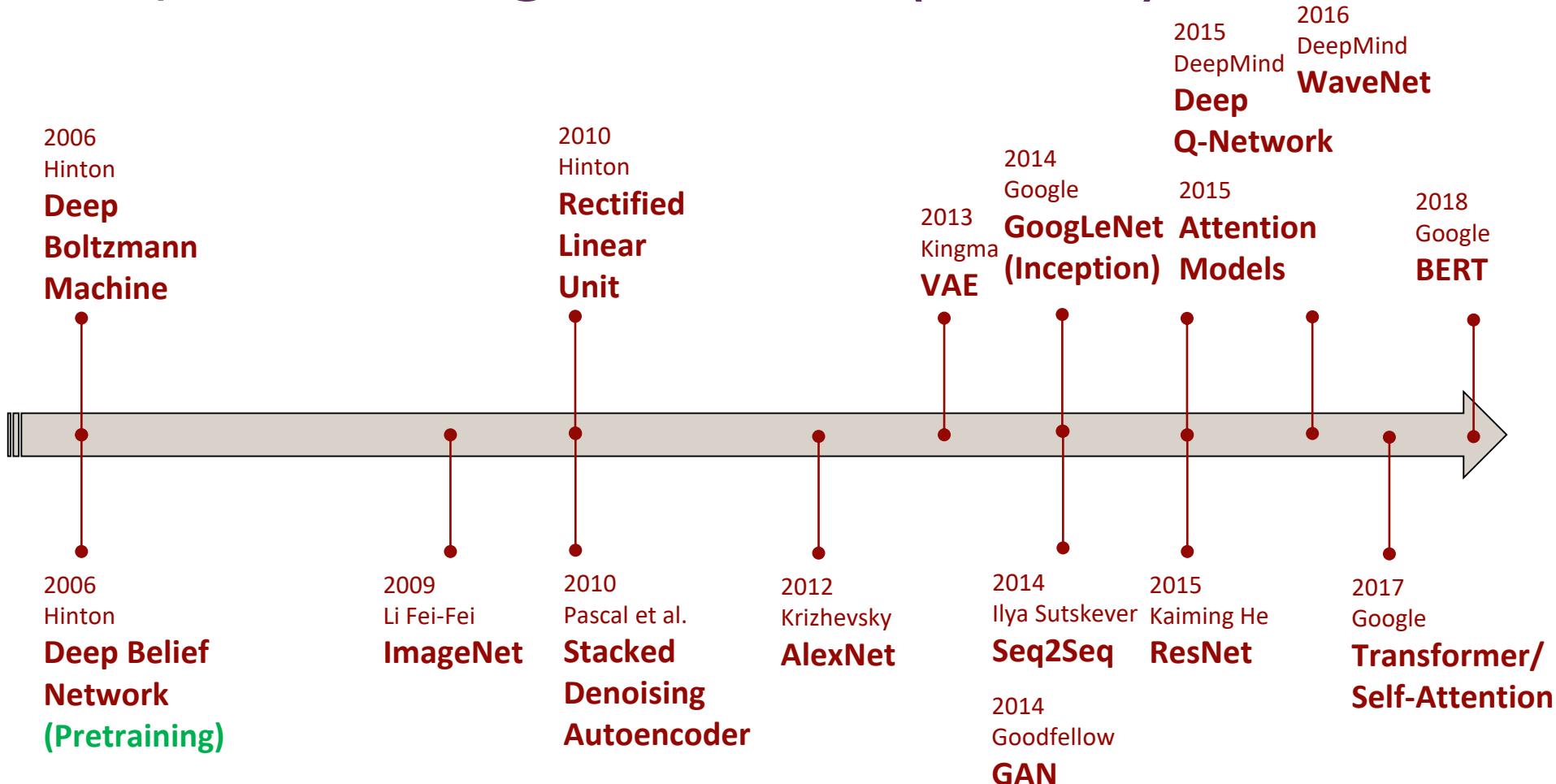
- In training phase, drop some visible and hidden units to make learned features be more general
 - Avoid that some units may bias learning
- At test phase, use weight to mimic the dropout in training
 - By averaging the outputs of several dropout networks, it would approximate the output of the traditional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



Deep Learning Timeline

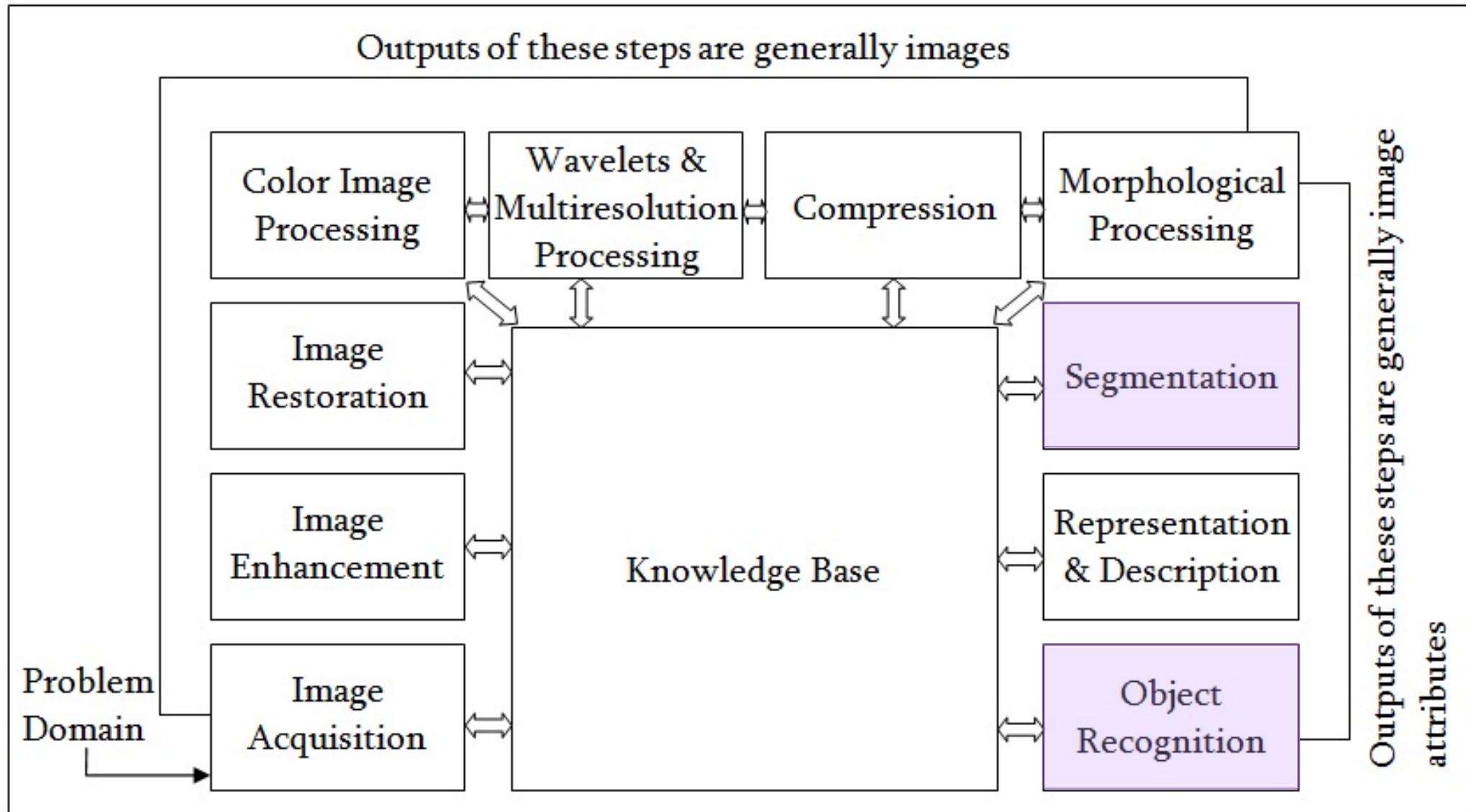


Deep Learning Timeline (Cont.)



Application to Images

Fundamental Steps in DIP



Computer Vision Problems

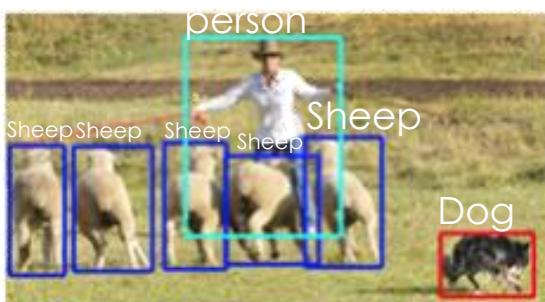
➤ Object Classification

- 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



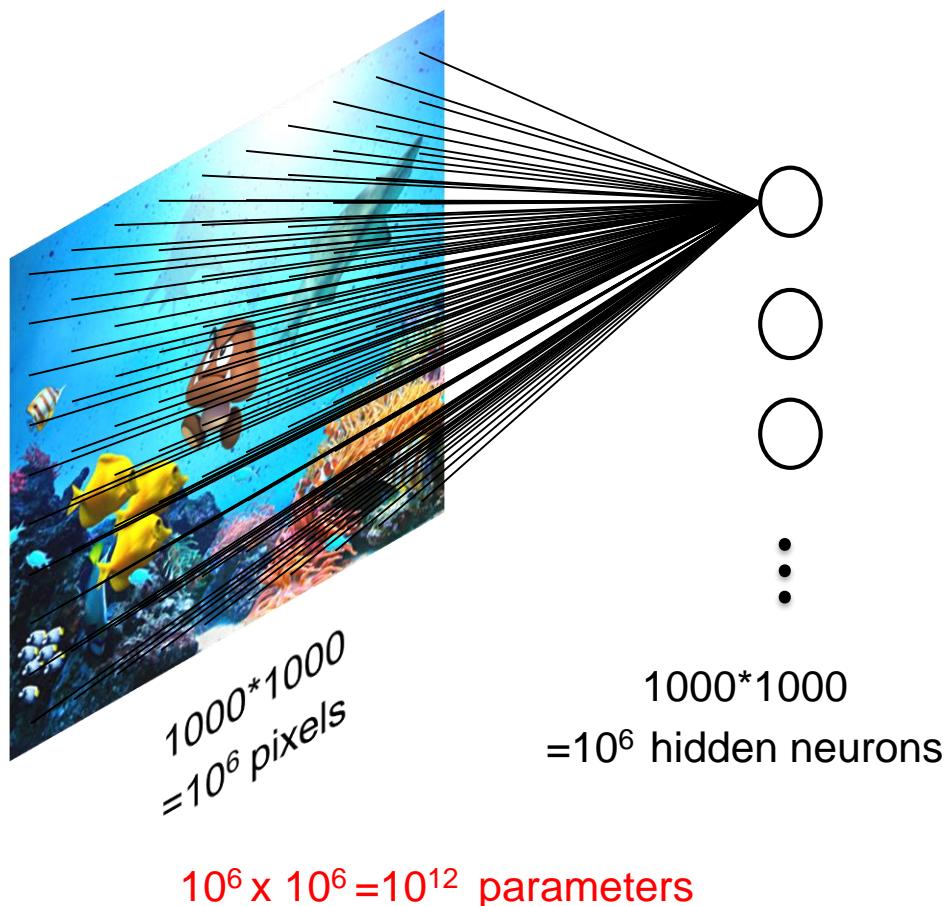
➤ Instance Segmentation

- Segment each object instance pixel-by-pixel

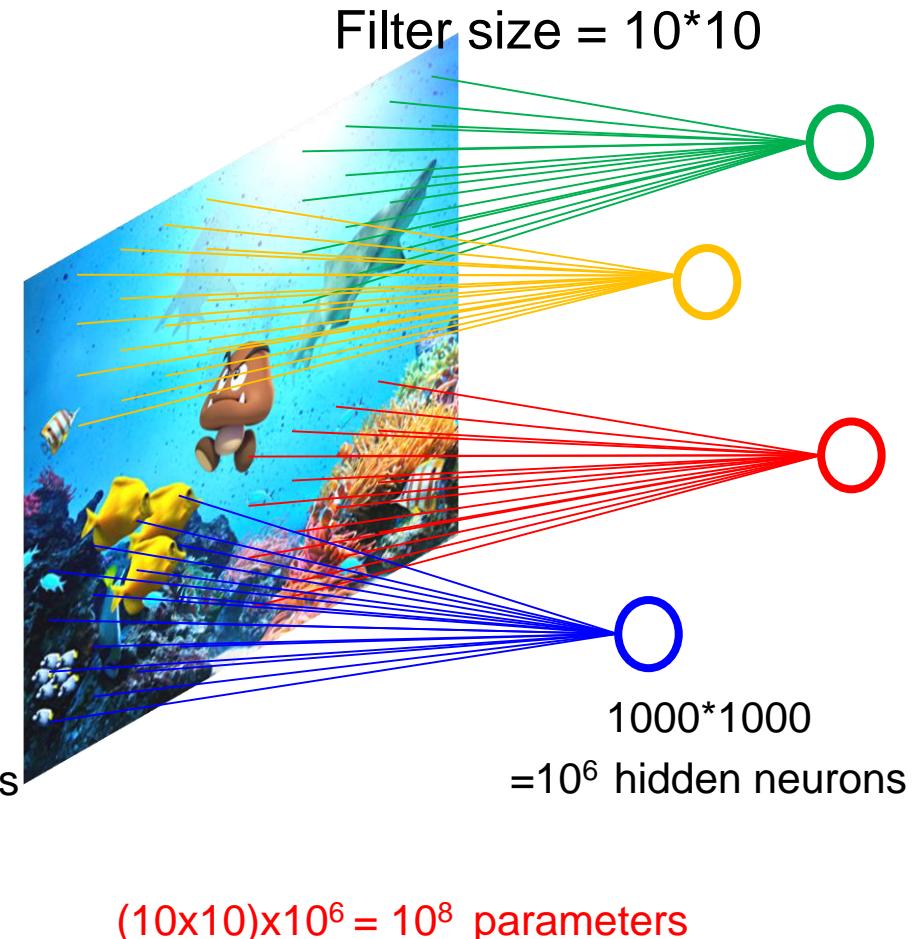


Convolutional Neural Network (CNN)

Fully Connected Neural Net

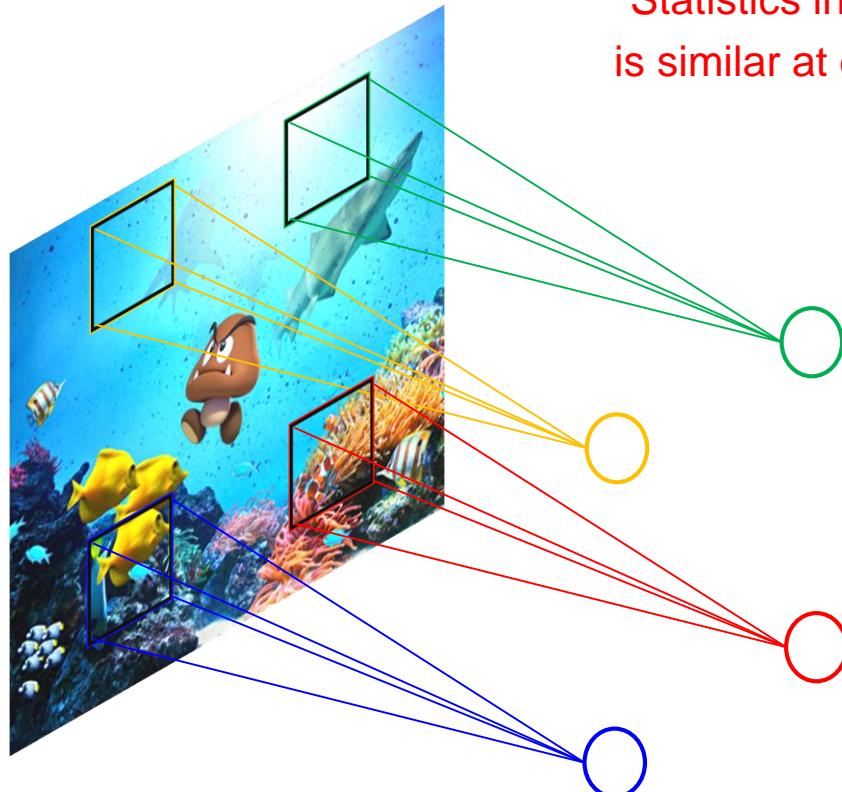


Locally Connected Neural Net



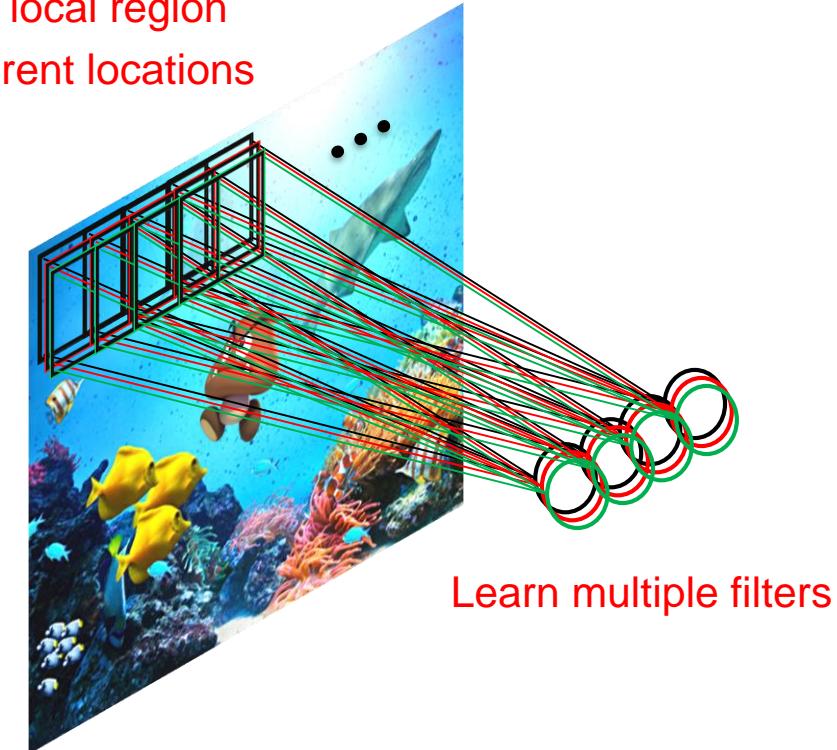
Convolutional Neural Network (CNN)

Locally Connected Neural Net



Statistics in the local region
is similar at different locations

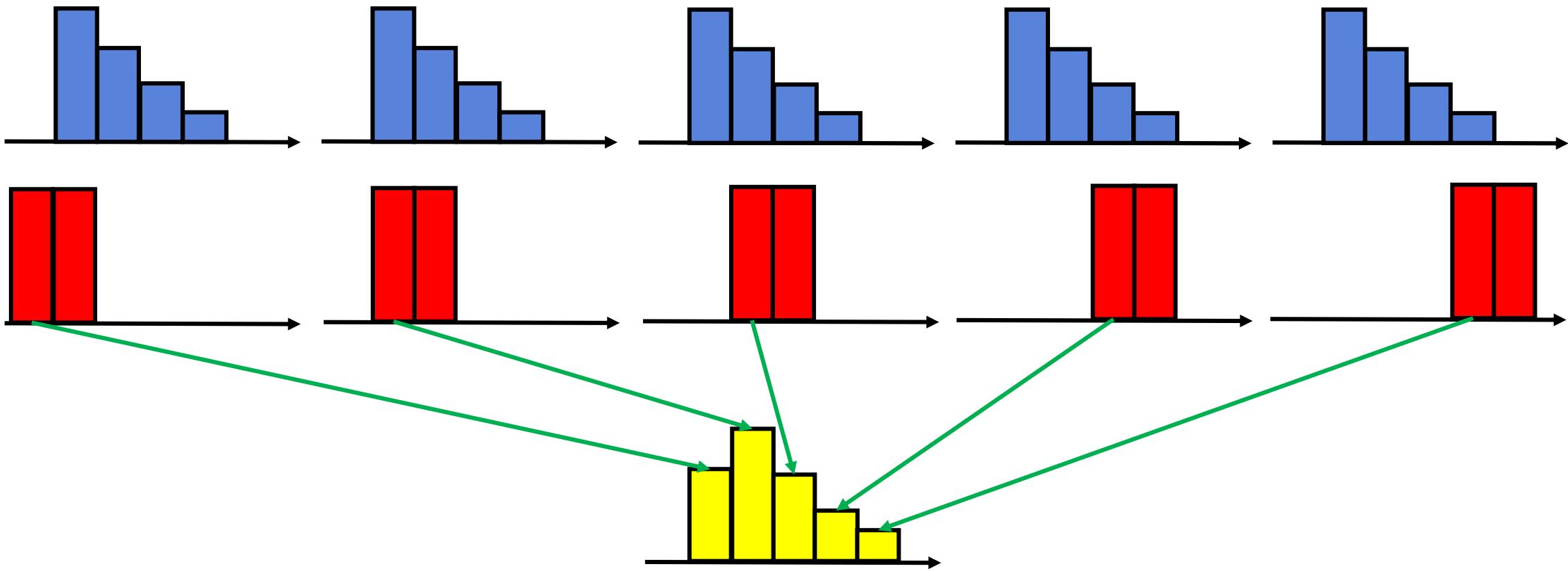
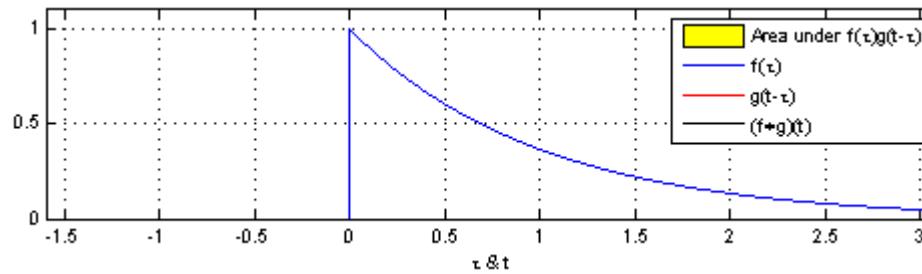
Convolutional Net



Learn multiple filters

Continuous-Time Convolution

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



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

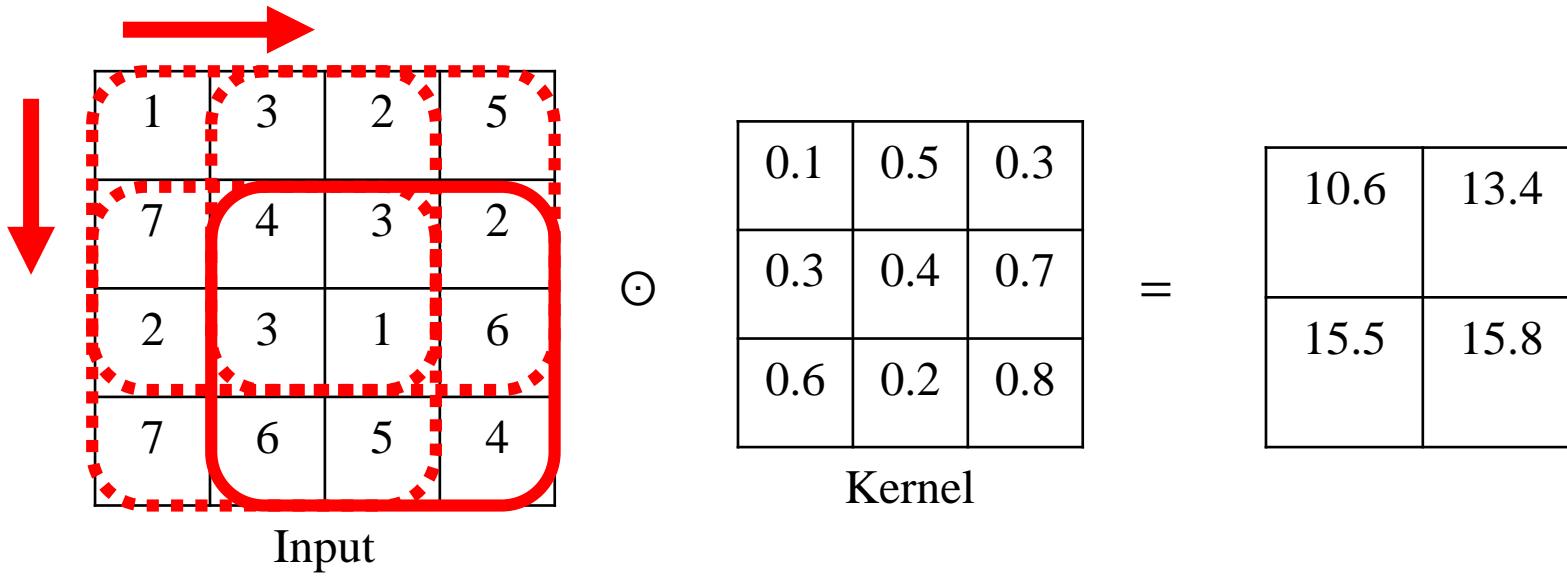
1	3	2
7	4	3
2	3	1

○

0.1	0.5	0.3
0.3	0.4	0.7
0.6	0.2	0.8

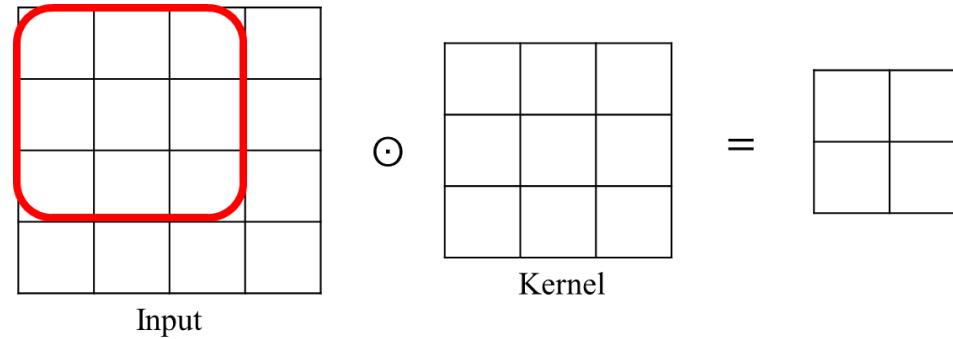
$$\begin{aligned} & 1 \times 0.1 \\ & + 3 \times 0.5 \\ & + 2 \times 0.3 \\ & + 7 \times 0.3 \\ & + 4 \times 0.4 \\ & + 3 \times 0.7 \\ & + 2 \times 0.6 \\ & + 3 \times 0.2 \\ & + 1 \times 0.8 \end{aligned} = 10.6$$

Convolutional Operation

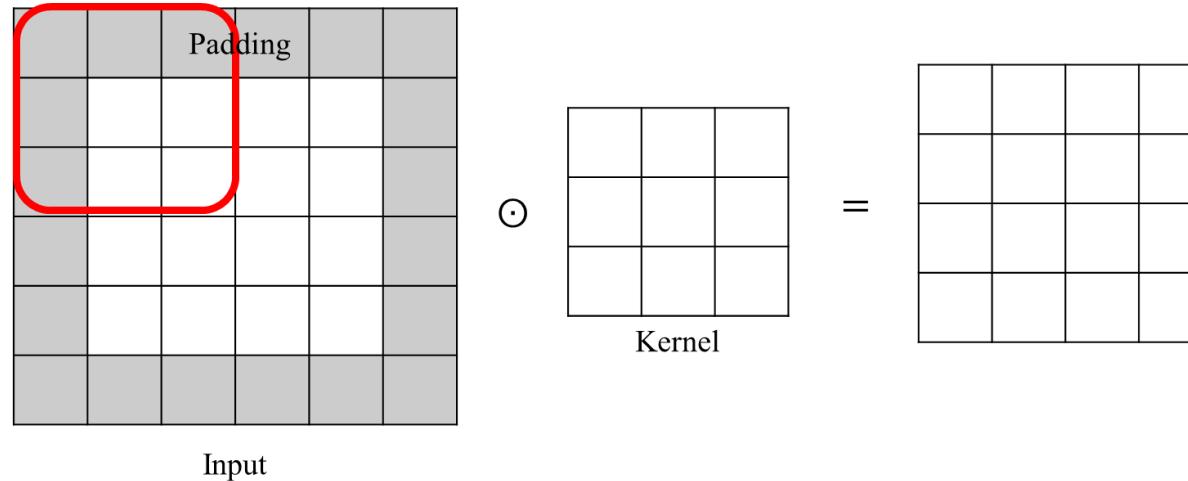


Convolutional Operation

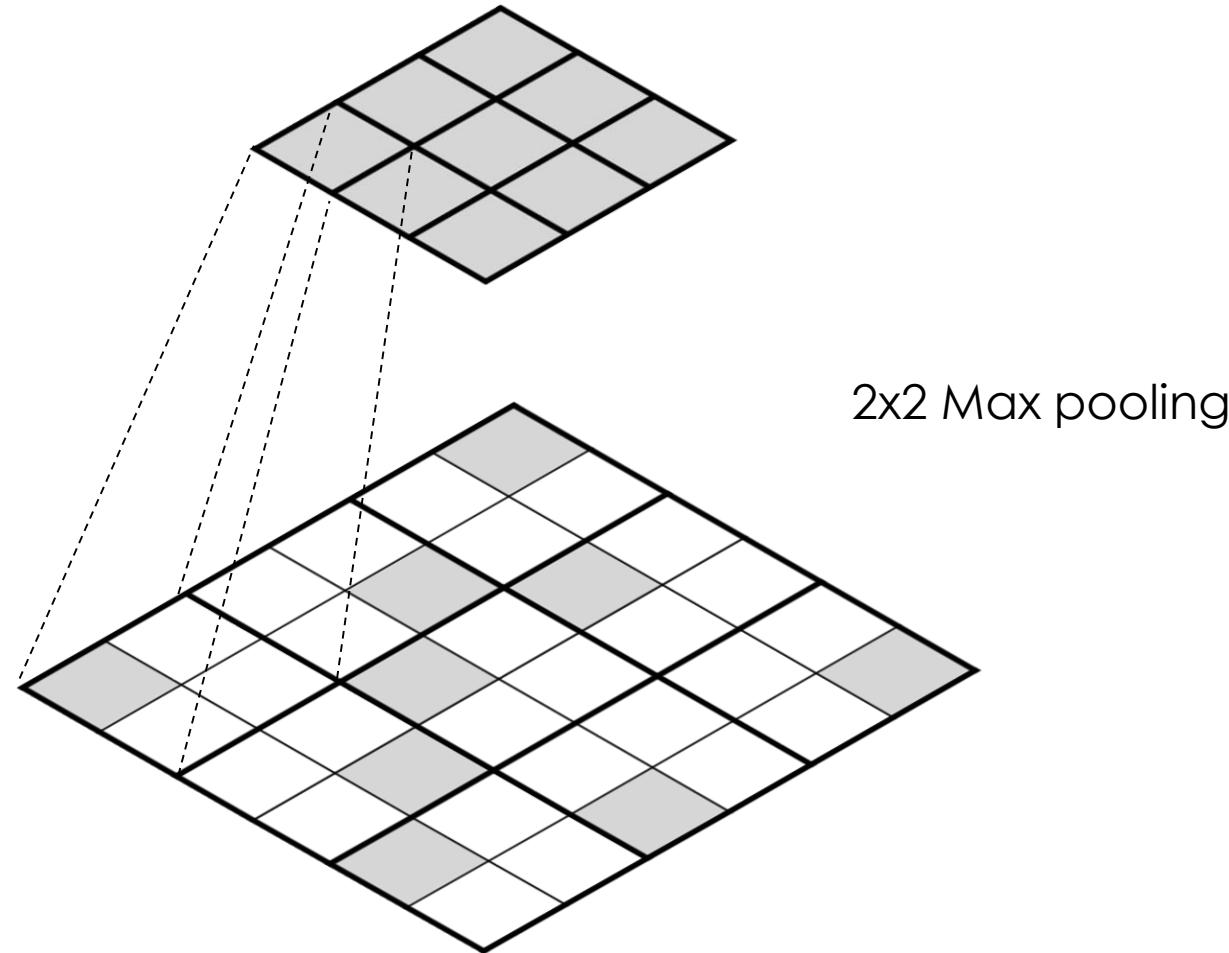
- Standard Convolution (In TensorFlow, padding='VALID')



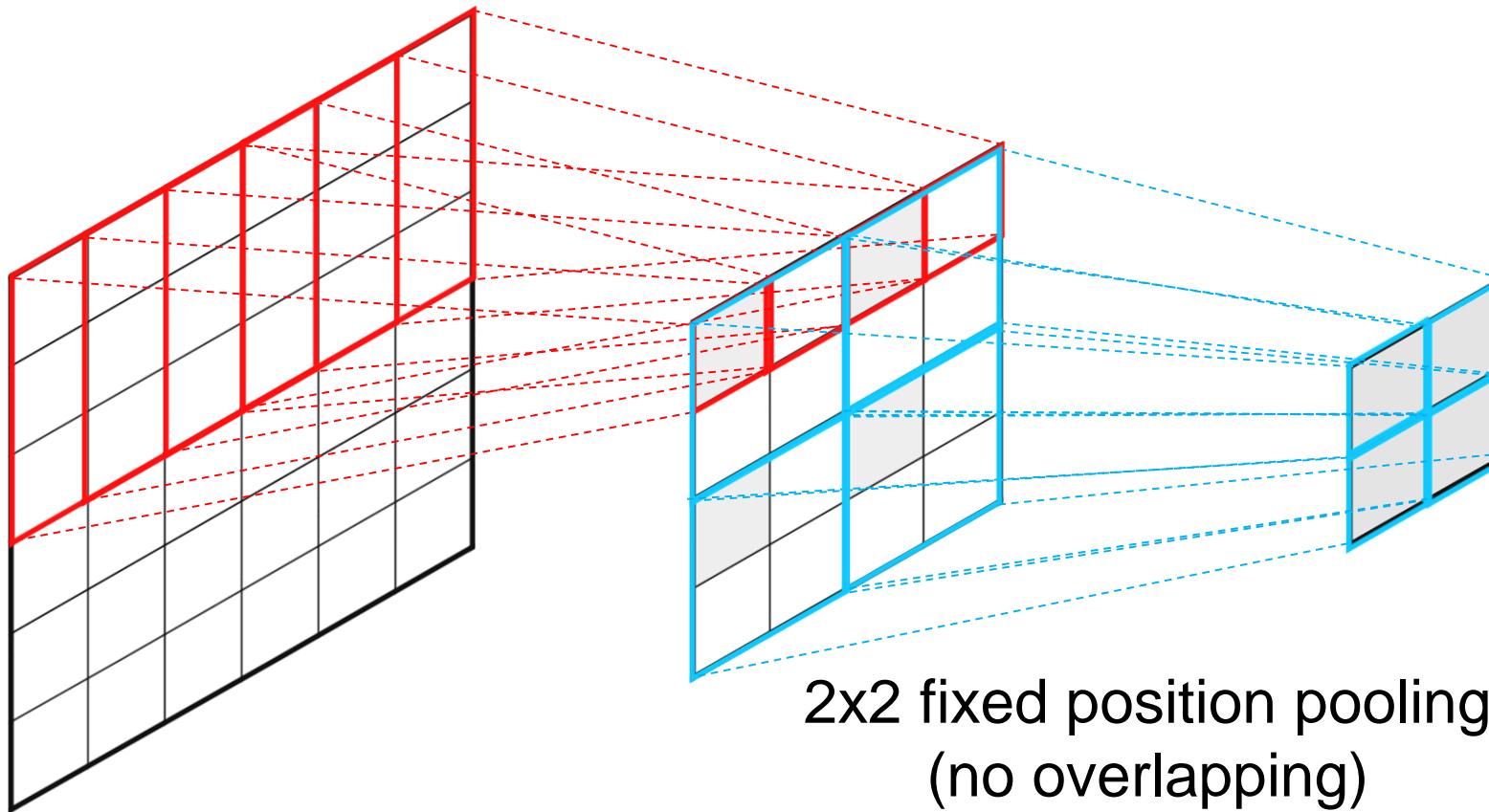
- Padding (In TensorFlow, padding='SAME')



Pooling

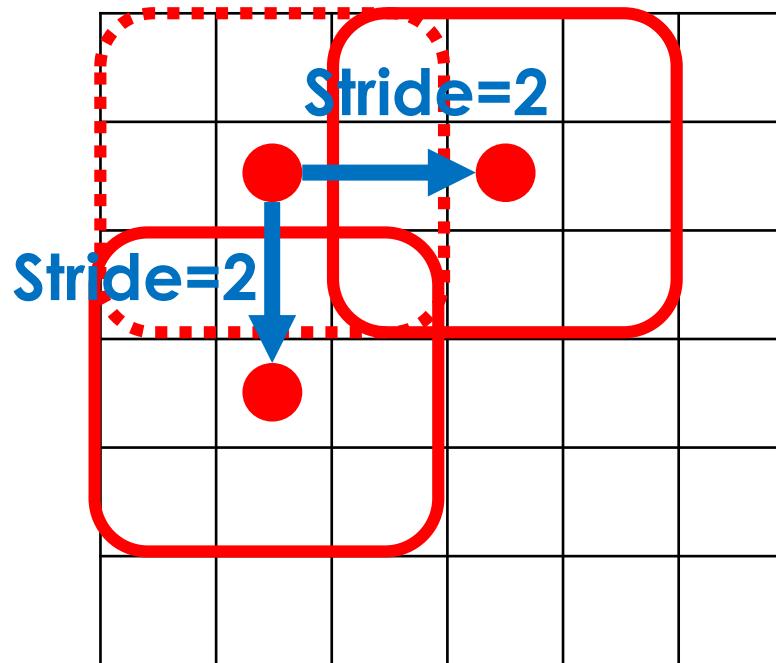


Convolution + Pooling



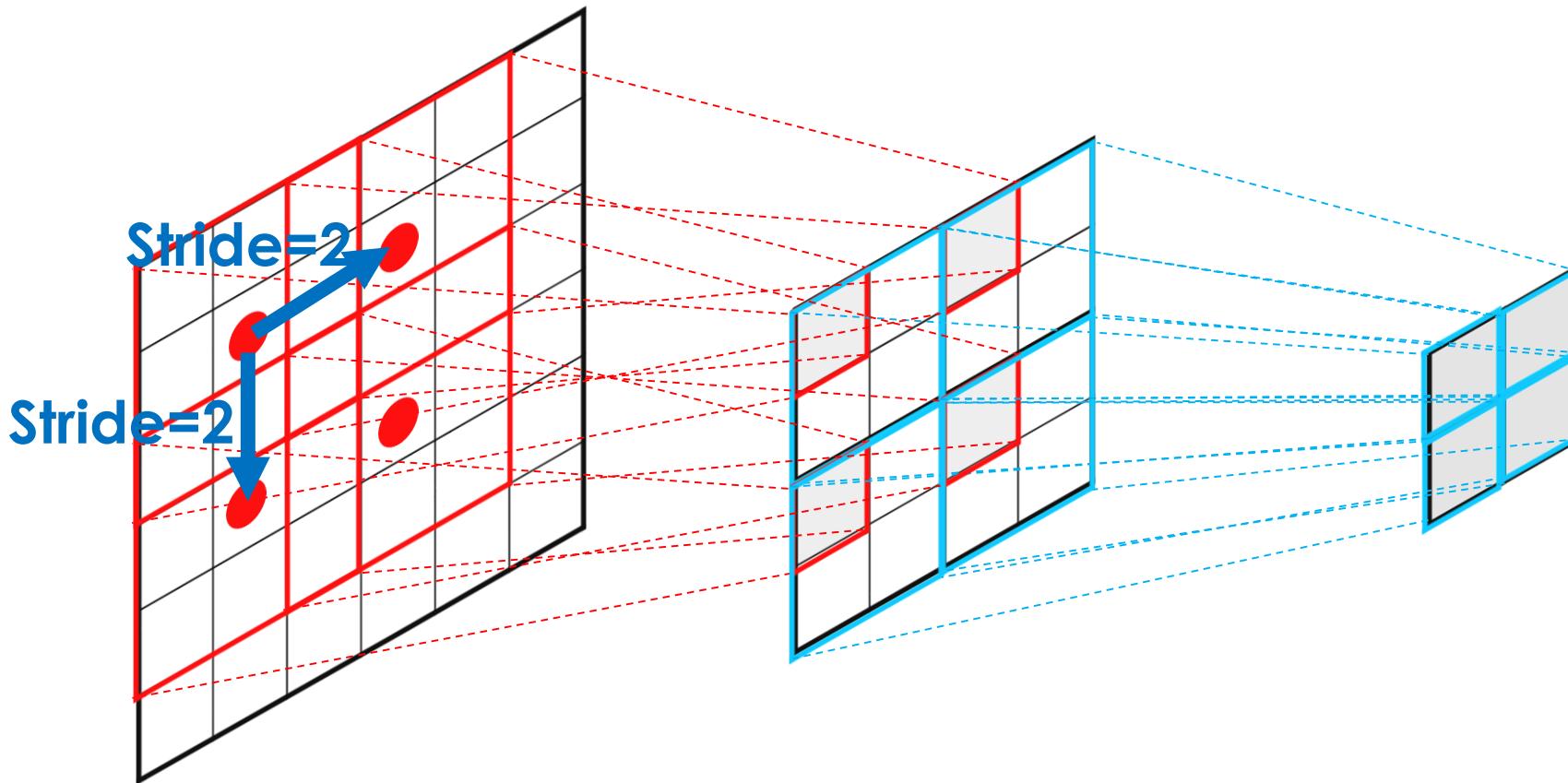
Convolution with stride=1

Stride Convolution



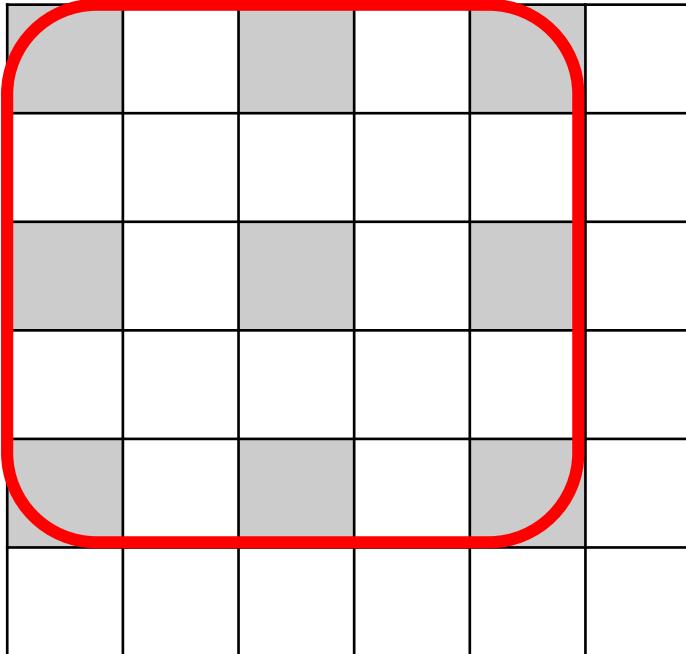
Can replace the pooling operation
(Usually be used in generative model)

Convolution+ Pooling

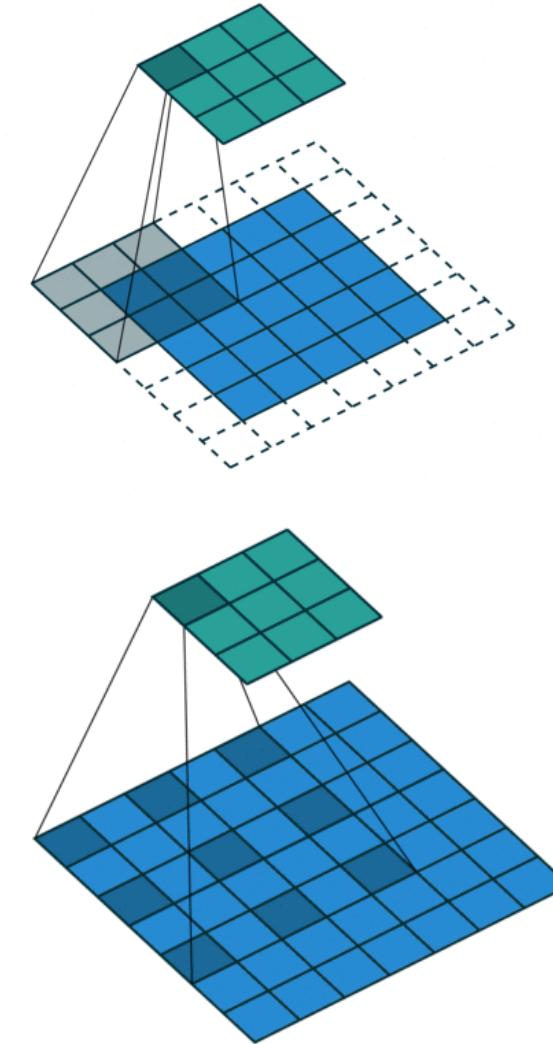


Equal to Convolution with stride=1 + 2x2 fixed position pooling
(no overlapping)

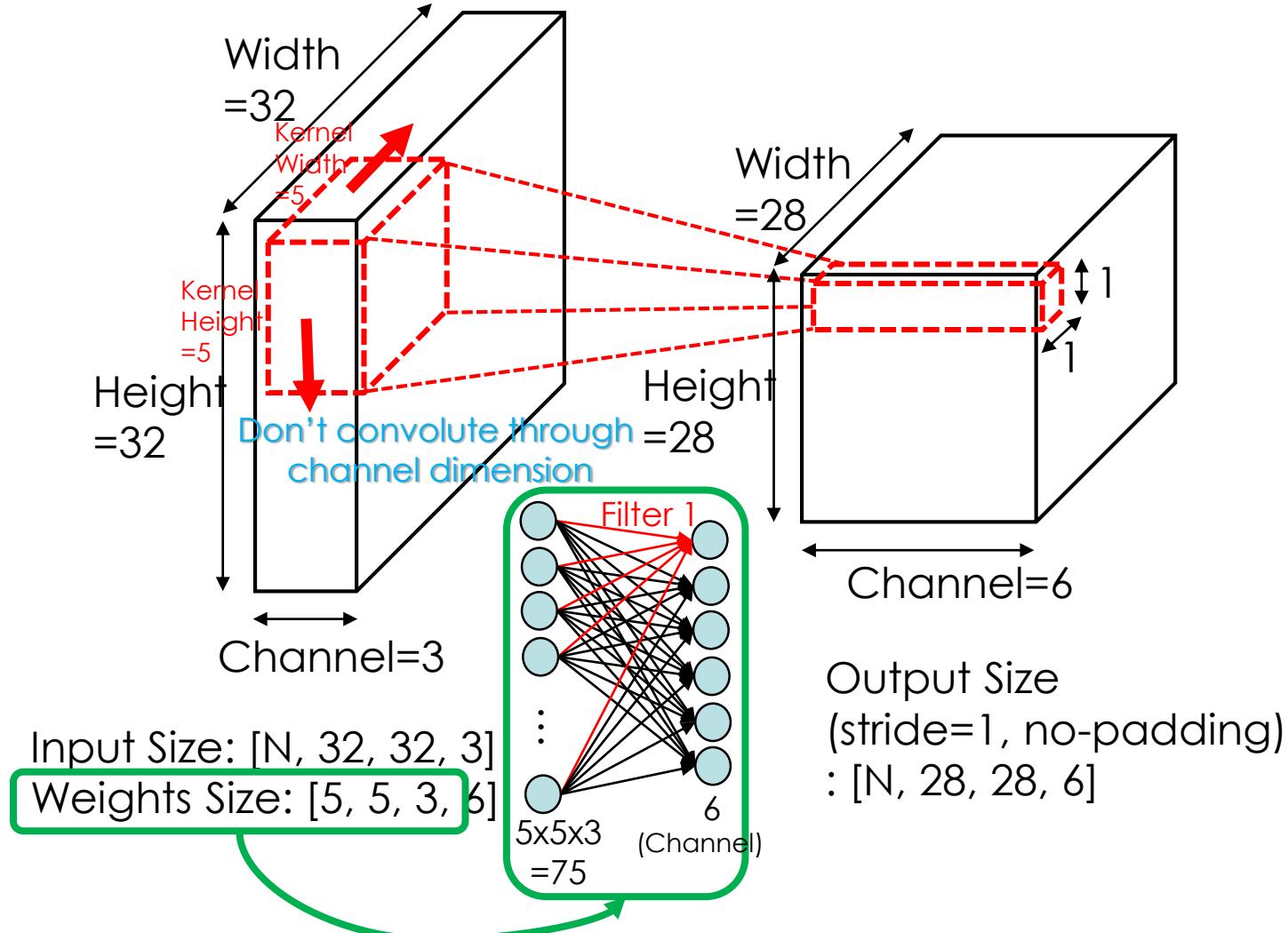
Dilated Convolution



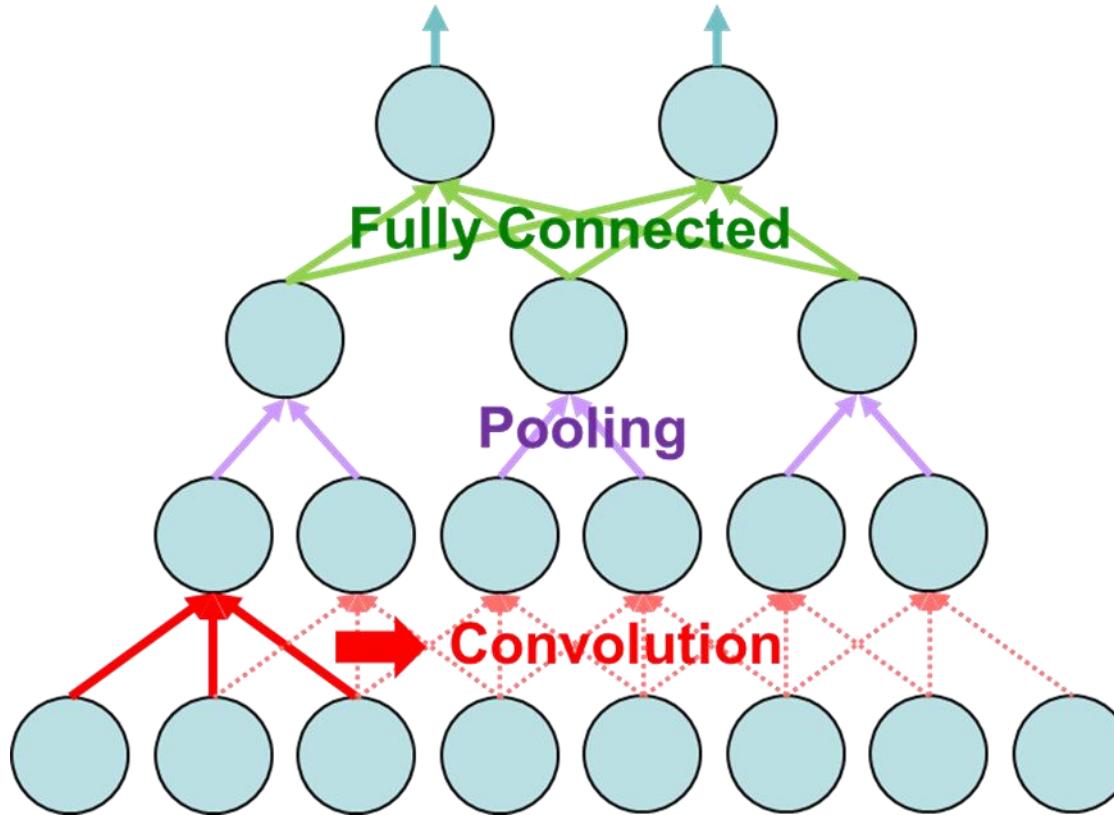
Increase the Receptive Field
(3x3 Kernel with dilation rate 2)



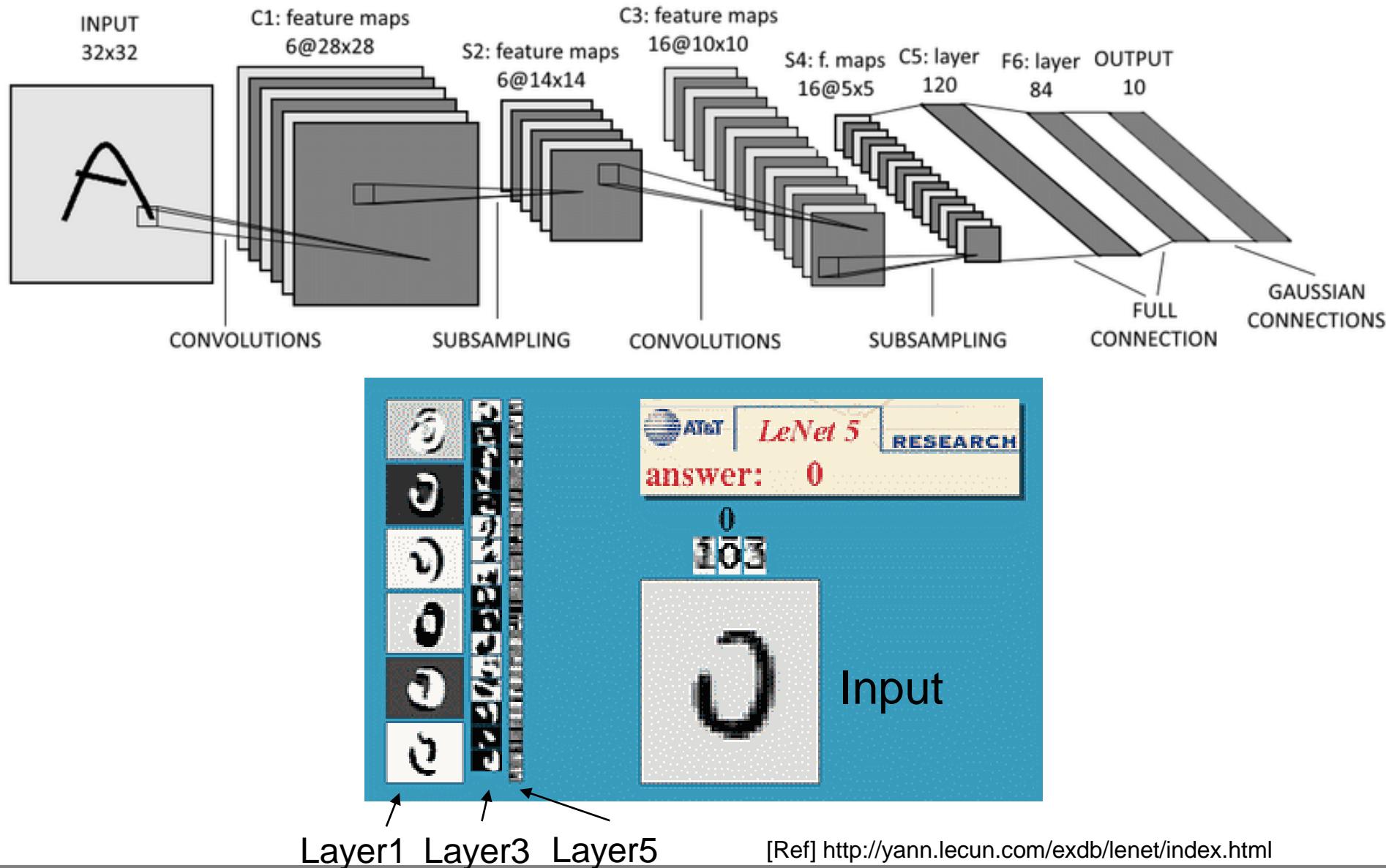
Take a Deep Look at Convolutional Layer



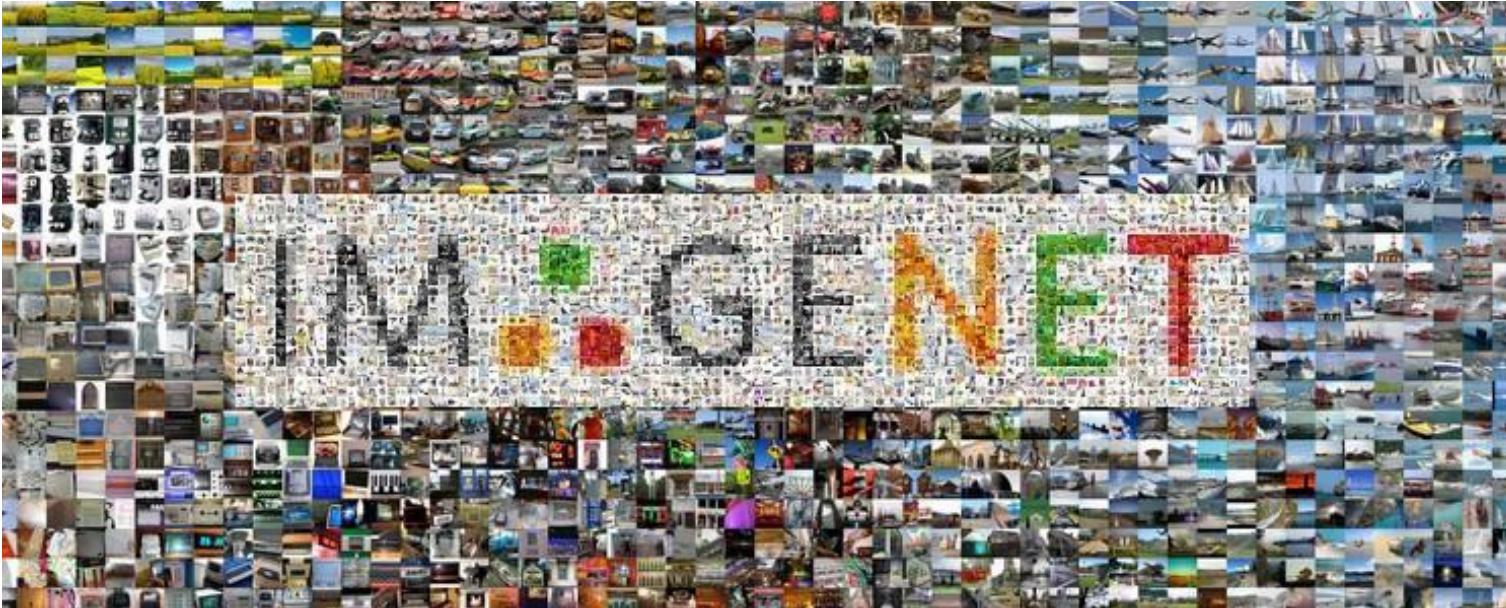
Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN): LeNet-5



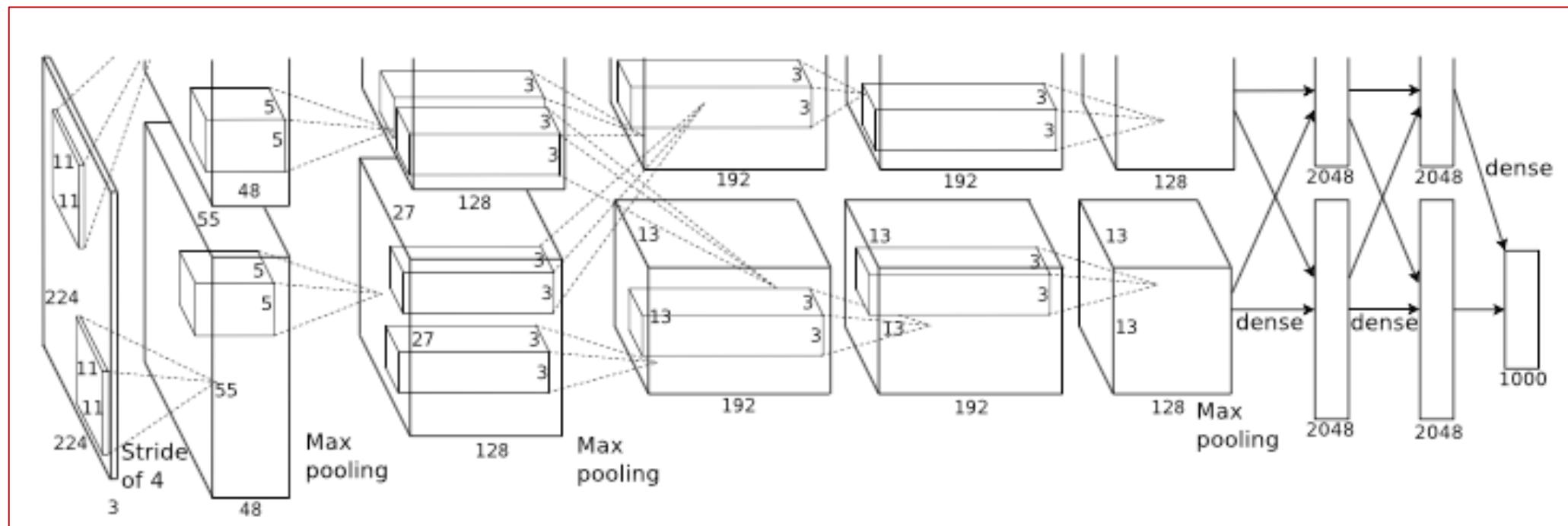
ImageNet Dataset



- 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

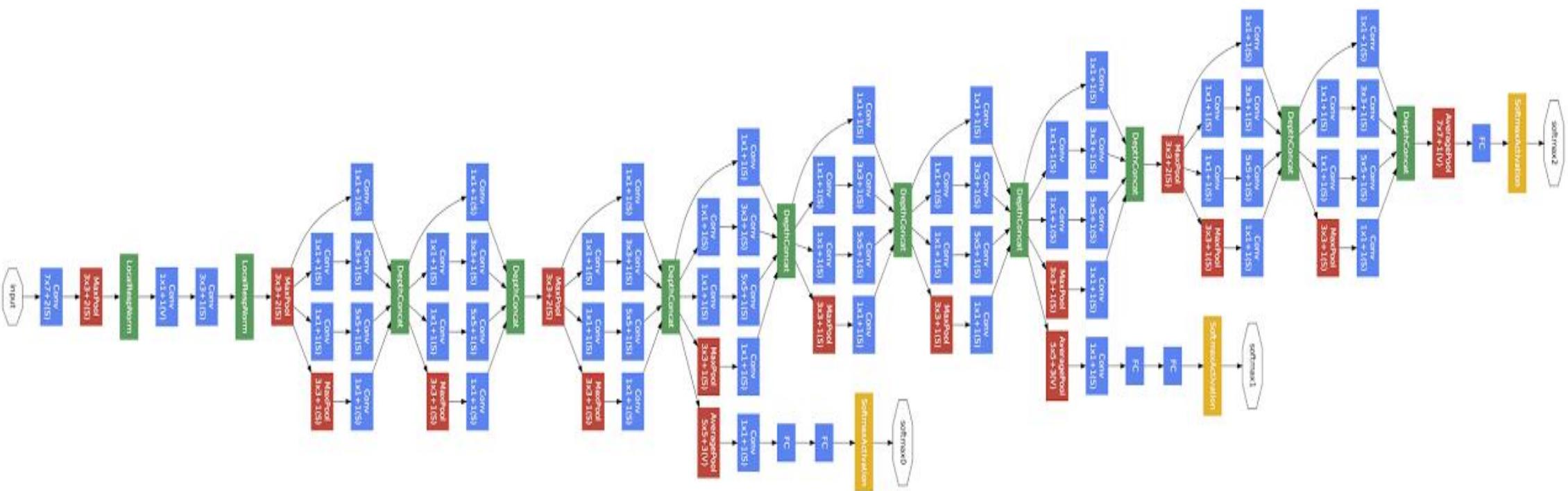
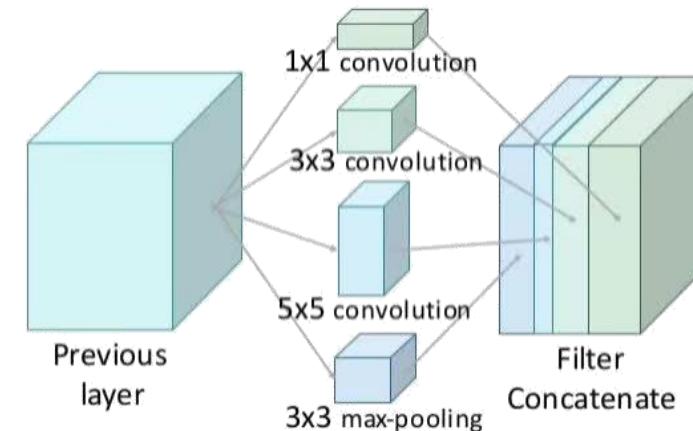
- Alex net (Alex, Hinton, 2012)
 - 5 convolution layers (with max-pooling) and 3 fully-connected neural nets
 - Use **dropout** and **LRN** (Local Response Normalization)
 - GPU Acceleration



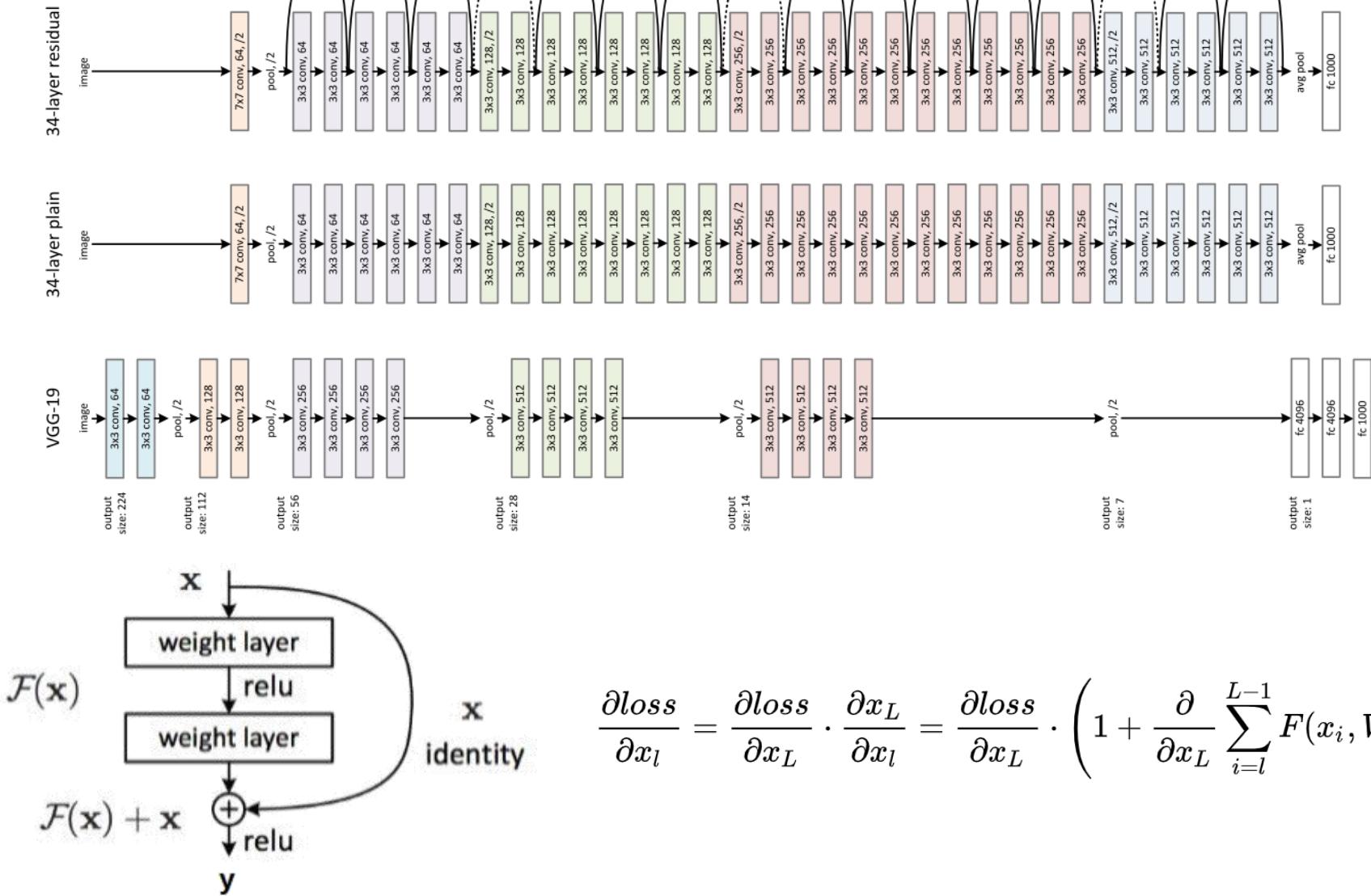
GoogLeNet

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

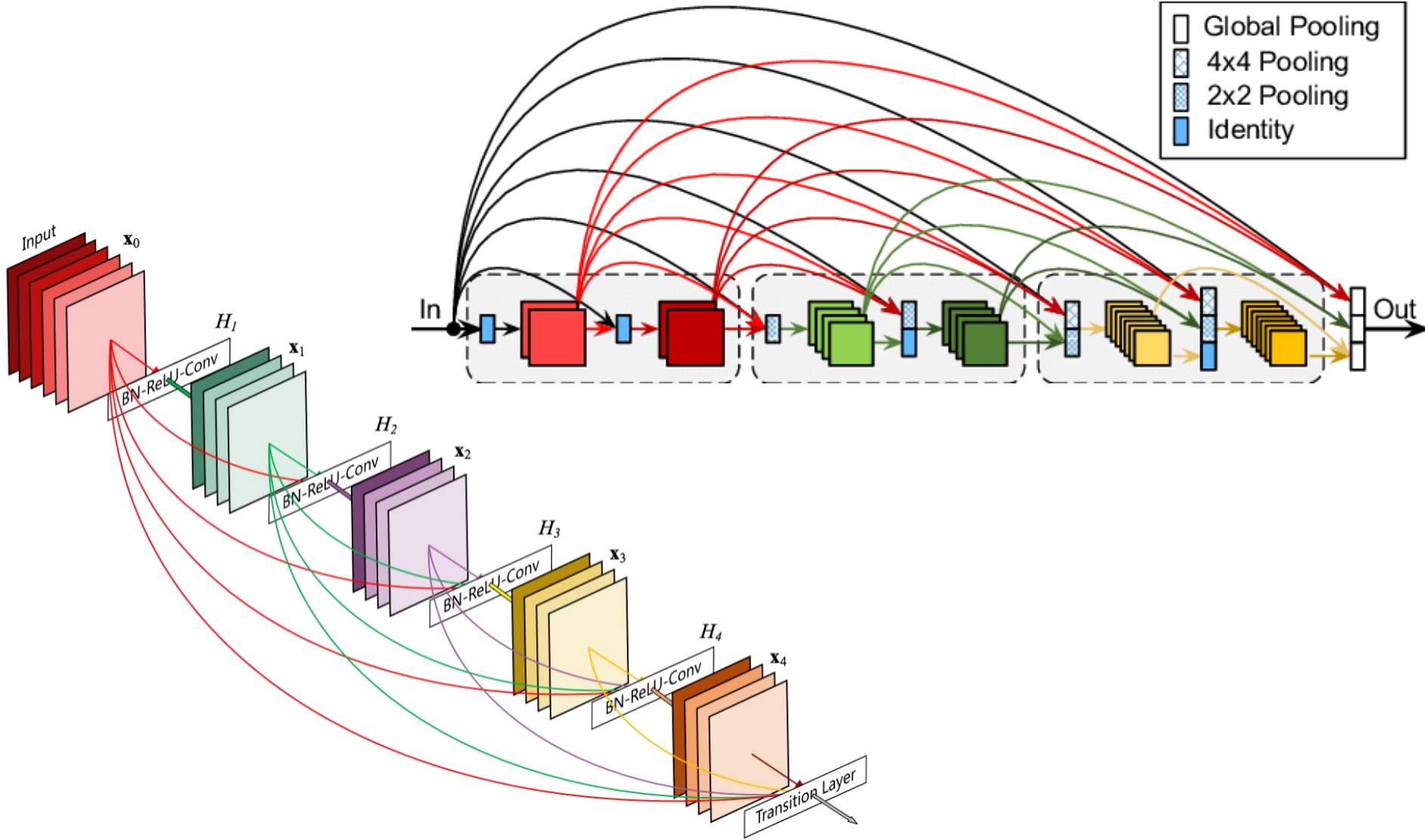
Inception Model



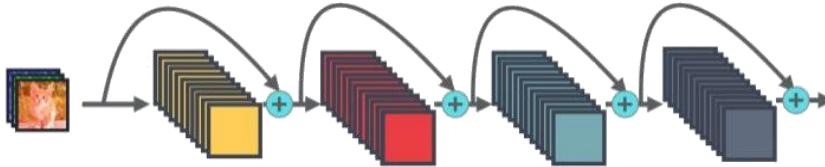
ResNet



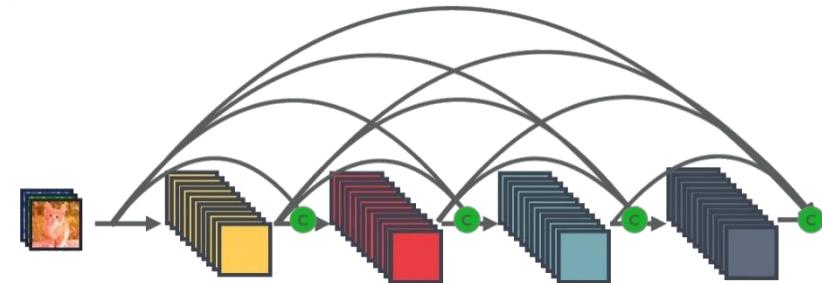
DenseNet



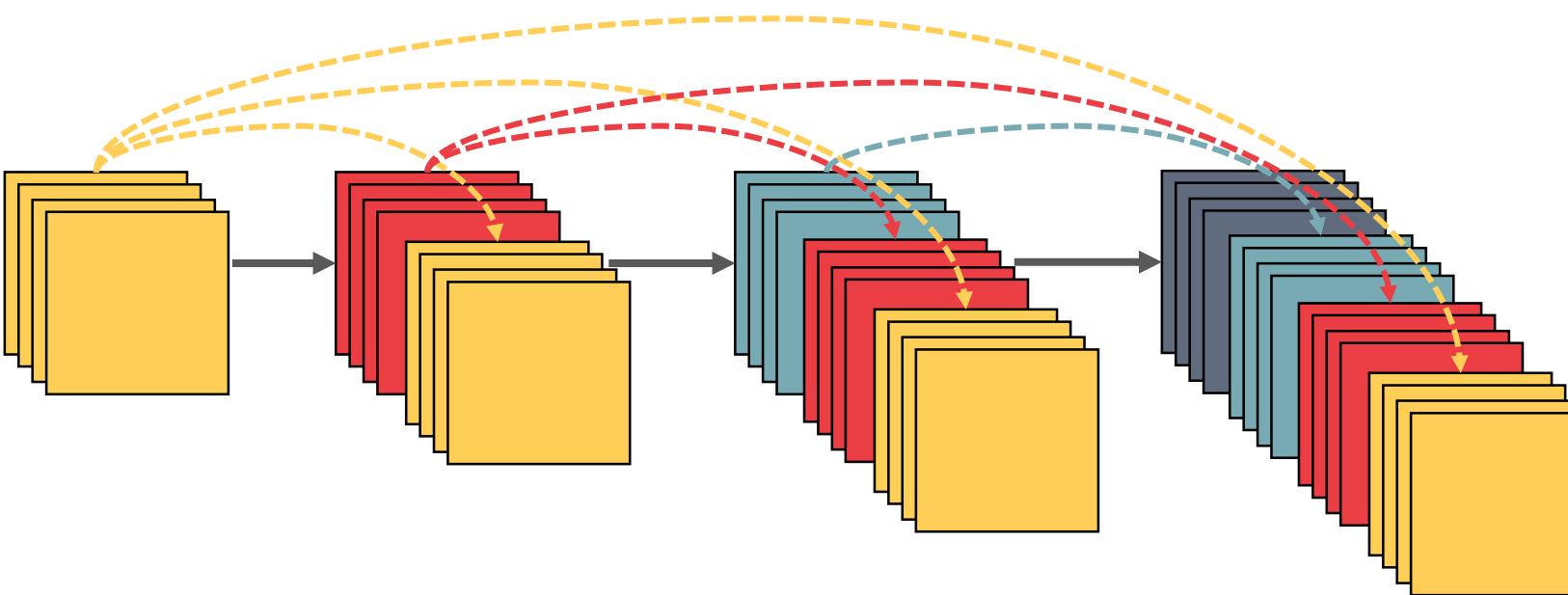
DenseNet



ResNet: Element-wise Addition

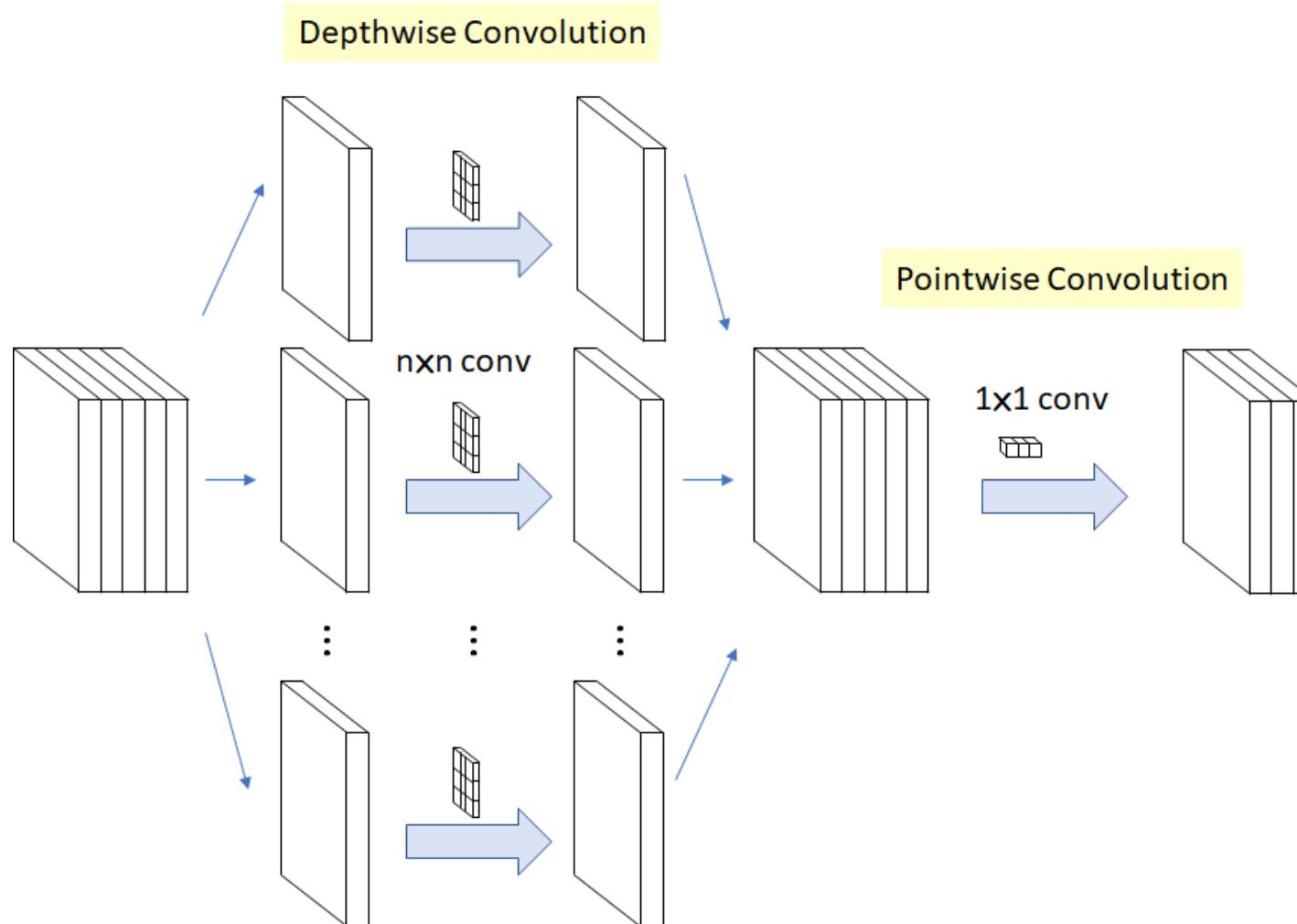


DenseNet: Channel-wise Concatenation



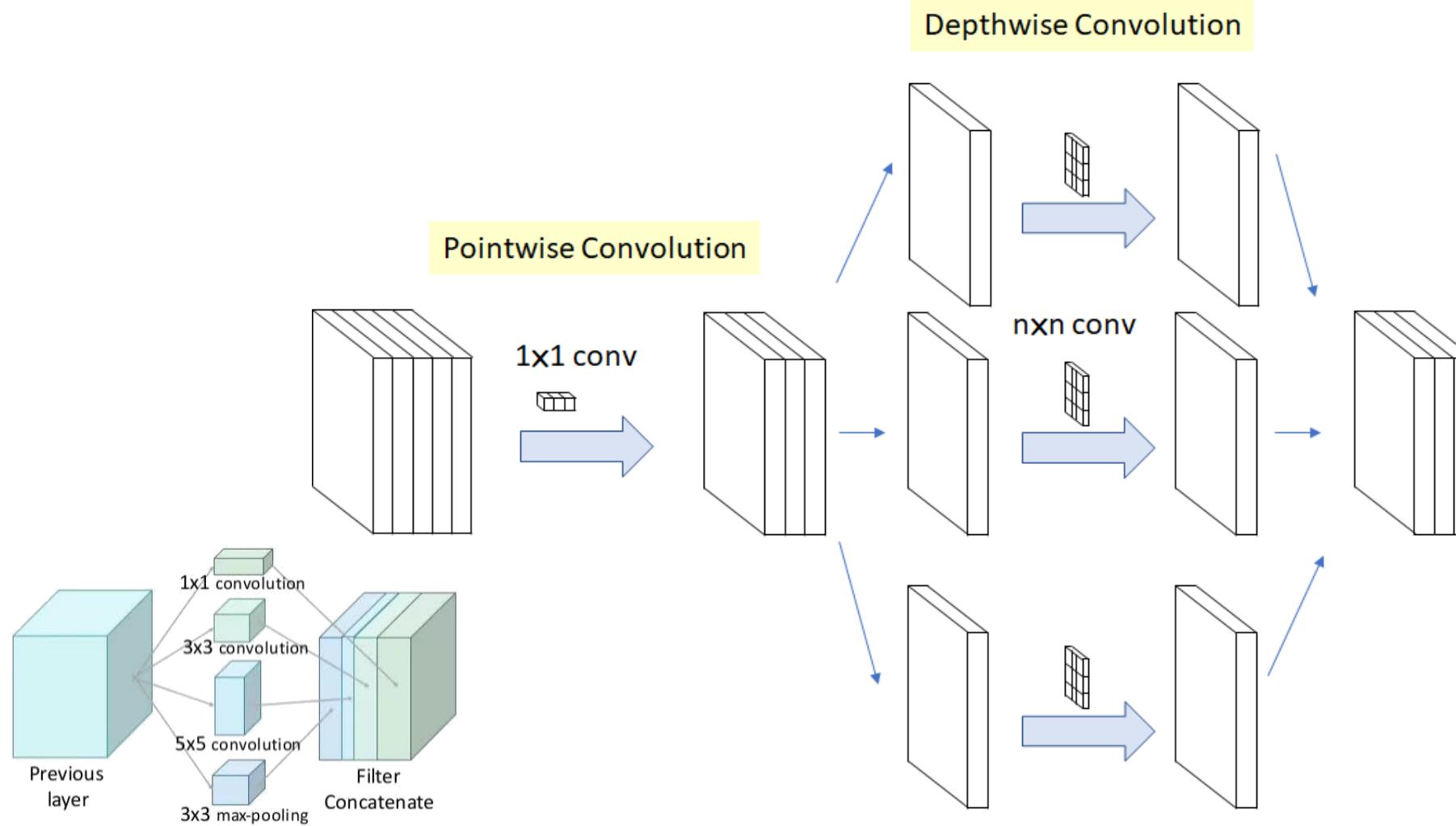
MobileNet

- Depthwise Separable Convolution



Xception

- Modified Depthwise Separable Convolution



SENet

- Squeeze-and-Excitation Networks

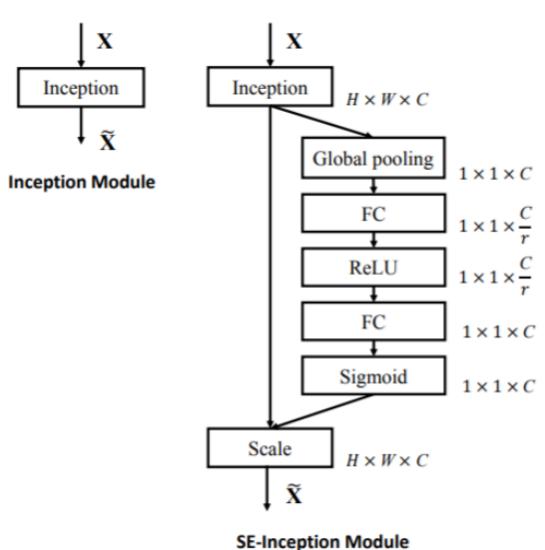
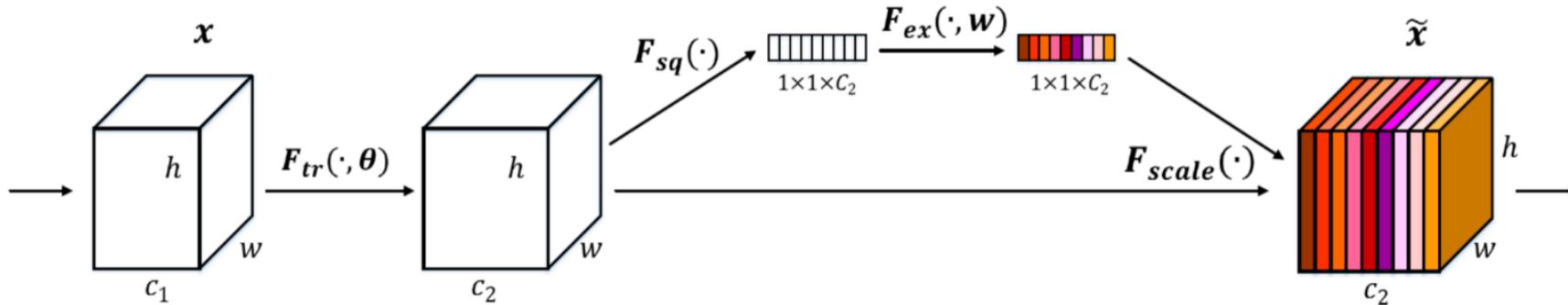


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

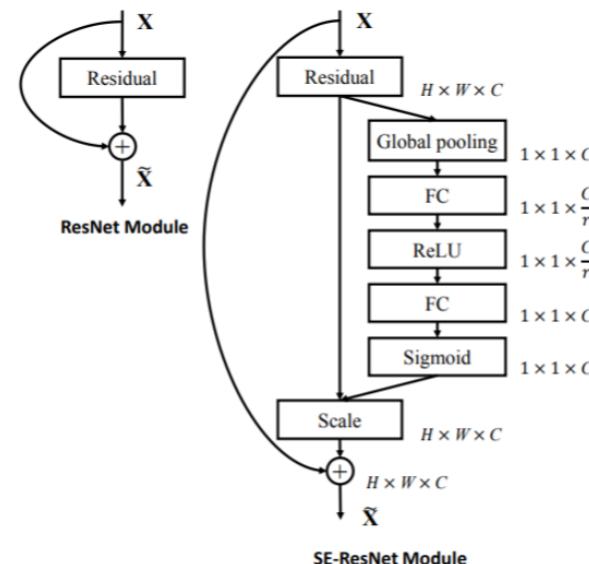
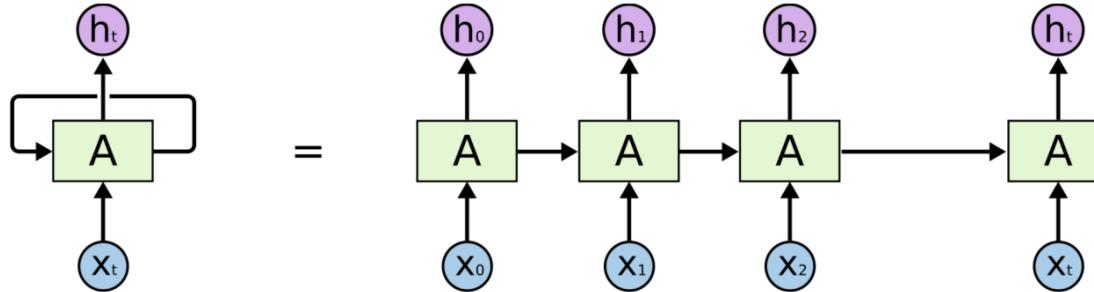


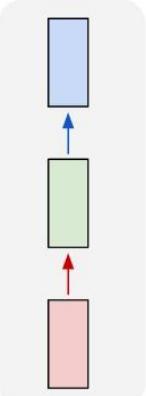
Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

Sequential Model

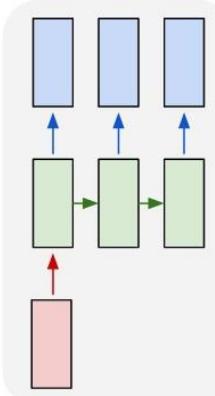
Recurrent Neural Network



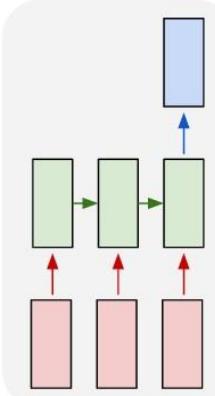
one to one



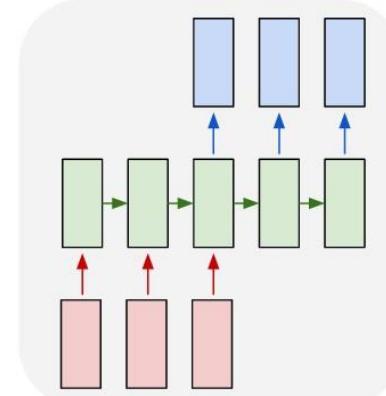
one to many



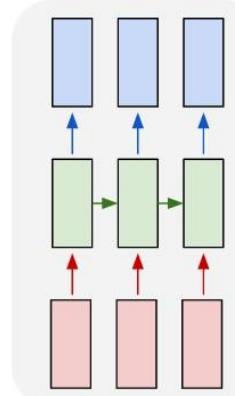
many to one



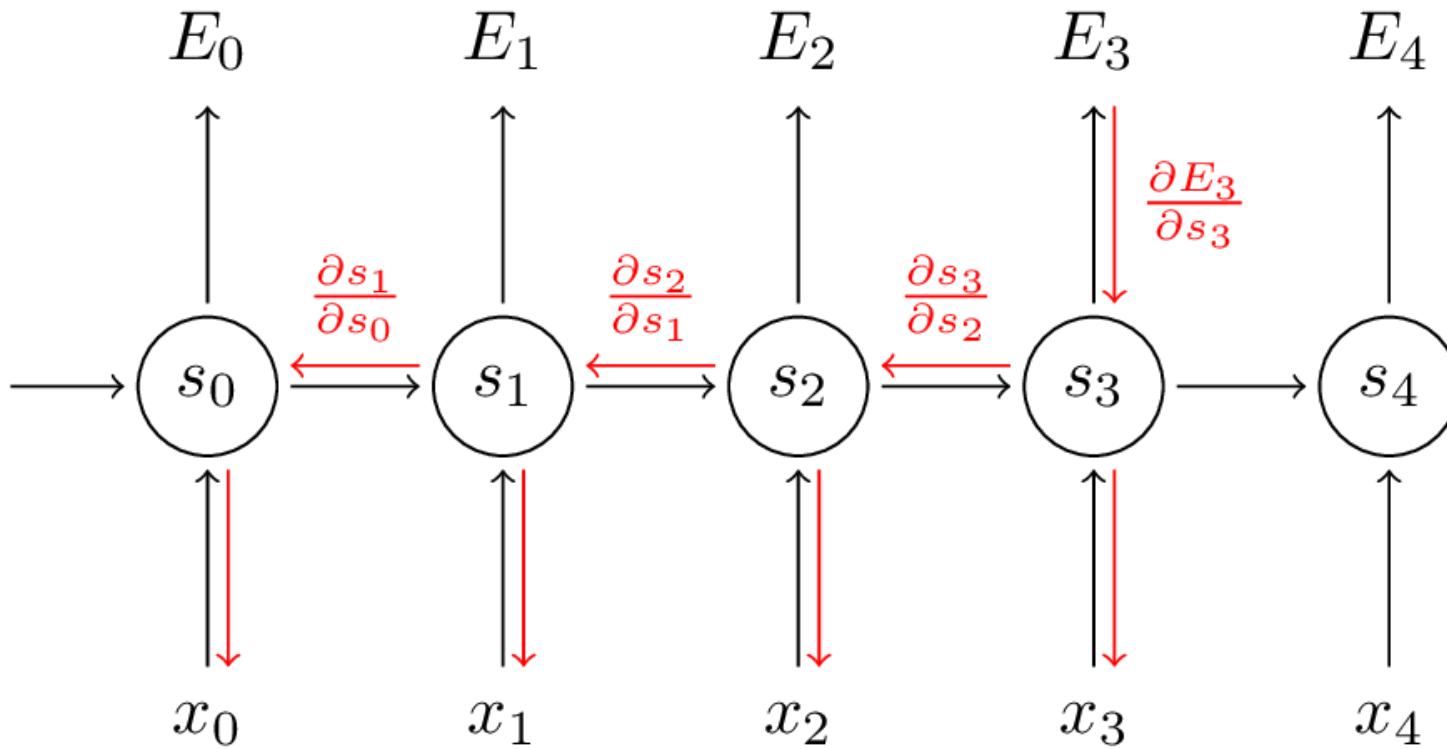
many to many



many to many

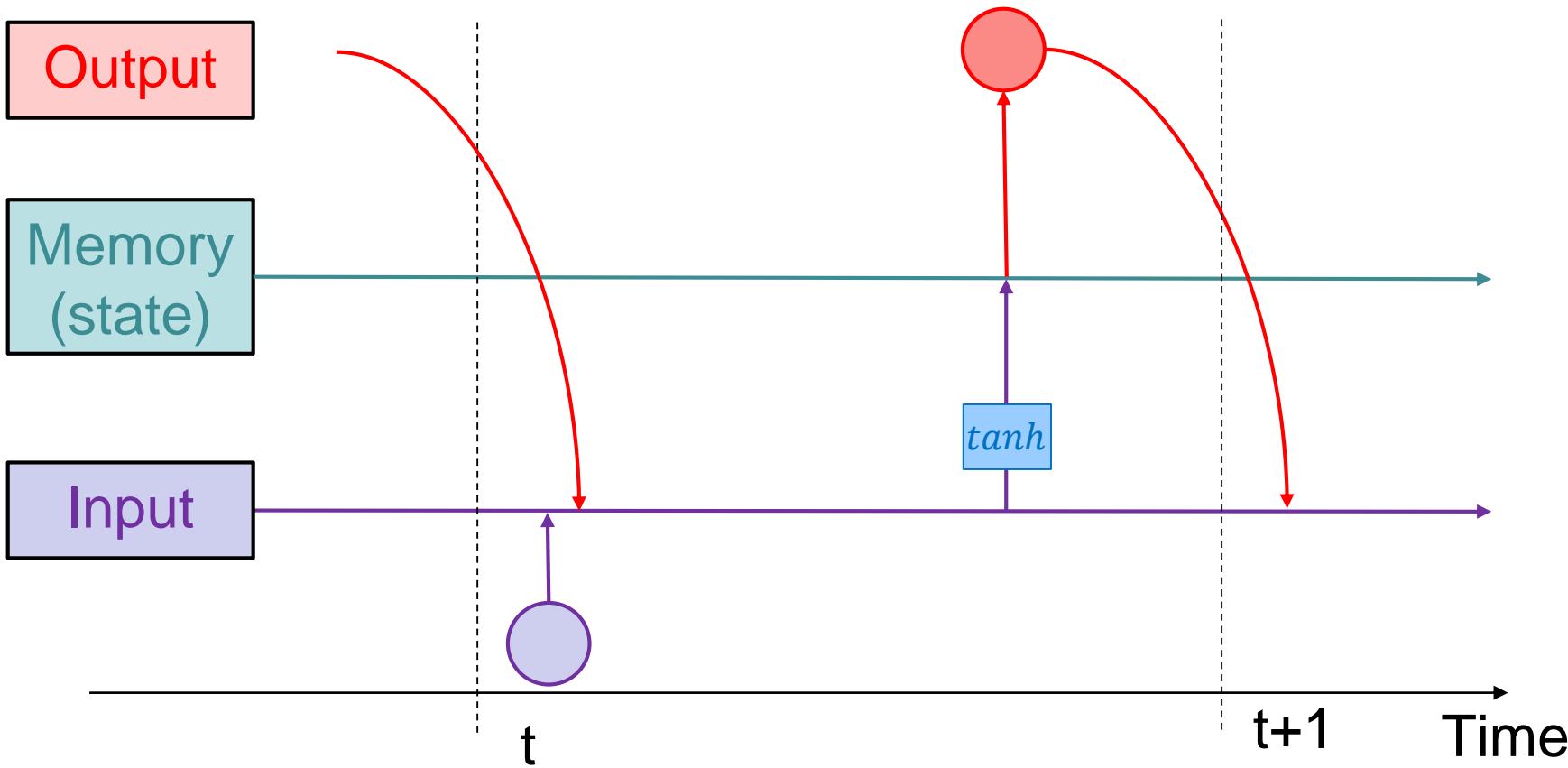


Back-Propagation Through Time (BPTT)

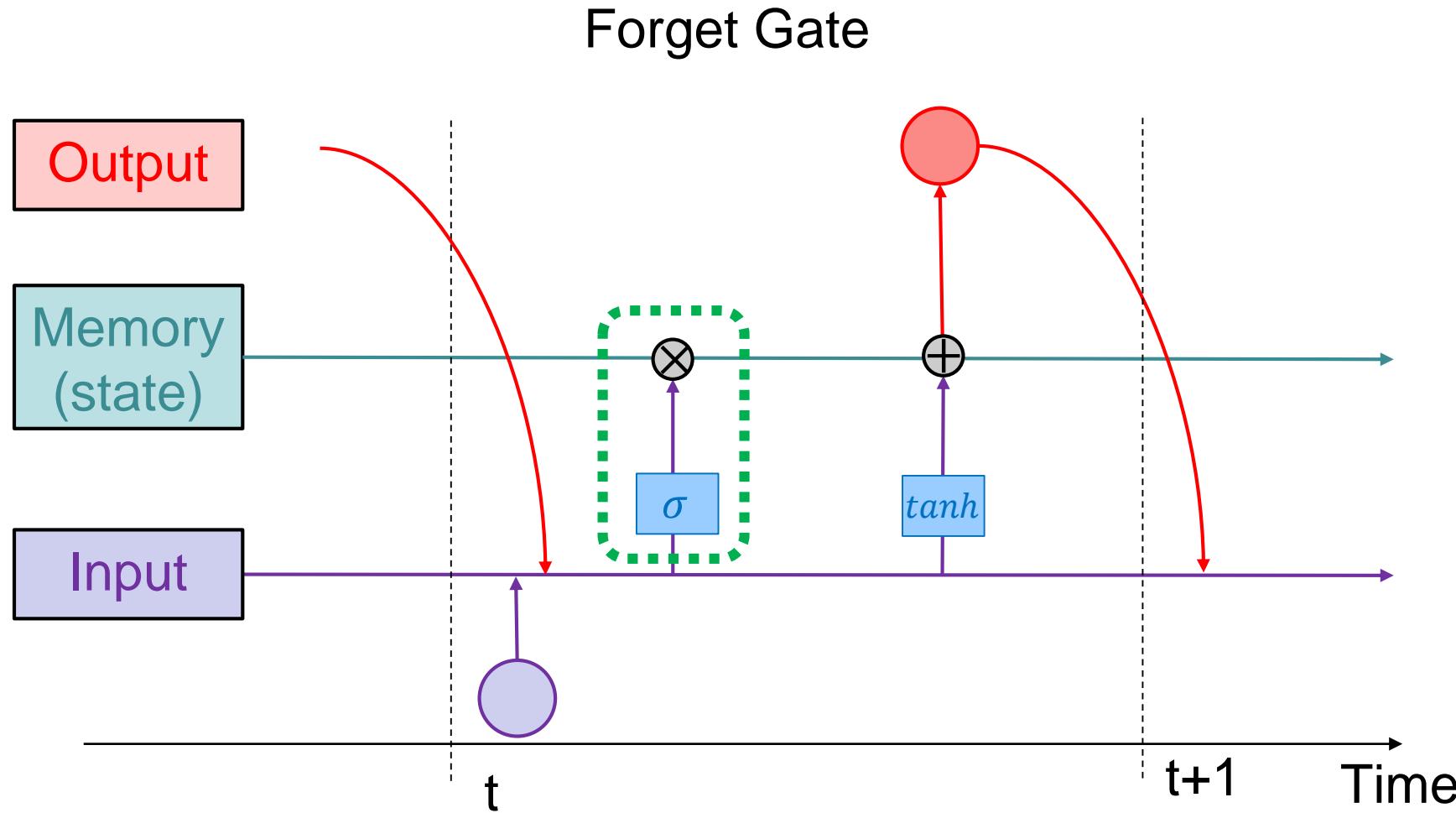


Long-Short Term Memory Network (LSTM)

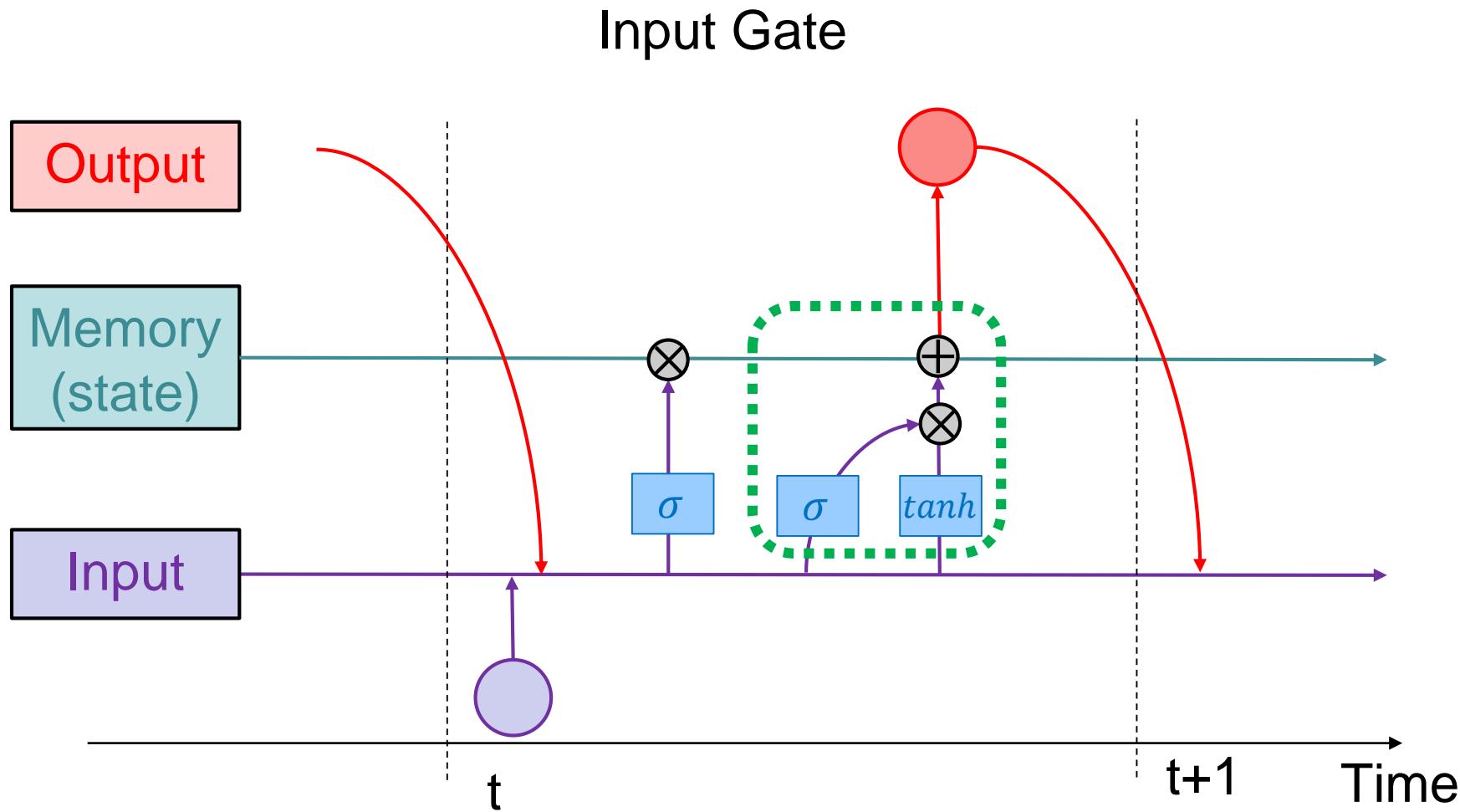
Vanilla Recurrent Neural Network



Long-Short Term Memory Network (LSTM)

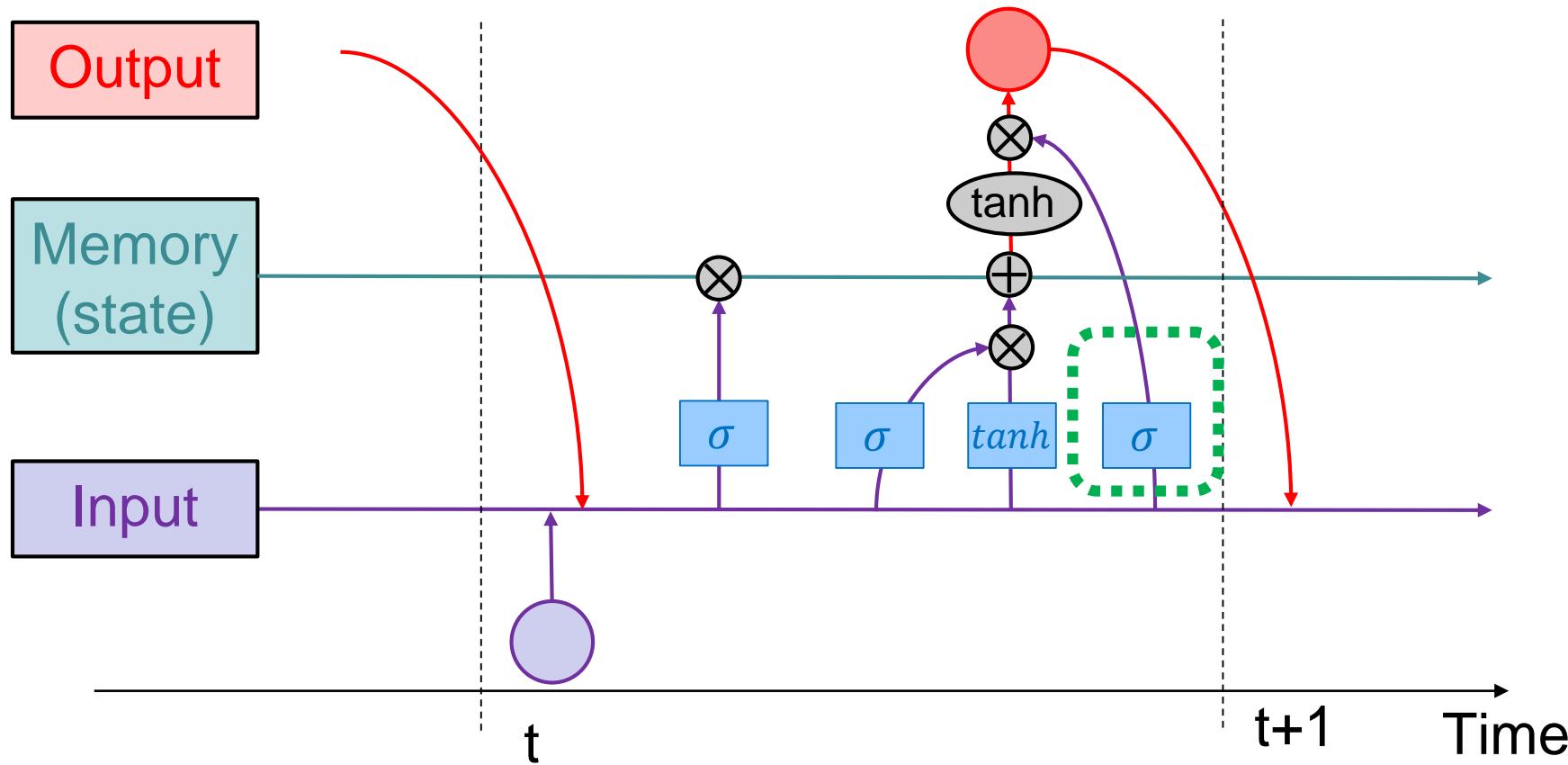


Long-Short Term Memory Network (LSTM)

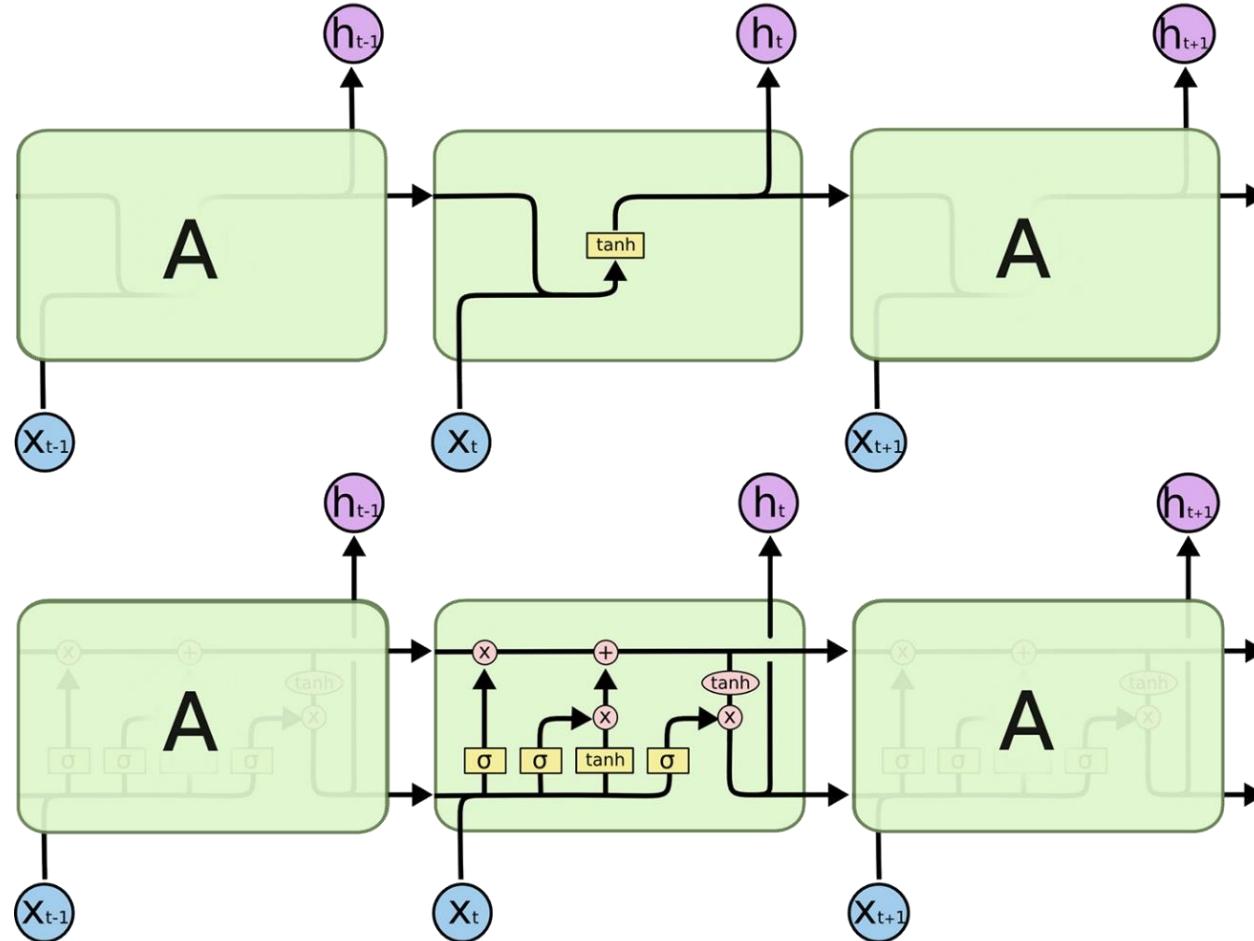


Long-Short Term Memory Network (LSTM)

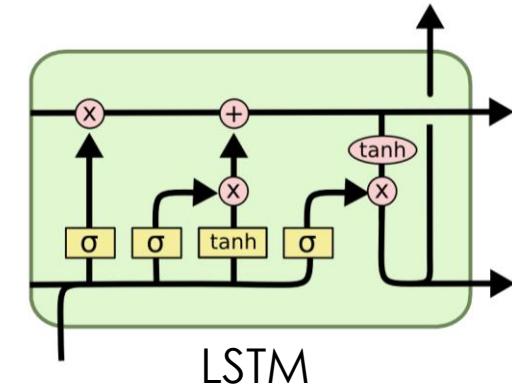
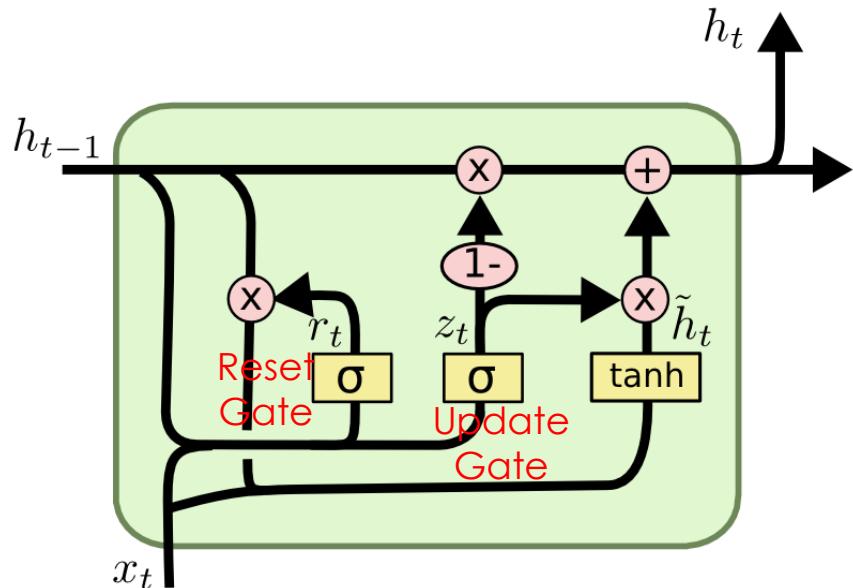
Output Gate



Long-Short Term Memory Network (LSTM)



Gated Recurrent Unit (GRU)



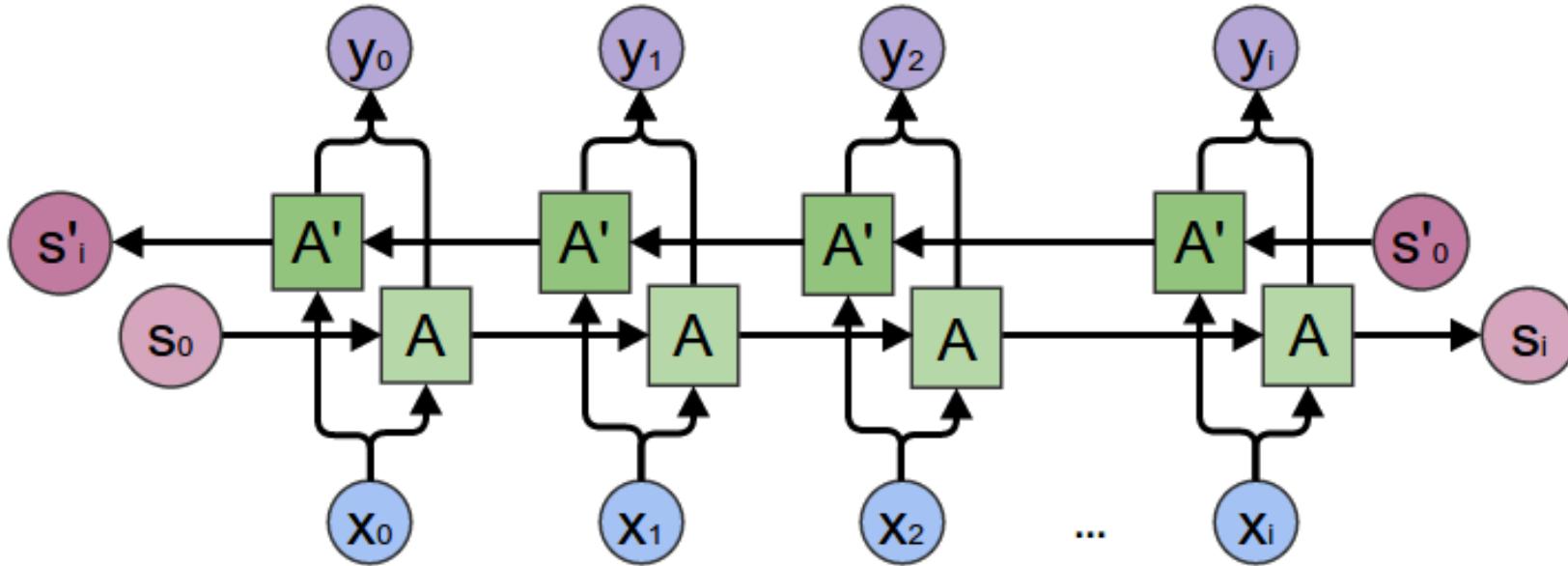
$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

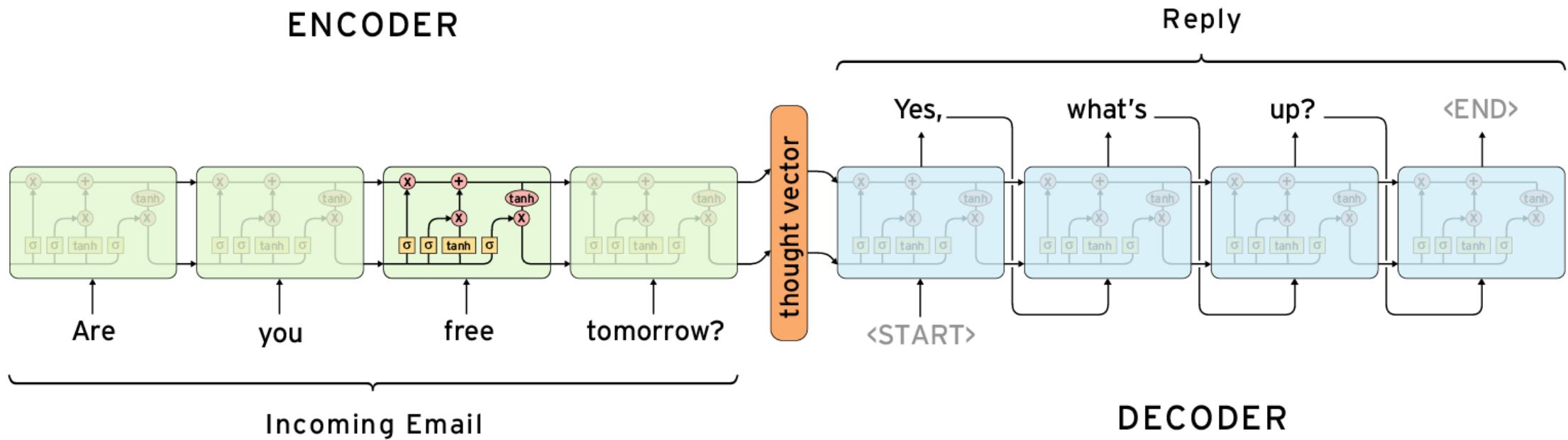
$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

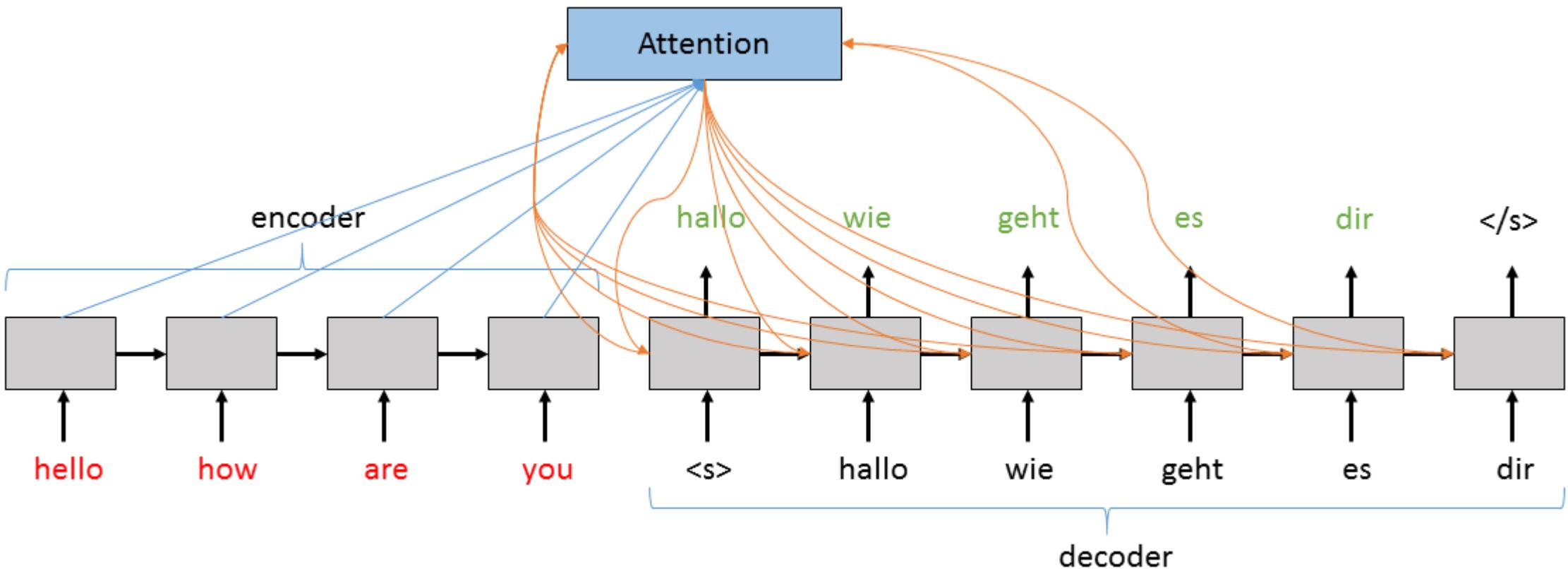
Bidirectional RNN



Sequence to Sequence (Seq2Seq)

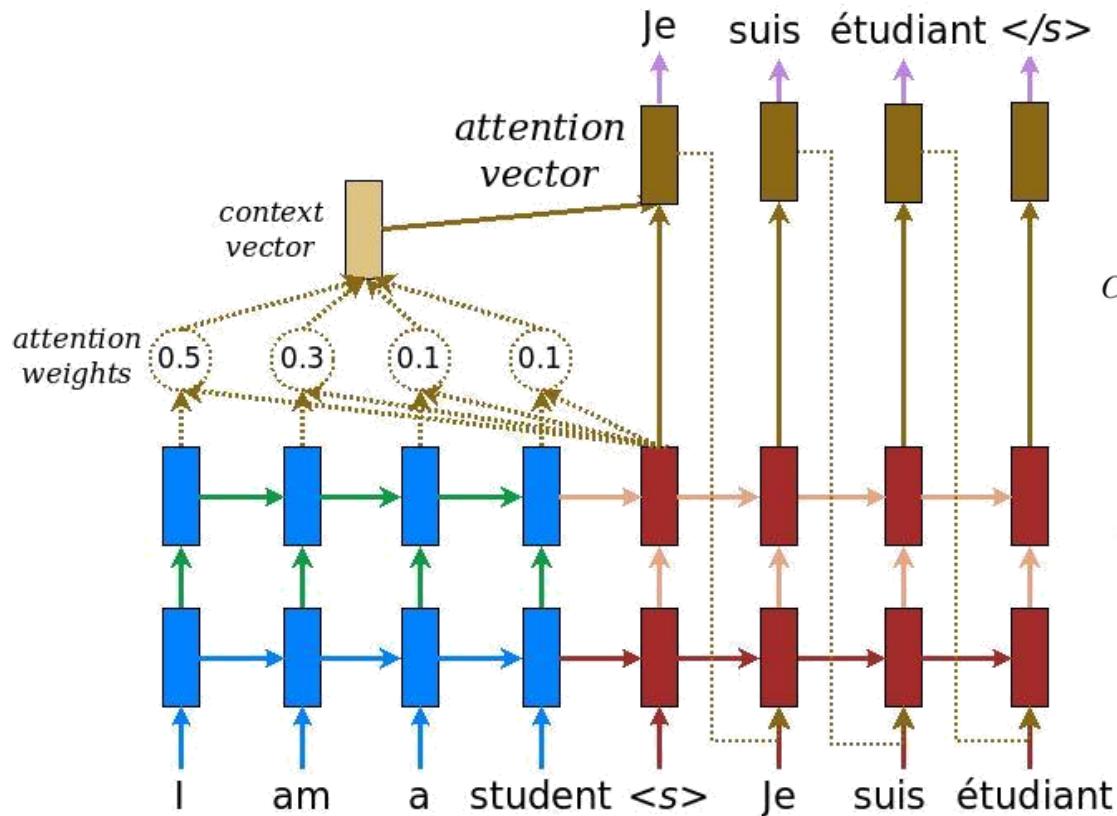


Attention-based Seq2Seq



Attention Mechanism

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \mathbf{W} \bar{\mathbf{h}}_s & \text{[Luong's multiplicative style]} \\ \mathbf{v}_a^\top \tanh (\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \bar{\mathbf{h}}_s) & \text{[Bahdanau's additive style]} \end{cases}$$

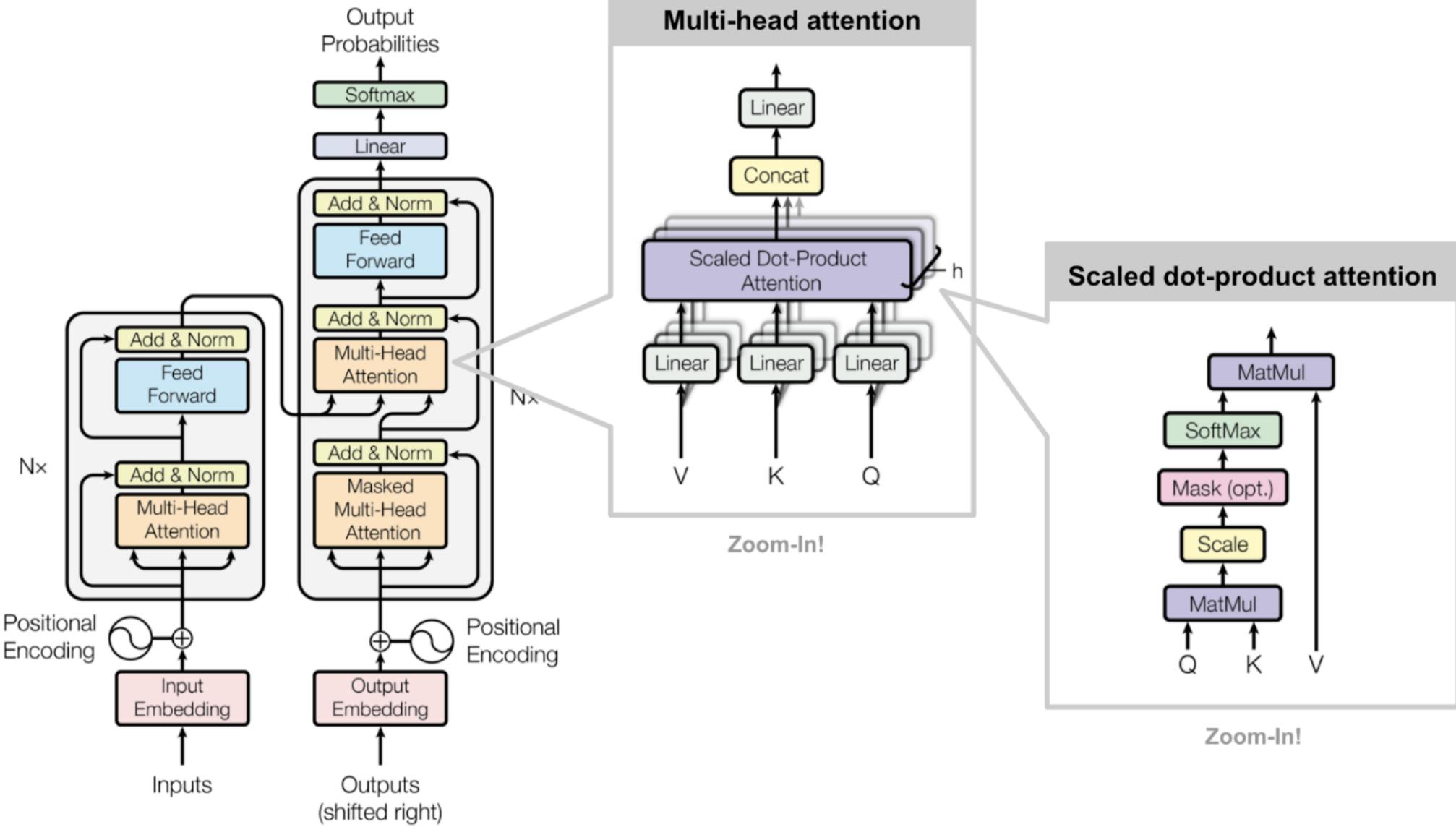


$$\alpha_{ts} = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'=1}^S \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))} \quad \text{[Attention weights]}$$

$$\mathbf{c}_t = \sum_s \alpha_{ts} \bar{\mathbf{h}}_s \quad \text{[Context vector]}$$

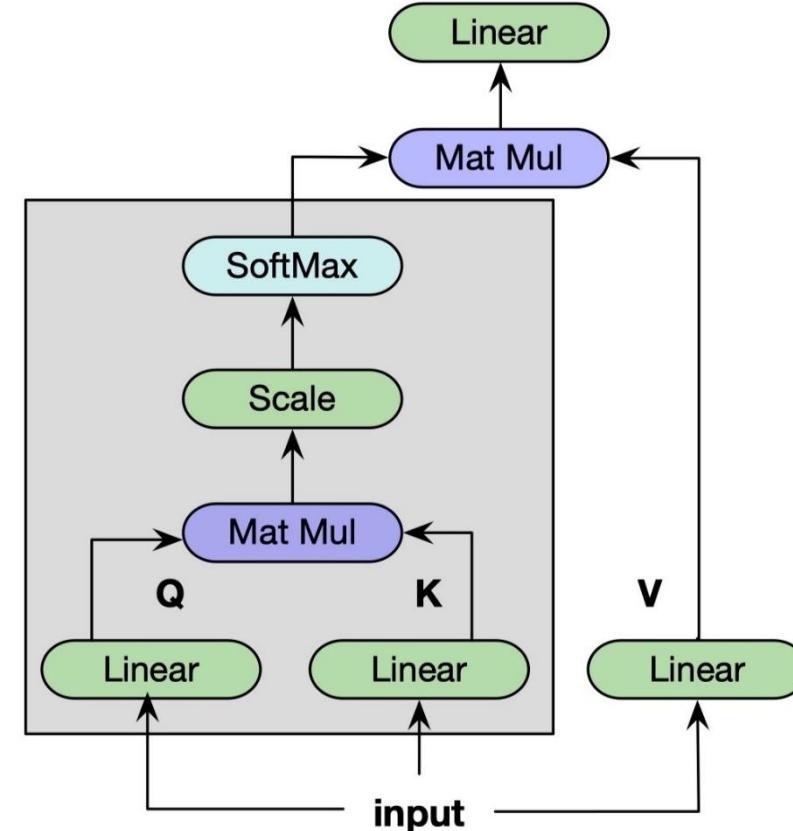
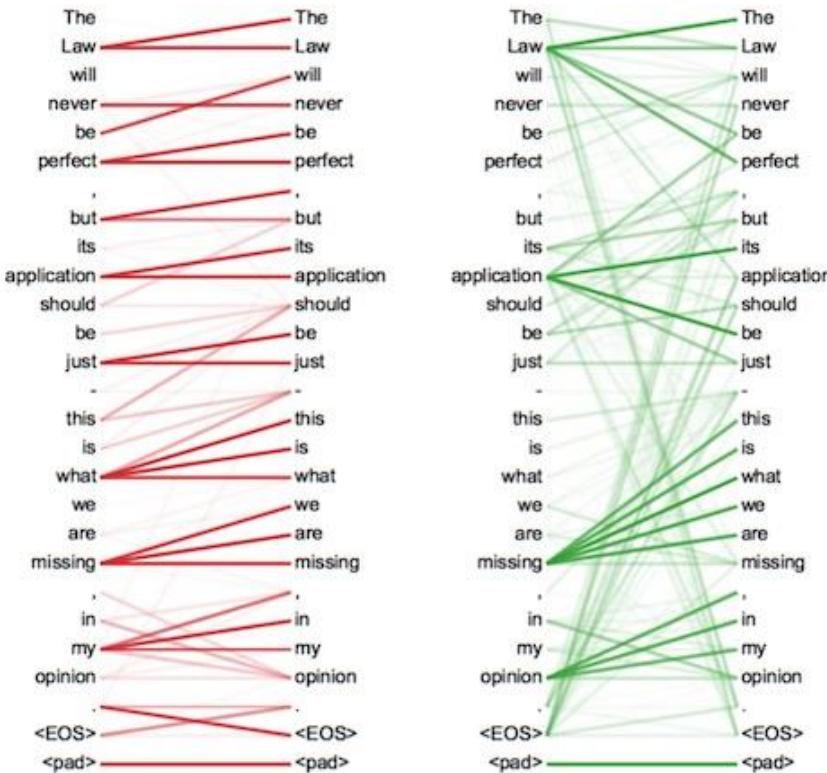
$$\mathbf{a}_t = f(\mathbf{c}_t, \mathbf{h}_t) = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \quad \text{[Attention vector]}$$

Transformer



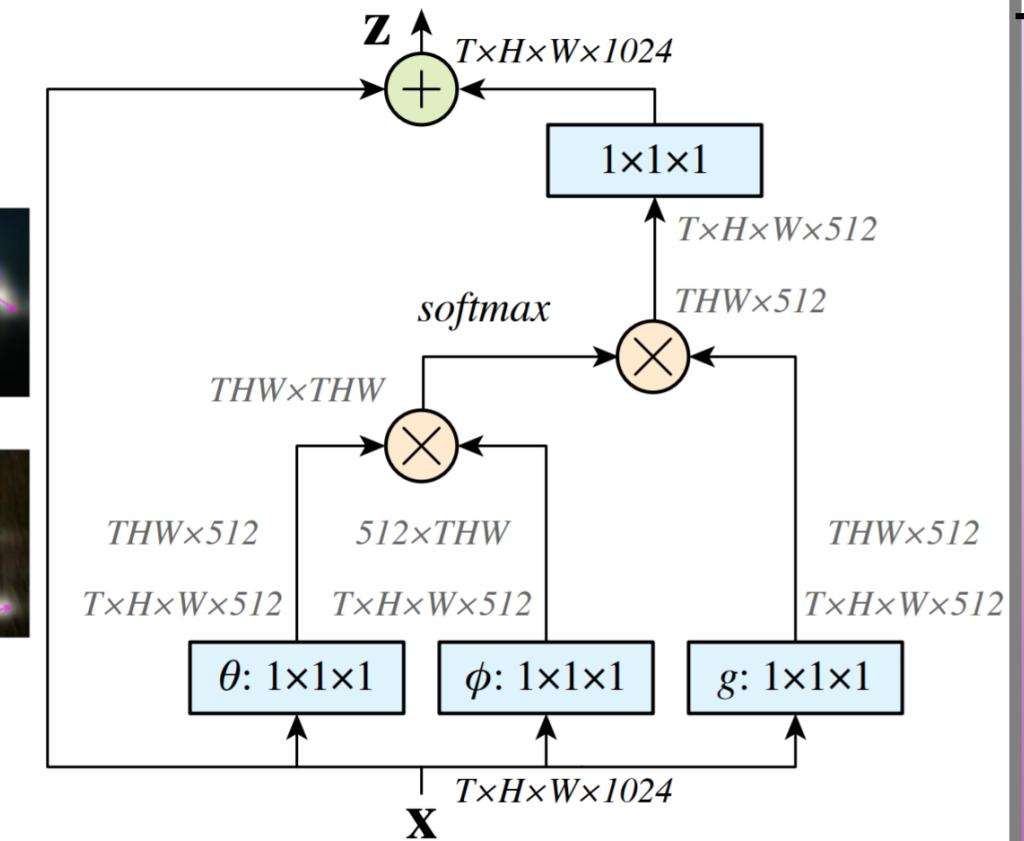
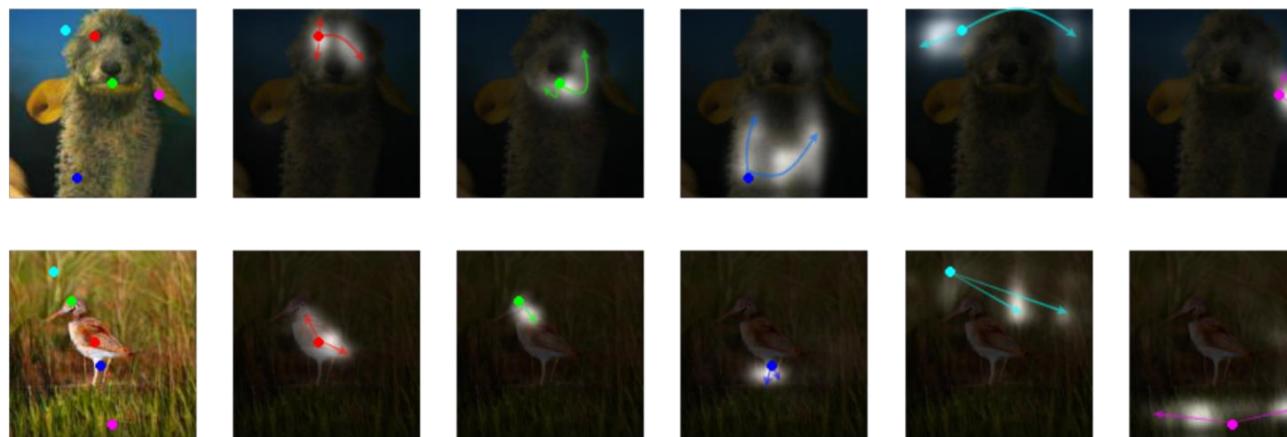
Self-Attention

$$\text{Attention}(Q, K, V) = \text{Softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$



(a) Self-attention

Self-Attention on Image



Application on CV

Computer Vision Problems

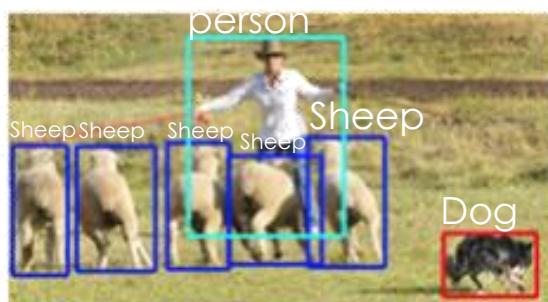
➤ Object Classification

- 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



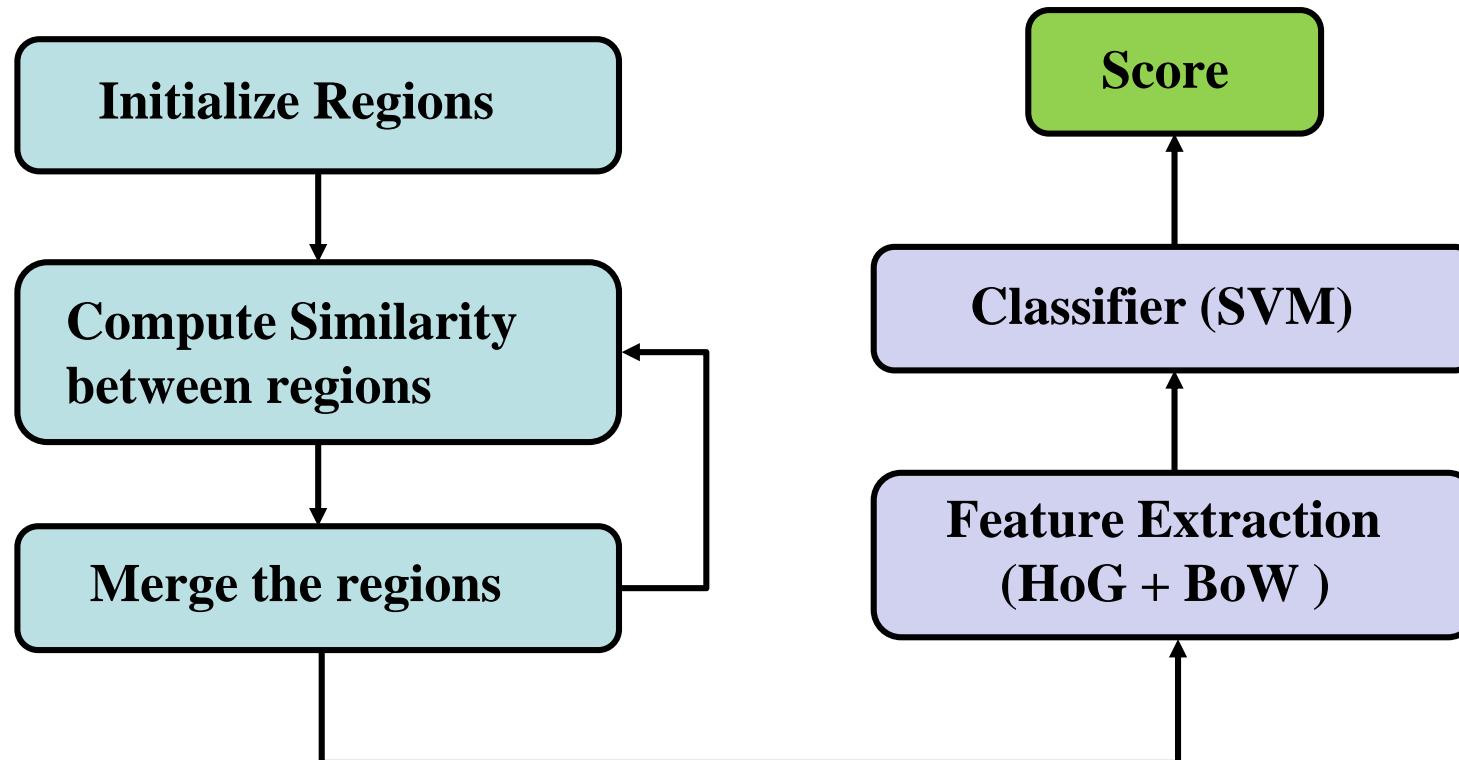
➤ Instance Segmentation

- Segment each object instance pixel-by-pixel

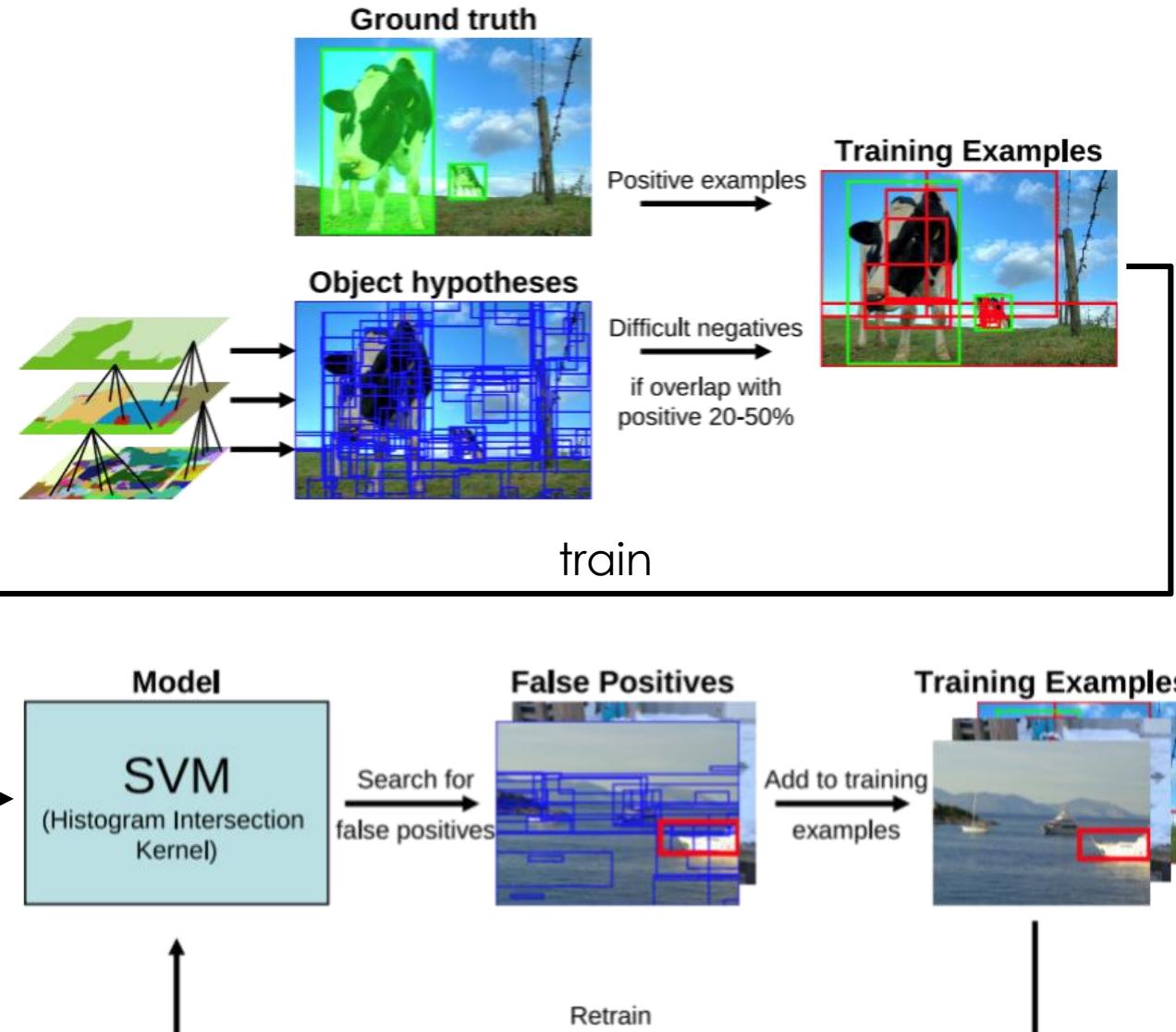


Two-stage Object Detection

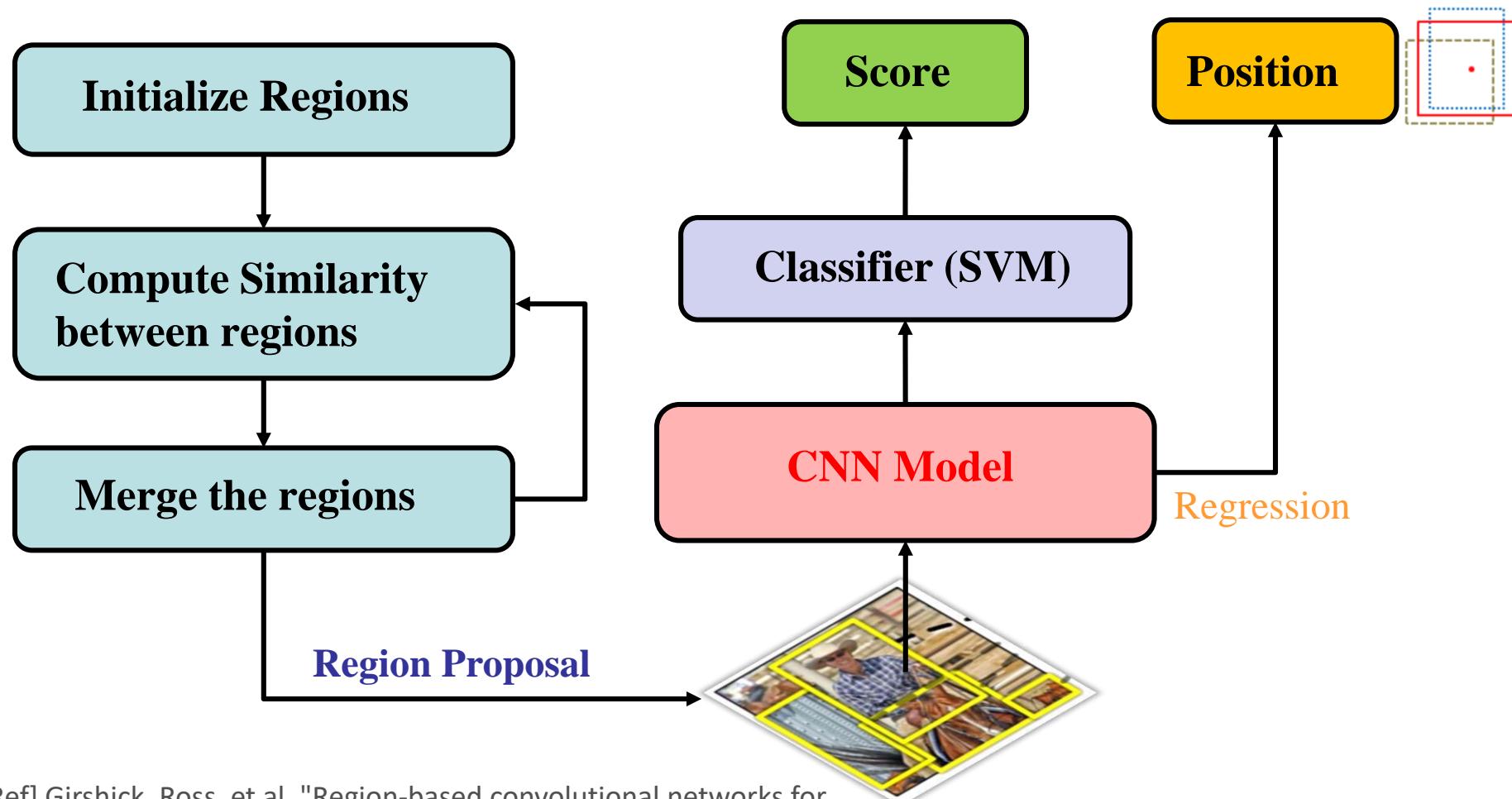
Selective Search



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



crop

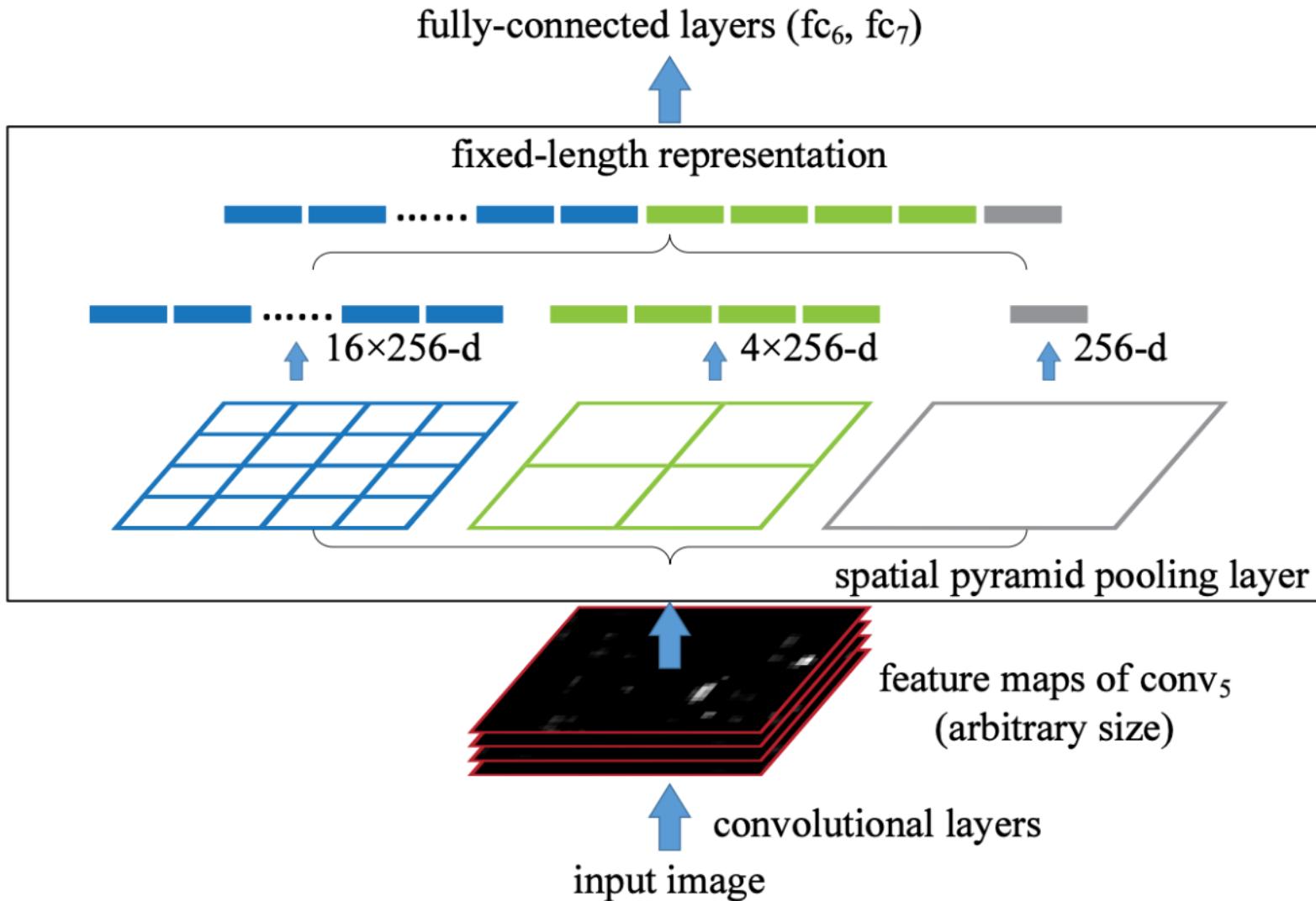


warp

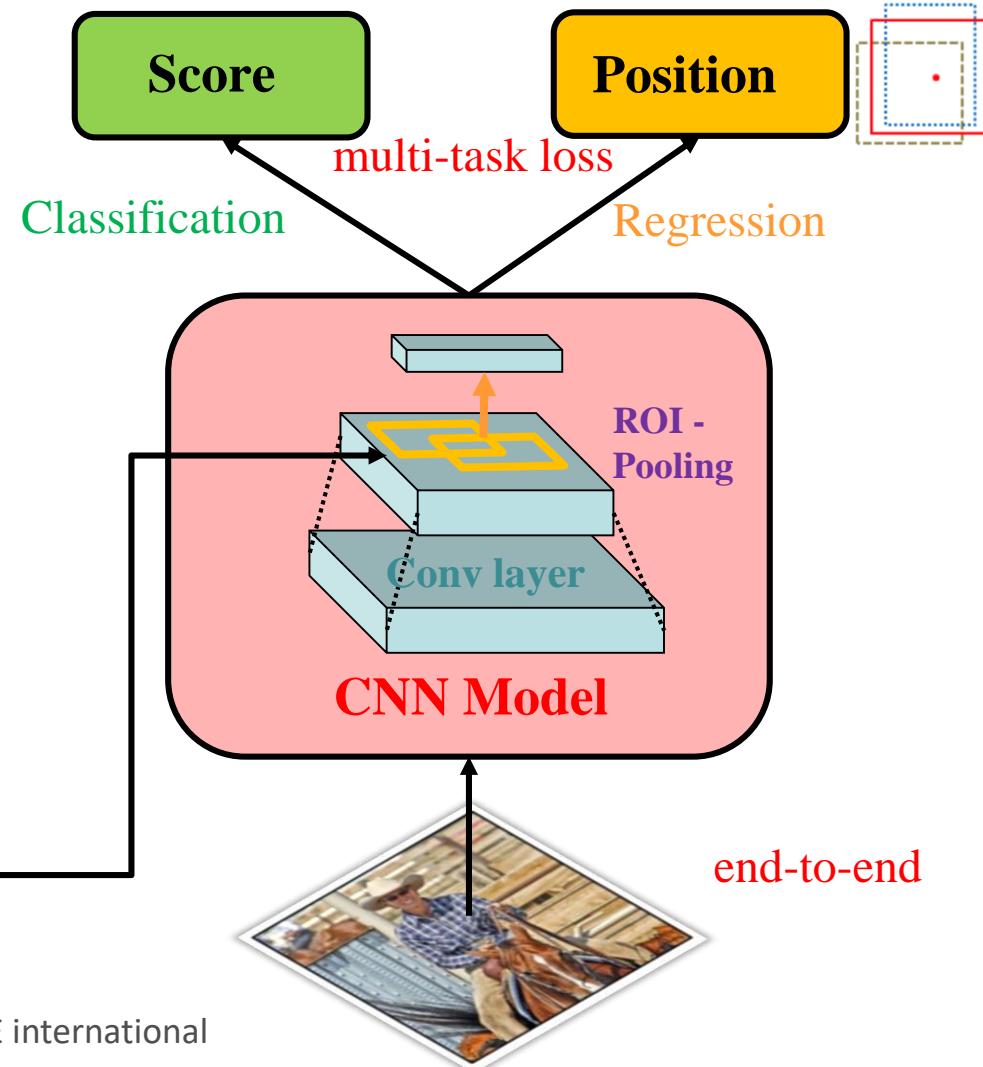
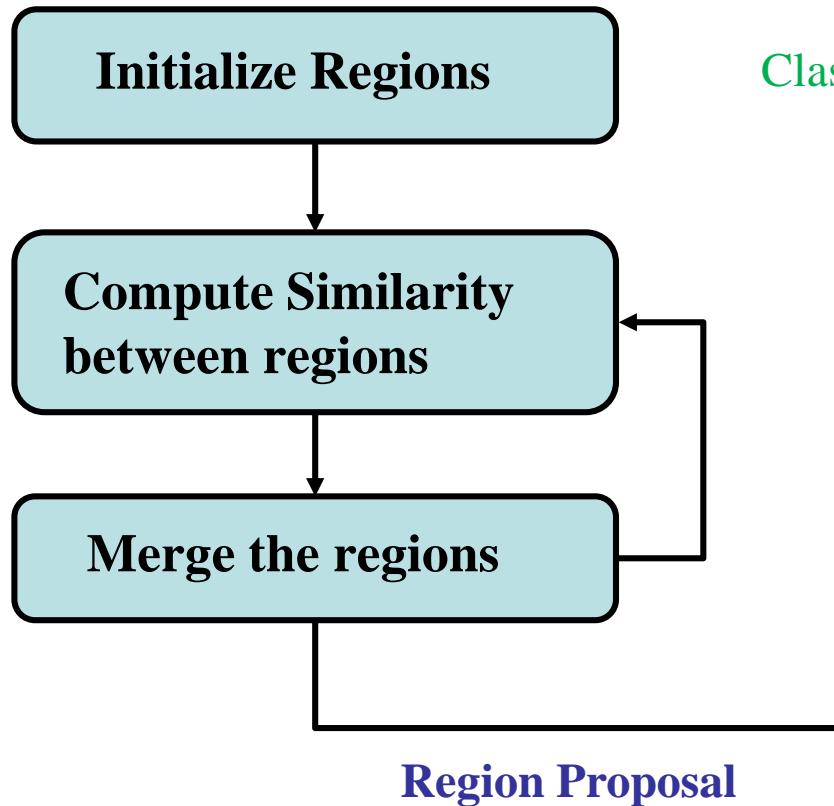


[Ref] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." European conference on computer vision. Springer, Cham, 2014.

Spatial Pyramid Pooling

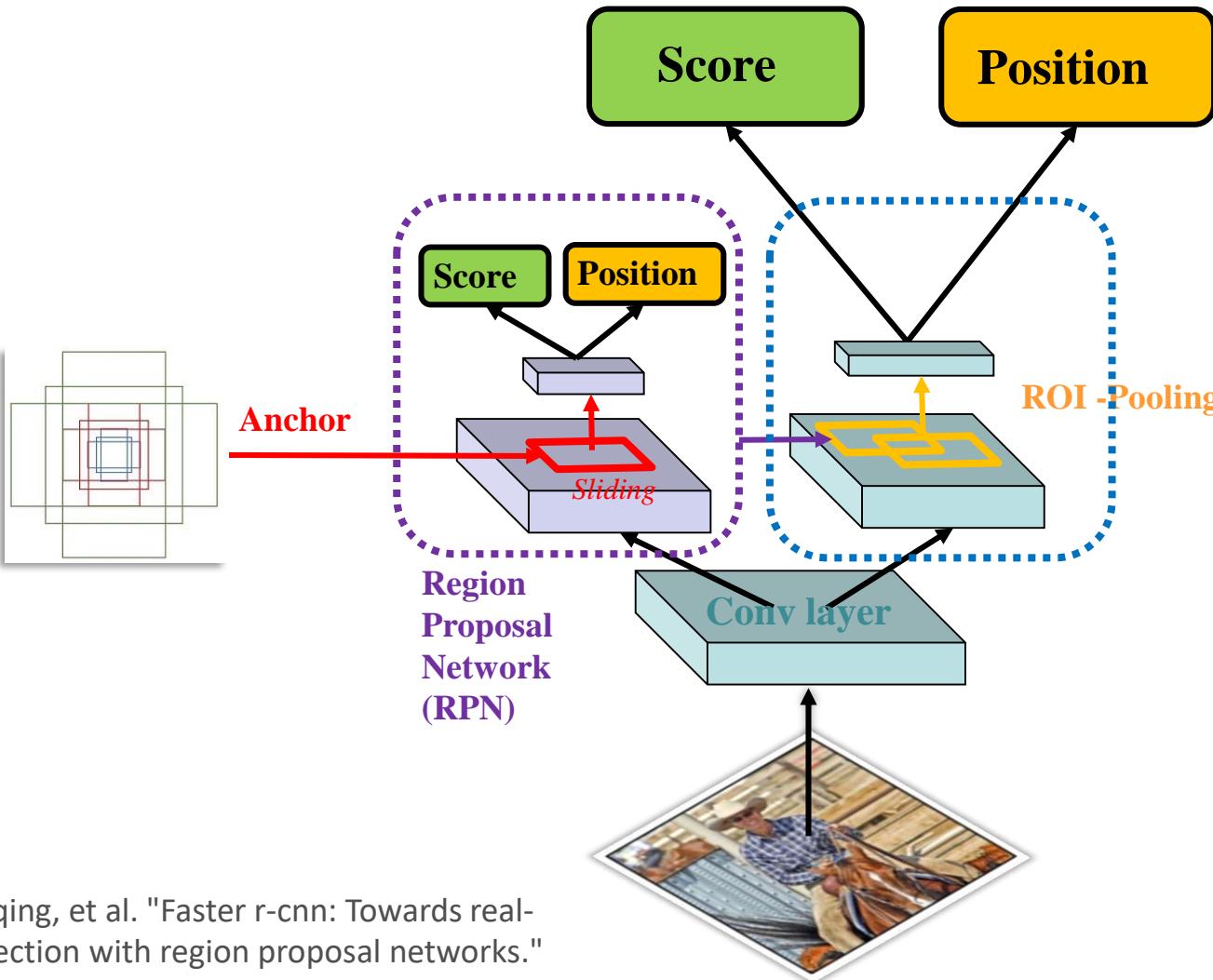


Fast R-CNN



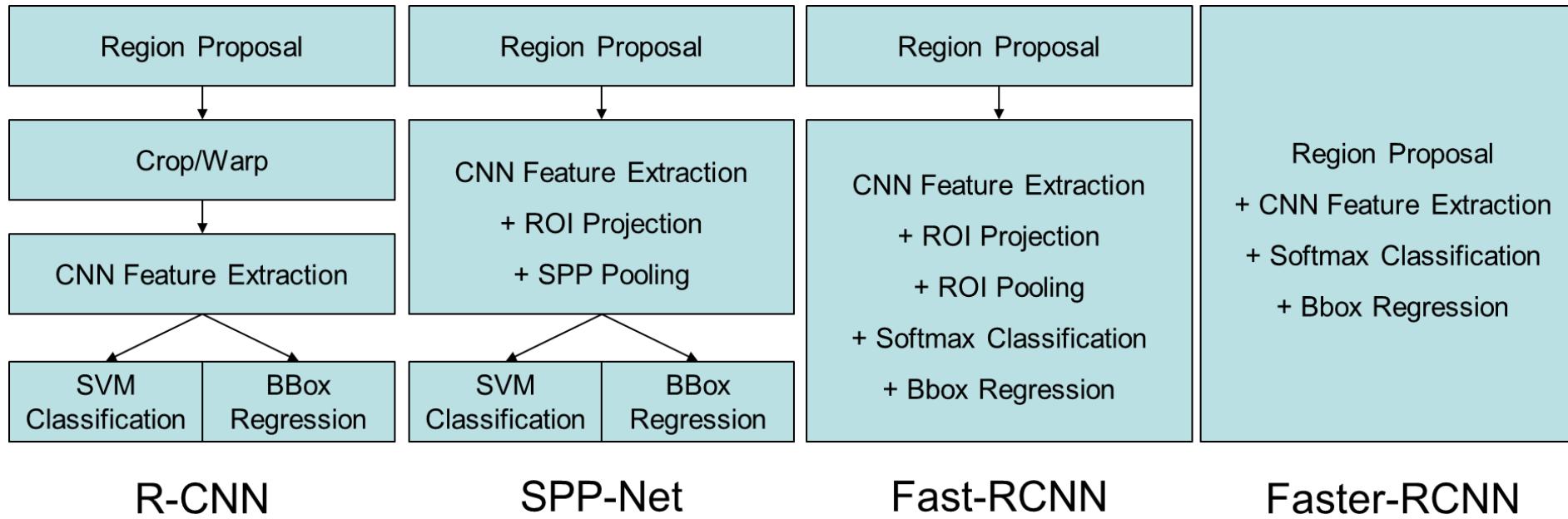
[Ref] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

Faster R-CNN



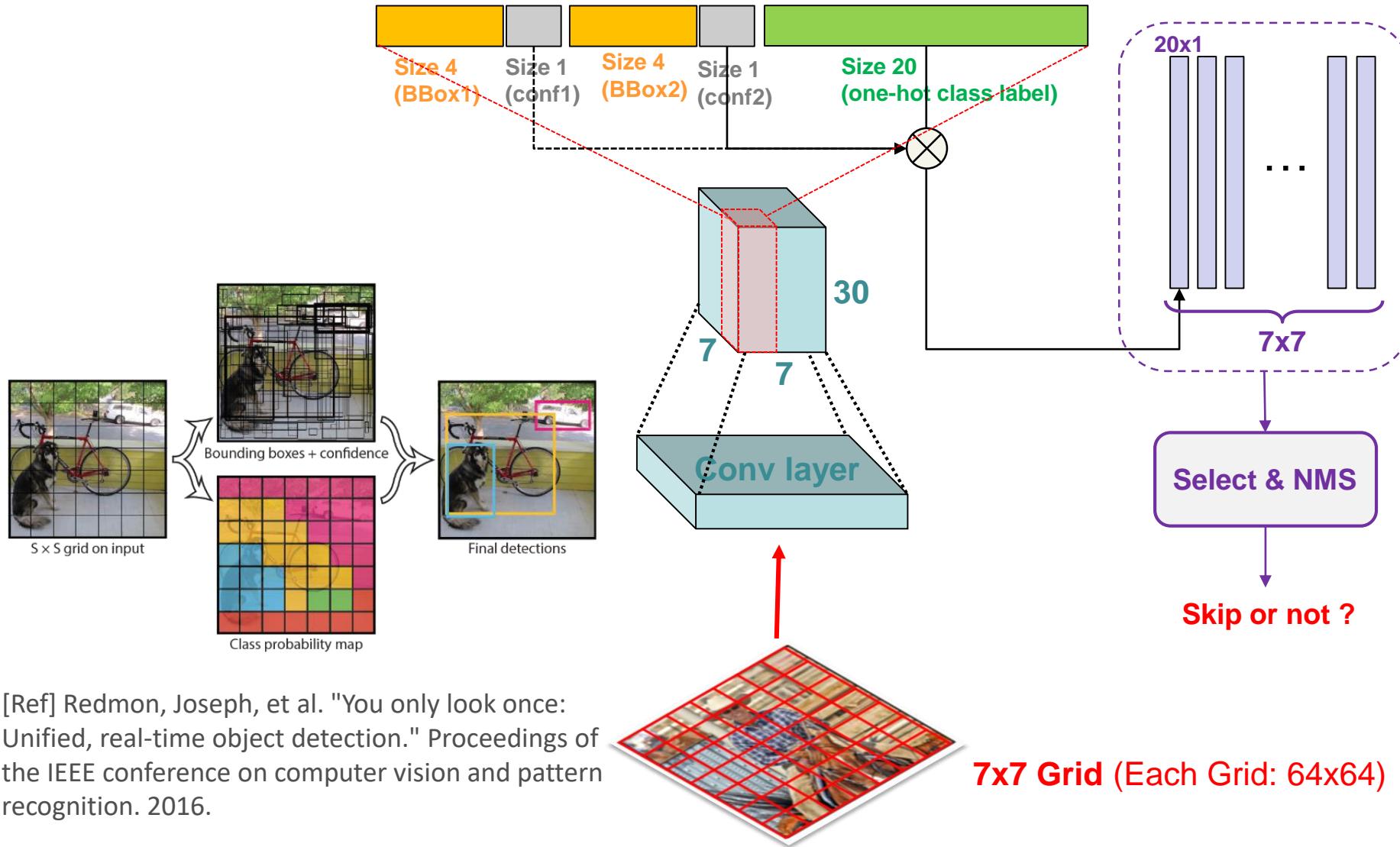
[Ref] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

Revolution of R-CNN



One-stage Object Detection

You Only Look Once (Yolo)



You Only Look Once (Yolo)

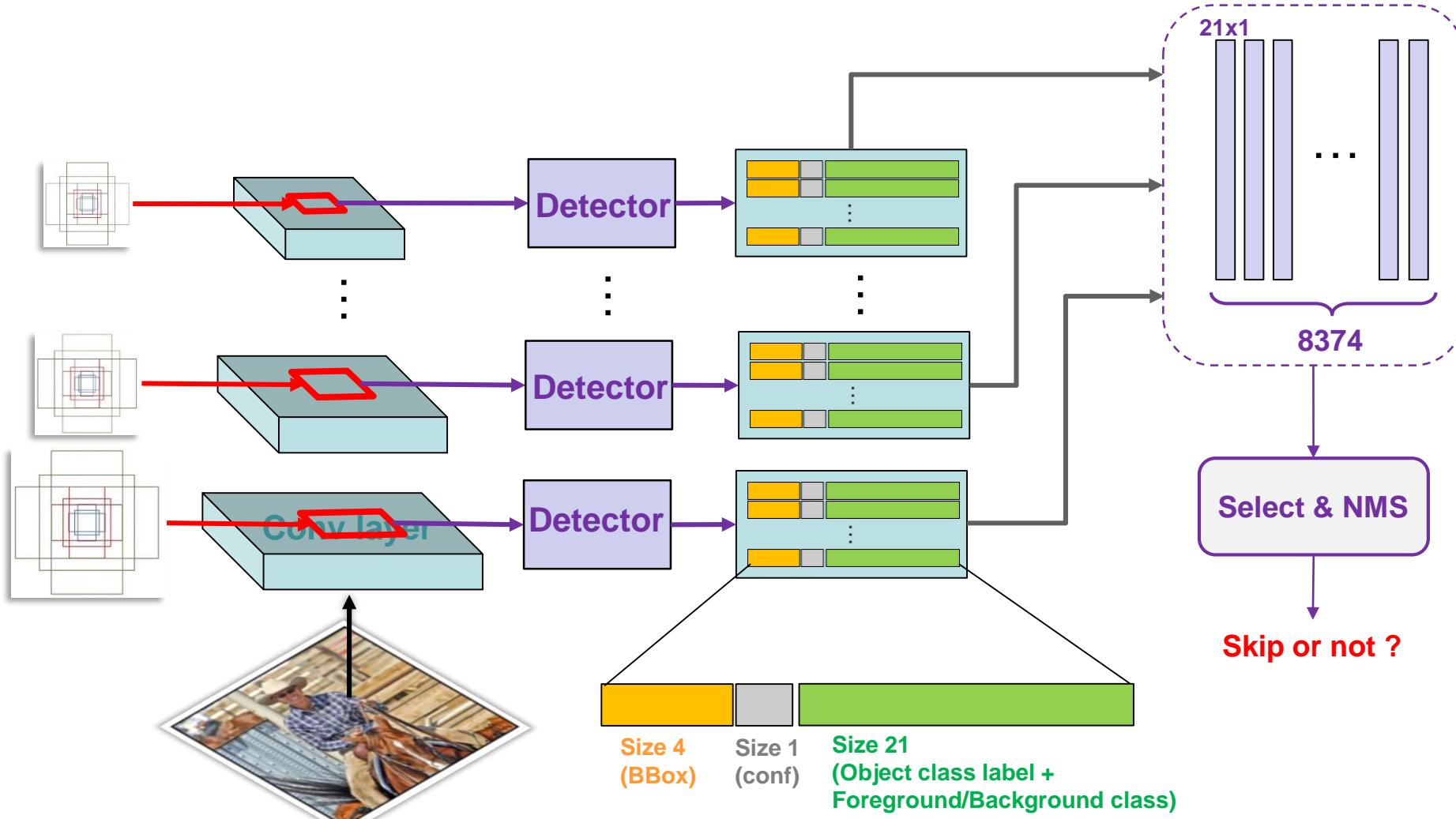
- Loss Function

$$\begin{aligned} \text{loss}_{YOLO} = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\ & + \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2 \end{aligned}$$

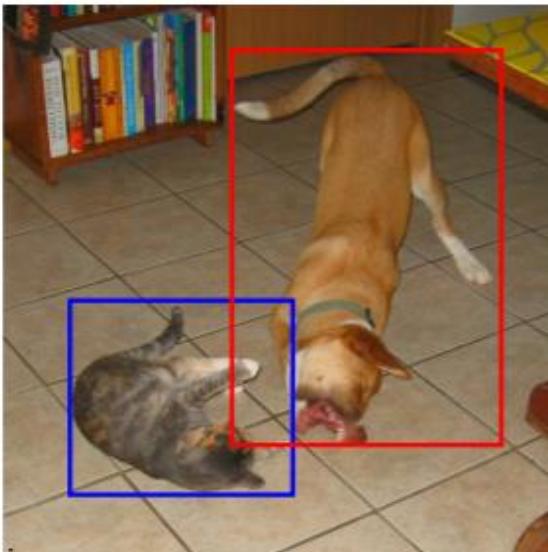
1_i^{obj} : 是物件有出現在 grid cell i 。

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

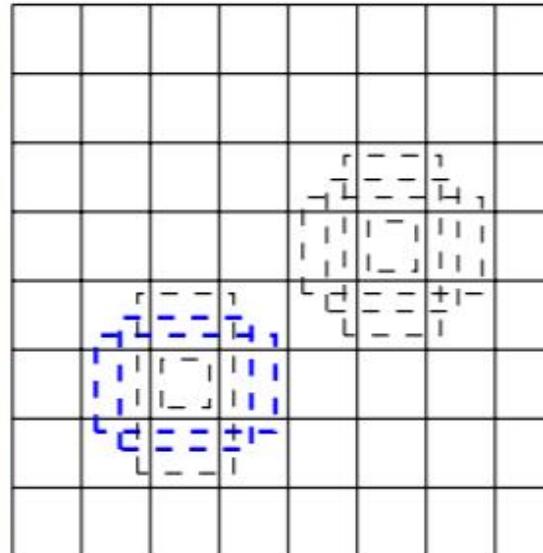
Single Shot MultiBox Detection (SSD)



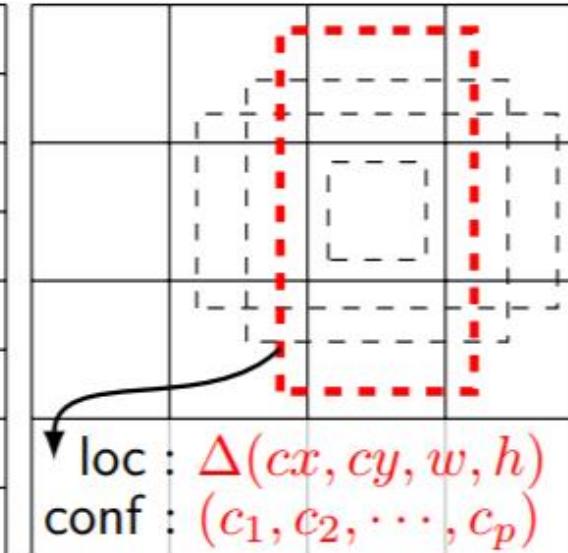
Single Shot MultiBox Detection (SSD)



(a) Image with GT boxes



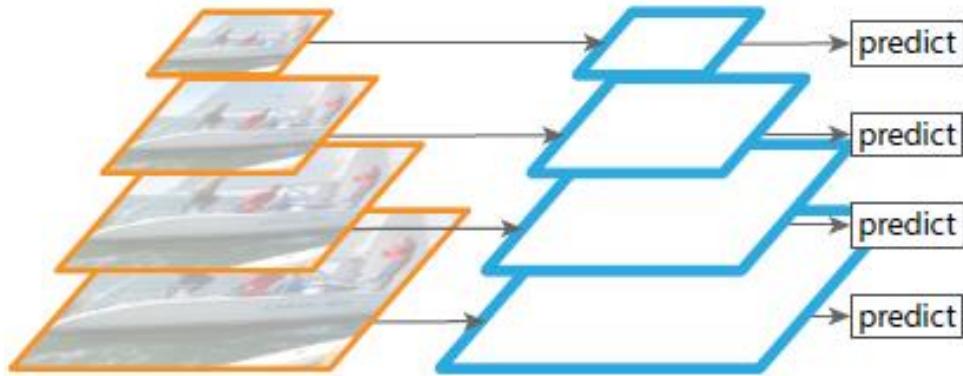
(b) 8×8 feature map



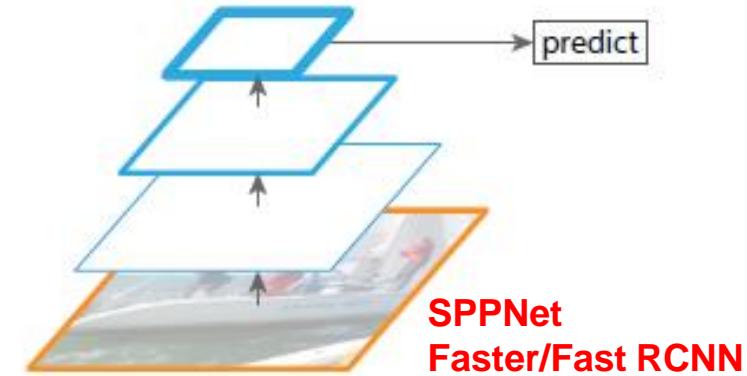
loc : $\Delta(cx, cy, w, h)$
conf : (c_1, c_2, \dots, c_p)

(c) 4×4 feature map

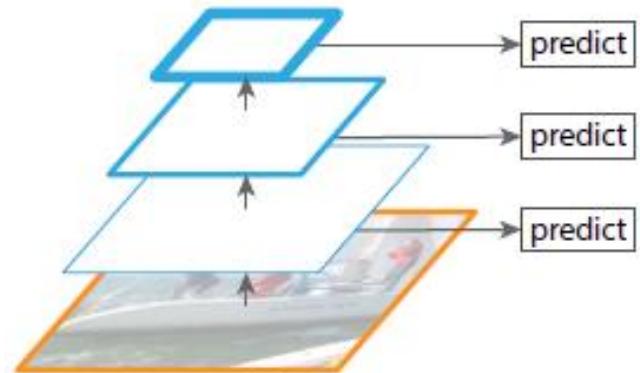
Feature Pyramid Network



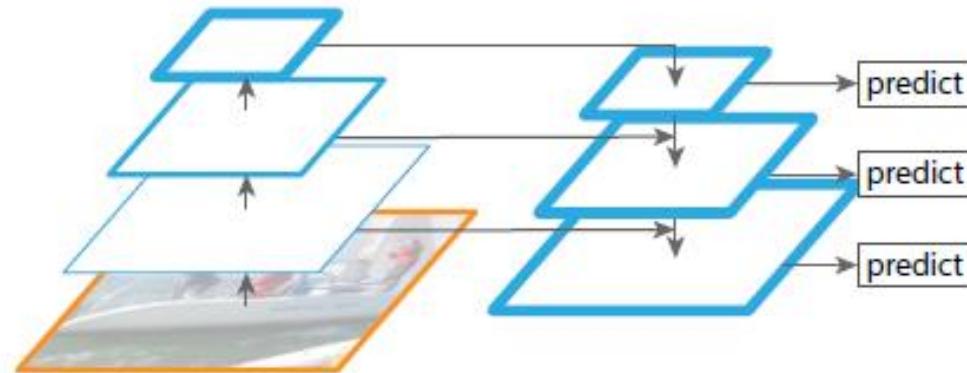
(a) Featurized image pyramid



(b) Single feature map



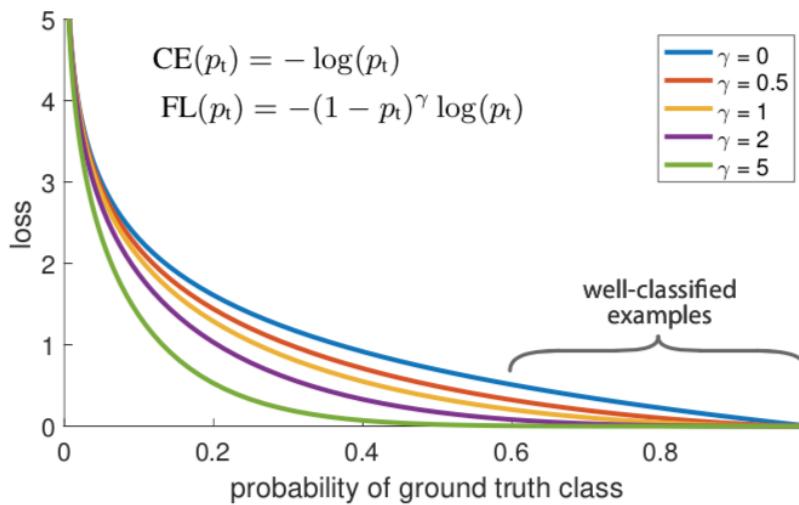
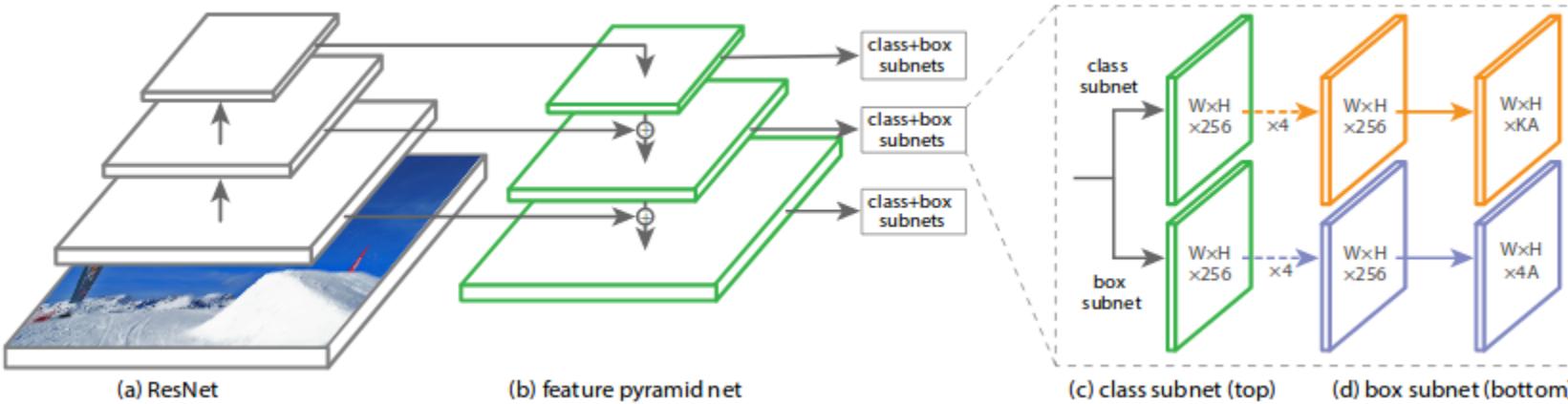
(c) Pyramidal feature hierarchy
SSD



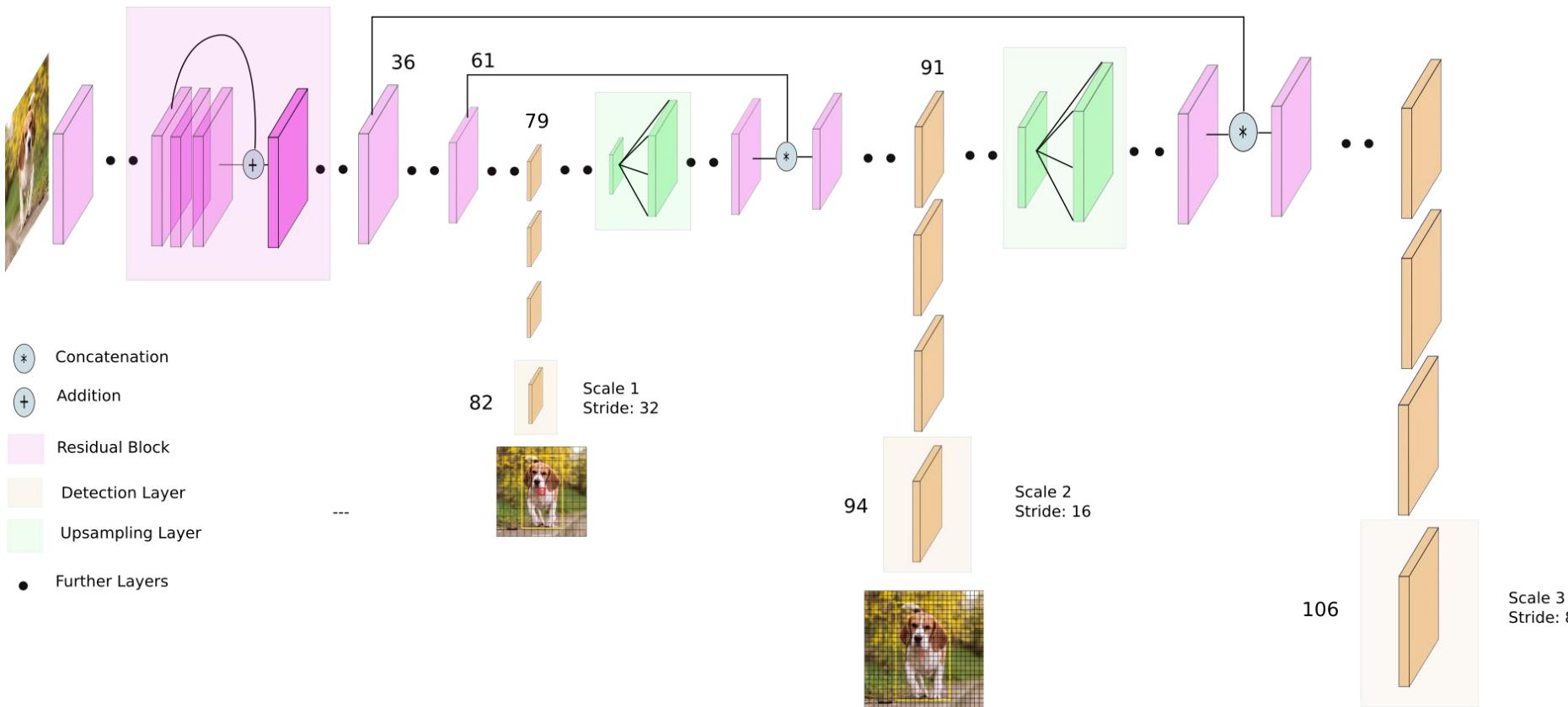
(d) Feature Pyramid Network

RetinaNet

- Focal Loss

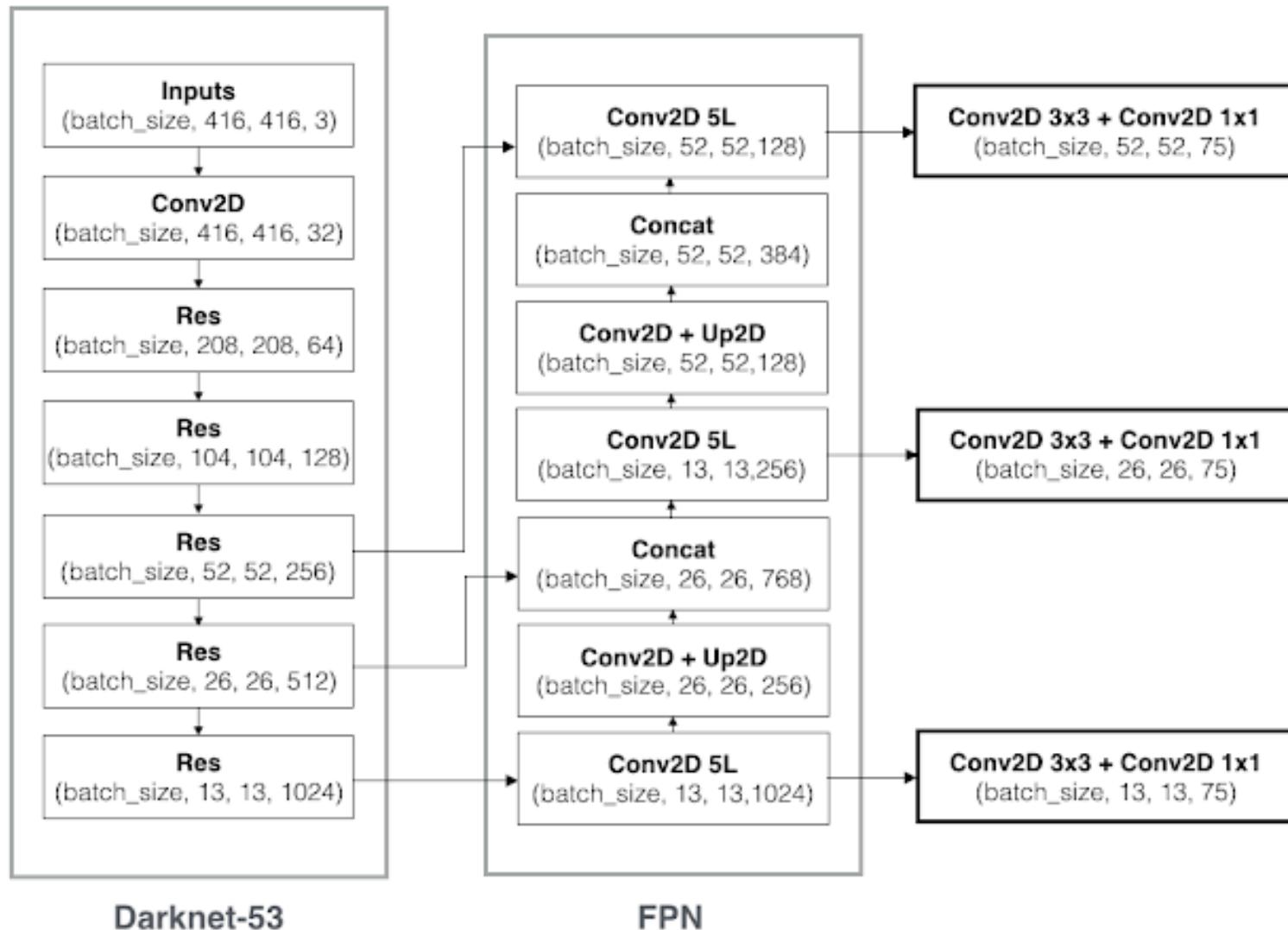


Yolo-v3



YOLO v3 network Architecture (RetinaNet + Anchor)

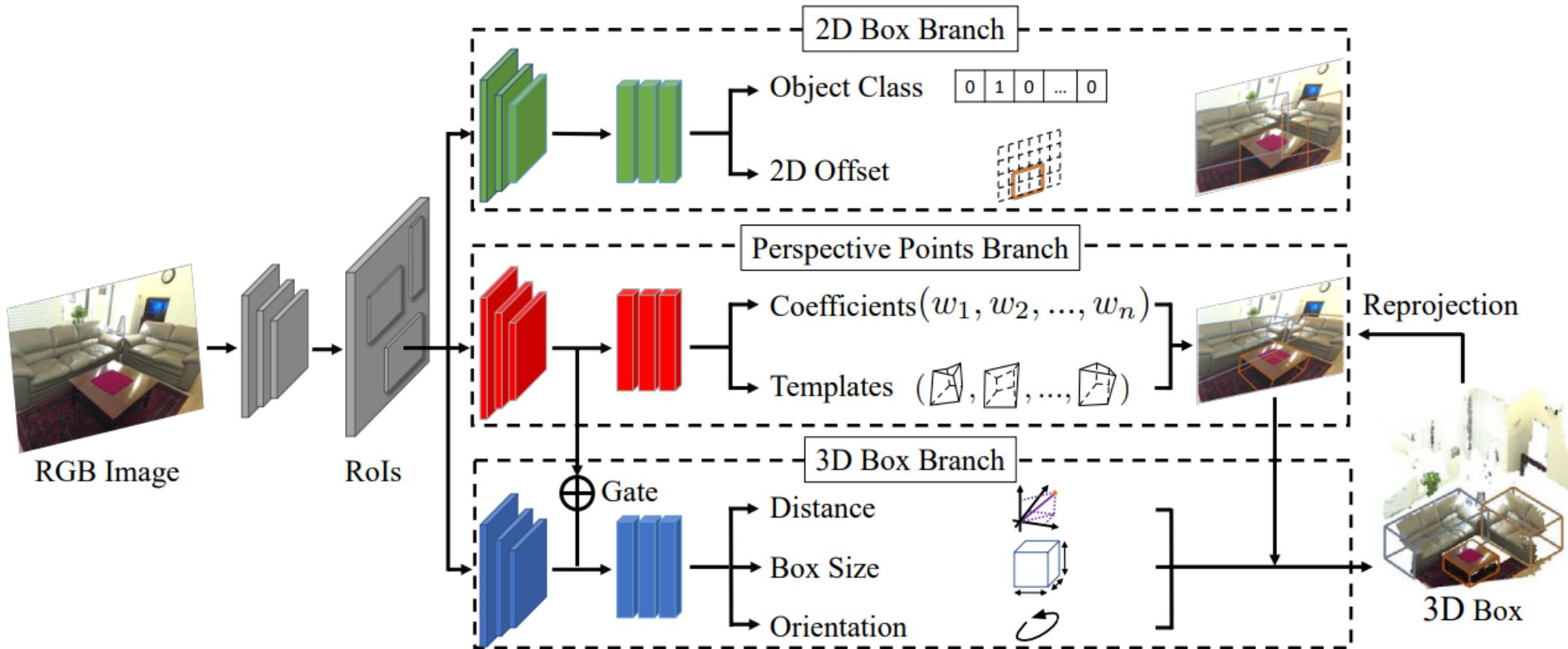
Yolo-v3



Object Detection with 3D Bounding Box

PerspectiveNet

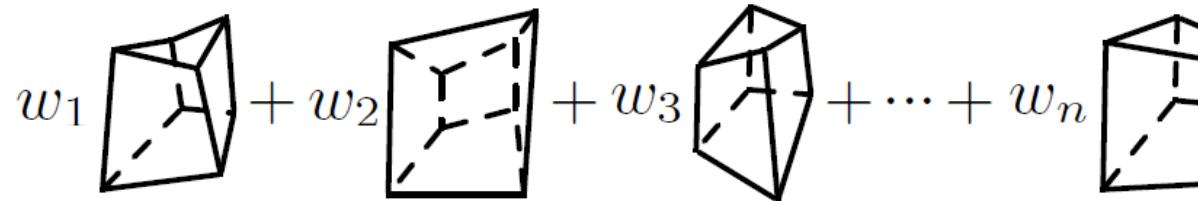
- PerspectiveNet: 3D Object Detection from a Single RGB Image via Perspective Points (NeurIPS 2019)



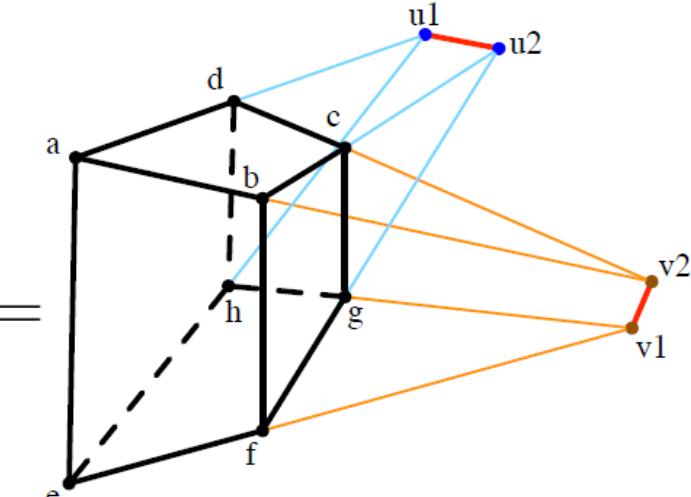
PerspectiveNet

- Template Based Perspective Point Regression

C(class) * K(templates) * 2(x,y) * 9(3d corner + center)



(a) Mixture of templates



(b) Perspective Loss

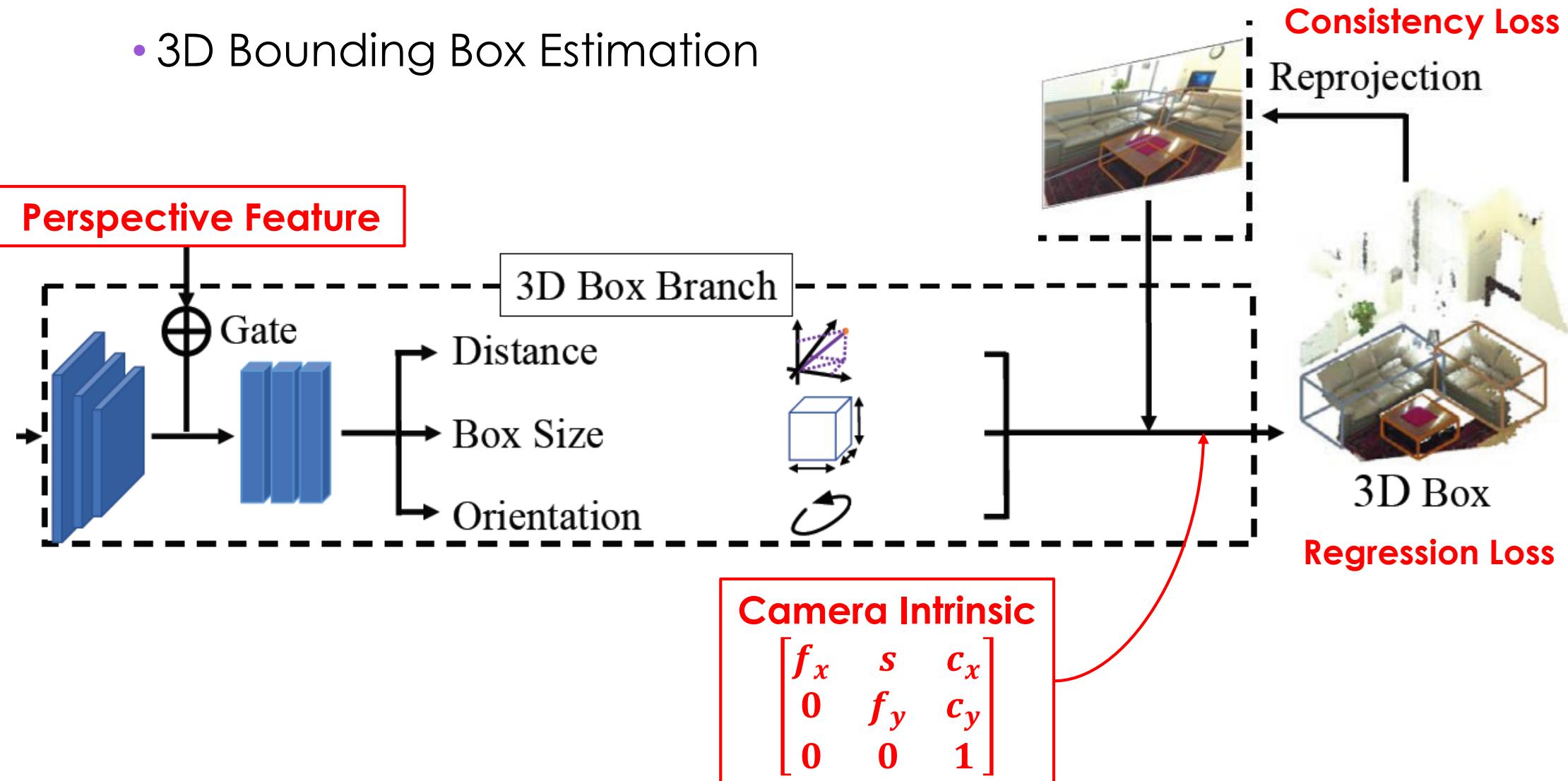
$$\hat{P}_i = \sum_{k=1}^K w_{ik} T_{ik}, \quad \forall i = 1, \dots, C.$$



(b) 2D Perspective Points

PerspectiveNet

- 3D Bounding Box Estimation

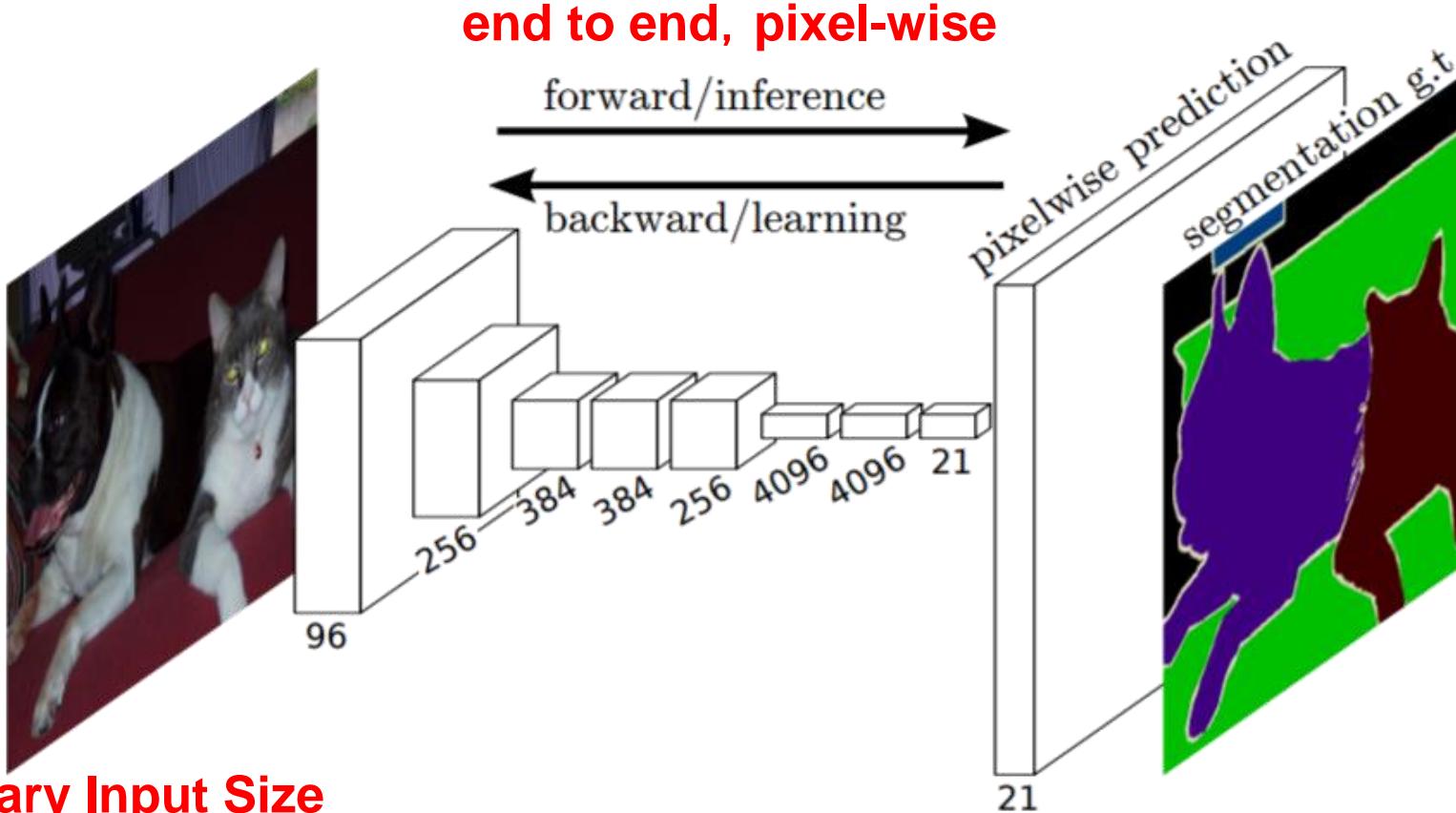


Experimental Results



Segmentation

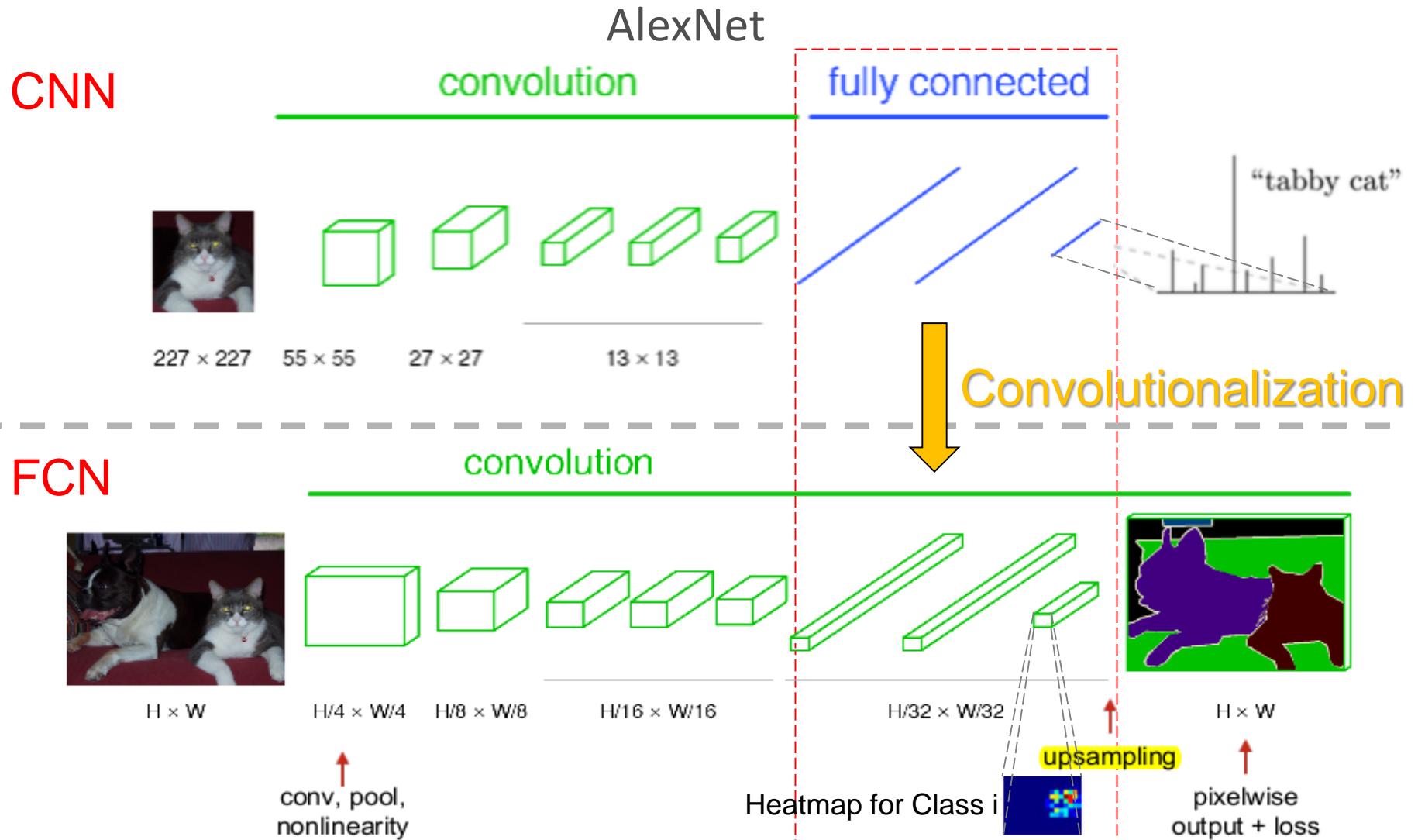
Fully Convolution Network



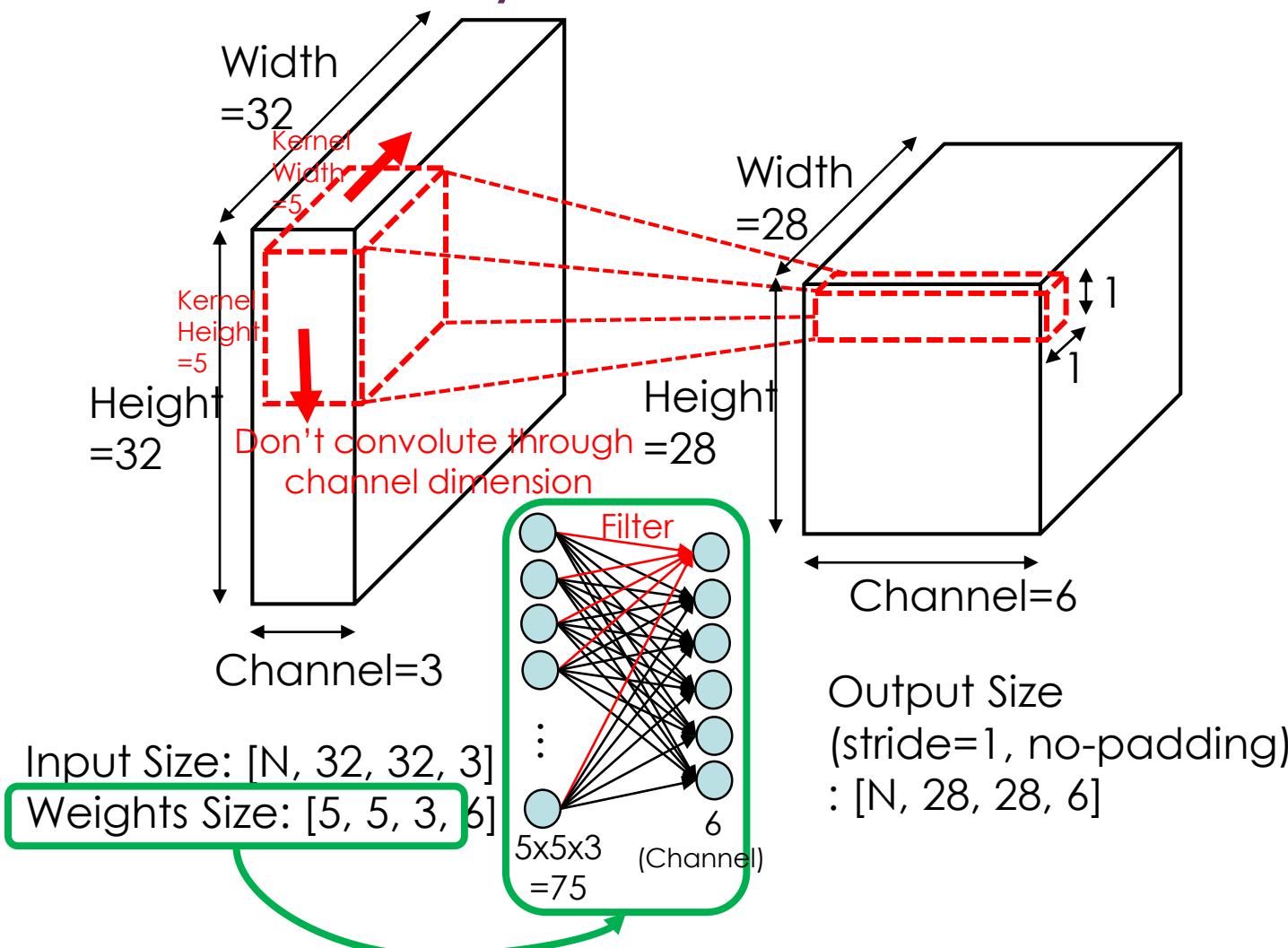
**Arbitrary Input Size
(Similar Object Scale)**

[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.

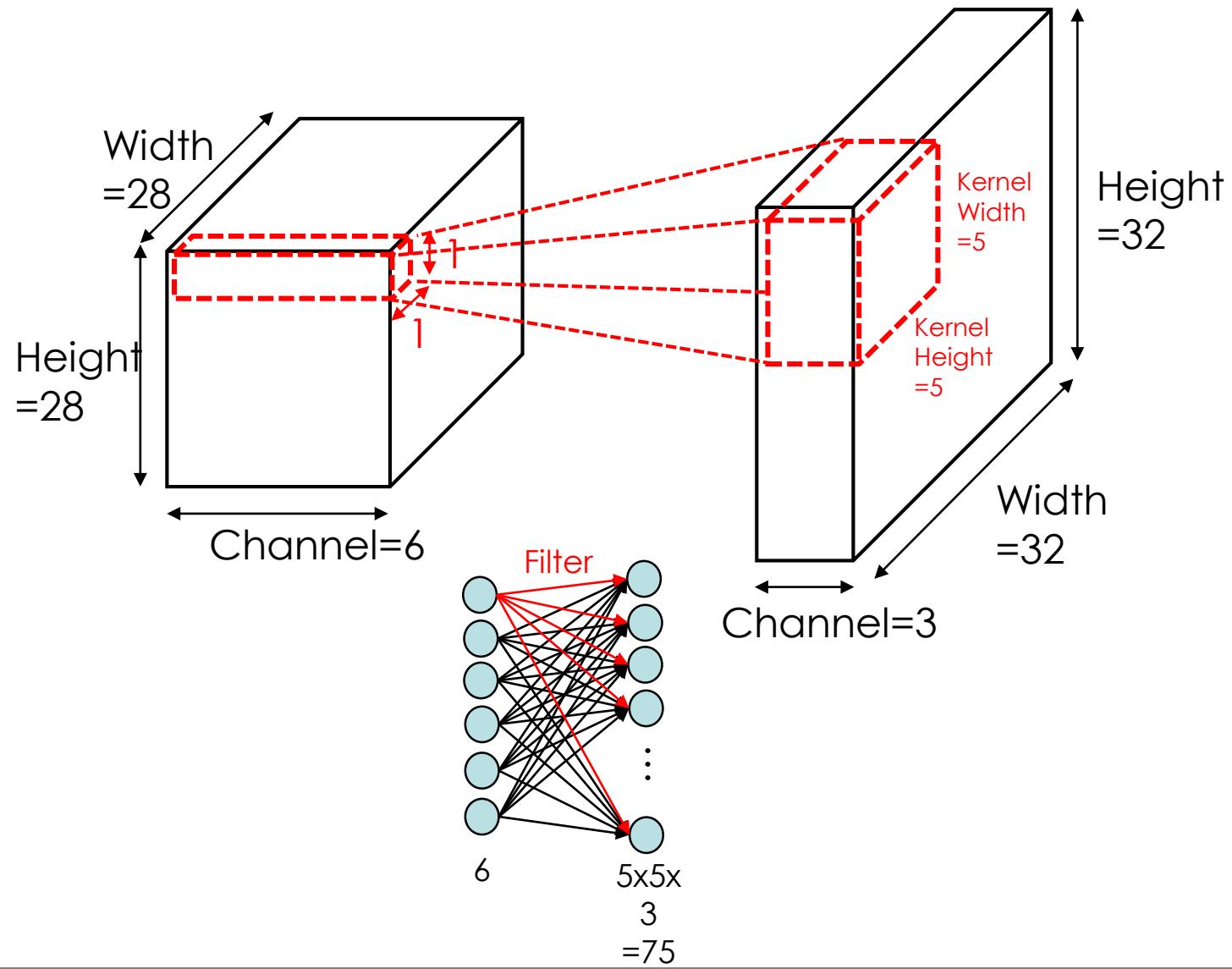
CNN vs FCN



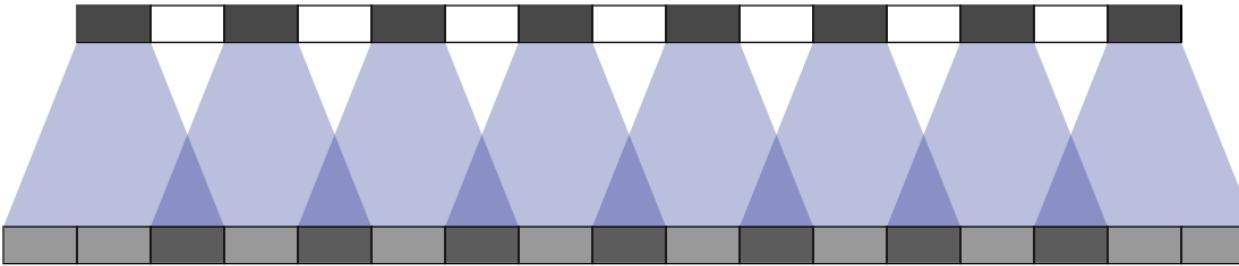
Convolutional Layer



De-convolutional Layer

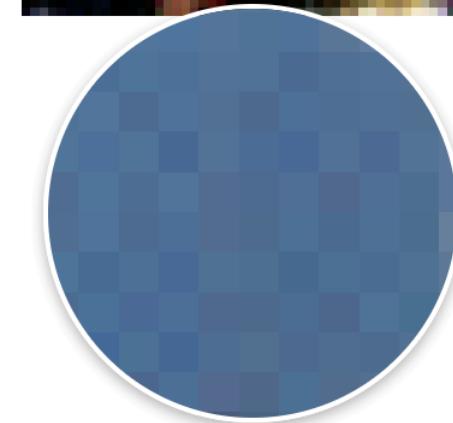
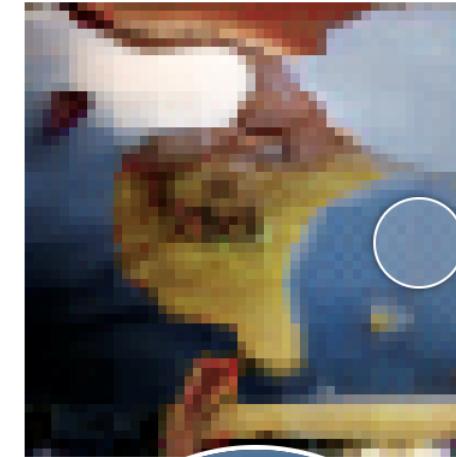


Checkerboard Artifacts

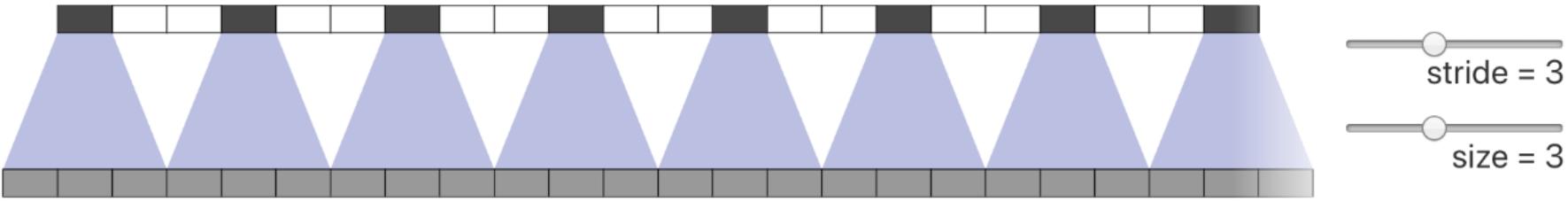


stride = 2

size = 3



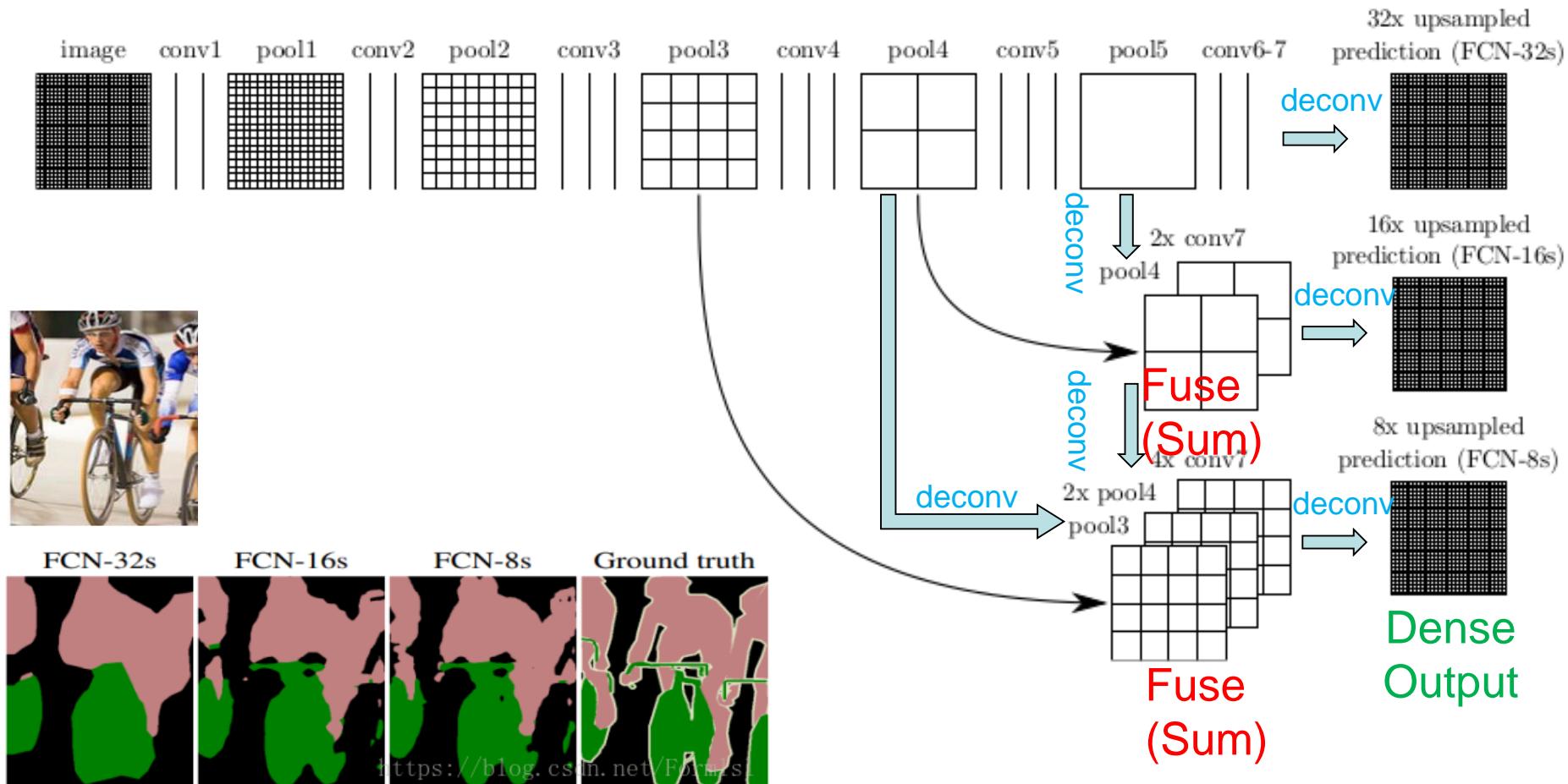
Checkerboard Artifacts



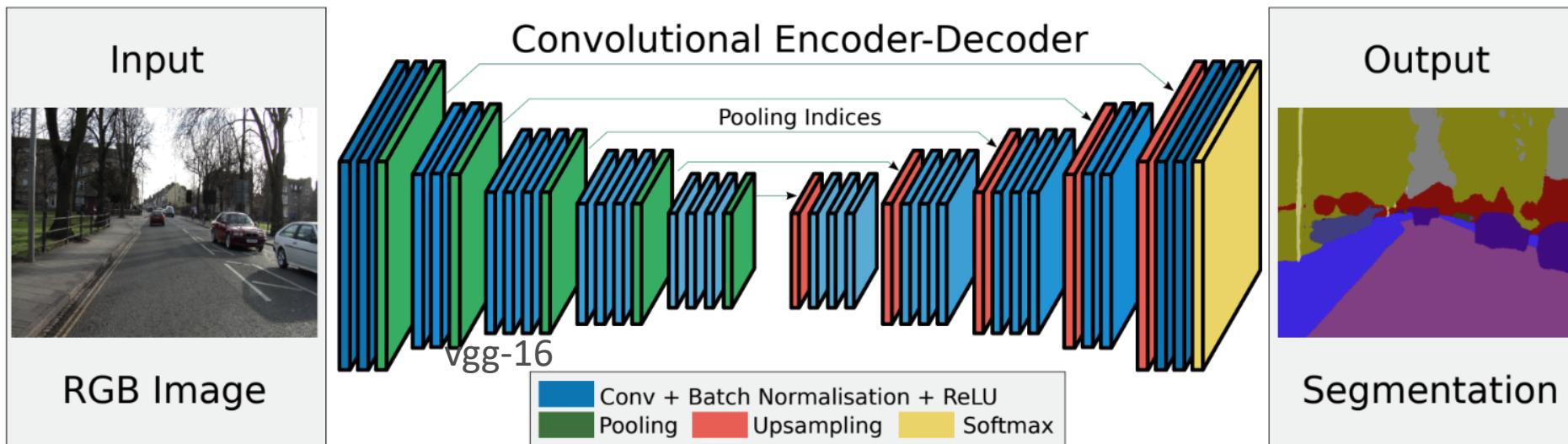
Solution 1:
Adjust stride and filter size to non-overlapping deconvolution

Solution 2:
Bilinear/Max Upsampling
+ 1 stride convolution

Upsample in FCN



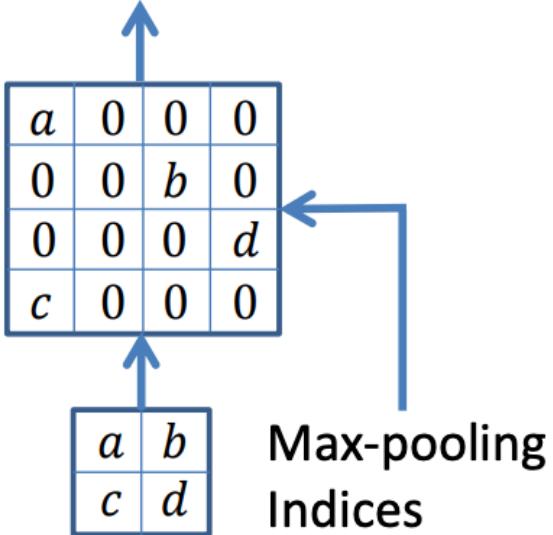
SegNet



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

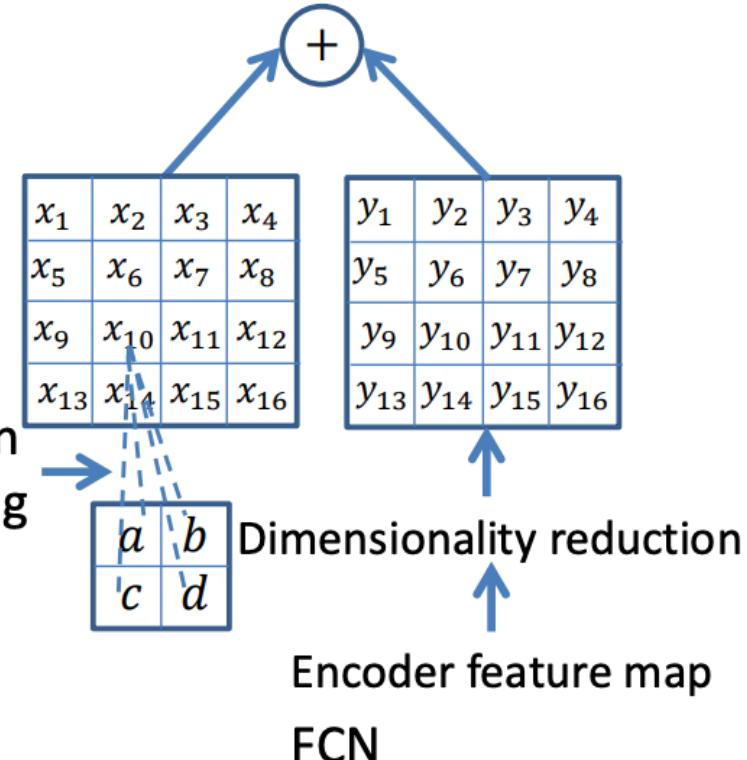
SegNet vs FCN

Convolution with trainable decoder filters



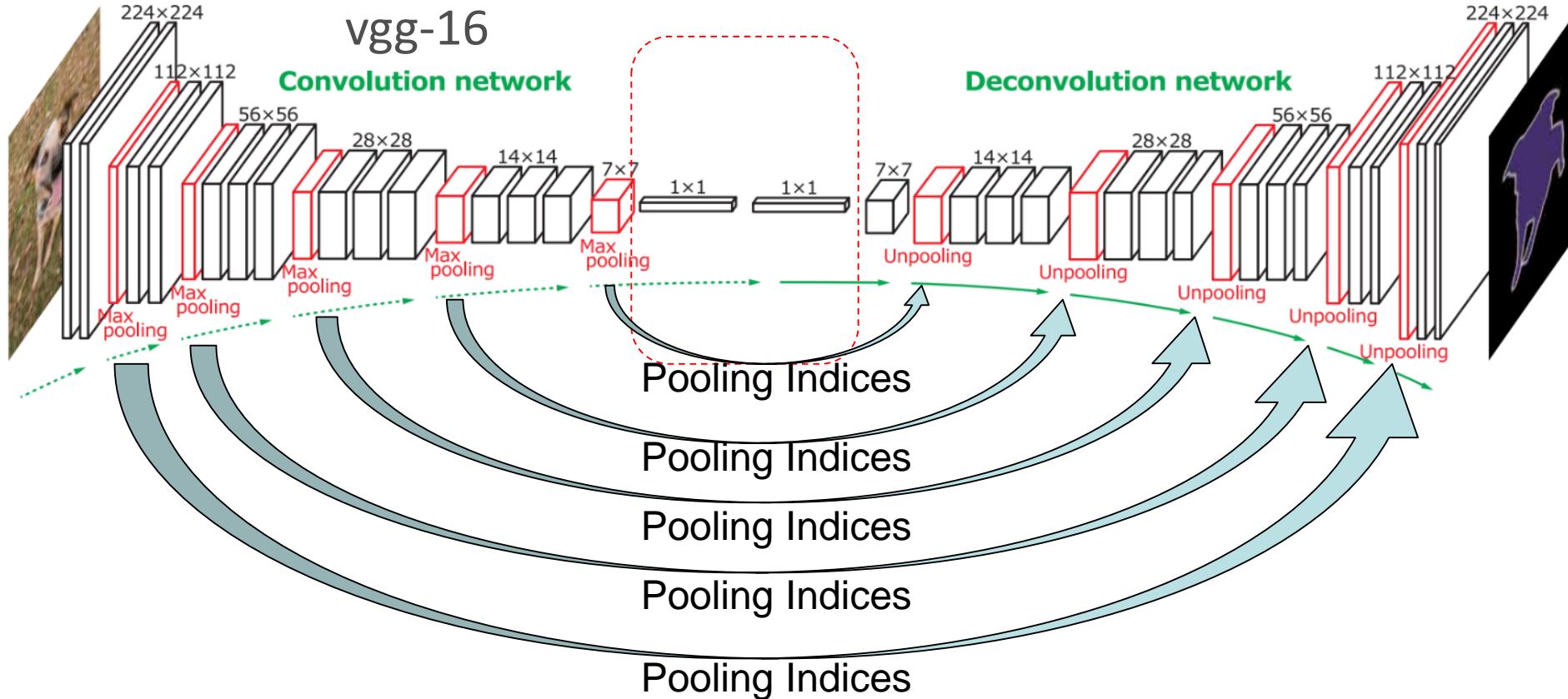
SegNet

Deconvolution
for upsampling



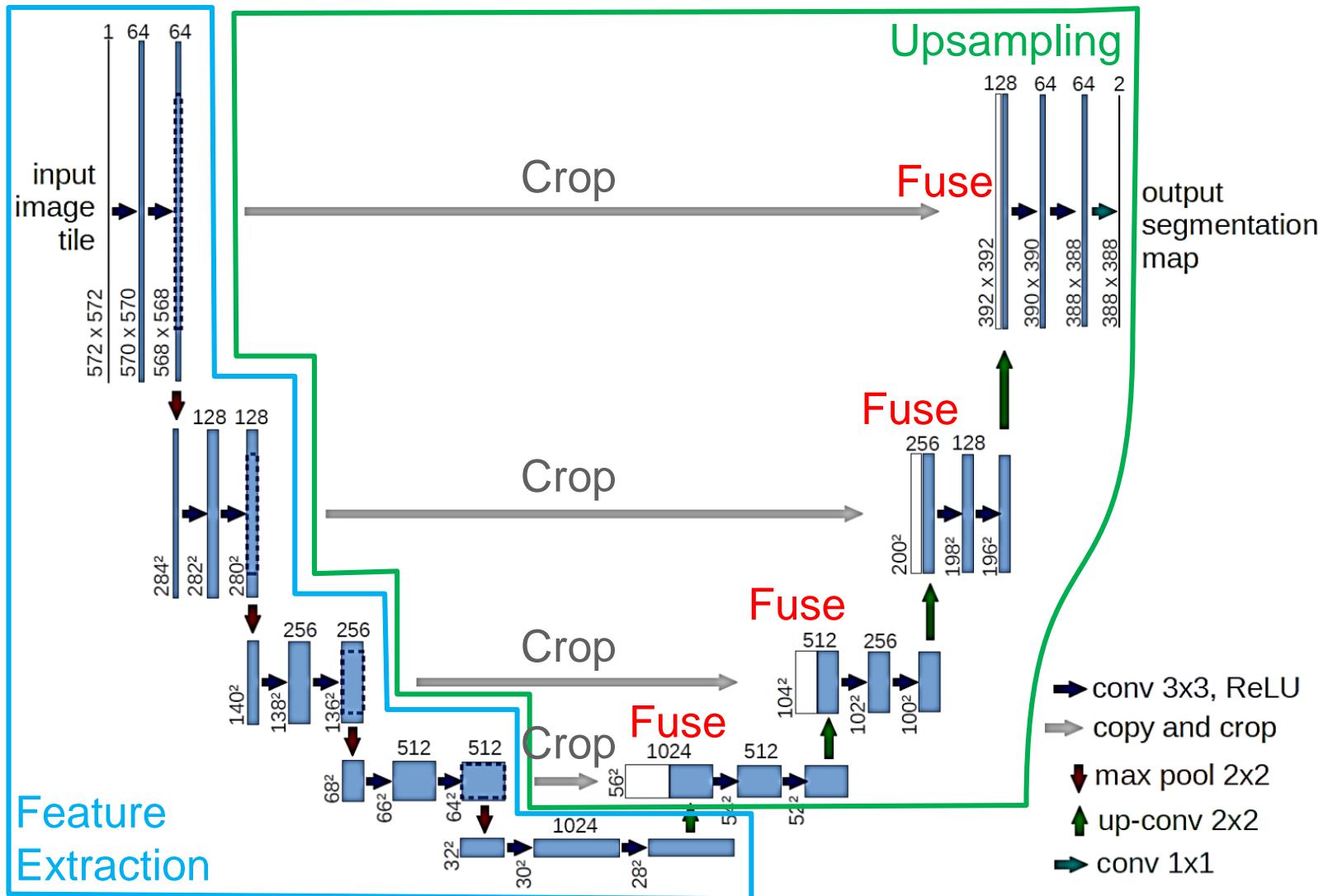
FCN

DeconvNet



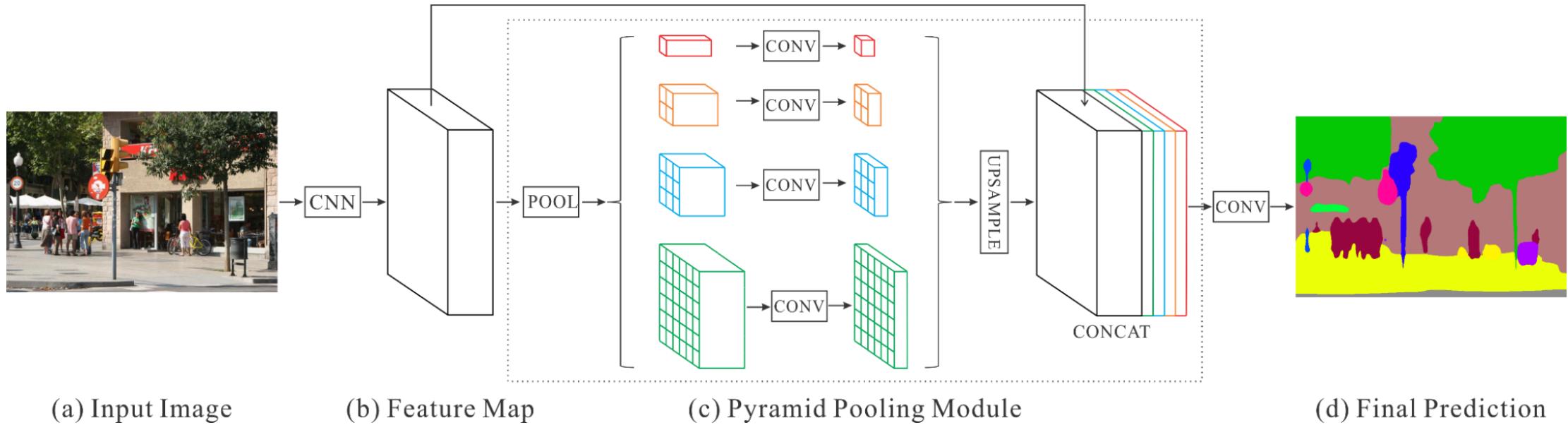
[Ref] Noh, Hyenwoo, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." Proceedings of the IEEE international conference on computer vision. 2015.

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.

Pyramid Scene Parsing Network(PSPNet)



DeepLab v3

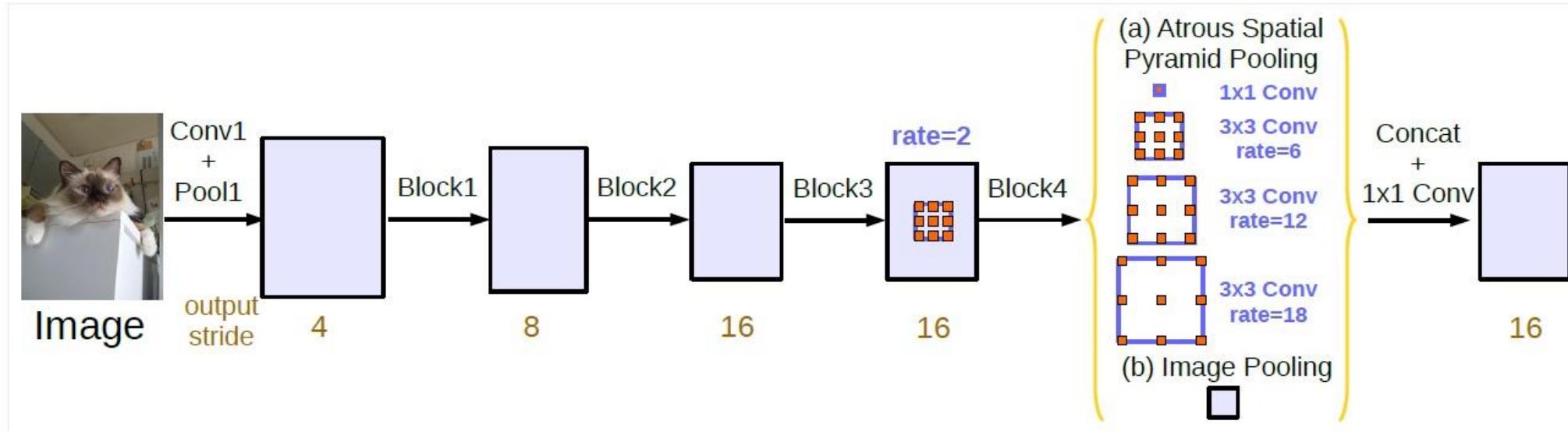
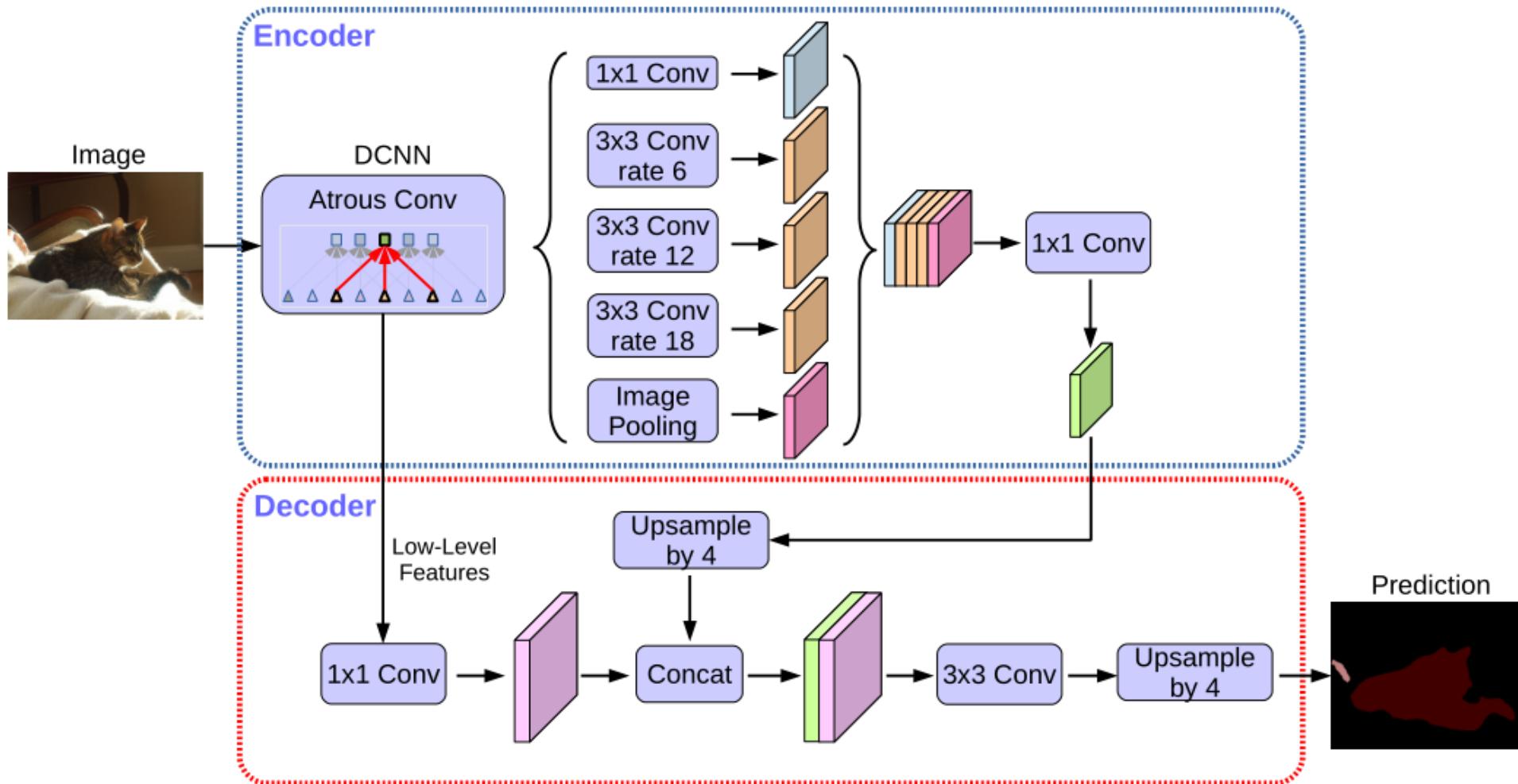


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

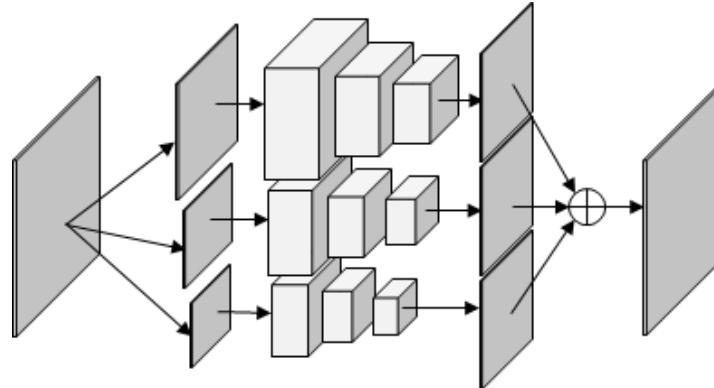
DeepLab v3+



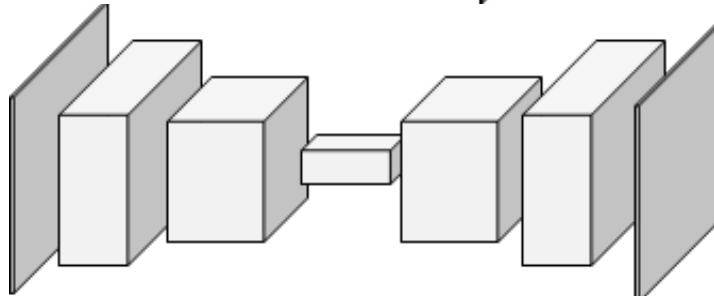
Different Segmentation Structures

- Capture the multiscale information of the image

Image Pyramid Structure

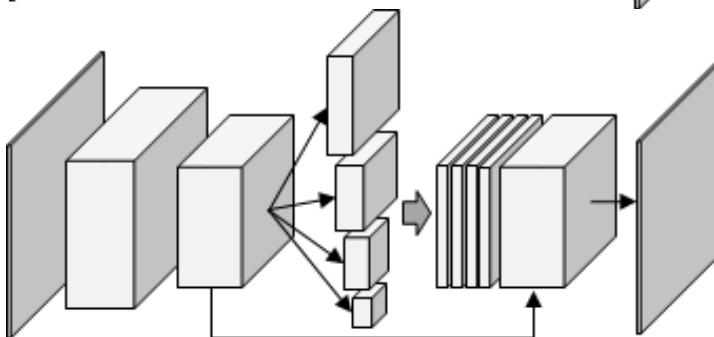


Encoder-Decoder Structure



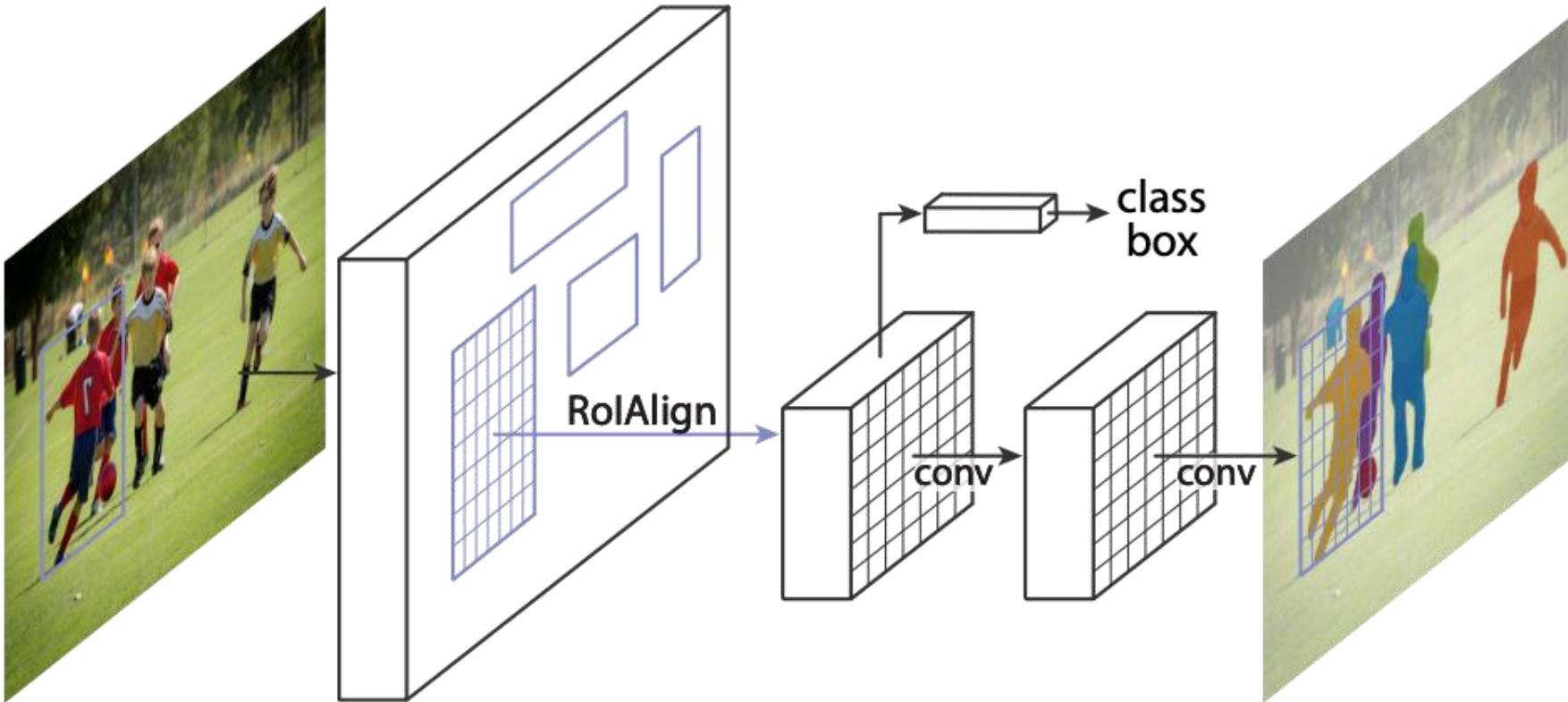
**FCN
U-Net**

Pyramid Pooling Structure



**PSPNet
DeepLab**

Mask R-CNN

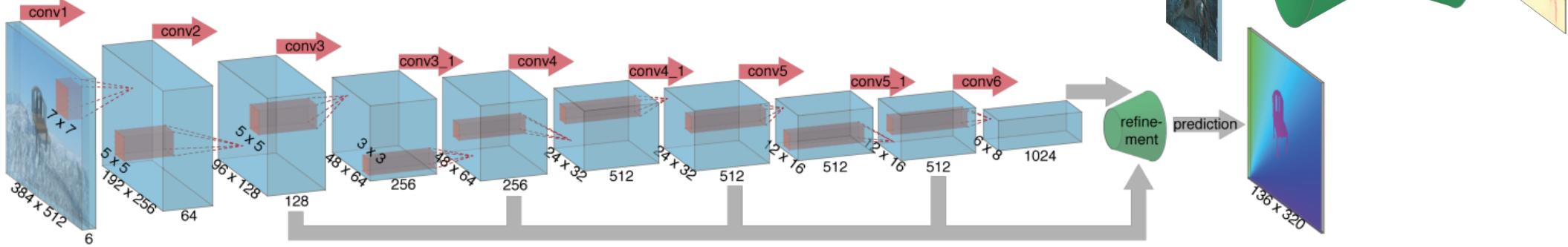


[Ref] He, Kaiming, et al. "Mask r-cnn." Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017.

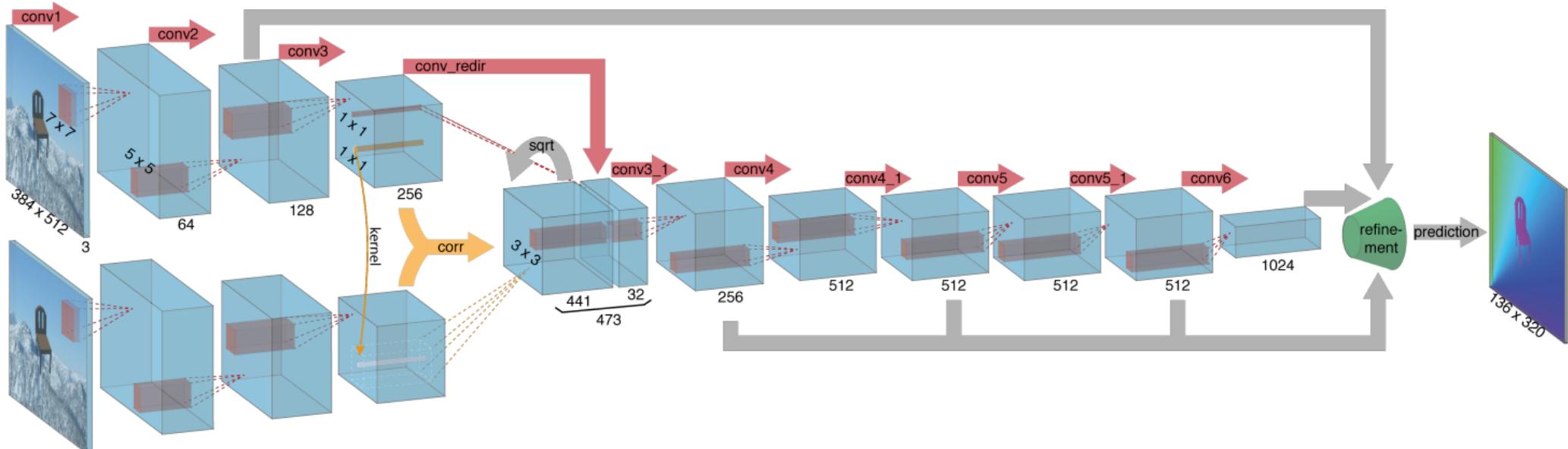
Depth/Pose Estimation

FlowNet

FlowNetSimple

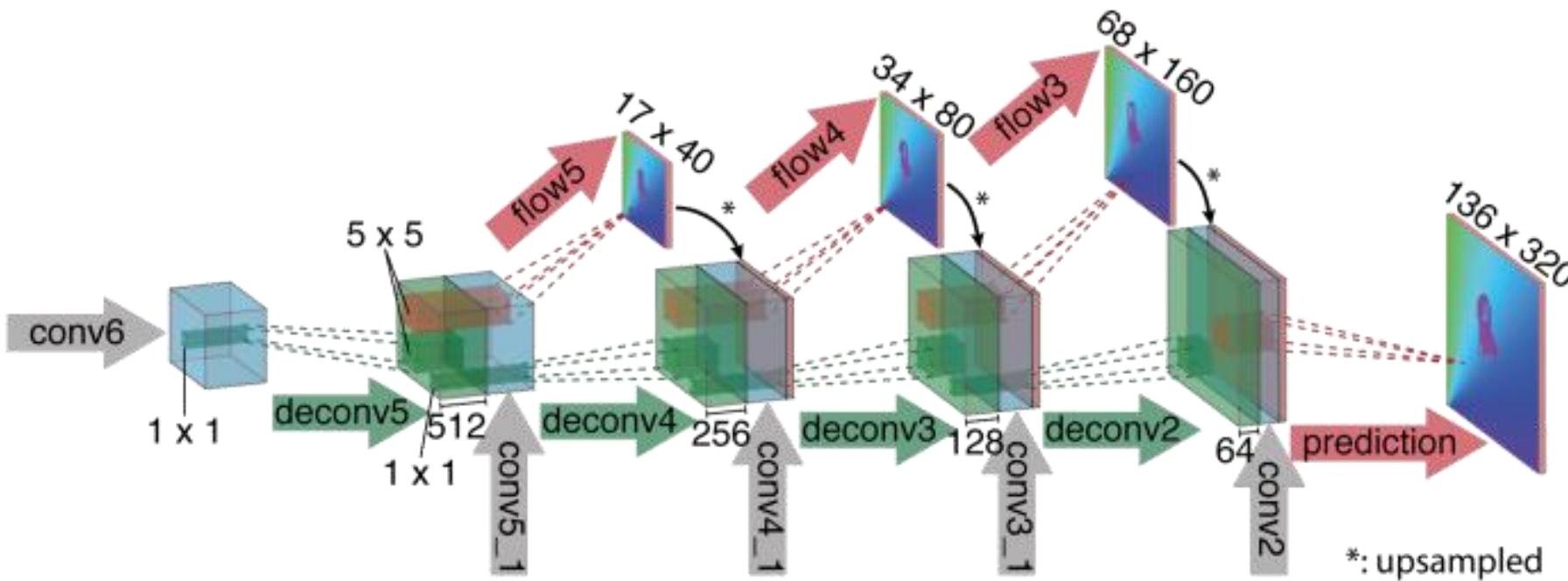


FlowNetCorr



FlowNet

- Refinement Decoder

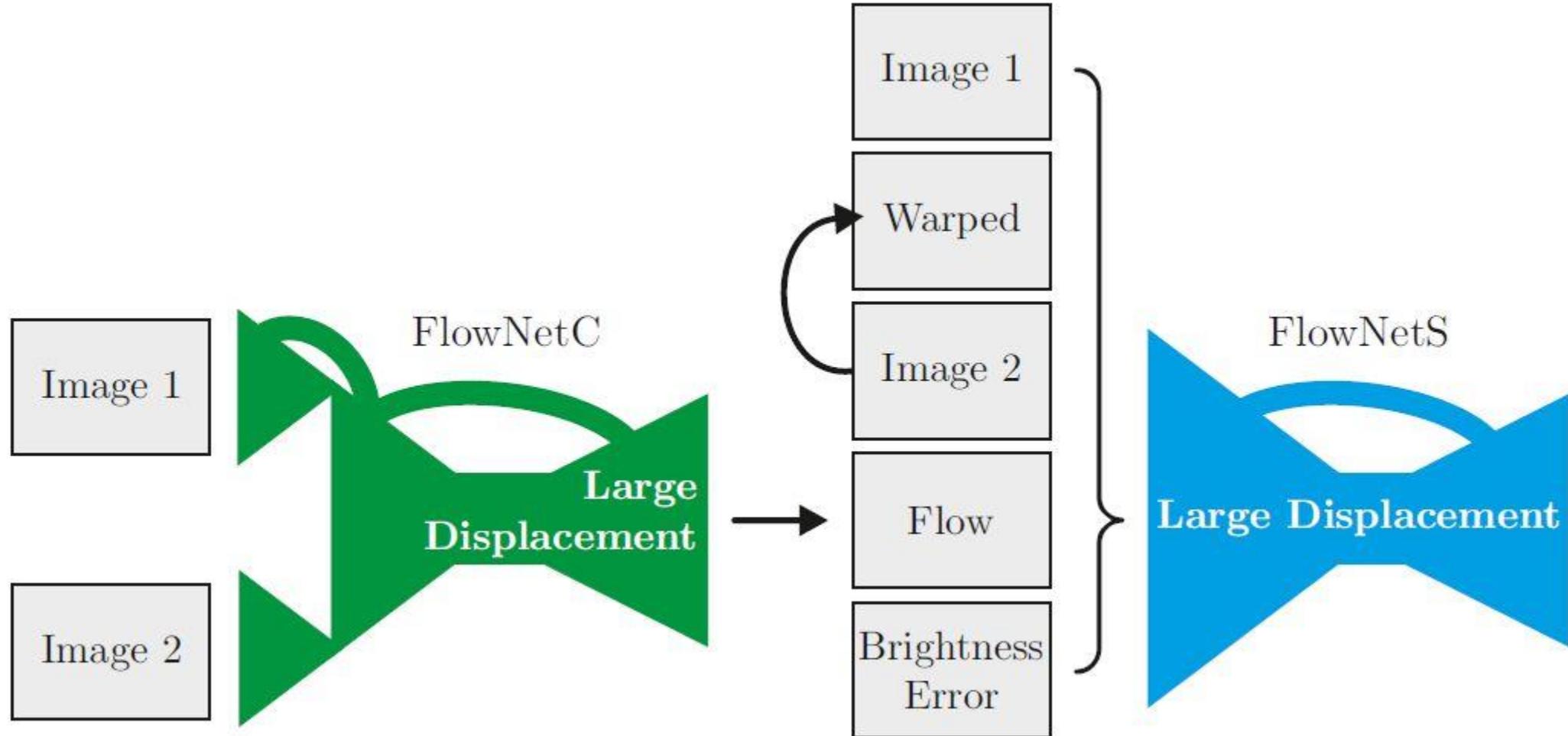


FlowNet

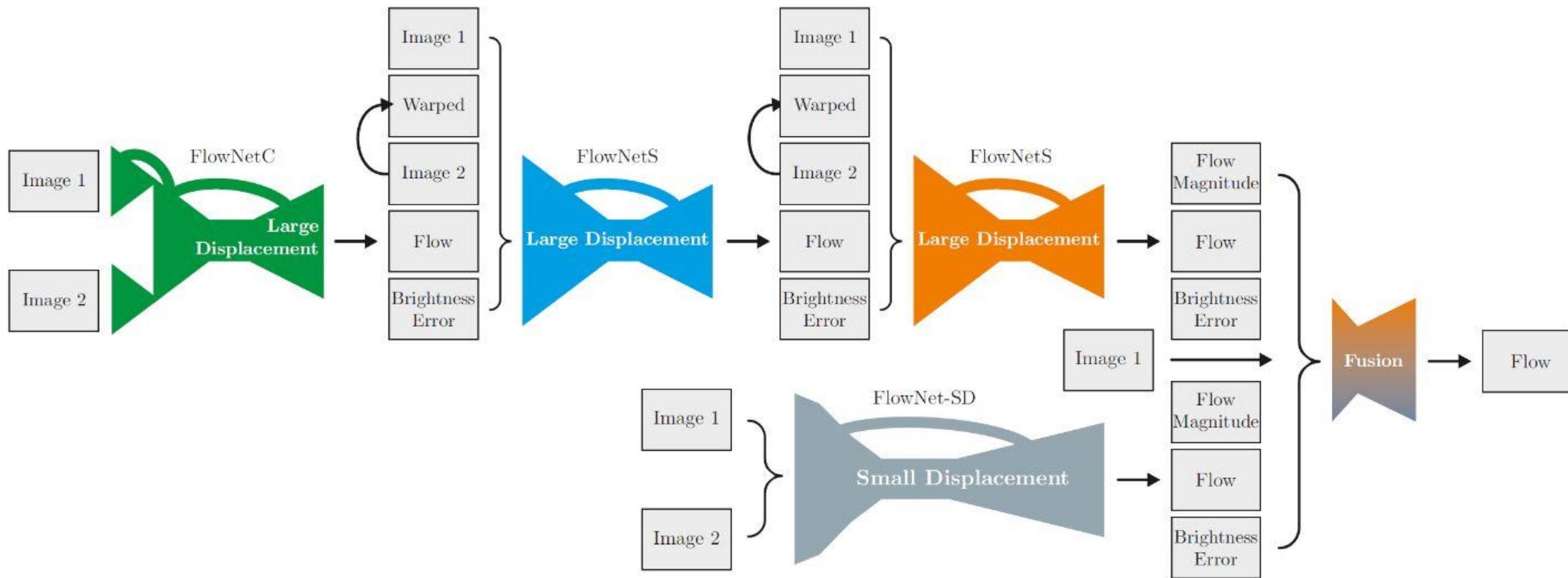
- Dataset Generation



FlowNet 2

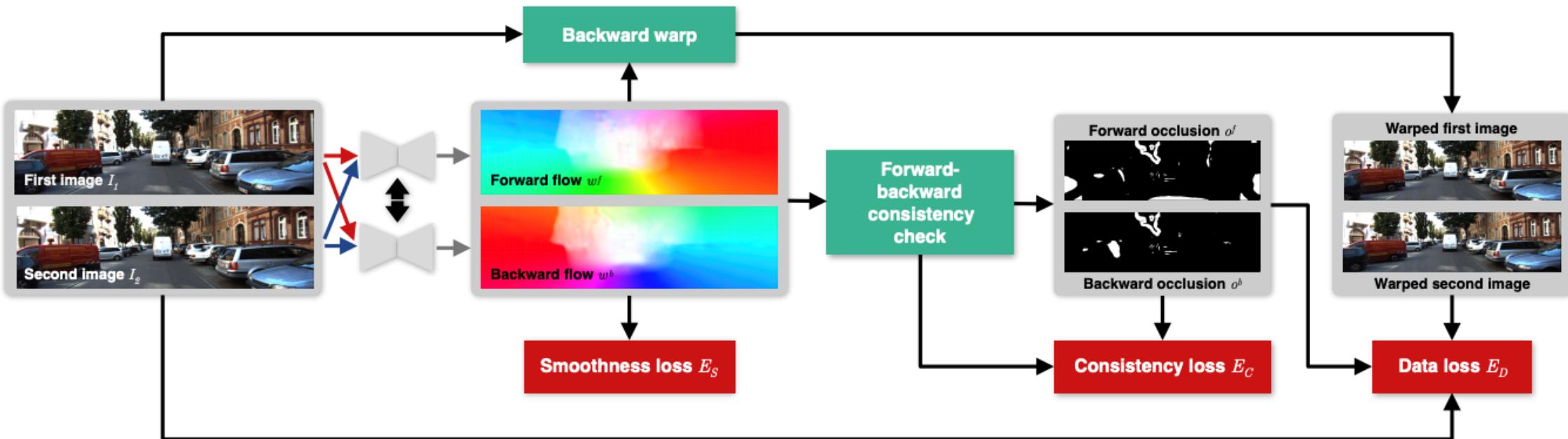


FlowNet 2

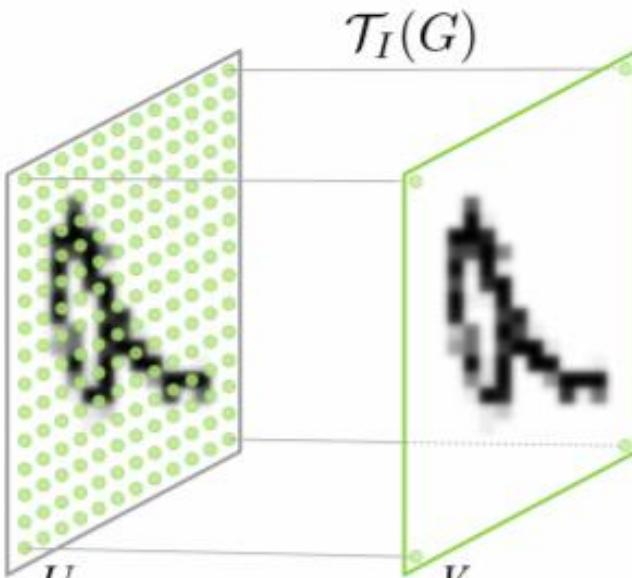


UnFlow

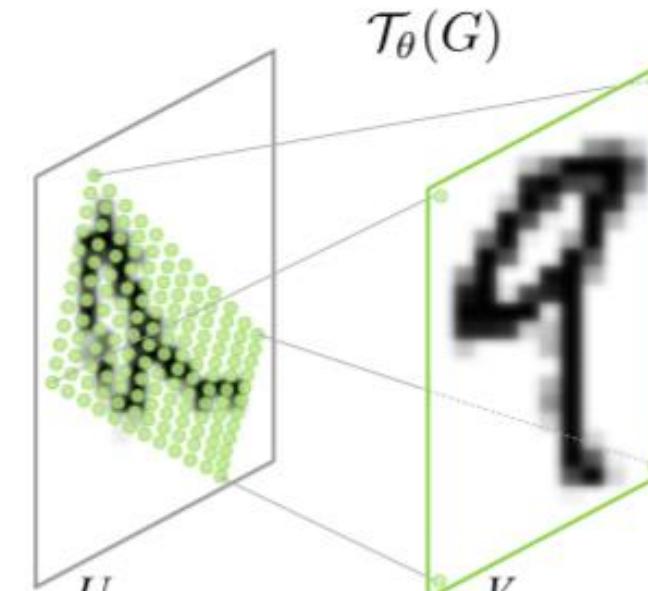
- Bidirectional training of FlowNet with a comprehensive unsupervised loss.



Differentiable Image Sampling



(a)

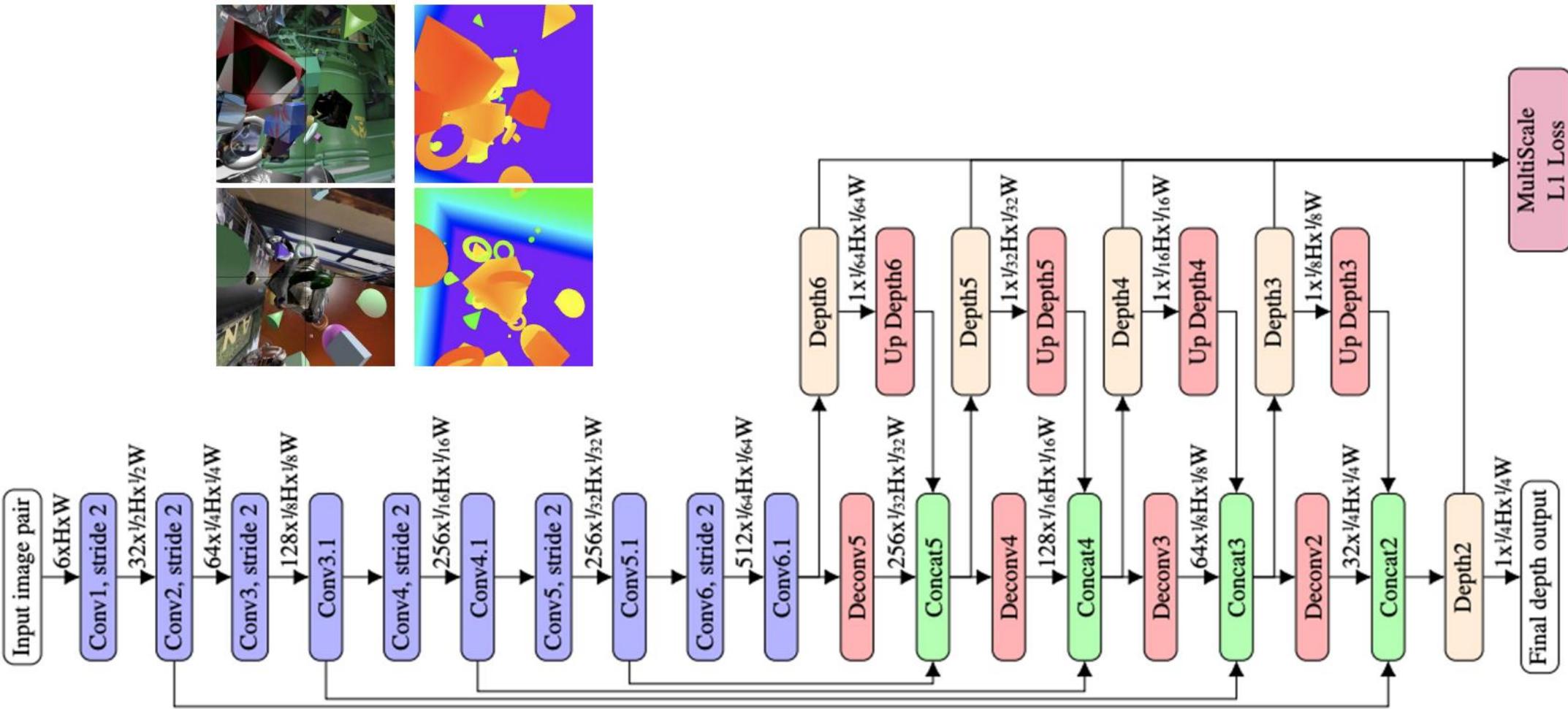


(b)

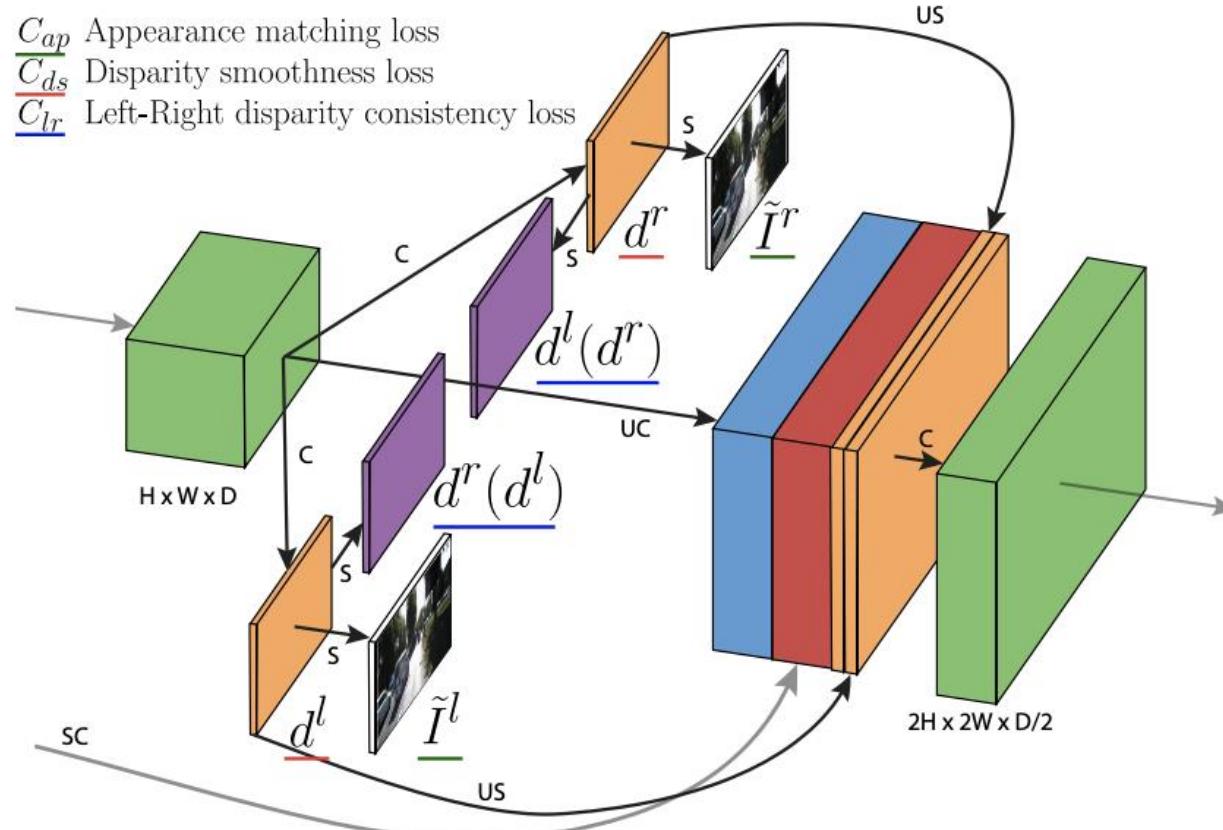
$$V_i^c = \sum_{n=1}^H \sum_{m=1}^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) \quad \forall i \in [1 \dots H'W'] \quad \forall c \in [1 \dots C]$$

Target $\sum_n \sum_m$ Source

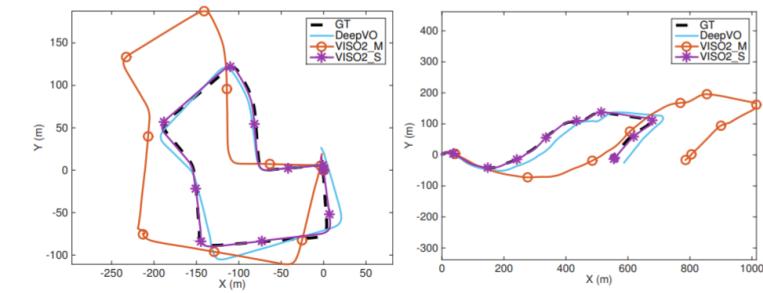
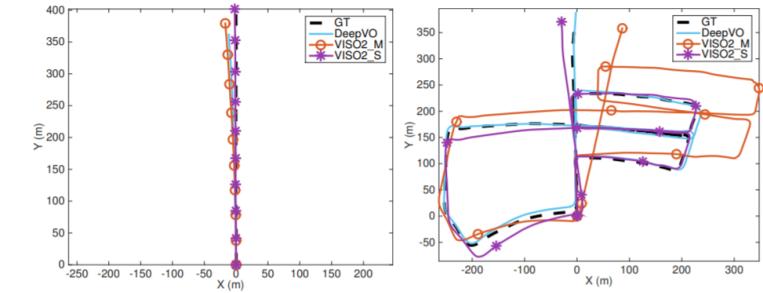
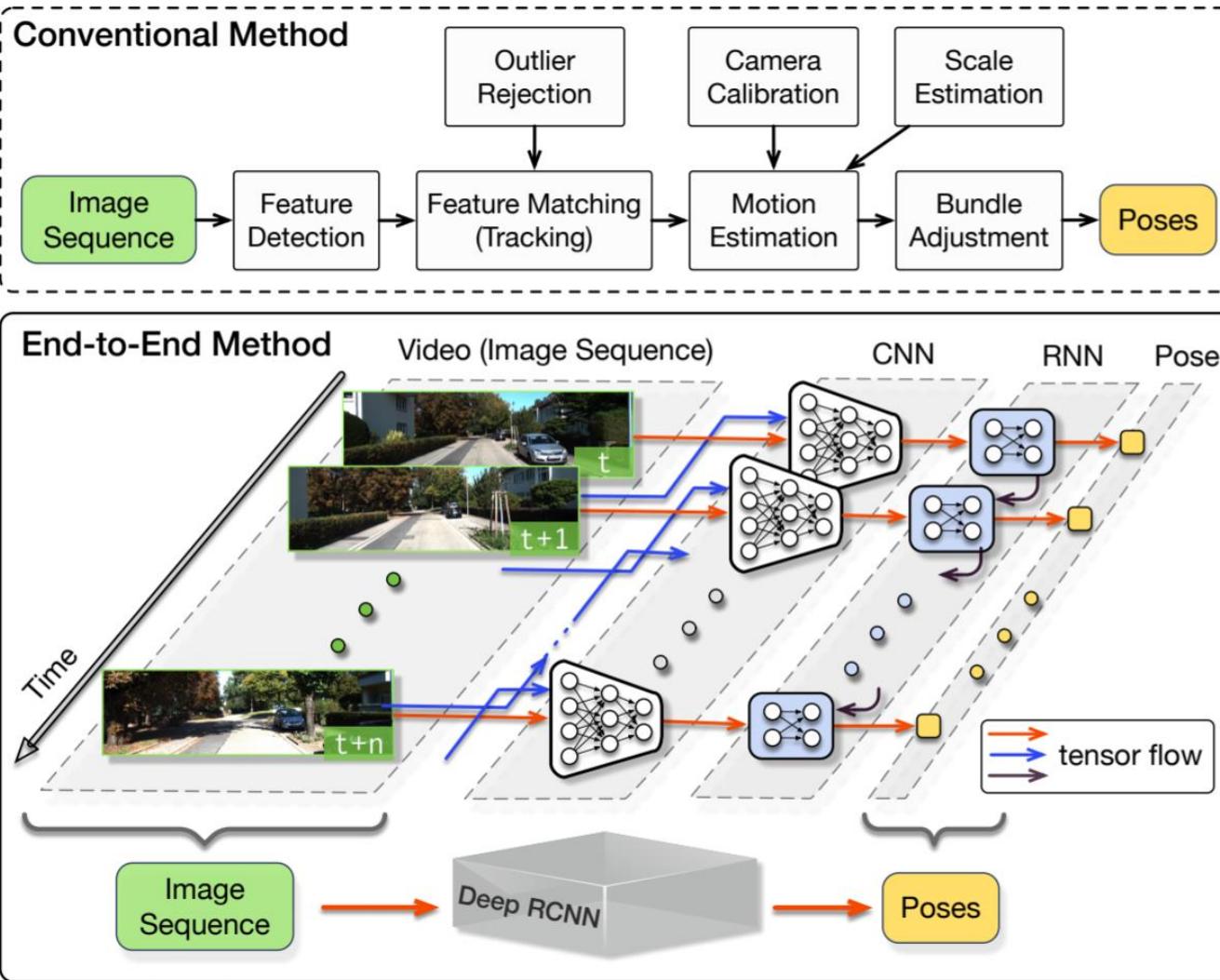
End-to-end depth from motion with stabilized monocular videos



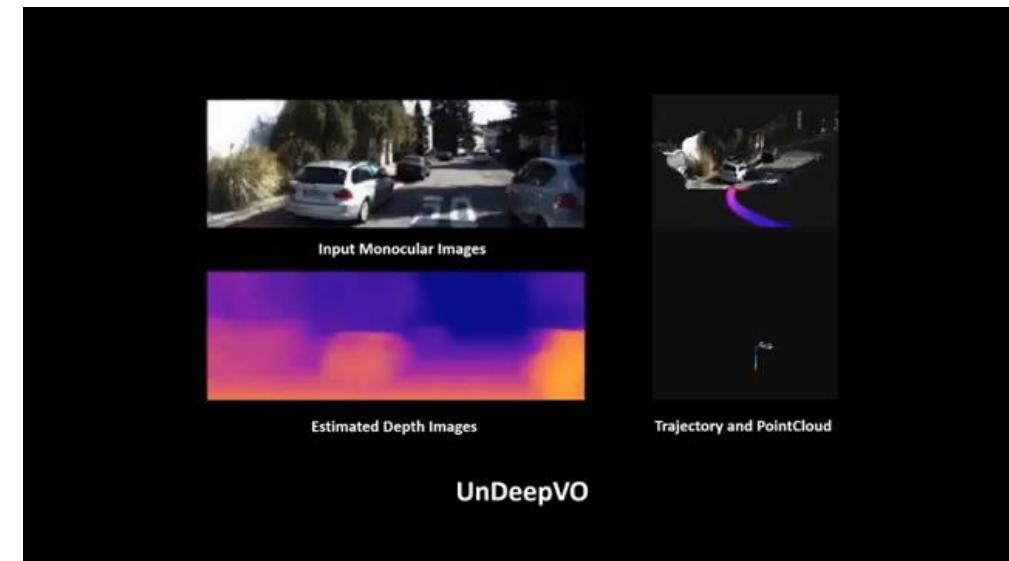
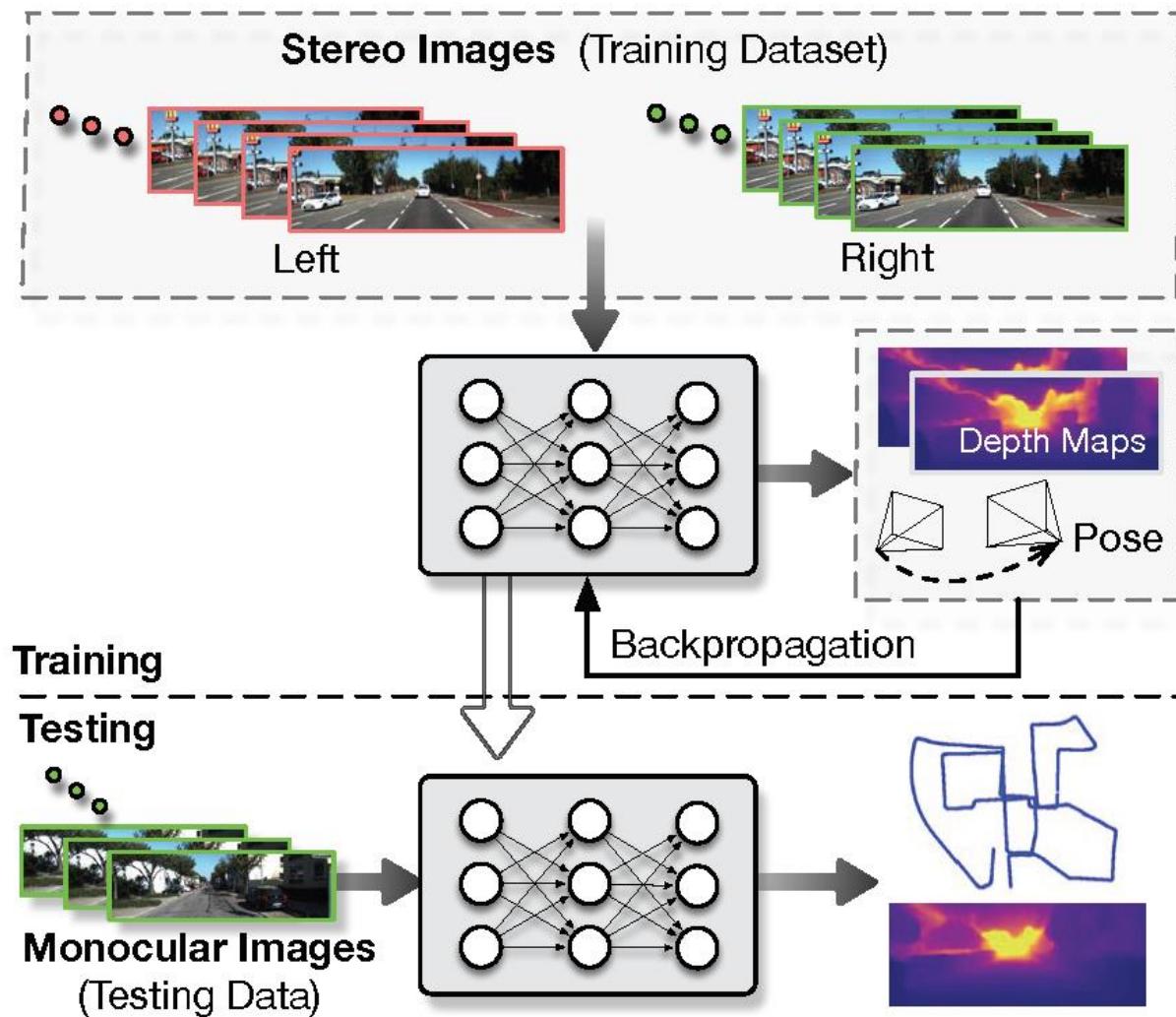
Unsupervised Monocular Depth Estimation with Left-Right Consistency



Deep Visual Odometry

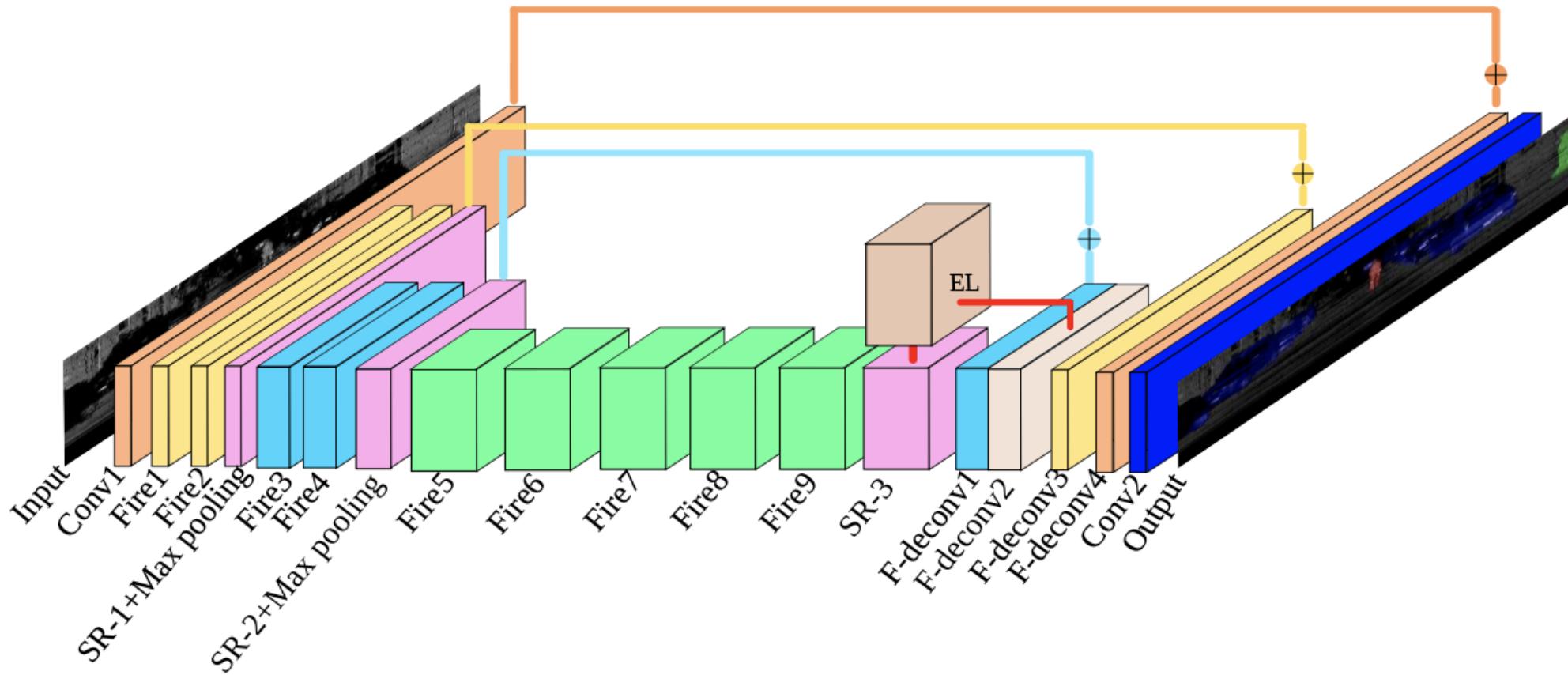


UnDeepVO

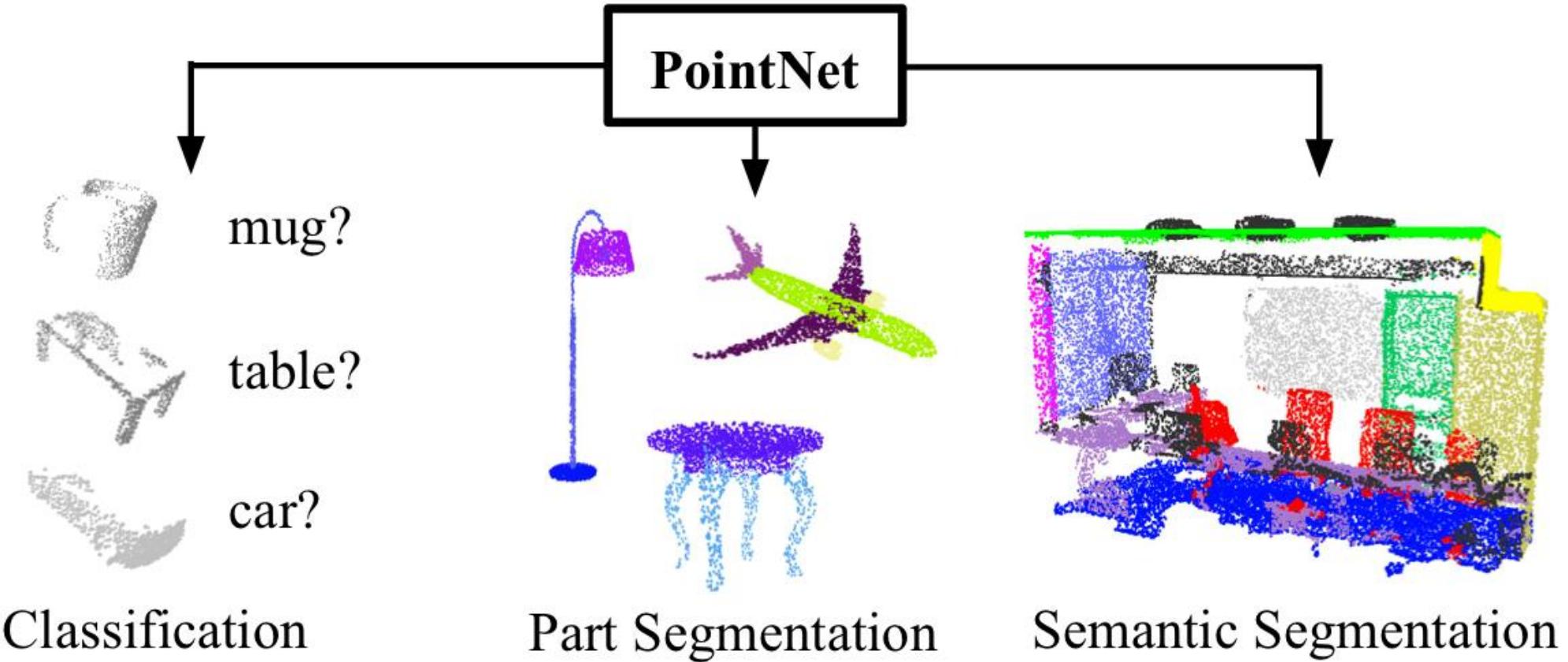


Point Cloud Segmentation

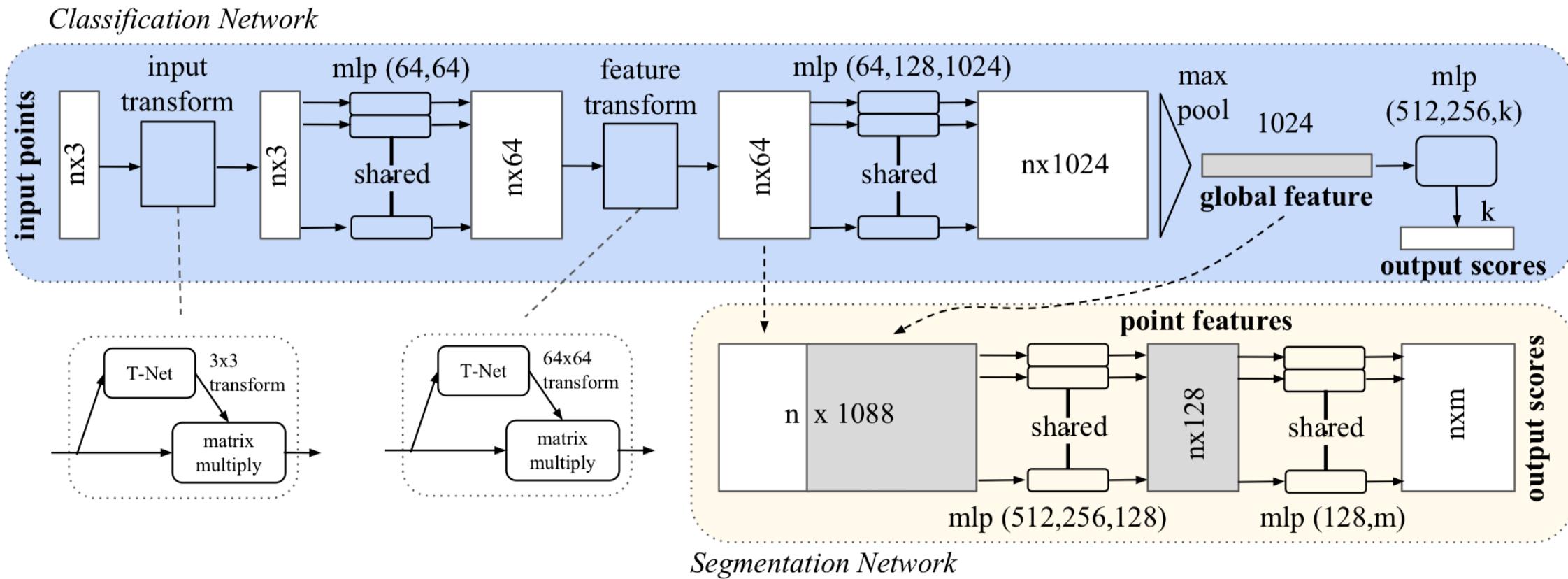
PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud



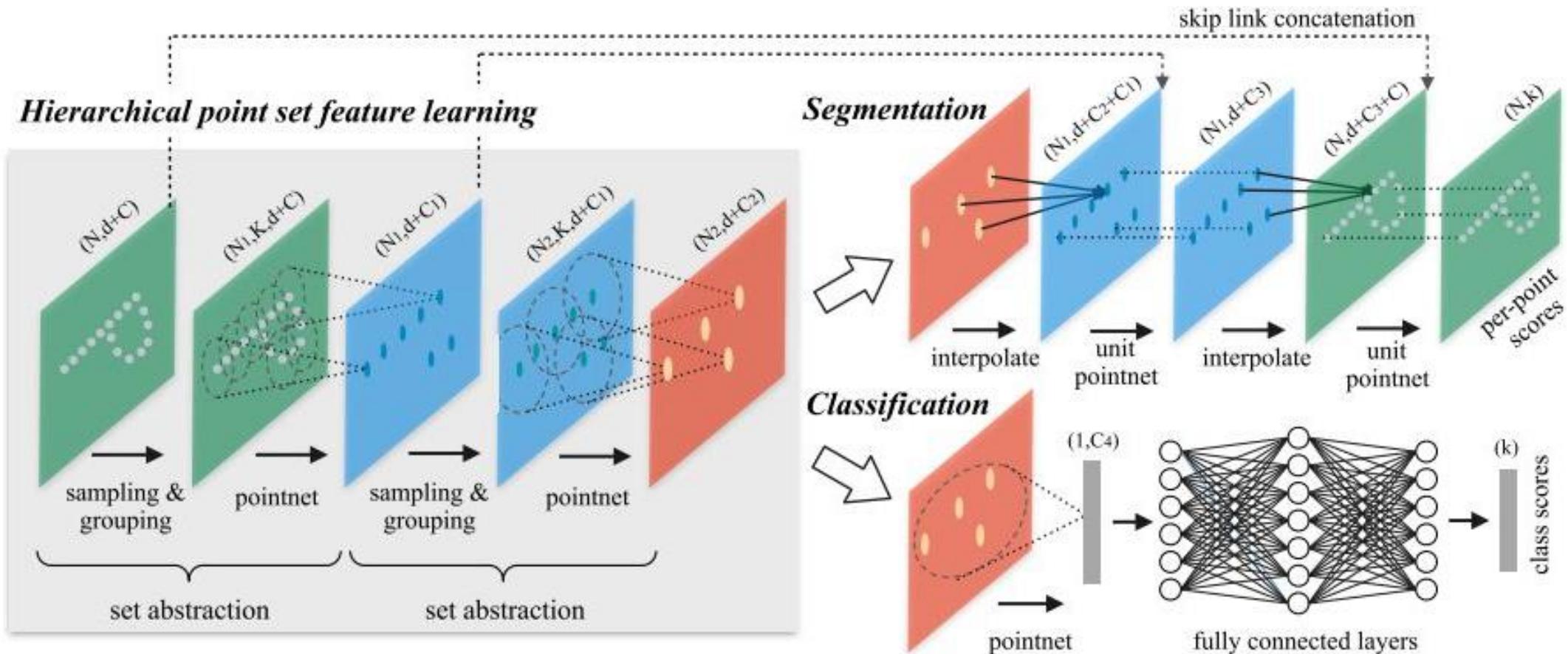
PointNet



PointNet



PointNet++



Q & A