Deep Learning in Computer Vision

Jongwoo Lim

Dept. of Computer Science
Hanyang University

Artificial Neural Network

Artificial Neural Network

Computer algorithms that try to imitate the operations/functionalities of the neuron and brain.

- Developed since 1940≃1950's. – 등장
- Revisited in 1980's, and recently. – 역전파법 정립 but 실제활용x
- Multi-layer perceptrons, vanishing gradient problem, GPU

↳ 복잡하지 않은 문제.

Perceptron

Binary classifier

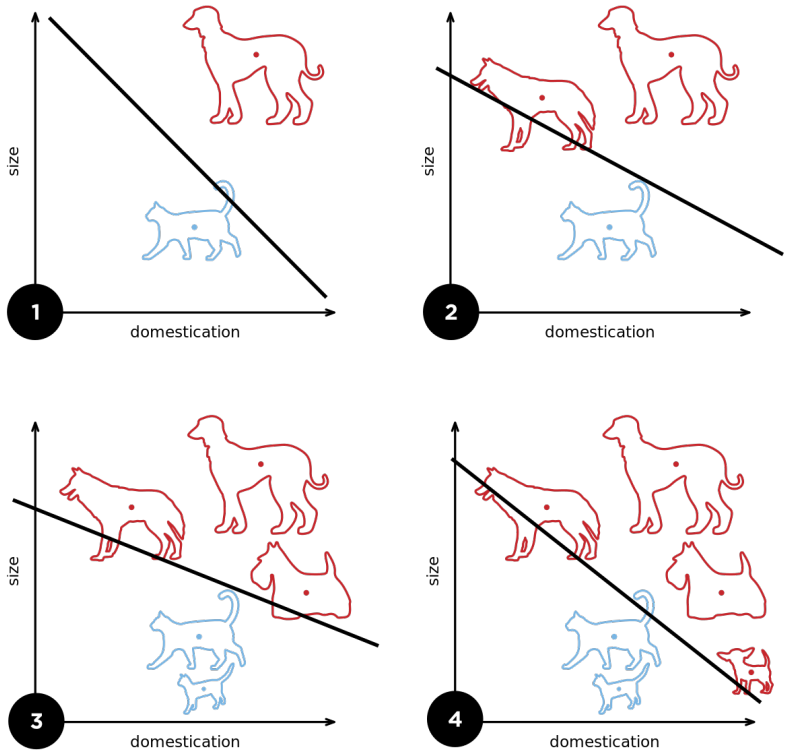
- For an input $\mathbf{x}$

$$y(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Weighted sum + bias.
- $y(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}})$,

  where $\tilde{\mathbf{w}} = (\mathbf{w}^\top, b)^\top$,

  and $\tilde{\mathbf{x}} = (\mathbf{x}^\top, 1)^\top$.

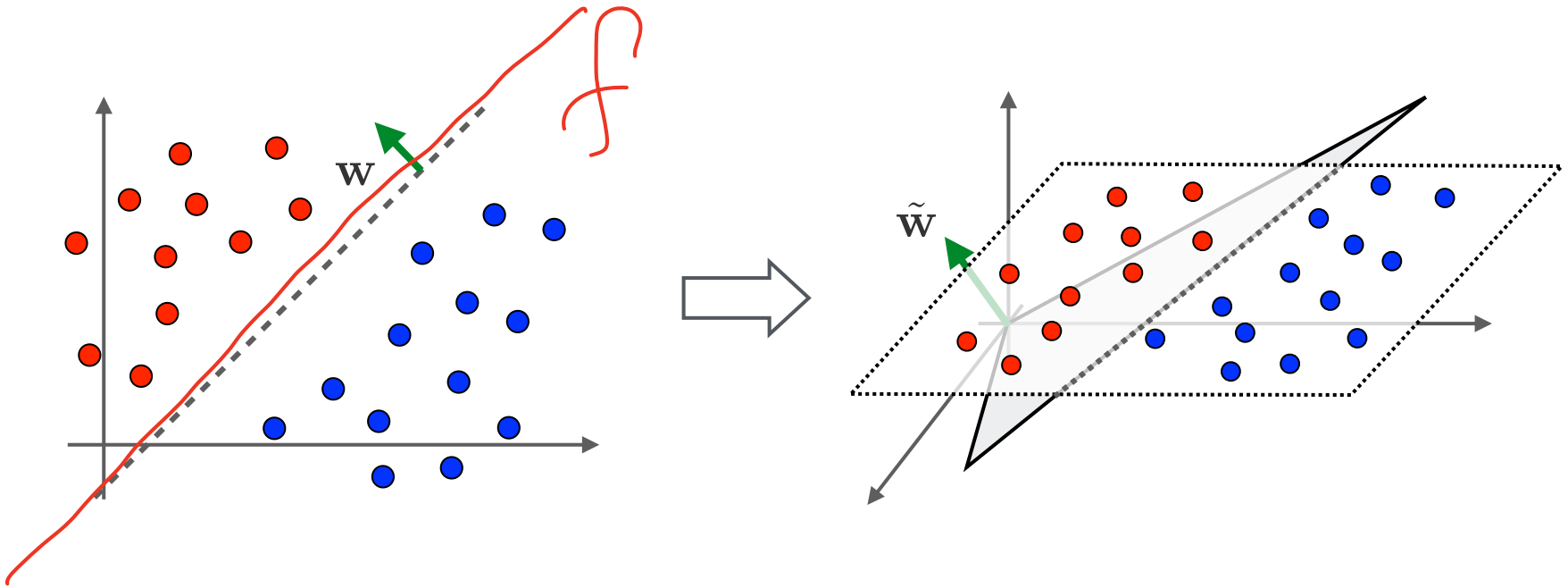


wikipedia

Perceptron Learning Algorithm

When a training data set $\{(\mathbf{x}_i, y_i)\}$ is given,
learn a classifier $\mathbf{w}$ and b which satisfies $y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) > 0, \forall i$.

- Initialize $\tilde{\mathbf{w}} = 0$.
- Until all data is correctly classified,
  if $(\mathbf{x}_i, y_i)$ is misclassified,
  update $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} + \alpha\ \mathrm{sign}(y(\mathbf{x}_i))\ \tilde{\mathbf{x}}_i$



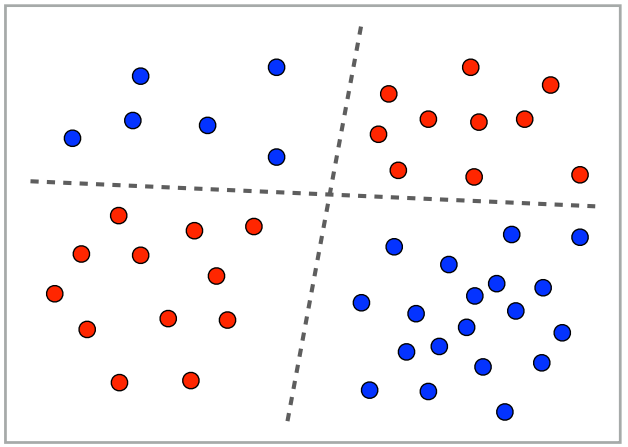
Multi-layer Perceptrons

A perceptron can handle linearly separable problems only.

When $\tilde{\mathbf{x}} = (a, b, 1)^\top$ and $a, b \in \{-1, 1\}$,

- $\neg a \quad : \quad \tilde{\mathbf{w}} = (-1, 0, 0)$,
- $a \wedge b \quad : \quad \tilde{\mathbf{w}} = (1, 1, 1)$,
- $a \vee b \quad : \quad \tilde{\mathbf{w}} = (1, 1, -1)$.
- a XOR b ?

$$a \text{ XOR } b = \neg(a \vee b) \vee (a \wedge b)$$

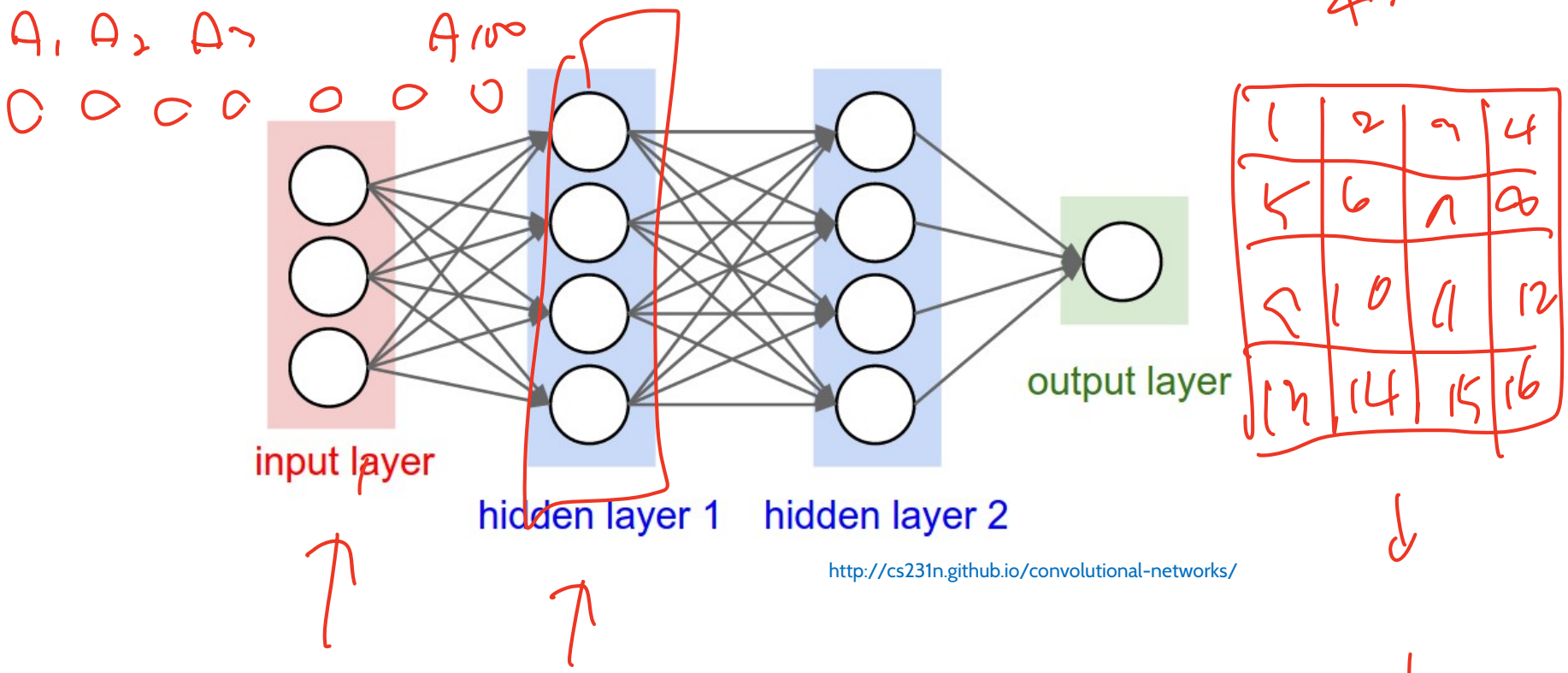


Multi-layer perceptrons are necessary for not linearly separable problems.

Training Multi-layer Perceptrons

However, training multi-layer perceptrons is not easy.

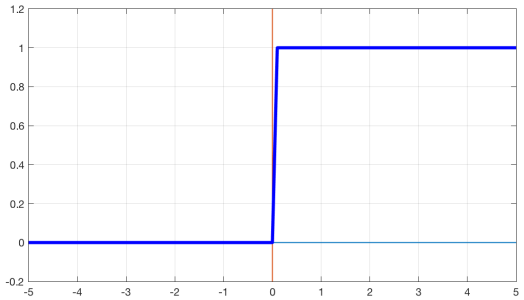
- For XOR, there are only 4 training data.
- How many layers and how many perceptrons are needed?
- How the weights and biases can be determined?



http://cs231n.github.io/convolutional-networks/

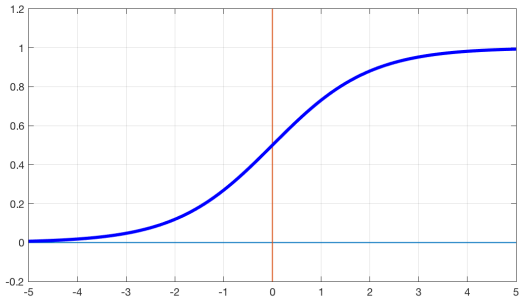
Activation Functions : Non-linearity

Binary



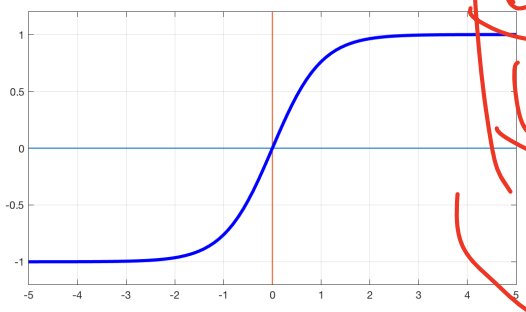
$$f(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

Logistic



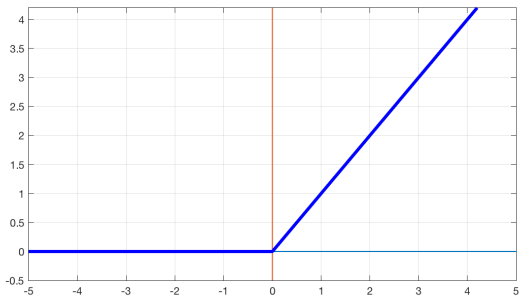
$$f(x) = \frac{1}{1 + e^{-x}}$$

Tanh



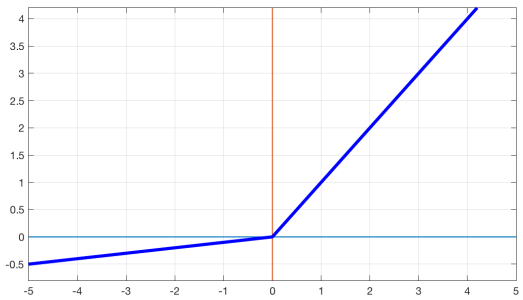
$$f(x) = \tanh(x)$$

ReLU



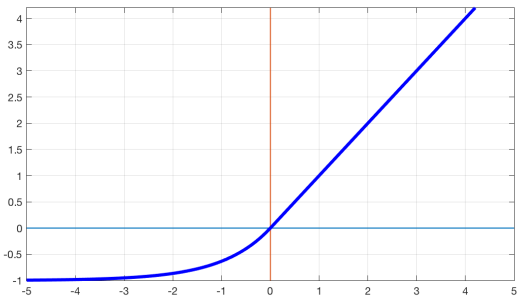
$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Leaky ReLU



$$f(x) = \begin{cases} 0.01x & x \leq 0 \\ x & x > 0 \end{cases}$$

ELU



$$f(x) = \begin{cases} \alpha(e^x - 1) & x \leq 0 \\ x & x > 0 \end{cases}$$

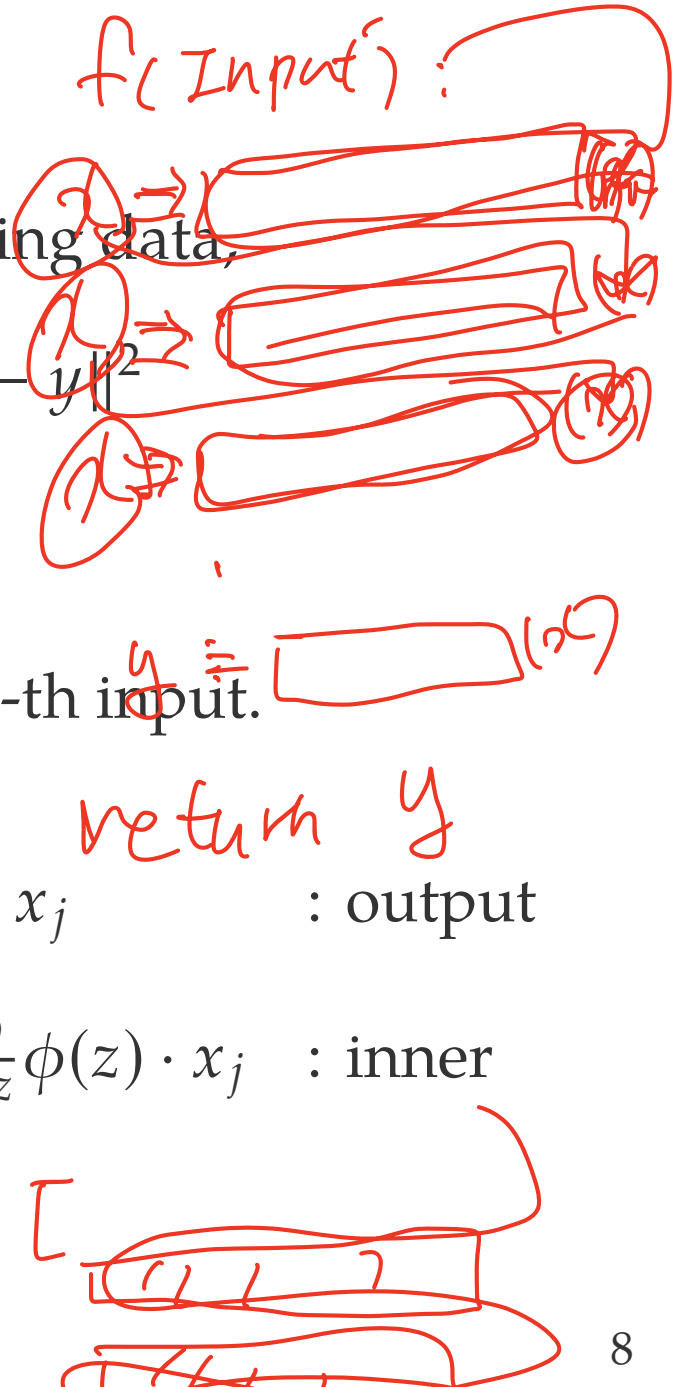
Backpropagation

To train a multi-layer network using the training data,

- Error function : $E(\mathbf{x}, y) = \frac{1}{2}\|\phi(\mathbf{w} \cdot \mathbf{x} + b) - y\|^2$
  where $\phi(\cdot)$ is an activation function.
- For each neuron, its output o is given as
  $\phi(z) = \phi\left(\sum_j w_j x_j + b\right)$. where x_j is the j-th input.

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial z}\frac{\partial z}{\partial w_j} = (y - o) \cdot \frac{\partial}{\partial z}\phi(z) \cdot x_j \quad : \text{output}$$

$$= \sum_k \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial z}\frac{\partial z}{\partial w_j} = \sum_k \frac{\partial E}{\partial o}\frac{\partial o}{\partial z}(y - o) \cdot \frac{\partial}{\partial z}\phi(z) \cdot x_j \quad : \text{inner}$$



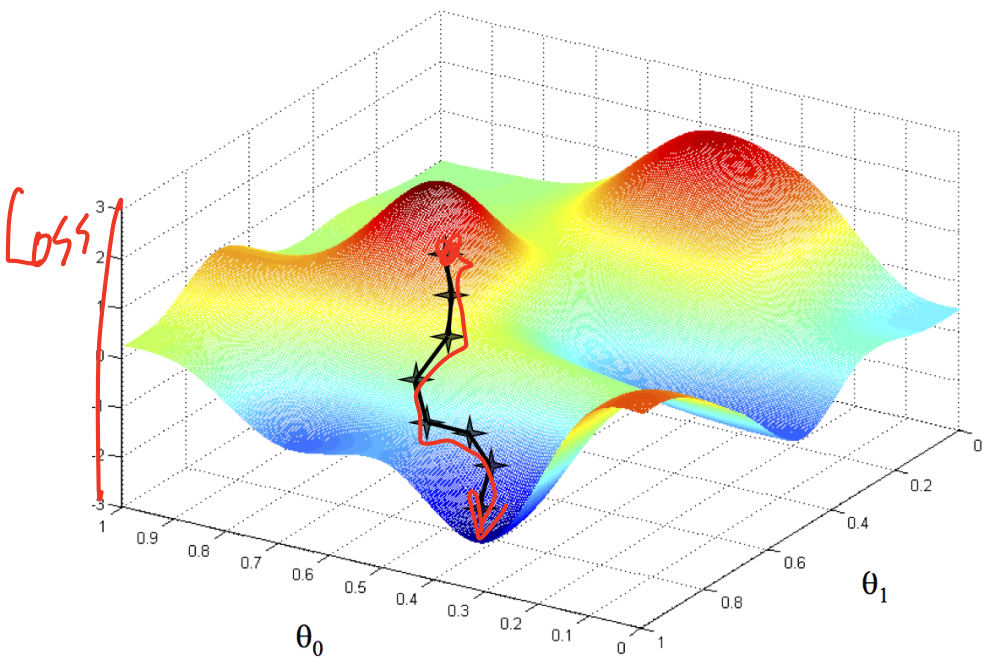
Gradient Descent Algorithm

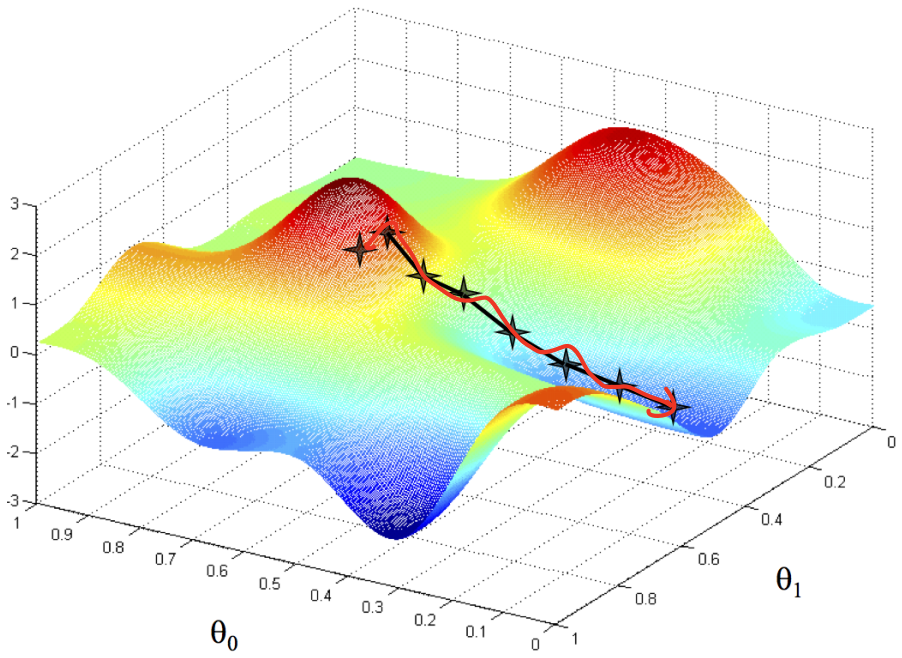
For a cost function $C(\theta)$, find

$$\min_{\theta} C(\theta).$$

- Start at some (good) $\theta = \theta_0$.
- Update θ in the direction that C reduces :

$$\theta \leftarrow \theta - \eta \nabla C(\theta)$$





Stochastic Gradient Descent

SGD : Momenterm.

Nes, Adam.

Stochastic / Incremental / On-line gradient descent :

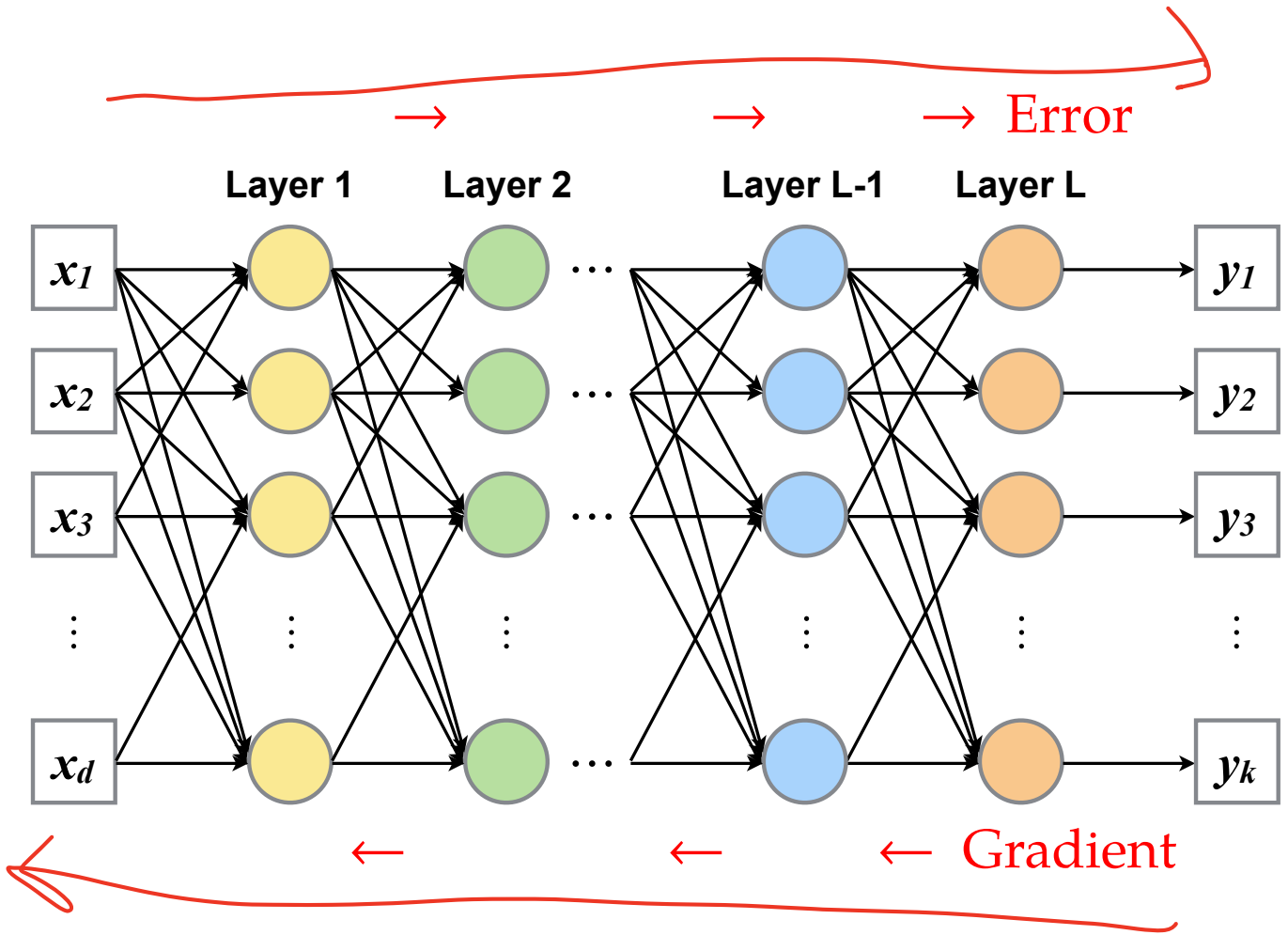
$$C(\boldsymbol{\theta}) = \frac{1}{n} \sum_i C_i(\boldsymbol{\theta})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla C(\boldsymbol{\theta}) \quad \simeq \quad \{ \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla C_i(\boldsymbol{\theta}) \}_i$$

- Simulate the gradient of the cost function by one example.
- Iteratively update with different examples.

optimizer

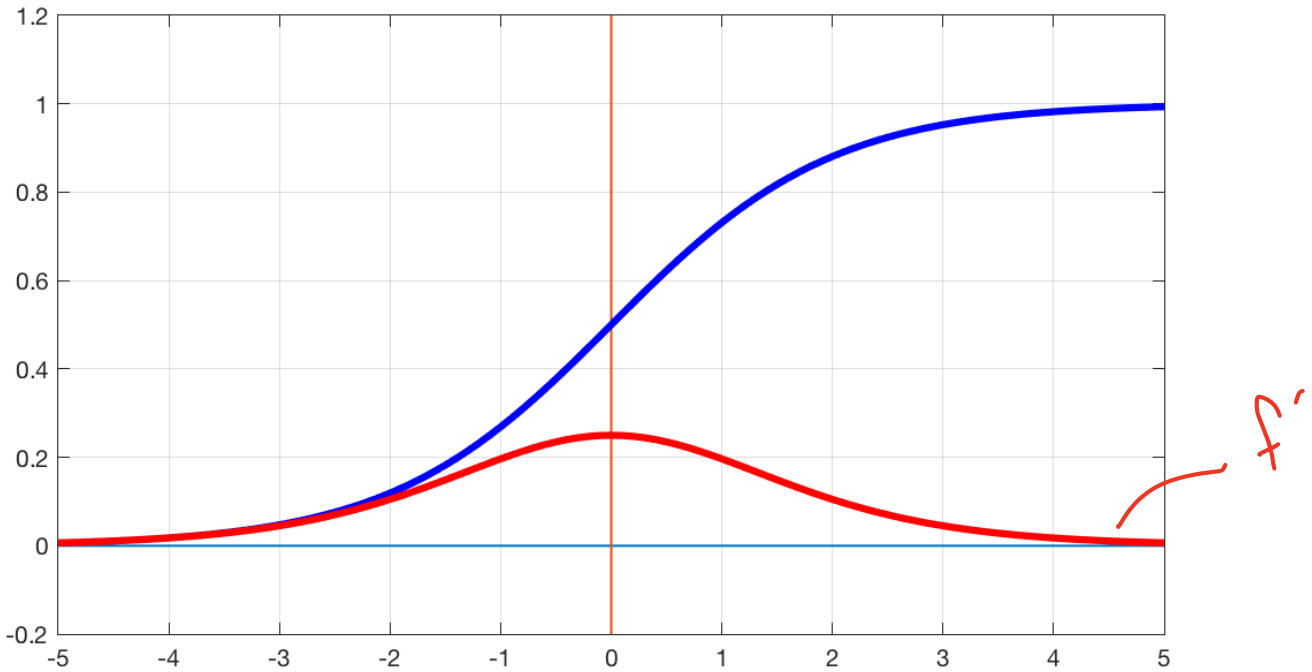
Back Propagation



- Compute the error at the output by forward pass.
- Back-propagate the error according to the gradient values.

Vanishing Gradient Problem

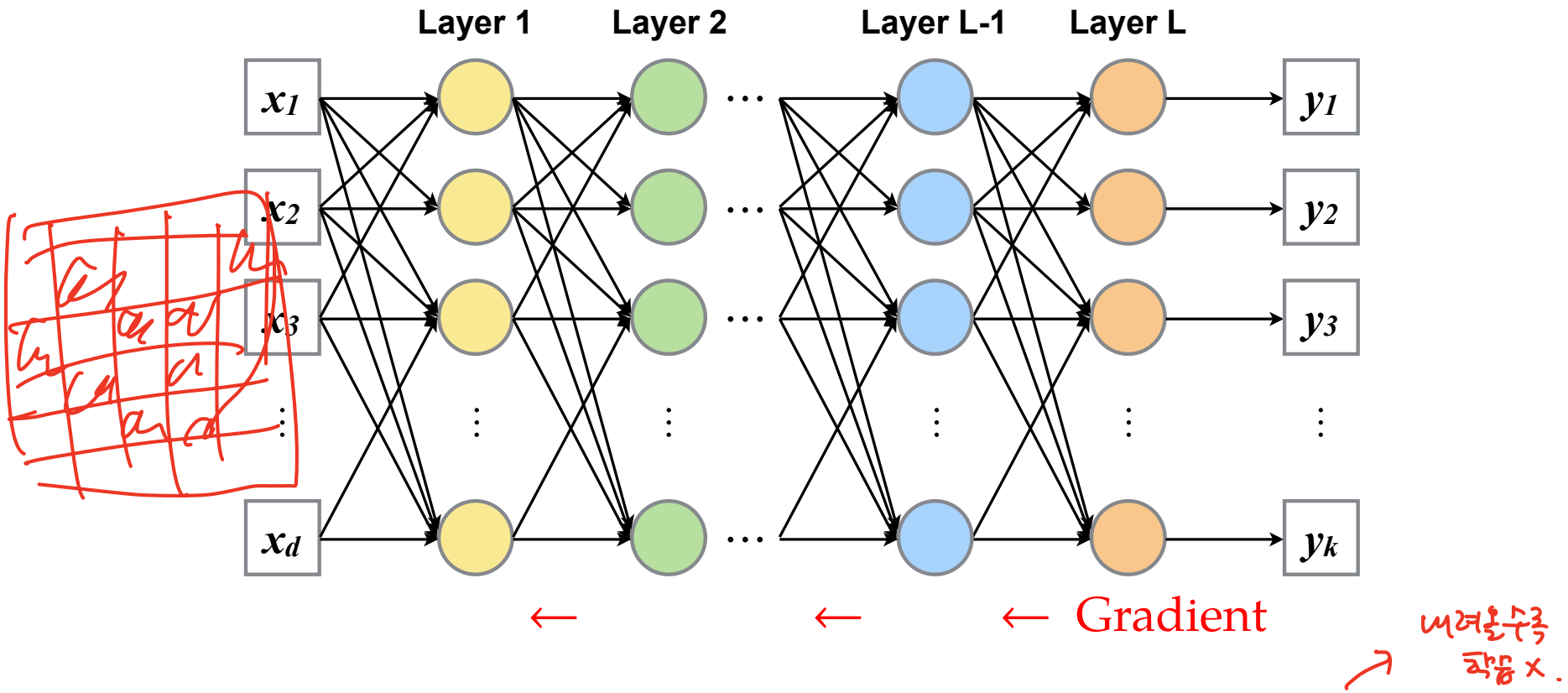
Logistic (sigmoid) activation function



- Derivative is always much smaller than 1.
- Derivative is almost 0 except the center (near 0) region.

Vanishing Gradient Problem

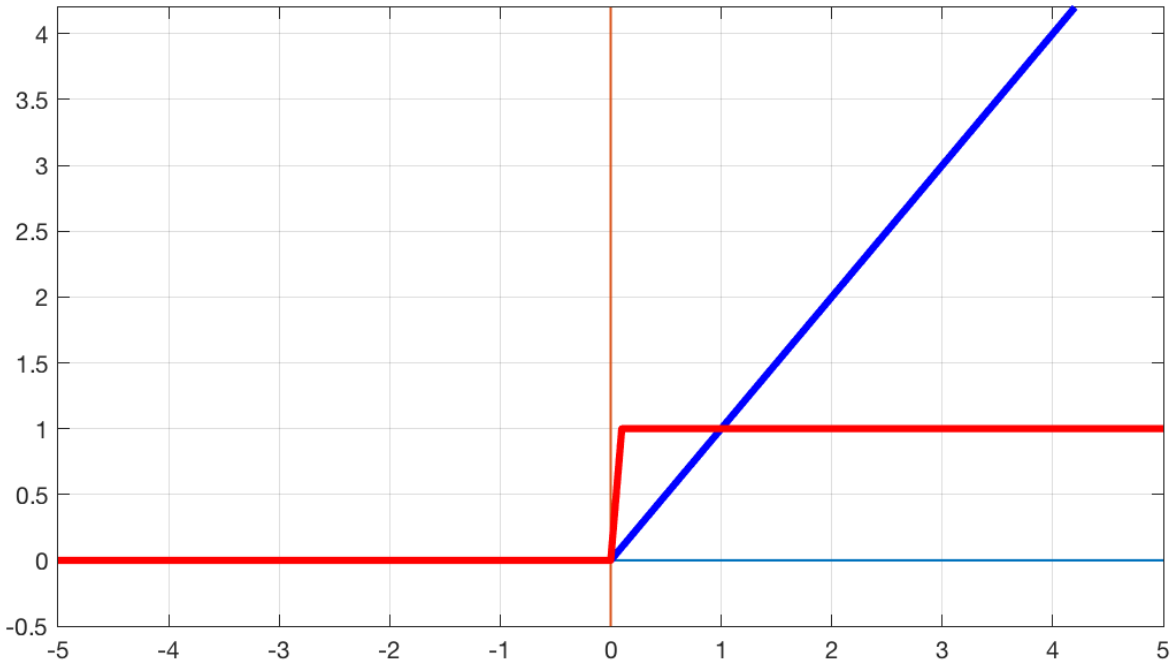
Slide adapted from Hung-yi Lee's tutorial



- Last layers have large gradients, but early layers have small.
- Last layers learn fast, but early layers learn slow.
- Last layers are converged, but early layers are still random?

ReLU

Rectified Linear Unit activation function



- Derivative is always 0 or 1 - no vanishing gradient
- Derivative is not smooth at 0.

Softmax Classification

Slide adapted from Hung-yi Lee's tutorial

The output for final classification is the class indication vector.

$$\hat{\mathbf{y}} = (0, 0, \ldots, 0, 1, 0, \ldots, 0)^\top$$

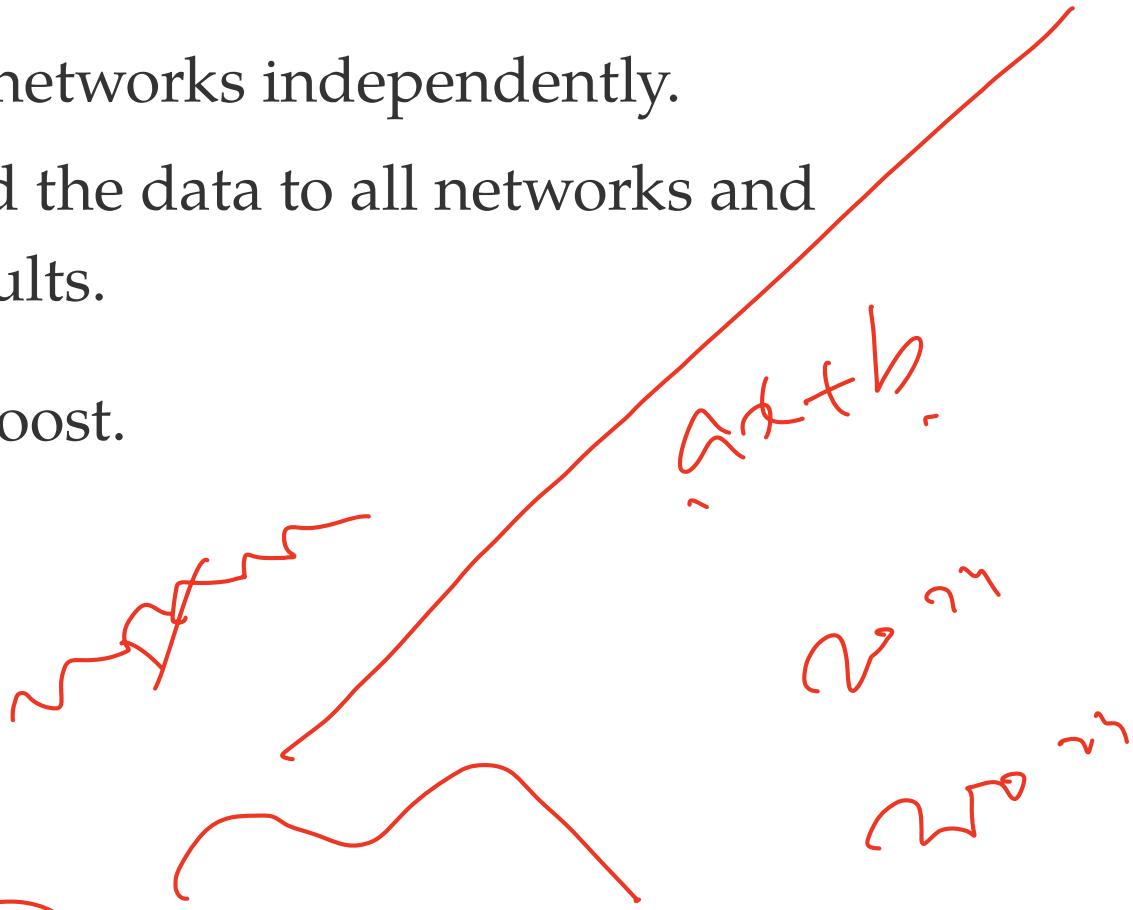
- ReLU's output is not bounded ($[0, \infty]$).
- Softmax : compute $\mathbf{y} = \frac{1}{\sum_i e^{x_i}}(e^{x_0}, e^{x_1}, \ldots, e^{x_{n-1}})$, then take $\arg\max_i(\mathbf{y})$.

Model Ensemble

Ensemble algorithm:

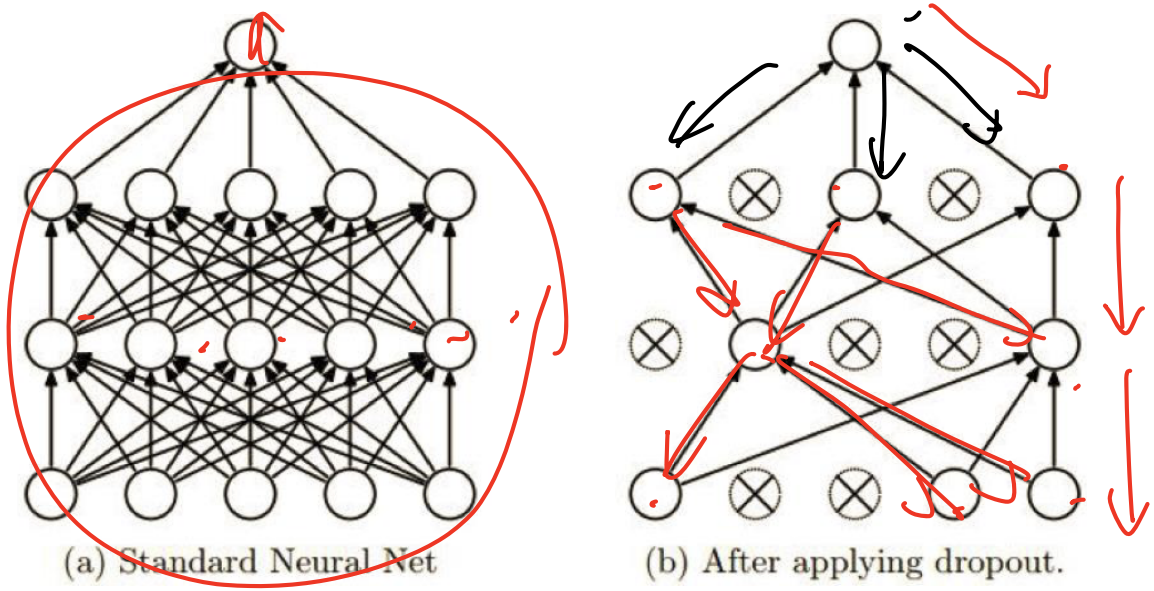
- Train multiple networks independently.
- For testing, feed the data to all networks and average the results.

~2% performance boost.



Dropout

Slide adapted from Hung-yi Lee's tutorial



(a) Standard Neural Net (b) After applying dropout.

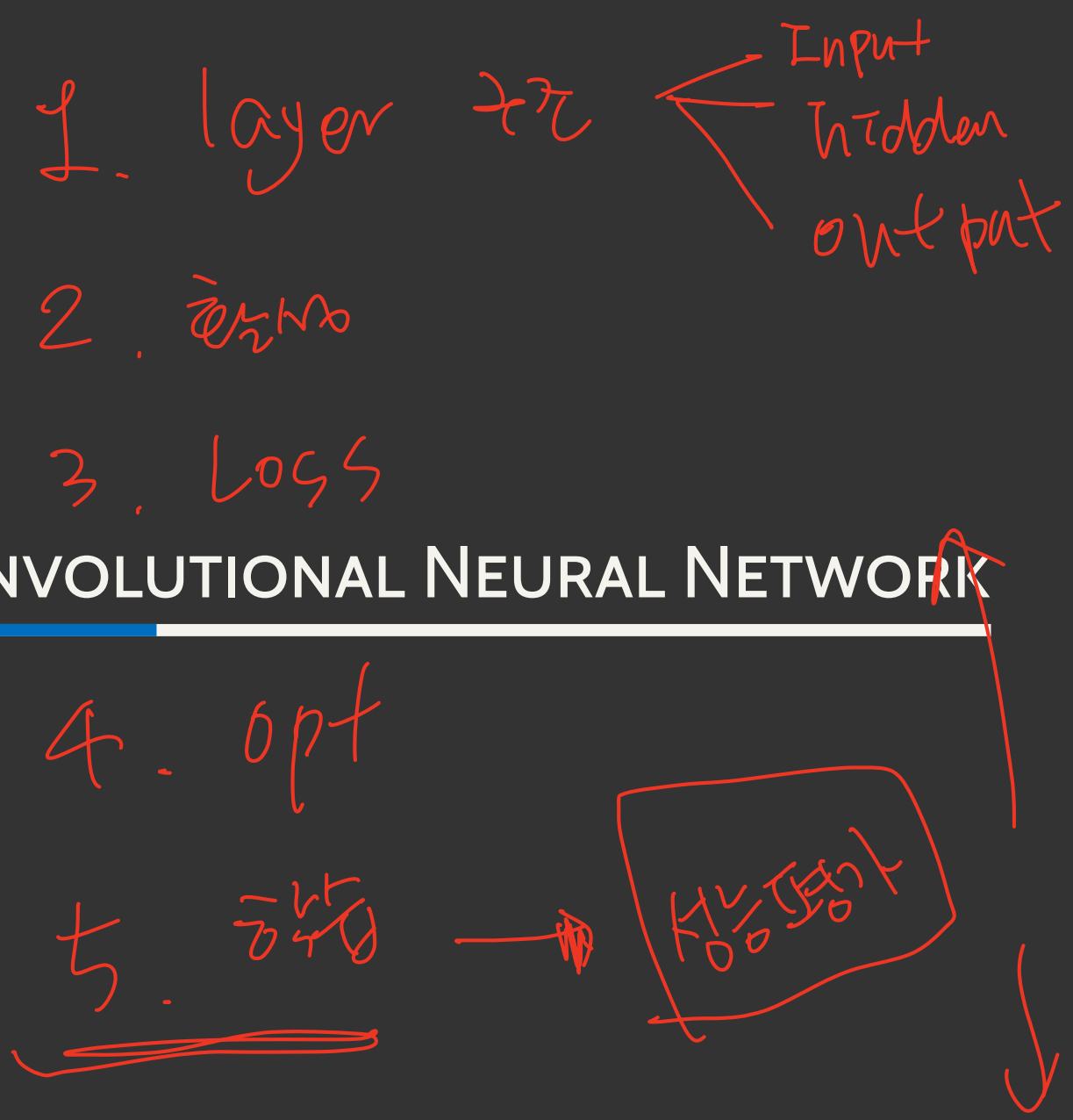
Srivastava et.al. 2014

In training, randomly turn off nodes in the network.

Interpretation:

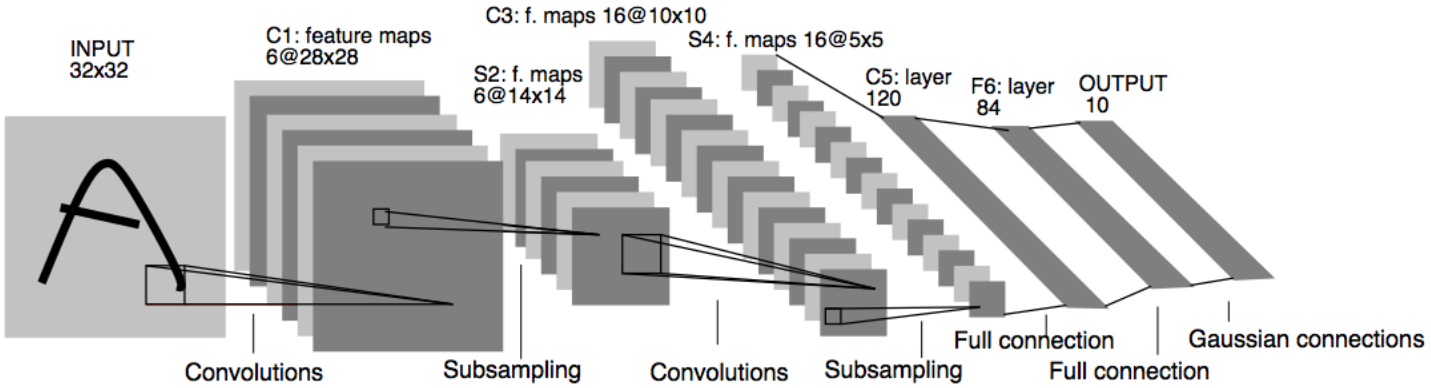
- Makes the network have redundant features - robustness.
- Can be seen as an ensemble of networks which shares the parameters.

Convolutional Neural Network

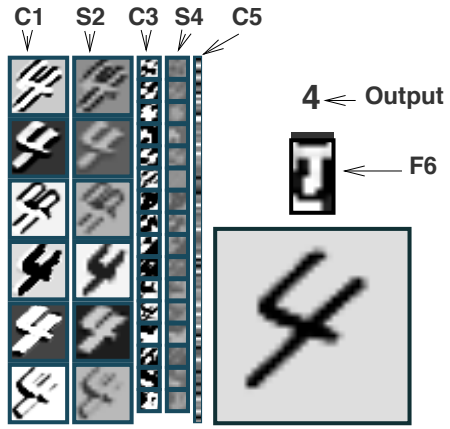


Convolutional Neural Network

Slide adapted from Ranzato's CVPR14 tutorial



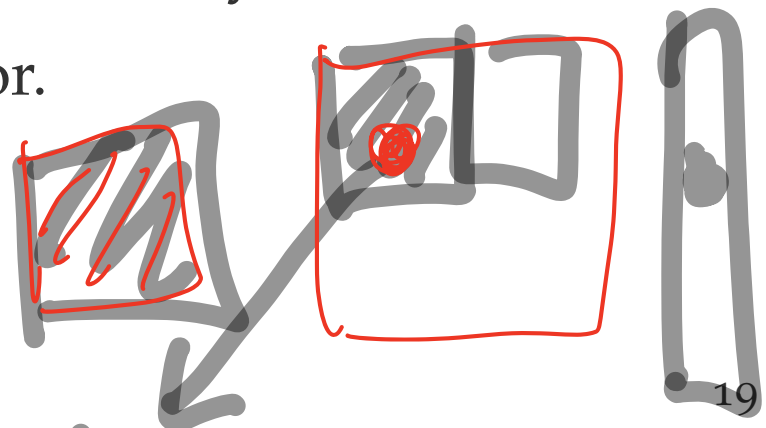
Lecun et.al. 1998



Lecun et.al. 1998

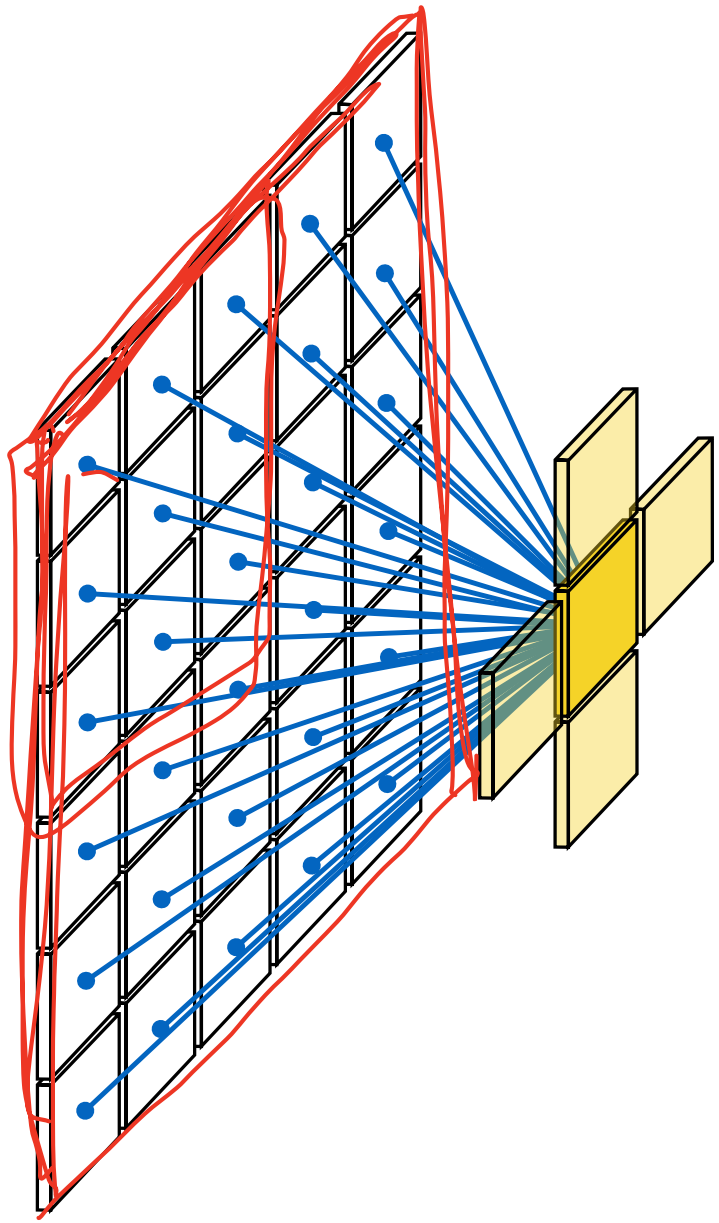
LeNet-5 for digit recognition

- Neural network with specialized connectivity structures.
- Low, mid, high-level feature extractor.



Fully Connected Layer

Slide adapted from Ranzato's CVPR14 tutorial



- Image features are locally defined.
- Huge parameters :

  200×200 image =

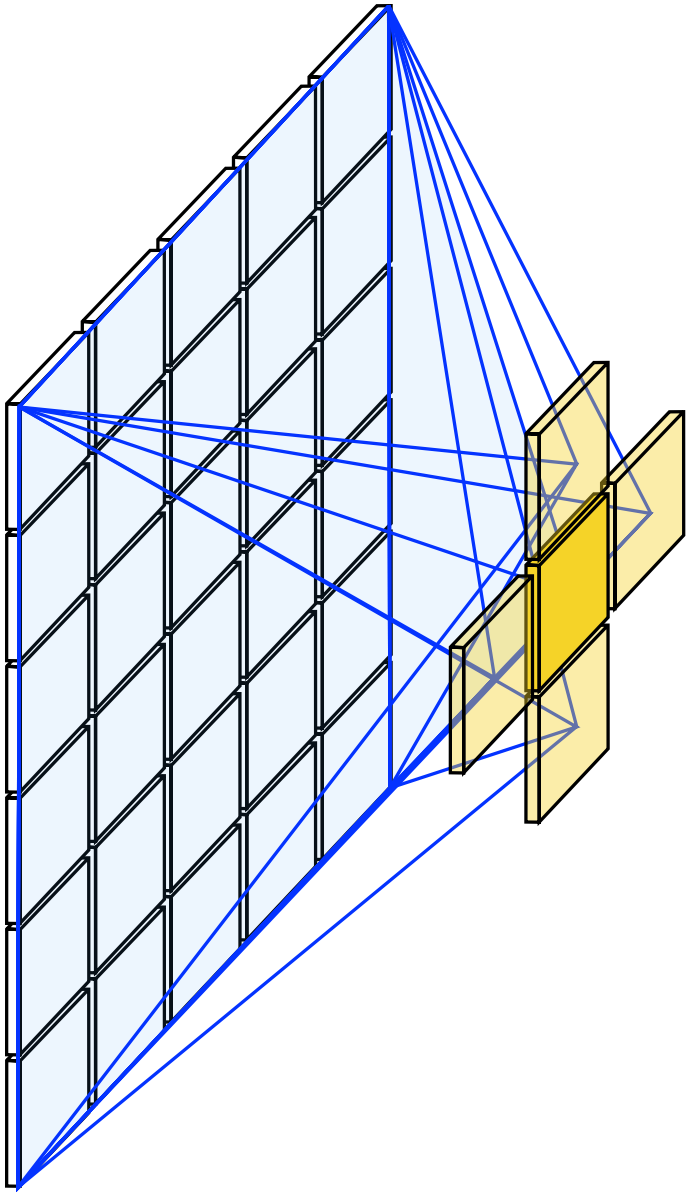
  40K weights per node,

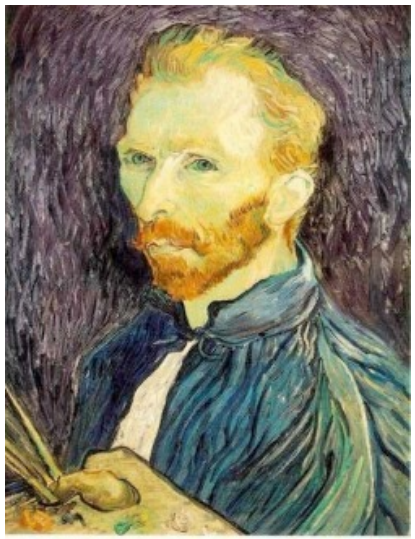
  100×100 nodes =

  400M parameters.

Fully Connected Layer

Slide adapted from Ranzato's CVPR14 tutorial





- Image features are locally defined.
- Huge parameters :

  200×200 image =

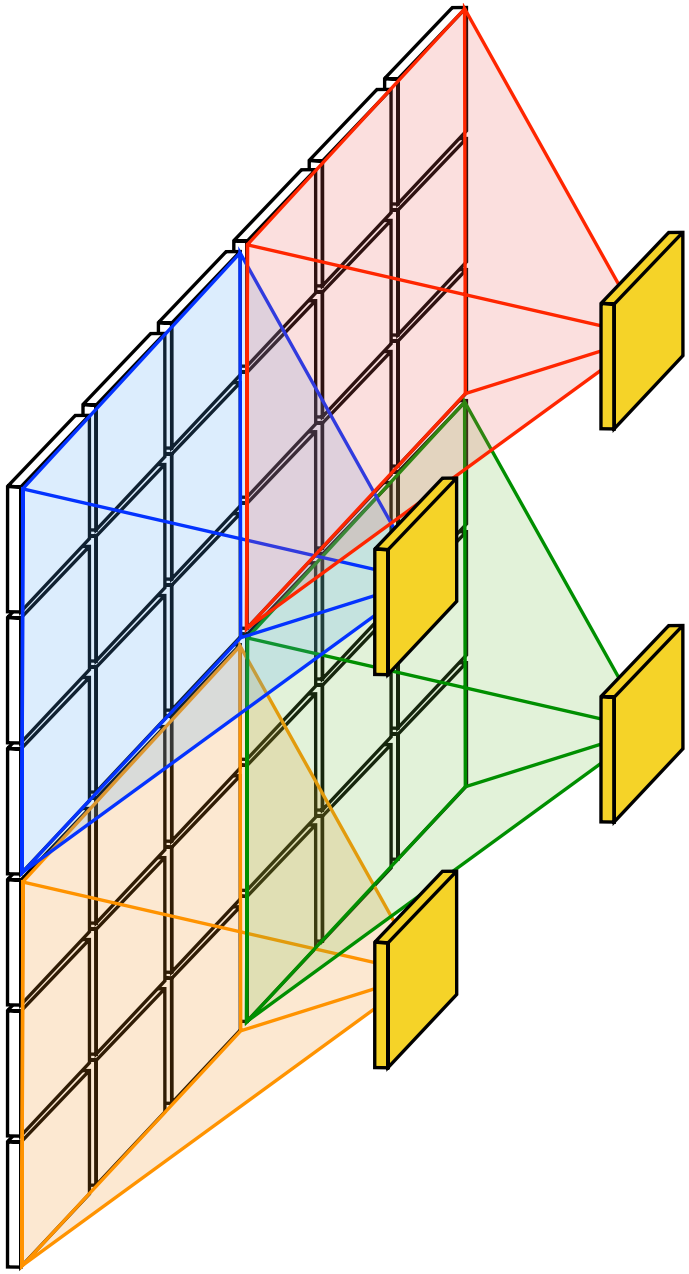
  40K weights per node,

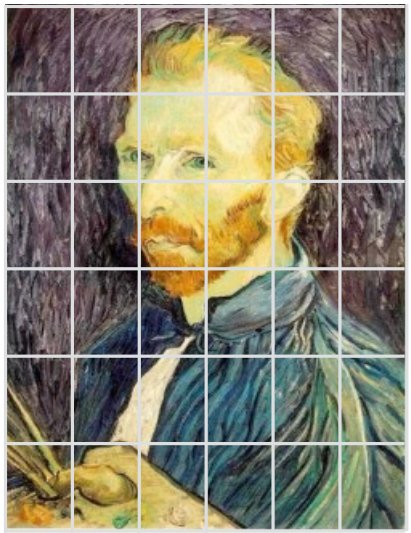
  100×100 nodes =

  400M parameters.

Locally Connected Layer

Slide adapted from Ranzato's CVPR14 tutorial

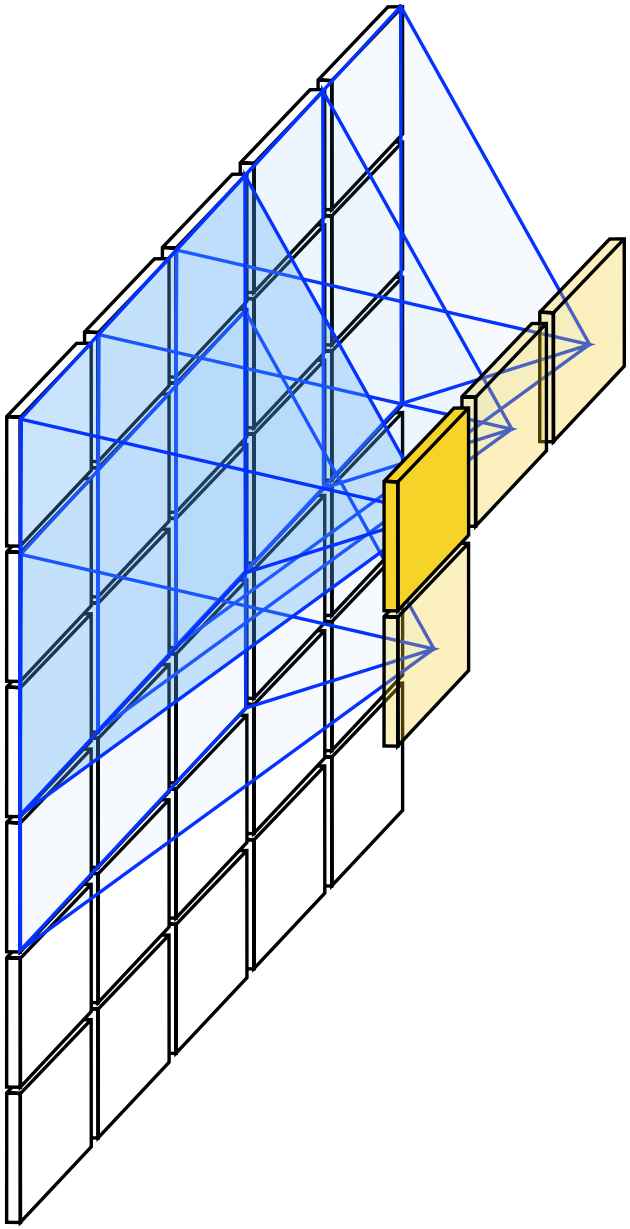


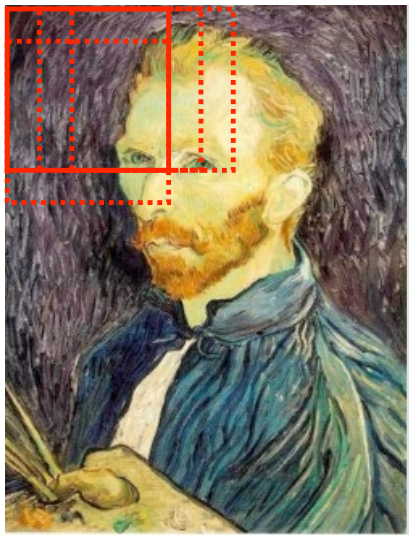


- Learn parts of the image :
  good if the image is registered.
- Large parameters :
  40×40 patch =
  1.6K weights per node,
  5×5 nodes =
  40K parameters.

Convolutional Layer

Slide adapted from Ranzato's CVPR14 tutorial

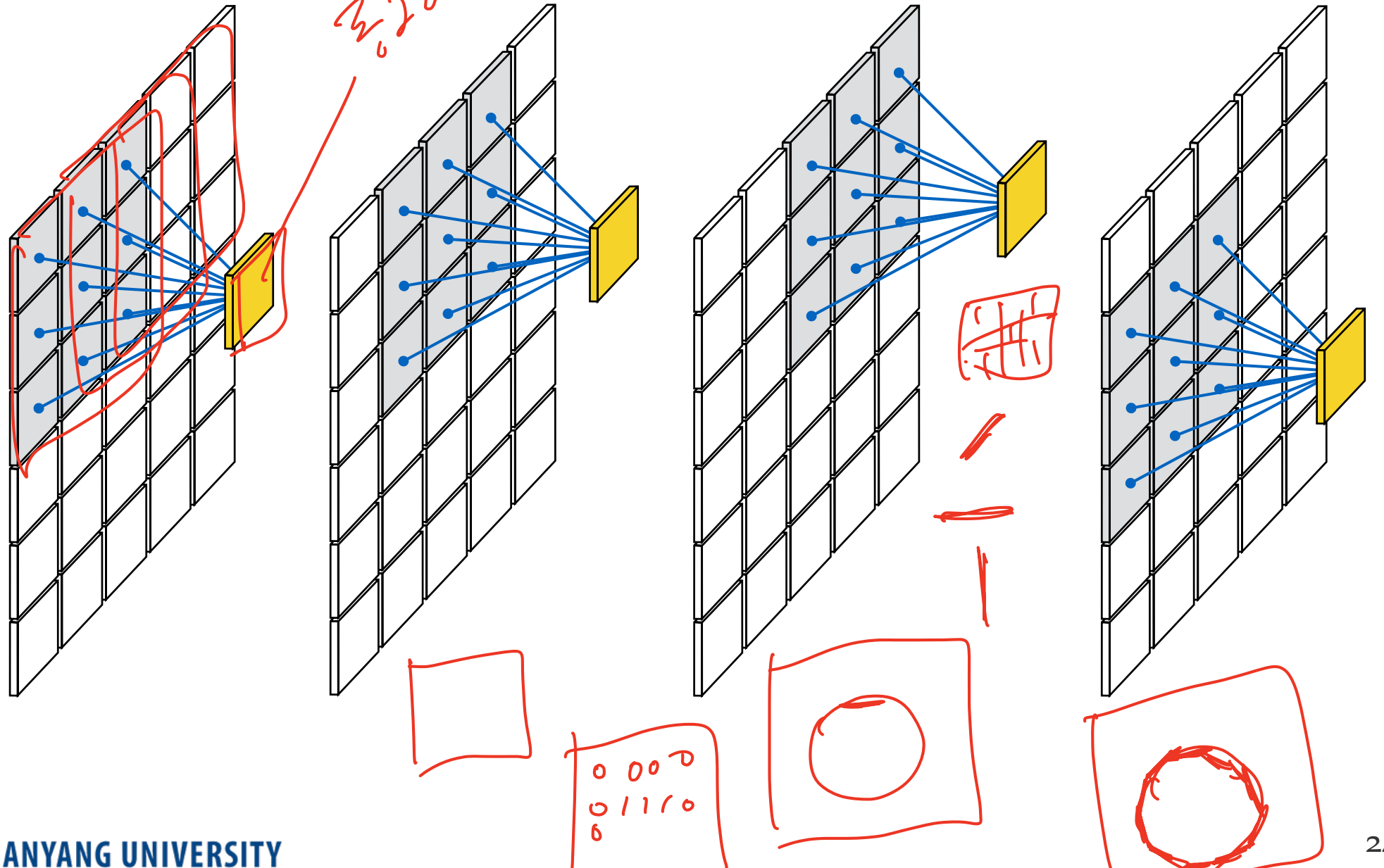




- Learn feature detector :
  high response at the learned pattern
- Small parameters :
  40×40 patch =
  1.6K weights per kernel.
  Shared with all sliding windows.

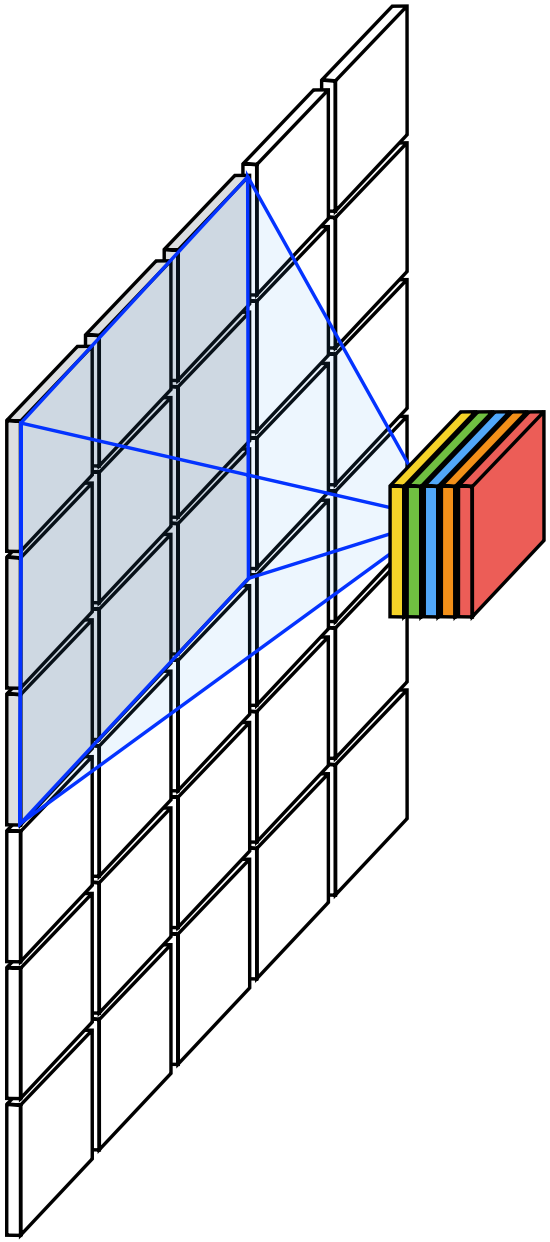
Convolutional Layer Example

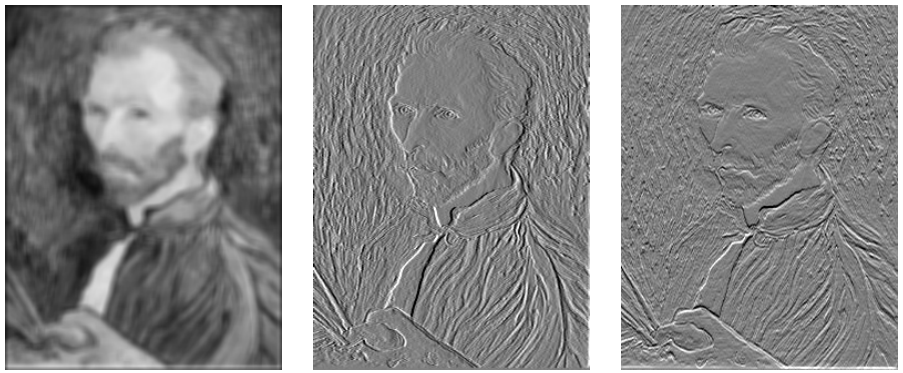
Slide adapted from Ranzato's CVPR14 tutorial

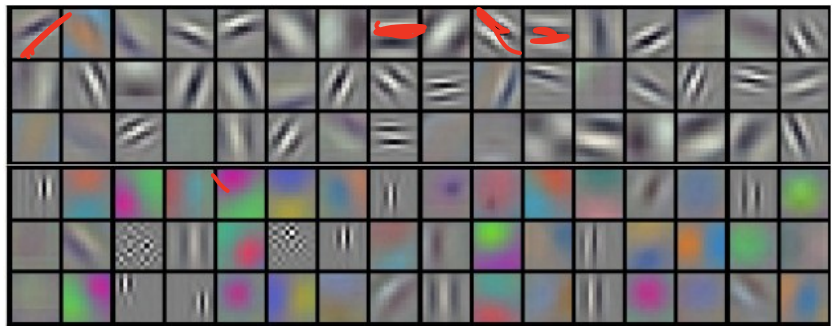


Multiple Channels

Slide adapted from Ranzato's CVPR14 tutorial





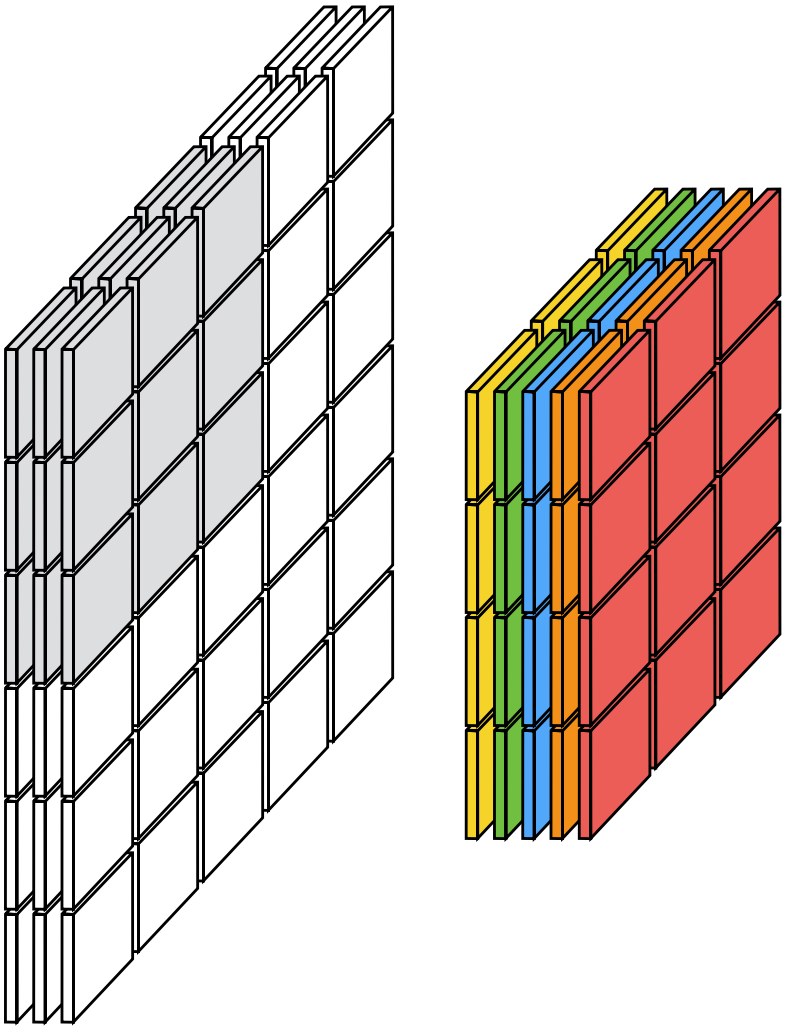


Krizhevsky et.al. NIPS 2012

- Need to learn various features to represent and summarize image contents.
- Number of parameters : Kernel size × number of channels.

Multi-channel Convolutional Layer

Slide adapted from Ranzato's CVPR14 tutorial

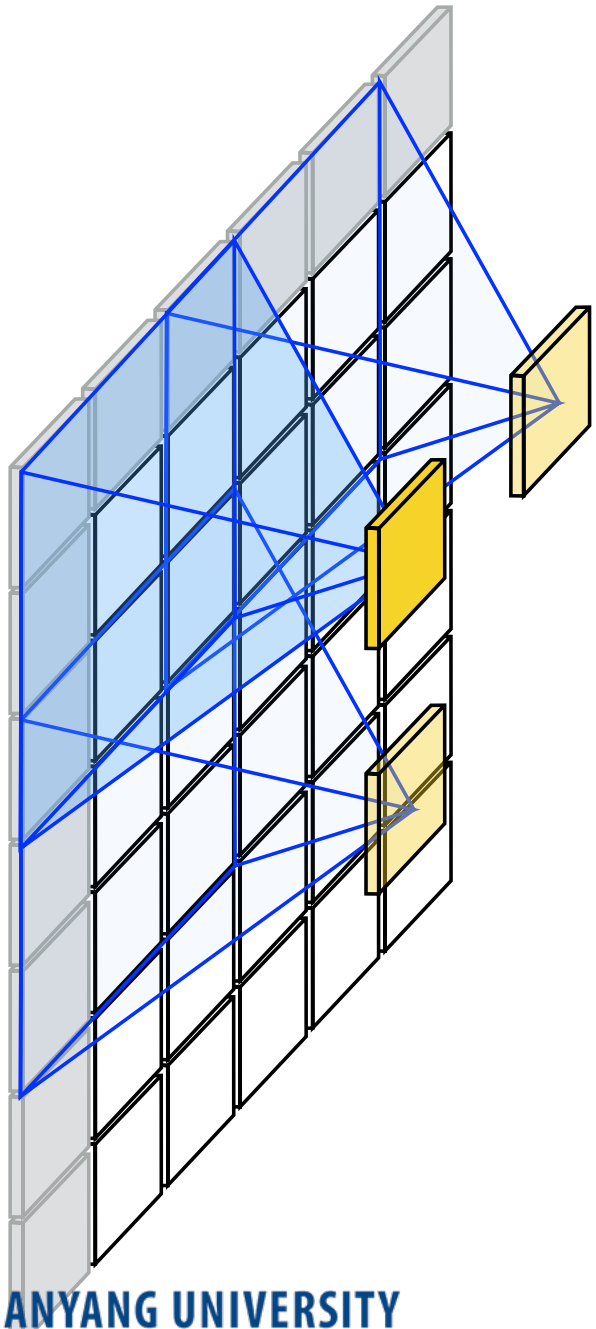


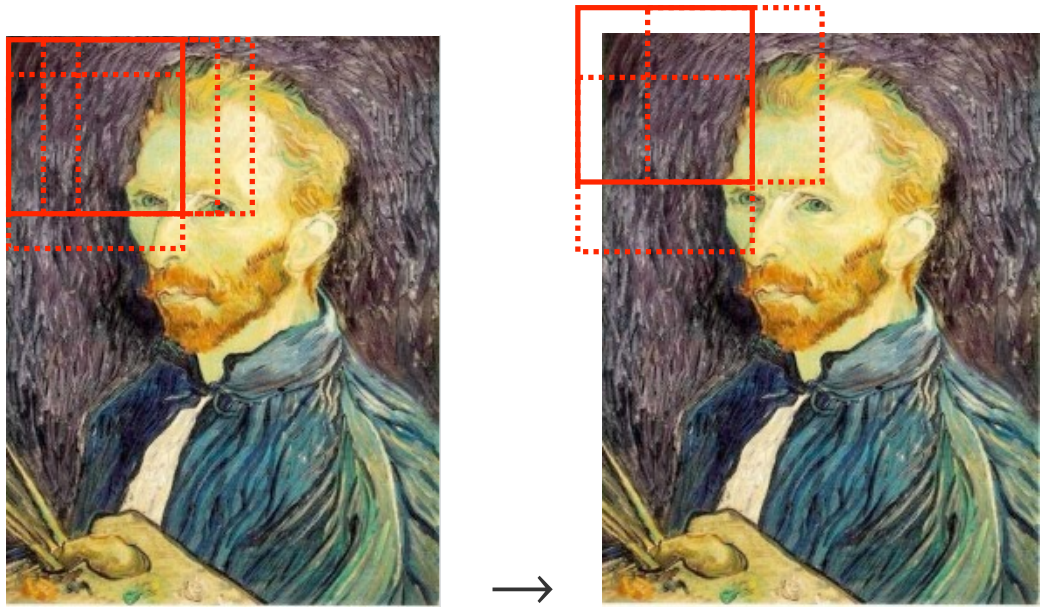
$$F_c^{(l)}(\mathbf{x}) = \sum_{c'} \sum_{\mathbf{z} \in K_W \times K_H} F_{c'}^{(l')}(\mathbf{x}+\mathbf{z})\, w_{c'}^{(c)}(\mathbf{z})$$

- The kernel $K^{(c)} = \{w_{c'}^{(c)}(\mathbf{z})\}_{c'}$ generates the channel c of the output feature map.
- The kernel for each channel is a 3-dim tensor $(K_W \times K_H \times C')$.
- #parameters in one layer is $(K_W \times K_H \times C' \times C)$

Padding and Stride

Slide adapted from Ranzato's CVPR14 tutorial





- Padding (P) increases the input size by $2P$.
- Stride (S) reduces the output size by $1/S$.
- When the input feature map width is W, and the kernel width is K_W, the output width $= (W - K_W + 2P)/S + 1$.

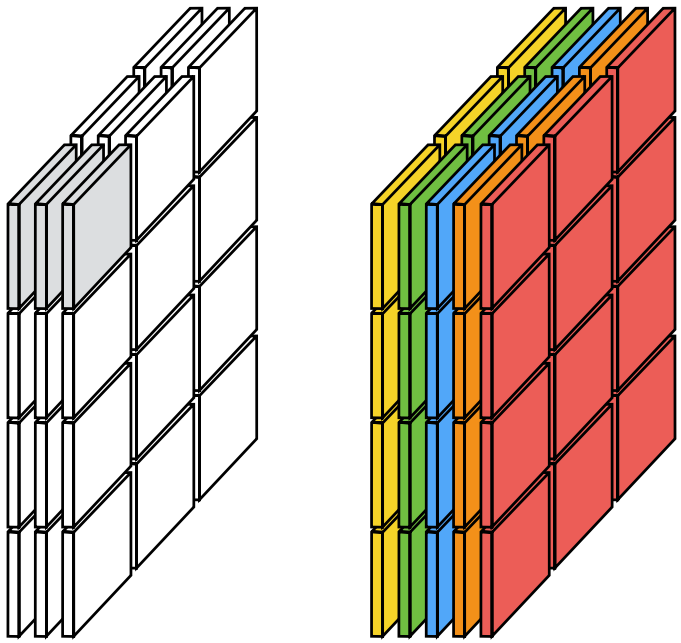
Non-linearity and 1×1 Convolutional Layer

Slide adapted from Ranzato's CVPR14 tutorial

Convolution is a linear operation.

An activation function adds non-linearity.

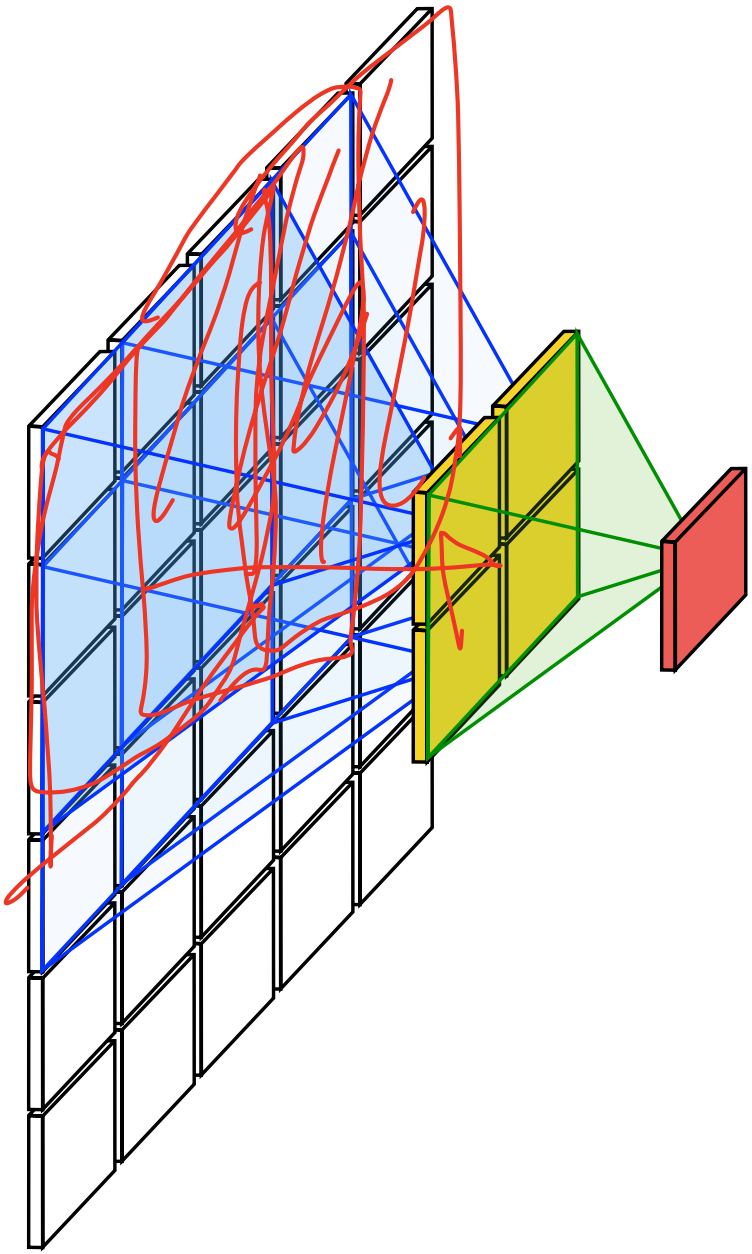
$$F_c^{(l)}(\mathbf{x}) = \max\left(0, \sum_{c'} \sum_{\mathbf{z} \in K_W \times K_H} F_{c'}^{(l')}(\mathbf{x}+\mathbf{z})\, w_{c'}^{(c)}(\mathbf{z})\right)$$

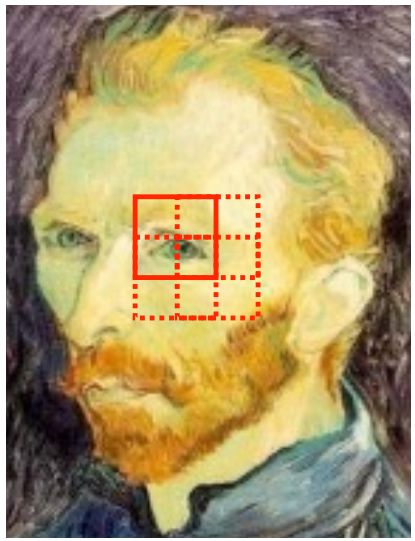


- 1×1 convolution makes sense since there are multiple channels in the input.
- In fact, $1x1$ convolution is channel-wise weighted sum.
- The input and output feature map only differs in the number of channels.

Spatial Pooling Layer

Slide adapted from Ranzato's CVPR14 tutorial

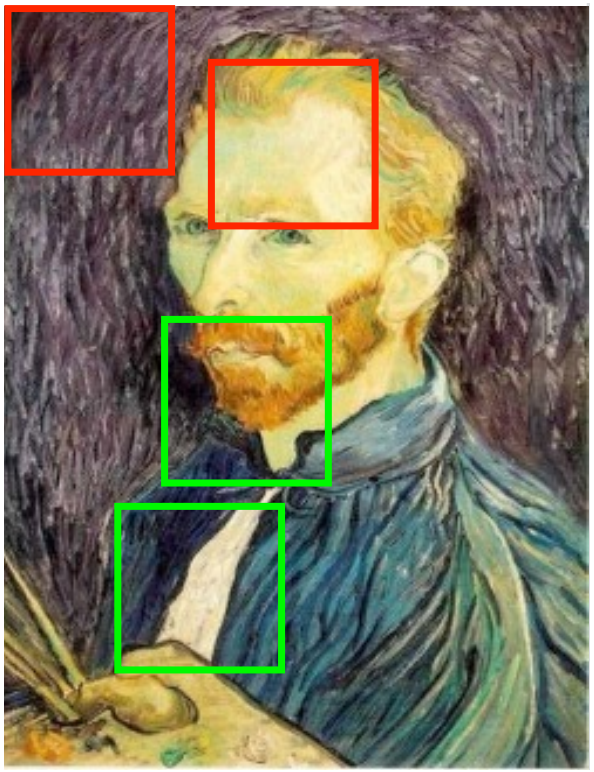




- Kernel as an 'eye' detector : robustness over image shifts.
- The pooling layer aggregates the responses in the window.
- Max pooling, Average pooling, etc.

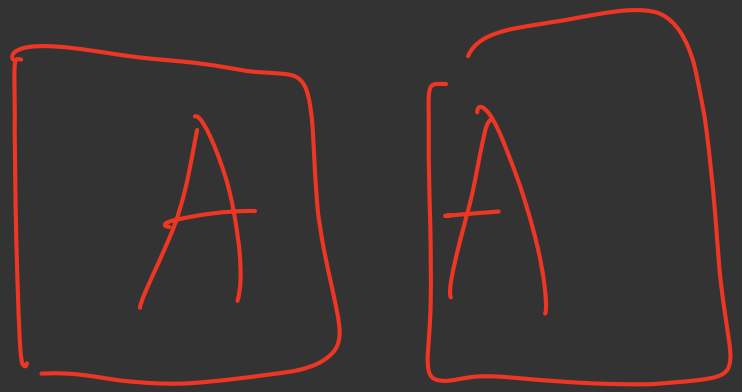
Local Constant Normalization

Slide adapted from Ranzato's CVPR14 tutorial



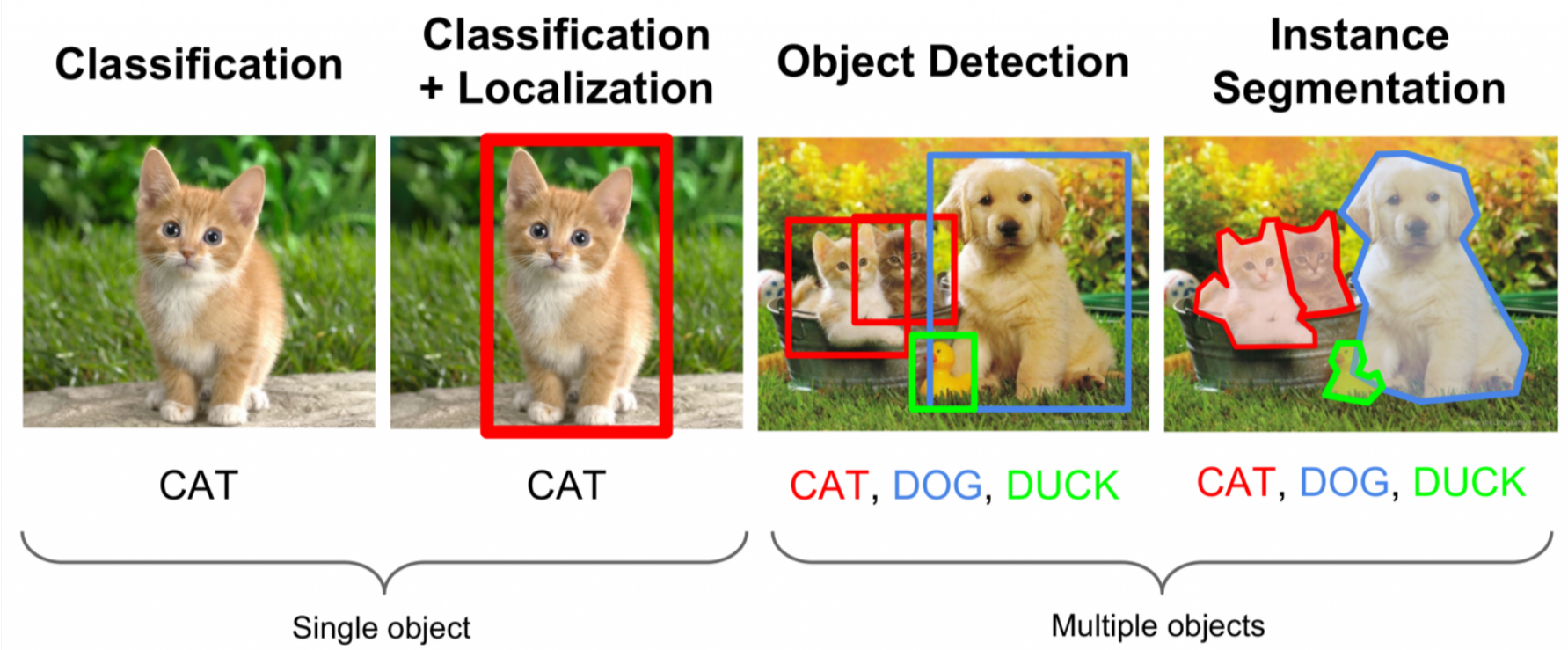
$$\hat{F}(\mathbf{x}) = \frac{F(\mathbf{x}) - \mu\left(F(\mathbf{x}') ; \mathbf{x}' \in N(\mathbf{x})\right)}{\max\left(\epsilon, \sigma\left(F(\mathbf{x}') ; \mathbf{x}' \in N(\mathbf{x})\right)\right)}$$

- Image patches have different contrast levels.
- Normalization levels the contrast of the patches.
- However, it may amplify the noise.



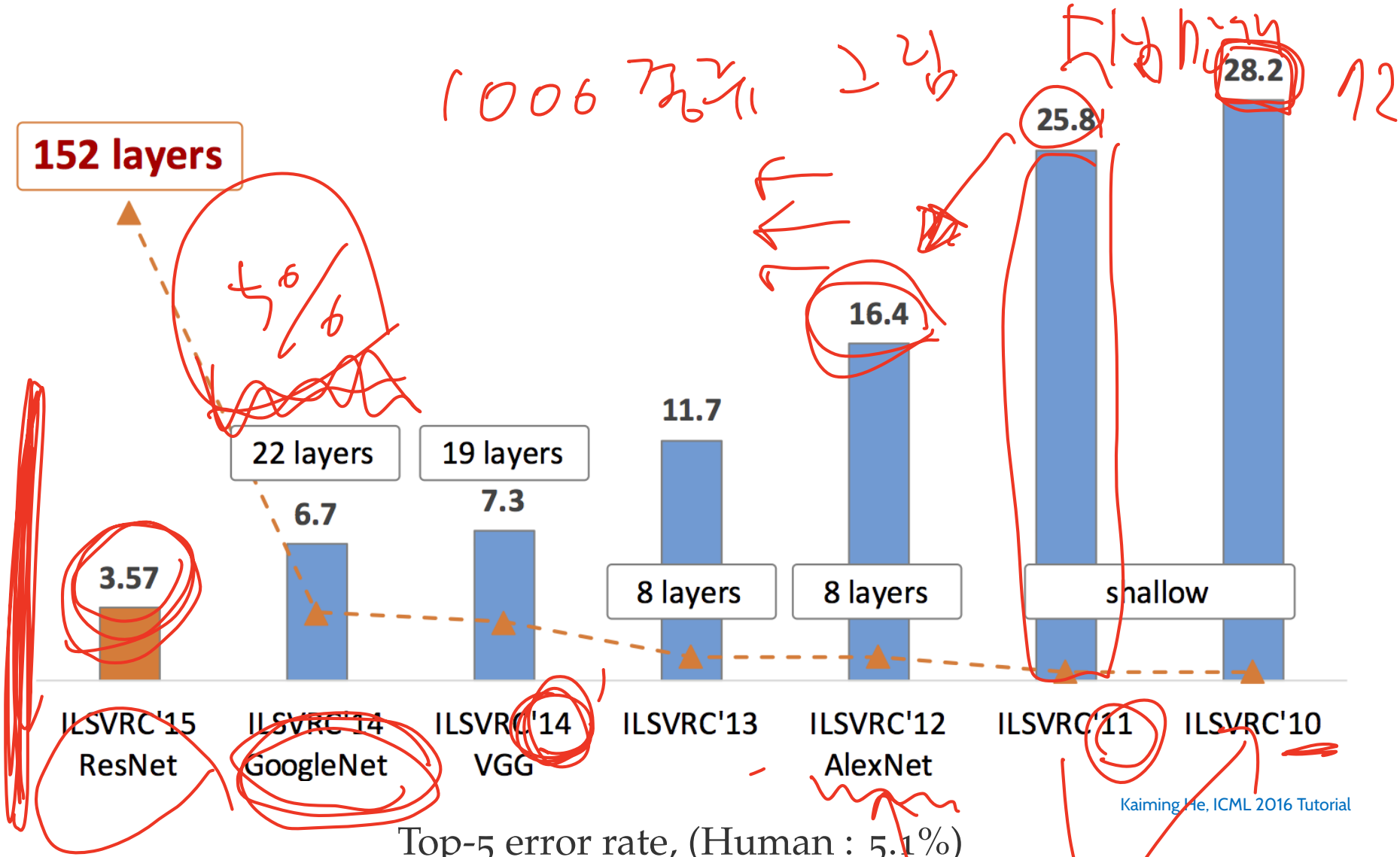
Popular CNN Architectures

Visual Recognition Problems



ImageNet Classification Challenge

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Top-5 error rate, (Human : 5.1%)

AlexNet

Slide adapted from CS231n by F. F. Li et.al.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, NIPS 2012

- Winner of ILSVRC'12 with 9%p improvement over the previous year.
- 8 layers : convolutional, max-pooling, normalization, and fully-connected.

AlexNet Architecture

Slide adapted from CS231n by F. F. Li et.al.

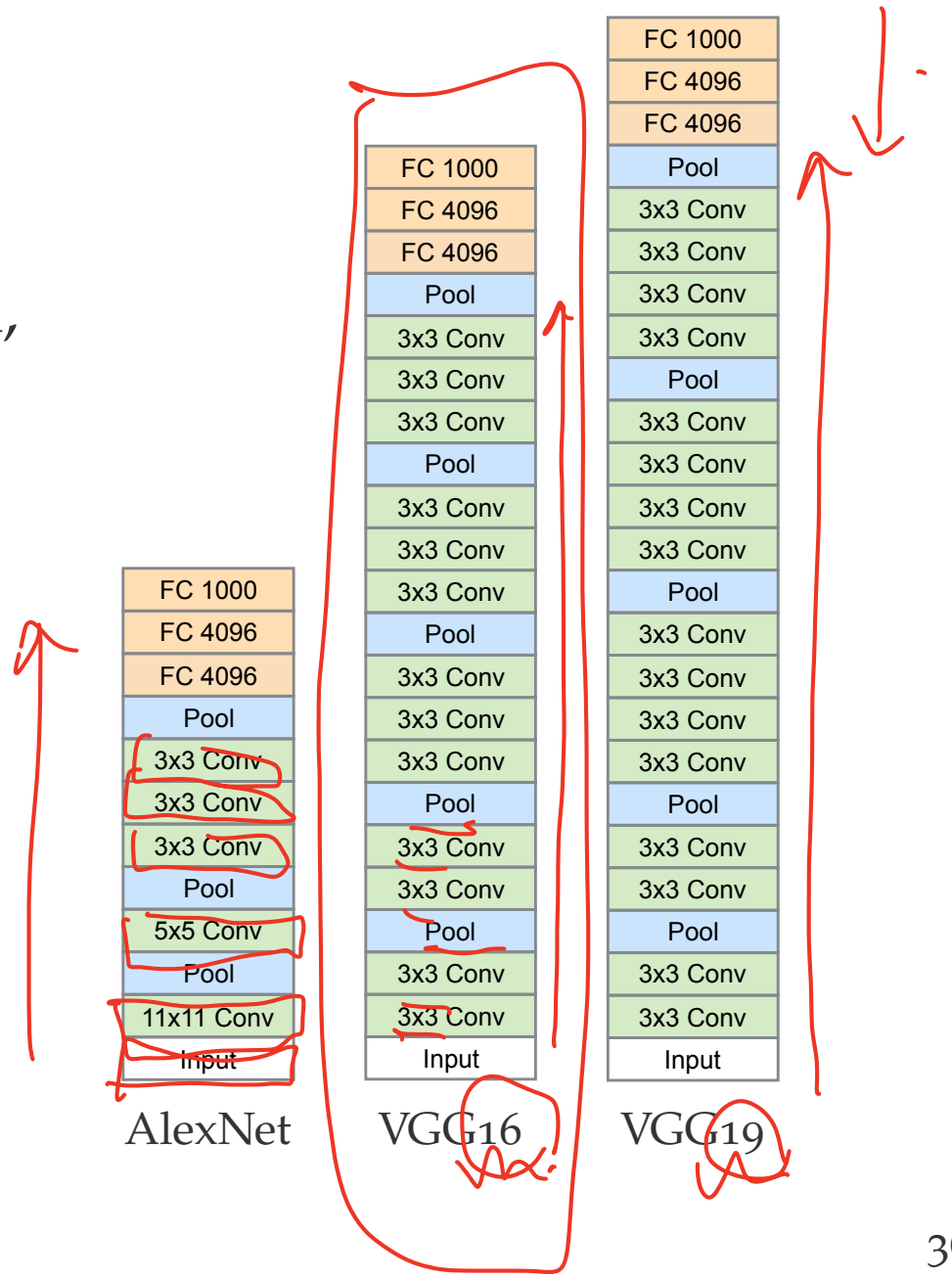
	Feature Map Size	Kernel Size	#Parameters	Memory
Input	$227 \times 227 \times 3$			150K
Conv1	$55 \times 55 \times 48$x2	$(11 \times 11 \times 3)$ S4 $\times$ 48x2	35K	145Kx2
Norm1	$27 \times 27 \times 48$x2			35Kx2
MaxPool1	$27 \times 27 \times 48$x2	(3×3) S2		35Kx2
Conv2	$27 \times 27 \times 128$x2	$(5 \times 5 \times 48)$ P2 $\times$ 128x2	307K	93Kx2
Norm2	$13 \times 13 \times 128$x2			22Kx2
MaxPool2	$13 \times 13 \times 128$x2	(3×3) S2		22Kx2
Conv3	$13 \times 13 \times 192$x2	$(3 \times 3 \times 256)$ P1 $\times$ 192x2	885K	32Kx2
Conv4	$13 \times 13 \times 192$x2	$(3 \times 3 \times 192)$ P1 $\times$ 192x2	664K	32Kx2
Conv5	$13 \times 13 \times 192$x2	$(3 \times 3 \times 192)$ P1 $\times$ 128x2	442K	32Kx2
MaxPool3	$6 \times 6 \times 128$x2	(3×3) S2		5Kx2
FC6	4096	$(6 \times 6 \times 256) \times 4096$	37,749K	4K
FC7	4096	$(4096) \times 4096$	16,777K	4K
FC8	1000	$(4096) \times 1000$	4,096K	1K

VGGNet

Slide adapted from CS231n by F. F. Li et.al.

K. Simonyan and A. Zisserman, arXiv 2014, ICLR 2015

- Winner of ILSVRC'14.
- Deeper networks with smaller filters.
  - 3 × 3 Conv
  - 2 × 2 (S2) MaxPool



VGG-16 Architecture

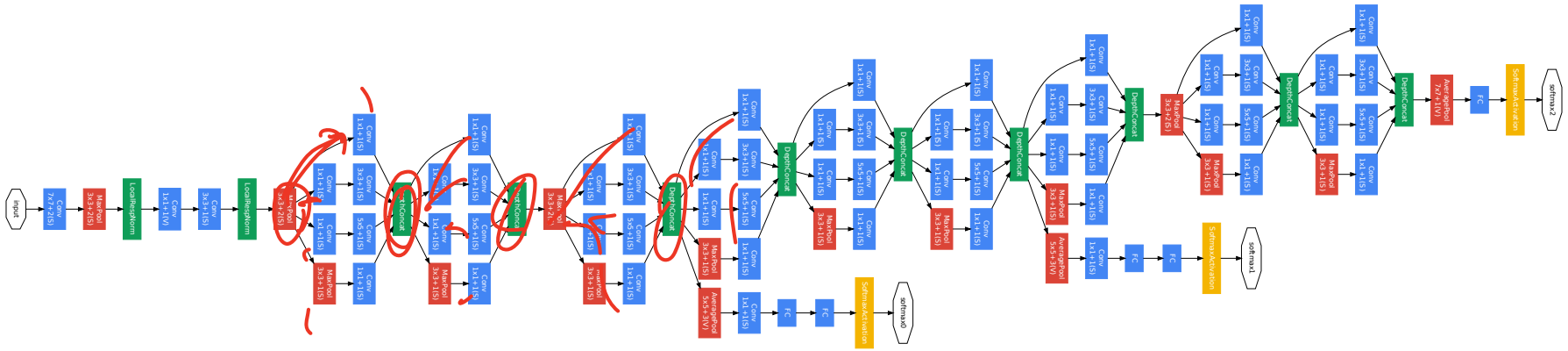
Slide adapted from CS231n by F. F. Li et.al.

	Feature Map Size	Kernel Size	#Parameters	Memory
Input	$227 \times 227 \times 3$			150K
Conv 1-1	$224 \times 224 \times 64$	$(3 \times 3 \times 3)$ P1 $\times$ 64	1.7K	3,211K
Conv 1-2	$224 \times 224 \times 64$	$(3 \times 3 \times 64)$ P1 $\times$ 64	37K	3,211K
Pool 1	$112 \times 112 \times 64$		(2×2) S2	803K
Conv 2-1	$112 \times 112 \times 128$	$(3 \times 3 \times 64)$ P1 $\times$ 128	74K	1,606K
Conv 2-2	$112 \times 112 \times 128$	$(3 \times 3 \times 128)$ P1 $\times$ 128	147K	1,606K
Pool 2	$56 \times 56 \times 128$	(2×2) S2		401K
Conv 3-1	$56 \times 56 \times 256$	$(3 \times 3 \times 128)$ P1 $\times$ 256	295K	803K
Conv 3-2	$56 \times 56 \times 256$	$(3 \times 3 \times 256)$ P1 $\times$ 256	590K	803K
Conv 3-3	$56 \times 56 \times 256$	$(3 \times 3 \times 256)$ P1 $\times$ 256	590K	803K
Pool 3	$28 \times 28 \times 256$	(2×2) S2		200K
Conv 4-1	$28 \times 28 \times 512$	$(3 \times 3 \times 256)$ P1 $\times$ 512	1,180K	401K
Conv 4-2	$28 \times 28 \times 512$	$(3 \times 3 \times 512)$ P1 $\times$ 512	2,359K	401K
Conv 4-3	$28 \times 28 \times 512$	$(3 \times 3 \times 512)$ P1 $\times$ 512	2,359K	401K
Pool 4	$14 \times 14 \times 512$	(2×2) S2		100K
Conv 5-1	$14 \times 14 \times 512$	$(3 \times 3 \times 512)$ P1 $\times$ 512	2,359K	100K
Conv 5-2	$14 \times 14 \times 512$	$(3 \times 3 \times 512)$ P1 $\times$ 512	2,359K	100K
Conv 5-3	$14 \times 14 \times 512$	$(3 \times 3 \times 512)$ P1 $\times$ 512	2,359K	100K
Pool 5	$7 \times 7 \times 512$	(2×2) S2		25K
FC 6	4096	$(7 \times 7 \times 512) \times 4096$	102,760K	4K
FC 7	4096	$(4096) \times 4096$	16,777K	4K
FC 8	1000	$(4096) \times 1000$	4,096K	1K

GoogLeNet

Slide adapted from CS231n by F. F. Li et.al.

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



Szegedy et.al. CVPR 2015

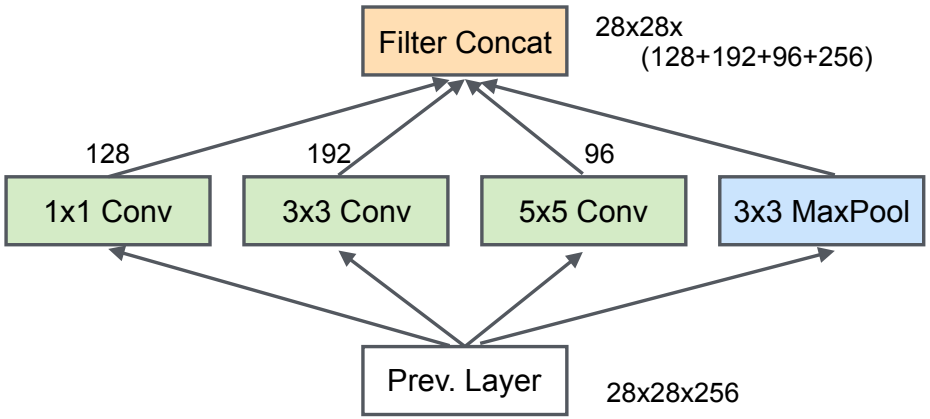
[Convolution Pool Softmax Norm/DepthConcat]

- Inception module : network within a network
- Drastic reduction in parameters using smaller convolutions
- Auxiliary classification outputs to inject additional gradient at lower layers.

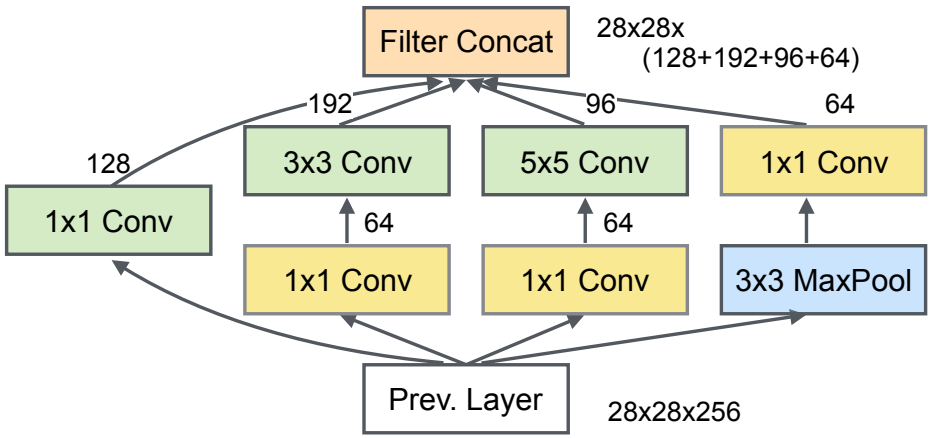
Inception Module

Slide adapted from CS231n by F. F. Li et.al.

Naive Inception



Inception with dimensionality reduction



- Concat layer concatenates all inputs depthwise.
- For the given input, output, and channel prameters
  Naive inception needs $854M$ operations to output $529K$.
- After installing three 1×1 bottleneck layers,
  it needs $358M$ operations to output $376K$.

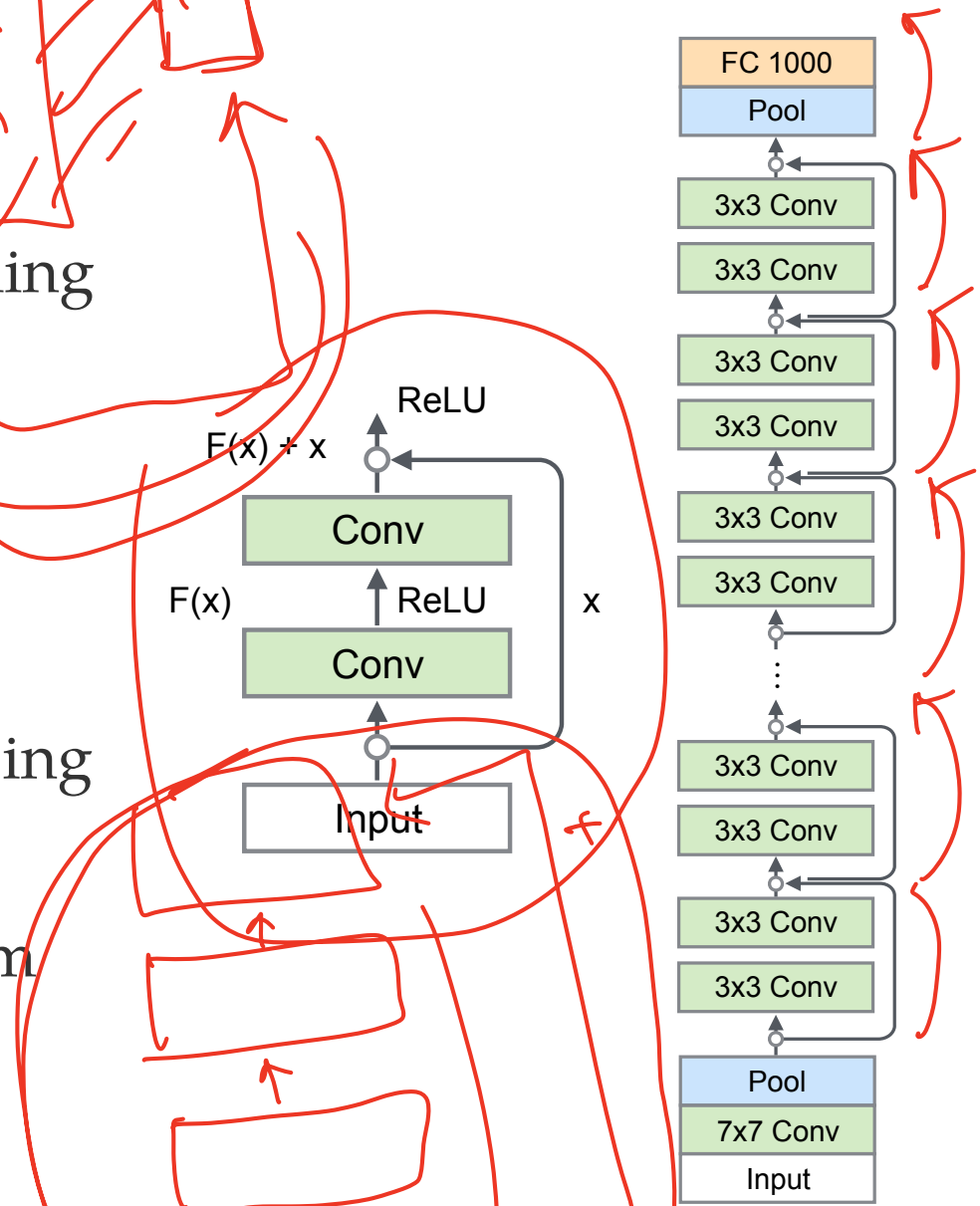
ResNet

Slide adapted from CS231n by F. F. Li et.al.

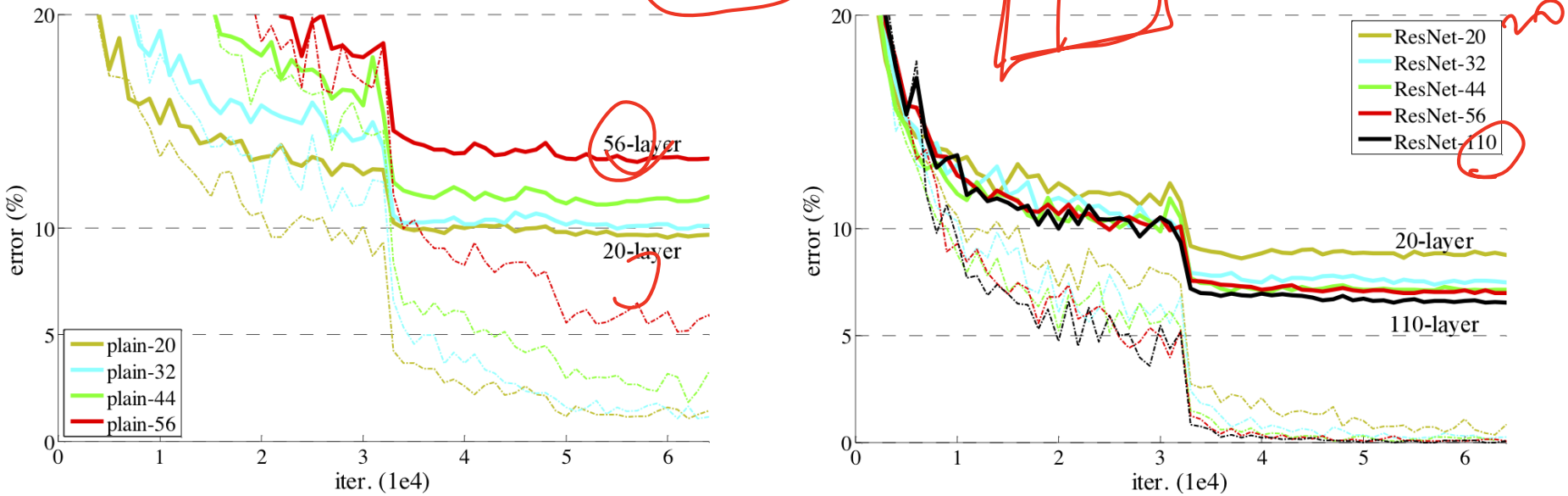
K. He, et.al., Deep Residual Learning for Image Recognition, CVPR 2016

Very deep networks using residual connections.

- 152-layer model for ImageNet.
- Idea : learn the residual mapping instead of direct mapping.
- No vanishing gradient problem



ResNet Training/Test Errors



He et.al. CVPR 2016

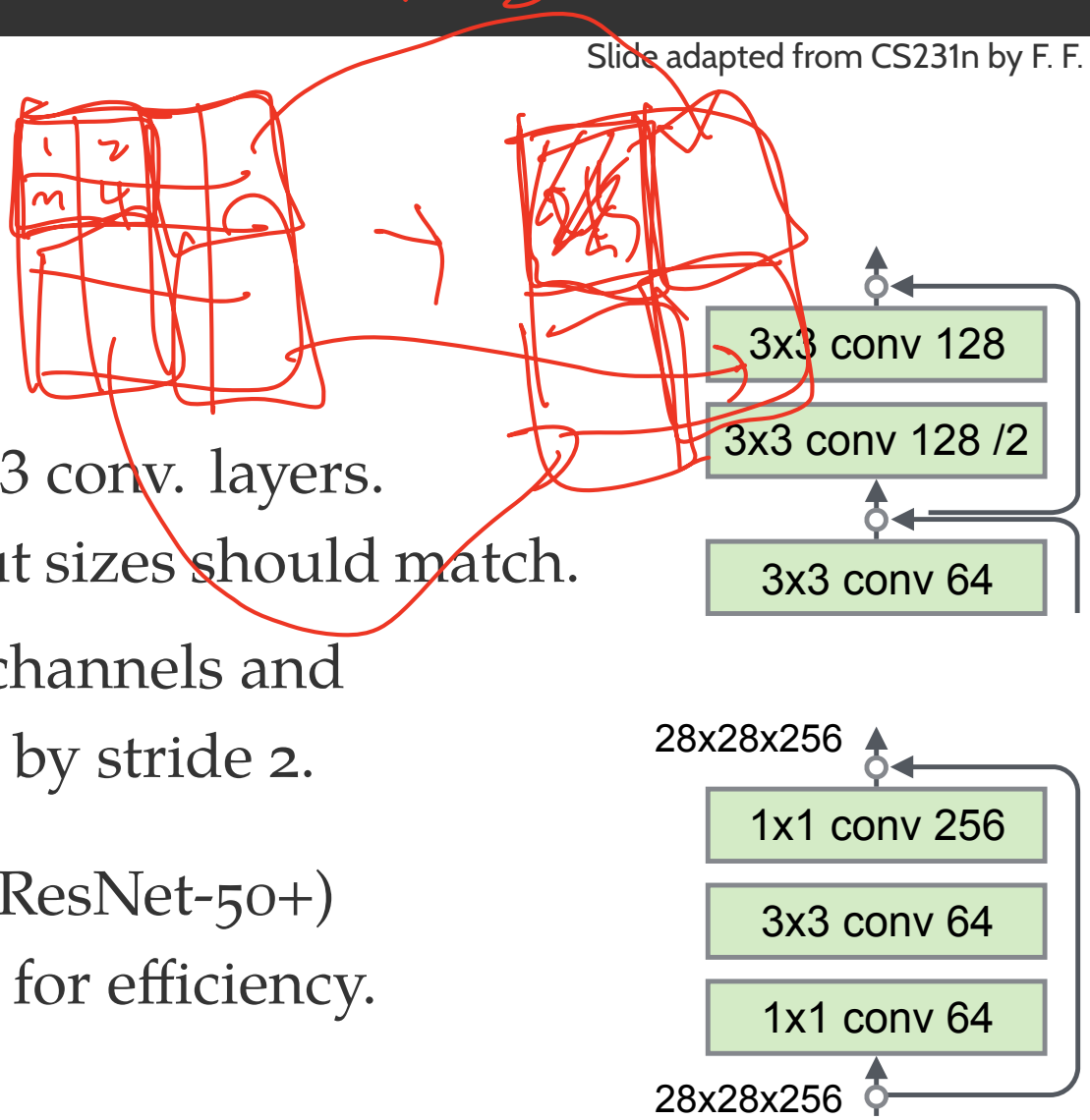
[CIFAR-10 dashed: training error / solid : test error]

- Both training and test errors of plain networks do not decrease as the depth increases.
- With residual blocks, deeper networks give smaller errors.

ResNet Architecture

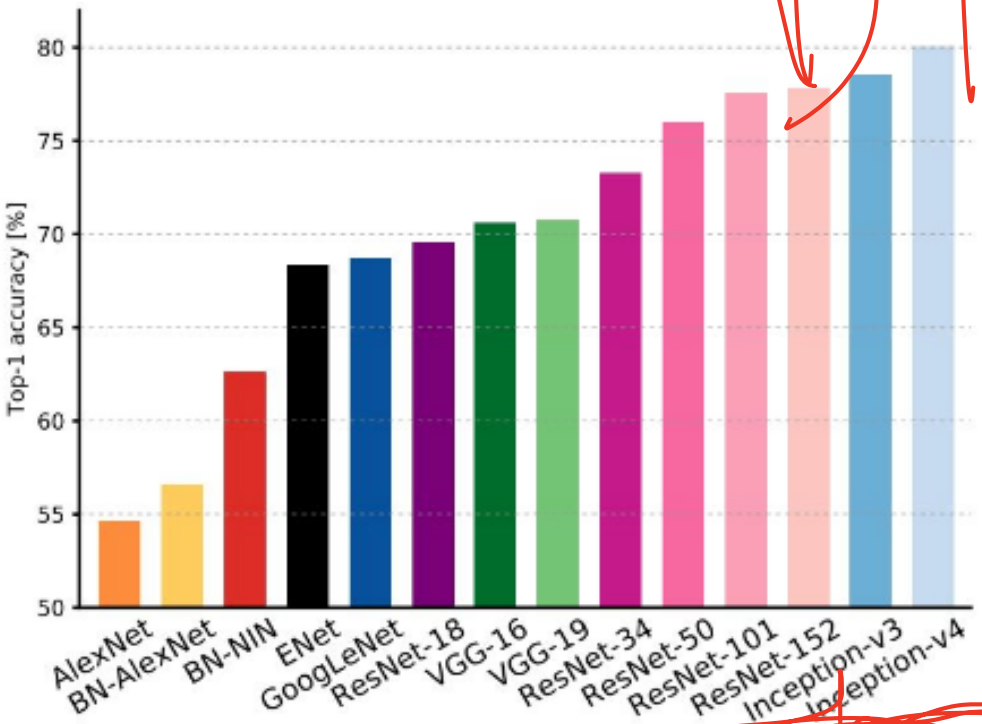
Slide adapted from CS231n by F. F. Li et.al.

- Stack residual blocks.
- Each block as two 3×3 conv. layers.
  Note: input and output sizes should match.
- Periodically, double #channels and downsample spatially by stride 2.
- For deeper networks (ResNet-50+) use 'bottleneck' layers for efficiency.



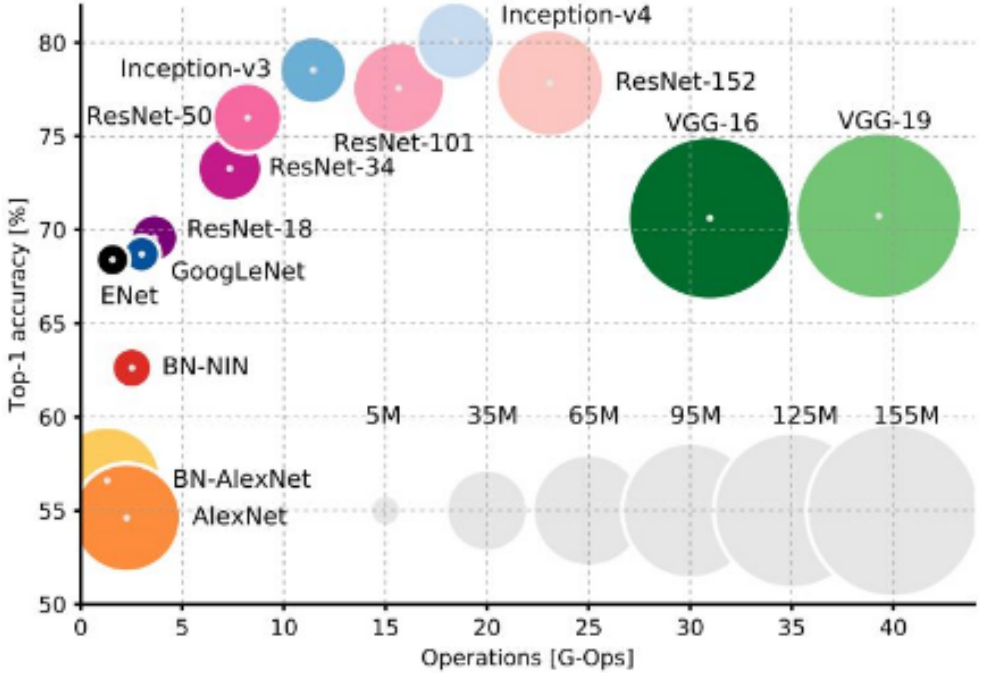
Network Performance and Complexity

Canziani et.al. An Analysis of Deep Neural Network Models for Practical Applications, arXiv 2017



Canziani et.al. arXiv 2017

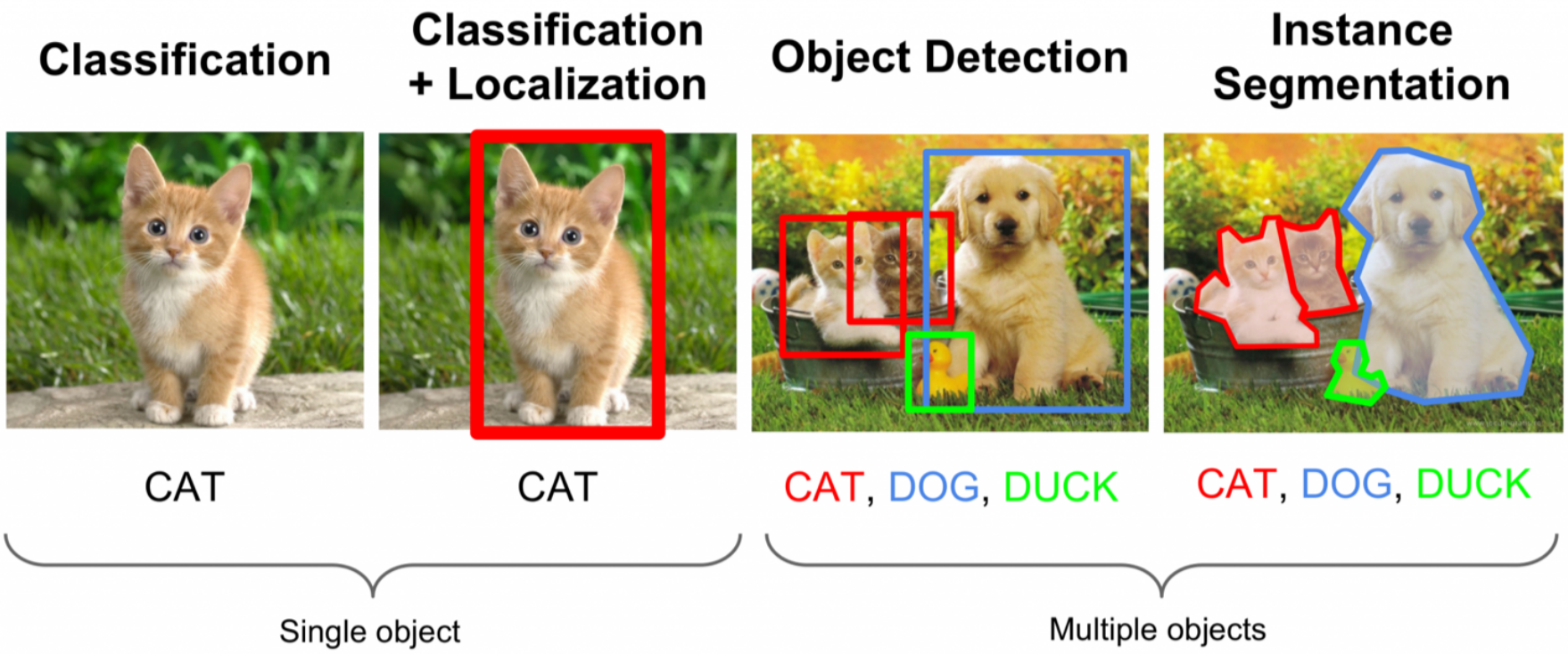
Top1 accuracy



Canziani et.al. arXiv 2017

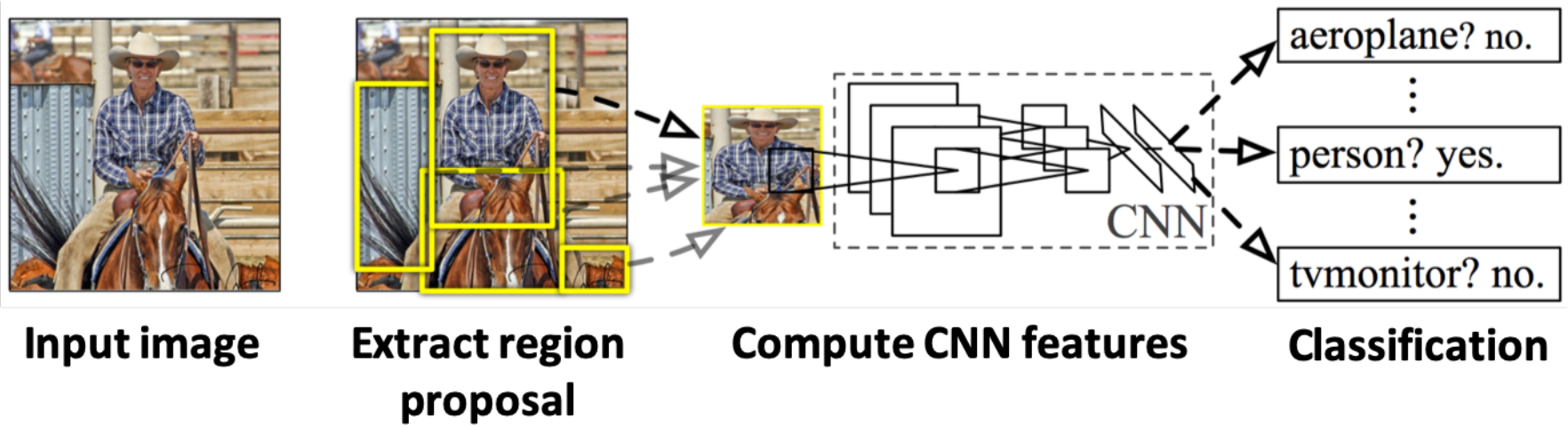
Top 1 accuracy vs. Operations

Object Detection



R-CNN

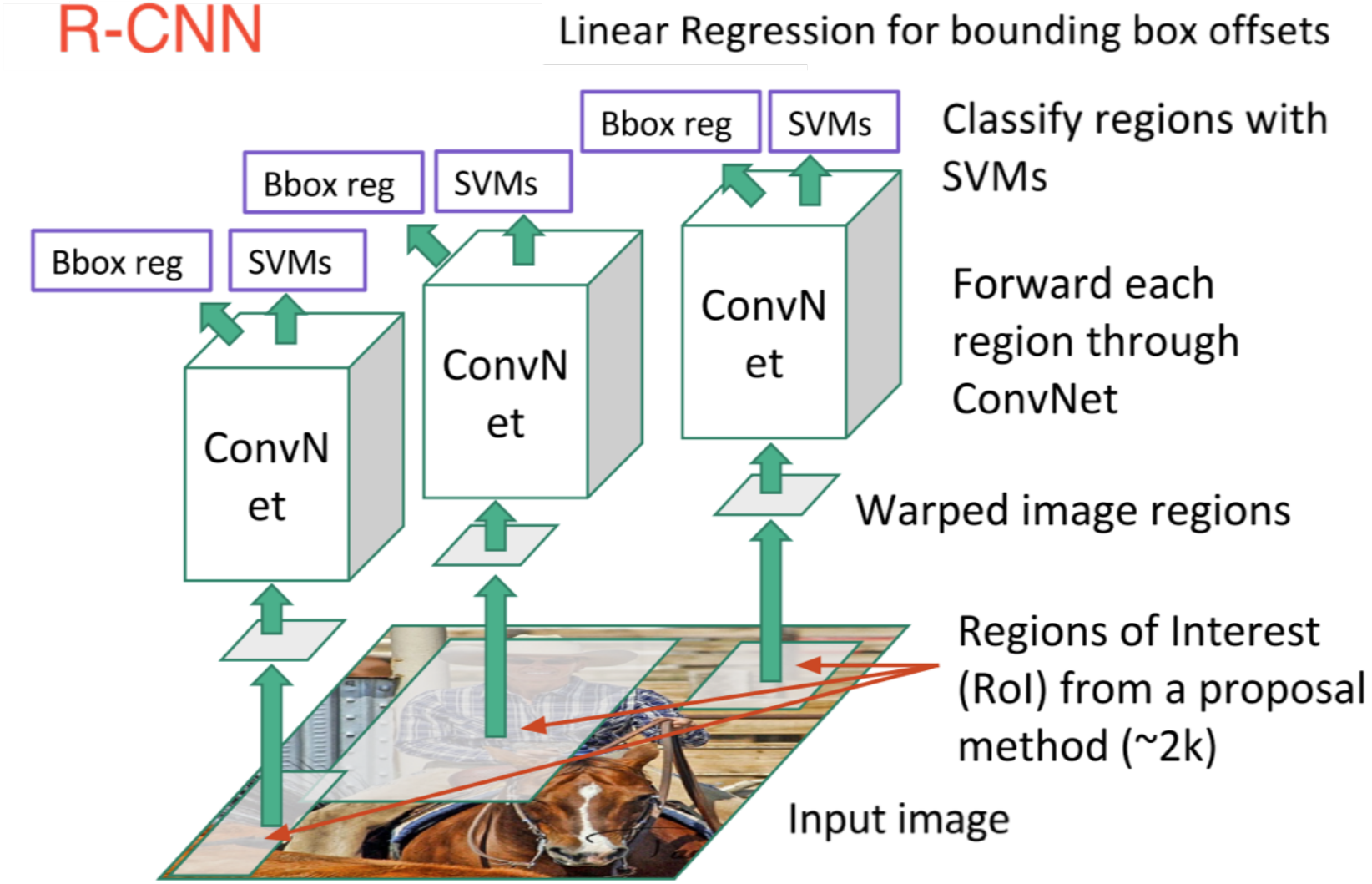
Girshick et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014



© Girshick et al.

- Classify the content in proposed regions
- Region proposal + Image classification

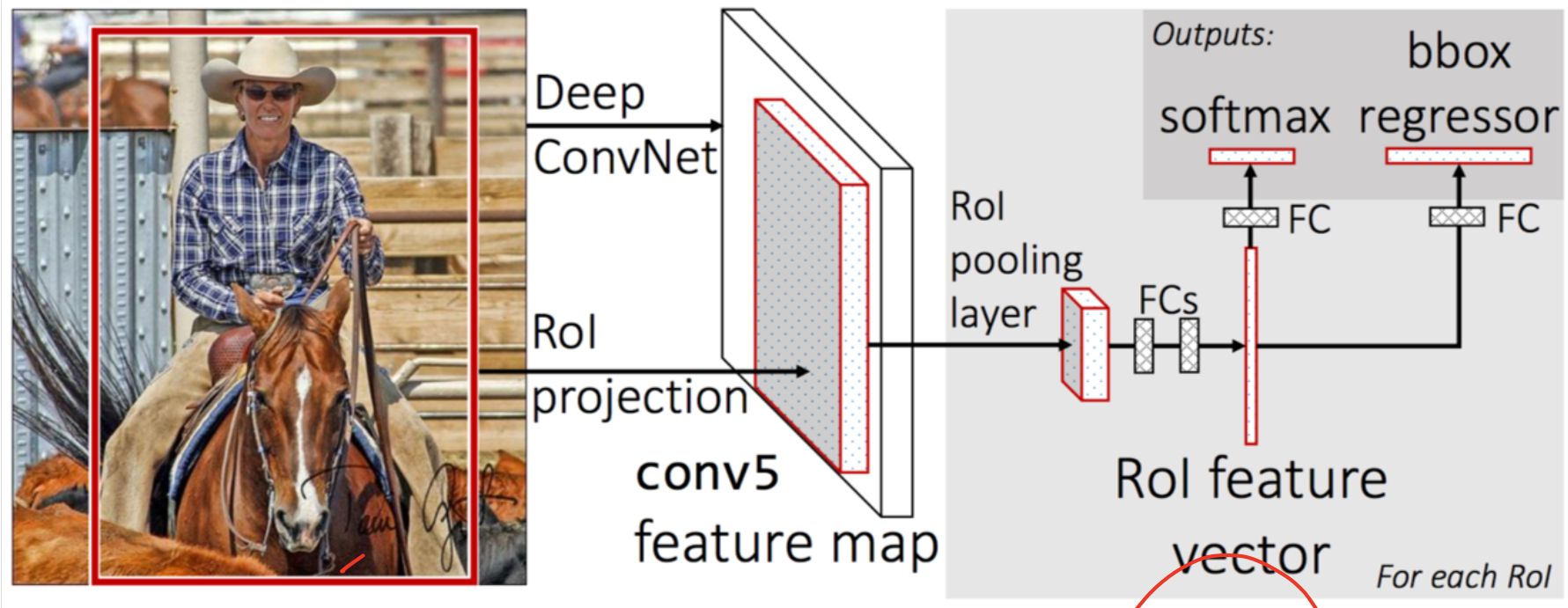
R-CNN Architecture



https://medium.com/towards-data-science/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

Fast R-CNN

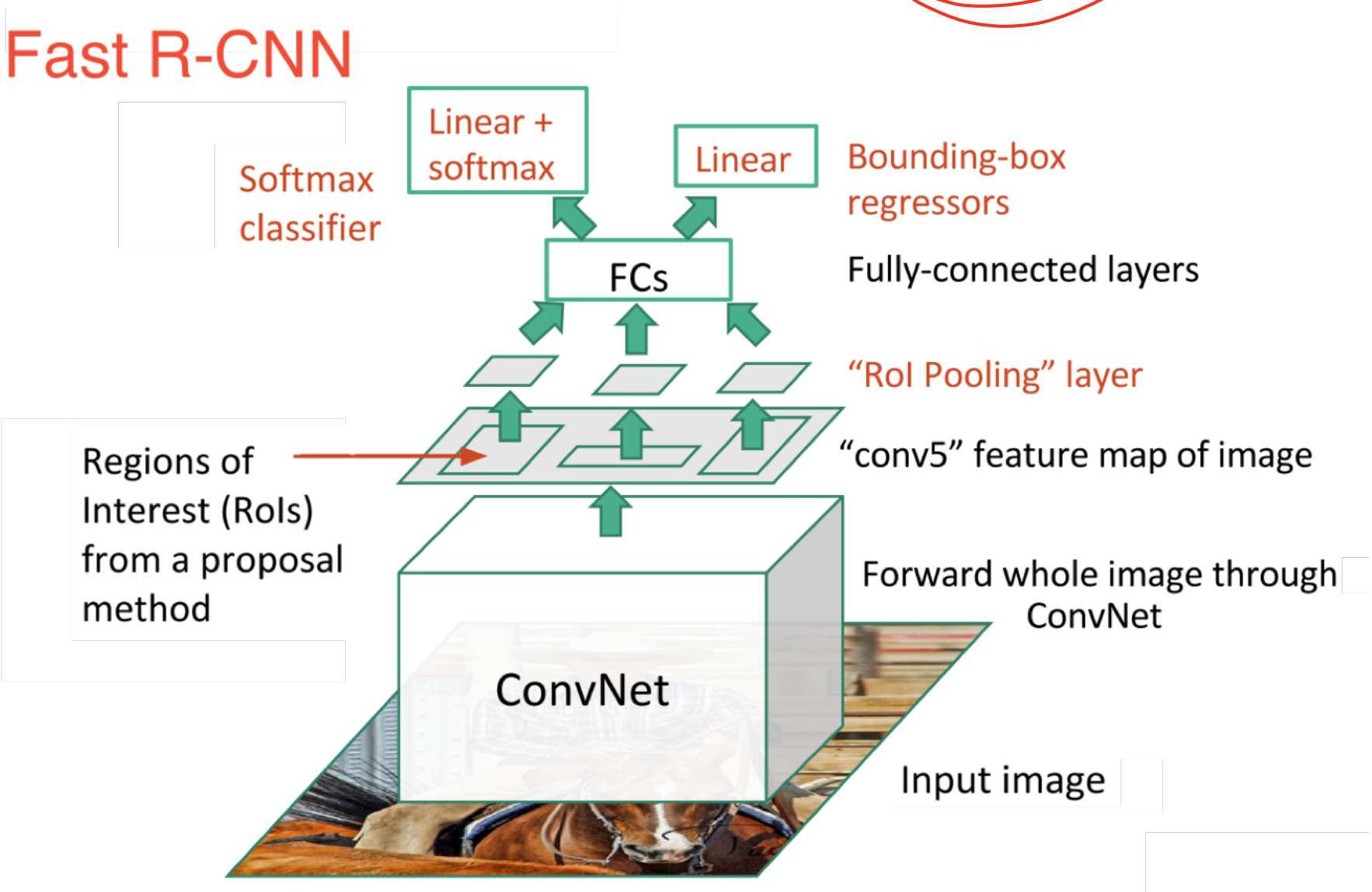
Girshick, Fast R-CNN, ICCV 2015



- Single feature computation
- ROI pooling using object proposals

code

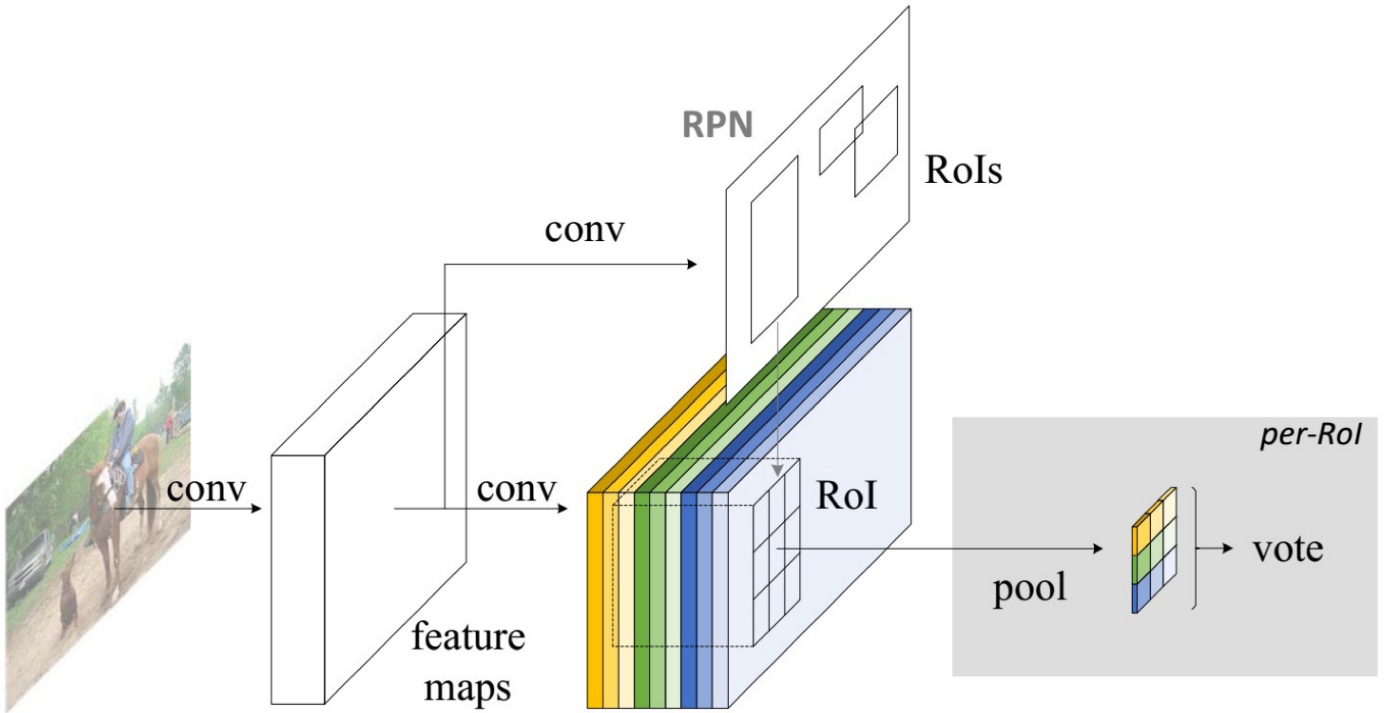
Fast R-CNN Architecture



https://medium.com/towards-data-science/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

Faster R-CNN Architecture

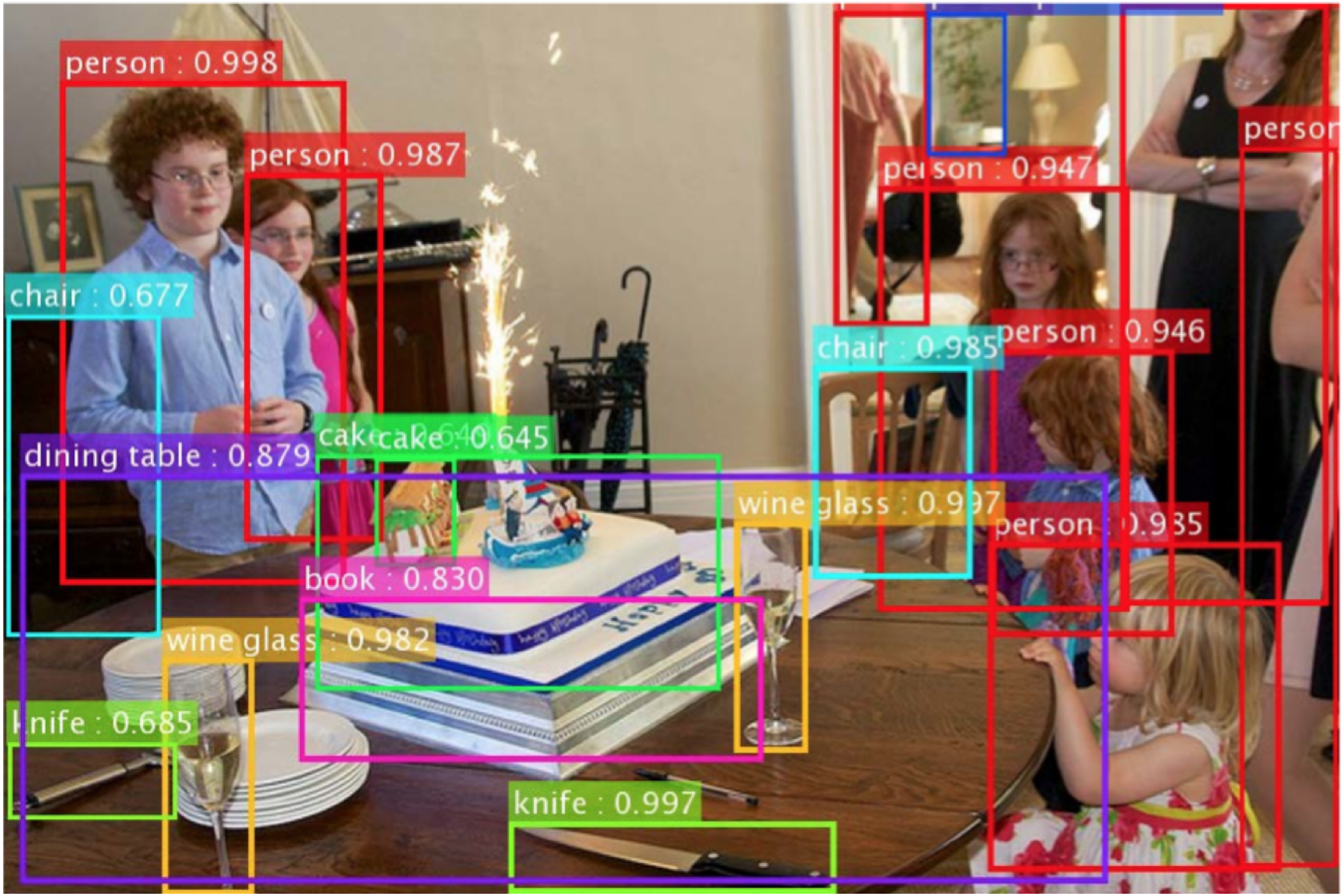
Ren et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015



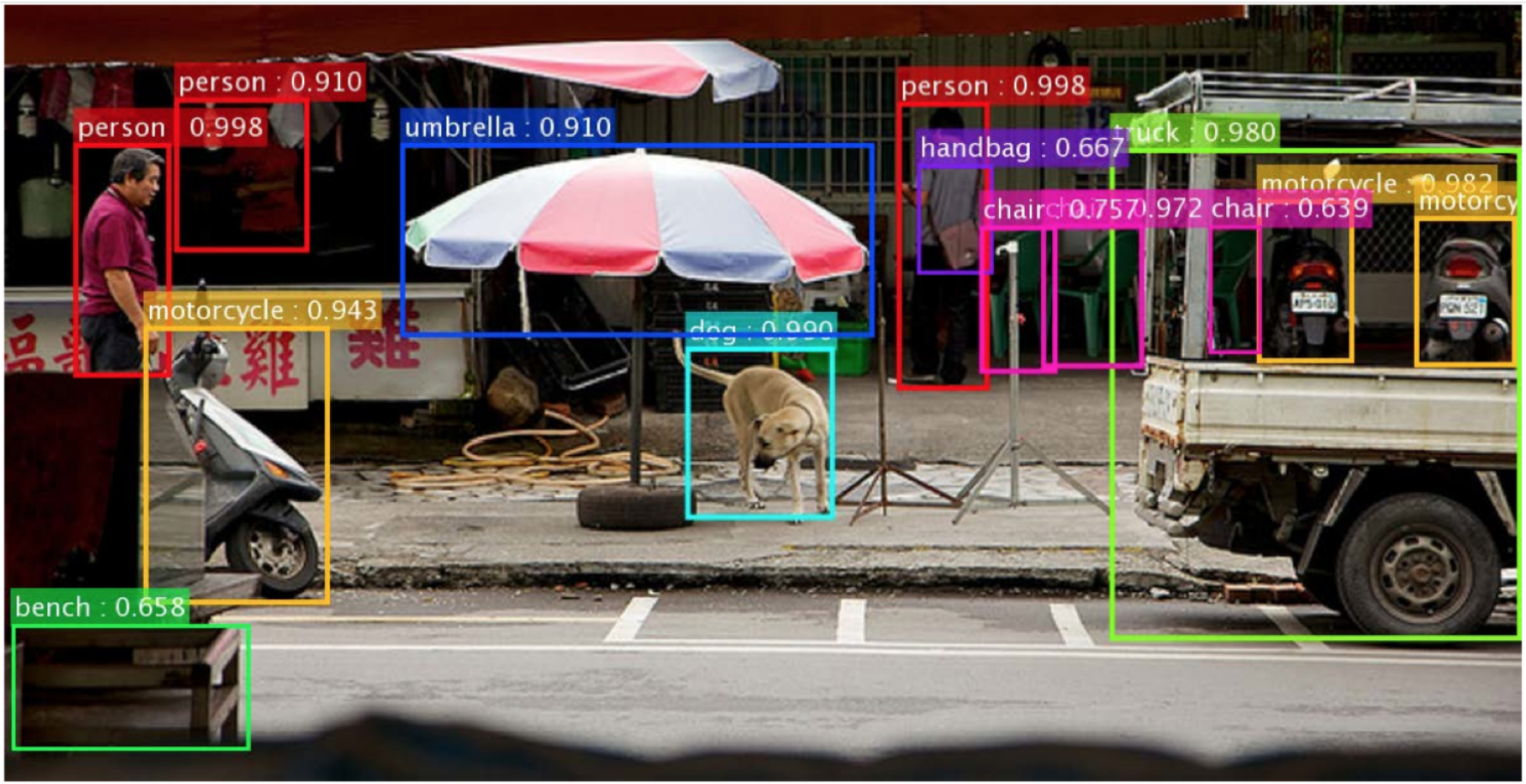
https://medium.com/towards-data-science/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

- Region proposal network inside

Faster R-CNN Result



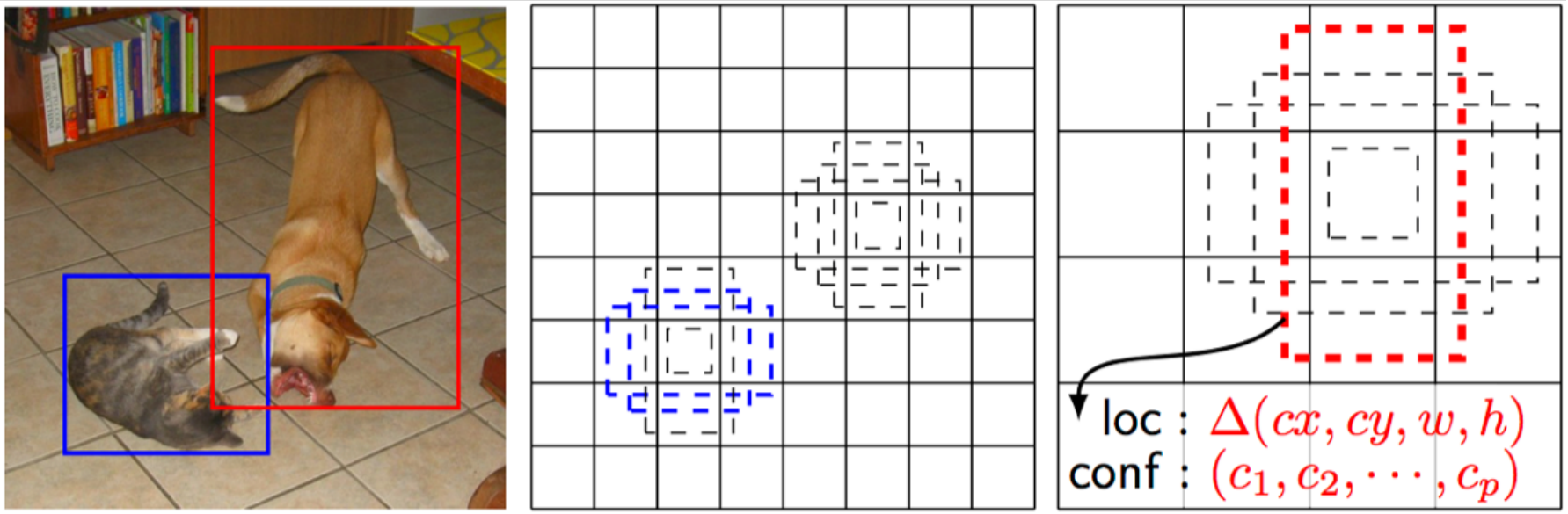
Faster R-CNN Result



© Ren et al.

Single Shot Multibox Detector (SSD)

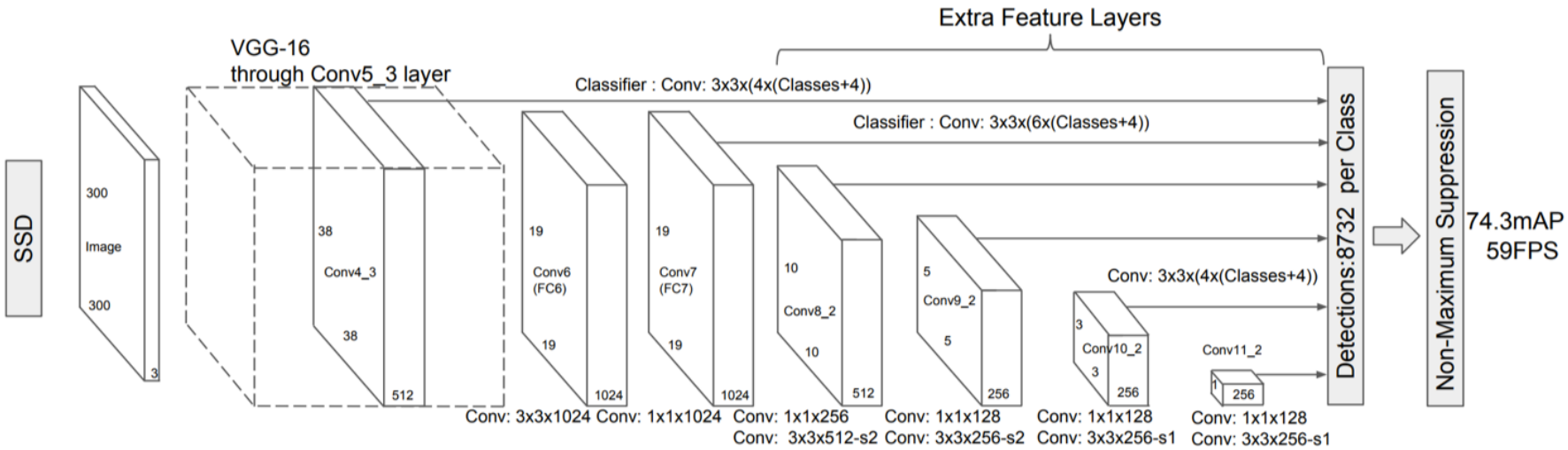
Liu, et al., SSD: Single Shot MultiBox Detector, ECCV 2016



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

- No object proposal
- A set of default boxes at each feature map location

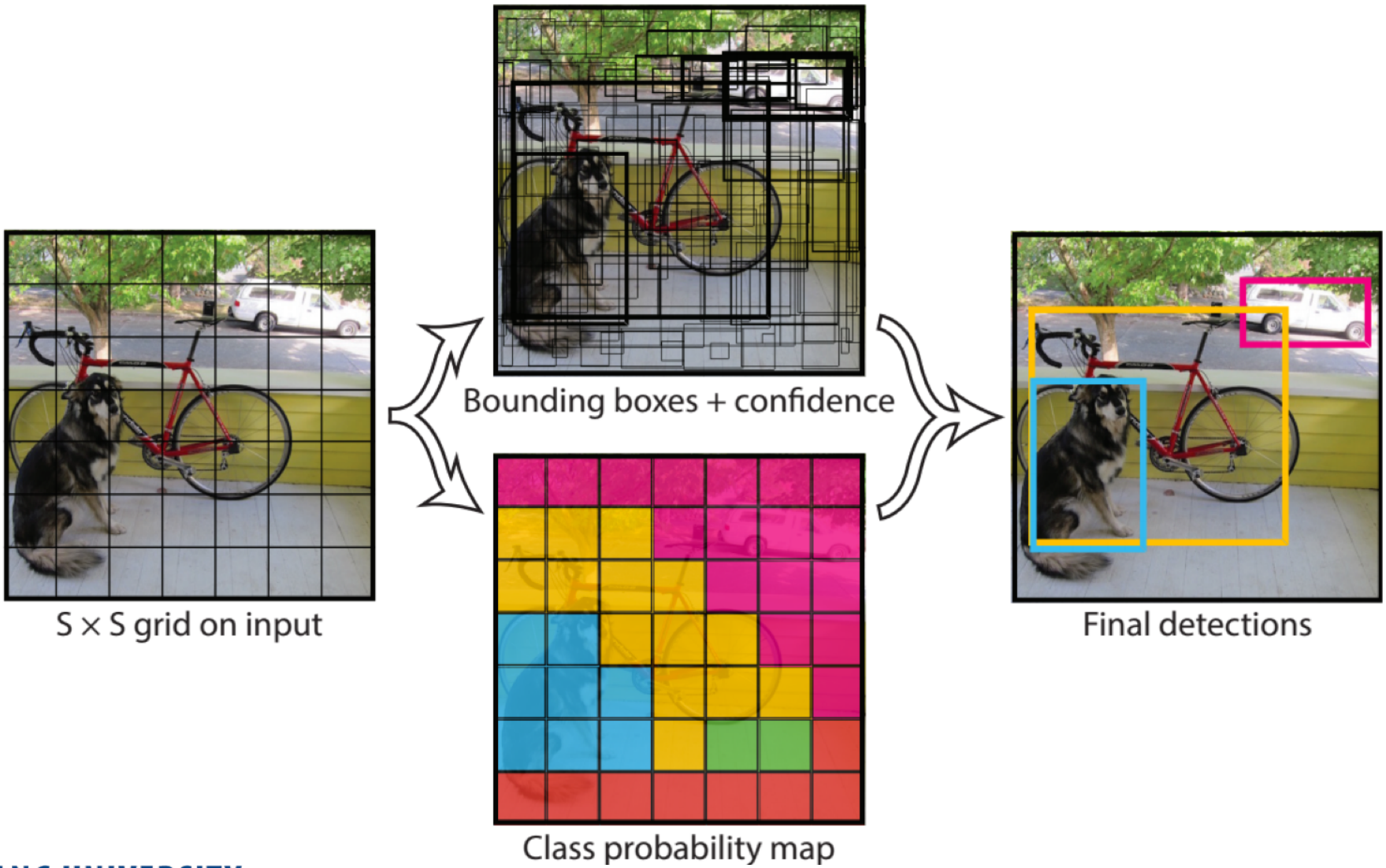
SSD Architecture



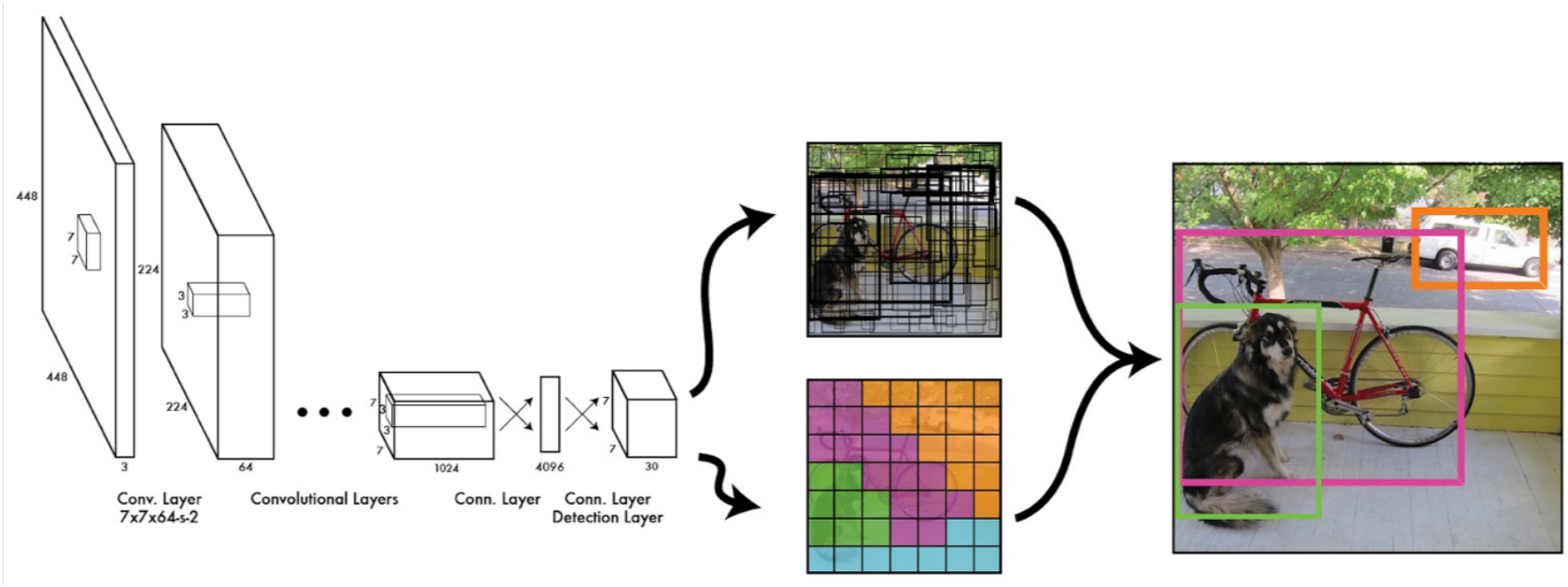
© Liu et al.

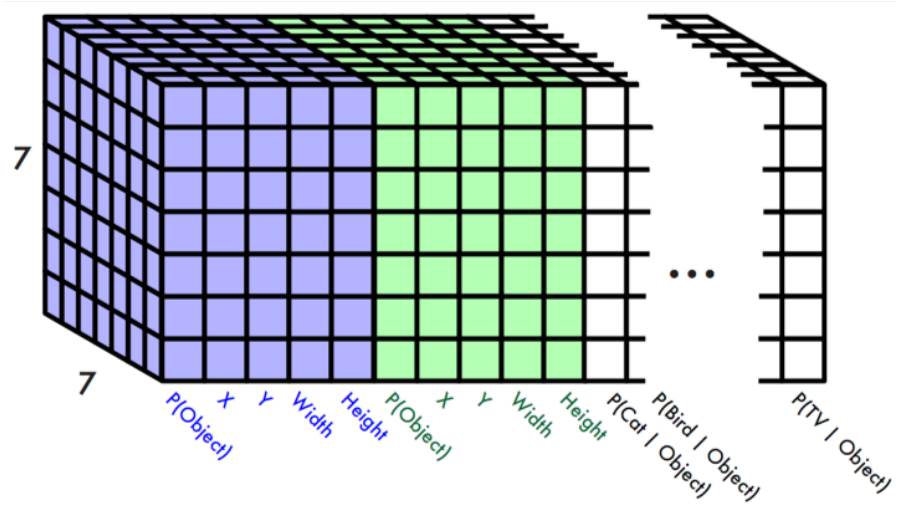
You Only Look Once (YOLO)

Redmon et al., You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

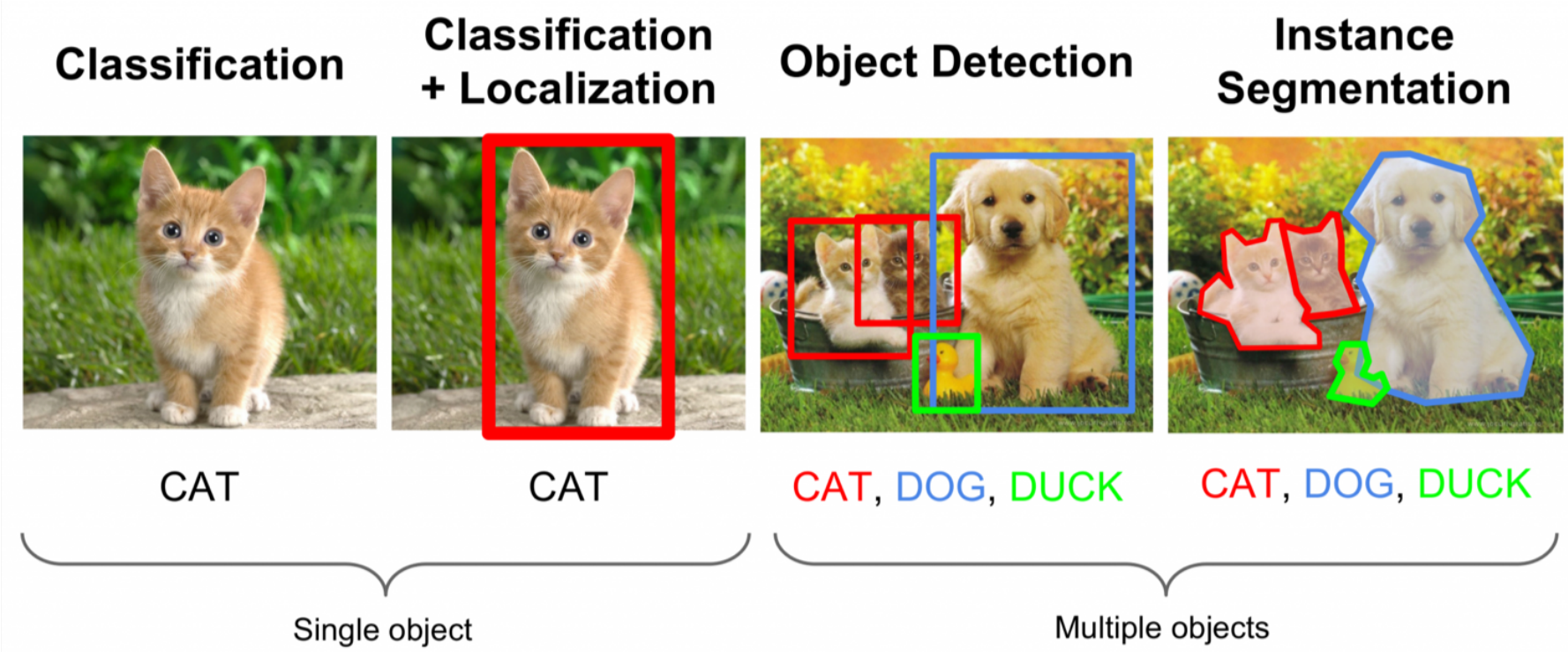


YOLO Architecture

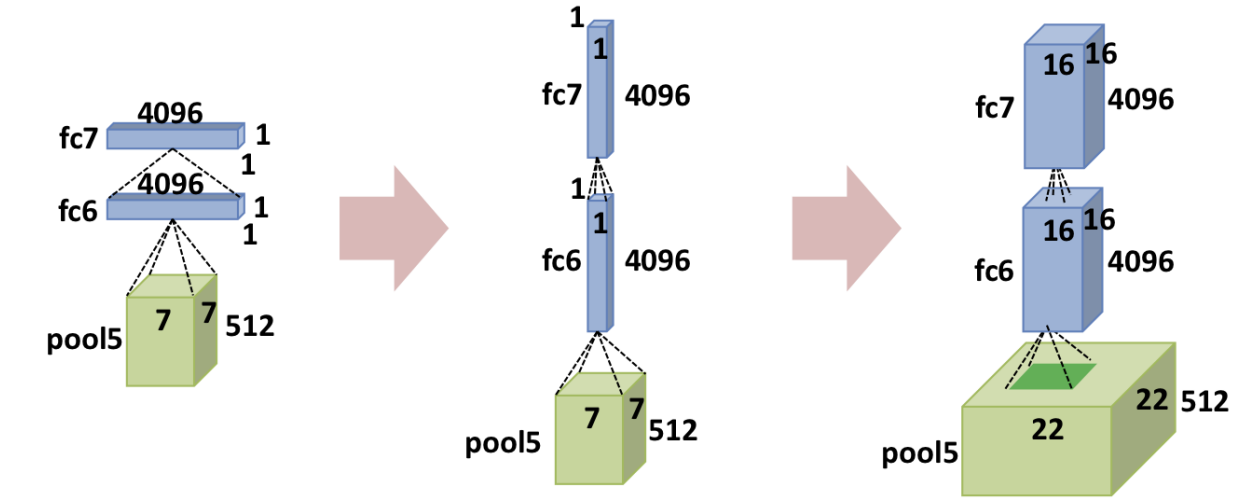




Semantic Segmentation

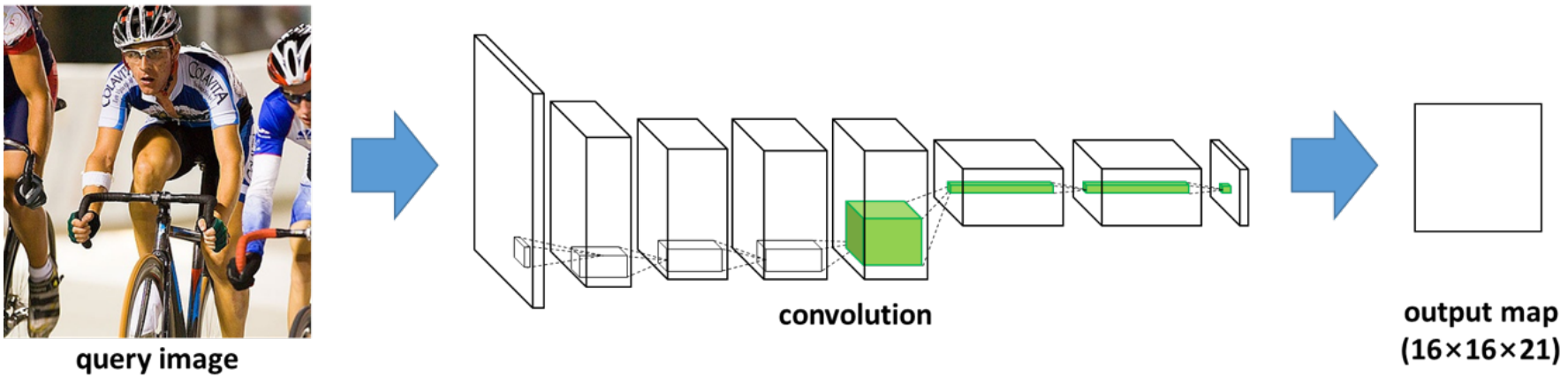


Fully Convolutional Network (FCN)



Fully connected layers | Convolution layers | For the larger Input field

© Bohyung Han

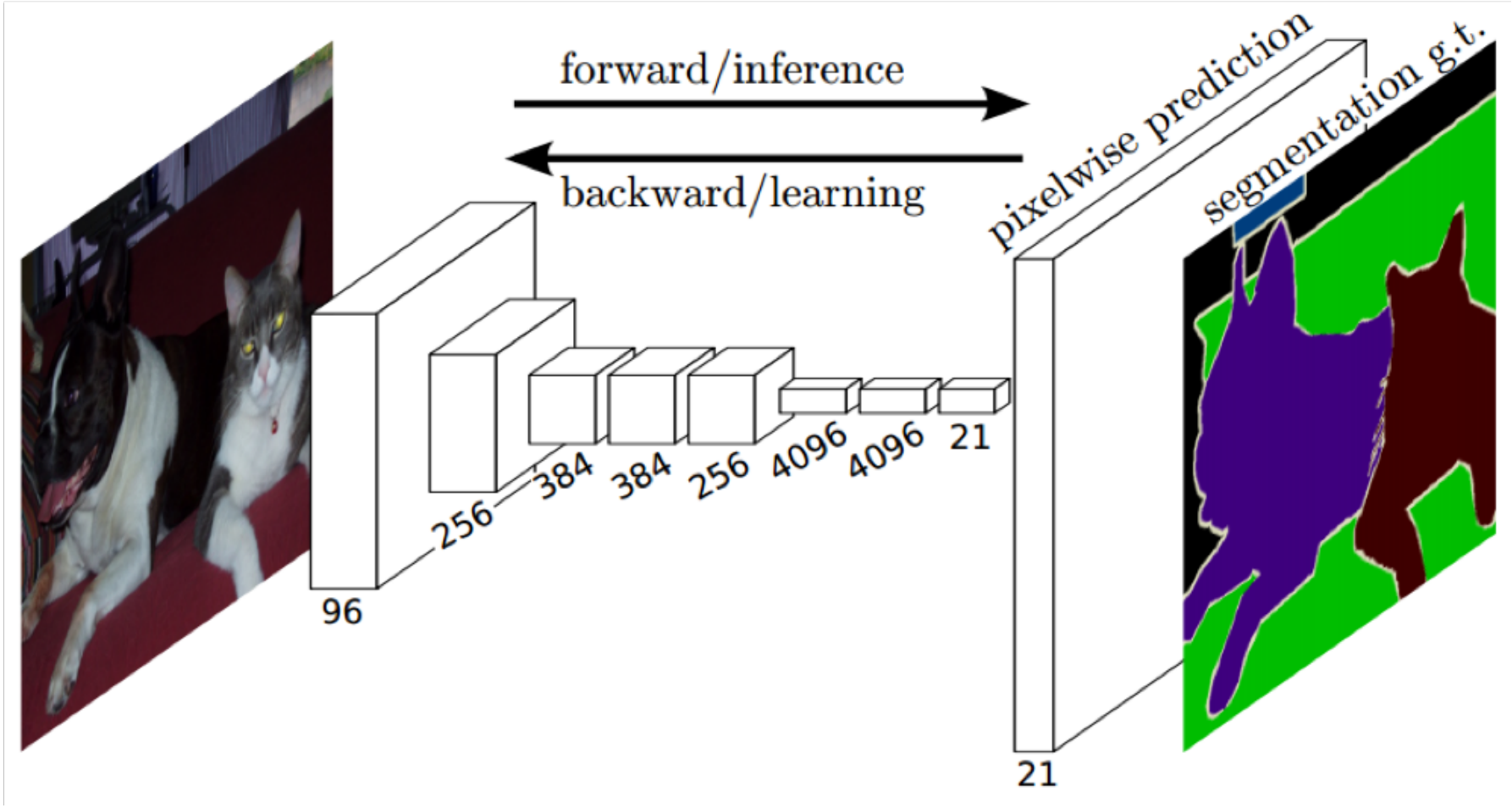


© Girshick et al.

- Fully connected layer = convolution with a large filter

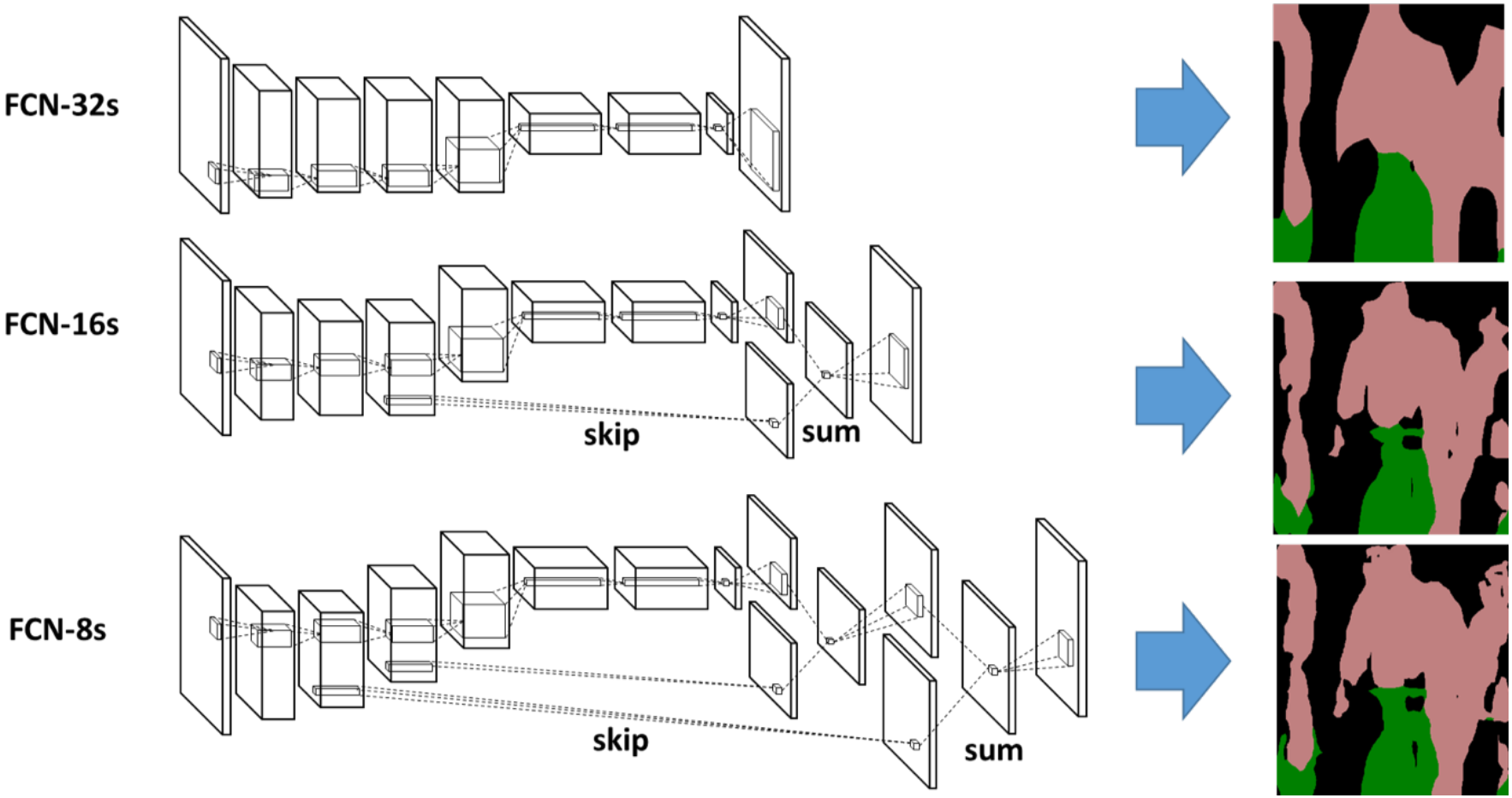
FCN Architecture

Long et al., Fully Convolutional Network for Semantic Segmentation, CVPR 2015



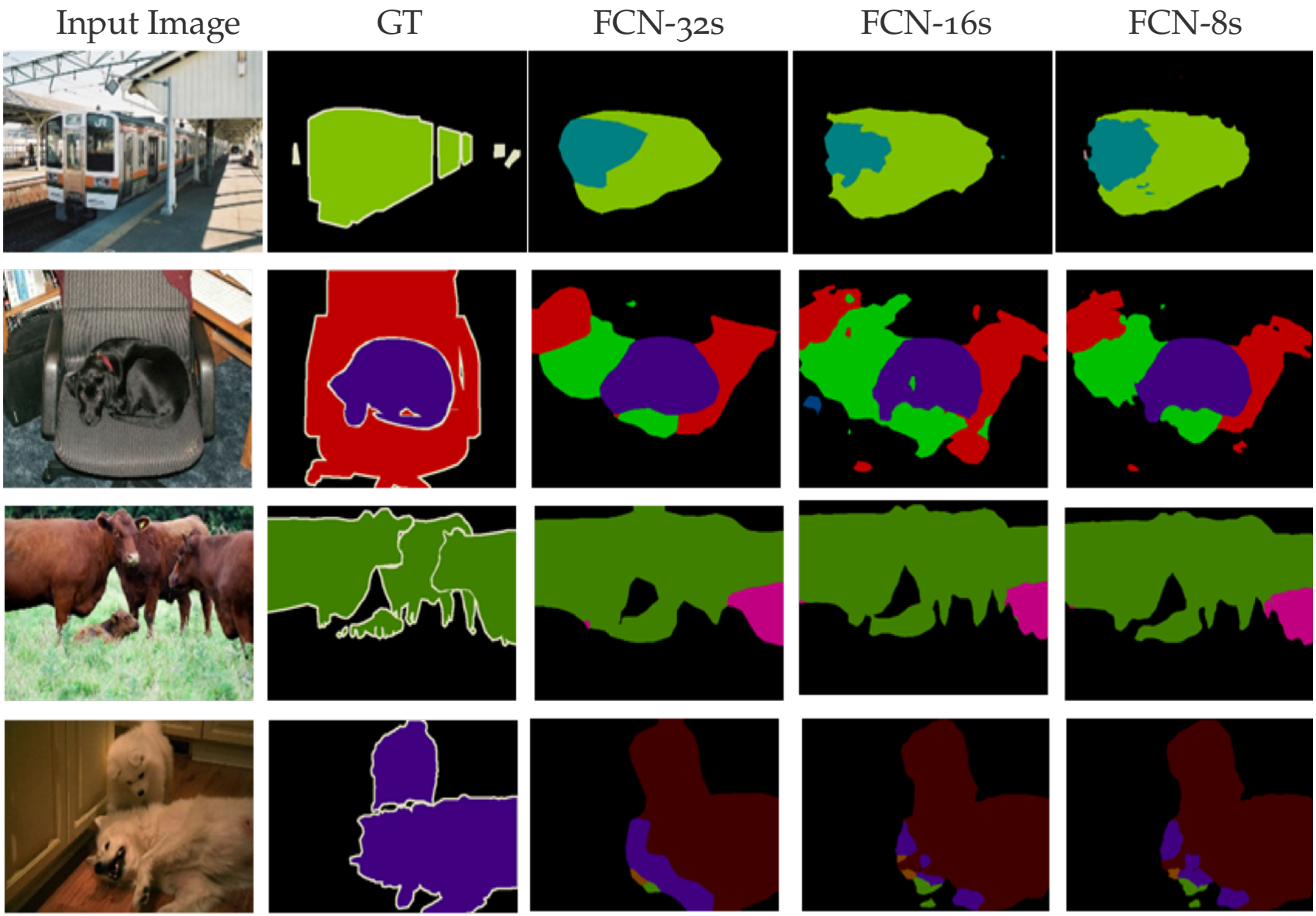
© Long et al.

FCN Architecture



© Long et al.

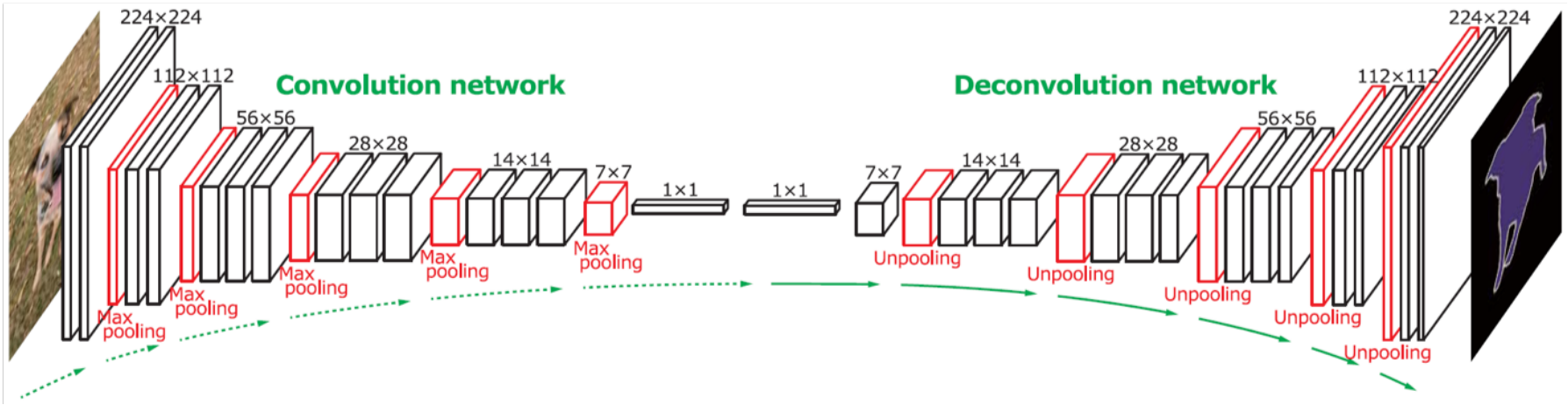
FCN Result



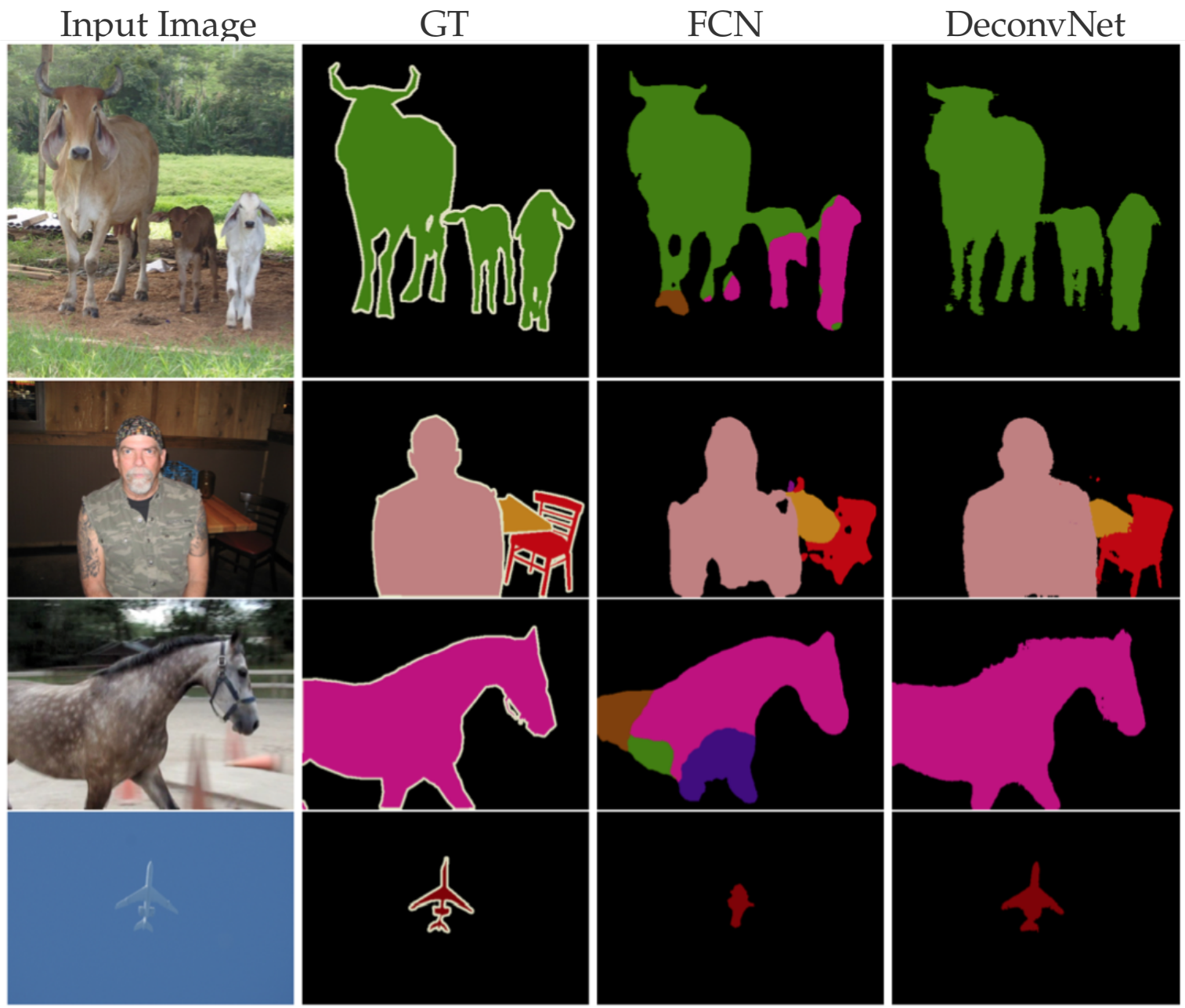
© Long et al.

DeconvNet Architecture

Noh et al., Learning Deconvolution Network for Semantic Segmentation, ICCV 2015



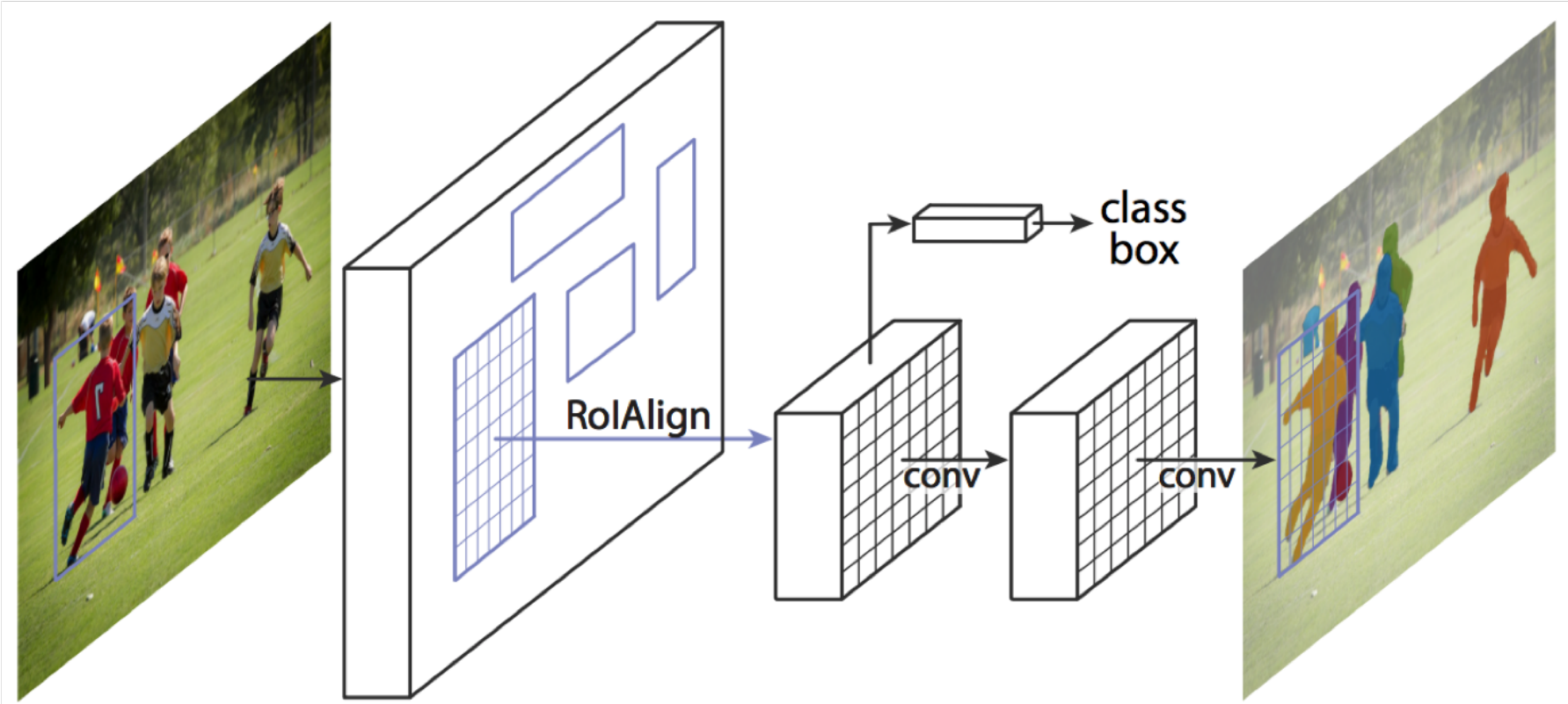
DeconvNet Result



Mask R-CNN

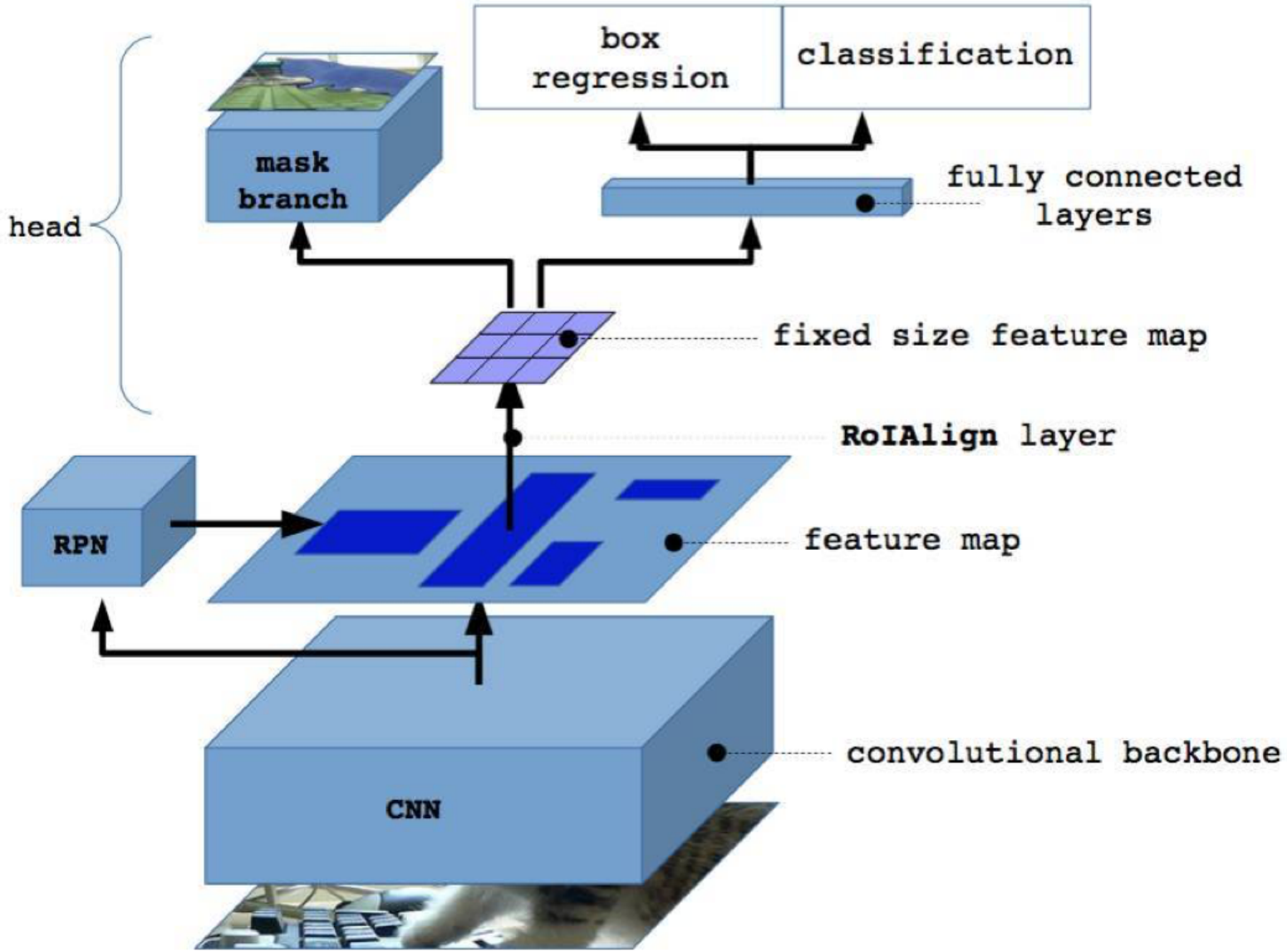
He et al., Mask R-CNN. ICCV 2017

- Extension of Faster R-CNN for instance segmentation



© He et al.

Mask R-CNN Architecture



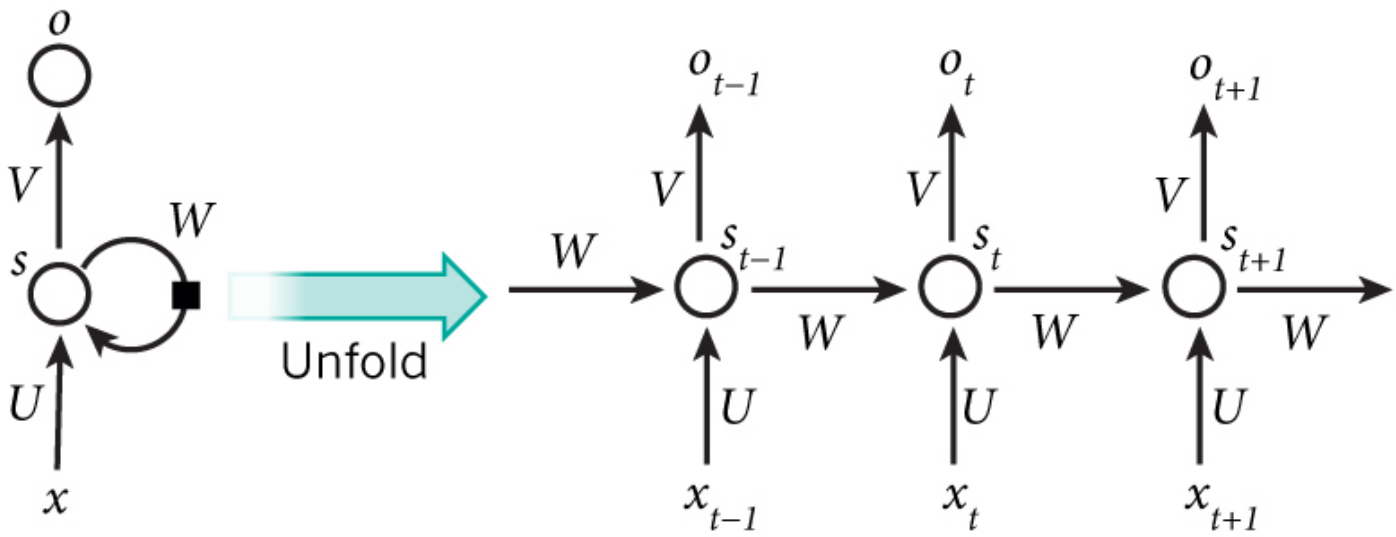
© He et al.

Mask R-CNN Result

© He et al.

Recurrent Neural Network

LeCun, Bengio, and Hinton, Deep Learning, Nature 2015

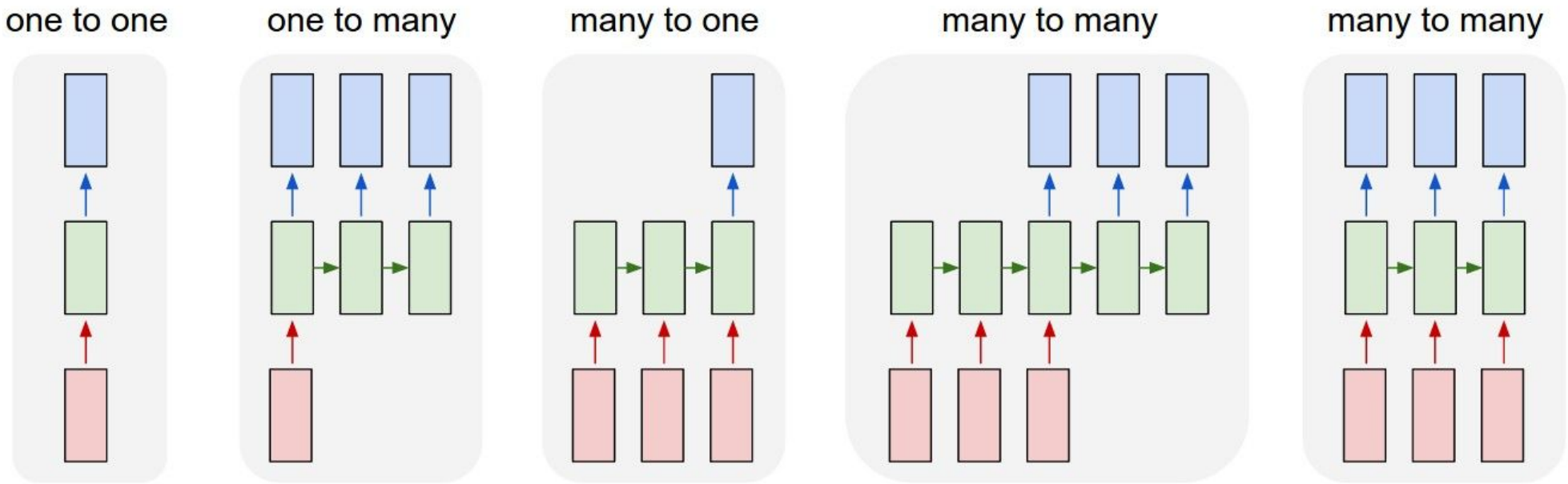


LeCun et.al. Nature 2015

- x_t is the input, s_t is the state, and o_t is the output at time t.
- s_{t-1} also affects s_t through W.
- The parameters U, V, W are shared across all time steps - reduced parameters.

Recurrent Neural Network

Slide adapted from CS231n by F. F. Li et.al.

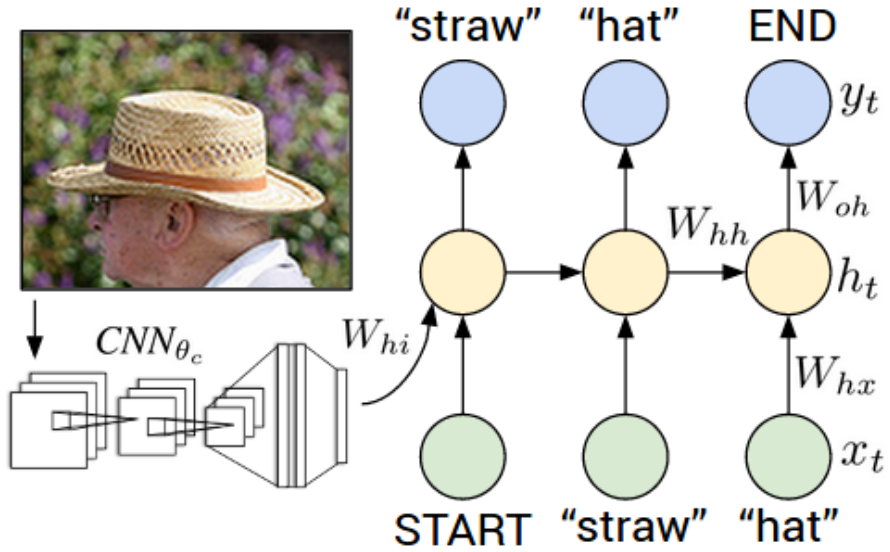


F. F. Li et.al. CS231n

- One-to-many : image captioning
- Many-to-one : sentence classification
- Many-to-many : machine translation, video classification

Image Captioning

Slide adapted from CS231n by F. F. Li et.al.



Karpathy and Li, CVPR 2015

- Combination of RNN and CNN
- FC7 (4096-dim) outputs are used as visual features.
- RNN generates the words for the visual input.

Summary

- Deep learning is a very active research field.
- Deep neural networks, especially CNNs, are good at extracting robust and semantic features from images.
- Lots of applications are possible.