Introduction to Machine Learning

Deep Discriminative Models

SHENG-JYH WANG

NATIONAL YANG MING CHIAO TUNG UNIVERSITY, TAIWAN

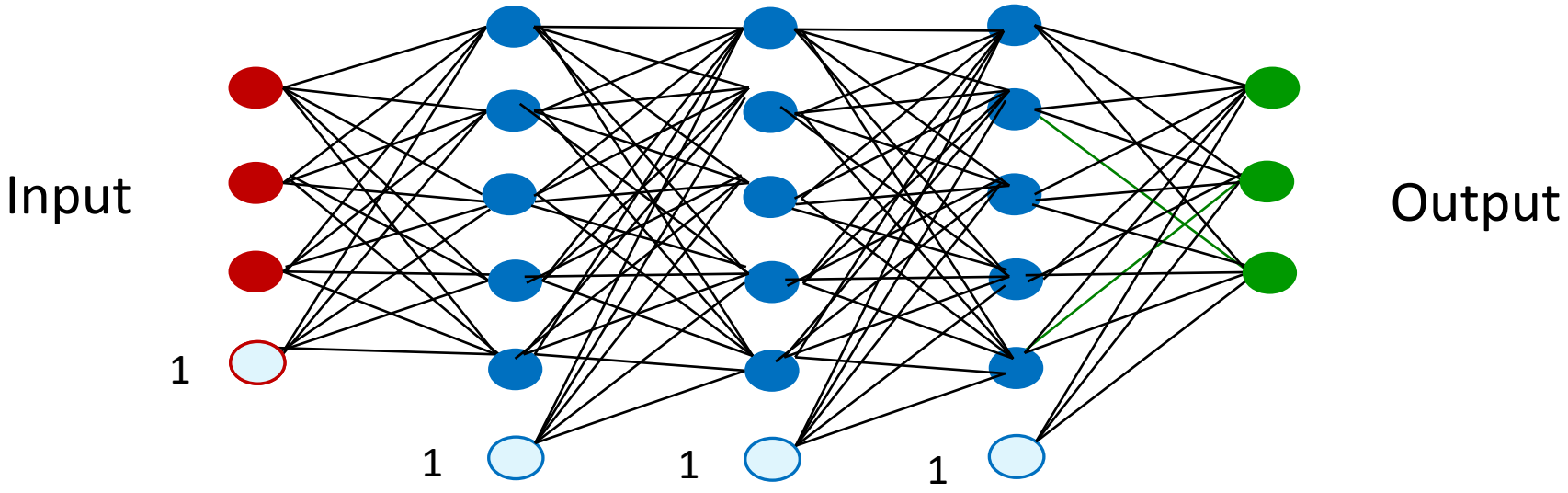
SPRING, 2024

Outline

Deep Discriminative Models

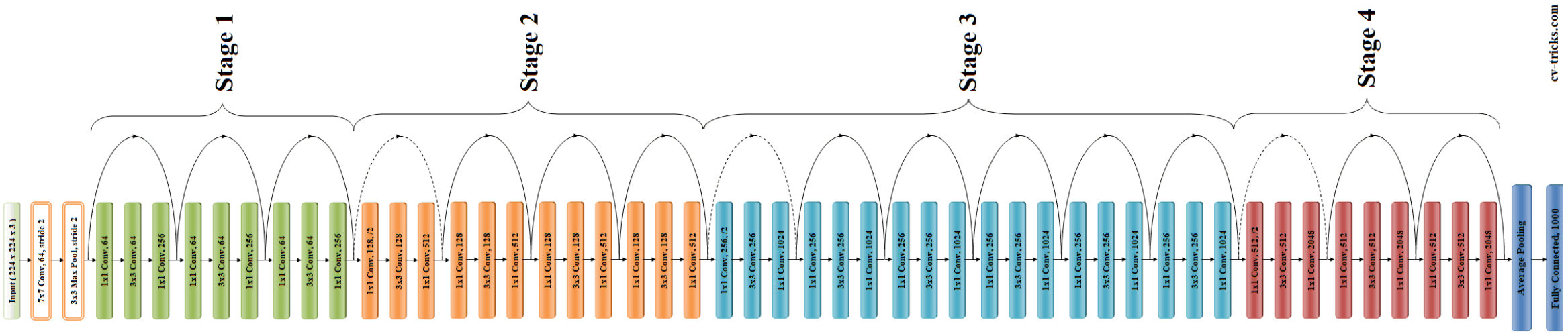
- Deep Convolutional Neural Network (DCNN)
- Recurrent Neural Network (RNN)
- Transformer

Neural Network



Deep Convolutional Neural Network

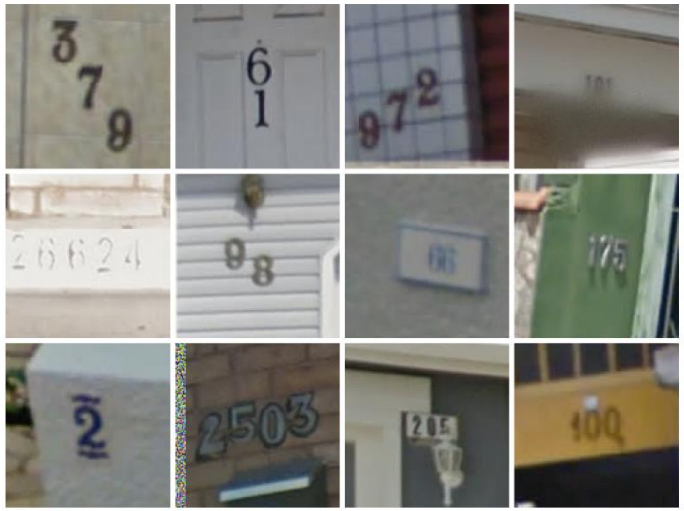
Example: ResNet 50

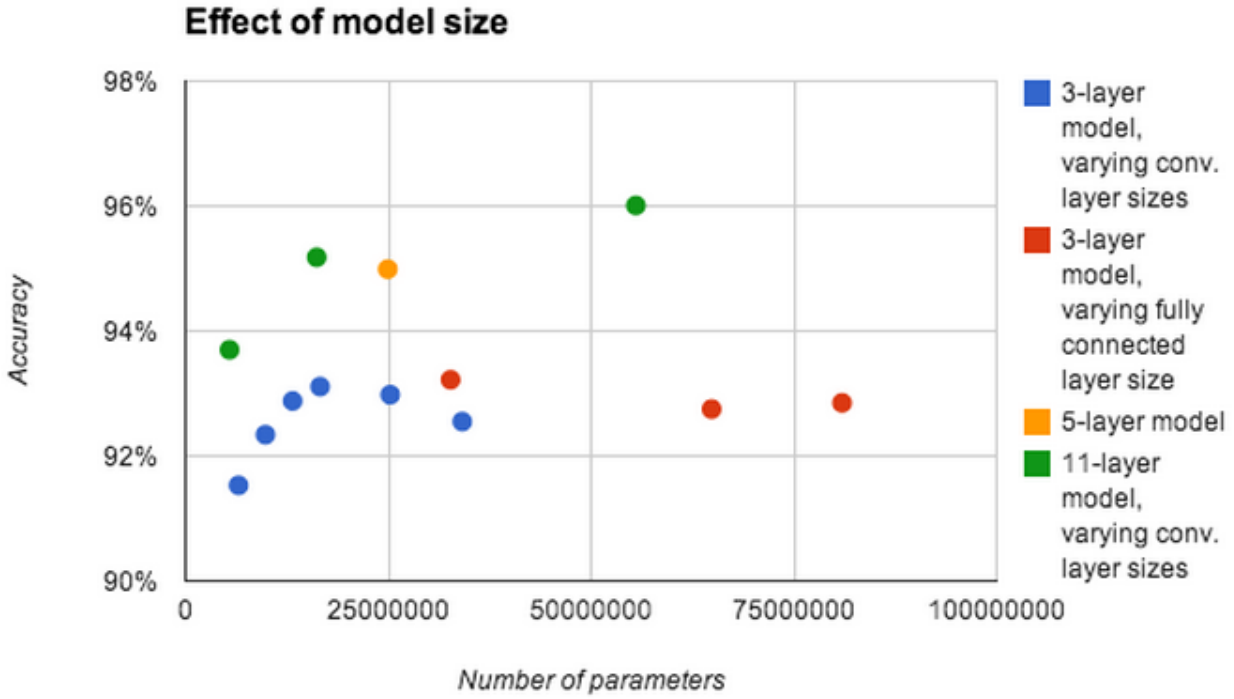


Ref: https://cv-tricks.com/keras/understand-implement-resnets/

Depth of Neural Networks

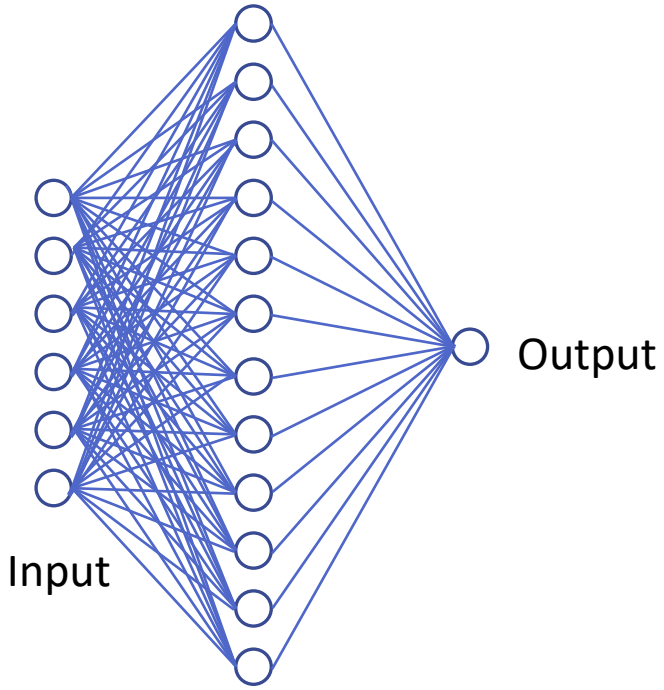
Example: Multi-digit Number Recognition

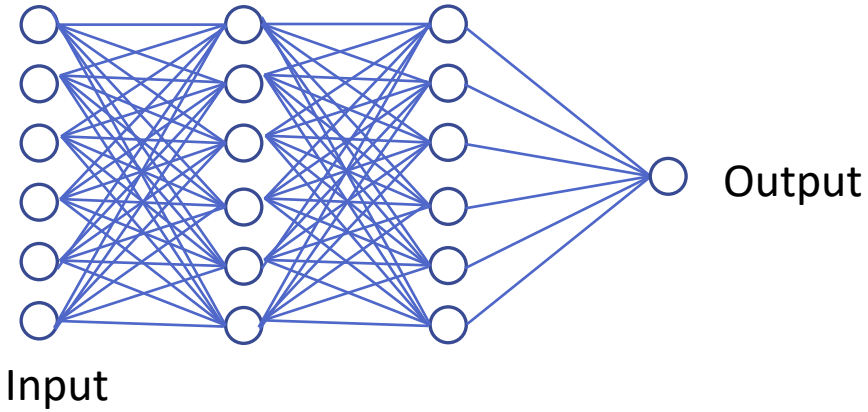




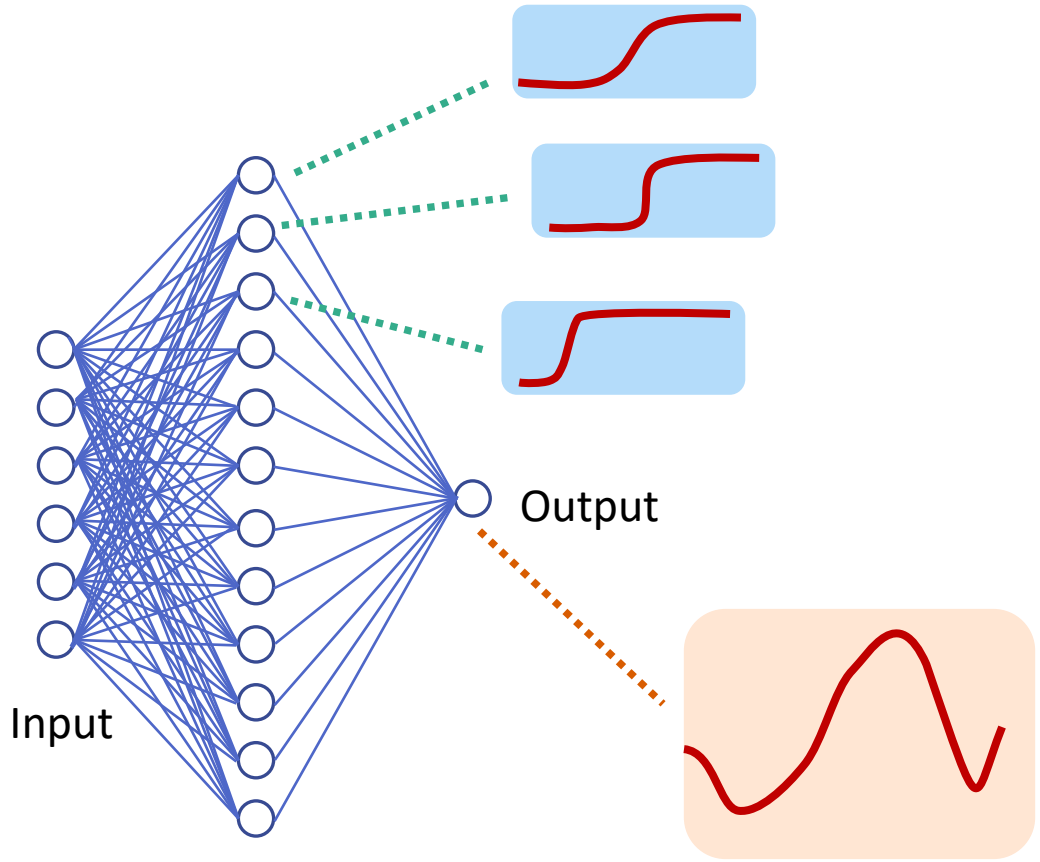
Ref: Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Shallow Networks versus Deep Networks

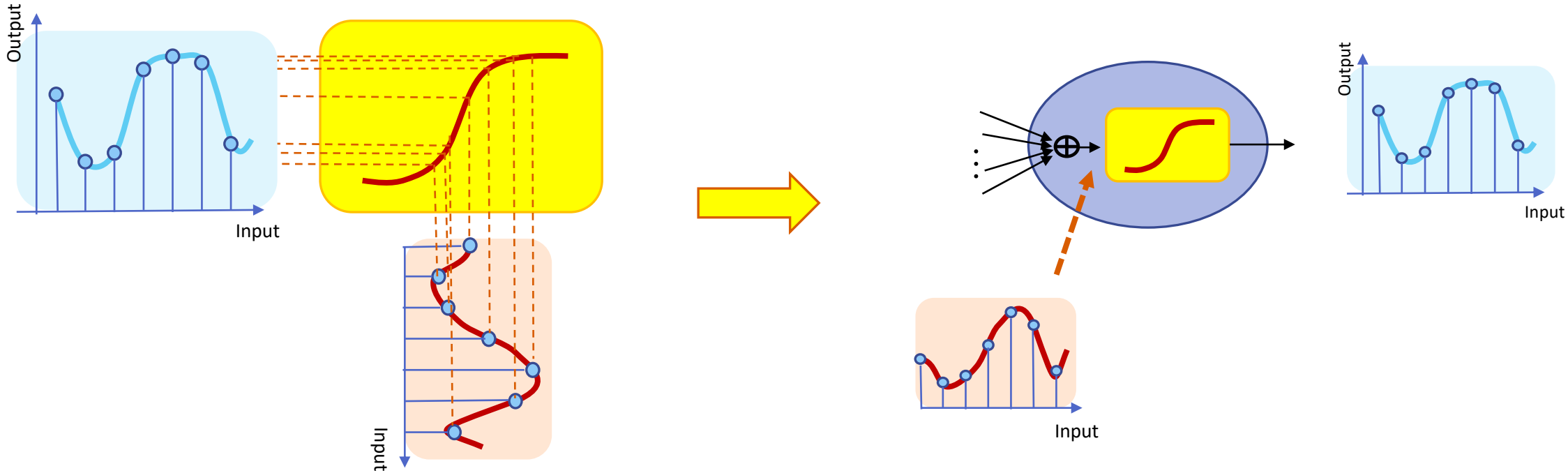




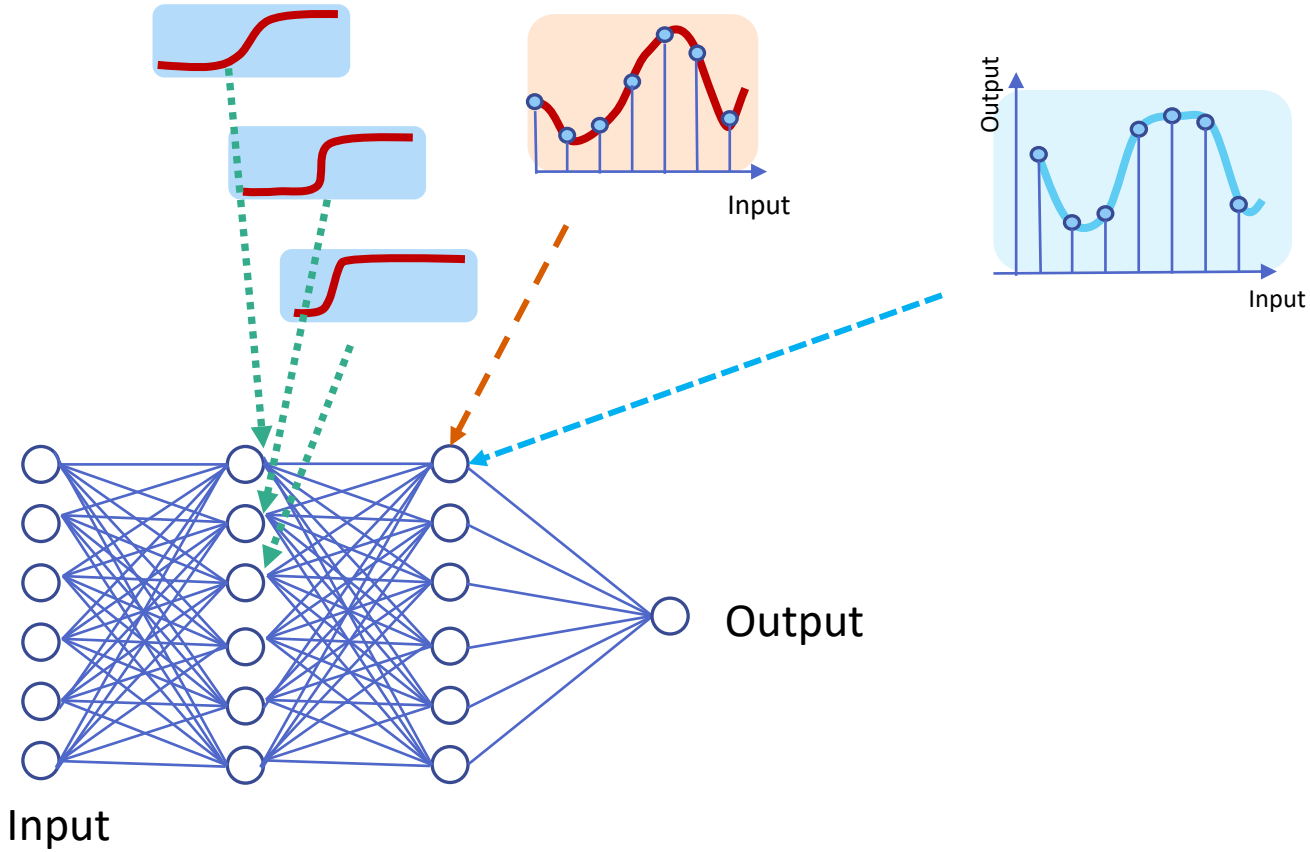
Shallow Networks versus Deep Networks

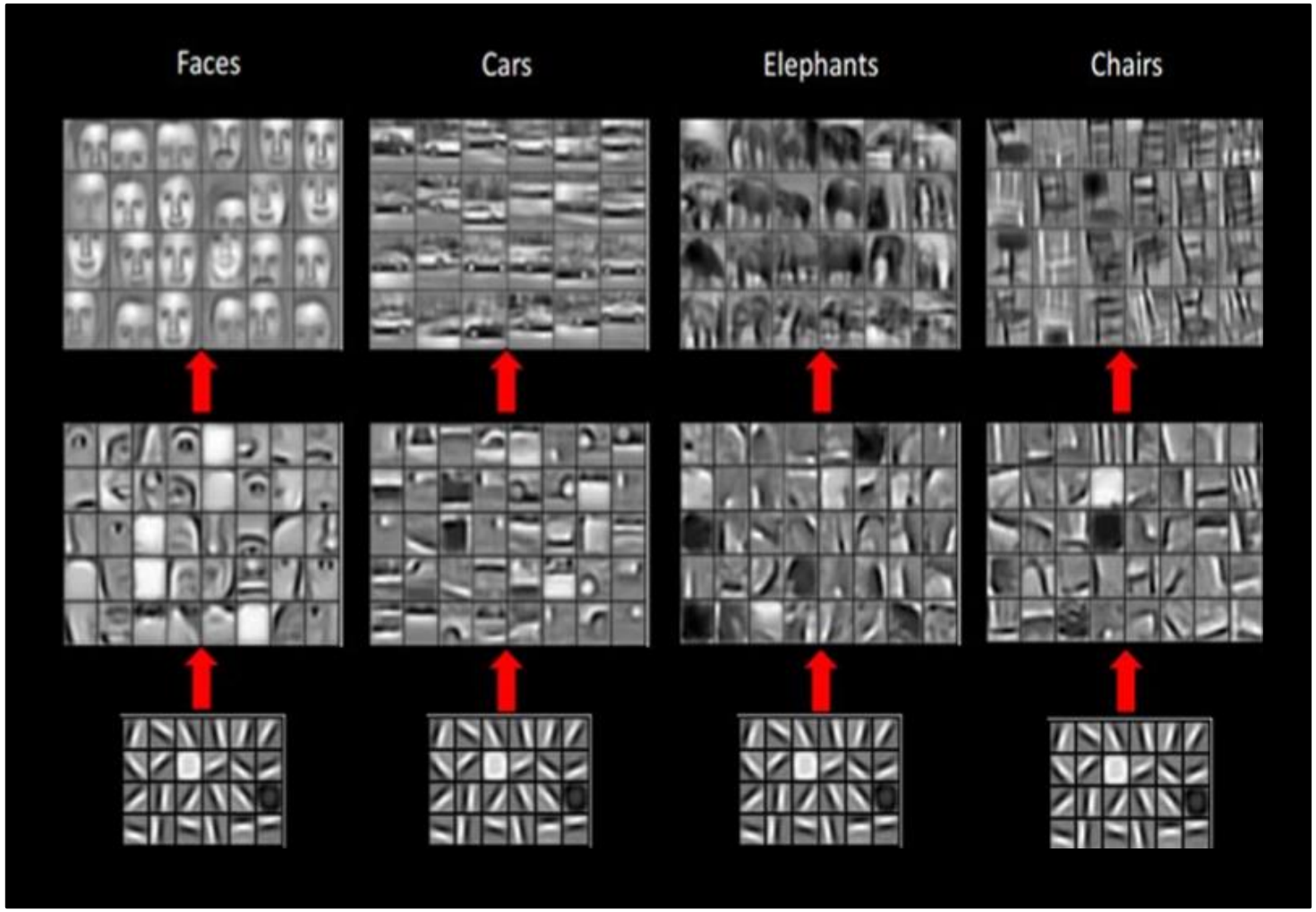


Shallow Networks versus Deep Networks



Shallow Networks versus Deep Networks





Ref:
chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://pdfs.semanticscholar.org/425f/c05d848f0318289f496f7b5e8cad26b123f6.pdf

Deep Convolutional Neural Networks

History

Year	Contributer	Contribution
300 BC	Aristotle	introduced Associationism, started the history of human's attempt to understand brain.
1873	Alexander Bain	introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.
1943	McCulloch & Pitts	introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.
1949	Donald Hebb	considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.
1958	Frank Rosenblatt	introduced the first perceptron, which highly resembles modern perceptron.
1974	Paul Werbos	introduced Backpropagation
1980	Teuvo Kohonen	introduced Self Organizing Map
	Kunihiko Fukushima	introduced Neocogitron, which inspired Convolutional Neural Network
1982	John Hopfield	introduced Hopfield Network
1985	Hilton & Sejnowski	introduced Boltzmann Machine

(Ref: Haohan Wang and Bhiksha Raj, "On the Origin of Deep Learning", 2017)

History

1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

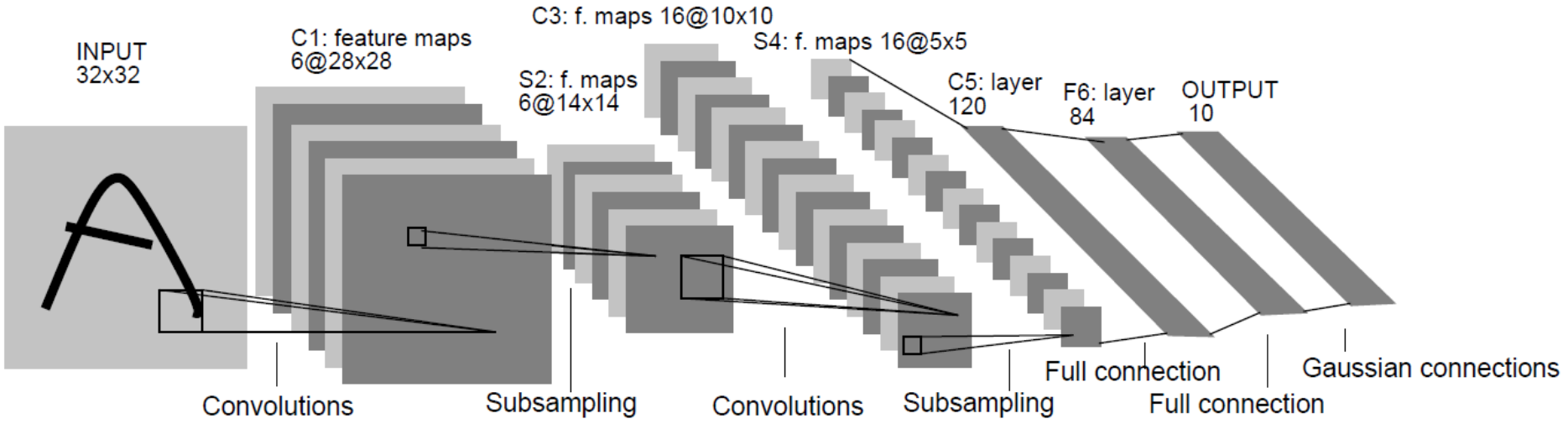
(Ref: Haohan Wang and Bhiksha Raj, “On the Origin of Deep Learning”, 2017)

LeNet

LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" Proceedings of the IEEE. 86 (11): 2278–2324.

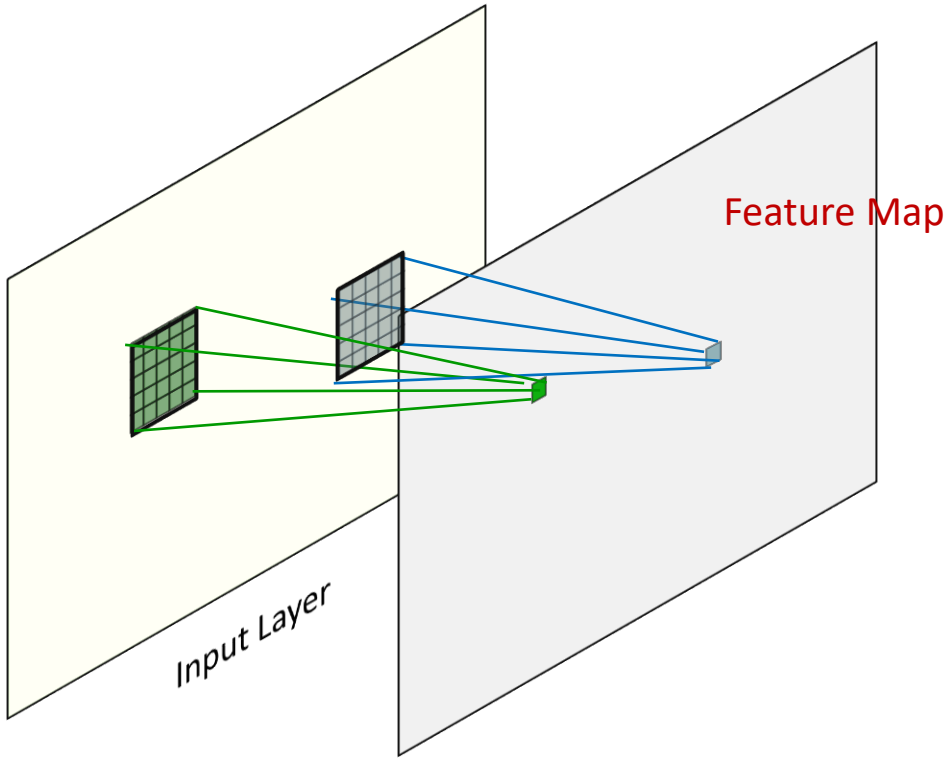
A type of feed-forward artificial neural network

Mimic the receptive fields of human vision systems



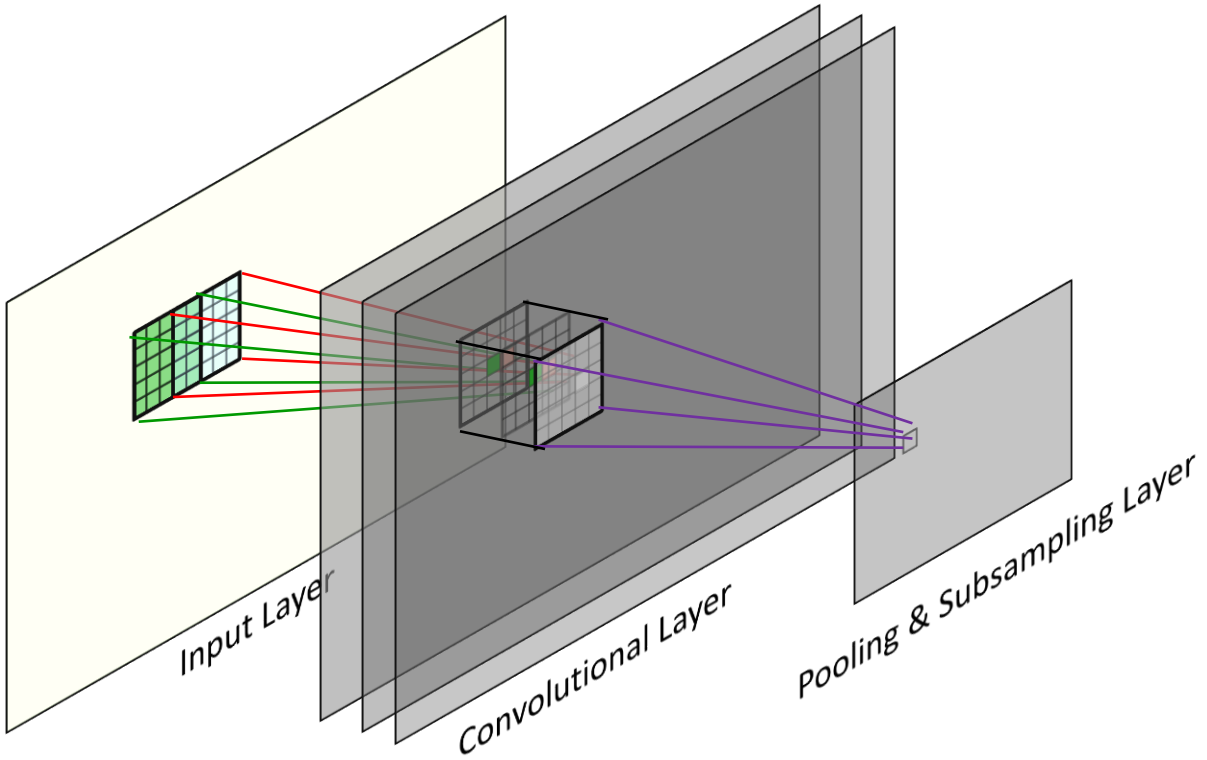
Convolutional Neural Network (CNN)

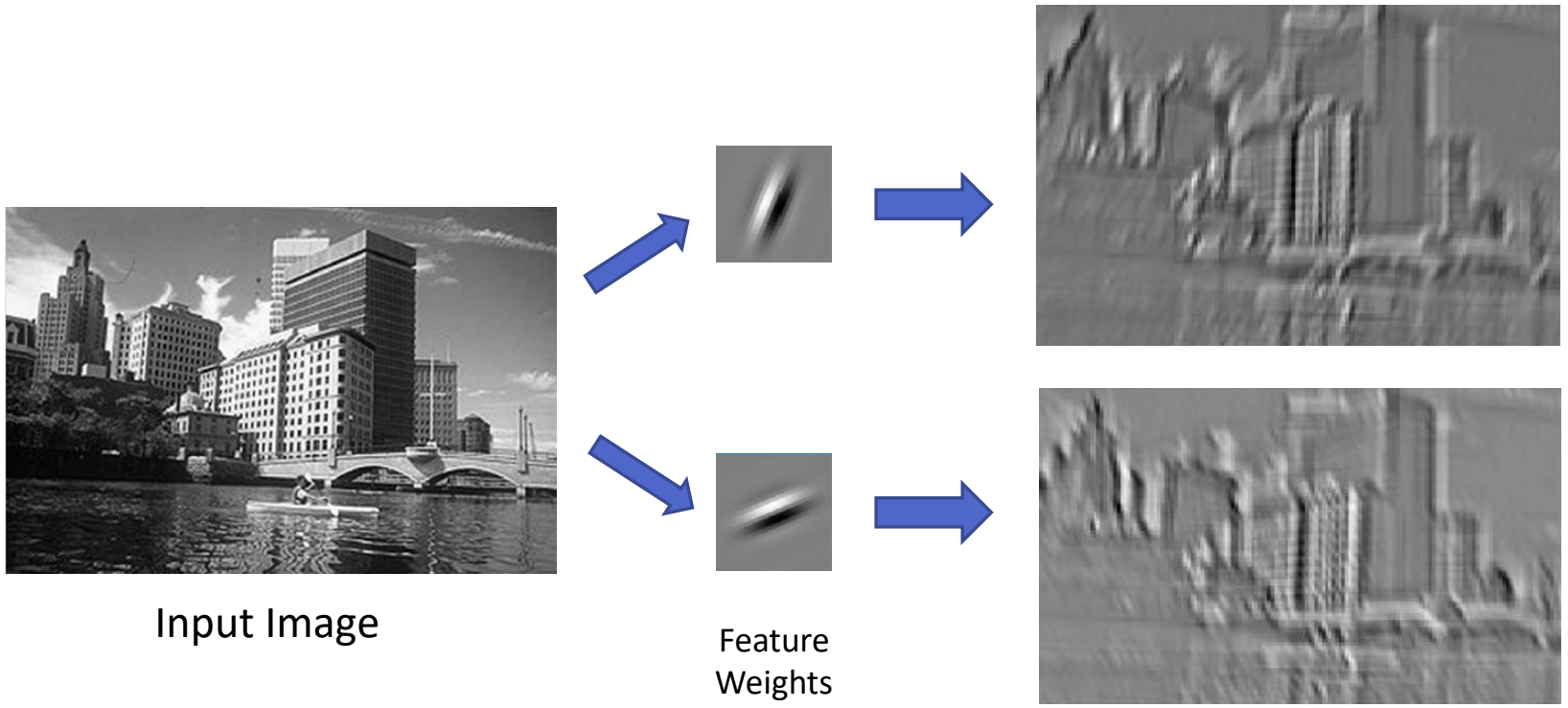
- Local Receptive Fields
- Weight Sharing



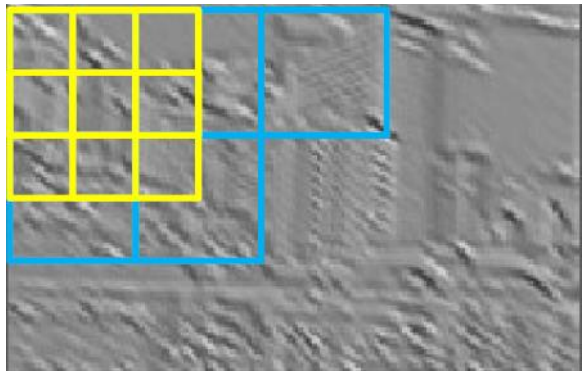
Convolutional Neural Network (CNN)

- Spatial Sub-sampling

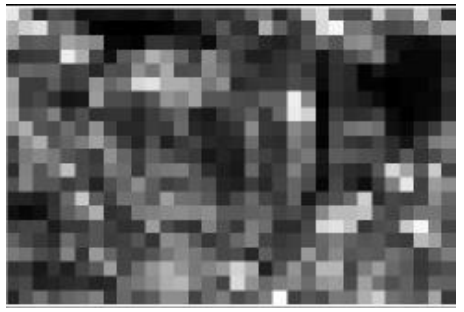



Input Image
Feature
Weights
Feature Maps

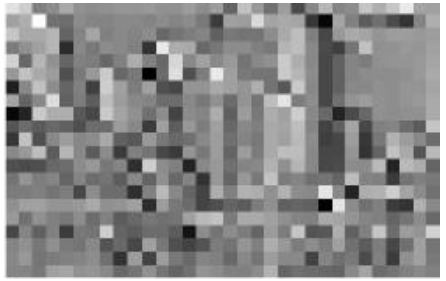
Spatial Pooling



Feature Map



Max Pooling



Average Pooling

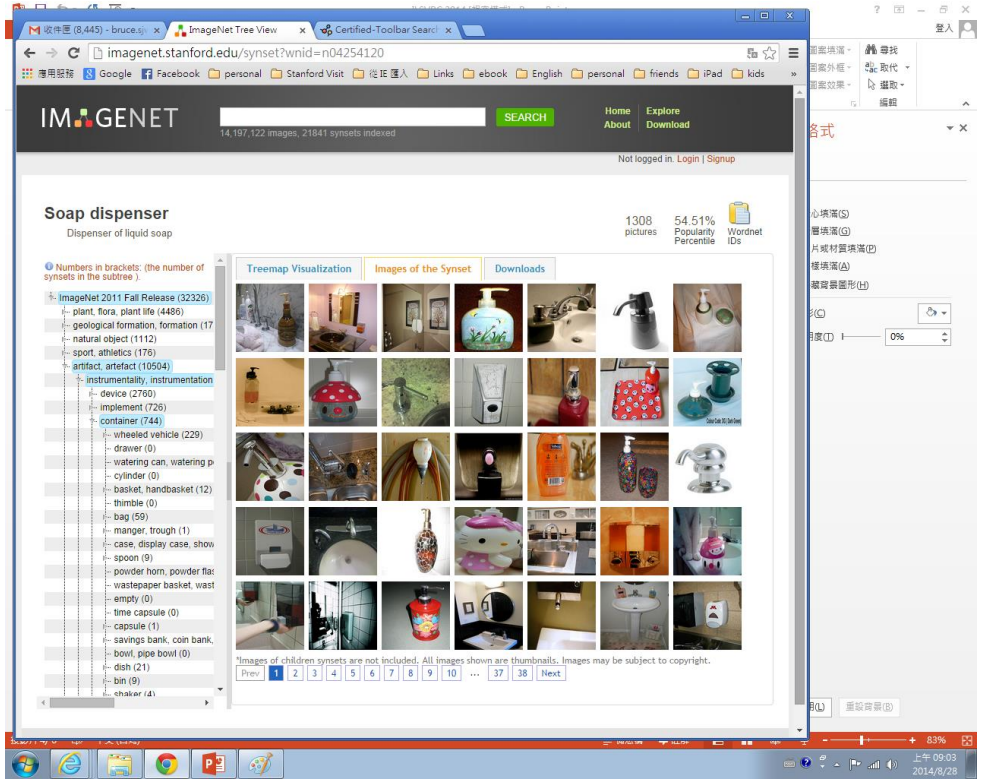
(Ref: Rob Fergus, “Deep Learning for Computer Vision”)

1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

Since 2006, deep architectures have attracted a lot of attentions, but the major trends had been DBNs and DBMs untill ***2012***.

ImageNet Database

Contain over 15M labeled images in over 22,000 categories.



ImageNet LSVRC Challenge 2012

Task 1: Classification

Car

- Predict a class label
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.

Task 2: Detection (Classification + Localization)

classification Car

- Predict a class label and a bounding box
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 40% of training.

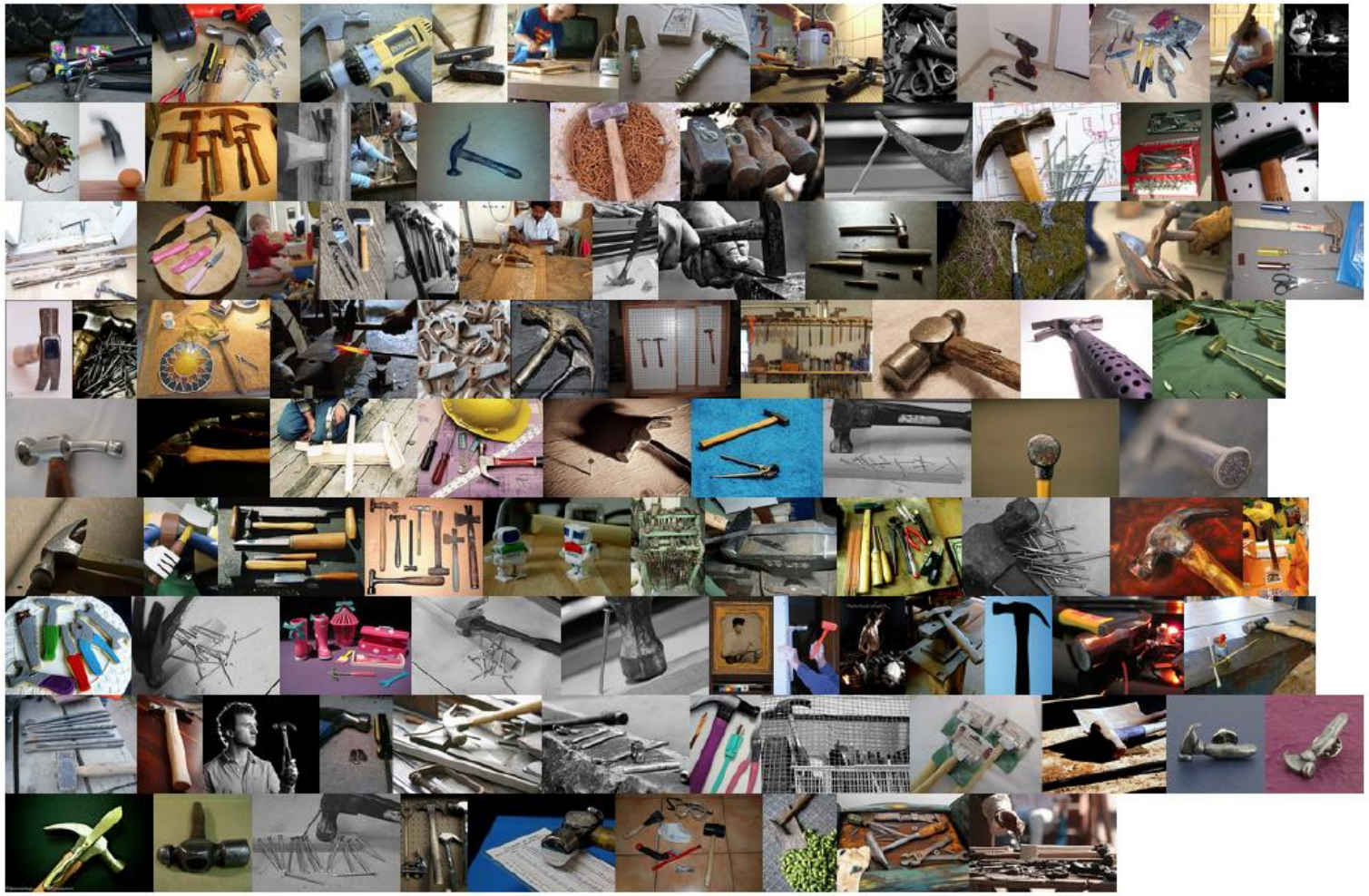
Task 3: Fine-grained classification

classification Walker hound

- Predict a class label given a bounding box in test
- 1 prediction / image
- 120 dog classes (subset)
- ~200 images per class for training (subset)
- Bounding boxes for 100% of training

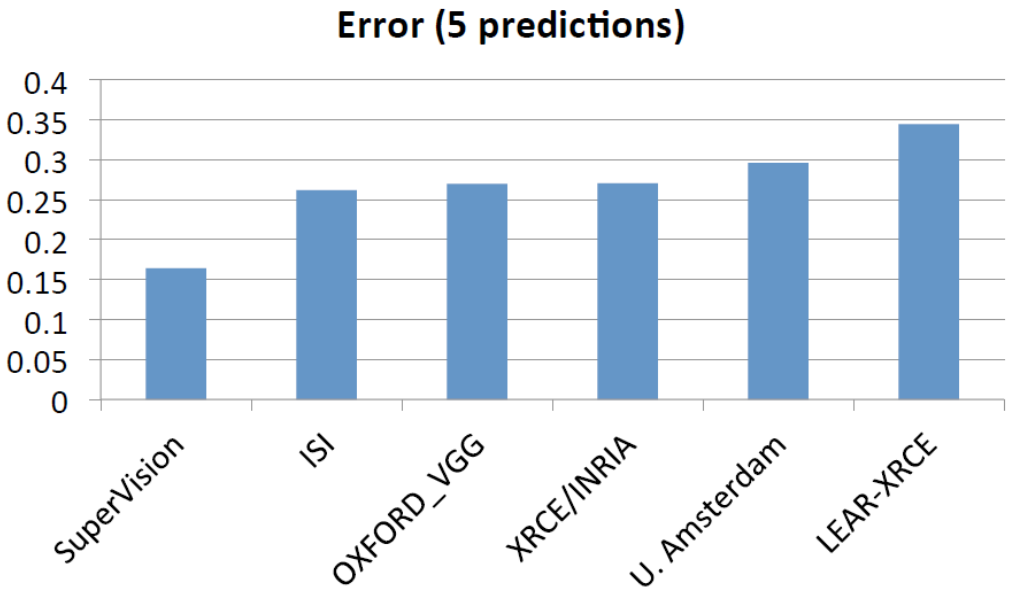
Ref: http://image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf

Test Images for “Hammer”

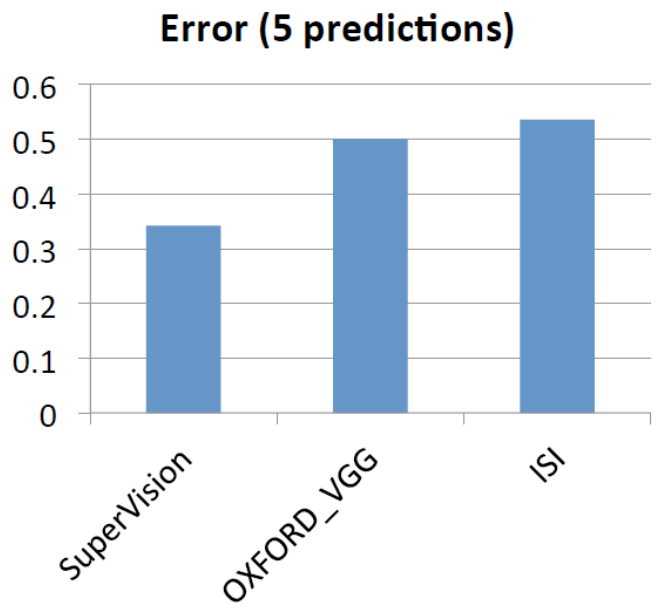


Ref: http://image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf

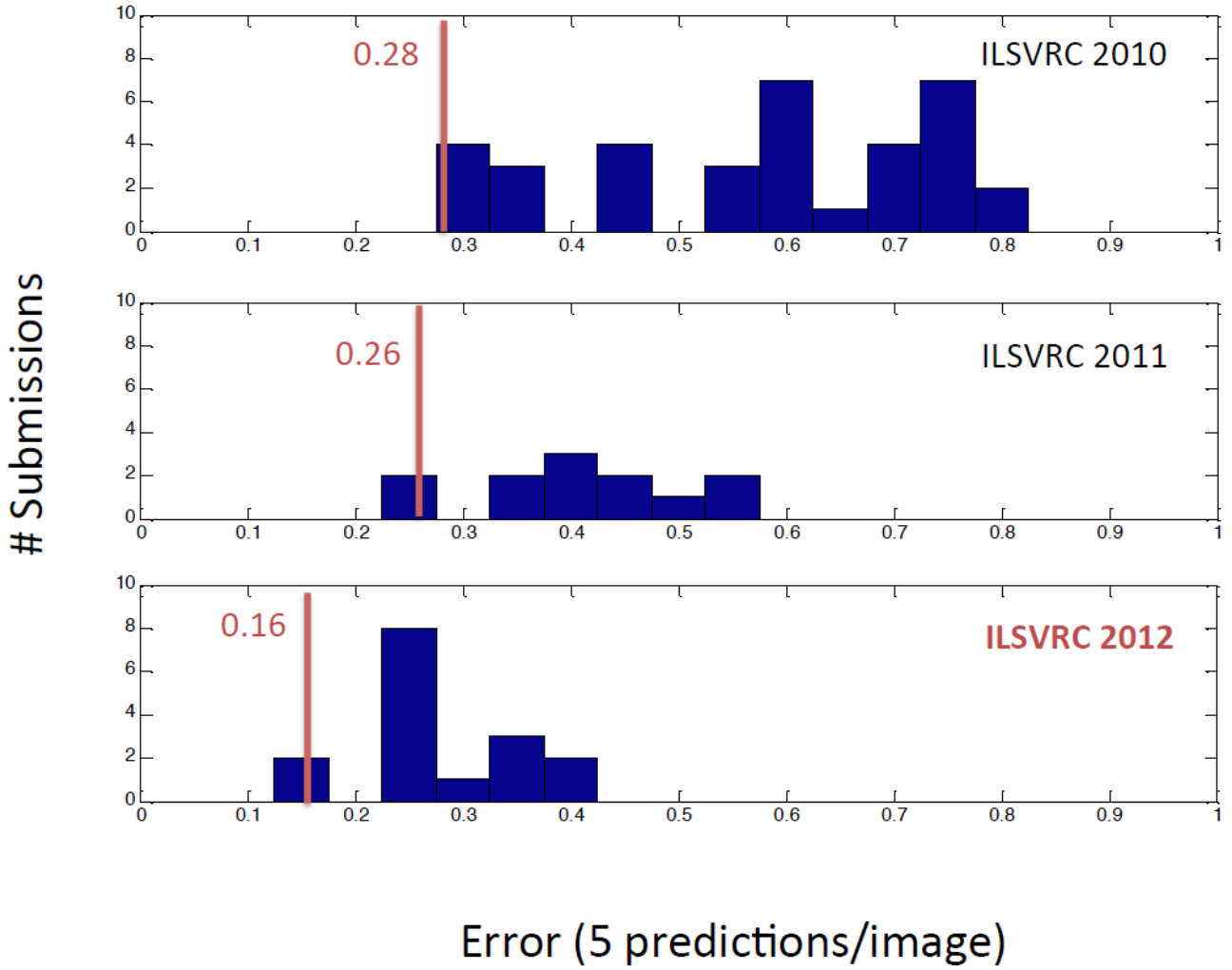
Classification



Detection



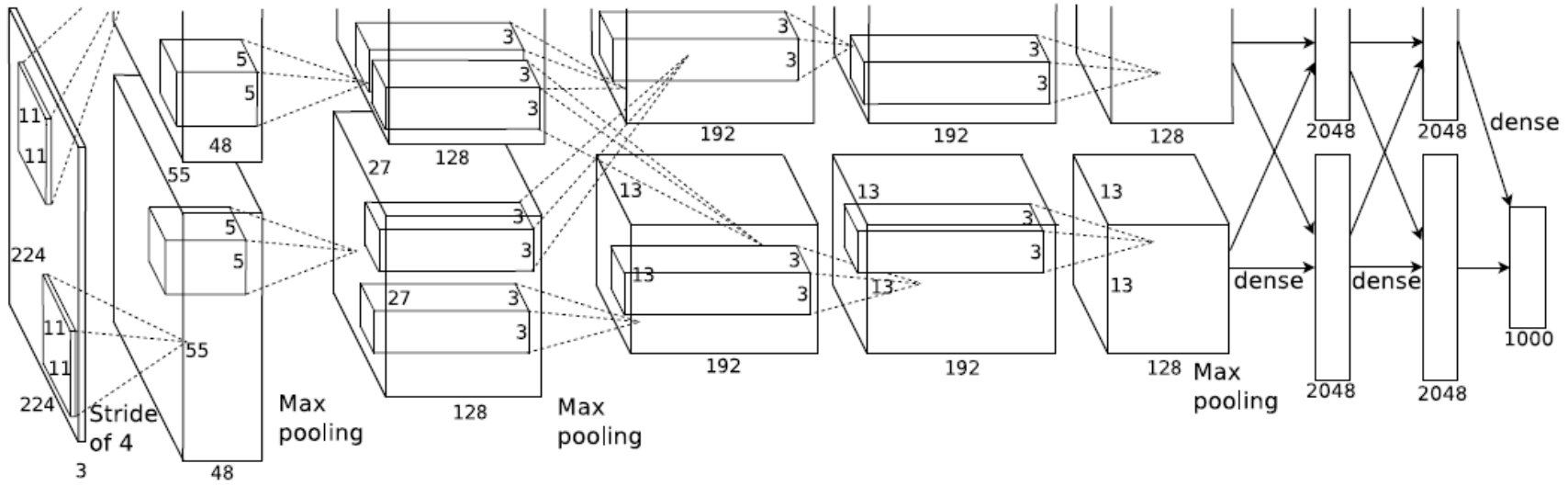
- SuperVision: **Convolutional Neural Network** (University of Toronto)
- ISI: Fisher based Features + Passive-Aggressive Algorithm (The University of Tokyo)
- OXFORD_VGG: Hand-Crafted Features (DPM)+SVM (Oxford University)



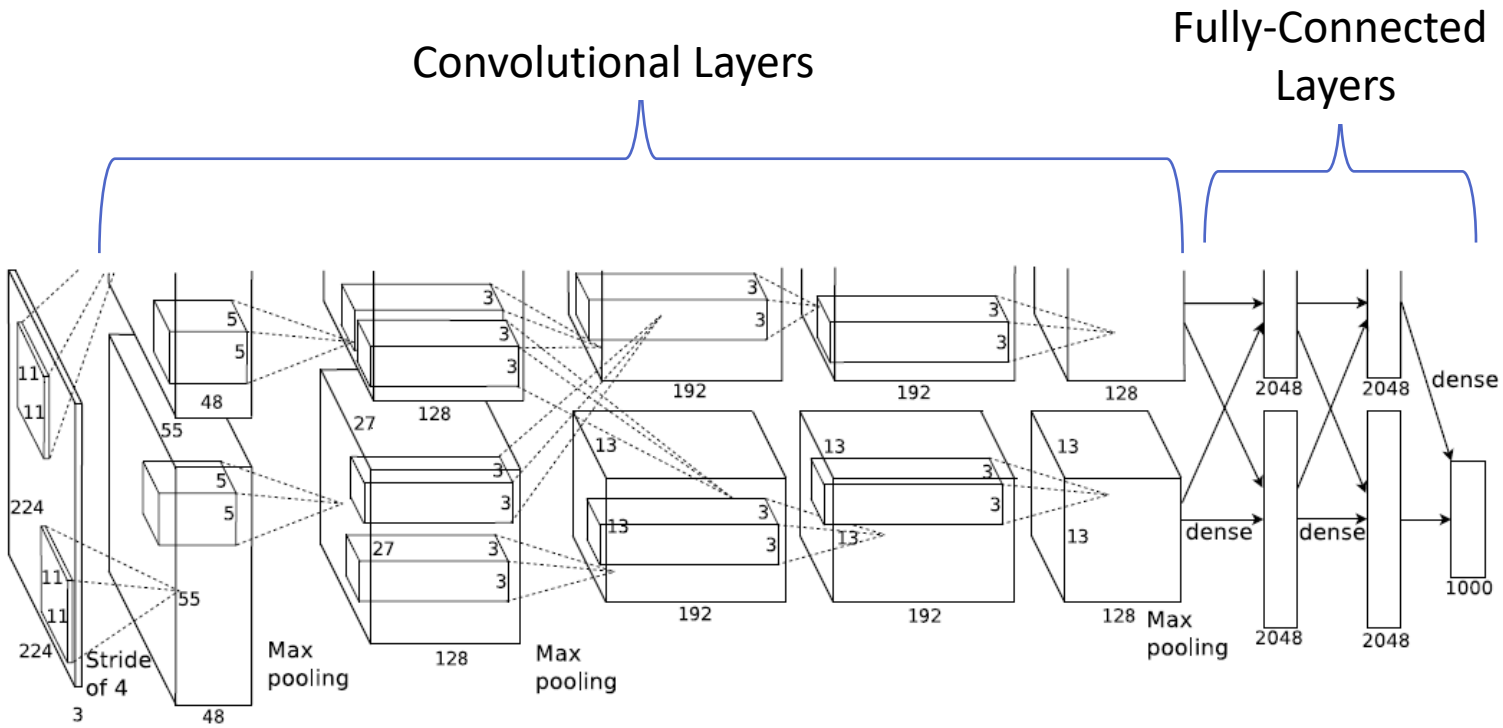
Ref: http://image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf

AlexNet

- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
- Winner of 2012 ImageNet LSVRC (Large Scale Visual Recognition Competition) Challenge.
- Same model as LeNet, but with a larger model and much more training data (10^6 v.s. 10^3)
- GPU implementation (50X speedup over CPU)



- 60 million parameters, 650,000 neurons
- 5 convolutional layers, 3 fully-connected layers, 1 1000-way softmax.
- Use Rectified Linear Units
- GPU implementation of the convolution operation
- Use the “dropout” method for regularization
- Training data: 1.2 million images in ImageNet
- Top-1 error rate: 36.7%
- Top-5 error rate: 15.3%

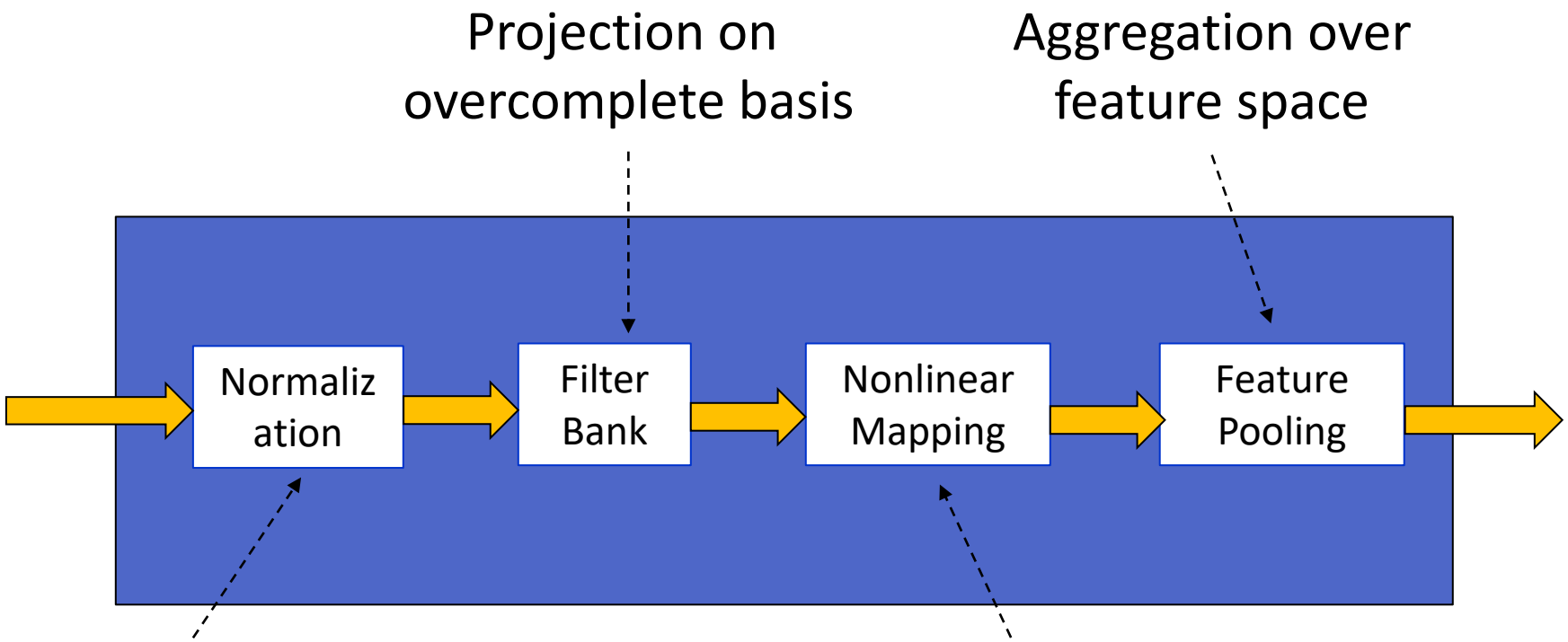


Nodes: 224*224*3 | 55*55*96 | 27*27*256 | 13*13*384 | 13*13*384 | 13*13*256 | 4096 | 4096 | 1000

Kernels: 11*11*3 | 5*5*48 | 3*3*256 | 3*3*192 | 3*3*192

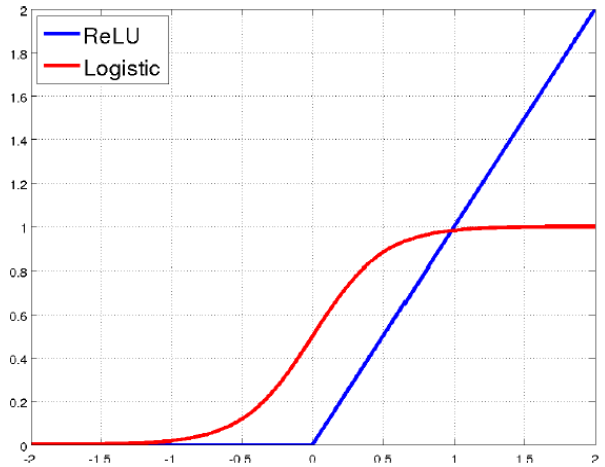
LRN + Pooling | LRN + Pooling | Pooling

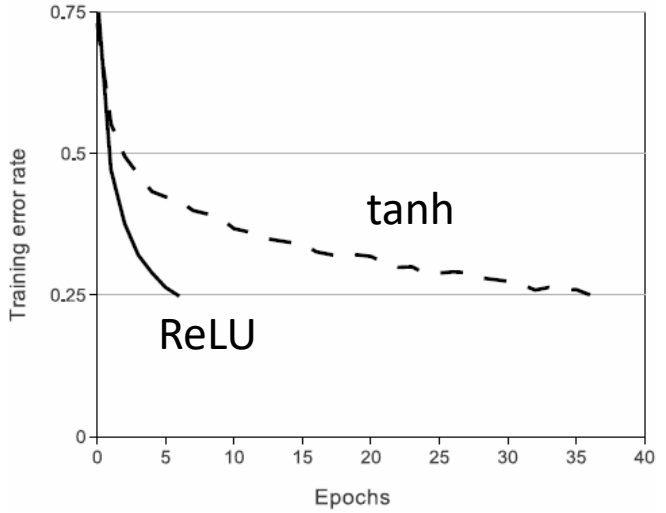
Hidden Layer of CNN



Rectified Linear Unit (ReLU)

- $f(x) = \max(0, x)$
- Simplify back-propagation
- Speed up the learning process





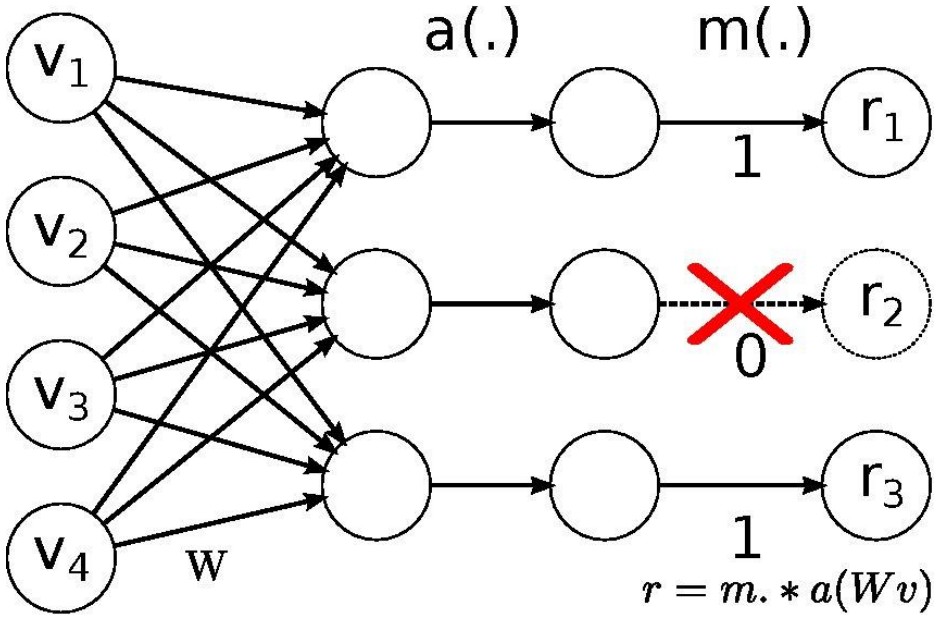
(Ref: Zeiler, Matthew D., et al. "On rectified linear units for speech processing," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013.
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

To avoid overfitting

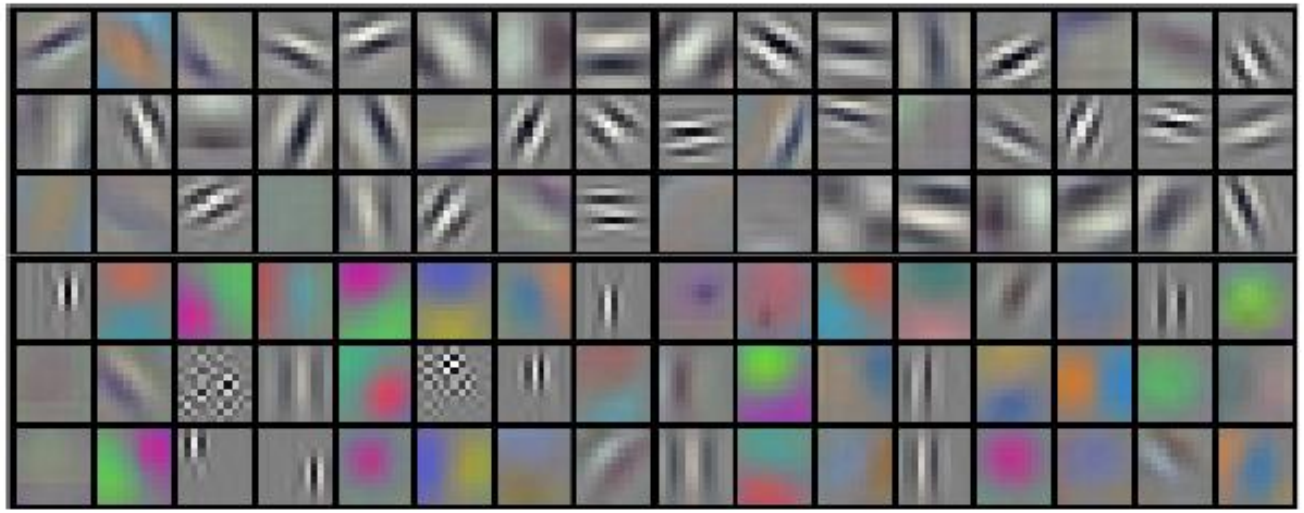
- Data augmentation
  - ✓ Randomly extract 224×224 patches from 256×256 images + horizontal reflection
  - ✓ Change the intensities of RGB channels
- Dropout
  - ✓ Training: set the output of each hidden neuron to zero with probability 0.5

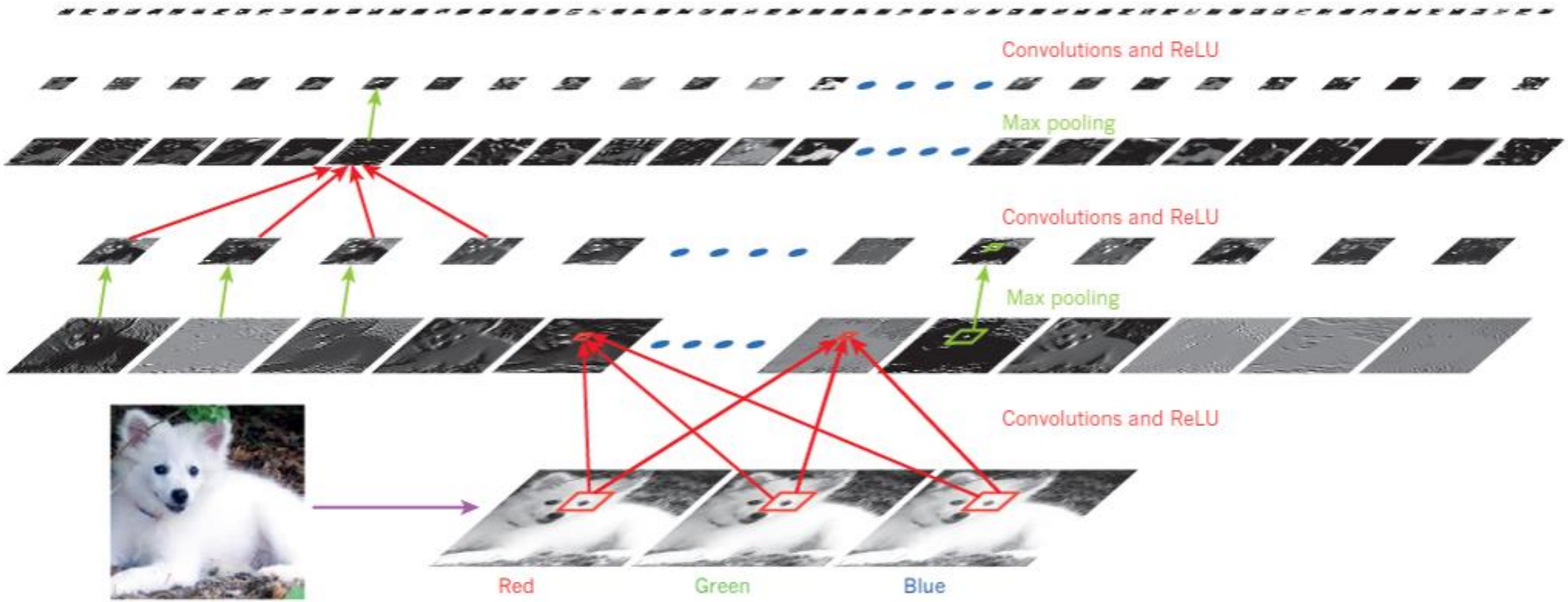
    Testing: use all neurons but multiply their outputs by 0.5
  - ✓ Use dropout in the first two fully-connected layers

Dropout

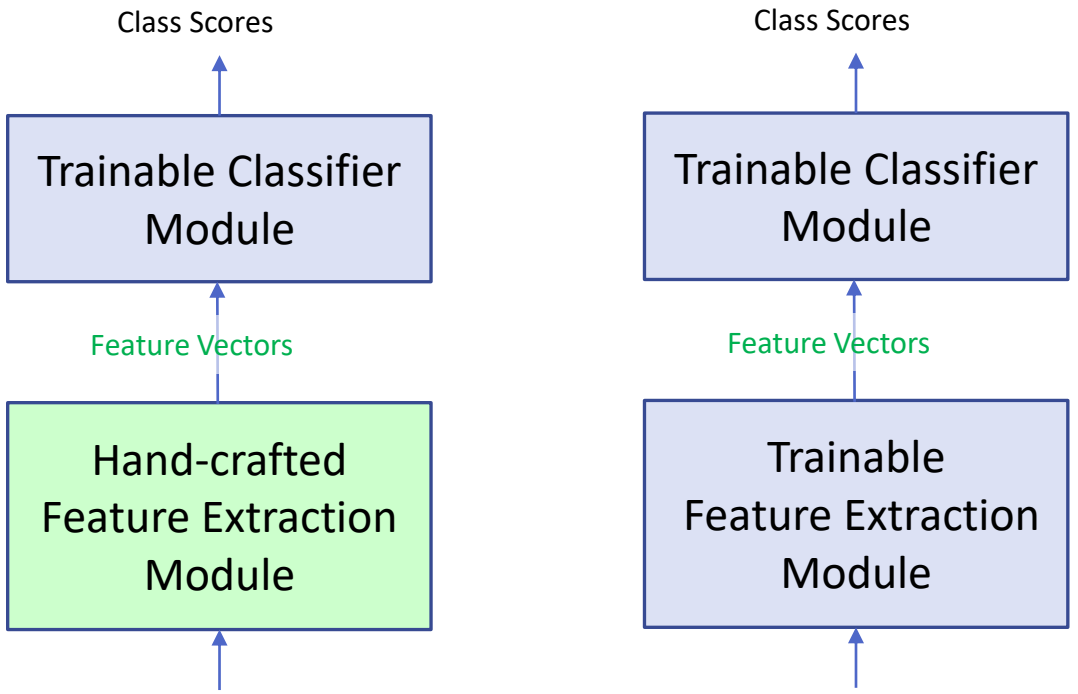


Learned 11×11×3 Features



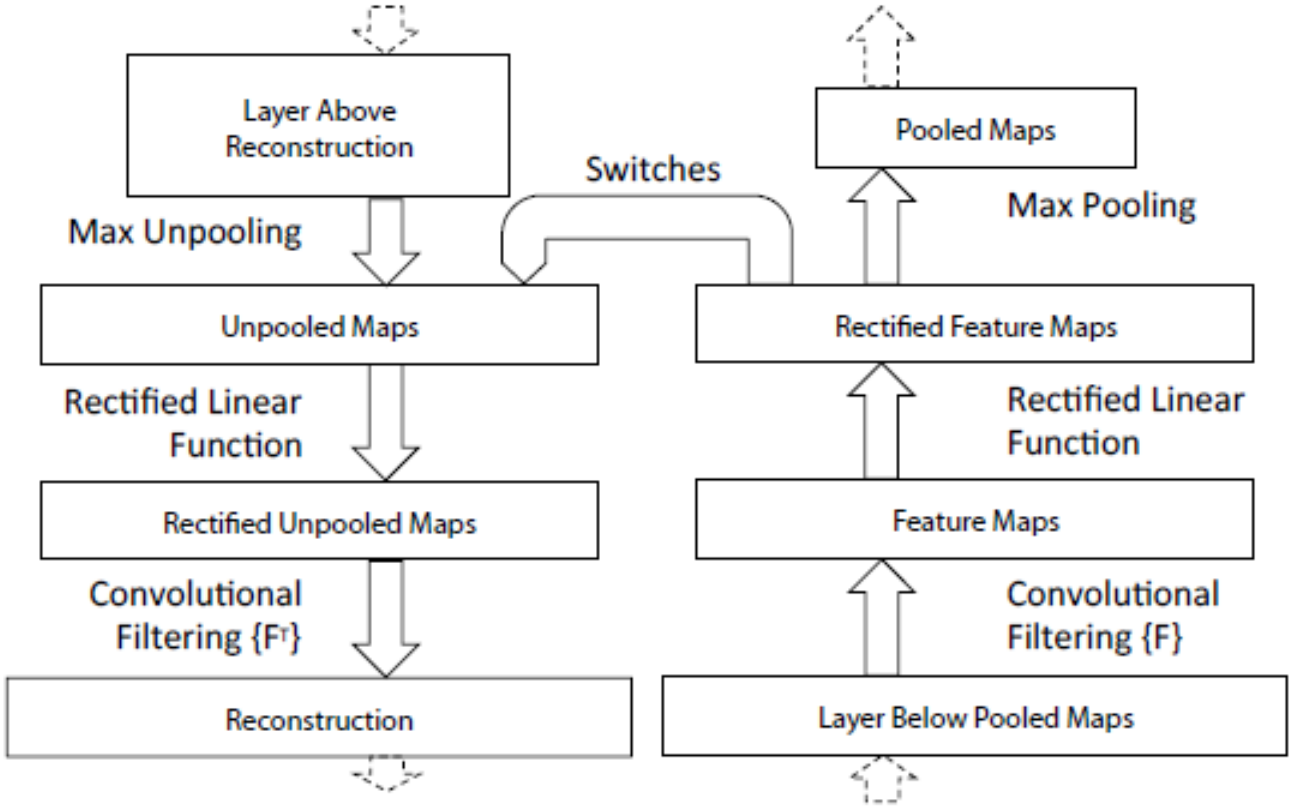


Ref: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.

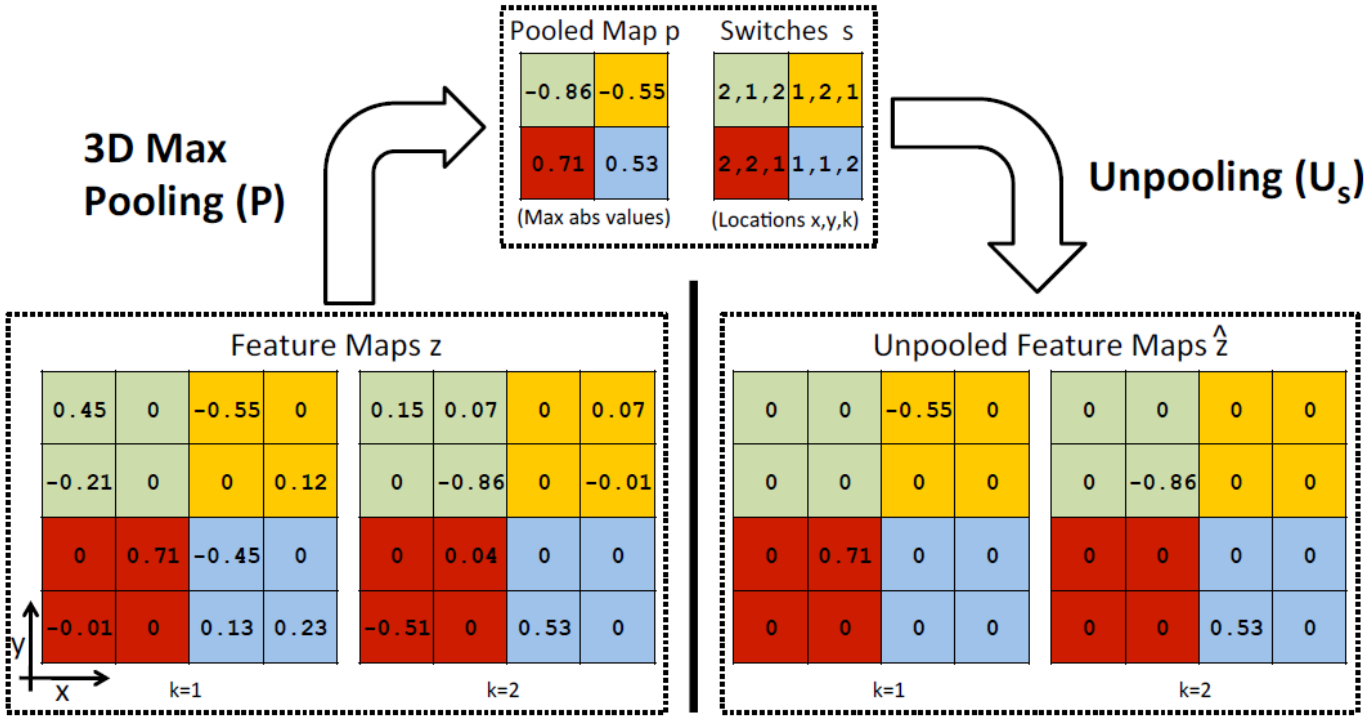

Class Scores
Trainable Classifier Module
Feature Vectors
Hand-crafted Feature Extraction Module
Raw Input
Traditional ML Approach
Class Scores
Trainable Classifier Module
Feature Vectors
Trainable Feature Extraction Module
Raw Input
CNN Approach

ZF-Net

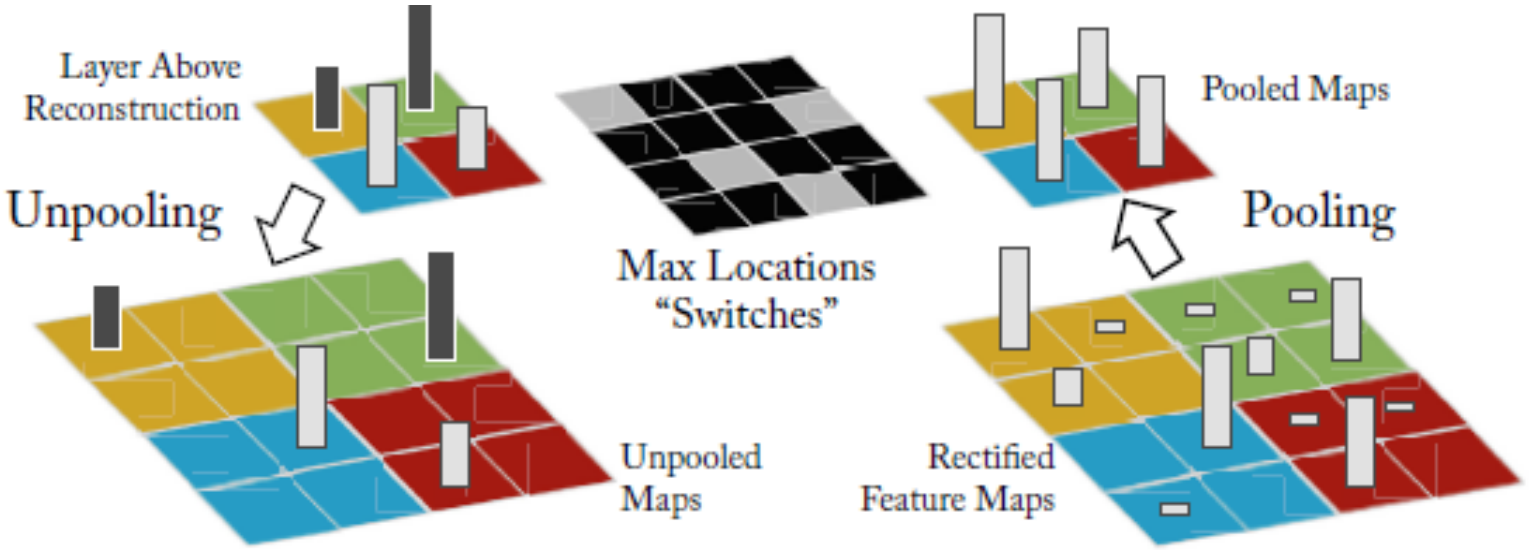
- Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.
- Winner of 2013 ILSVRC competition.
- Minor modifications over AlexNet
- Present a way to visualize CNN intermediate layers based on ***Deconvolutional Network***.


Layer Above Reconstruction
Max Unpooling
Unpooled Maps
Rectified Linear Function
Rectified Unpooled Maps
Convolutional Filtering {Fᵀ}
Reconstruction
Switches
Pooled Maps
Max Pooling
Rectified Feature Maps
Rectified Linear Function
Feature Maps
Convolutional Filtering {F}
Layer Below Pooled Maps

Unpooling

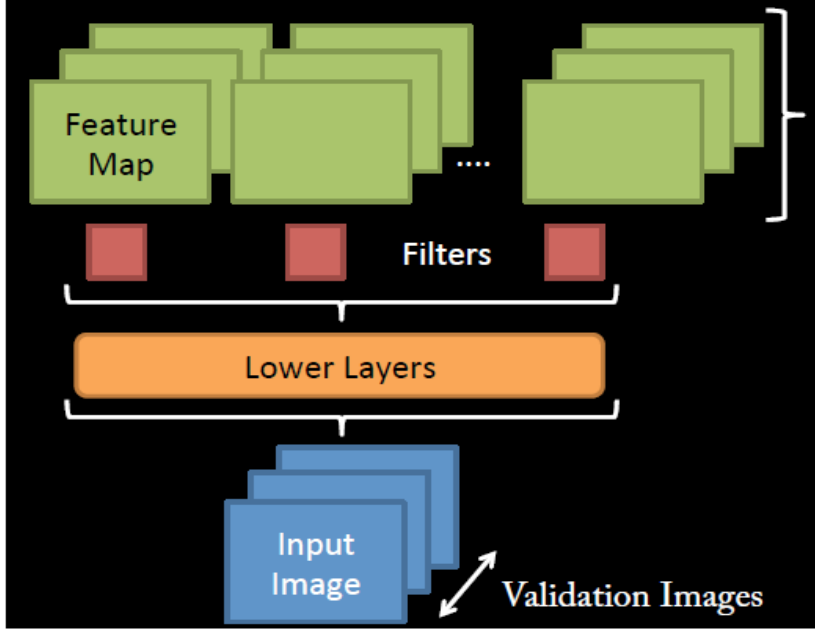


Ref: Zeiler, Matthew D., Graham W. Taylor, and Rob Fergus. "Adaptive deconvolutional networks for mid and high level feature learning." 2011 international conference on computer vision. IEEE, 2011.


Layer Above
Reconstruction
Unpooling
Unpooled
Maps
Max Locations
"Switches"
Pooled Maps
Pooling
Rectified
Feature Maps

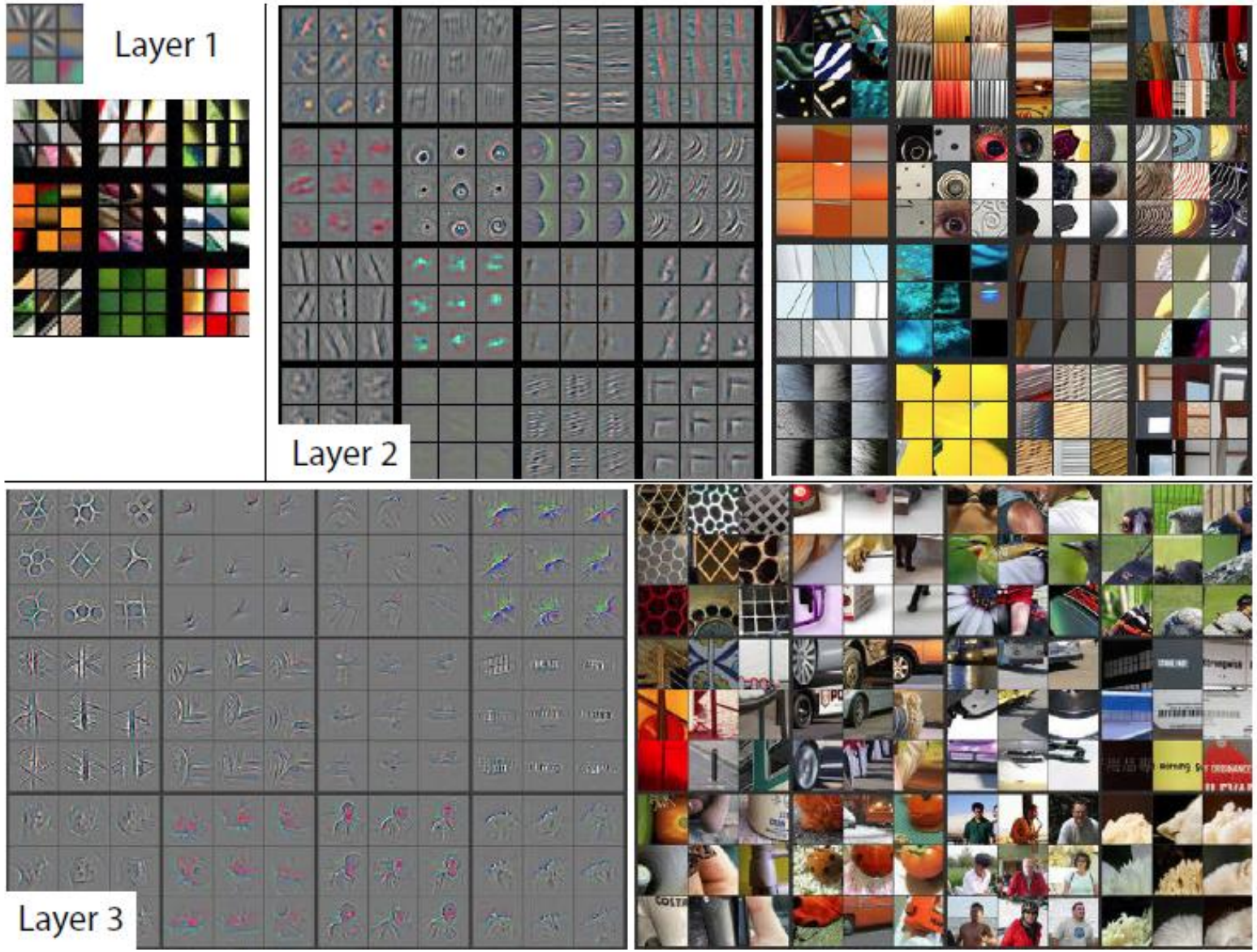
Visualizations of Higher Layers

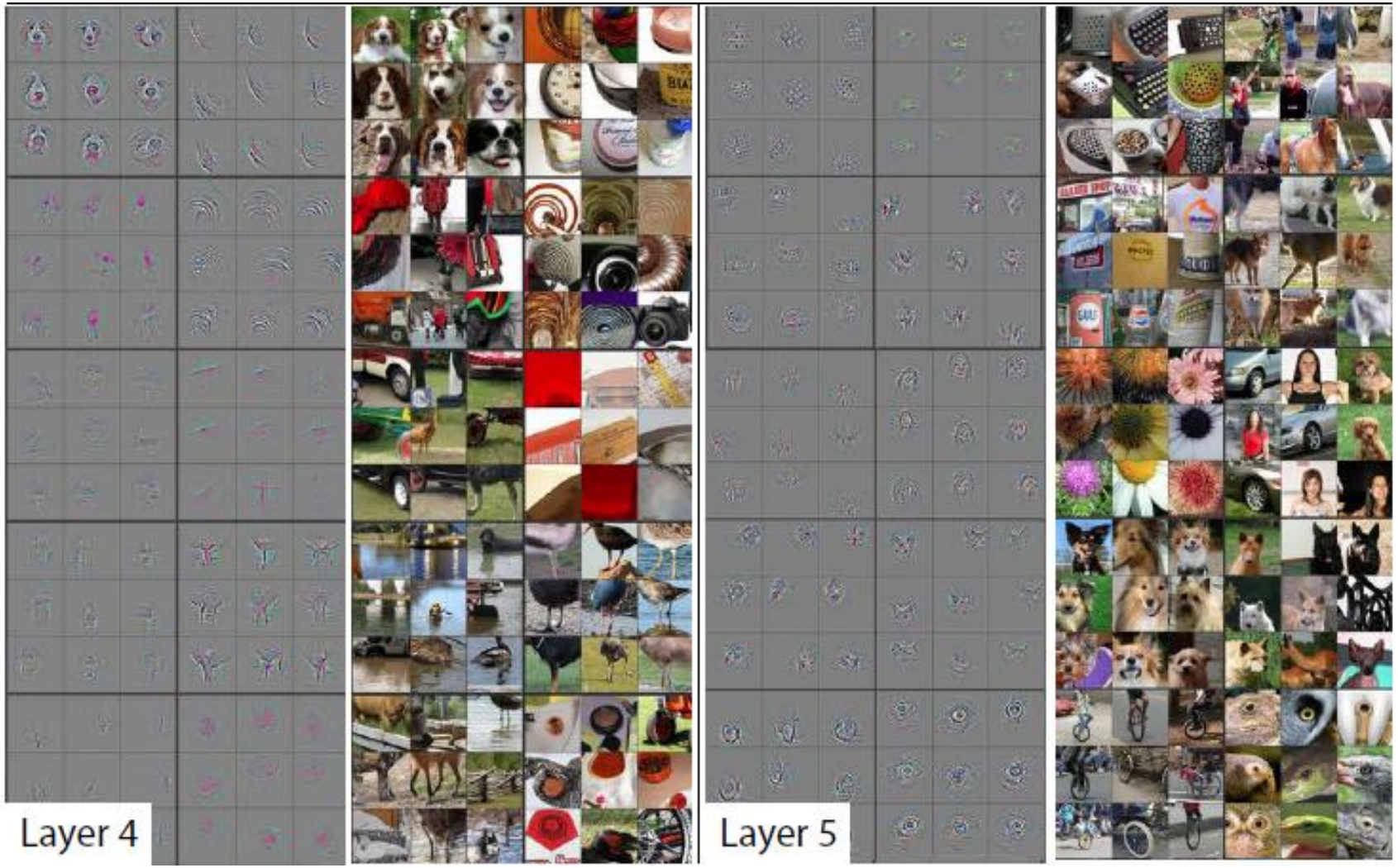
- Use ImageNet 2012 validation set [Zeiler and Fergus. arXiv'13]
- Push each image through network



- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling "switches" peculiar to that activation

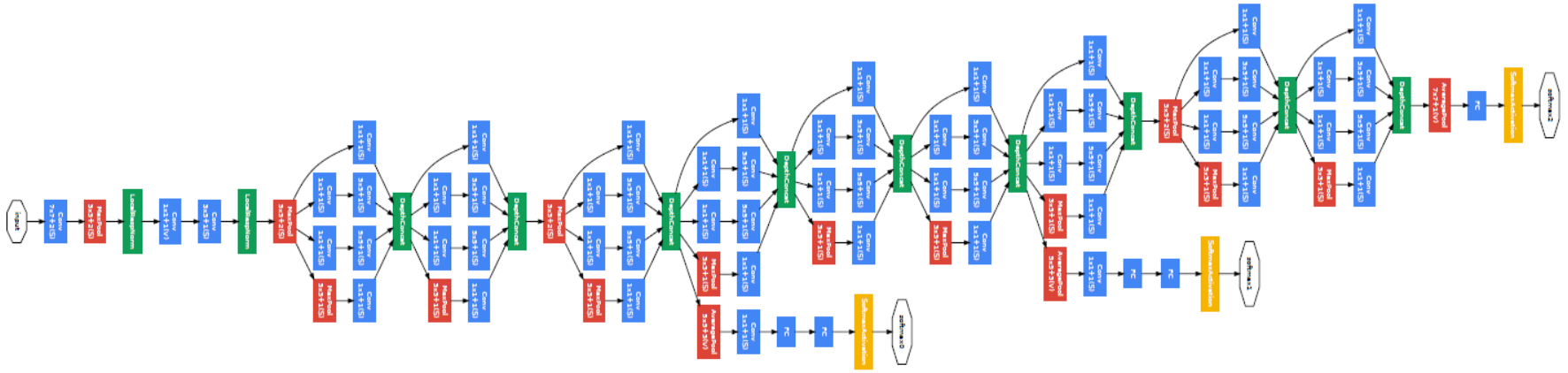
(Ref: LeCun & Ranzato, "Deep Learning Tutorial", ICML 2013)


Layer 1
Layer 2
Layer 3

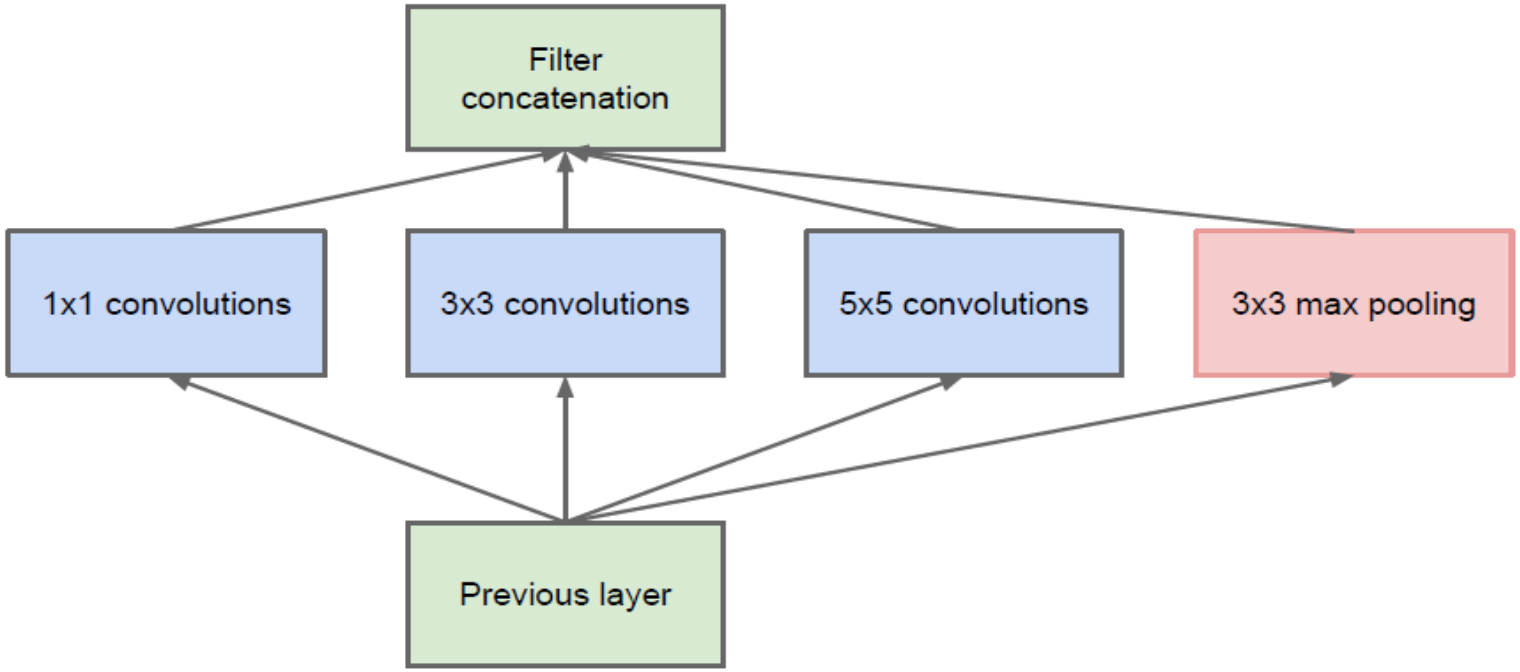

Layer 4
Layer 5

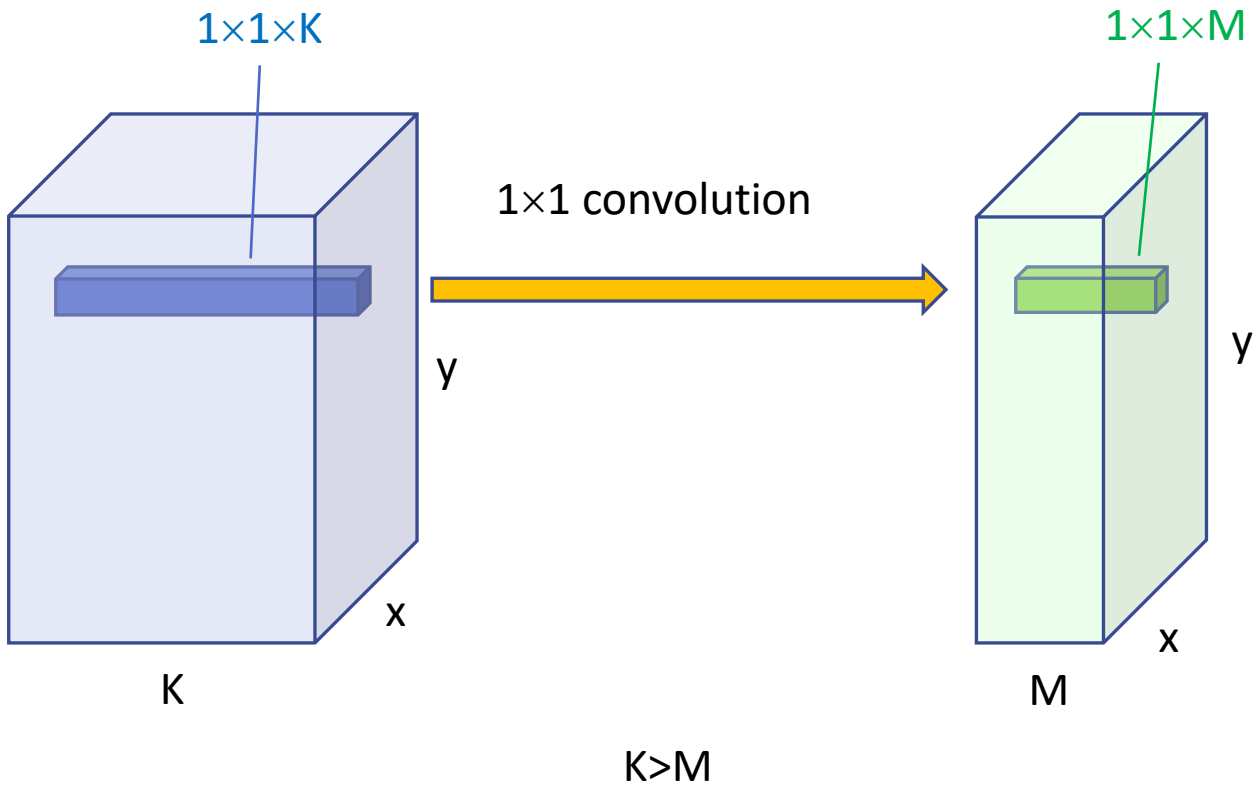
GoogLeNet

- Winner of 2014 ILSVRC competition
- Adopt Network-in-Network concept
- 22 layers (27 layers if including pooling)
- 12× fewer parameters than the AlexNet



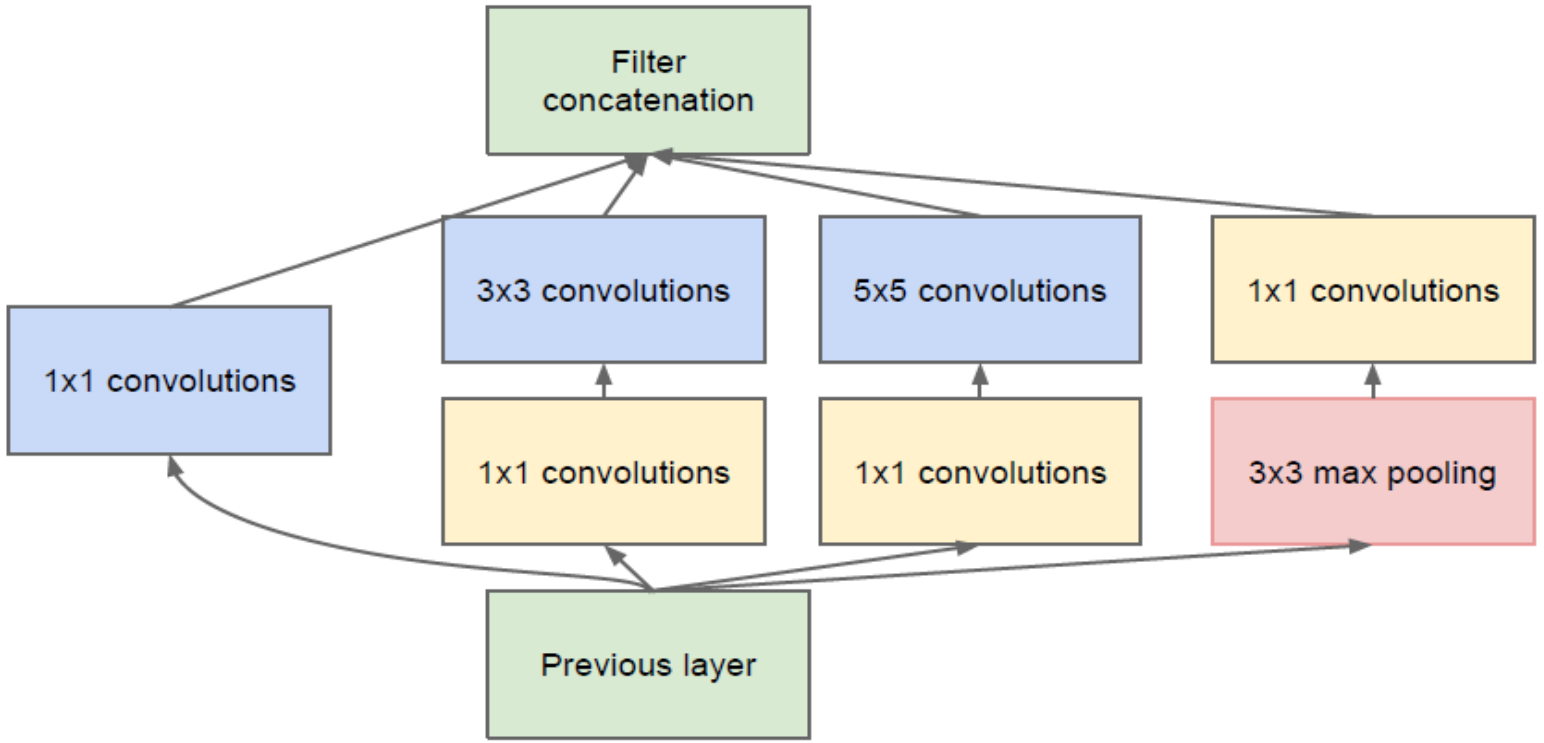
Inception Module

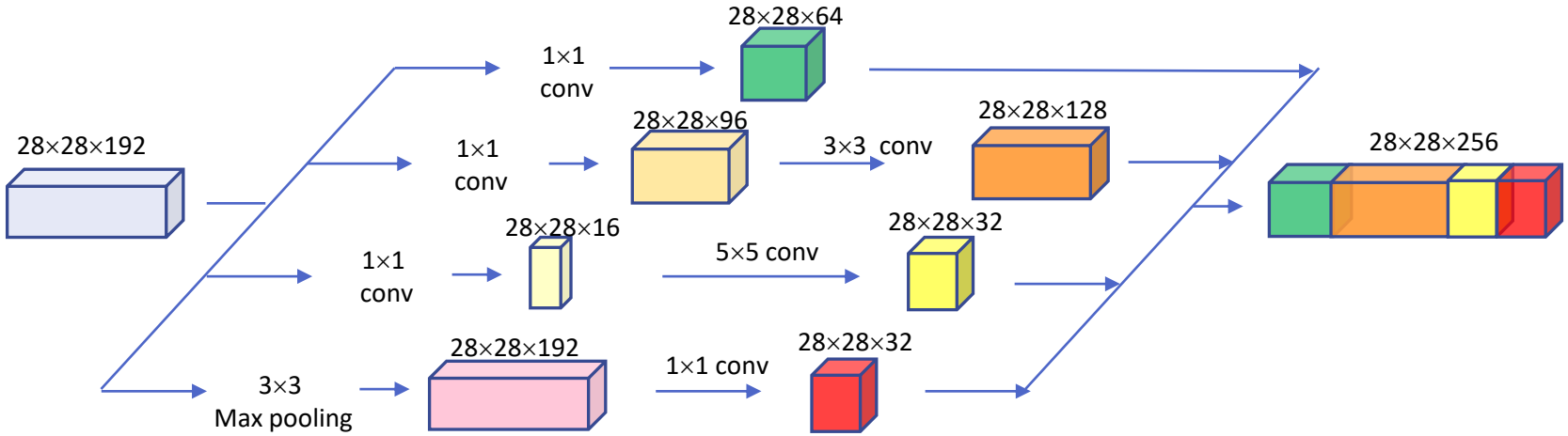



1×1×K
1×1 convolution
1×1×M
y
x
K
y
x
M
K>M

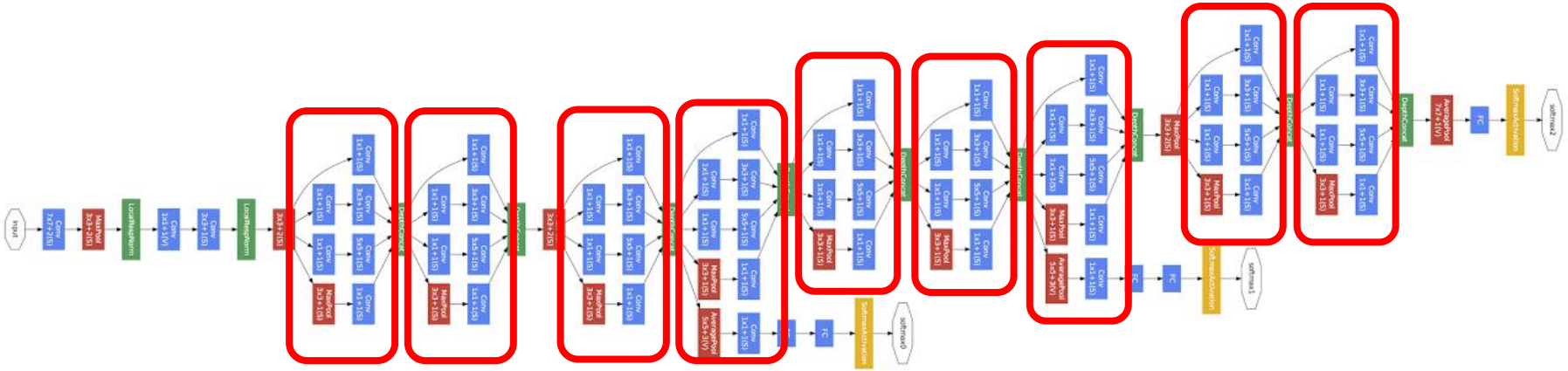
Inception Module with Dimension Reduction

- 1×1 convolutions are used for dimension reduction




28×28×192
1×1
conv
28×28×64
1×1
conv
28×28×96
3×3 conv
28×28×128
1×1
conv
28×28×16
5×5 conv
28×28×32
3×3
Max pooling
28×28×192
1×1 conv
28×28×32
28×28×256

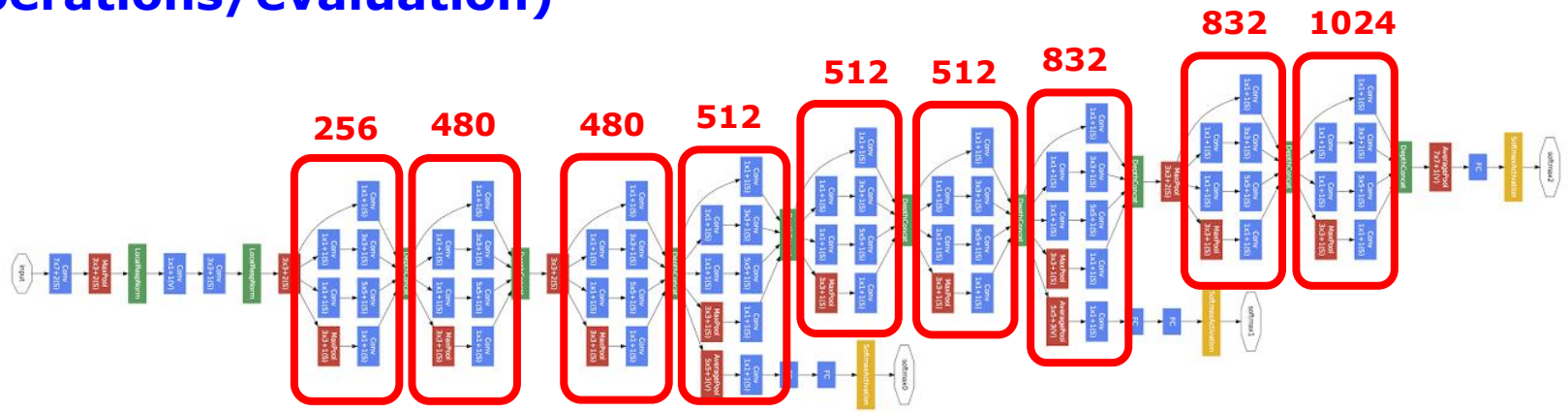
9 **Inception** modules



Convolution
Pooling
Softmax
Other

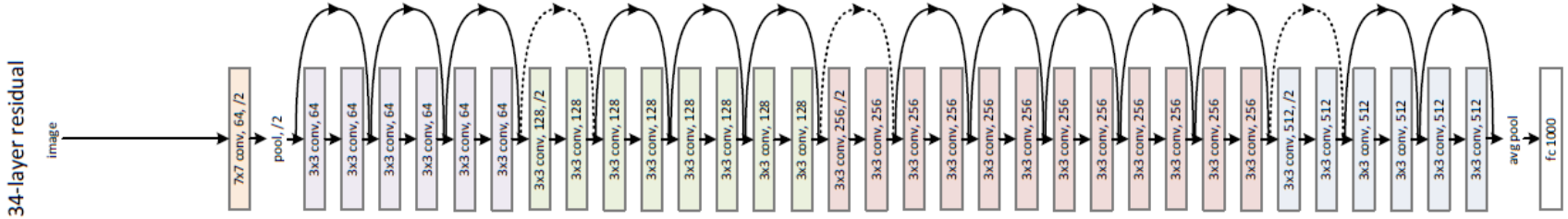
- Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.
- Can remove fully connected layers on top completely
- Number of parameters is reduced to 5 million

Computional cost is increased by less than 2X compared to AlexNet. (<1.5Bn operations/evaluation)



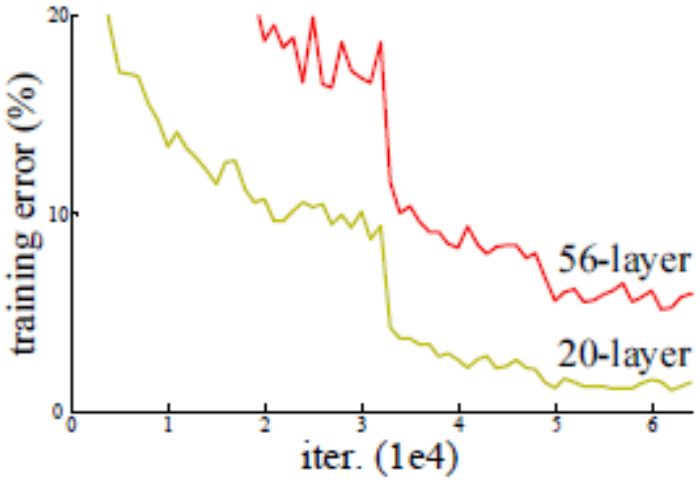
ResNet

- Winner of 2015 ILSVRC competition
- Adopt Residual Learning concept
- Use up to 152 layers on the ImageNet database

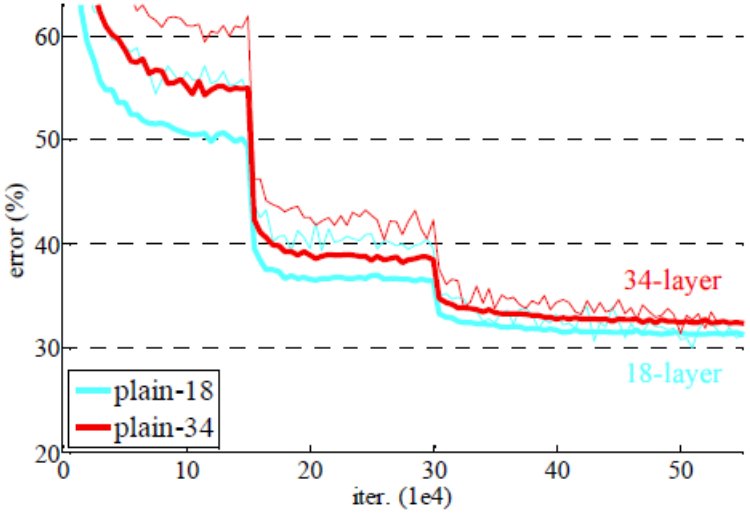


34-layer residual

Adding more layers leads to higher training error!

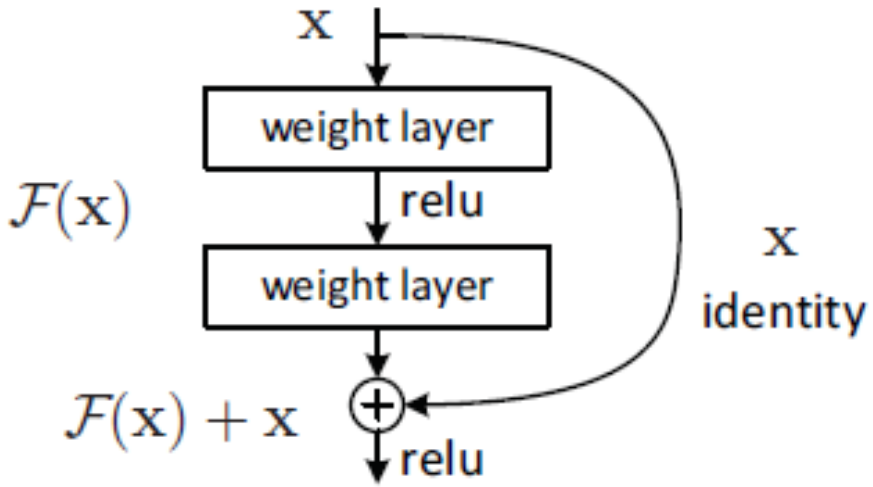


CIFAR-10 database



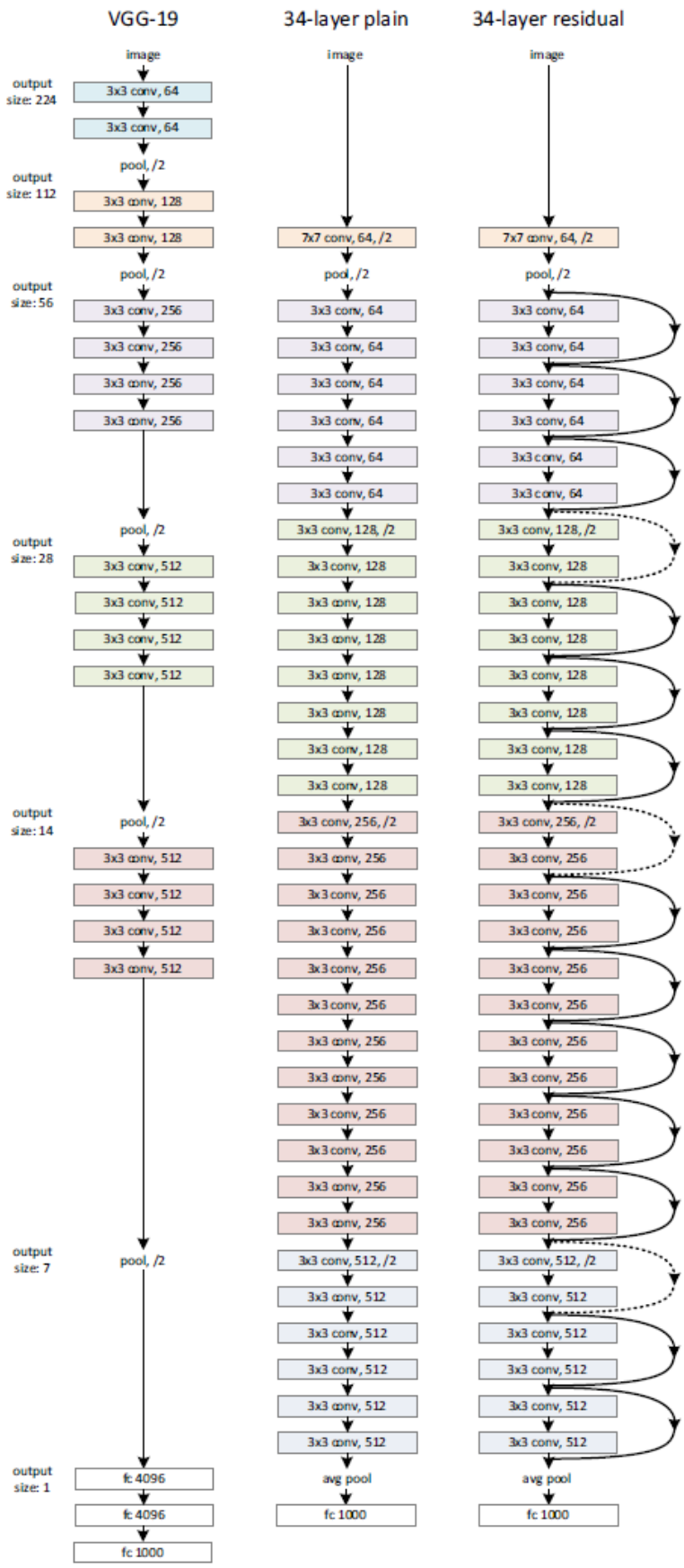
ImageNet database

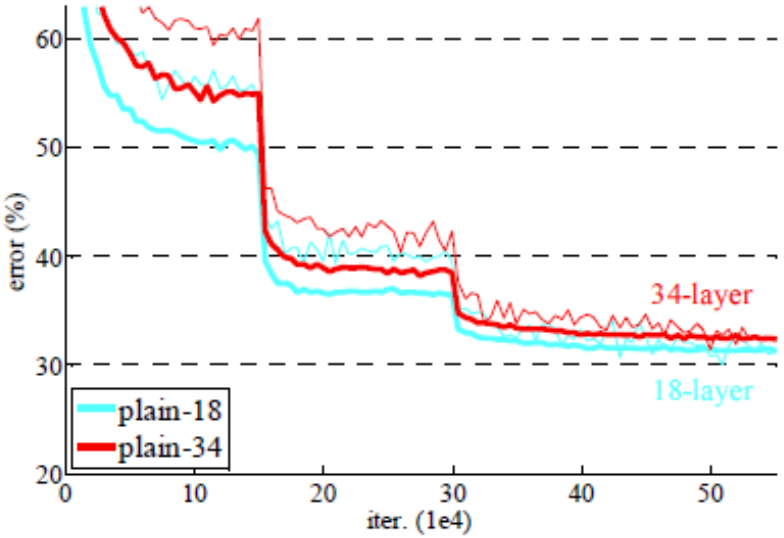
Residual Learning



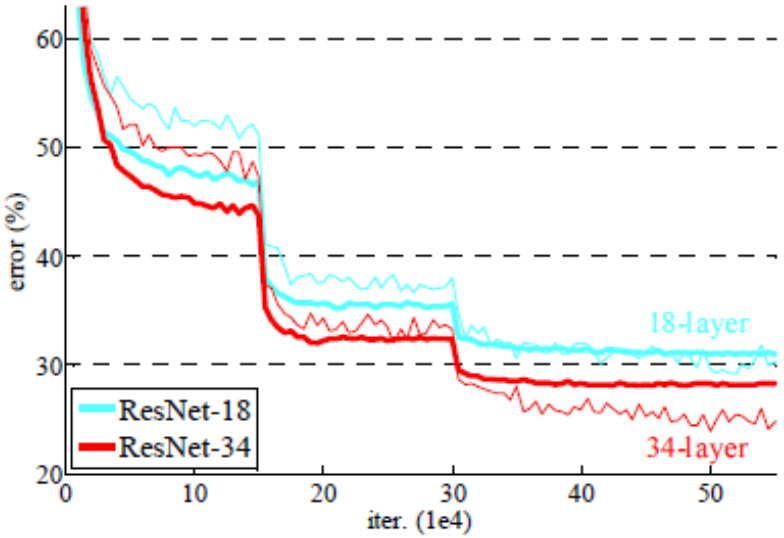
$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

or $$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$ if dim(**x**) ≠ dim(**y**)


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

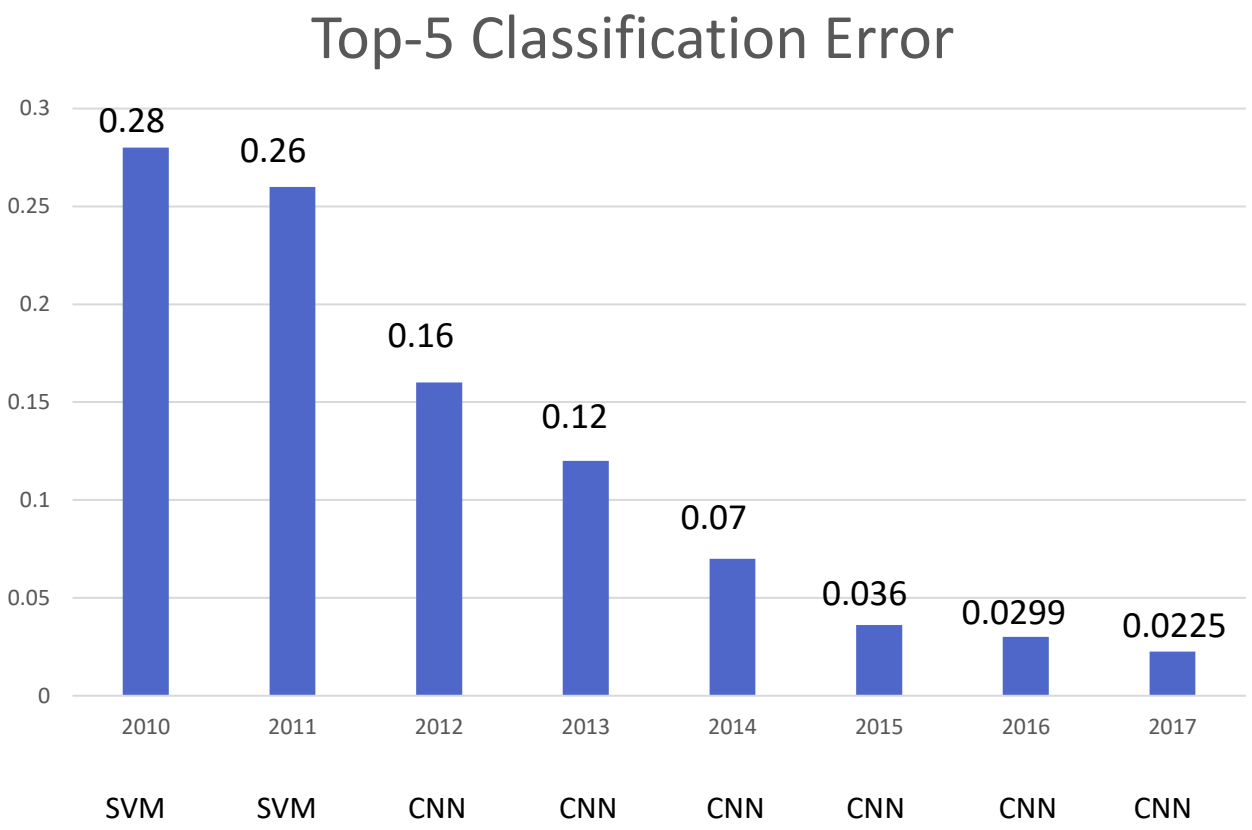


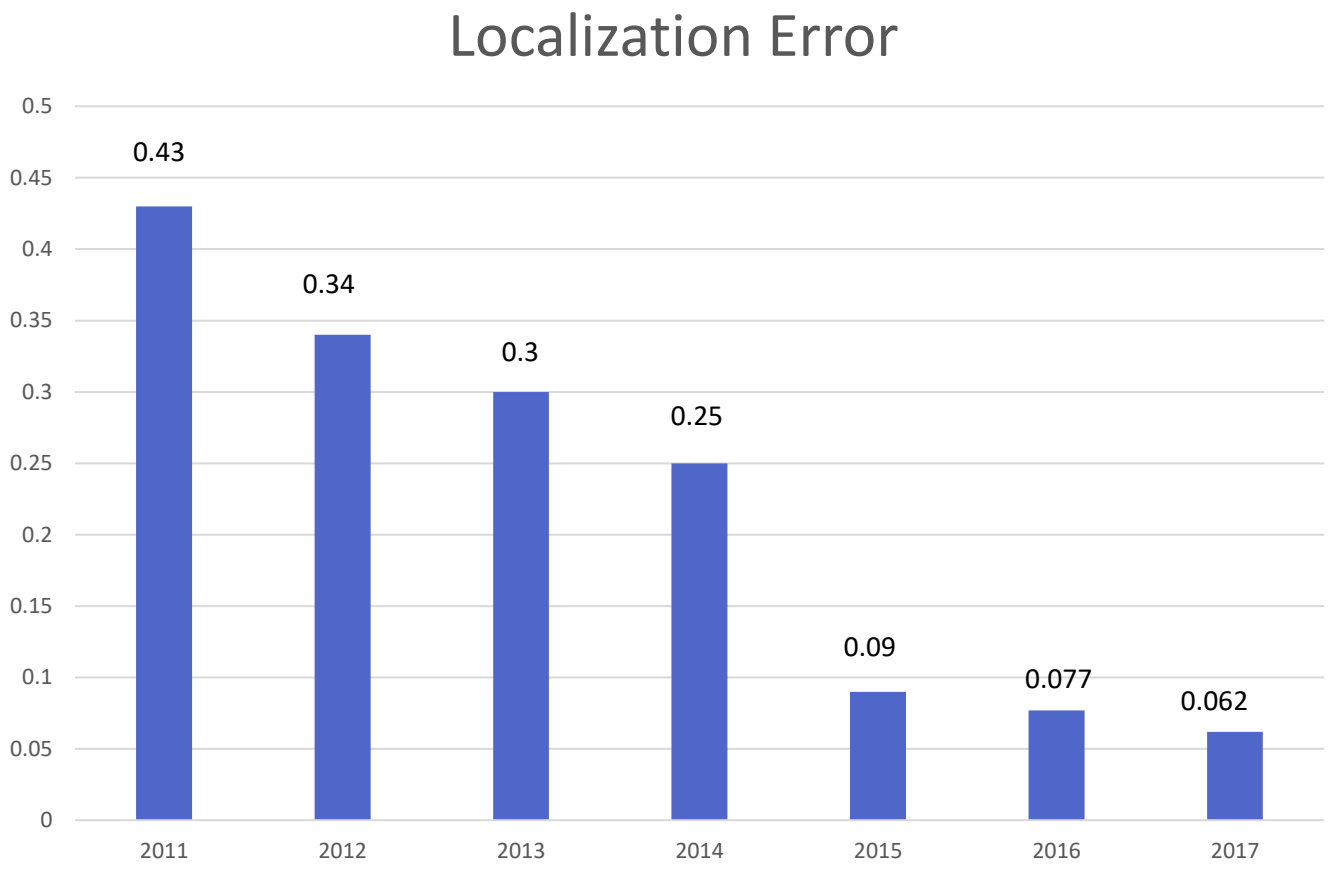
Plain Networks



Residual Networks

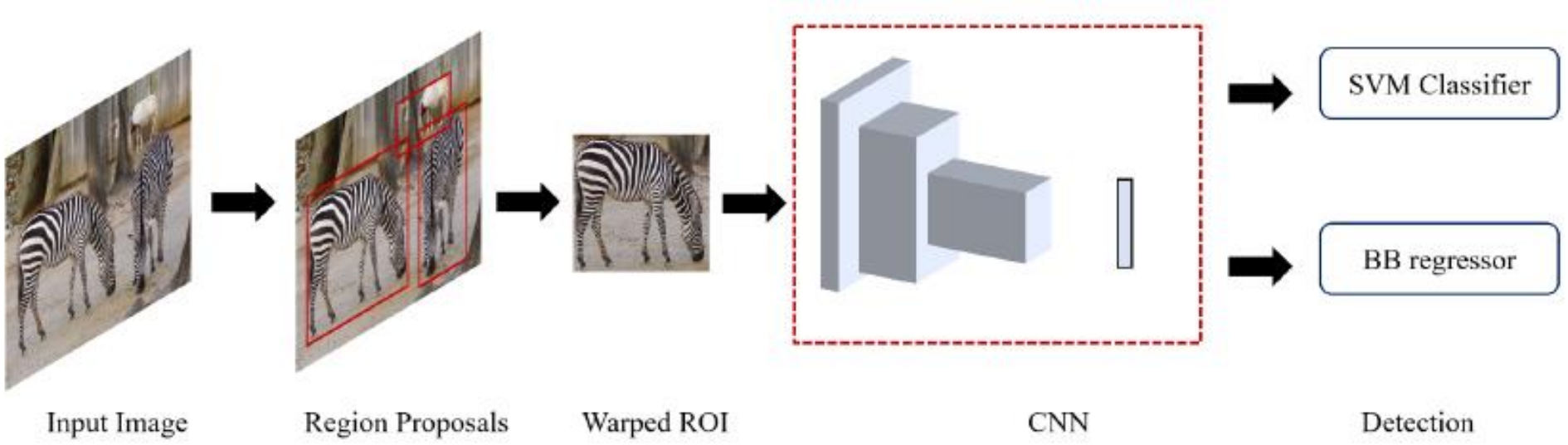
Classification Results of ILSVRC Challenge




Localization Error
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0.43
0.34
0.3
0.25
0.09
0.077
0.062
2011
2012
2013
2014
2015
2016
2017

Recent Evolution of CNN

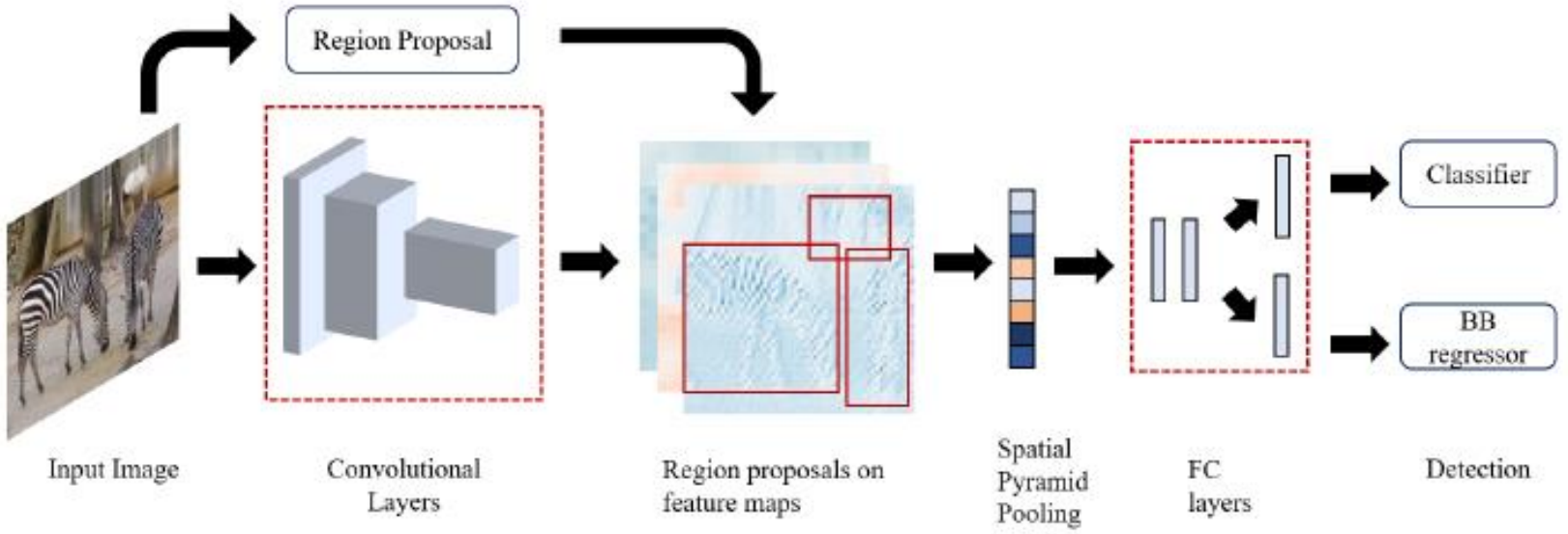
- 2014 R-CNN
  - ✓ Region Proposal + CNN



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

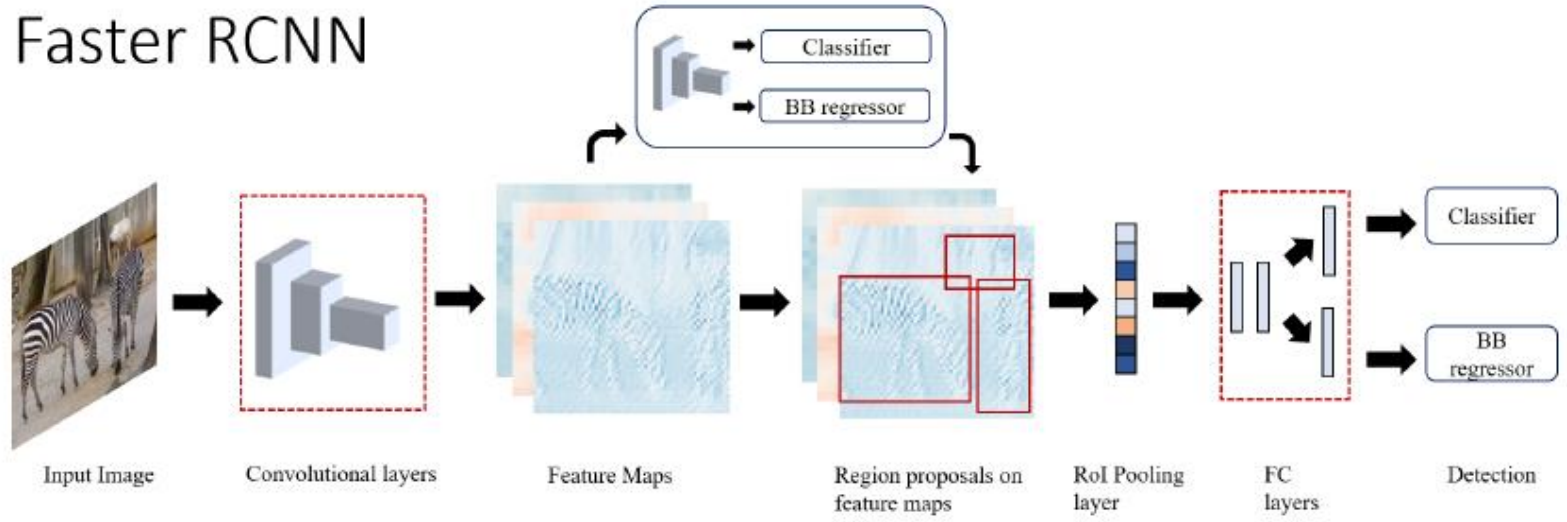
- 2015 Fast R-CNN:
  - Use CNN for both classification and localization (regression)



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

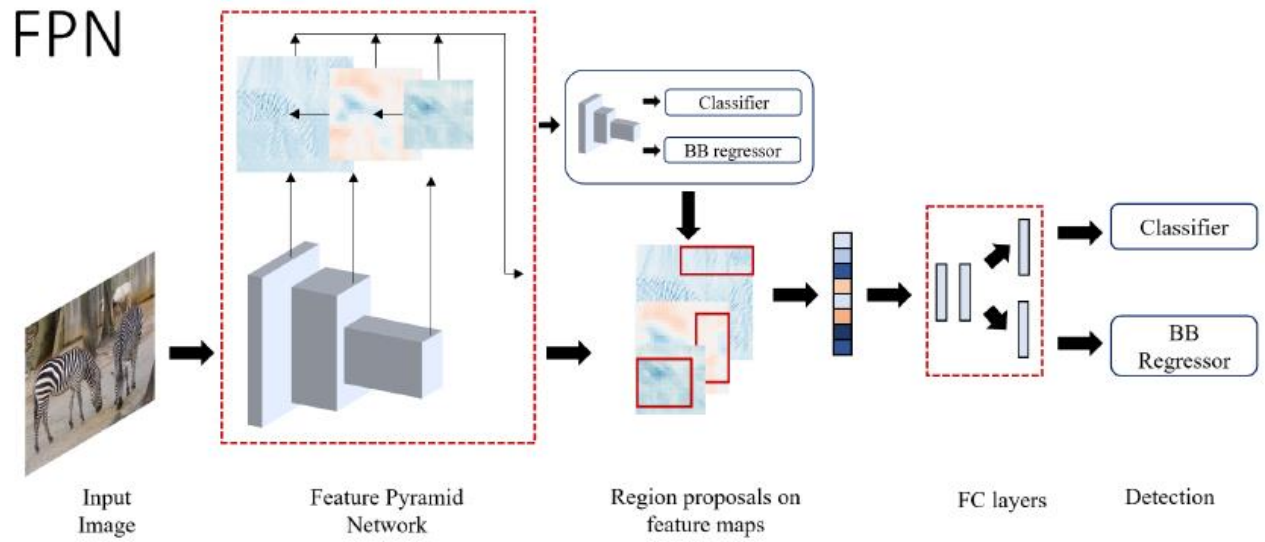
- 2016 Faster R-CNN:
  - Propose the use of RPN (Region Proposal Network) to speed up the generation of region proposals.



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

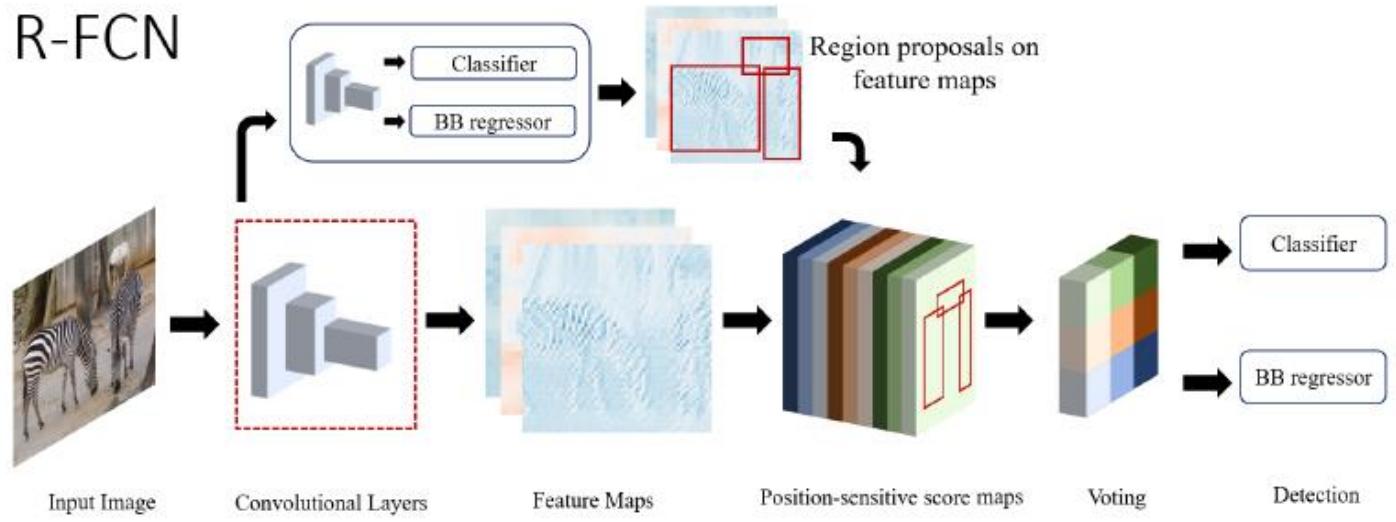
- 2017 FPN:
  - Propose FPN (Feature Pyramid Network) to extract multi-scale information.



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

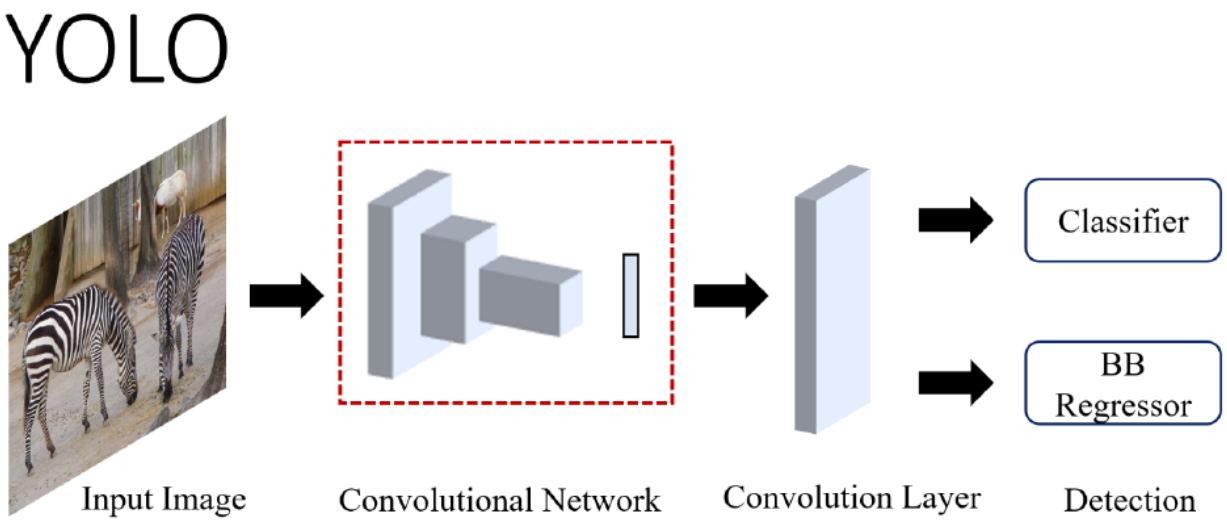
- 2016 R-FCN (Region-based Fully Convolutional Network)
  - Replace fully connected layers with convolutional layers.



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

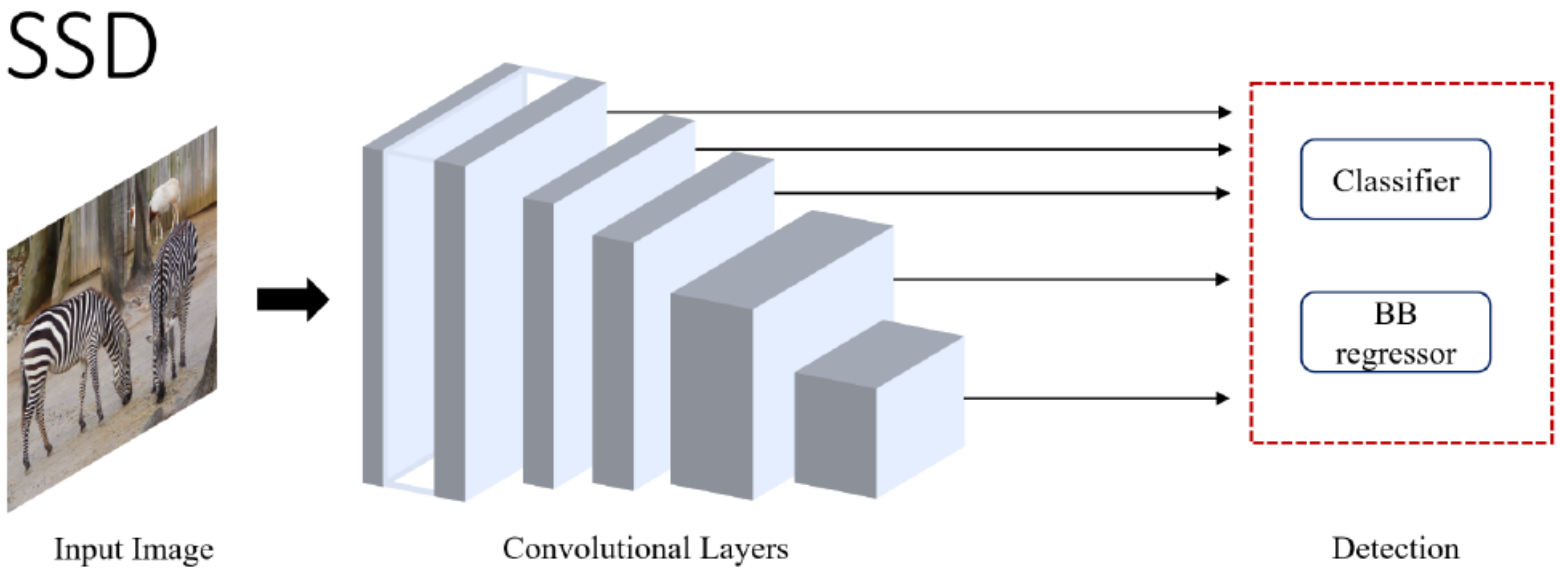
- 2016 YOLO (You Only Look Once)
  - Predict bounding boxes and class probabilities directly in one evaluation



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

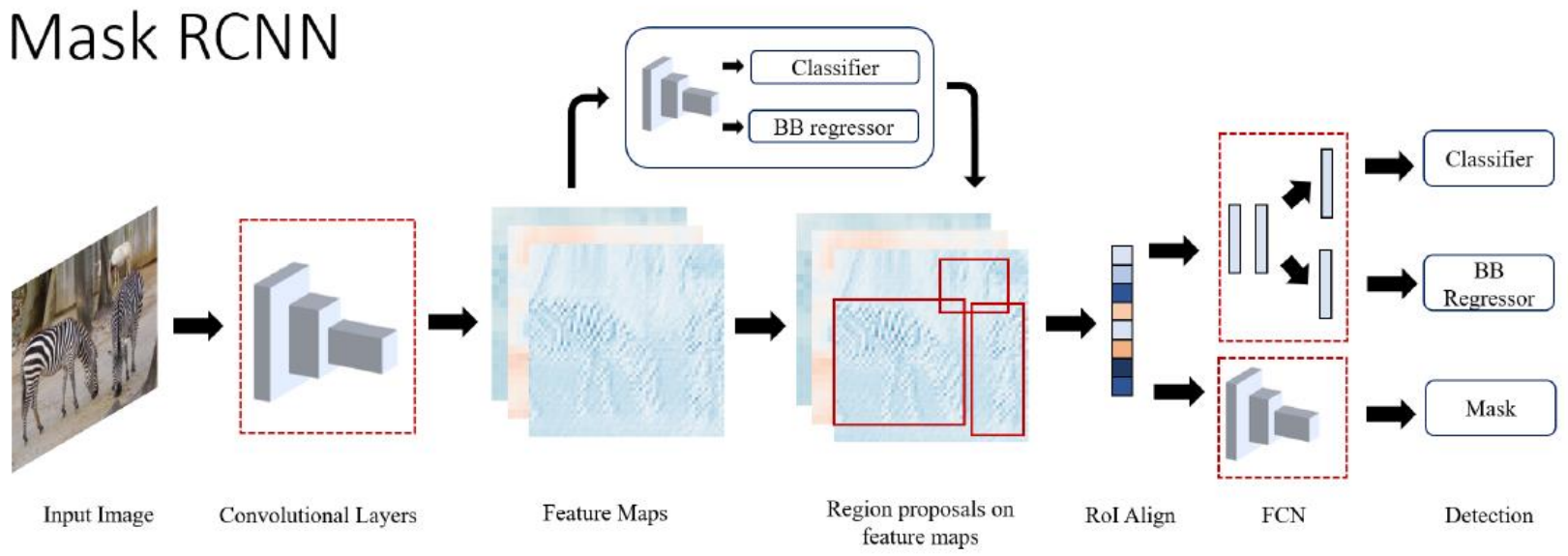
- 2016 SSD (Single-Shot MultiBox Detector)
  - Another popular one-stage detector



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

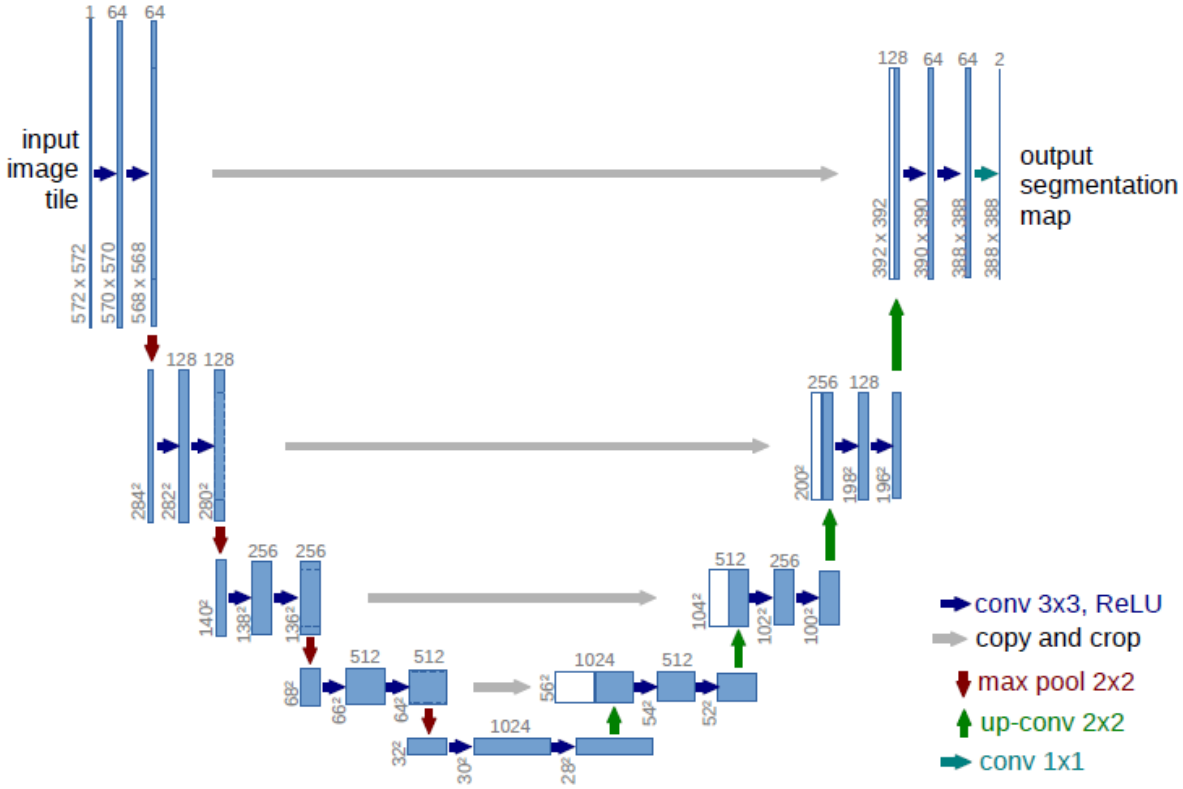
- 2017 MASK R-CNN:
  - Extend Faster R-CNN for pixel-wise segmentation



Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." Digital Signal Processing (2022): 103514.

Recent Evolution of CNN

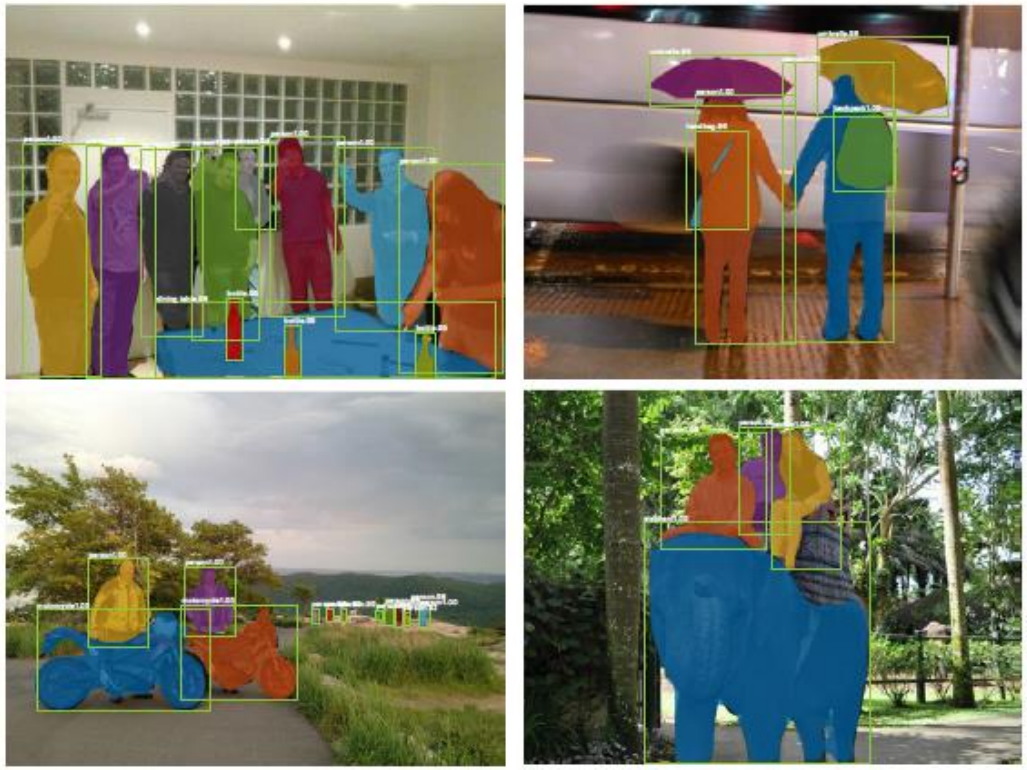
- 2015 U-Net:



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.

Recent Evolution of CNN

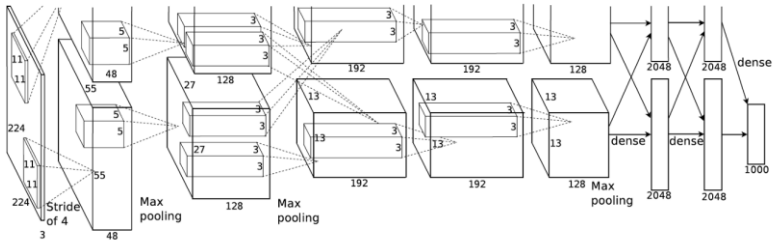
- MASK R-CNN:



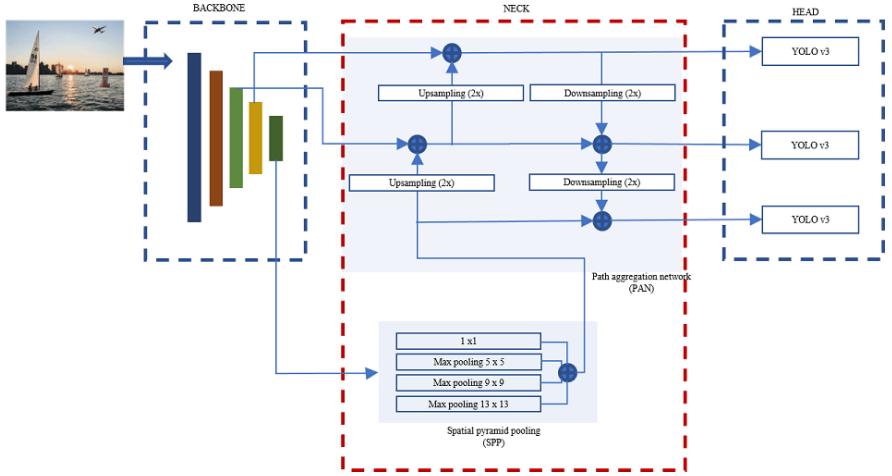
He, Kaiming, et al. "Mask r-cnn." *arXiv preprint arXiv:1703.06870* (2017).

Recent Evolution of CNN

Original Structure

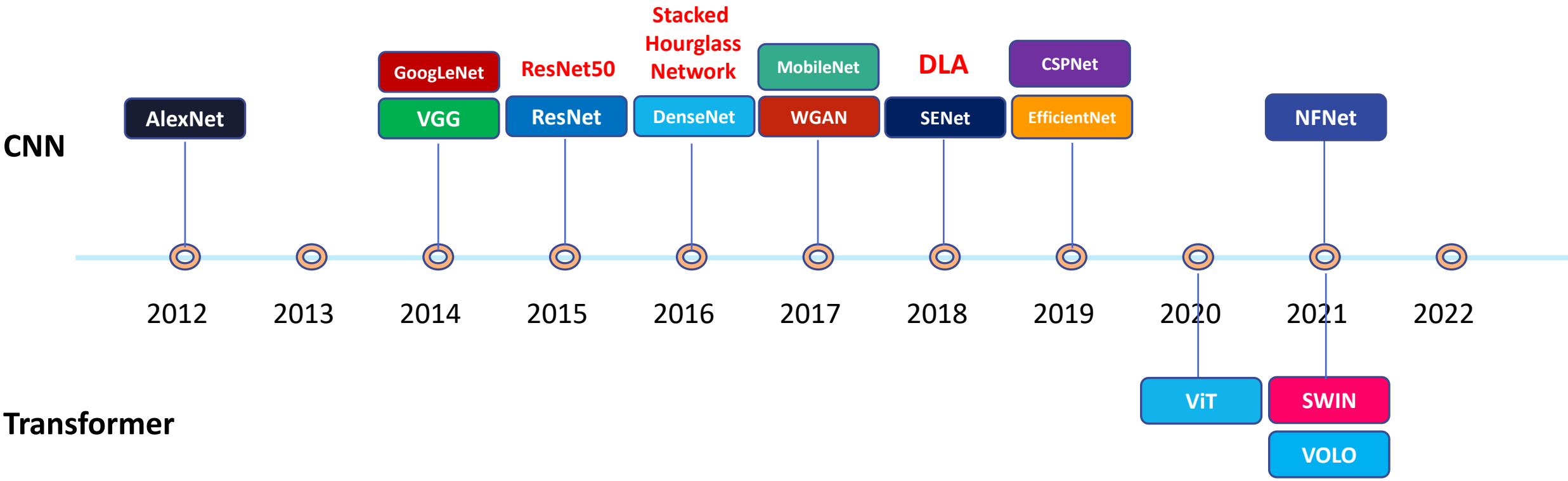


Modularized Structures



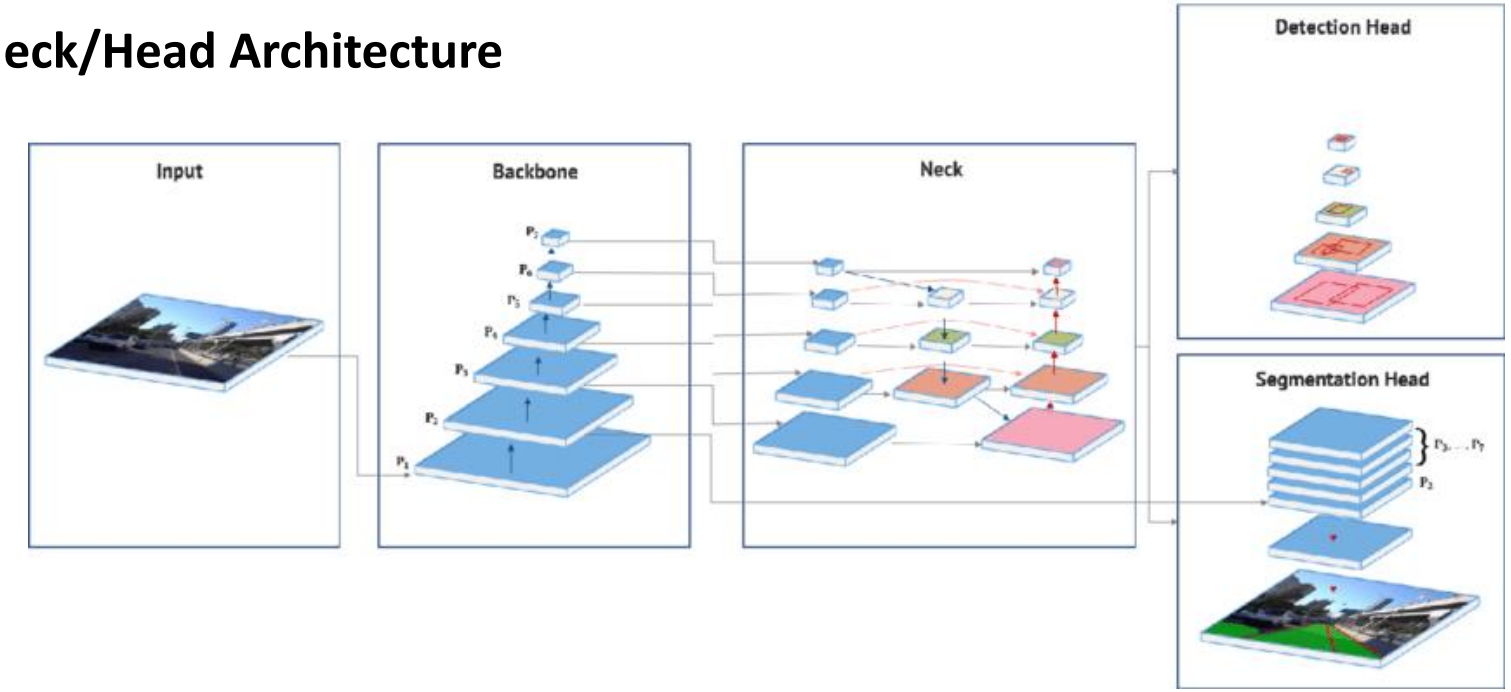
https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v4.html

Timeline of Popular Models



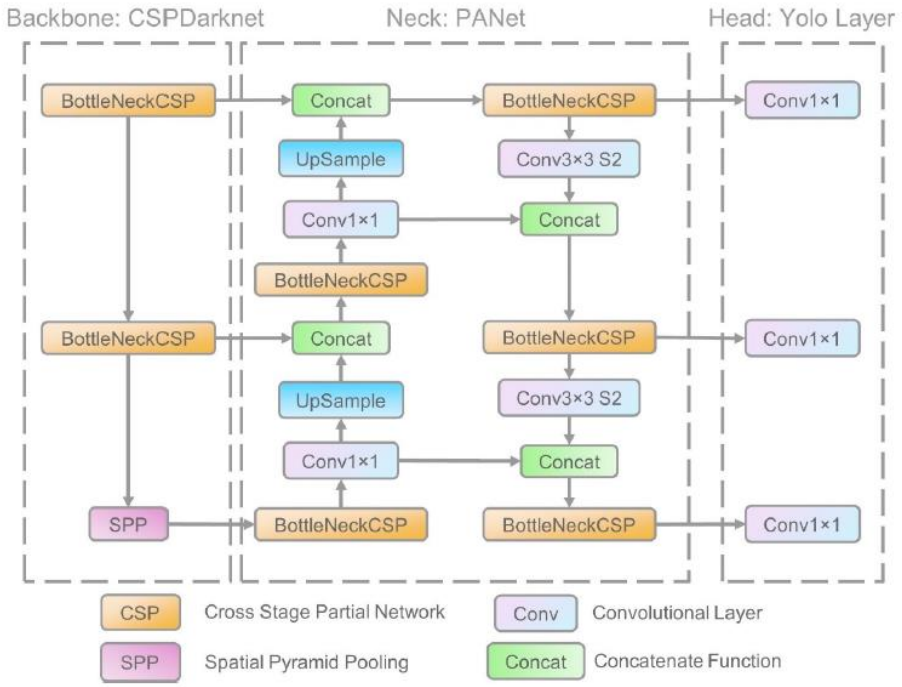
Recent Evolution of CNN

Backbone/Neck/Head Architecture

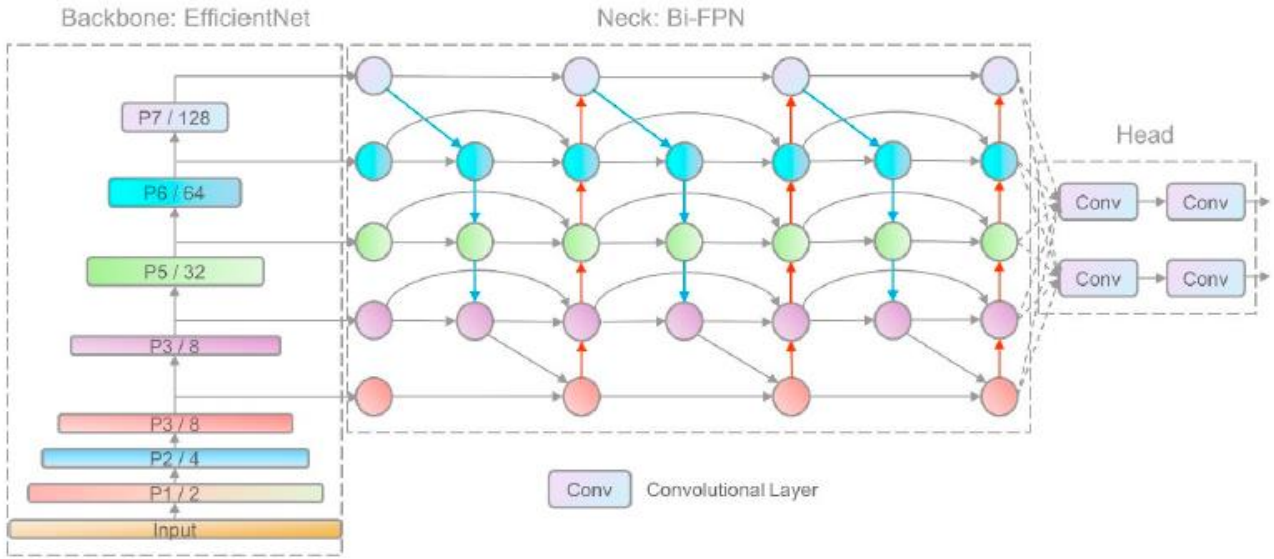


https://www.researchgate.net/figure/HybridNets-Architecture-has-one-encoder-backbone-network-and-neck-network-two-decoders_fig2_359309811

Examples of Recent CNN Structures



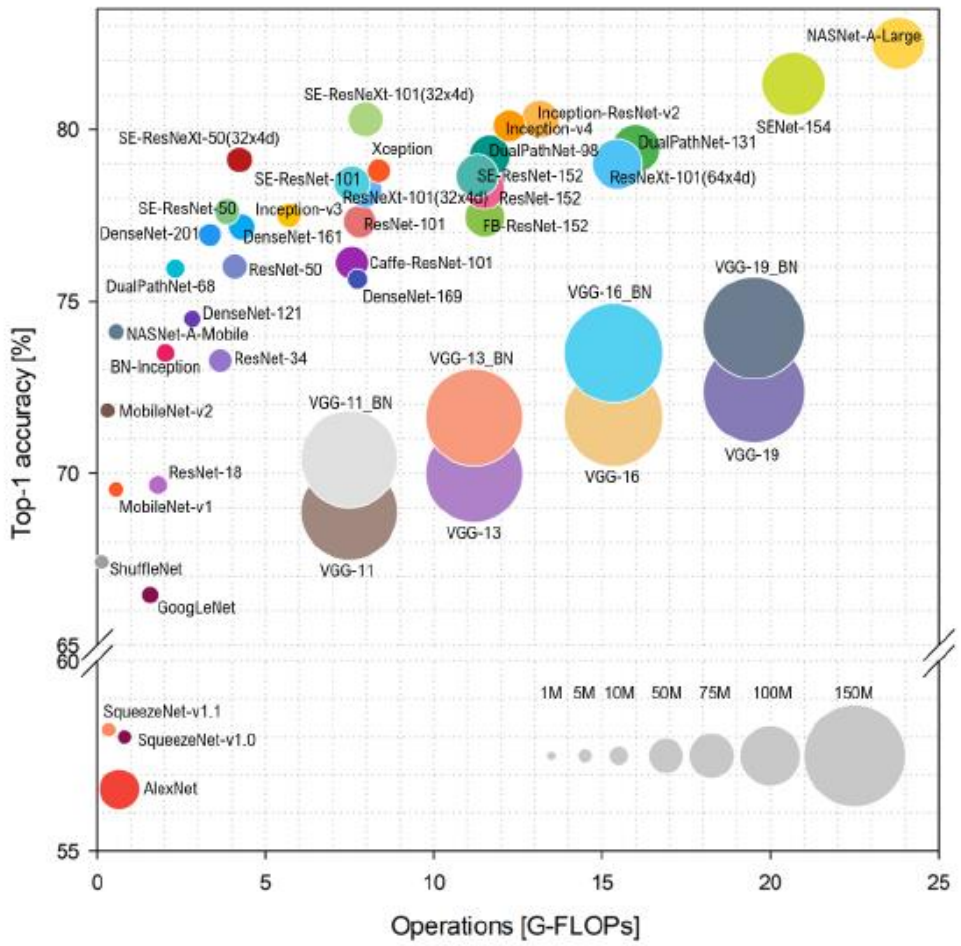
YOLOv5

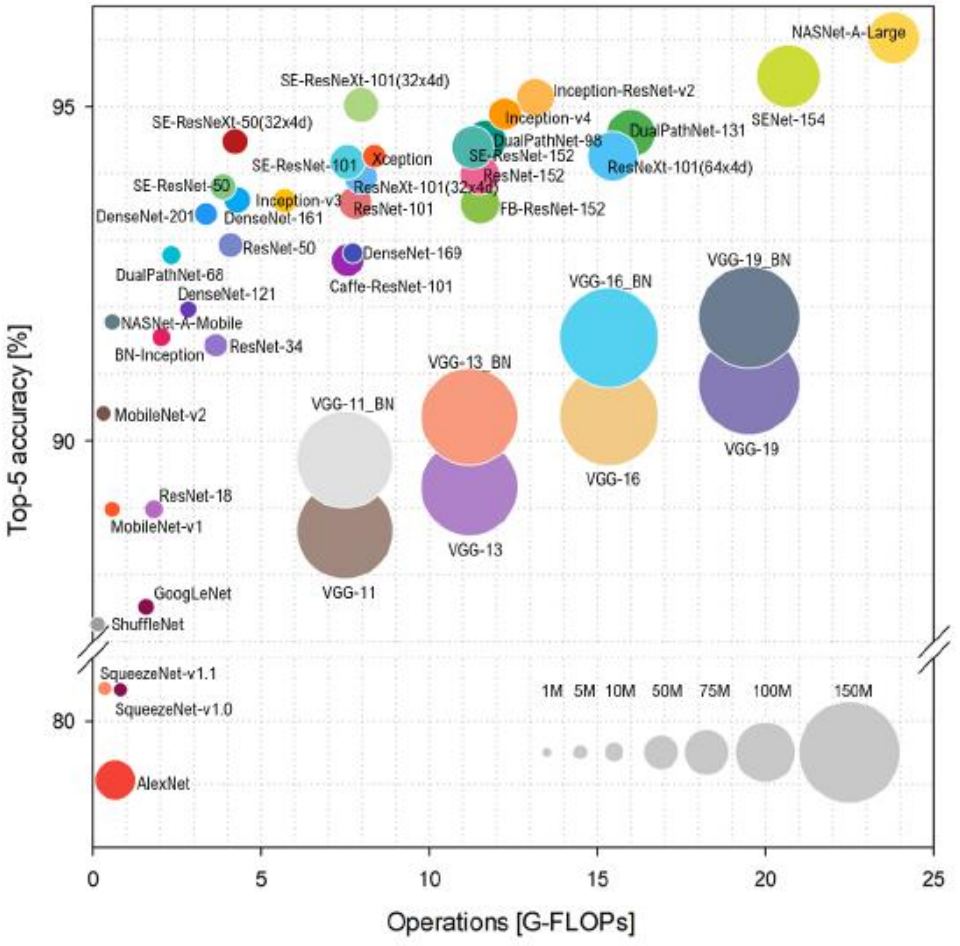


EfficientNet

Ref: Xu, R., Lin, H., Lu, K., Cao, L., & Liu, Y. (2021). A forest fire detection system based on ensemble learning. Forests, 12(2), 217.

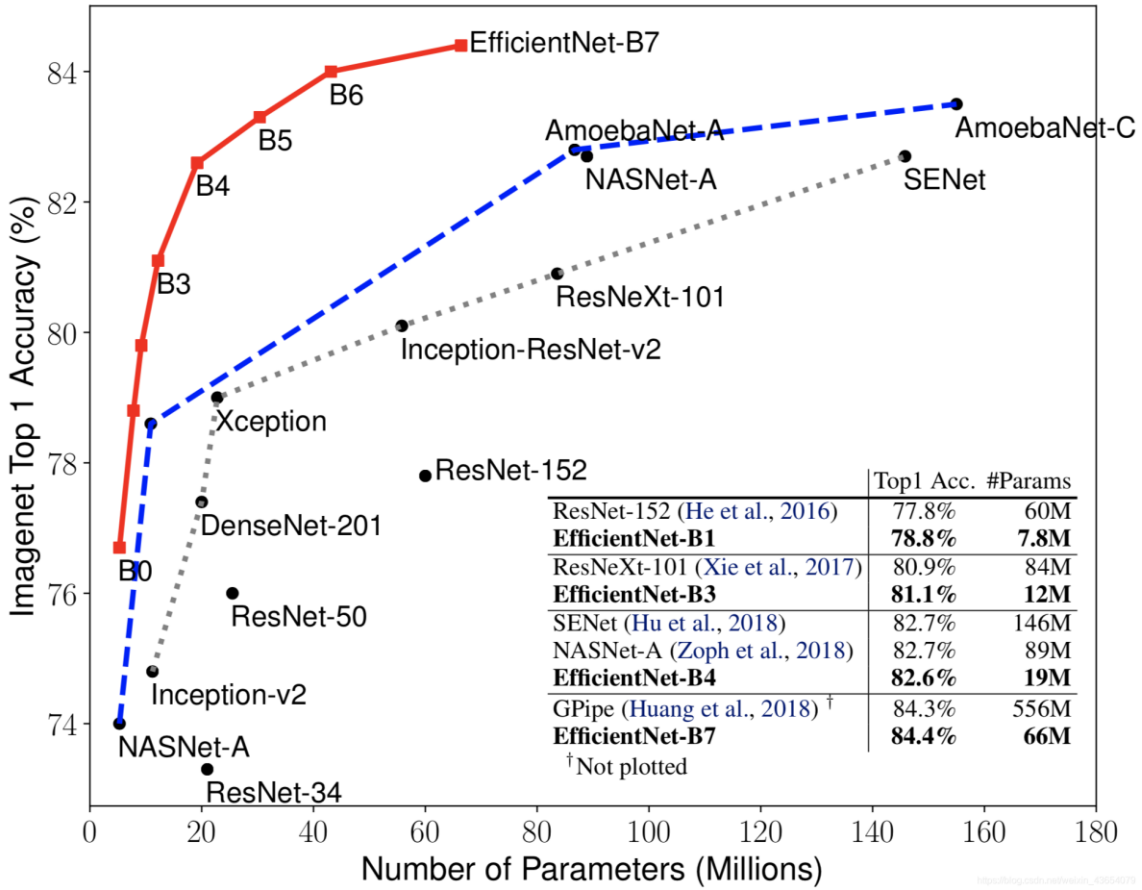
Image Classification on ImageNet (up to 2018)





Ref: Bianco, Simone, et al. "Benchmark analysis of representative deep neural network architectures." IEEE access 6 (2018): 64270-64277.

Improvement of DL Performance

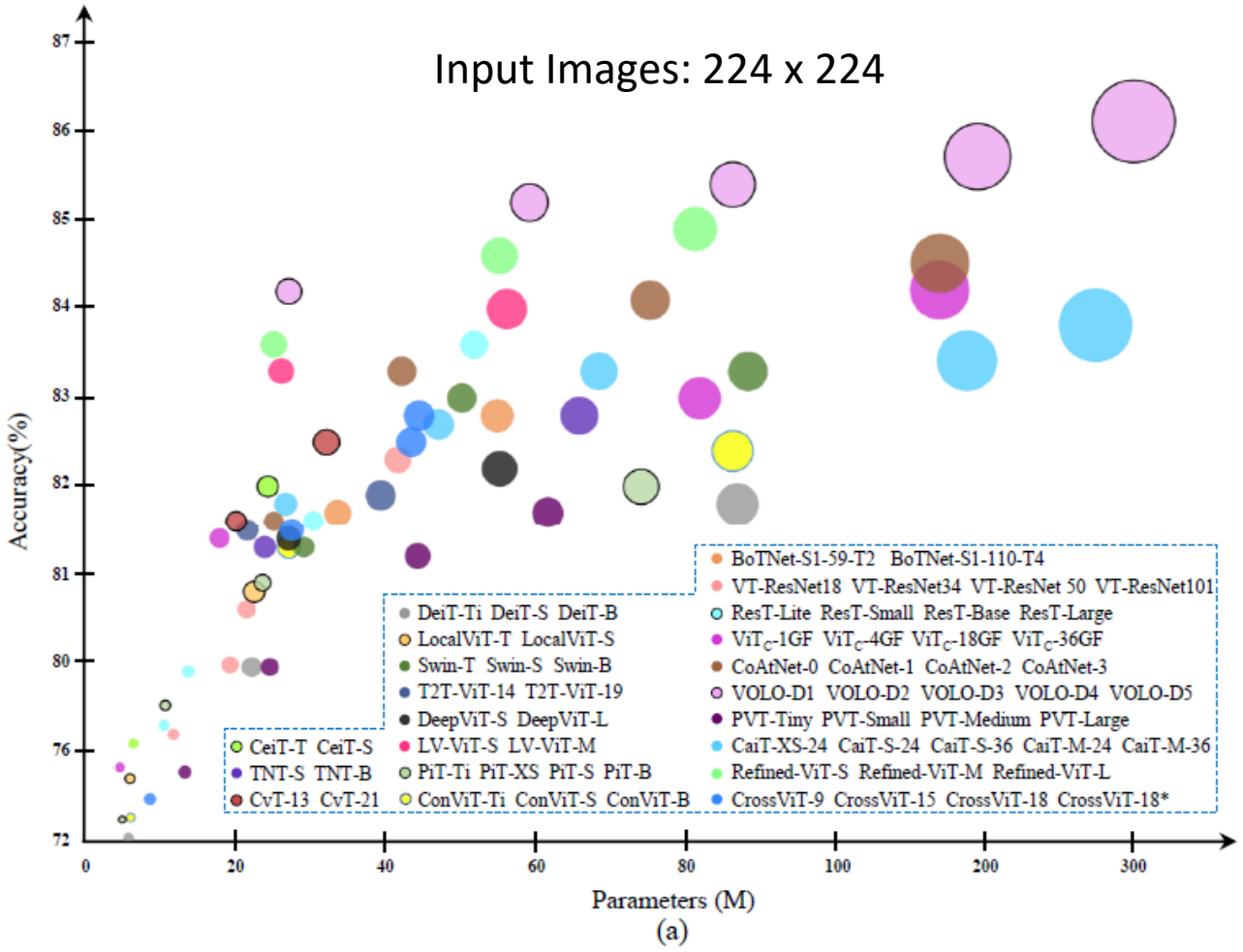


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

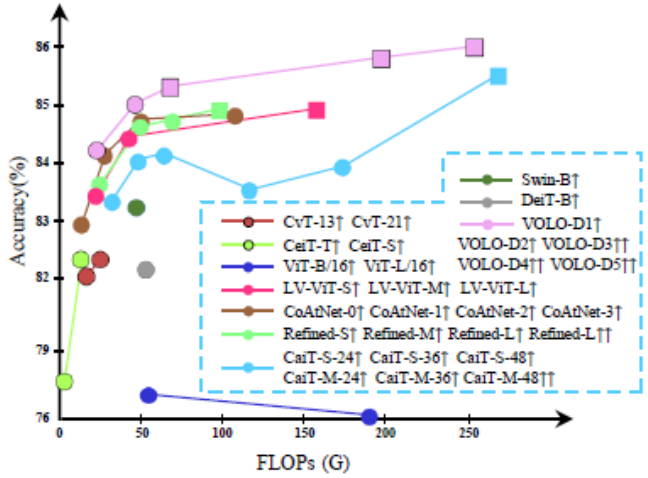
†Not plotted

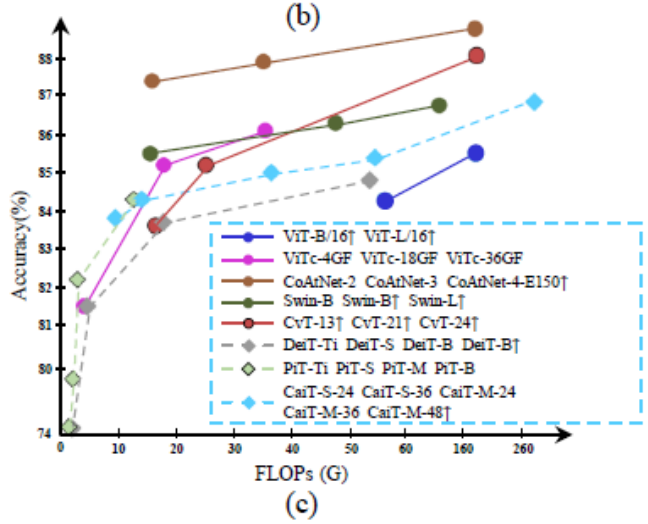
Image Classification on ImageNet (up to 2022)

Input Images: 224 x 224



Input Images: 448 x 448





Some pre-trained models on ImageNet-21K

Comparison with CNN Backbone Models

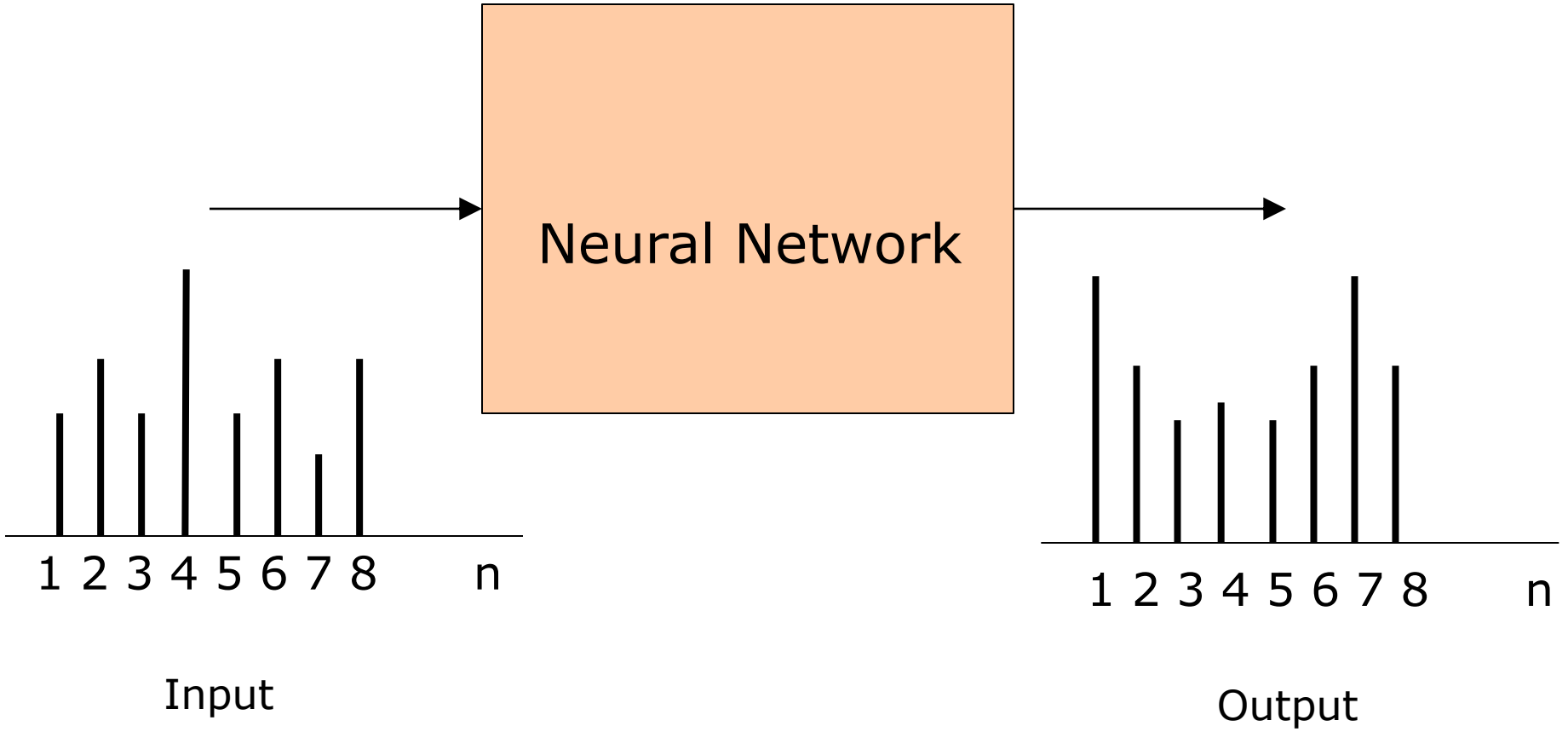
Table 3
Comparison of Backbone architectures.

Model	Year	Layers	Parameters (Million)	Top-1 acc%	FLOPs (Billion)
AlexNet	2012	7	62.4	63.3	1.5
VGG-16	2014	16	138.4	73	15.5
GoogLeNet	2014	22	6.7	-	1.6
ResNet-50	2015	50	25.6	76	3.8
ResNeXt-50	2016	50	25	77.8	4.2
CSPResNeXt-50	2019	59	20.5	78.2	7.9
EfficientNet-B4	2019	160	19	83	4.2

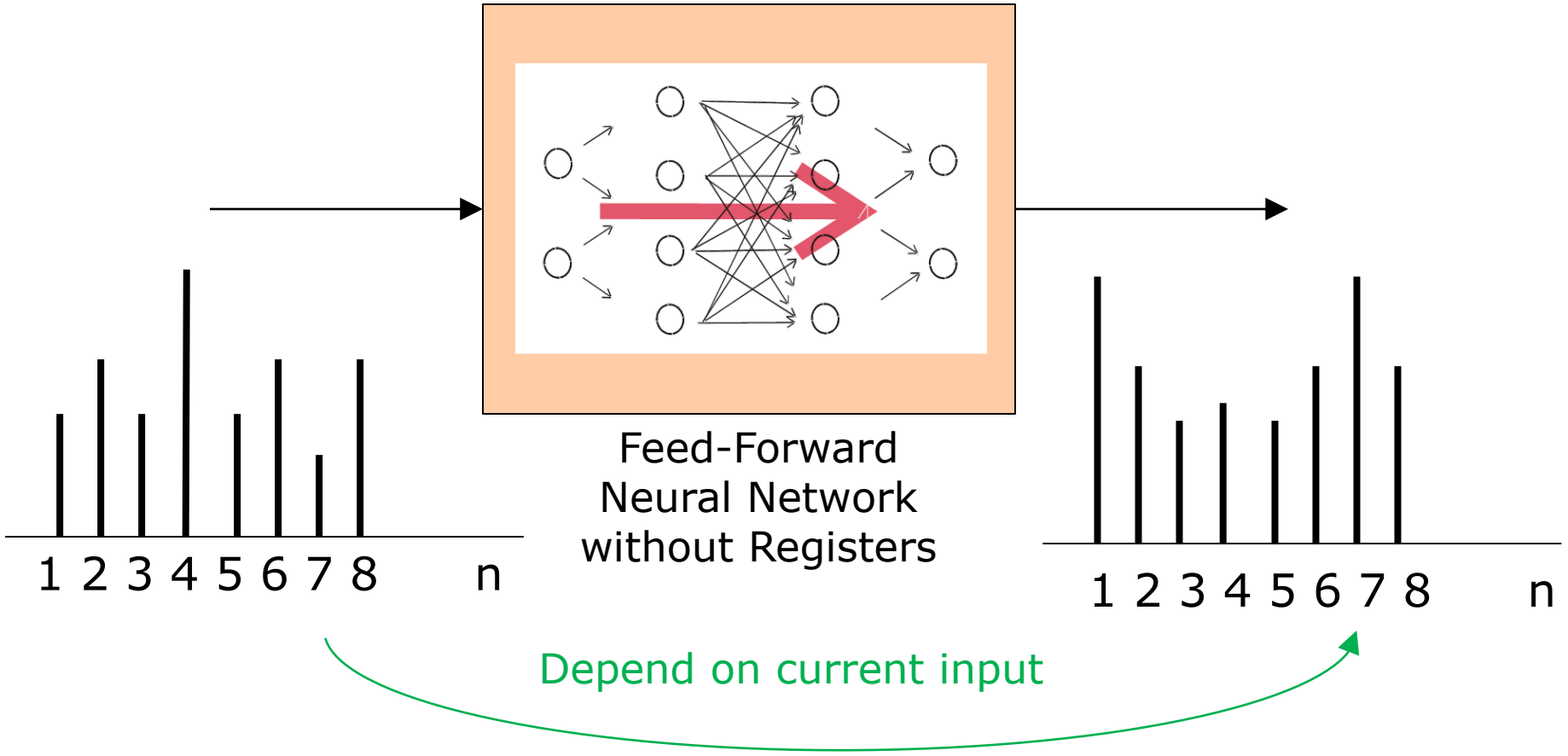
Backbone	Complexity (GFLOPs)	Top-1 err(%)	Top-5 err(%)
AlexNet [6]	0.72	42.8	19.7
GoogLeNet	1.5	-	7.89
VGG-16 [10]	15.3	28.5	9.9
ResNet-101 [17]	7.6	19.87	4.60
ResNet-152 [17]	11.3	19.38	4.49
DenseNet-121 [18]	0.525	25.02	7.71
DenseNet-264 [18]	1.125	22.15	6.12
DetNet-50 [35]	3.8	24.1	-
DetNet-59 [35]	4.8	23.5	-
DetNet-101 [35]	7.6	23.0	-
SqueezeNet [36]	0.861	42.5	19.7
ResNeXt-50 [40]	4.1	22.2	-
ResNeXt-101 (32x4d)[40]	7.8	21.2	5.6
ResNeXt-101 (64x4d)[40]	15.6	20.4	-
Xception [41]	11	21.0	5.5
BN-Inception [19]	2	22.0	5.8
Inception-v2 [20]	-	21.2	5.6
Inception-v3 [20]	5.72	18.7	4.2
Inception-ResNet-v1 [21]	-	21.3	5.5
Inception-v4 [21]	12.27	20.0	5.0
Inception-ResNet-v2 [21]	13.1	19.9	4.9
WideResNet-50 [37]	-	21.9	5.79
MobileNet-224 [38]	0.569	29.4	-
ShuffleNet-v1-50[42]	2.3	25.2	-
ShuffleNet-v2-50 [43]	2.3	22.8	-
EfficientNet-B0 [44]	0.39	22.9	6.7
EfficientNet-B7 [44]	37	**15.7**	**3.0**

Recurrent Neural Networks

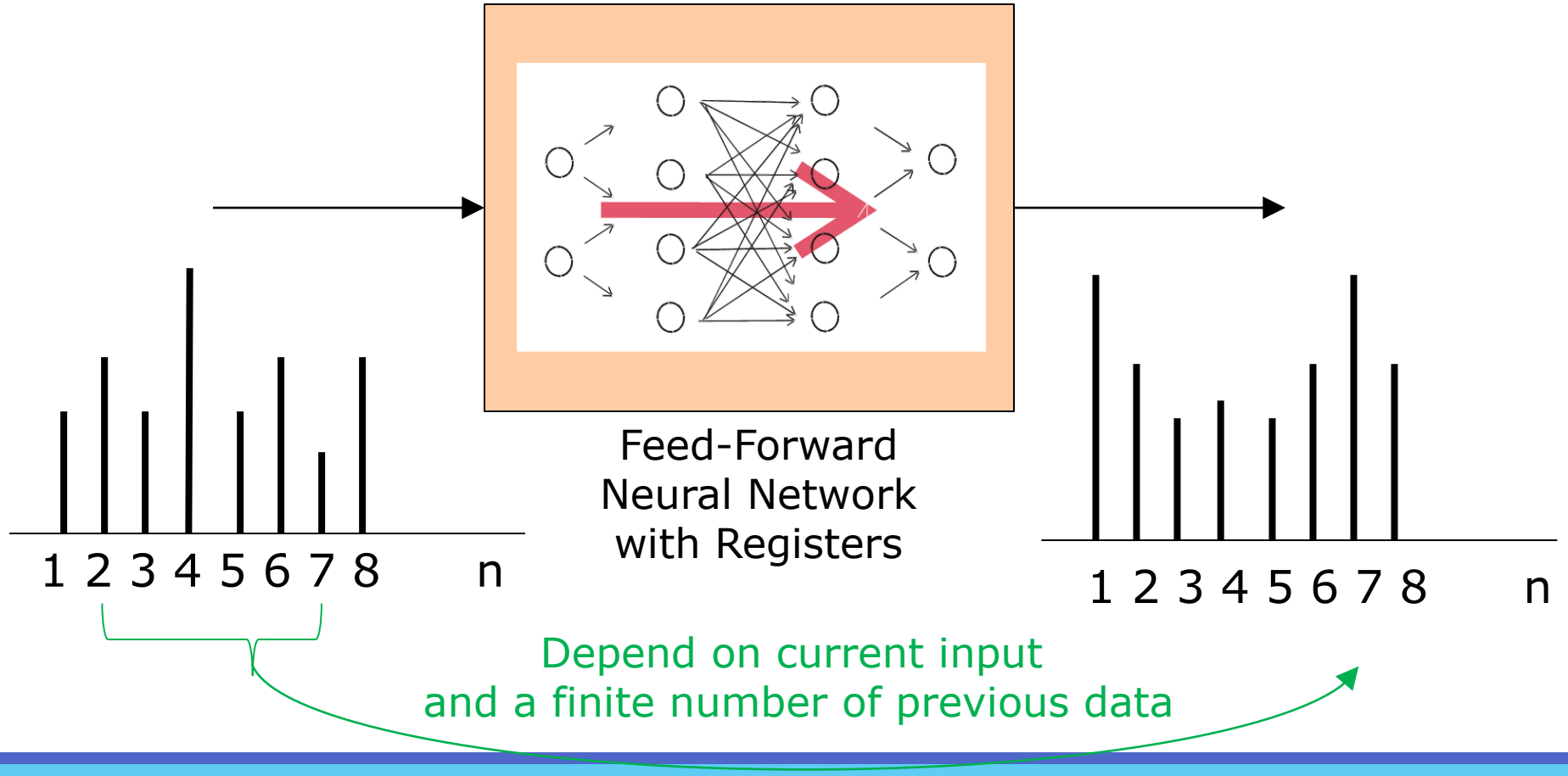
Overview



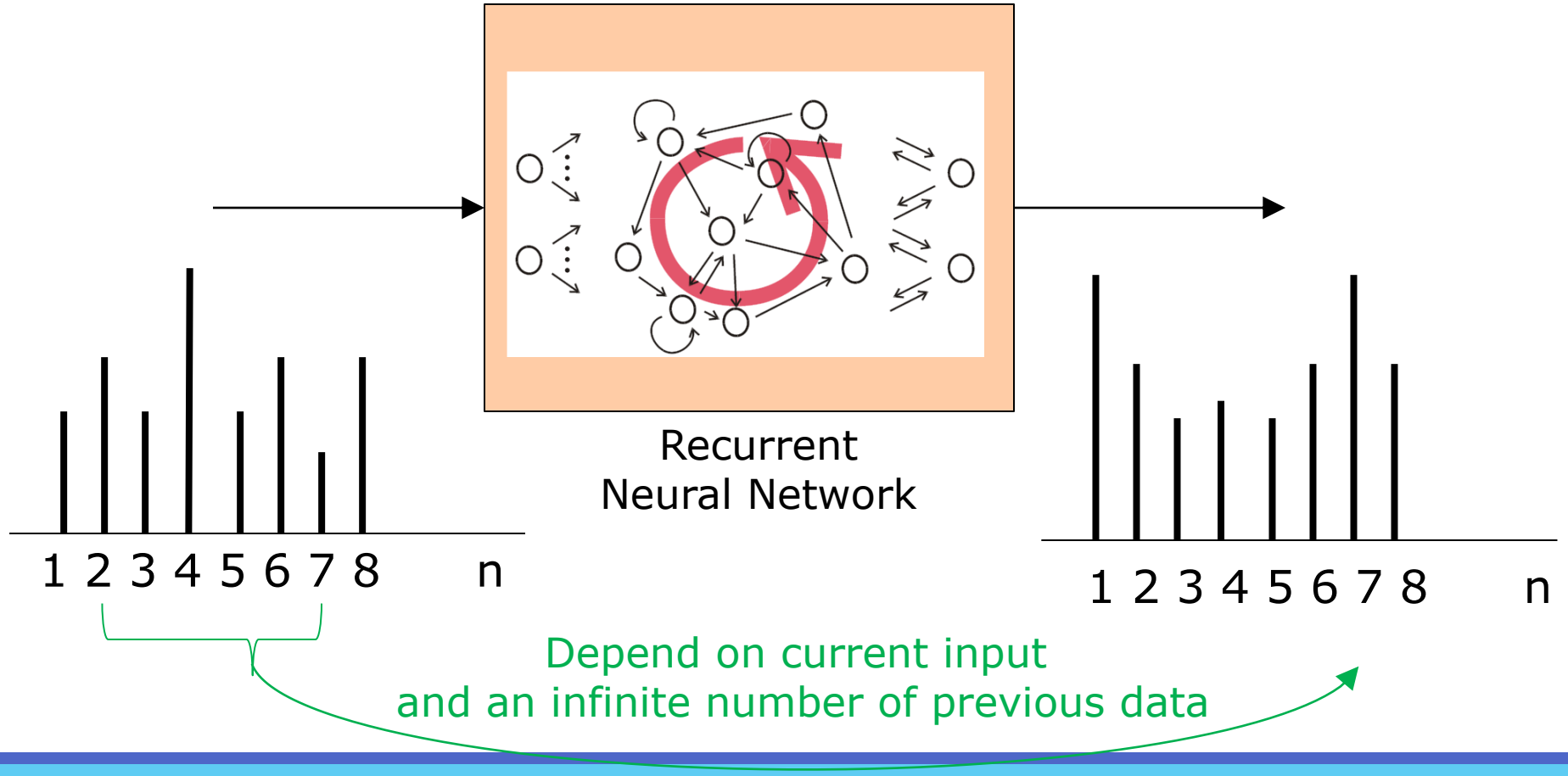
Overview


1 2 3 4 5 6 7 8 n
Feed-Forward
Neural Network
without Registers
1 2 3 4 5 6 7 8 n
Depend on current input

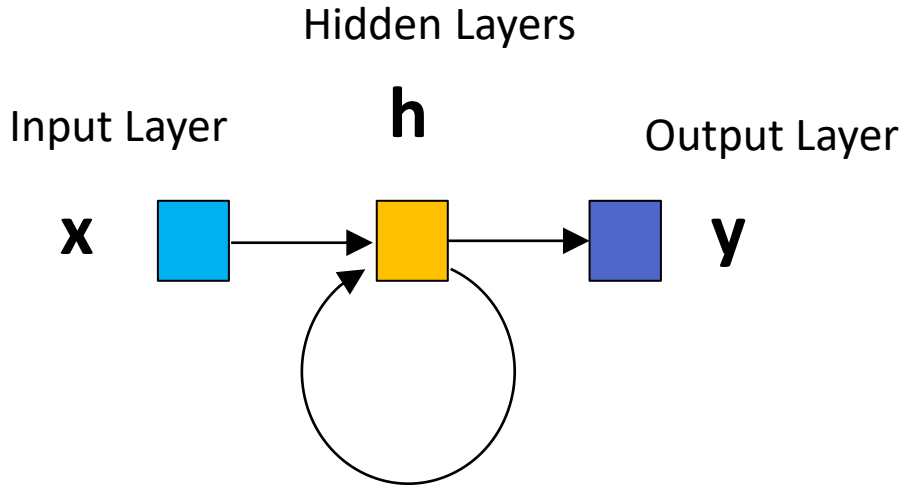
Overview


1 2 3 4 5 6 7 8 n
Feed-Forward
Neural Network
with Registers
1 2 3 4 5 6 7 8 n
Depend on current input
and a finite number of previous data

Overview


Recurrent
Neural Network
1 2 3 4 5 6 7 8 n
1 2 3 4 5 6 7 8 n
Depend on current input
and an infinite number of previous data

Recurrent Neural Network



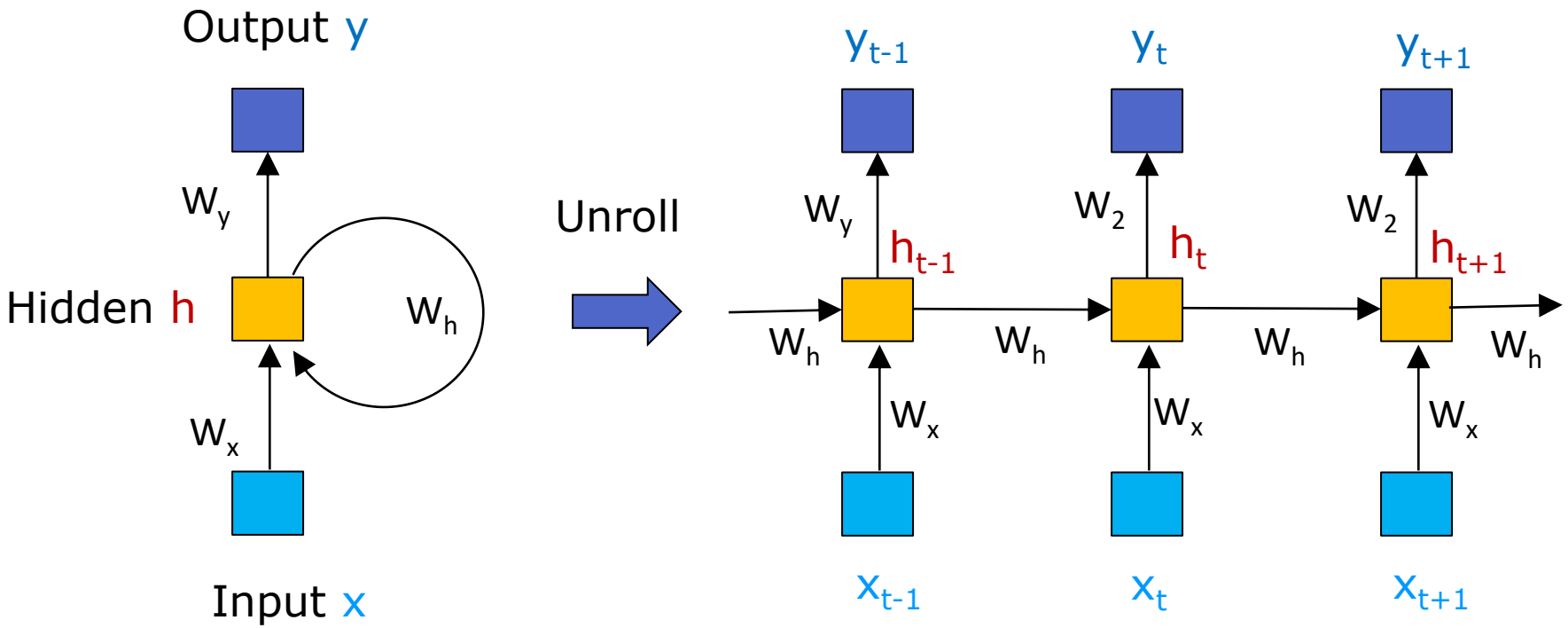
$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1})$$
$$\mathbf{y}_t = f_o(\mathbf{h}_t),$$

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} d(\mathbf{y}_t^{(n)}, f_o(\mathbf{h}_t^{(n)}))$$

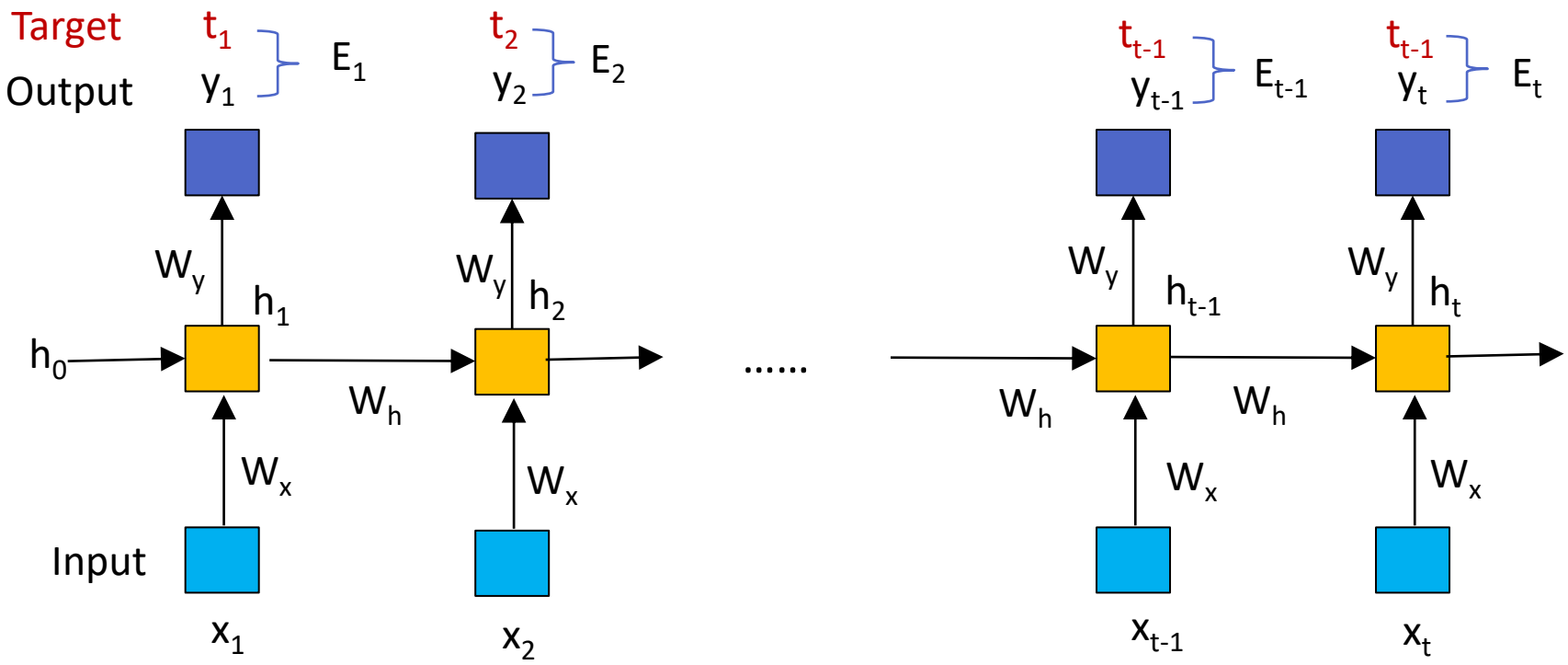
where $\mathbf{h}_t^{(n)} = f_h(\mathbf{x}_t^{(n)}, \mathbf{h}_{t-1}^{(n)})$ and $\mathbf{h}_0^{(n)} = \mathbf{0}$.

Back Propagation Through Time (BPTT)

Unrolling of RNN

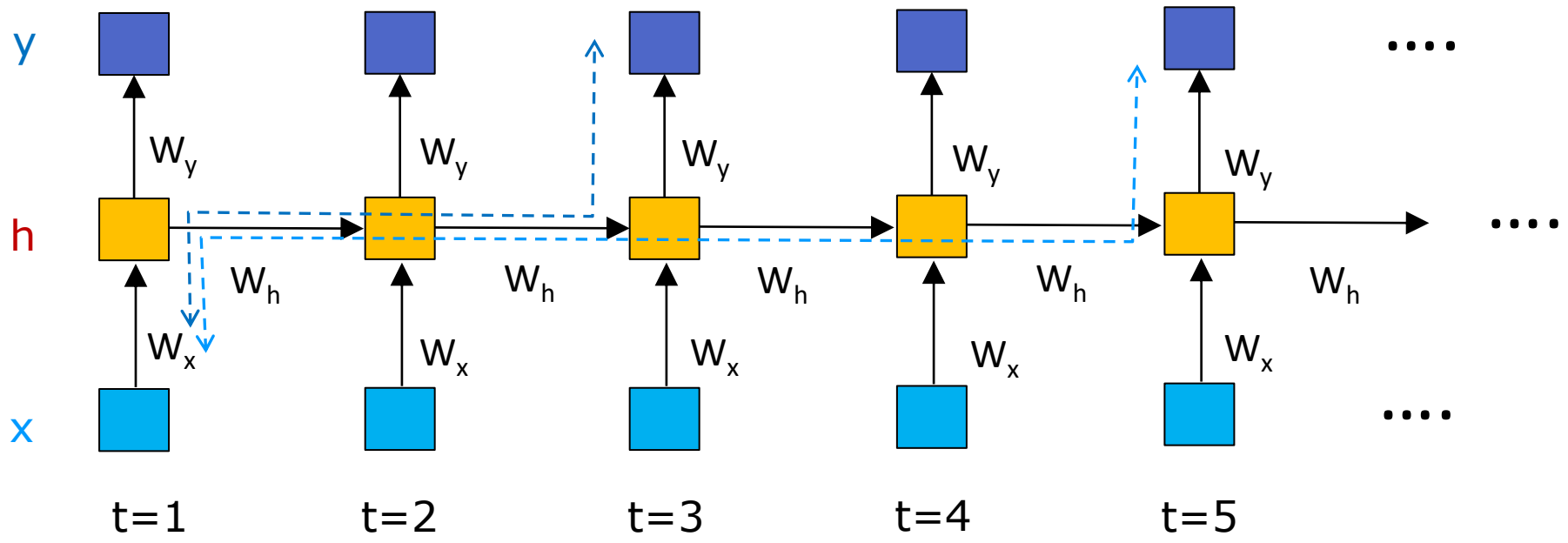


Back-Propagation through Time

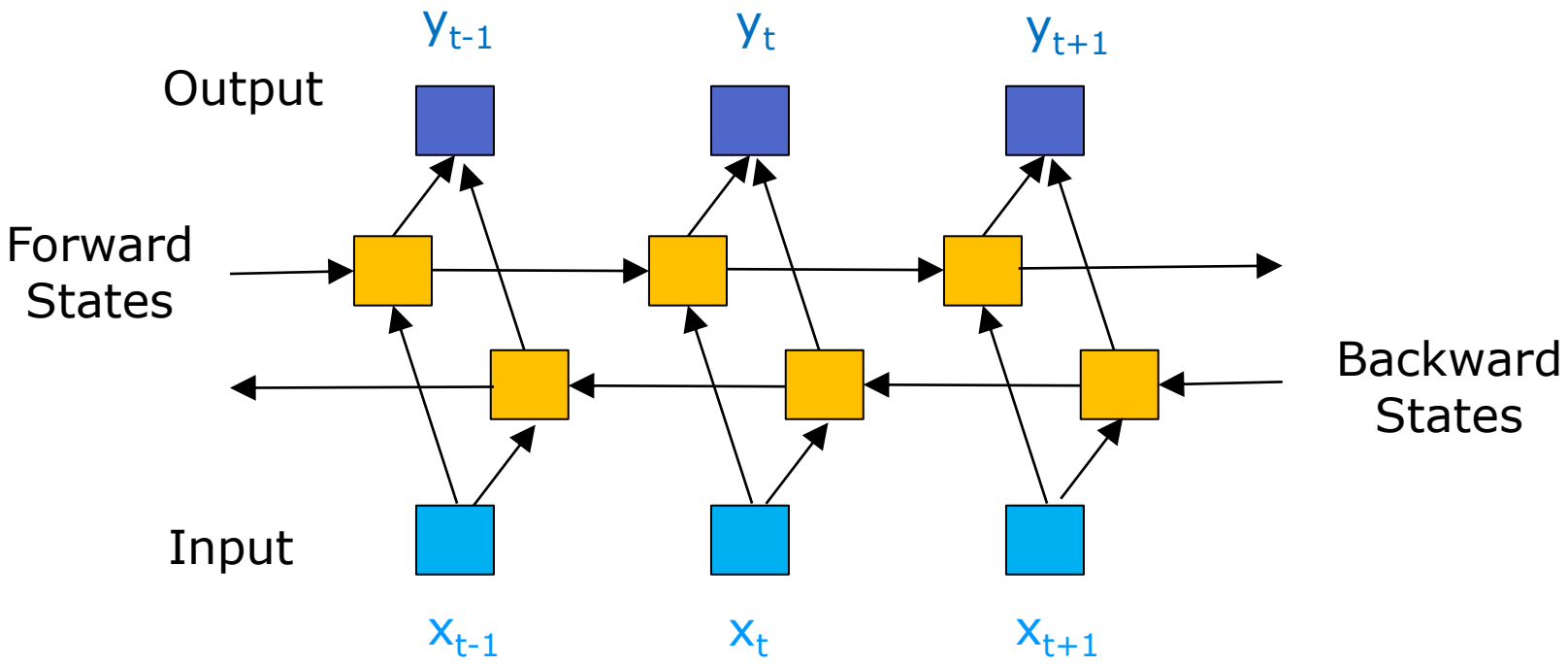


$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

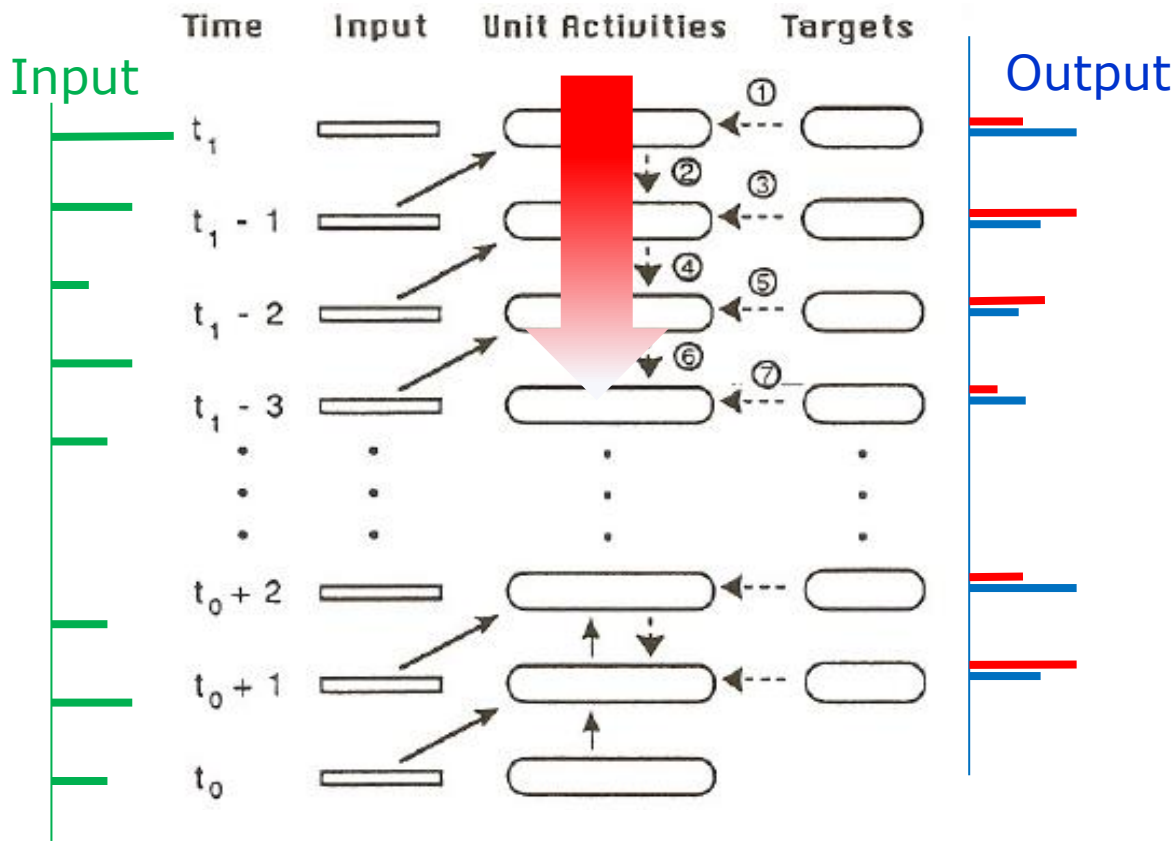
Gradient Explosion/Vanishing



Bidirectional RNN

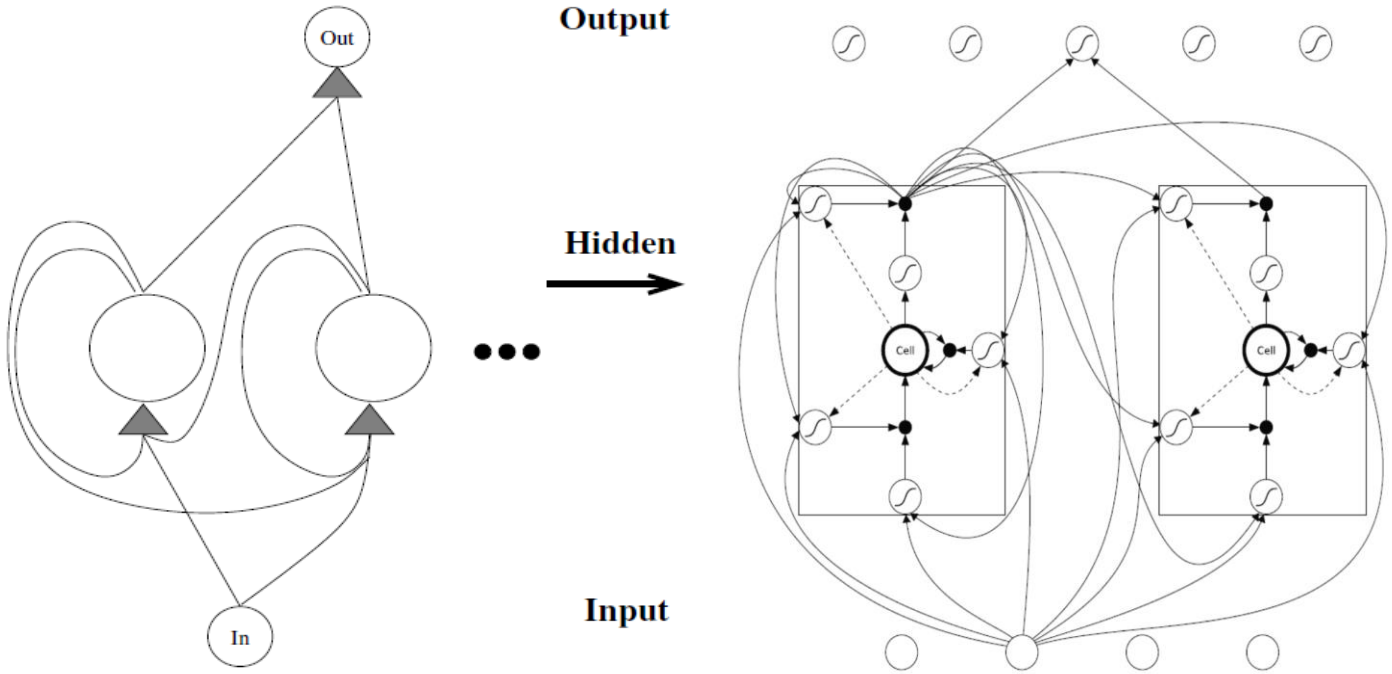


Exploding/Vanishing Gradients



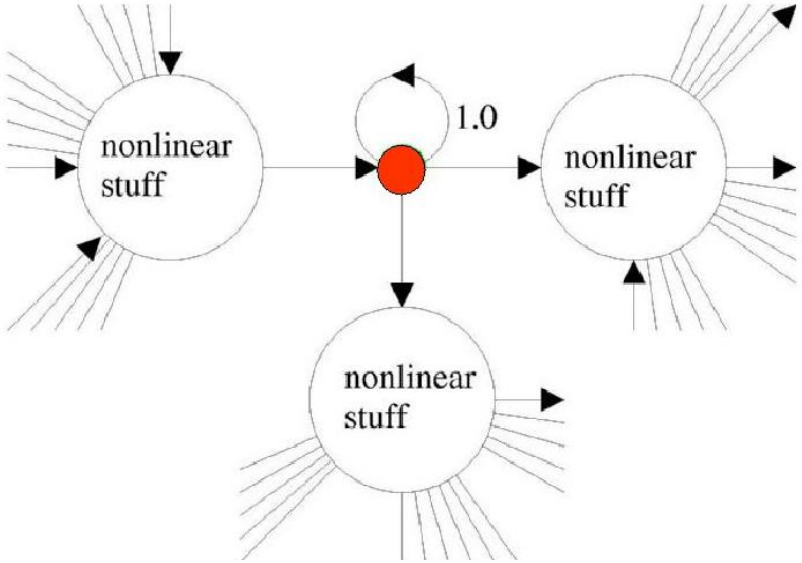
Long Short-Term Memory

Proposed in 1997 by Sepp Hochreiter & Jurgen Schmidhuber



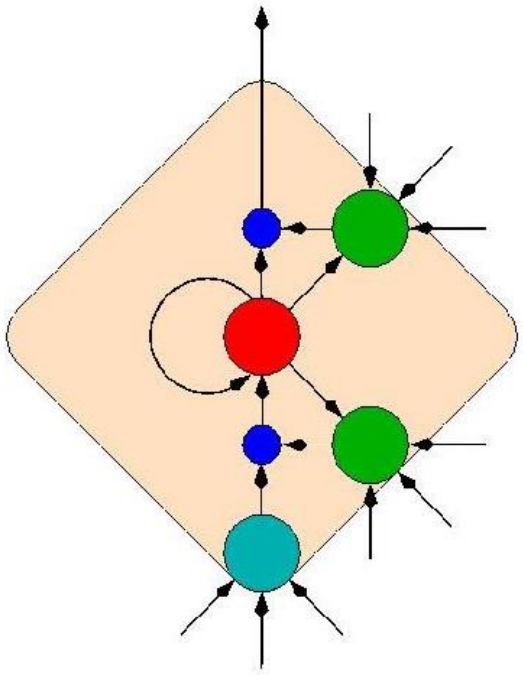
A natural choice: $f_j(x)=x$, $w_{jj} = 1$.

$\Rightarrow$ Just deliver errors, leave learning to other weights!



Ref: http://people.idsia.ch/~juergen/lstm/sld014.htm

One Possible LSTM Cell

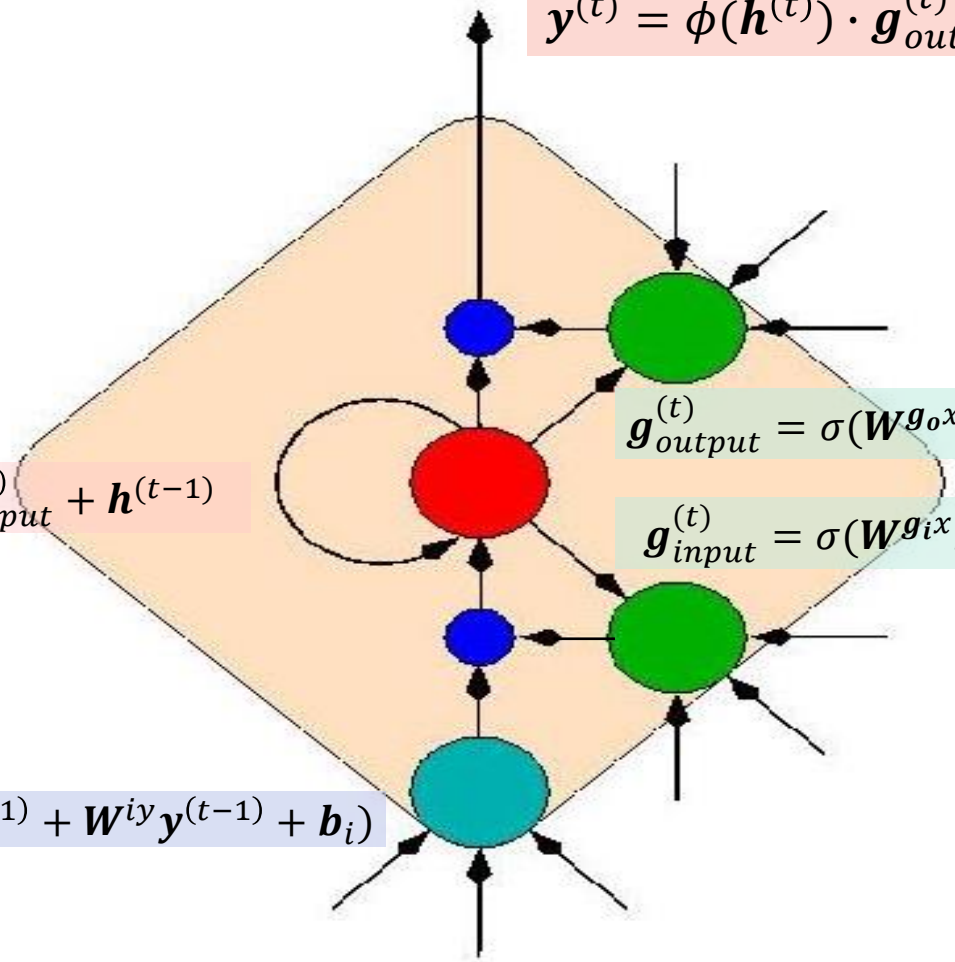


Linear unit (error carousel)

Sigmoid Gate
Open/protect access to error flow

Multiplicative openings or
shut-downs

$$\boldsymbol{y}^{(t)} = \phi(\boldsymbol{h}^{(t)}) \cdot \boldsymbol{g}_{output}^{(t)}$$



$$\boldsymbol{g}_{output}^{(t)} = \sigma(\boldsymbol{W}^{g_o x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_o h}\boldsymbol{h}^{(t)} + \boldsymbol{W}^{g_o y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_o})$$

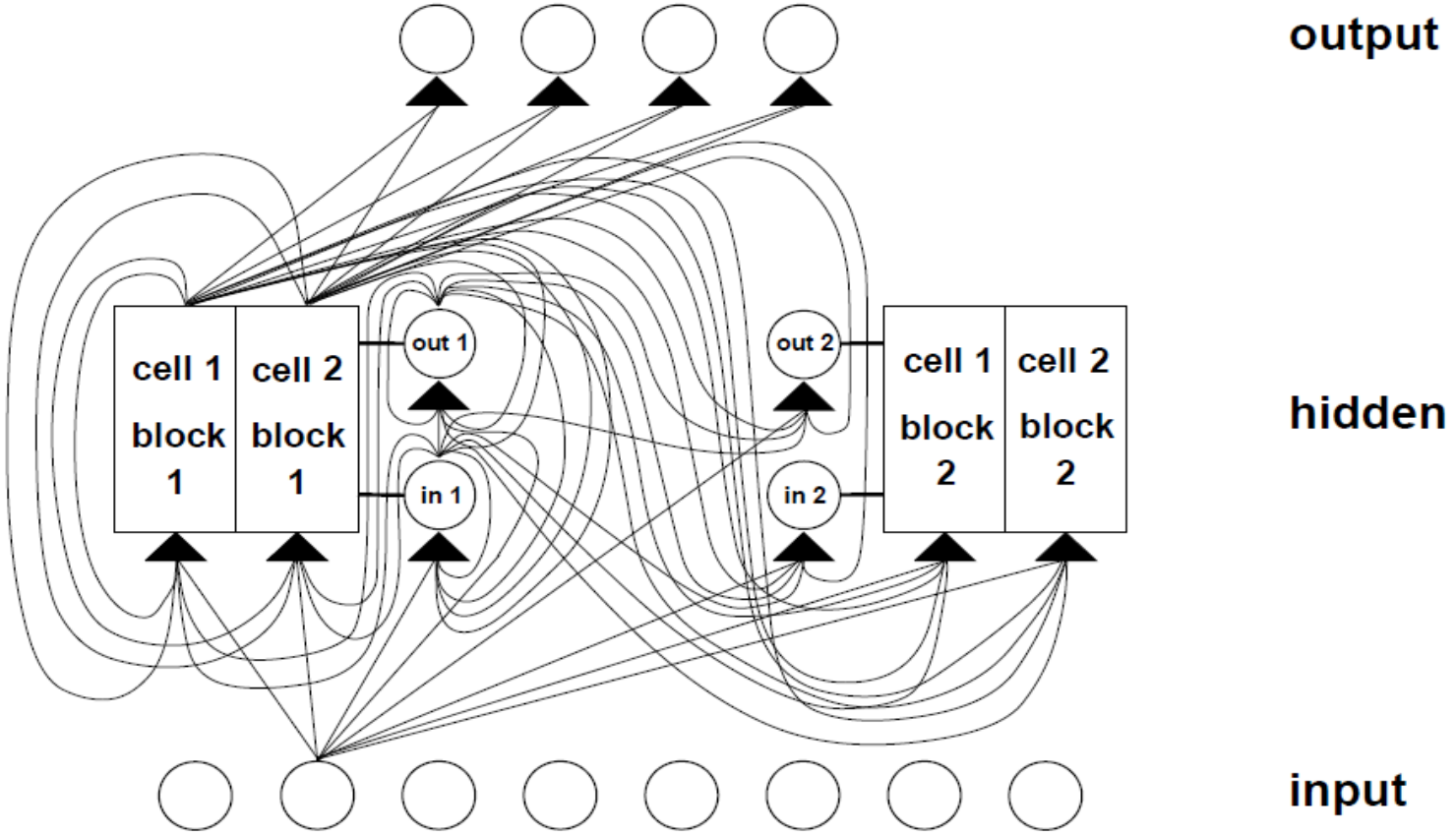
$$\boldsymbol{h}^{(t)} = \boldsymbol{i}^{(t)} \cdot \boldsymbol{g}_{input}^{(t)} + \boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{g}_{input}^{(t)} = \sigma(\boldsymbol{W}^{g_i x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_i h}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{g_i y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_i})$$

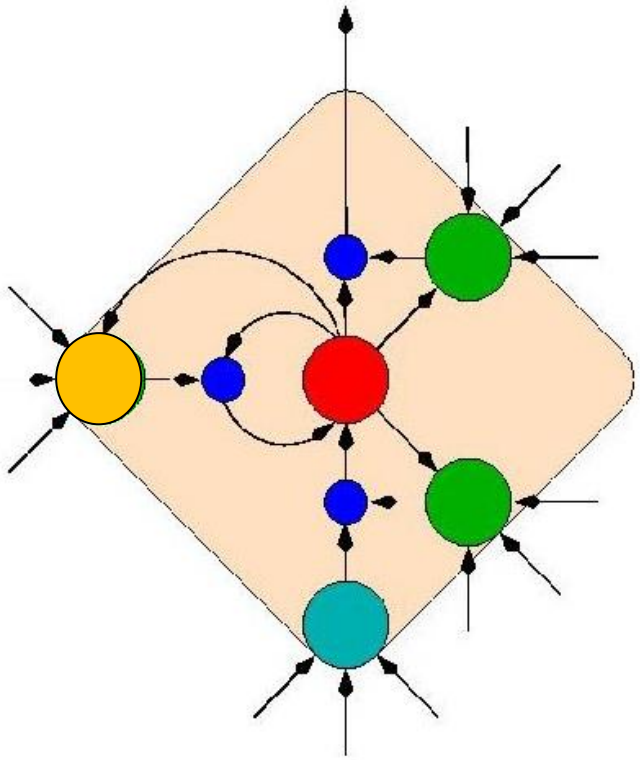
$$\boldsymbol{i}^{(t)} = \phi(\boldsymbol{W}^{ix}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{ih}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{iy}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_i)$$

Nothing can enter the cell unless the input gate is sufficiently active.

Example of LSTM RNN



Standard LSTM Cell



Linear unit (error carousel)

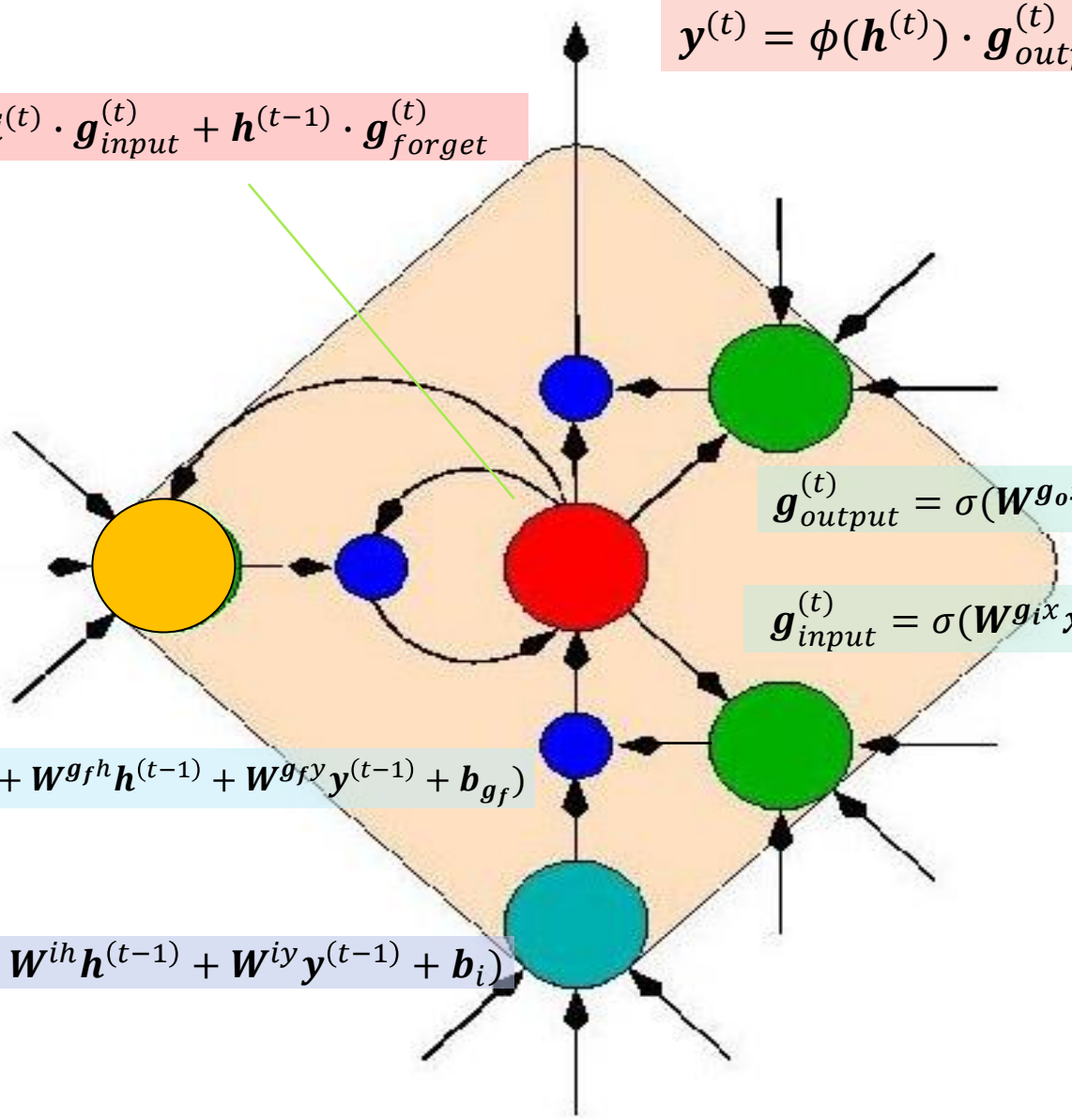
Sigmoid Gate
Open/protect access to error flow

Forget Gate

Multiplicative openings or
shut-downs

$$\boldsymbol{y}^{(t)} = \phi(\boldsymbol{h}^{(t)}) \cdot \boldsymbol{g}_{output}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{i}^{(t)} \cdot \boldsymbol{g}_{input}^{(t)} + \boldsymbol{h}^{(t-1)} \cdot \boldsymbol{g}_{forget}^{(t)}$$



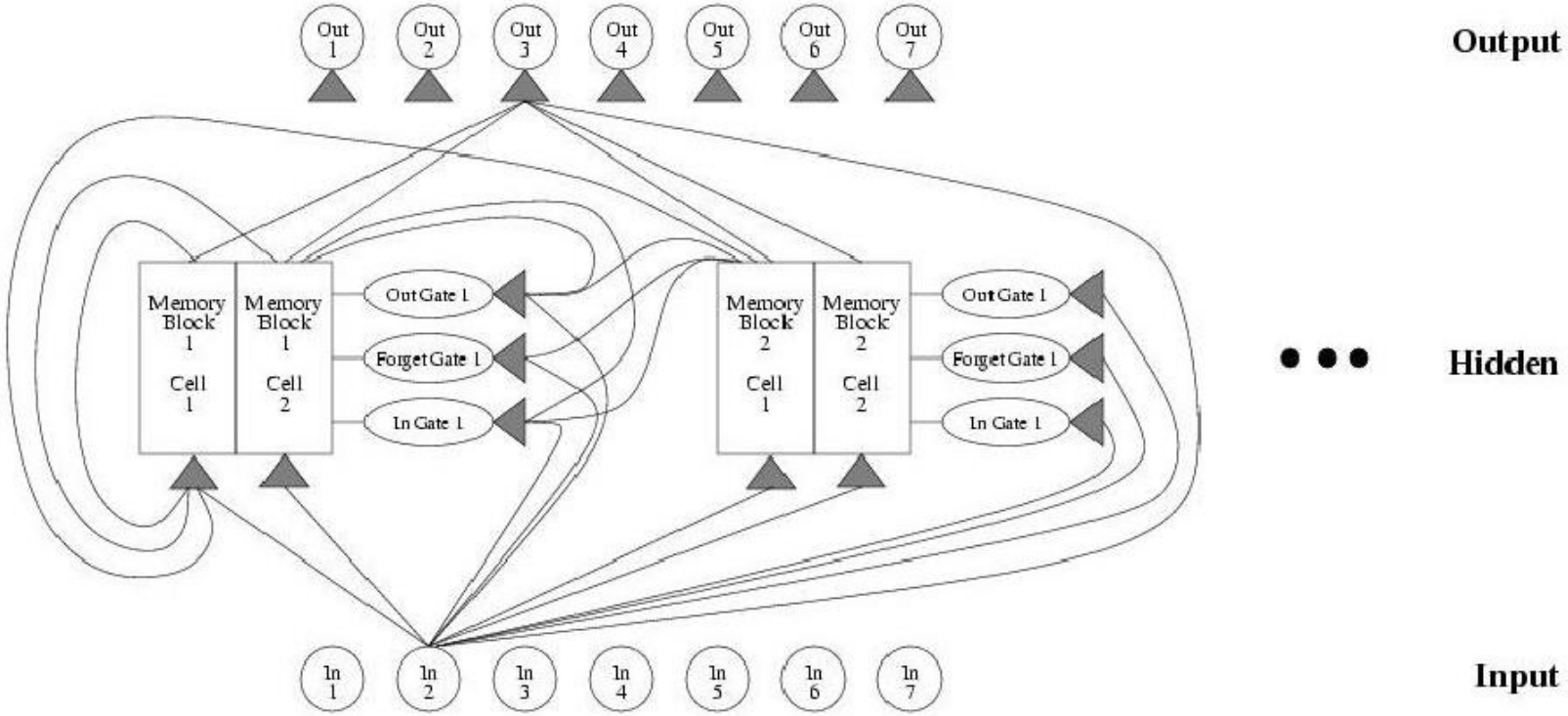
$$\boldsymbol{g}_{output}^{(t)} = \sigma(\boldsymbol{W}^{g_o x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_o h}\boldsymbol{h}^{(t)} + \boldsymbol{W}^{g_o y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_o})$$

$$\boldsymbol{g}_{input}^{(t)} = \sigma(\boldsymbol{W}^{g_i x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_i h}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{g_i y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_i})$$

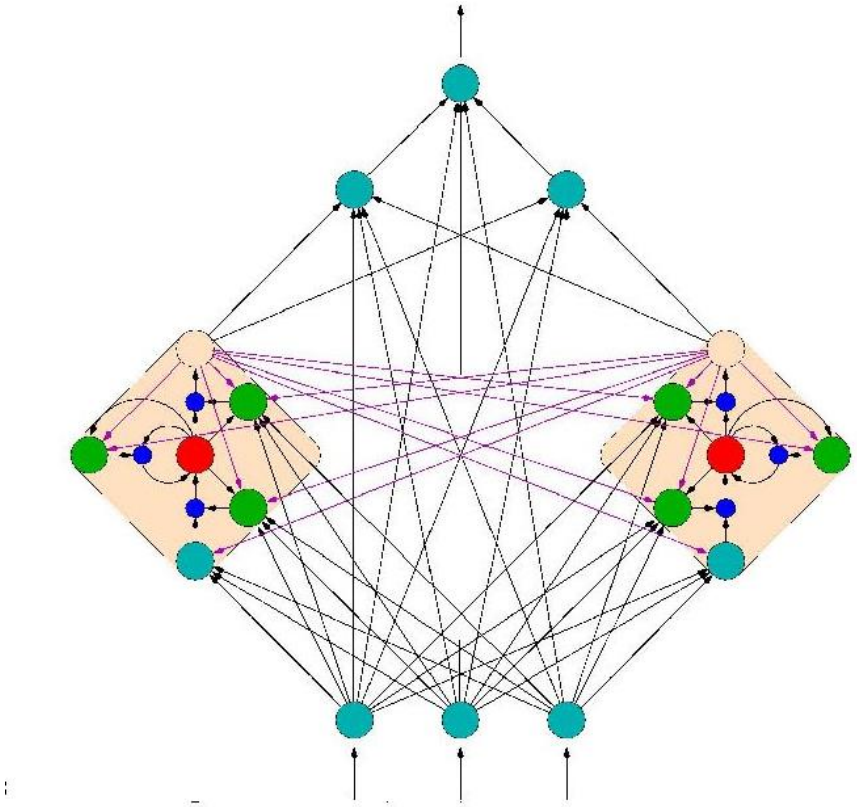
$$\boldsymbol{g}_{forget}^{(t)} = \sigma(\boldsymbol{W}^{g_f x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_f h}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{g_f y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_f})$$

$$\boldsymbol{i}^{(t)} = \phi(\boldsymbol{W}^{ix}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{ih}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{iy}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_i)$$

Example of LSTM RNN



Mix LSTM cells and others



Gated Recurrent Unit (GRU)

Proposed by K. Cho et al. in 2014.

- The activation $\boldsymbol{y}^{(t)}$ is a linear interpolation between the previous activation $\boldsymbol{y}^{(t-1)}$ and the candidate activation $\tilde{\boldsymbol{y}}^{(t-1)}$.

$$\boldsymbol{y}^{(t)} = \left(1 - \boldsymbol{z}^{(t)}\right) \cdot \boldsymbol{y}^{(t-1)} + \boldsymbol{z}^{(t)} \cdot \tilde{\boldsymbol{y}}^{(t)}$$

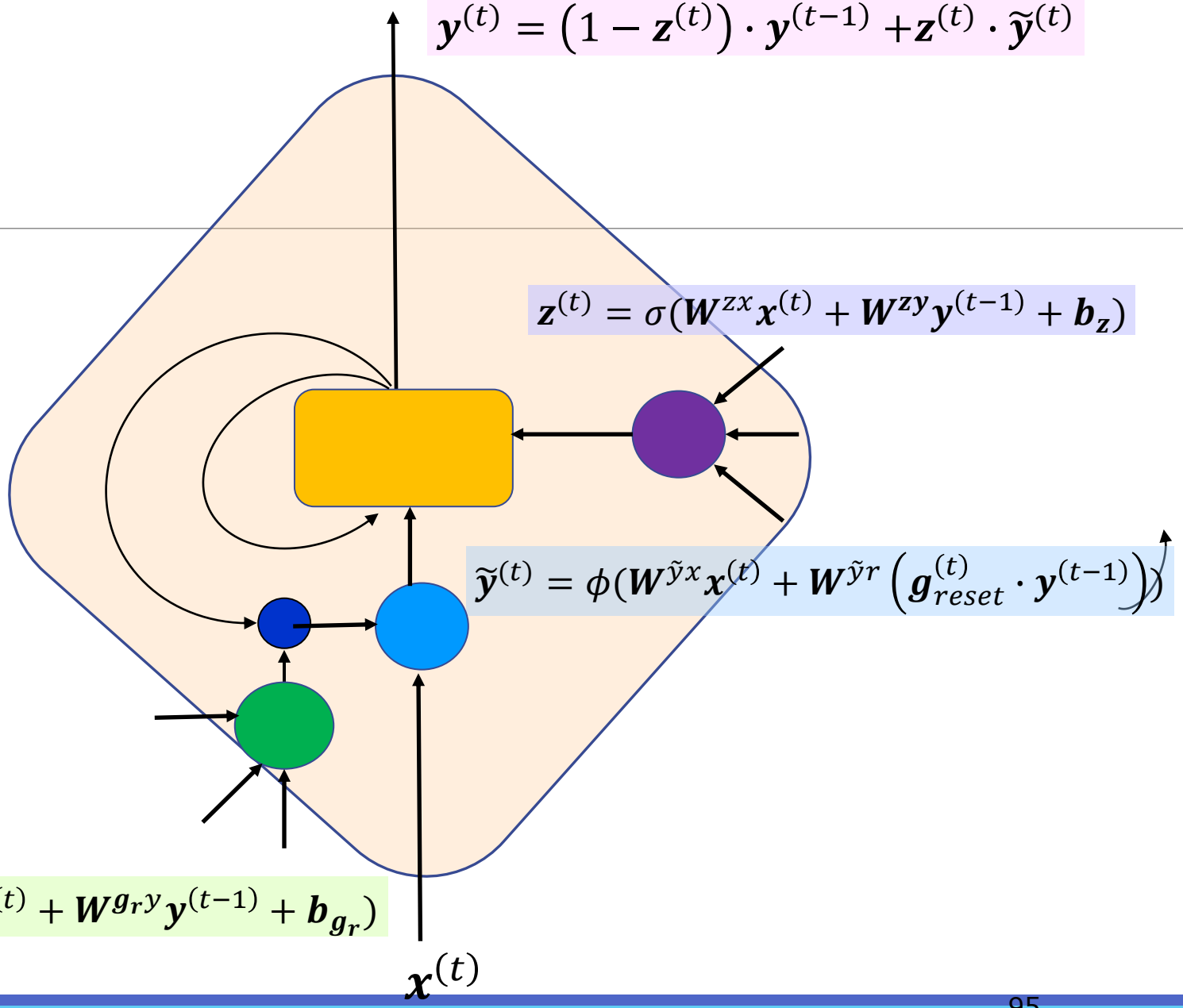
where $\quad \boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}^{zx}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{zy}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_z)$

- Typically, we forget something old when we want to input something new.
  ⇒ couple forget gate and input gate together

$$\tilde{\boldsymbol{y}}^{(t)} = \phi(\boldsymbol{W}^{\tilde{y}x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{\tilde{y}r}\left(\boldsymbol{g}_{reset}^{(t)} \cdot \boldsymbol{y}^{(t-1)}\right))$$

where $\quad \boldsymbol{g}_{reset}^{(t)} = \sigma(\boldsymbol{W}^{g_r x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g_r y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{g_r})$

$$\boldsymbol{y}^{(t)} = \left(1 - \boldsymbol{z}^{(t)}\right) \cdot \boldsymbol{y}^{(t-1)} + \boldsymbol{z}^{(t)} \cdot \tilde{\boldsymbol{y}}^{(t)}$$



$$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}^{zx}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{zy}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{\boldsymbol{z}})$$

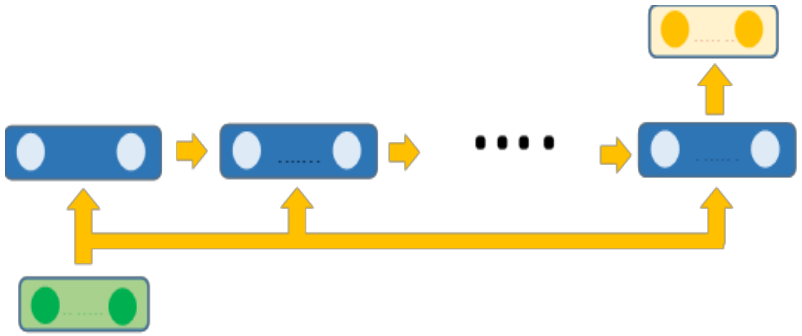
$$\tilde{\boldsymbol{y}}^{(t)} = \phi(\boldsymbol{W}^{\tilde{y}x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{\tilde{y}r}\left(\boldsymbol{g}_{reset}^{(t)} \cdot \boldsymbol{y}^{(t-1)}\right))$$

$$\boldsymbol{g}_{reset}^{(t)} = \sigma(\boldsymbol{W}^{\boldsymbol{g}_r x}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{\boldsymbol{g}_r y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_{\boldsymbol{g}_r})$$

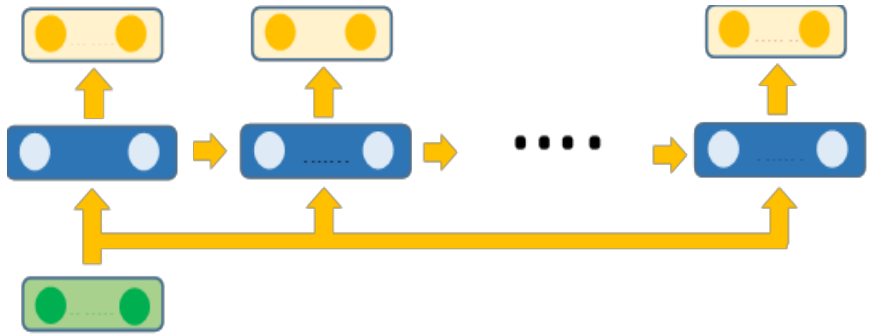
$\boldsymbol{x}^{(t)}$

Applications

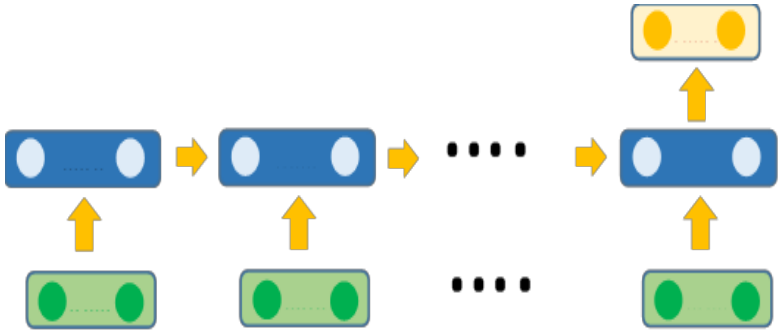
One-to-One: Classification/Identification



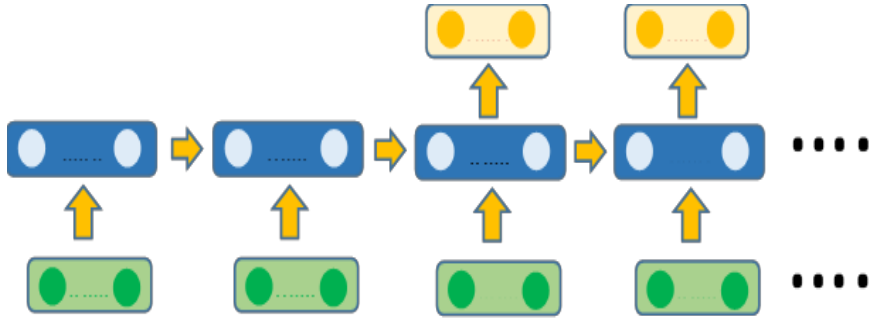
One-to-Many: Image Captioning



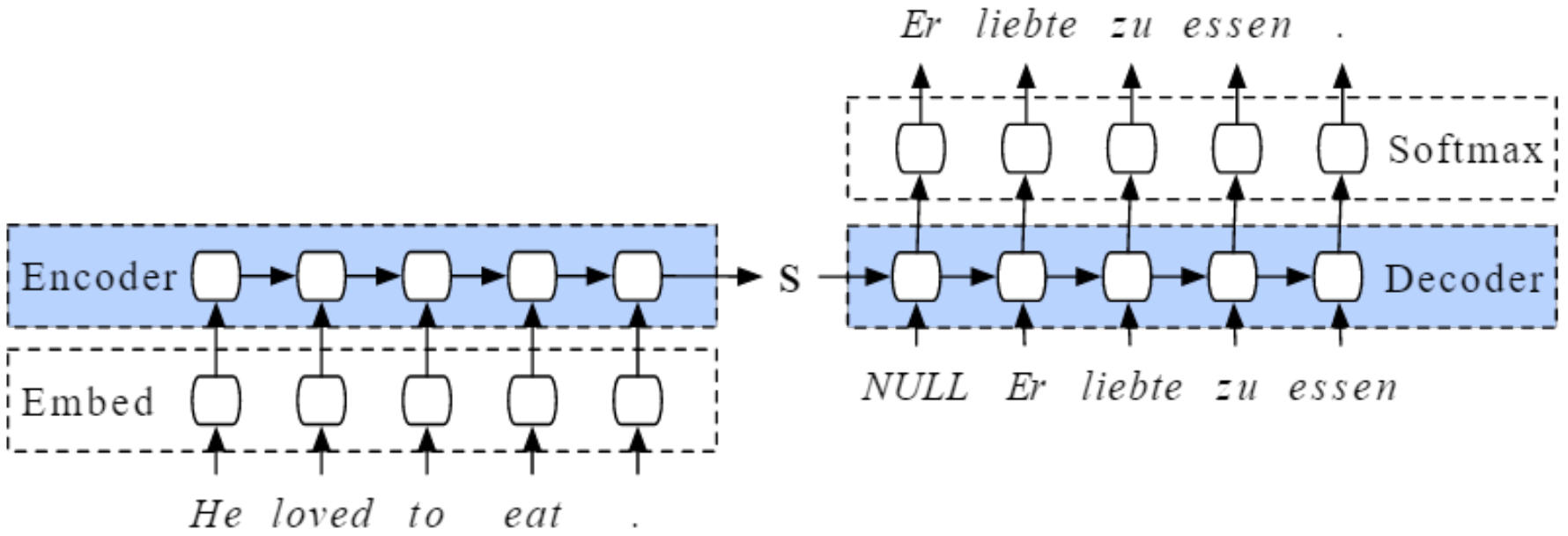
Many-to-One: Video Classification



Many-to-Many: Translation

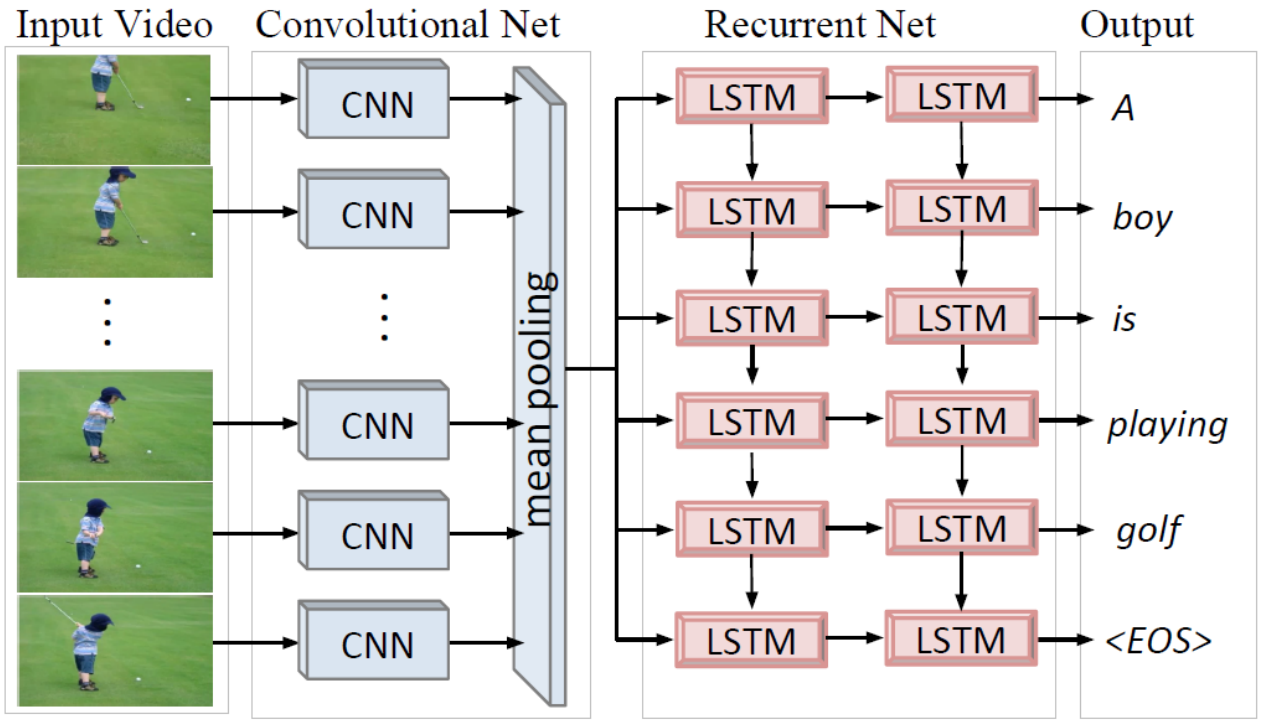


RNN Encoder-Decoder Structure



Ref: https://smerity.com/articles/2016/google_nmt_arch.html

Example: Video Description

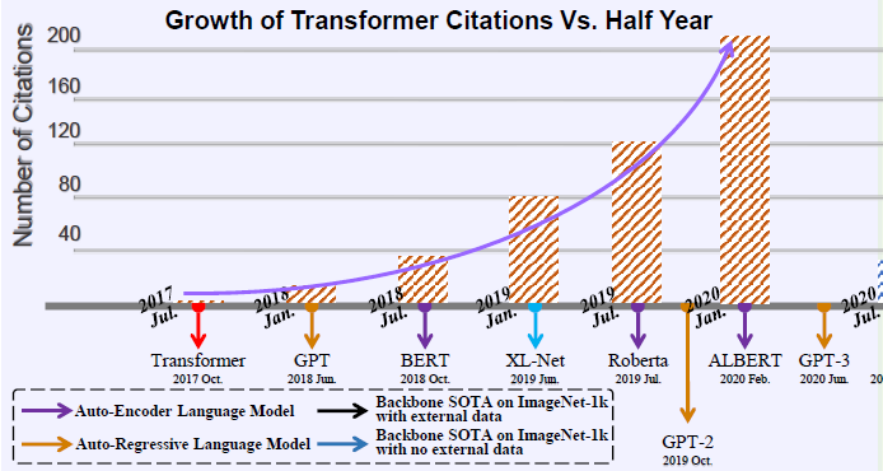


Ref: Venugopalan, Subhashini, et al. "Translating videos to natural language using deep recurrent neural networks." *arXiv preprint arXiv:1412.4729* (2014).

Transformers

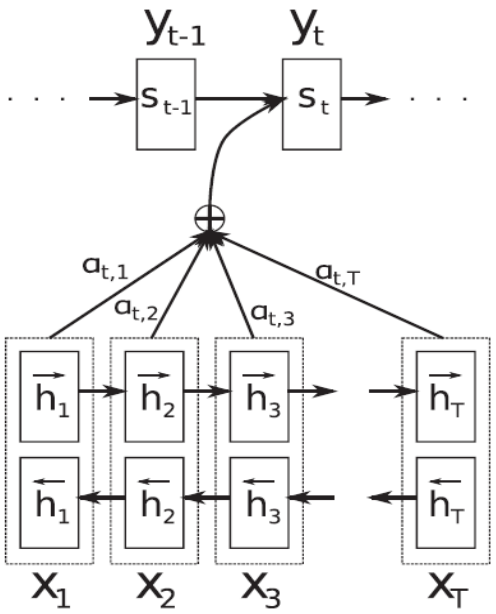
Transformer

- Based on the attention mechanism
- Enable modeling long dependencies between input sequence elements and support parallel processing of sequence
- Has replaced RNN as the mainstream technology in NLP



BERT: Bidirectional Encoder Representations from Transformers
GPT: Generative Pre-trained Transformer
RoBERTa: Robustly Optimized BERT Pre-training
ALBERT: A Lite BERT

Attention Mechanism



$$(\mathbf{h_1}, \dots, \mathbf{h_T}) = \text{BiRNN}(x_1, \dots, x_T) \qquad \text{where } \mathbf{h_i} = \begin{bmatrix} \vec{\mathbf{h}}_\mathbf{i} \\ \overleftarrow{\mathbf{h}}_\mathbf{i} \end{bmatrix}$$

$$e_{tj} = a(\mathbf{s_{t-1}}, \mathbf{h_j})$$

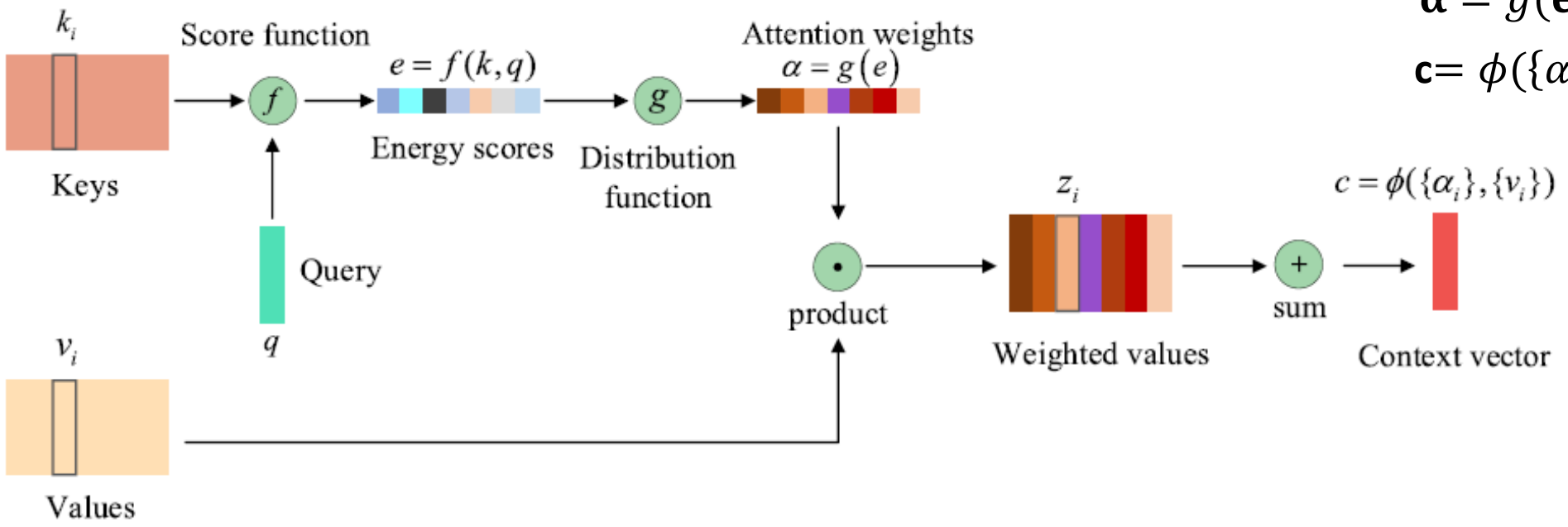
$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}$$

$$\mathbf{c_t} = \sum_{j=1}^{T} \alpha_{tj} \mathbf{h_j}$$ Context vector at t

Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning." Neurocomputing 452 (2021): 48-62.

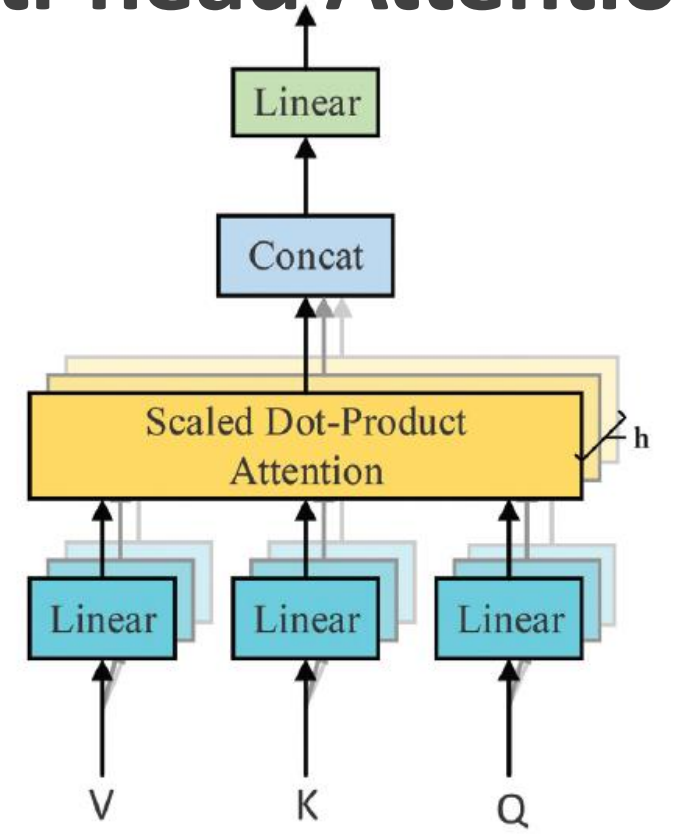
Unified Attention Model

$$\mathbf{e} = f(\boldsymbol{q}, \boldsymbol{K})$$
$$\boldsymbol{\alpha} = g(\mathbf{e})$$
$$\mathbf{c} = \phi(\{\alpha_i\}, \{\boldsymbol{v}_i\})$$



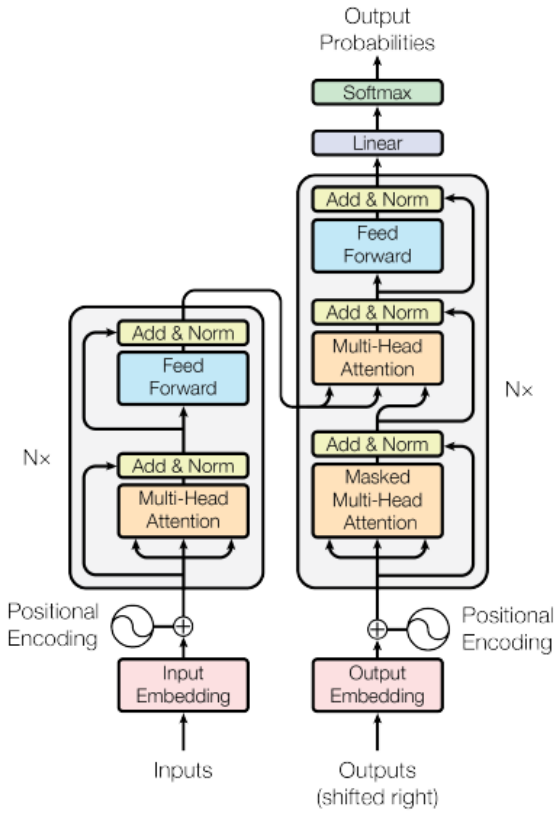
Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning." Neurocomputing 452 (2021): 48-62.

Multi-head Attention



Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning." Neurocomputing 452 (2021): 48-62.

Transformer Architecture



Ref: Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

CNN versus Transformer

(convolution) $y(\boldsymbol{x}, i) = \sum_{x_j \in N_{local}(x_i)} W_{i \to j} x_j$

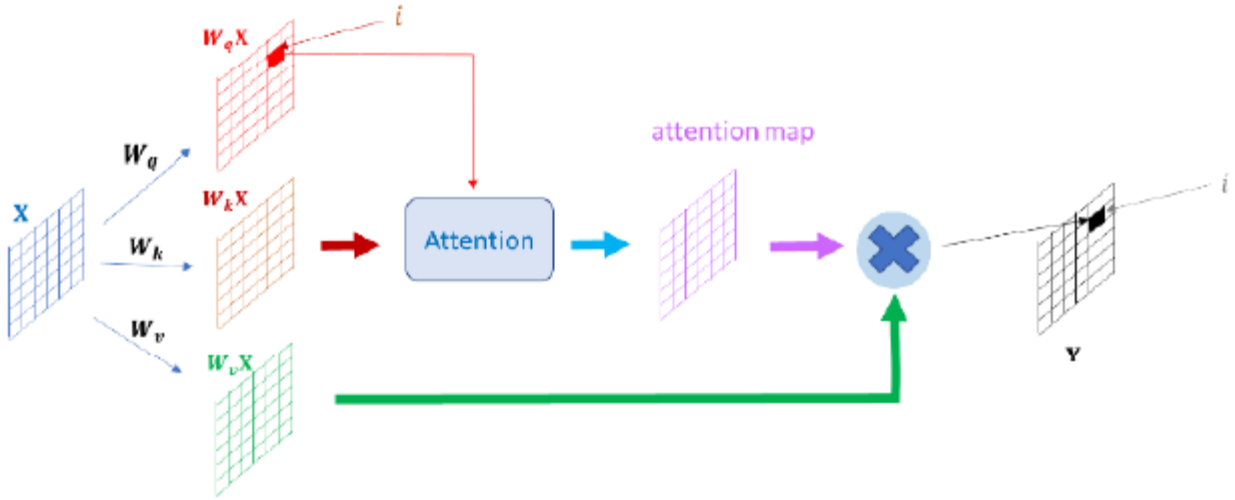
(self-attention) $y(\boldsymbol{x}, i) = \sum_{x_j \in N_{receptive\ field}(\boldsymbol{x})} \alpha_{i \to j}(\boldsymbol{x}) W_v x_j$

where $\alpha_{i \to j}(\boldsymbol{x}) = \frac{e^{(W_q x_i)^T W_k x_j}}{\sum_{z \in N(i)} e^{(W_q x_i)^T W_k x_z}}$

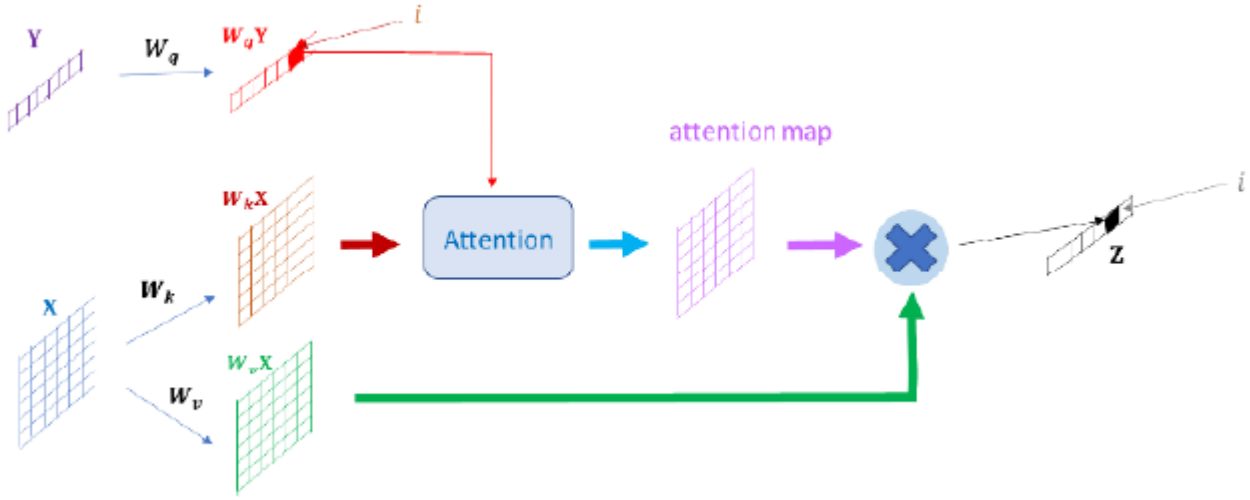
(cross-attention) $z(\boldsymbol{y}, \boldsymbol{x}, i) = \sum_{x_j \in N_{receptive\ field}(\boldsymbol{x})} \beta_{i \to j}(\boldsymbol{y}, \boldsymbol{x}) W_v x_j$

where $\beta_{i \to j}(\boldsymbol{y}, \boldsymbol{x}) = \frac{e^{(W_q y_i)^T W_k x_j}}{\sum_{z \in N(i)} e^{(W_q y_i)^T W_k x_z}}$

Self-attention



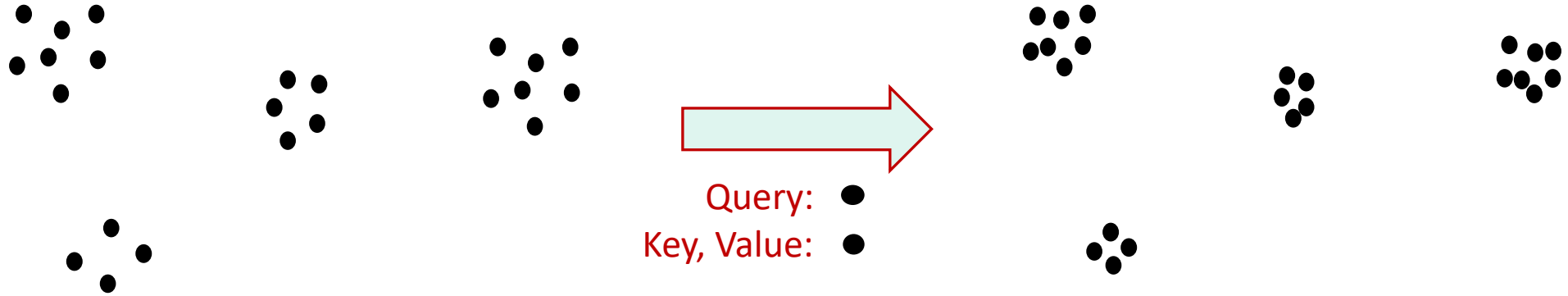
Cross-attention



Self-attention

$$\mathbf{W_v} = \mathbf{W_q} = \mathbf{W_k} = \mathbf{I}$$

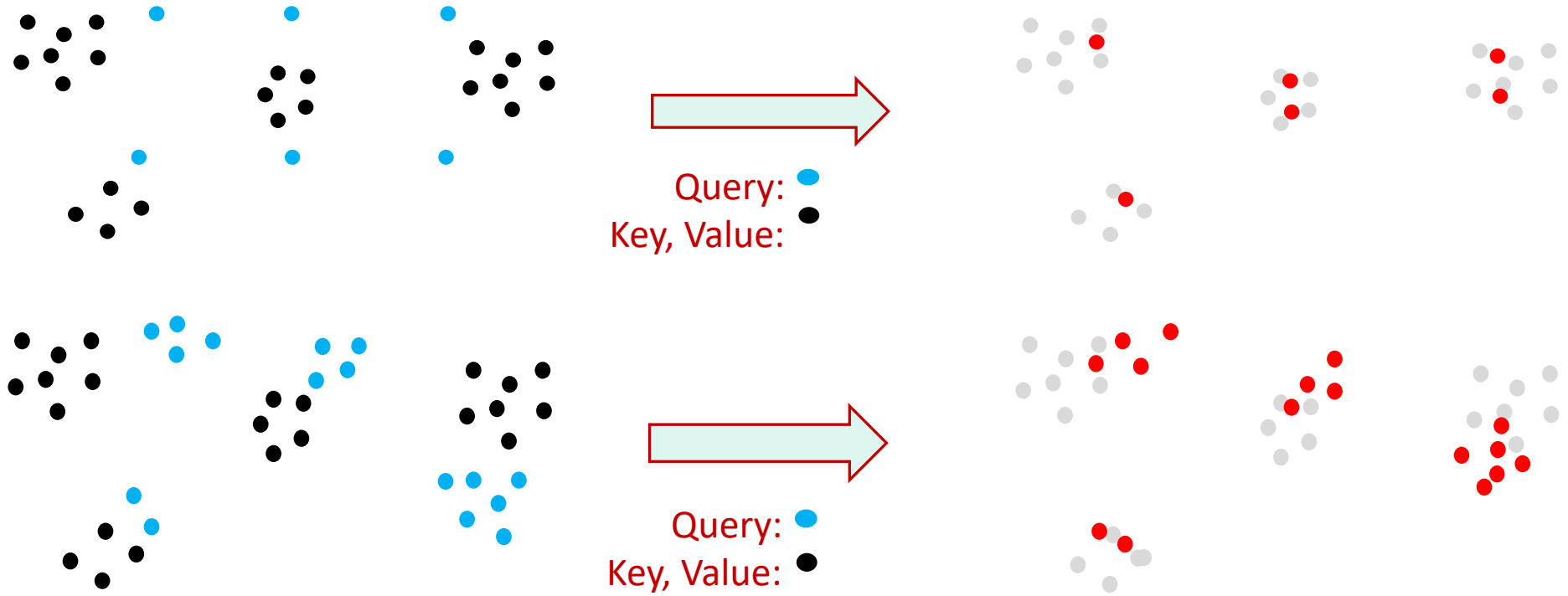
$$y(\boldsymbol{x}, i) = \sum_{x_j \in N_{receptive\ field}(\boldsymbol{x})} \alpha_{i \to j}(\boldsymbol{x}) x_j \quad \text{where} \quad \alpha_{i \to j}(\boldsymbol{x}) = \frac{e^{x_i{}^T x_j}}{\sum_{z \in N(i)} e^{x_i{}^T x_z}}$$



Cross-attention

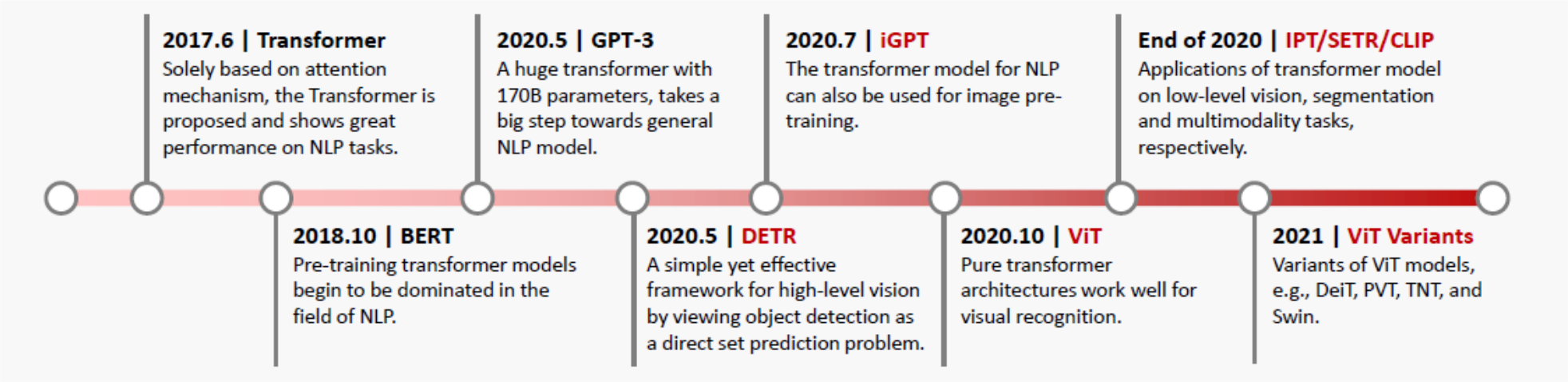
$$\mathbf{W_v} = \mathbf{W_q} = \mathbf{W_k} = \mathbf{I}$$

$$z(\boldsymbol{y}, \boldsymbol{x}, i) = \sum_{x_j \in N_{receptive\ field}(\boldsymbol{x})} \beta_{i \to j}(\boldsymbol{y}, \boldsymbol{x}) x_j \quad \text{where} \quad \beta_{i \to j}(\boldsymbol{y}, \boldsymbol{x}) = \frac{e^{y_i{}^T x_j}}{\sum_{z \in N(i)} e^{y_i{}^T x_z}}$$



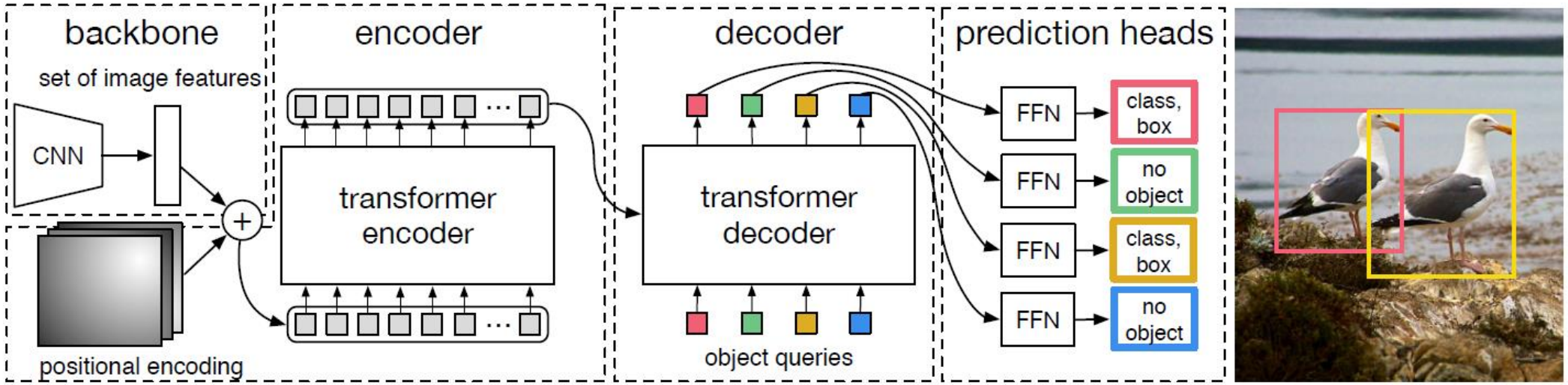
Transformer

- The **Transformer** concept has attracted great interest in the computer vision community.



Ref: Han, Kai, et al. "A survey on vision transformer." IEEE transactions on pattern analysis and machine intelligence (2022).

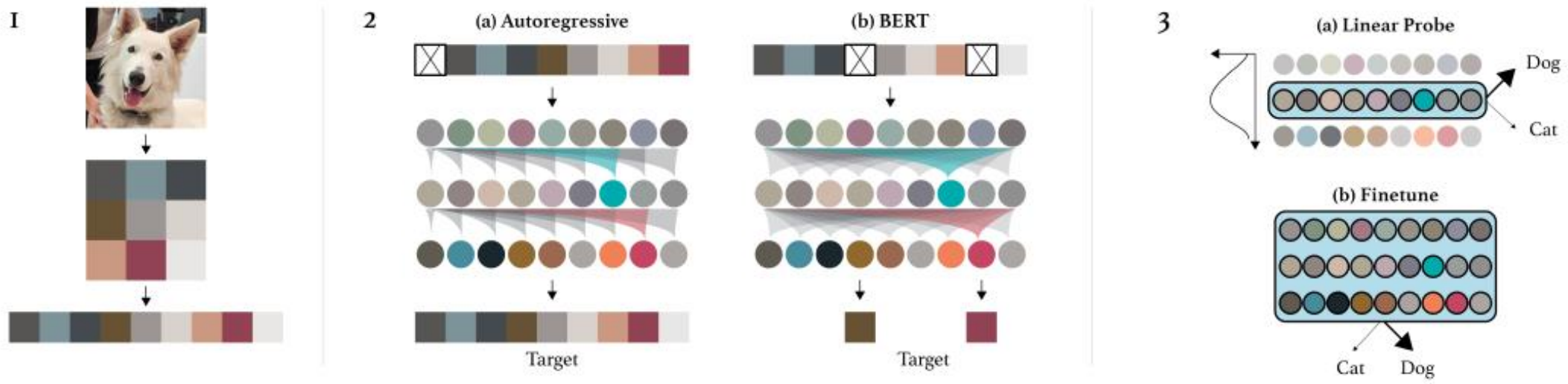
DETR



Ref: Carion, Nicolas, et al. "End-to-end object detection with transformers." European conference on computer vision. Springer, Cham, 2020.

iGPT: Generative Pretraining from Pixels

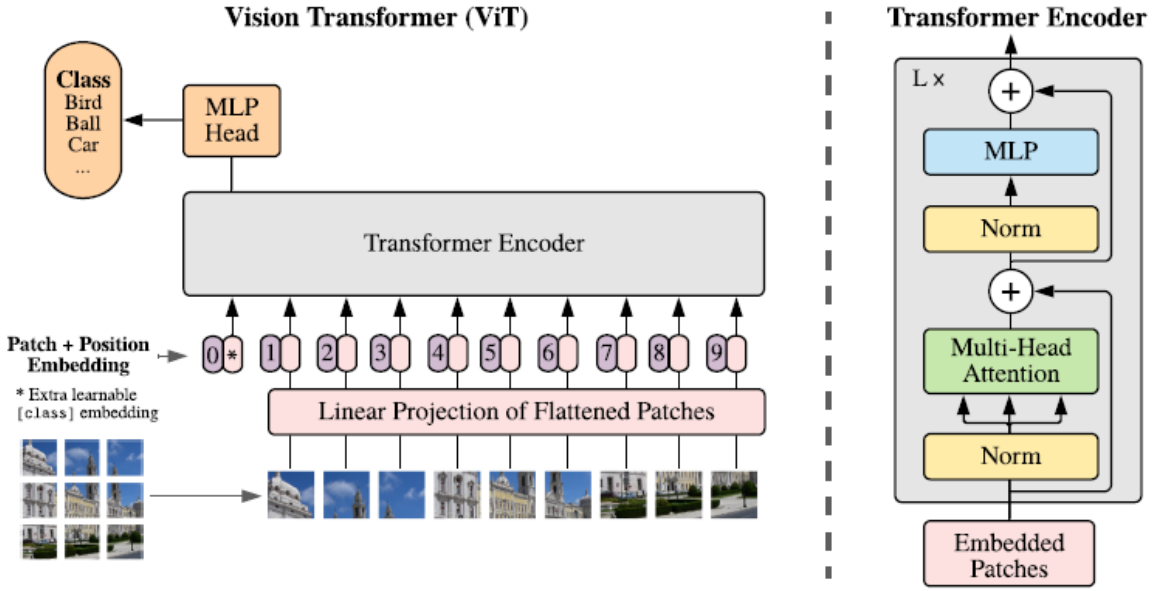
- Resize the image to a low-resolution one
- Reshape the low-resolution image into a 1-D sequence.



Ref: Chen, Mark, et al. "Generative pretraining from pixels." International conference on machine learning. PMLR, 2020.

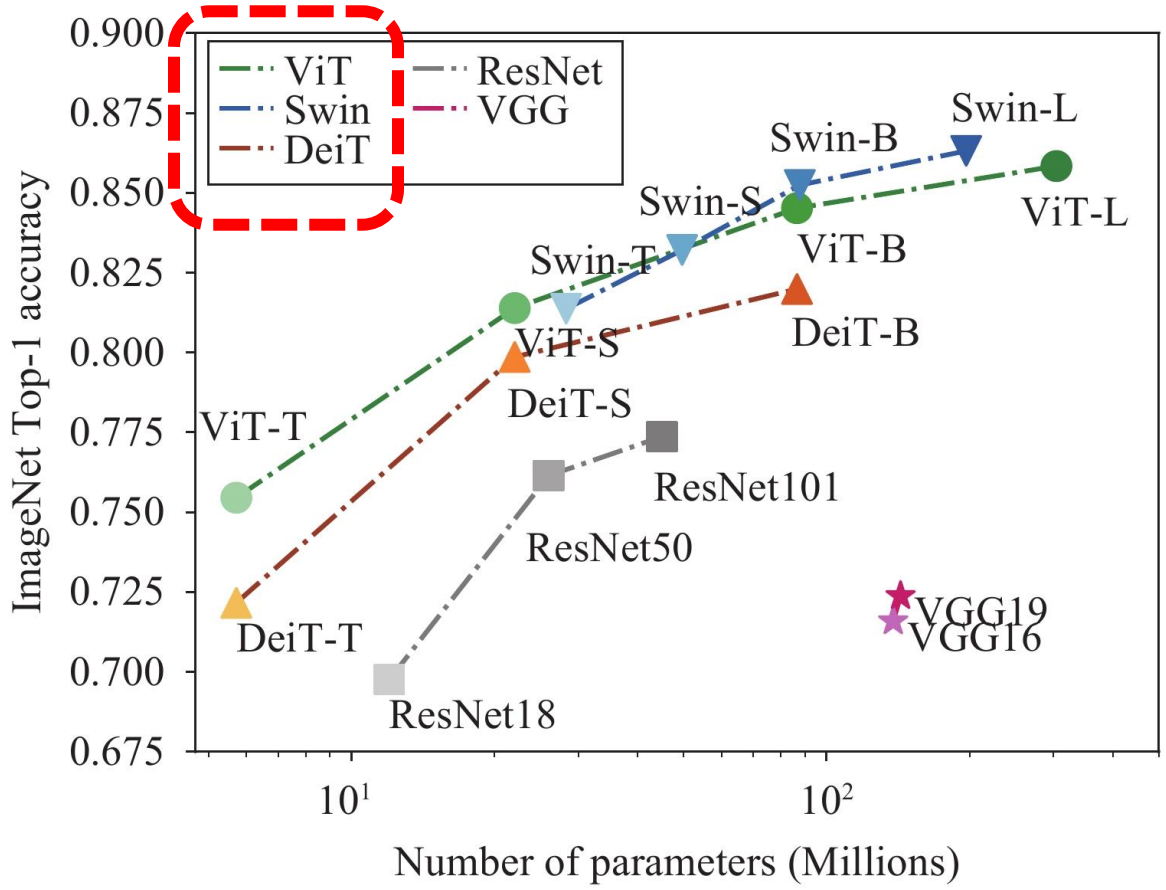
Vision Transformer (ViT)

- Apply transformer directly to sequences of image patches.
- Perform very well if trained on large datasets (14M – 300M images)

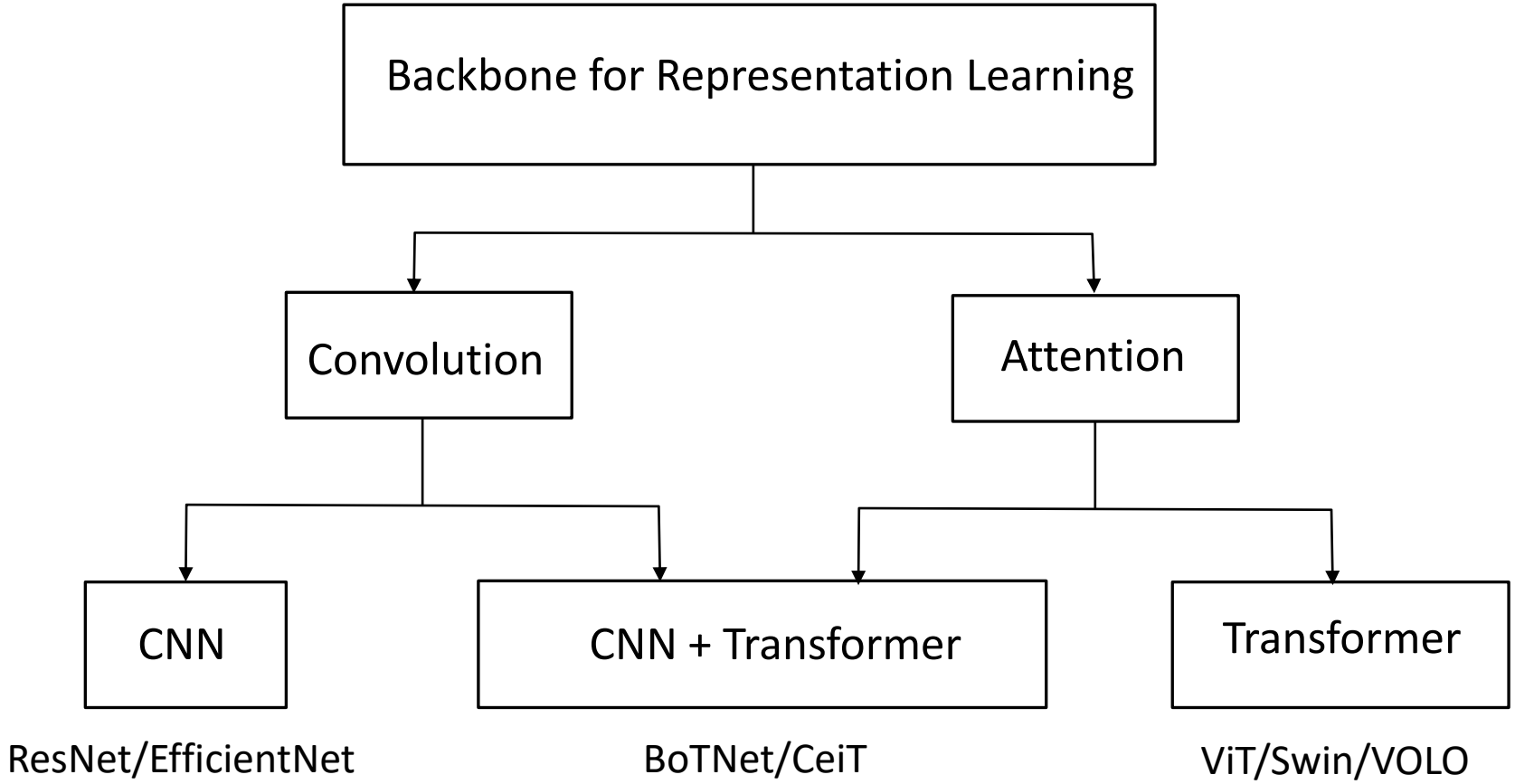


Ref: Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

Improvement of DL Performance



Classification of DL Models



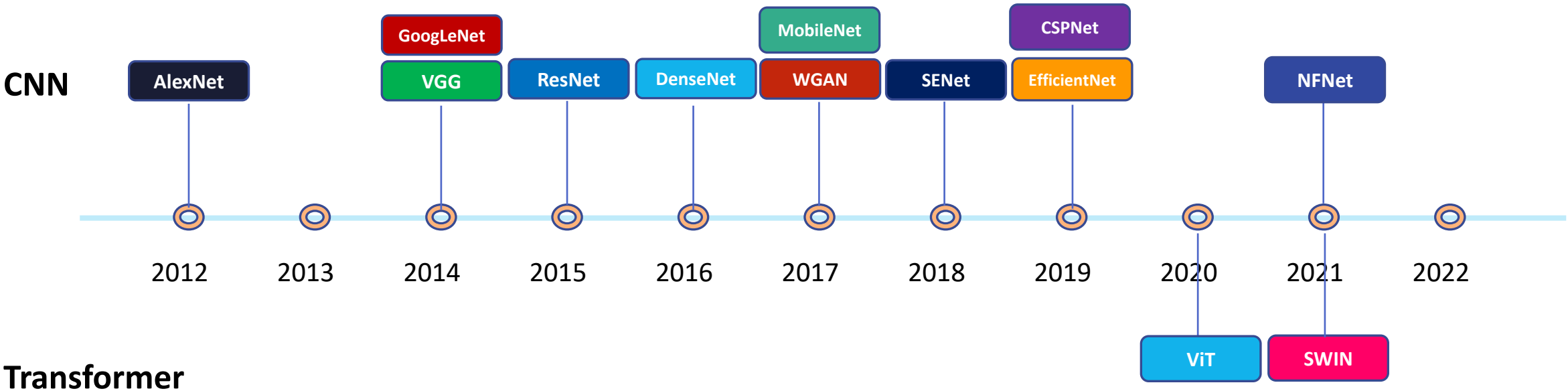
BoTNet (Bottleneck Transformer)

- Replace only the final three bottleneck blocks of a ResNet with BoT blocks without any other changes.

stage	output	ResNet-50	**BoTNet-50**
c1	512 × 512	7×7, 64, stride 2	7×7, 64, stride 2
c2	256 × 256	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
c3	128 × 128	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
c4	64 × 64	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
c5	32 × 32	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ \text{MHSA}, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
# params.		**25.5**×10^6	**20.8**×10^6
M.Adds		**85.4**×10^9	**102.98**×10^9
TPU steptime		**786.5** ms	**1032.66** ms

Ref: Srinivas, Aravind, et al. "Bottleneck transformers for visual recognition." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

Timeline of Popular Models

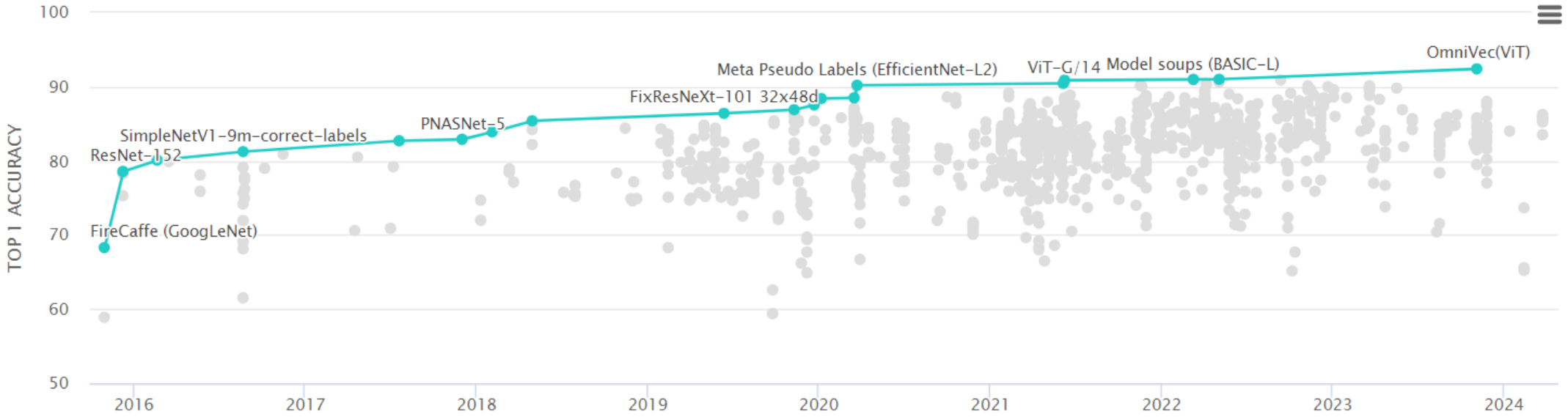


Improvement of DL Performance

Image Classification on ImageNet

1000 classes

Top-1 Accuracy: 63.3% (AlexNet, 2012) → 92.4% (OmniVec, 2023)



https://paperswithcode.com/sota/image-classification-on-imagenet (2024/05/20)

Performance Spectrum of DL Models

Object Detection

Performance comparison of various object detectors on MS COCO and PASCAL VOC 2012 datasets at similar input image size. Rows colored gray are real-time detectors (>30 FPS).

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	~0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	~0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	~0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	~3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	~4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
DeTR	2020	ResNet-101	-	43.50%	63.80%	20
Swin-L	2021	HTC++	-	57.70%	-	-

Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.

Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." *Digital Signal Processing* (2022): 103514.

Performance Spectrum of DL Models

Object Detection

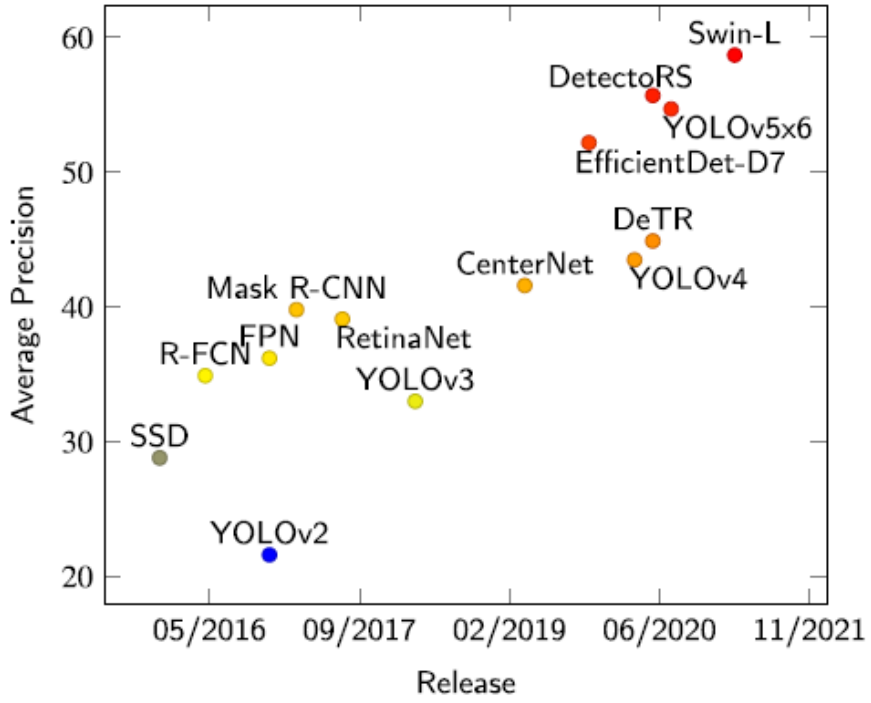


Fig. 10. Performance of object detectors on MS COCO dataset.

Ref: Zaidi, Syed Sahil Abbas, et al. "A survey of modern deep learning based object detection models." *Digital Signal Processing* (2022): 103514.