Deep Neural Networks II

Sen Wang

UDRC Co-Investigator

Associate Professor in Robotics and Autonomous Systems

Institute of Signals, Sensors and Systems

Heriot-Watt University

UDRC-EURASIP Summer School

30th June 2021

Slides adapted from Andrej Karpathy, Kaiming He

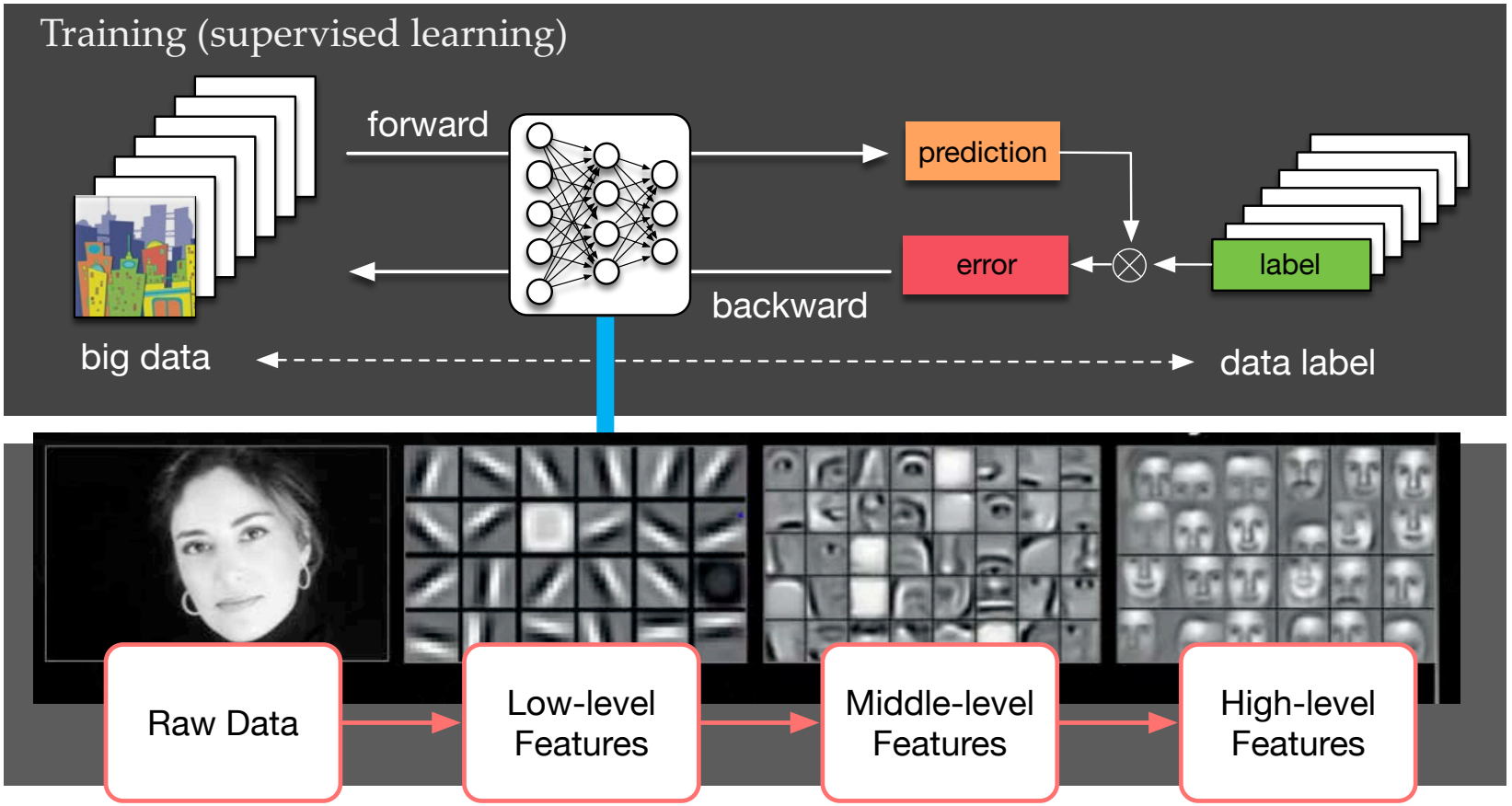
Outline

Learning features for machines to solve problems

- Convolutional Neural Networks (CNNs)
- CNN Architectures - **learning features**
- Some Deep Learning Applications - **problems**
  - Object detection (image, radar, sonar)
  - Semantic segmentation
  - Object detection and tracking using radar and sonar
  - Visual odometry
  - 3D reconstruction
  - Semantic mapping
  - Robot navigation
  - Manipulation and grasping
  - ……

Deep Learning

Deep Learning: a learning technique combining layers of neural networks to **automatically identify features** that are relevant to the problem to solve

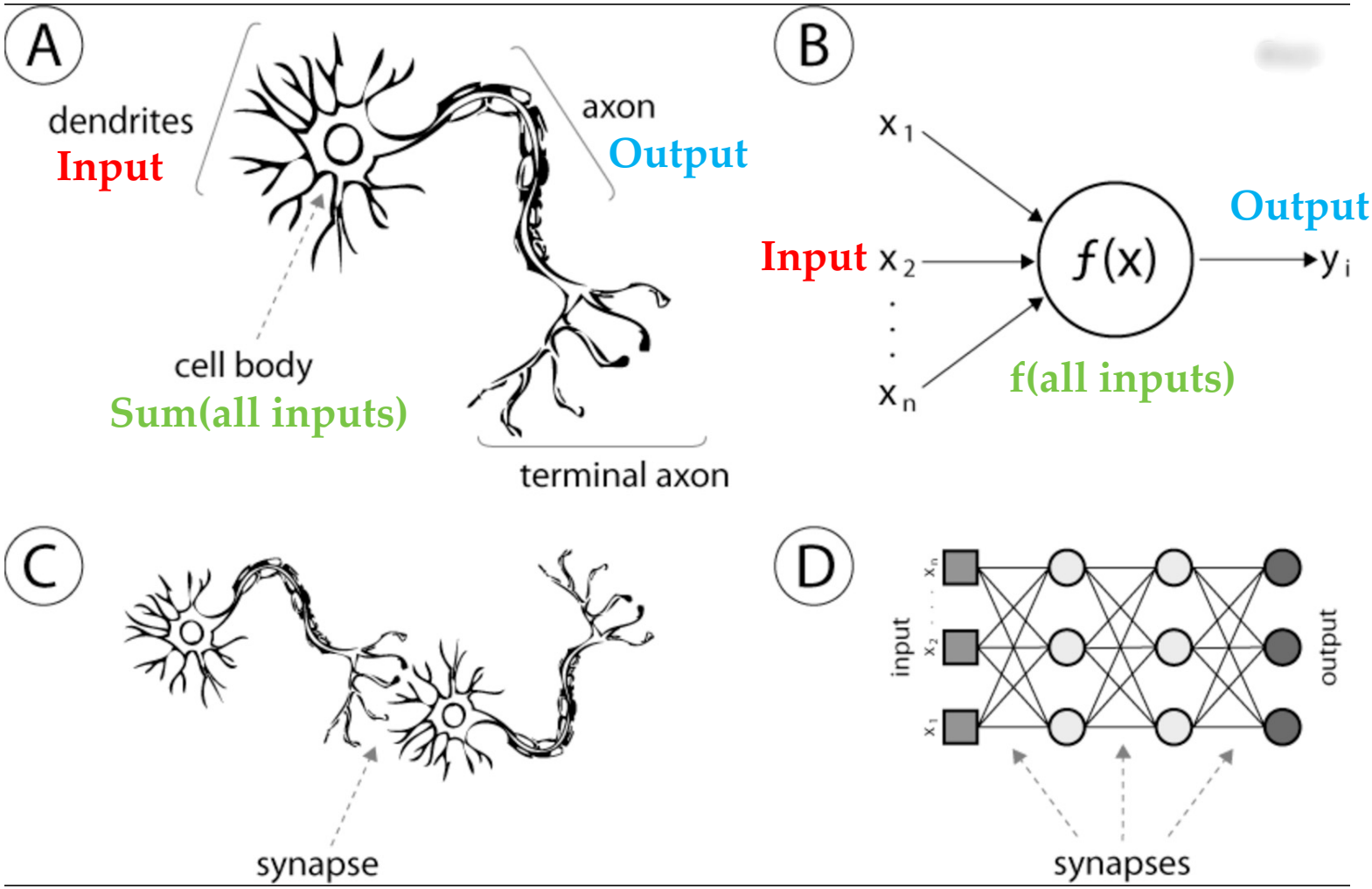


Deep Learning in Robotics

https://youtu.be/l8zKZLqkfII
https://youtu.be/qhUvQiKec2U
https://youtu.be/9j2a1oAHDL8
https://youtu.be/2N_wKXQ6MXA

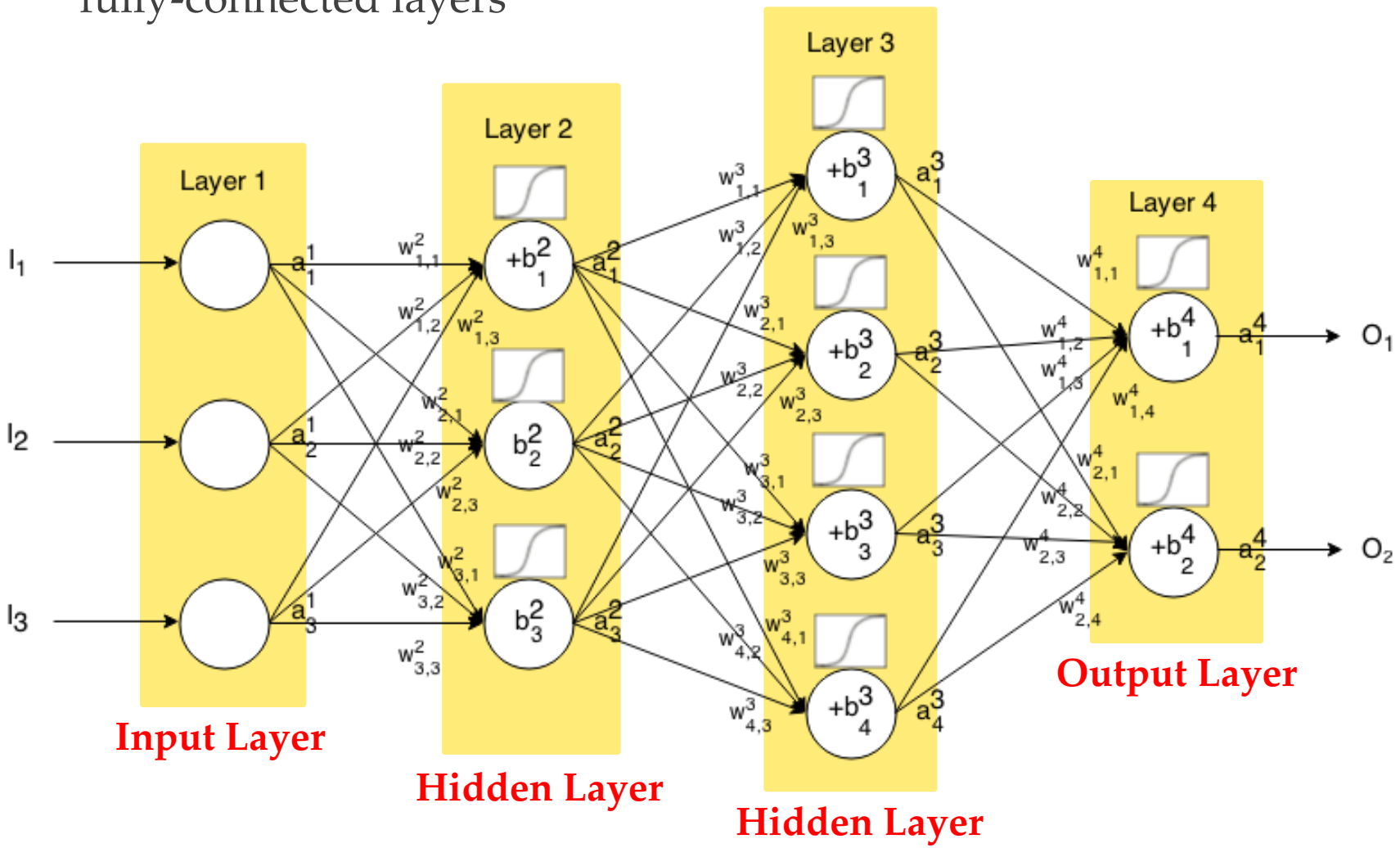
Feed-forward Neural Networks

From Biological Neuron to Neural Network

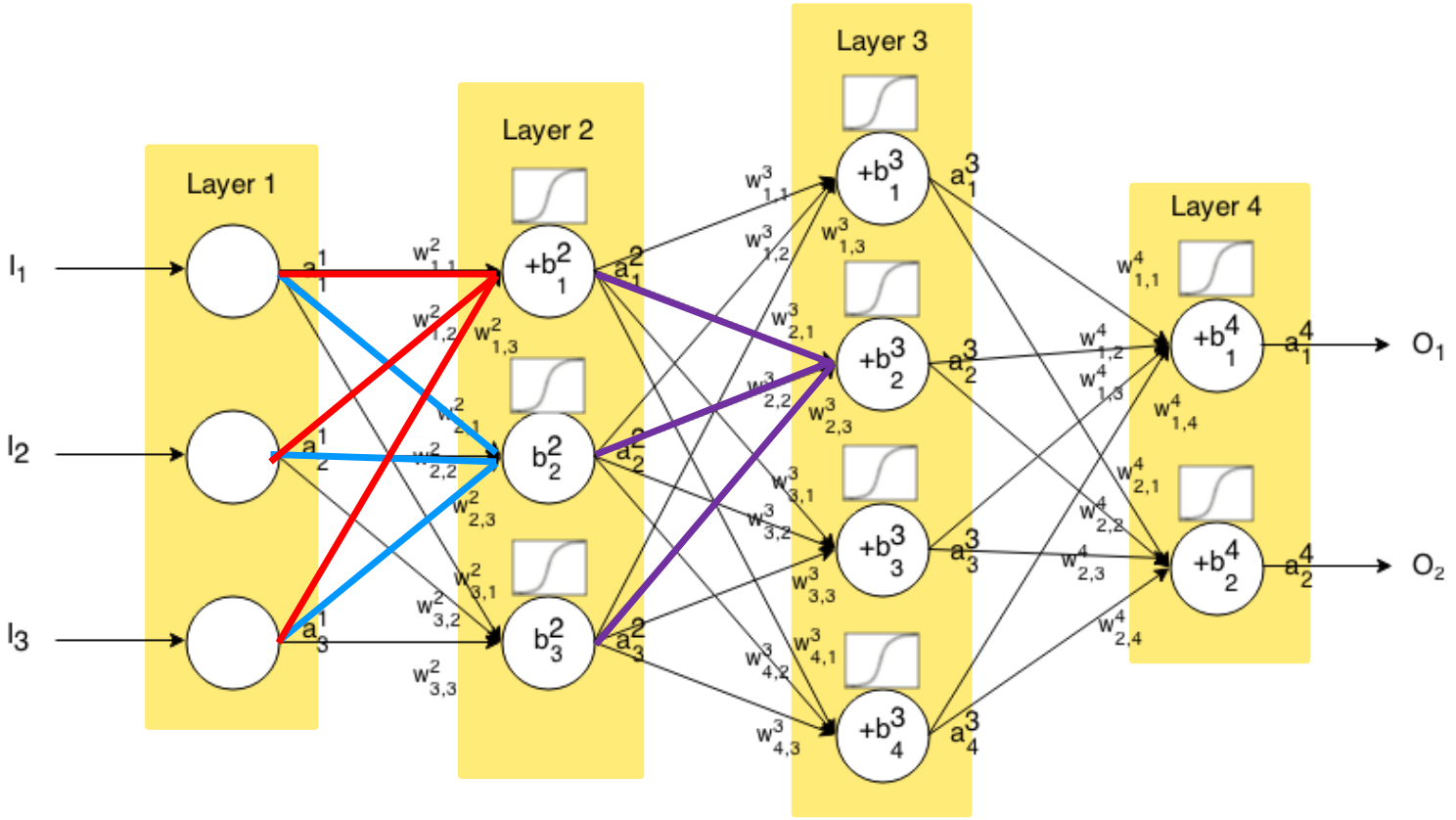


Feedforward Neural Networks

- Feedforward Neural Networks or Multi-Layer Perceptrons (MLPs): fully-connected layers



Feedforward Neural Networks



$a_1^2 = f(w_{1,1}^2 I_1 + w_{1,2}^2 I_2 + w_{1,3}^2 I_3 + b_1^2)$ $f(.)$ is an activation function

$a_2^2 = f(w_{2,1}^2 I_1 + w_{2,2}^2 I_2 + w_{2,3}^2 I_3 + b_2^2)$

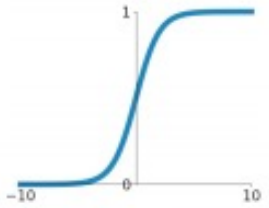
a_2^3?

Activation Functions f(*)

- activation function
  - introduce non-linearity into the neural network

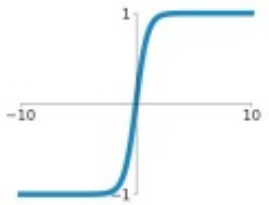
Sigmoid

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



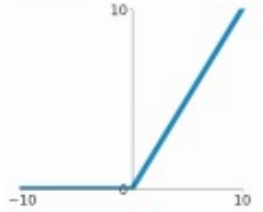
tanh

$$\tanh(x)$$



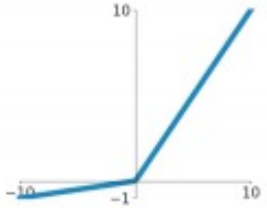
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

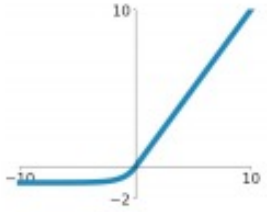


Maxout

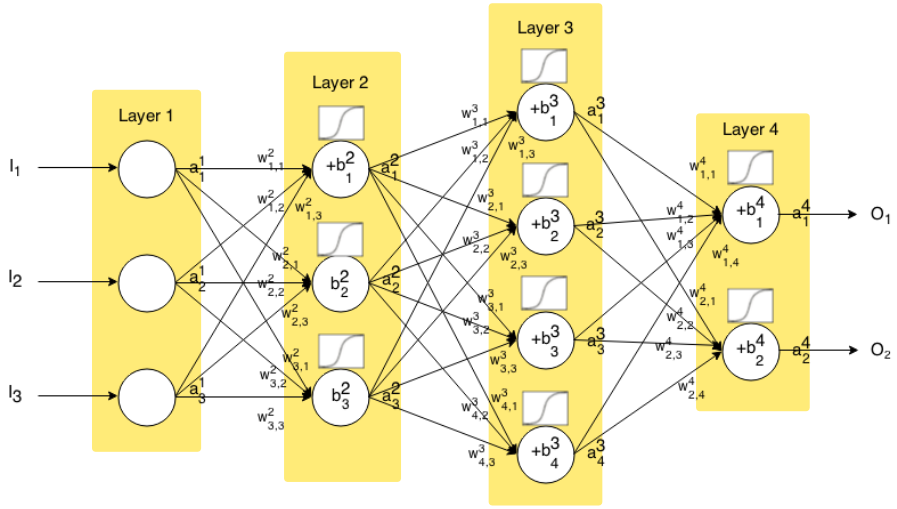
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Feedforward Neural Networks: Vector Formulation

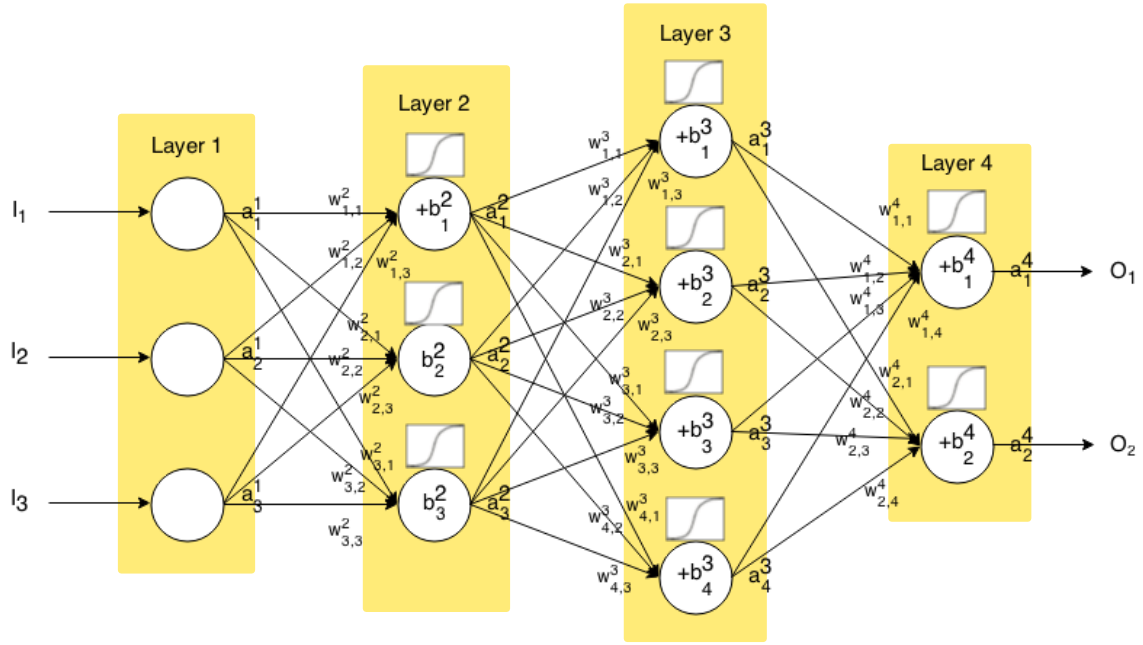


$$a_1^2 = f(w_{1,1}^2 I_1 + w_{1,2}^2 I_2 + w_{1,3}^2 I_3 + b_1^2) \Longrightarrow f(\begin{bmatrix} w_{1,1}^2 & w_{1,2}^2 & w_{1,3}^2 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} + b_1^2)$$

$$\begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} = f(\begin{bmatrix} w_{1,1}^2 & w_{2,1}^2 & w_{3,1}^2 \\ w_{1,2}^2 & w_{2,2}^2 & w_{3,2}^2 \\ w_{1,3}^2 & w_{2,3}^2 & w_{3,3}^2 \end{bmatrix}^T \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} + \begin{bmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \end{bmatrix}) \qquad a^2 = f({W^2}^T I + b^2)$$

$$a^3 = f({W^3}^T a^2 + b^3) \qquad O = f({W^4}^T a^3 + b^4)$$

Feedforward Neural Networks: Vector Formulation



$$a^2 = f(W^{2^T} I + b^2) \quad a^3 = f(W^{3^T} a^2 + b^3) \quad O = f(W^{4^T} a^3 + b^4)$$

```
# forward pass of a 3 layer feed-forward neural networks
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (sigmoid)
I = np.random.randn(3, 1) # random input vector
a2 = f(np.dot(W2, I) + b2) # first hidden layer activation
a3 = f(np.dot(W3, a2) + b3) # second hidden layer activation
O = f(np.dot(W4, a3) + b4) # output
```

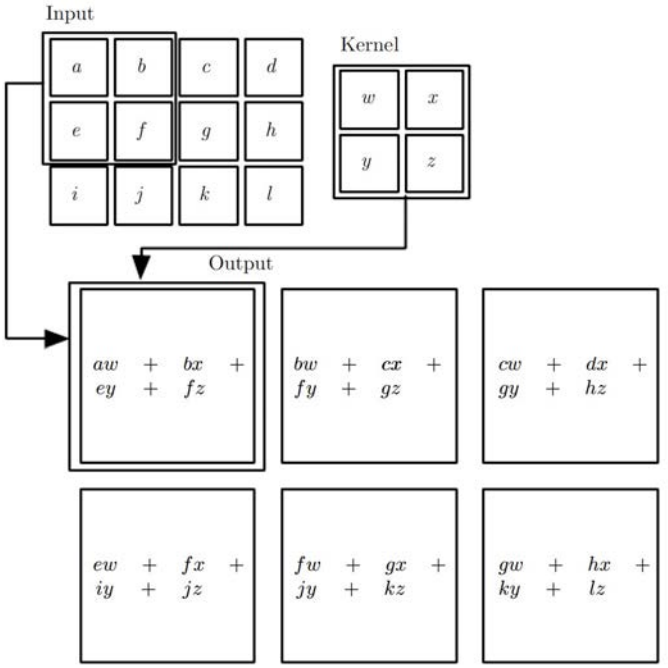
Convolutional Neural Networks (CNNs)

From MLPs to CNNs

- Feed-forward Neural Networks or Multi-Layer Perceptrons (MLPs)
  - many multiplications

- CNNs are similar to Feed-forward Neural Networks
  - convolution instead of general matrix multiplication

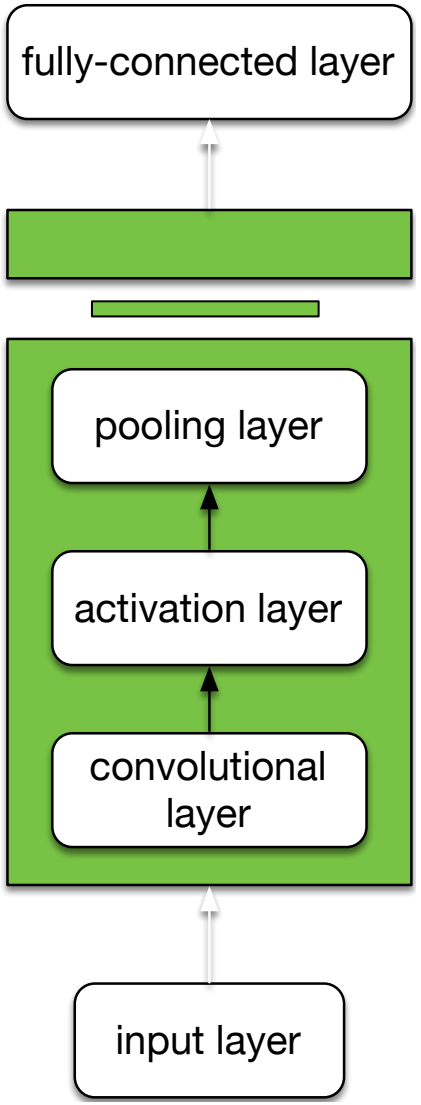
$$S(i,j) = (I * K)(i,j)$$
$$= \sum_m \sum_n I(i+m, j+n) K(m,n)$$



CNNs

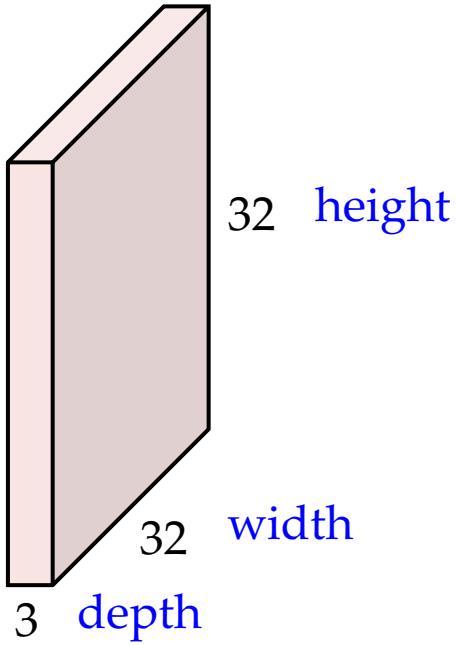
- 3 Main Types of Layers:
  - **convolutional layer**
  - **activation layer**
  - **pooling layer**

- repeat many times

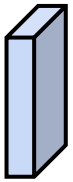


CNNs: Convolution Layer

32x32x3 image



5x5x3 filter



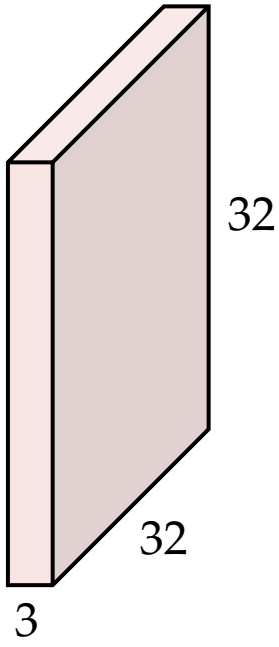
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Slides courtesy of Andrej Karpathy

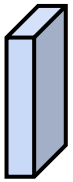
CNNs: Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

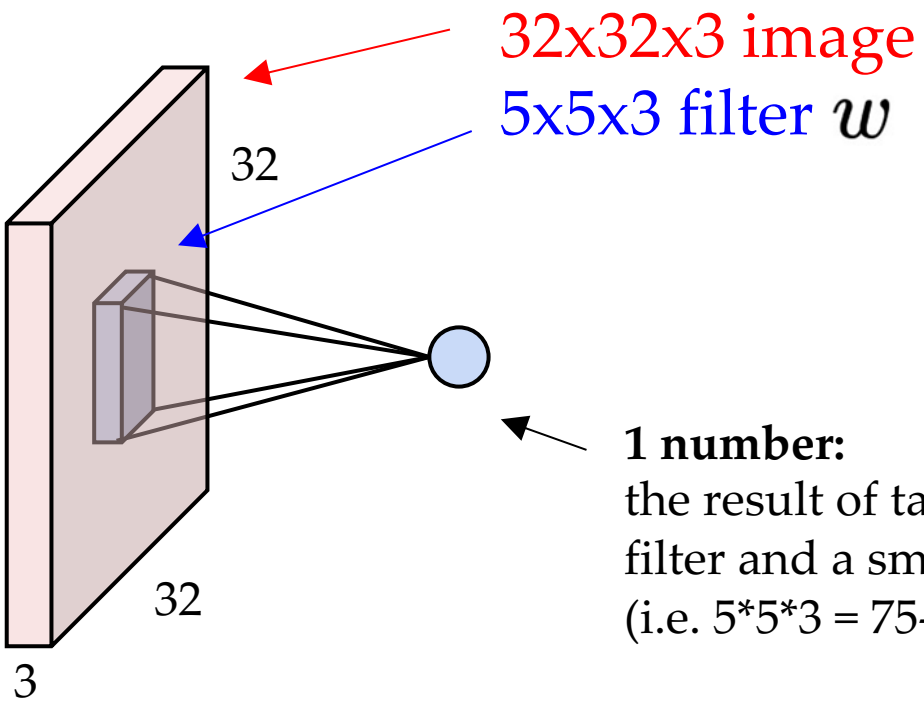


5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

CNNs: Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

CNNs: Convolution Layer

Stride 2

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

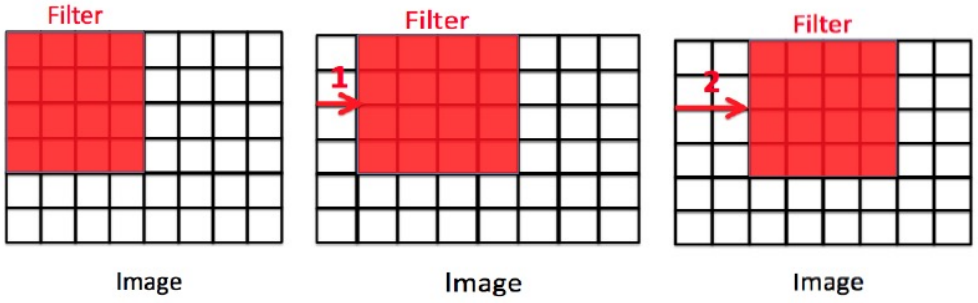
Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

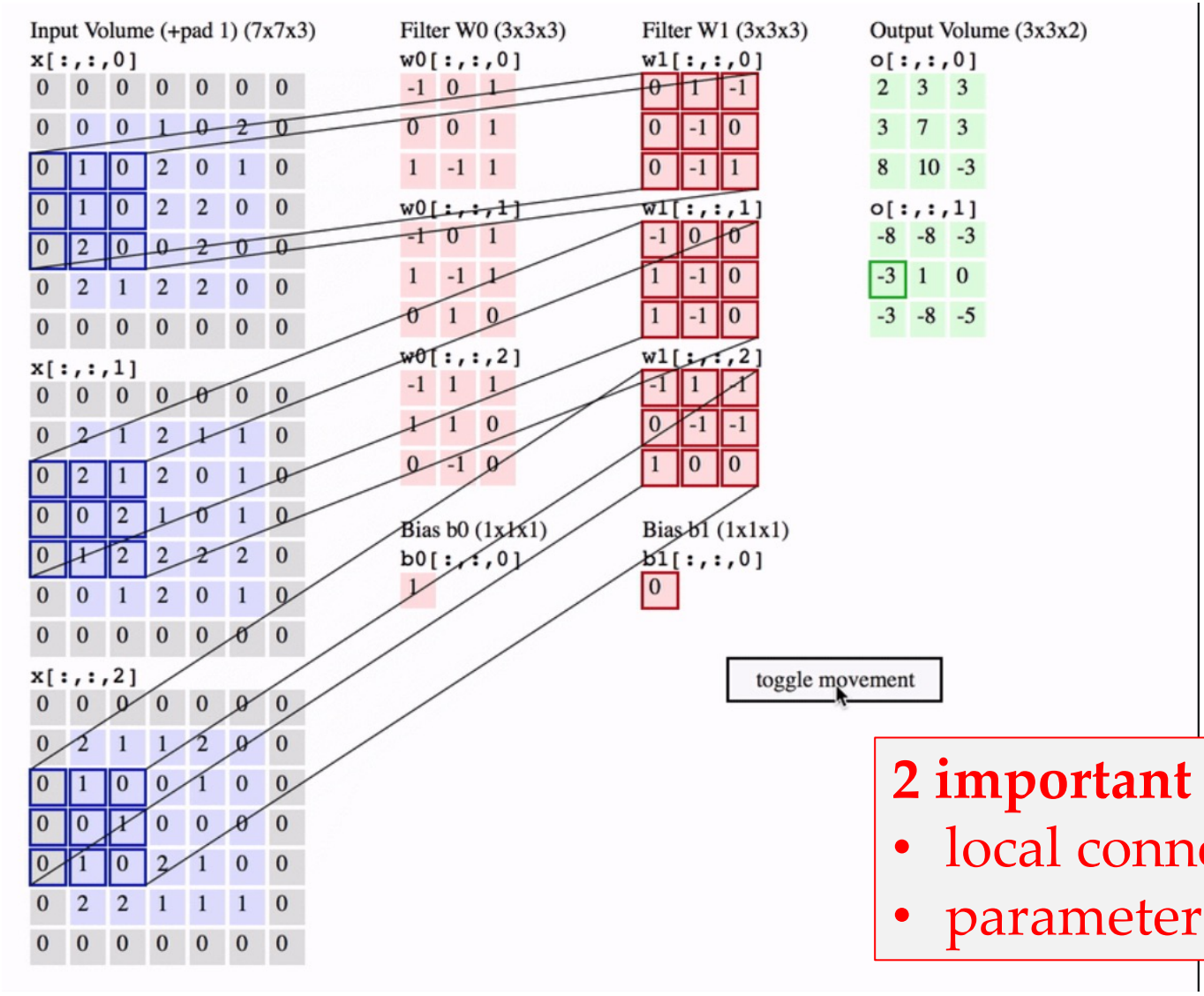
o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5



Stride is the number of pixels shifts over the input matrix.

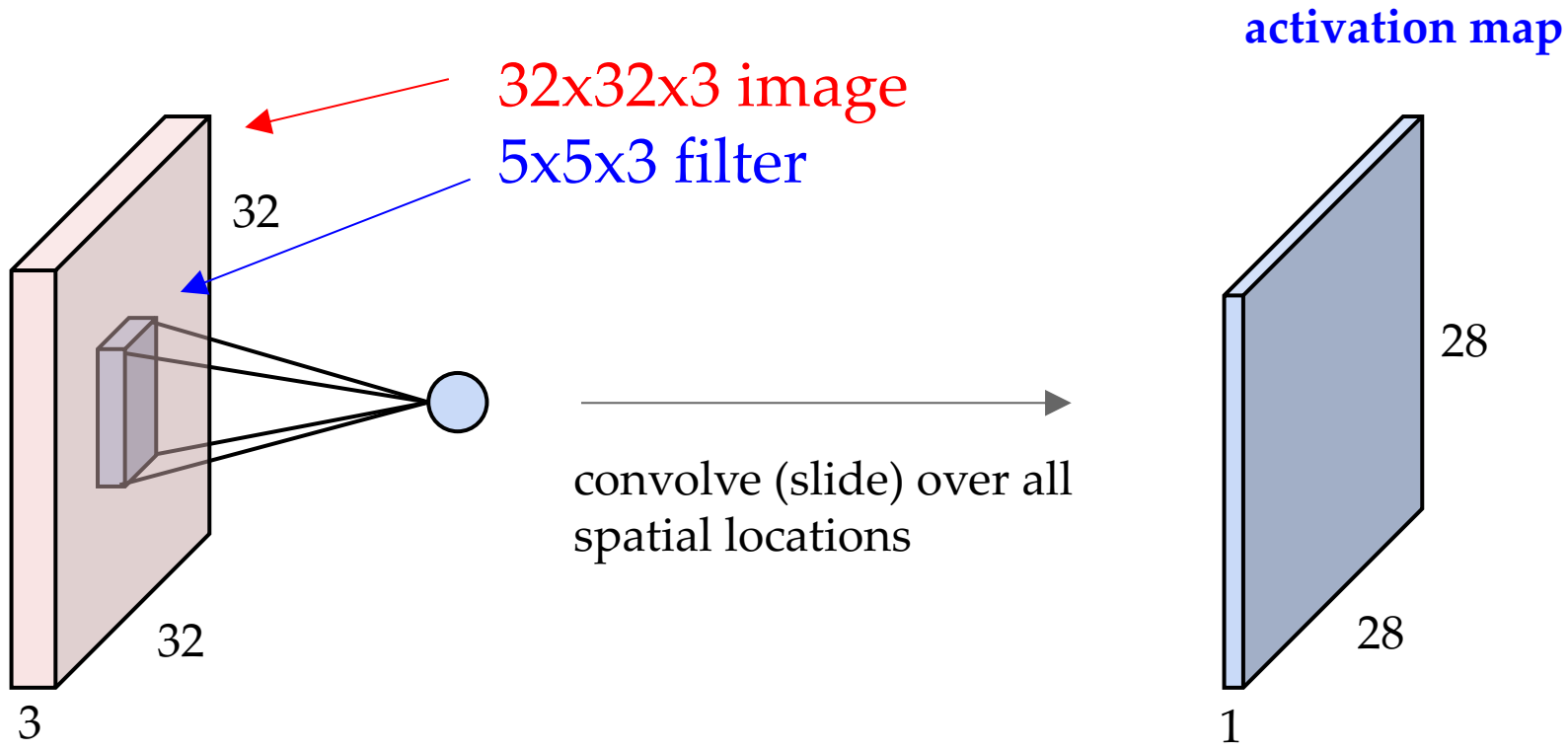
CNNs: Convolution Layer



2 important ideas:

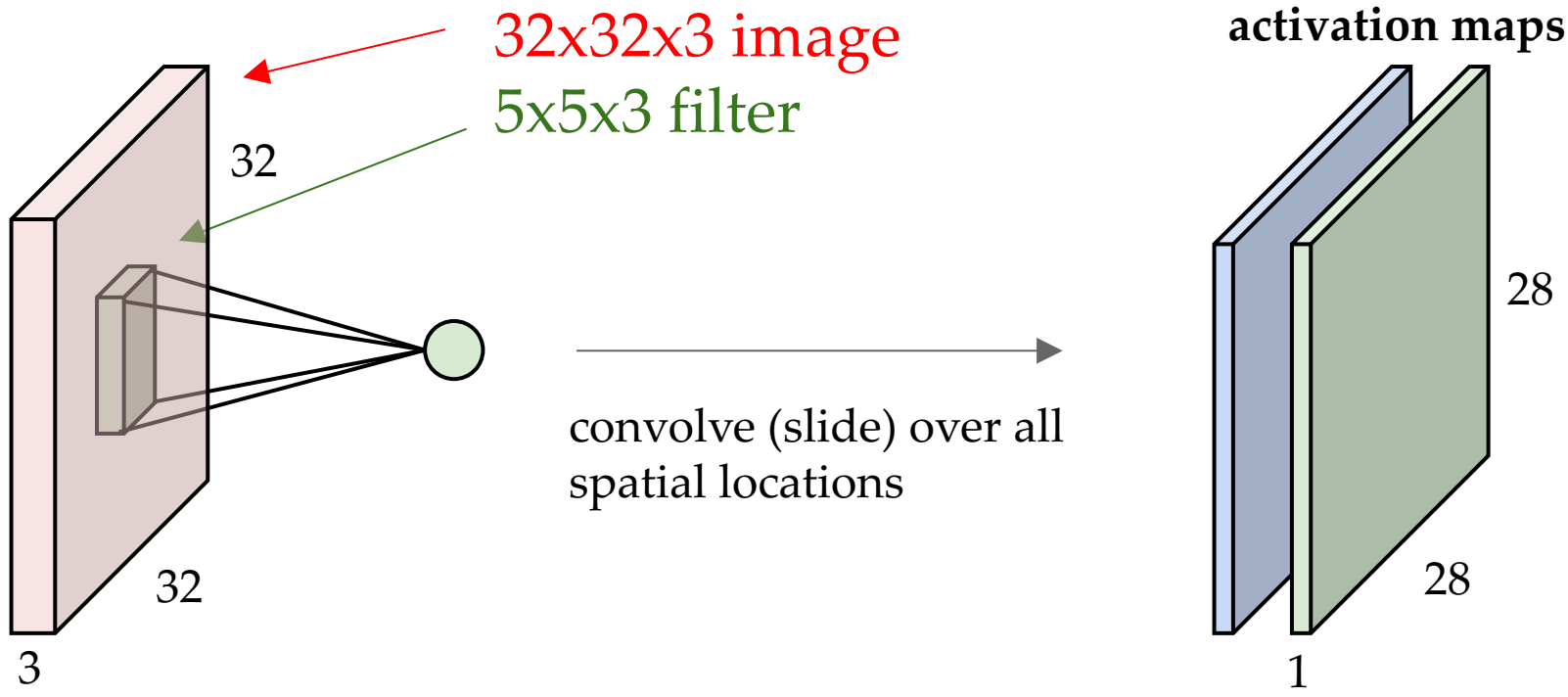
- local connectivity
- parameter sharing

CNNs: Convolution Layer



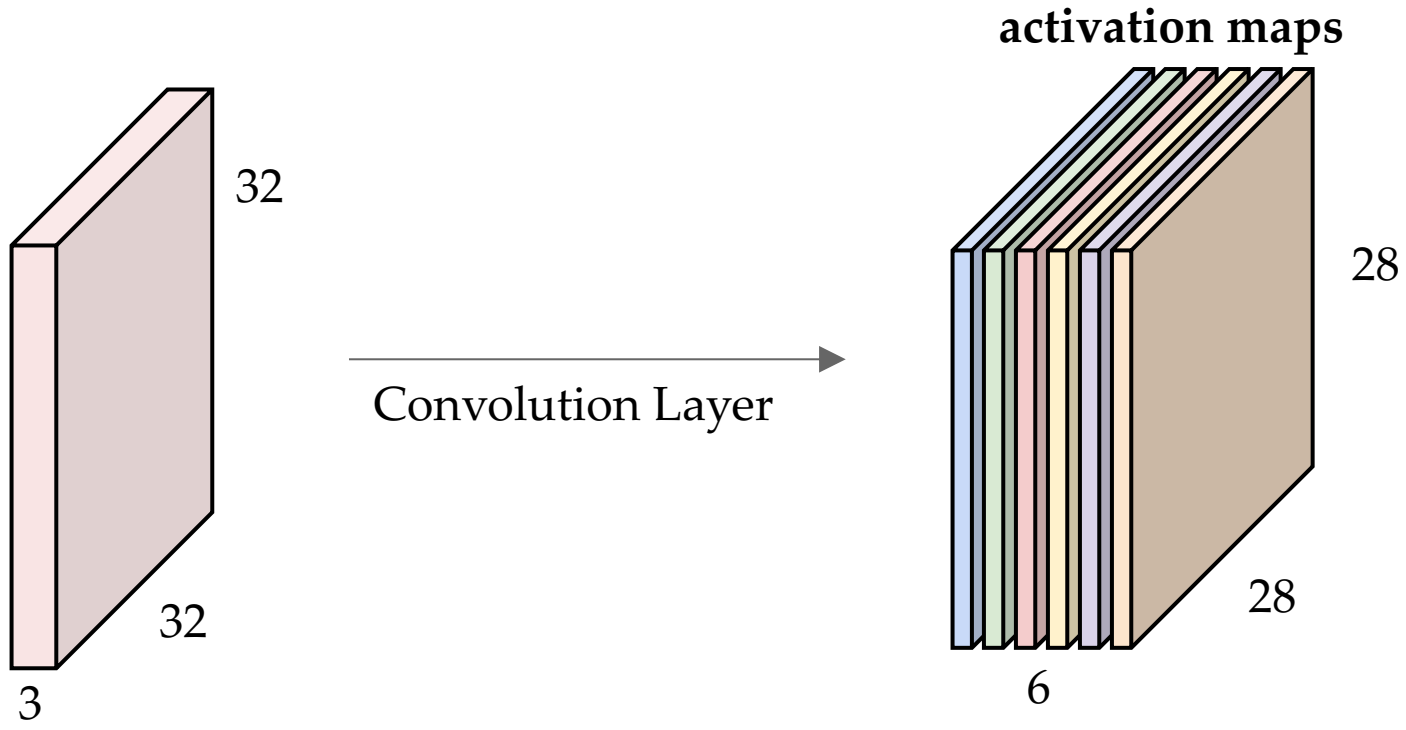
CNNs: Convolution Layer

consider a second, green filter



CNNs: Convolution Layer

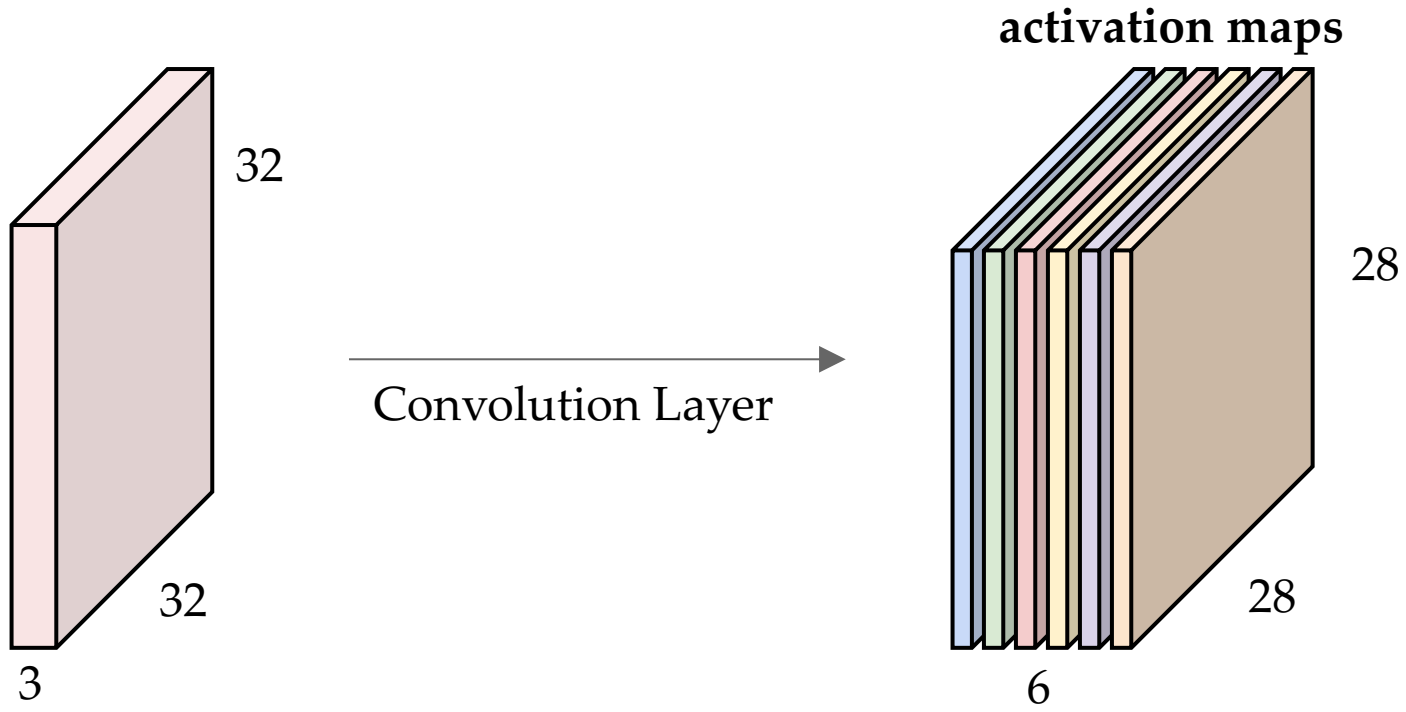
For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



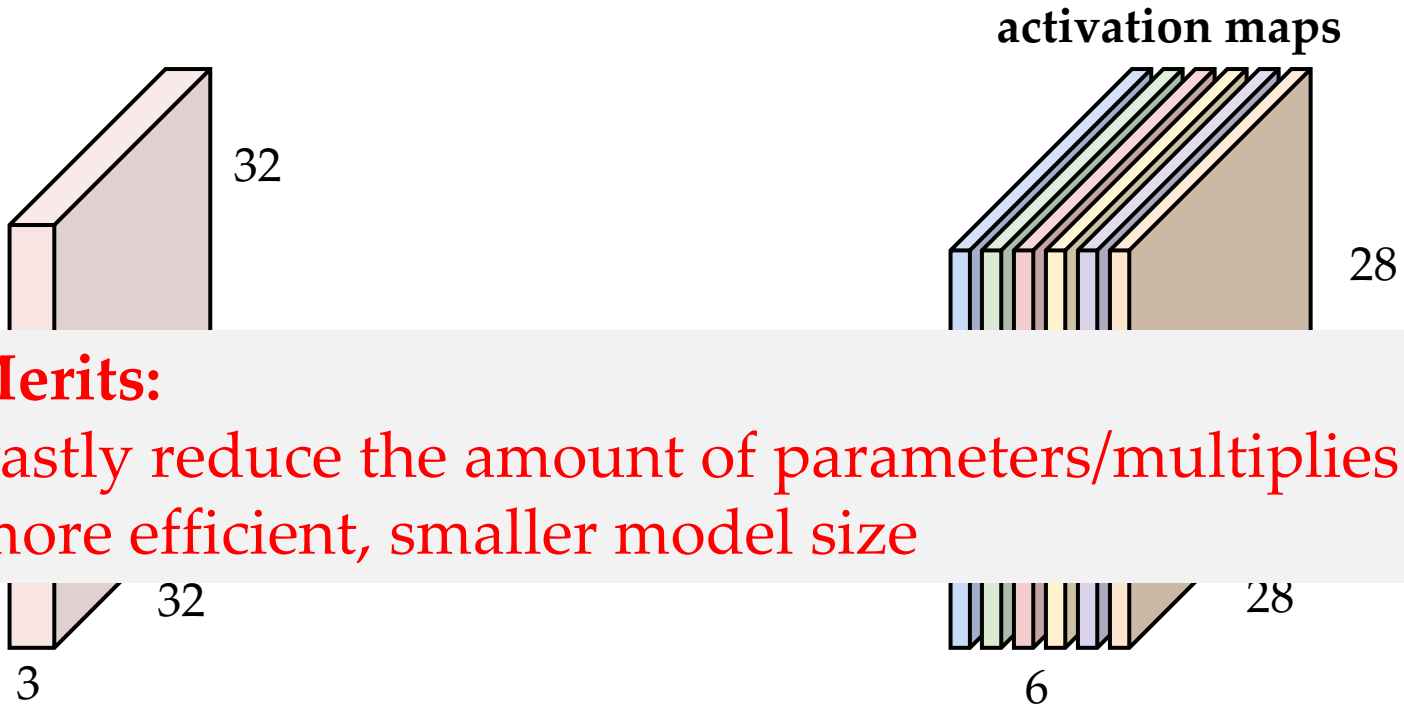
We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters and multiplies would this be if we used a fully connected layer instead?

A: (32*32*3)*(28*28*6) = **14.5M parameters, ~14.5M multiplies**

CNNs: Convolution Layer

For example, if we had 6 of 5x5 filters, we'll get 6 separate activation maps:



2 Merits:

- vastly reduce the amount of parameters/multiplies
- more efficient, smaller model size

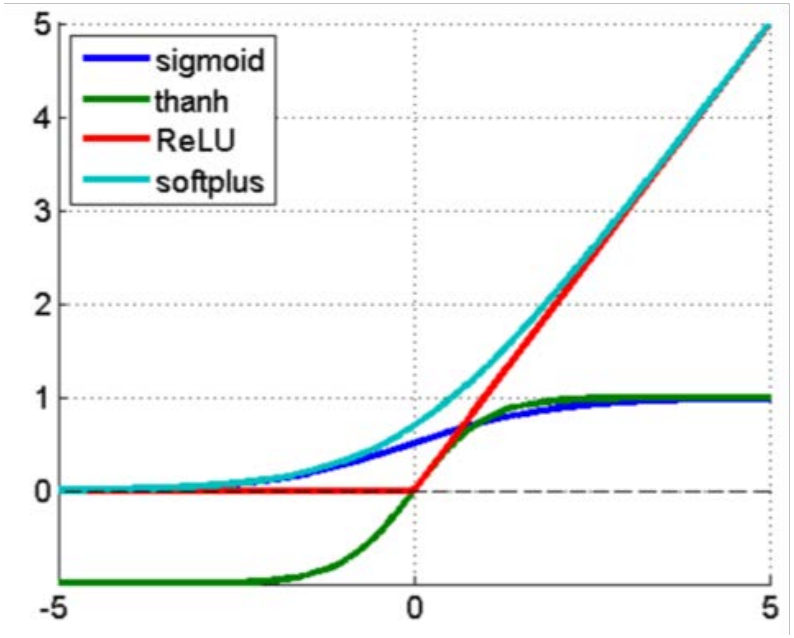
We processed [32x32x3] volume into [28x28x6] volume.

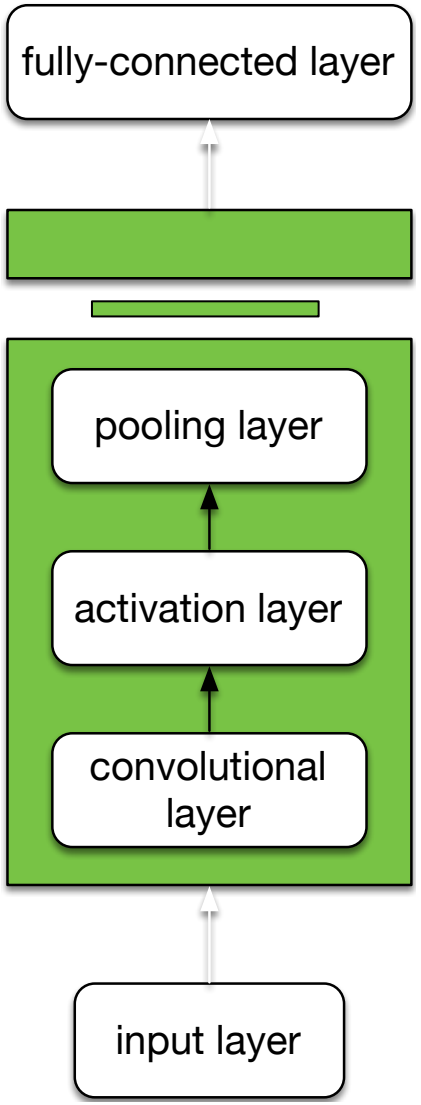
Q: how many parameters are used instead? --- And how many multiplies?

A: (5*5*3)*6 = **450 parameters**, (5*5*3)*(28*28*6) = **~350K multiplies**

CNNs: Activation Layer

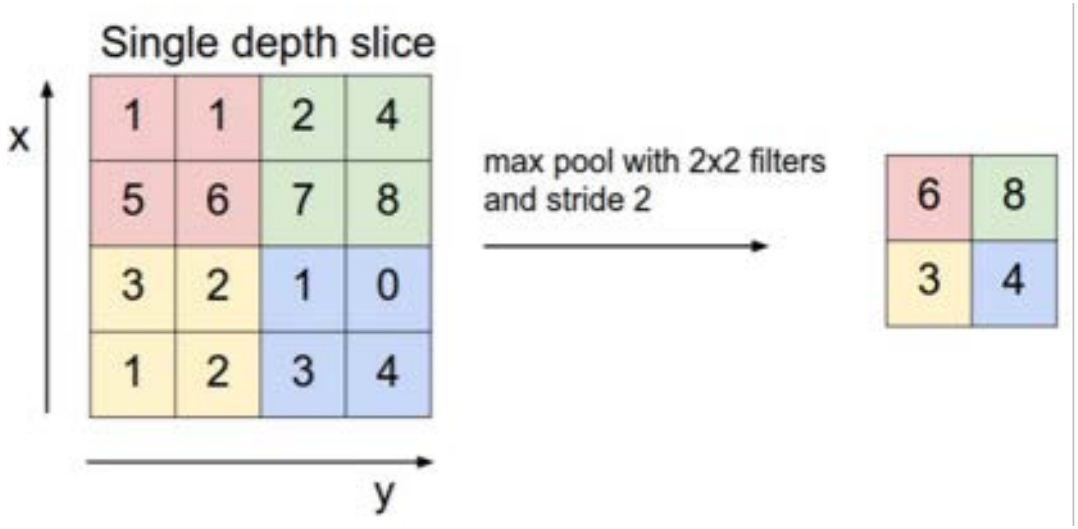
- 3 Main Types of Layers:
  - convolutional layer
  - **activation layer**
  - **pooling layer**

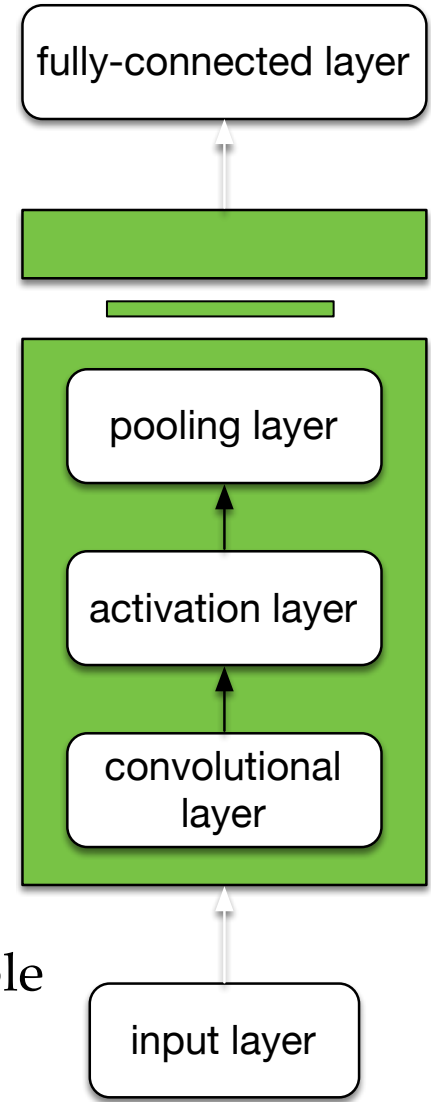




CNNs: Pooling Layer

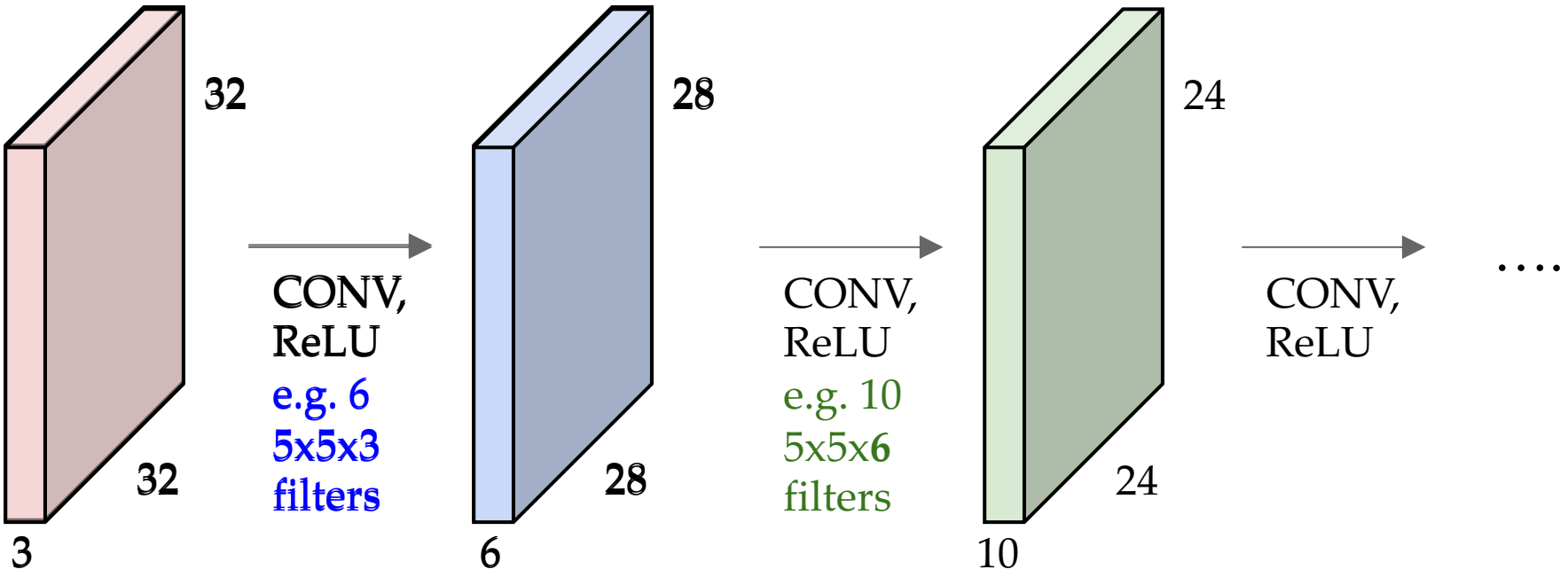
- 3 Main Types of Layers:
  - convolutional layer
  - activation layer
  - **pooling layer**





makes the representations smaller and more manageable

CNNs: A sequence of Convolutional Layers



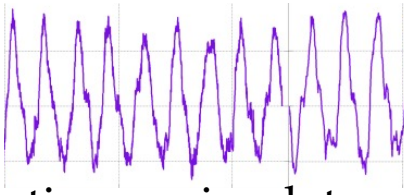
CNN Architectures

Hand-Crafted Features by Human

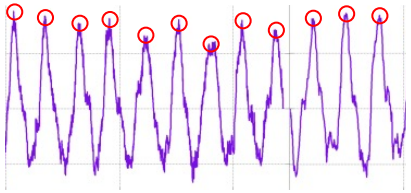
Pervasive Data

Feature Extraction (hand-crafted)

Inference



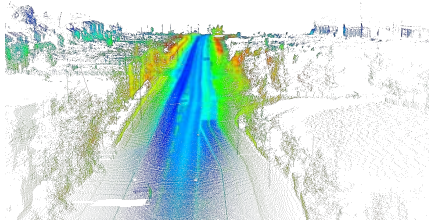
time-series data



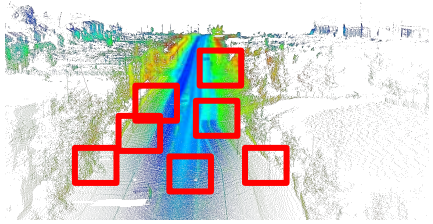
Activities,
Context, …

vision

Locations,
Scene types,
Semantics, …



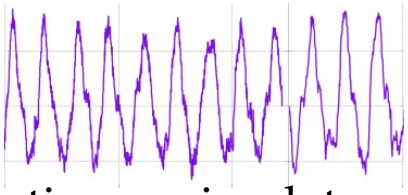
point cloud



Objects,
Structure, …

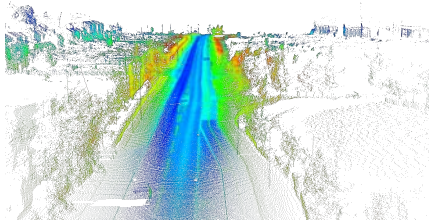
Feature Engineering and Representation

Pervasive Data



time-series data

vision



point cloud

$256^{3x800x600}$

$2^{?x?x?}$

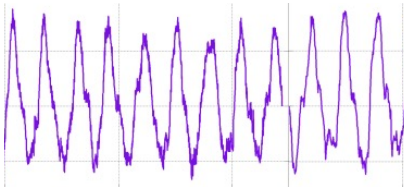
Raw data

≈

Bad Representation

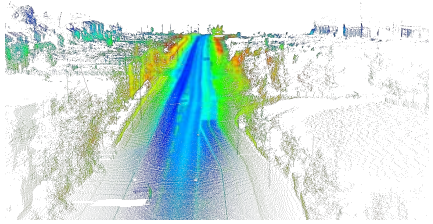
Deep Learning: Representation Learning

Pervasive Data



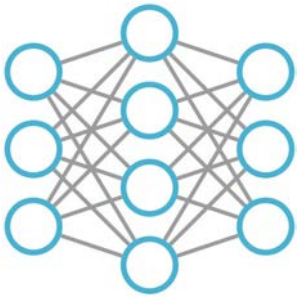
time-series data

vision



point cloud

End-to-End Learning



Inference

Activities,
Context, …
Locations,
Scene types, …
Structure,
Semantics, …

automatically learn effective feature representation to solve the problem

LeNet - 1998

- Convolution for neural networks
- Fully-connected outputs
- weight-sharing is a key to reduce number of parameters

Foundation of modern ConvNets

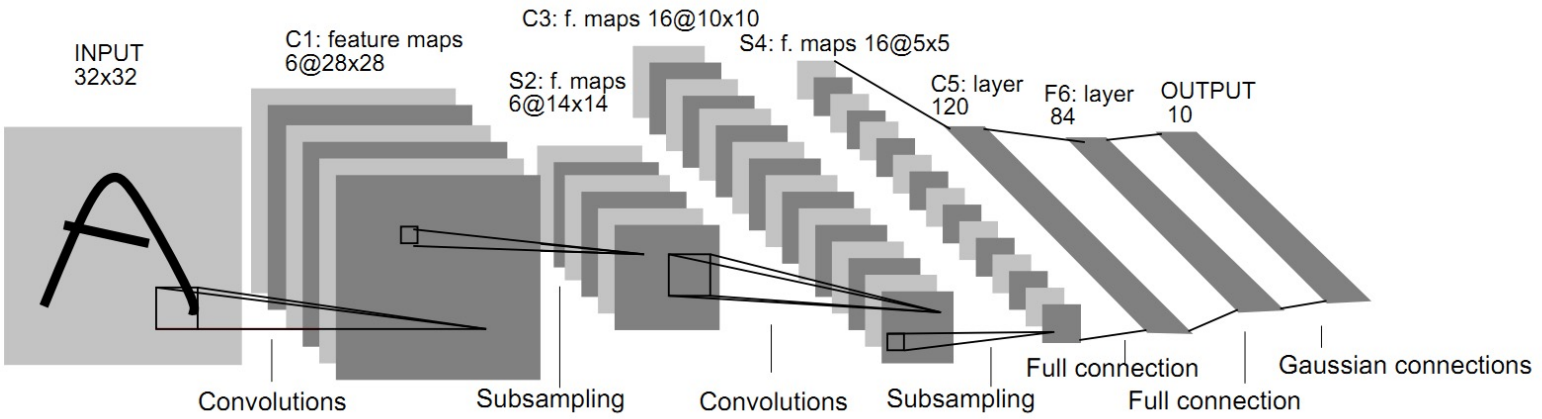


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

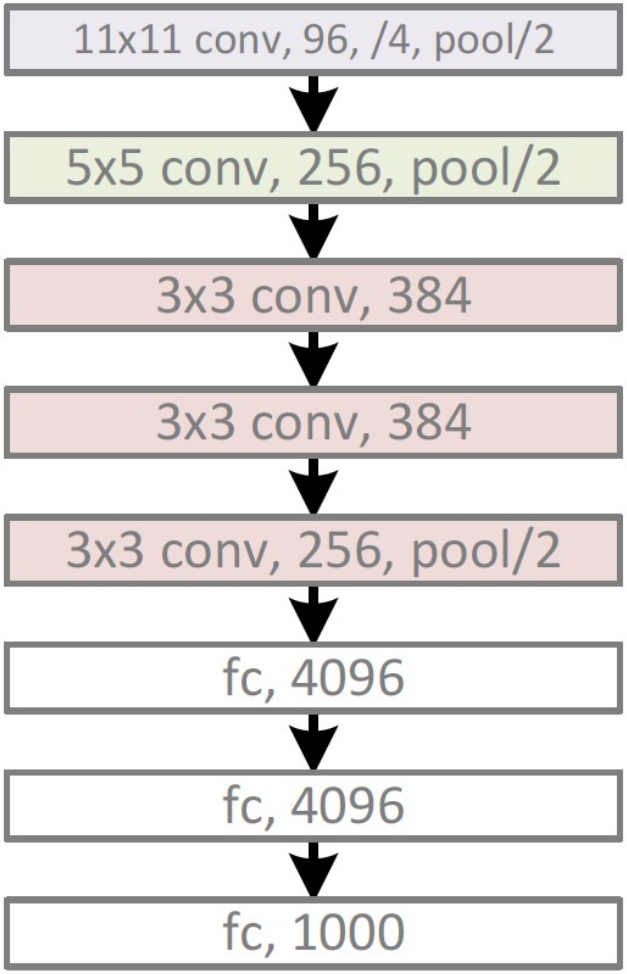
"Gradient-based learning applied to document recognition", LeCun et al. 1998

AlexNet – 2012

8 layers: 5 conv and max-pooling + 3 fully-connected

LeNet-style backbone, plus:

- ReLU
  - Accelerate training
  - better gradprop (vs. tanh)
- Dropout
  - Reduce overfitting
- Data augmentation
  - Image transformation
  - Reduce overfitting



"ImageNet Classification with Deep Convolutional Neural Networks", Krizhevsky, Sutskever, Hinton. NIPS 2012

VGG16/19 - 2014

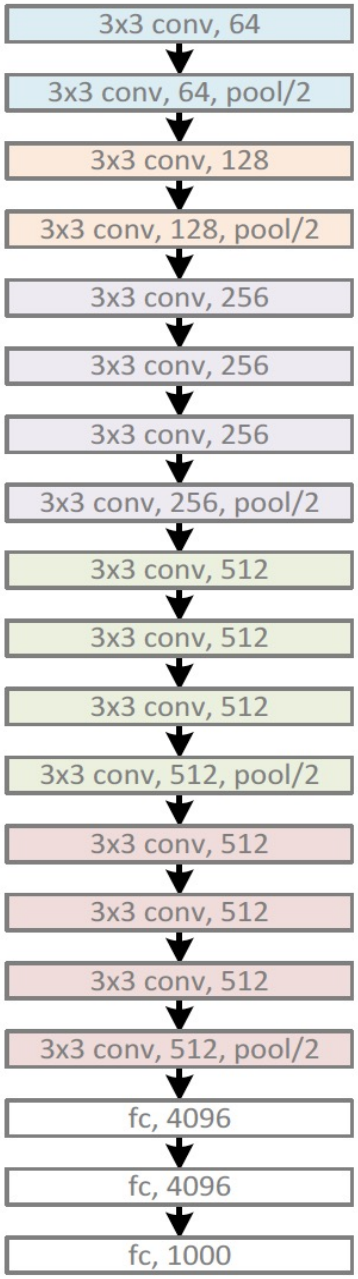
Very deep ConvNet

Modularized design

- 3x3 Conv as the module
- Stack the same module
- Same computation for each module

Stage-wise training

- VGG-11 => VGG-13 => VGG-16



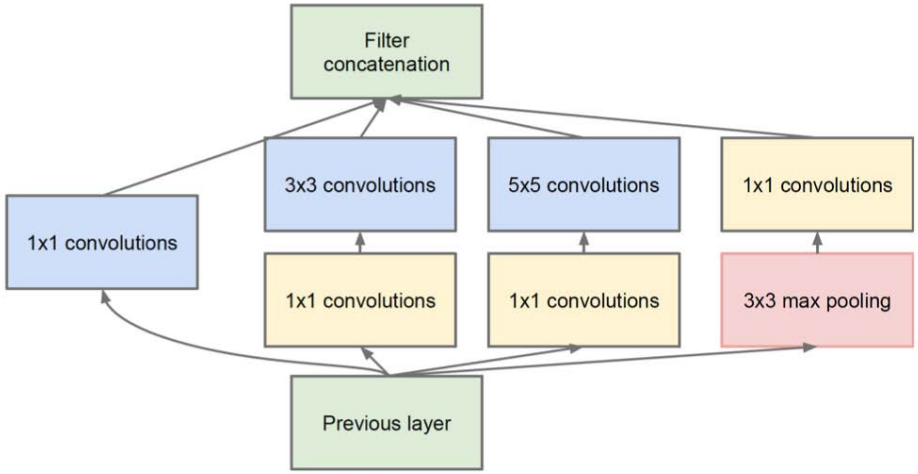
"Very Deep Convolutional Networks for Large-Scale Image Recognition", Simonyan & Zisserman. arXiv 2014 (ICLR 2015)

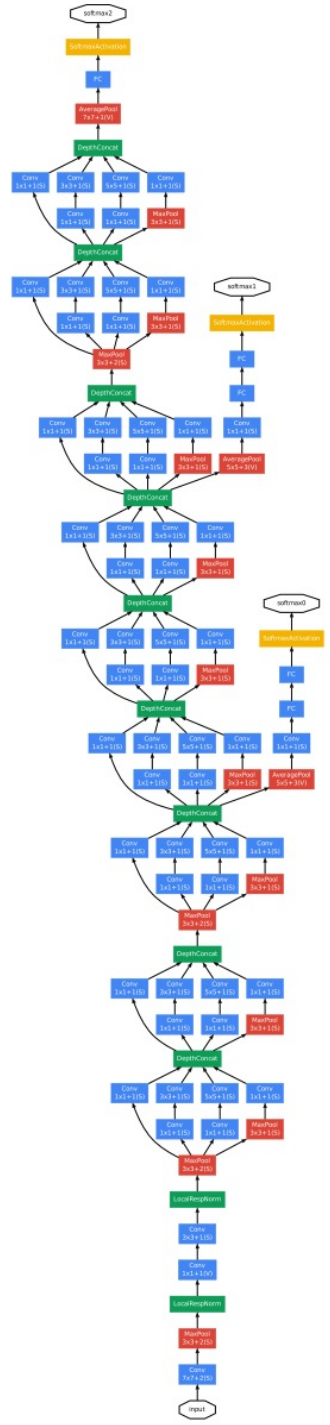
GoogleNet/Inception - 2014

22 layers

Multiple branches

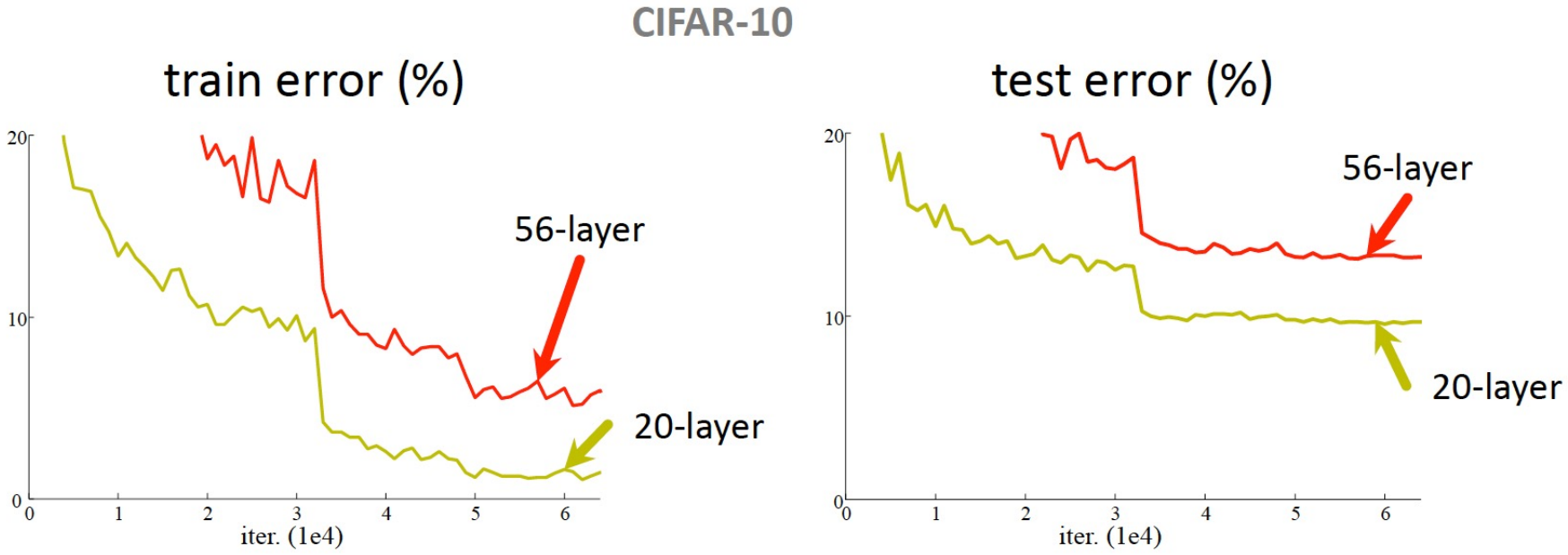
- e.g., 1x1, 3x3, 5x5 convolutions and pooling
- merged by concatenation
- Reduce dimensionality by 1x1 conv before expensive 3x3/5x5 conv (change num of channels)





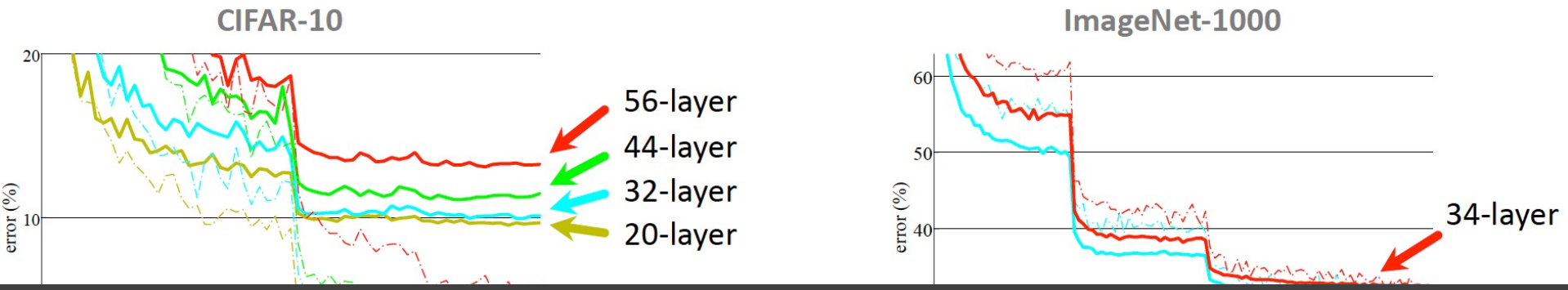
Szegedy et al. “Going deeper with convolutions”. arXiv 2014 (CVPR 2015)

Simply stacking layers?



- Plain nets: stacking 3x3 conv layers
- 56-layer net has **higher training error and test error** than 20-layer net
- A deeper model should not have higher training error

Going Deeper



Cannot go deeper for deep neural networks

Problem:

deeper plain nets have higher training error on various datasets

Optimization difficulties:

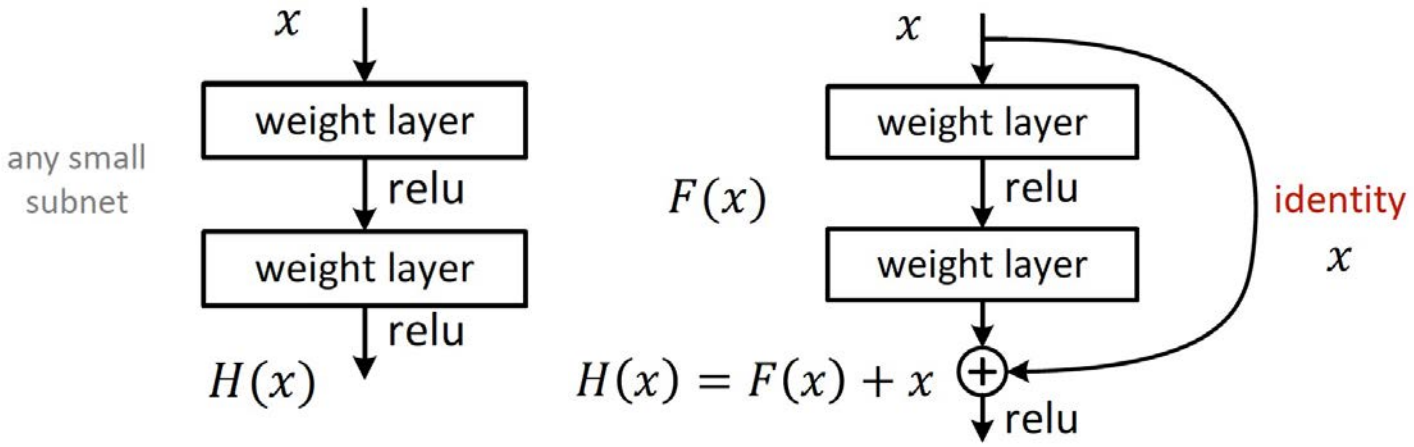
- vanishing gradient
- solvers struggle to find the solution when going deeper

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

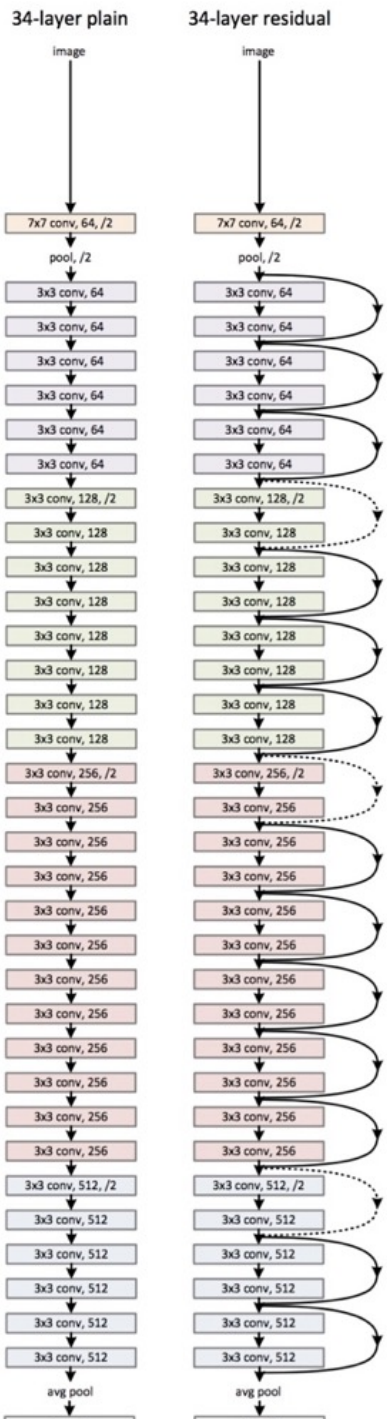
ResNets-2016

Plain net

Residual net

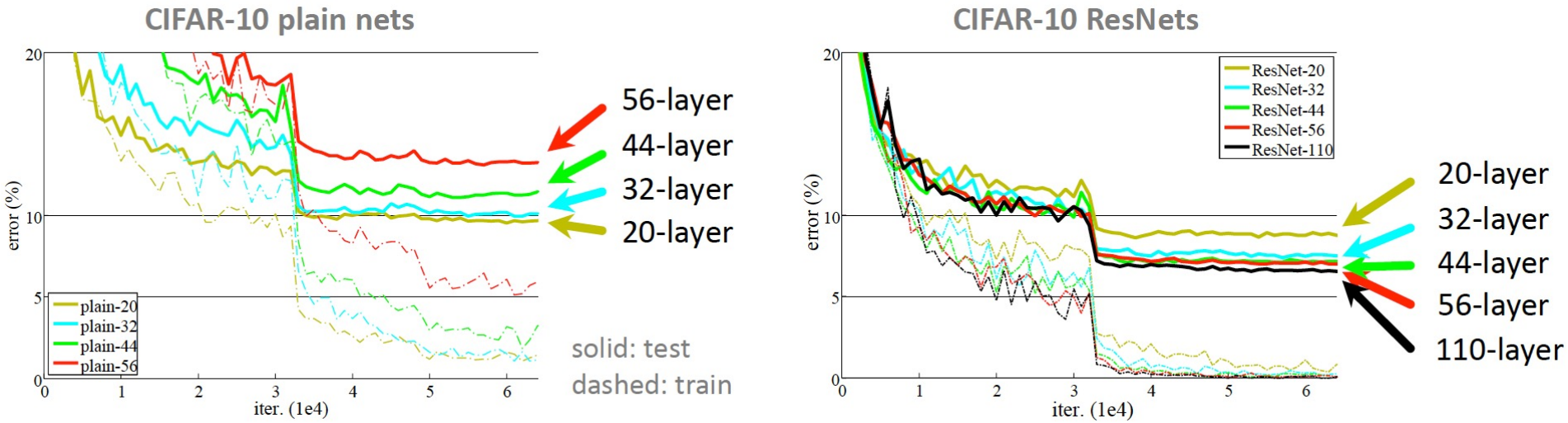


gradients can flow directly through
the skip connections backwards



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

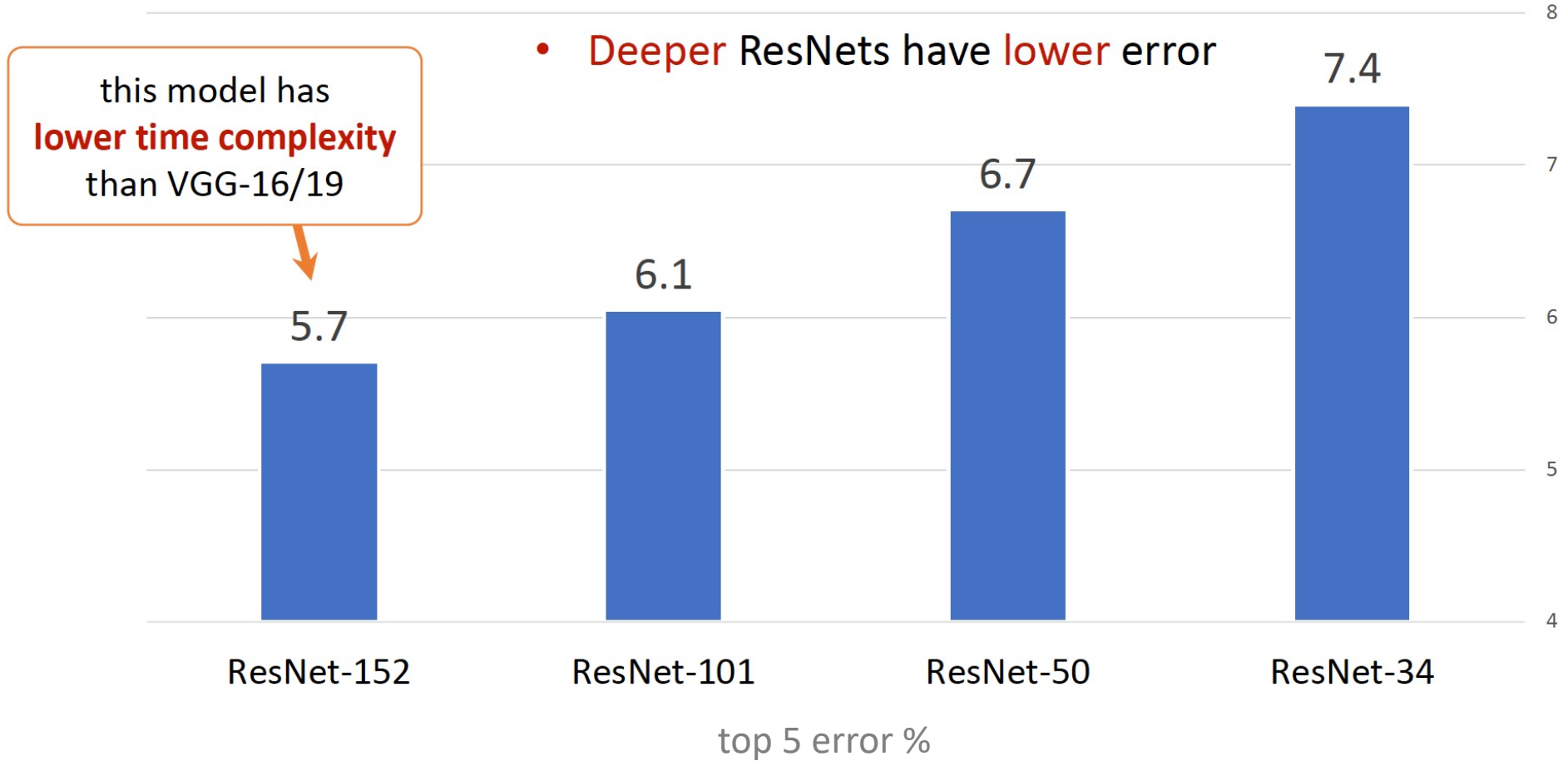
ResNets-2016



- Deep ResNets can be trained easier
- Deeper ResNets have lower training error, and also lower test error

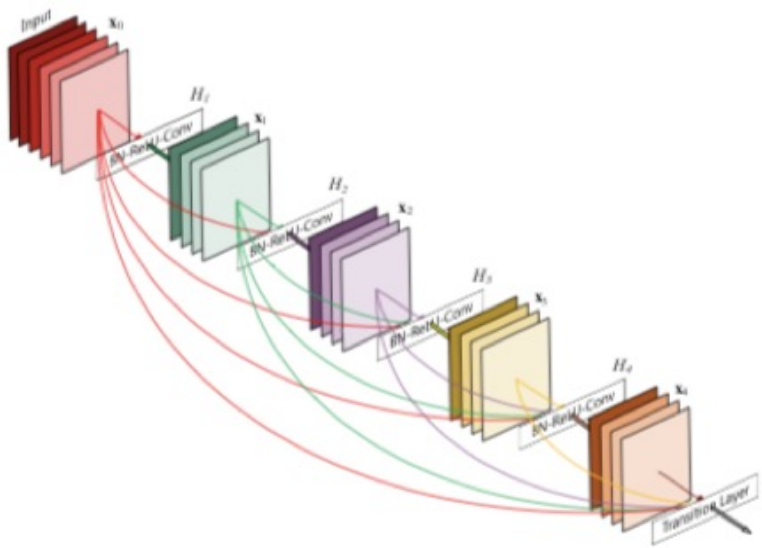
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

ImageNet experiments



DenseNets - 2017

- simply connect every layer directly with each other
  - each layer has direct access to the gradients from the loss function and the original input image



- DenseNets concatenate the output feature maps of the layer with the incoming feature maps.

$$x_l = H_l(x_{l-1})$$

$$x_l = H_l(x_{l-1}) + x_{l-1}$$

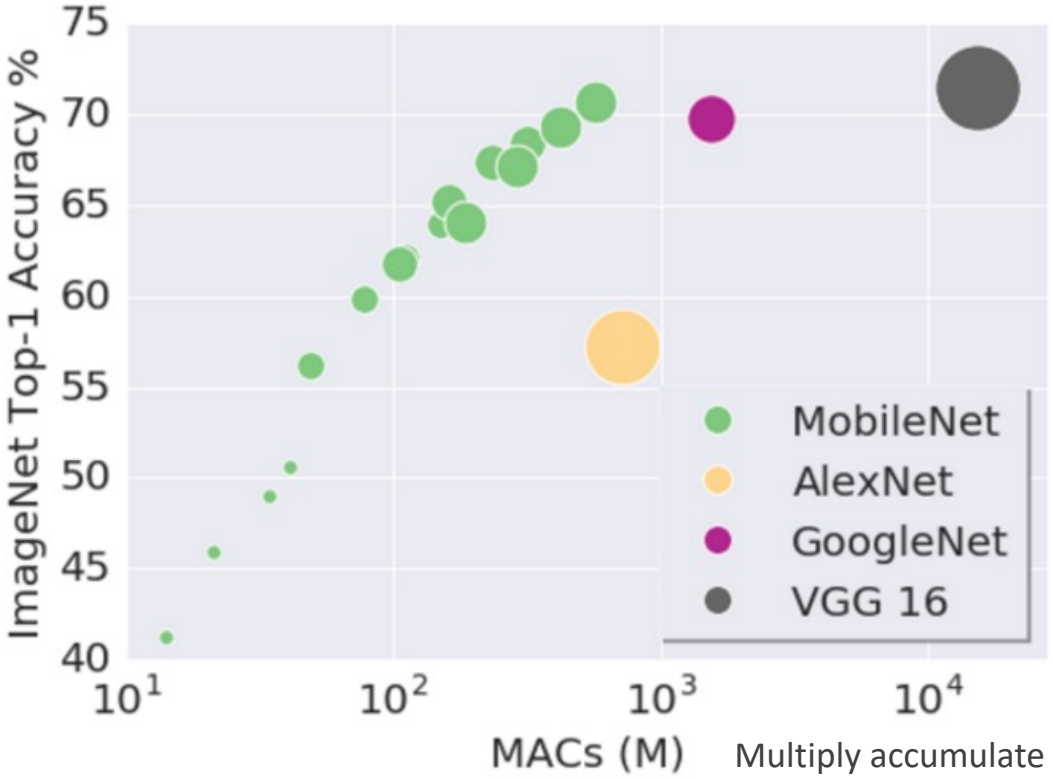
$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$$

G. Huang, Z. Liu and L. van der Maaten, "Densely Connected Convolutional Networks," 2017.

MobileNets - 2017

better accuracy vs better efficiency

Light-weight ConvNets for mobile applications using depth-wise convolutions



Multiply accumulate operation

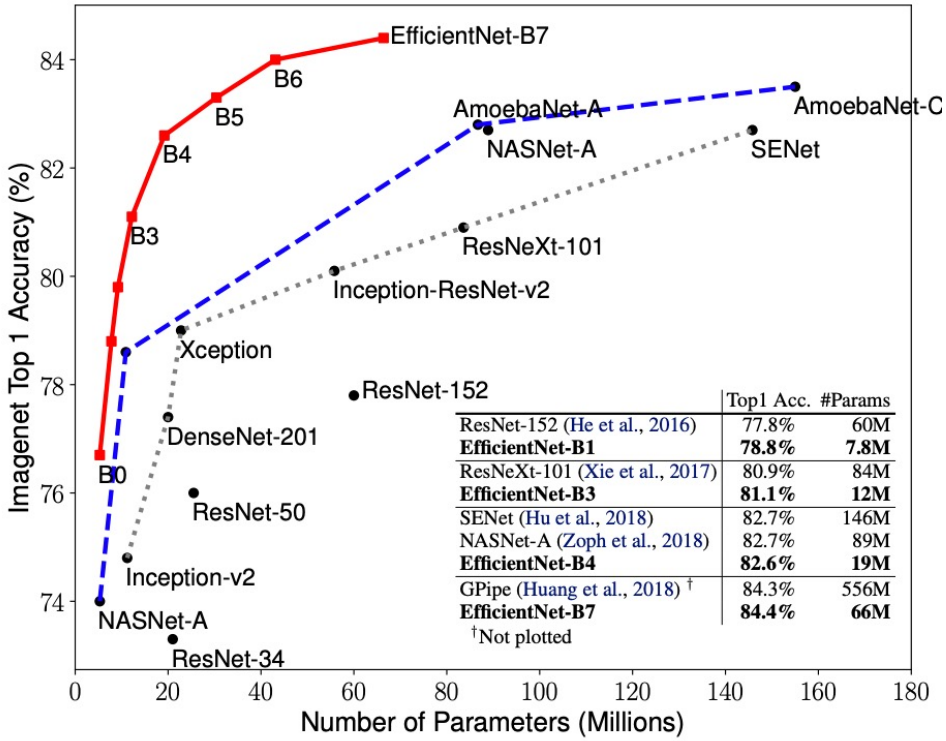
Howard. et. al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications 2017

EfficientNet - 2019

carefully design architecture to achieve top results with reasonable parameters

built upon a model (B0) from neural architecture search

Compound scaling: scale up network depth (more layers), width (more channels per layer), resolution (input image) simultaneously



	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**

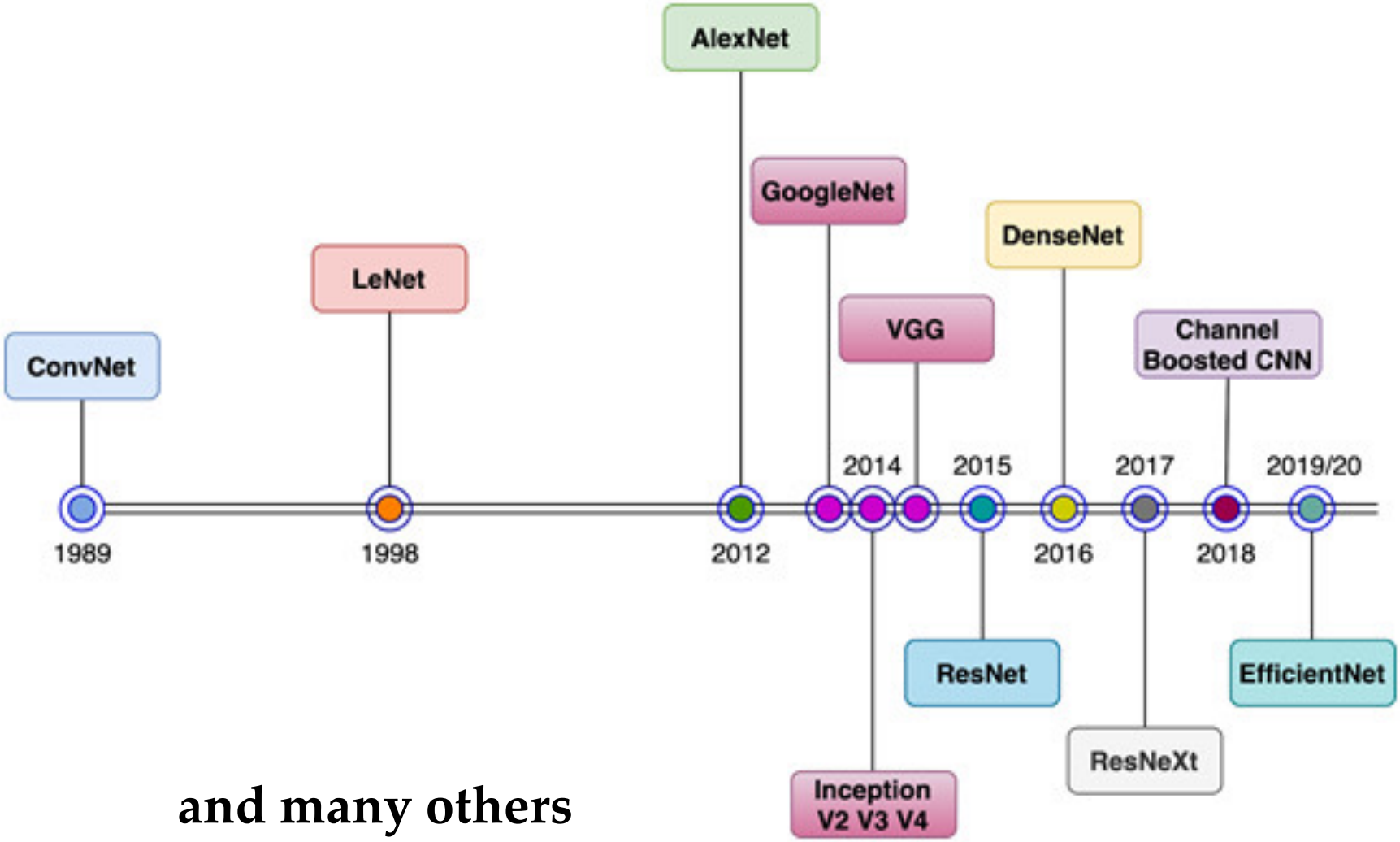
†Not plotted

EfficientNet-B7

- state-of-the-art 84.4% top-1 / 97.1% top-5 accuracy
- 8.4x smaller and 6.1x faster on inference than the best existing ConvNet.

Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." ICML 2019.

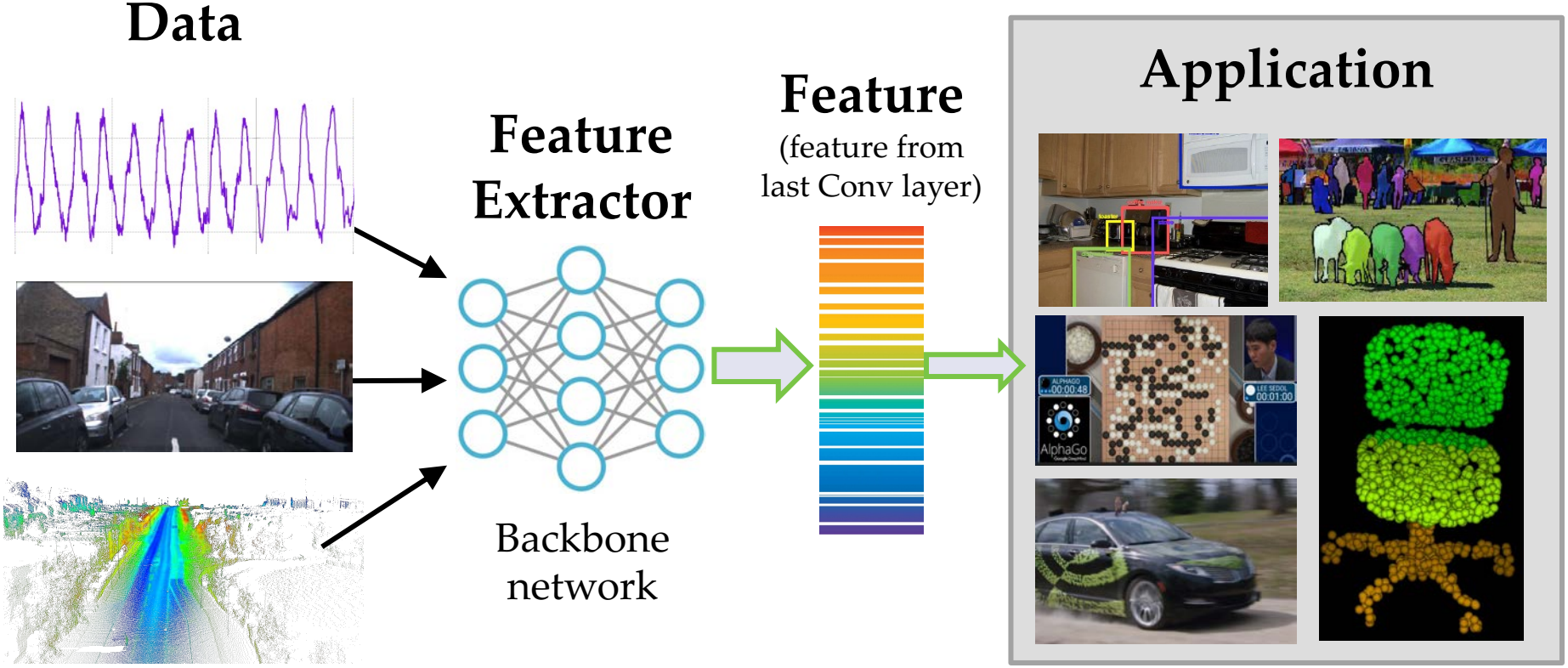
CNN Timeline



and many others

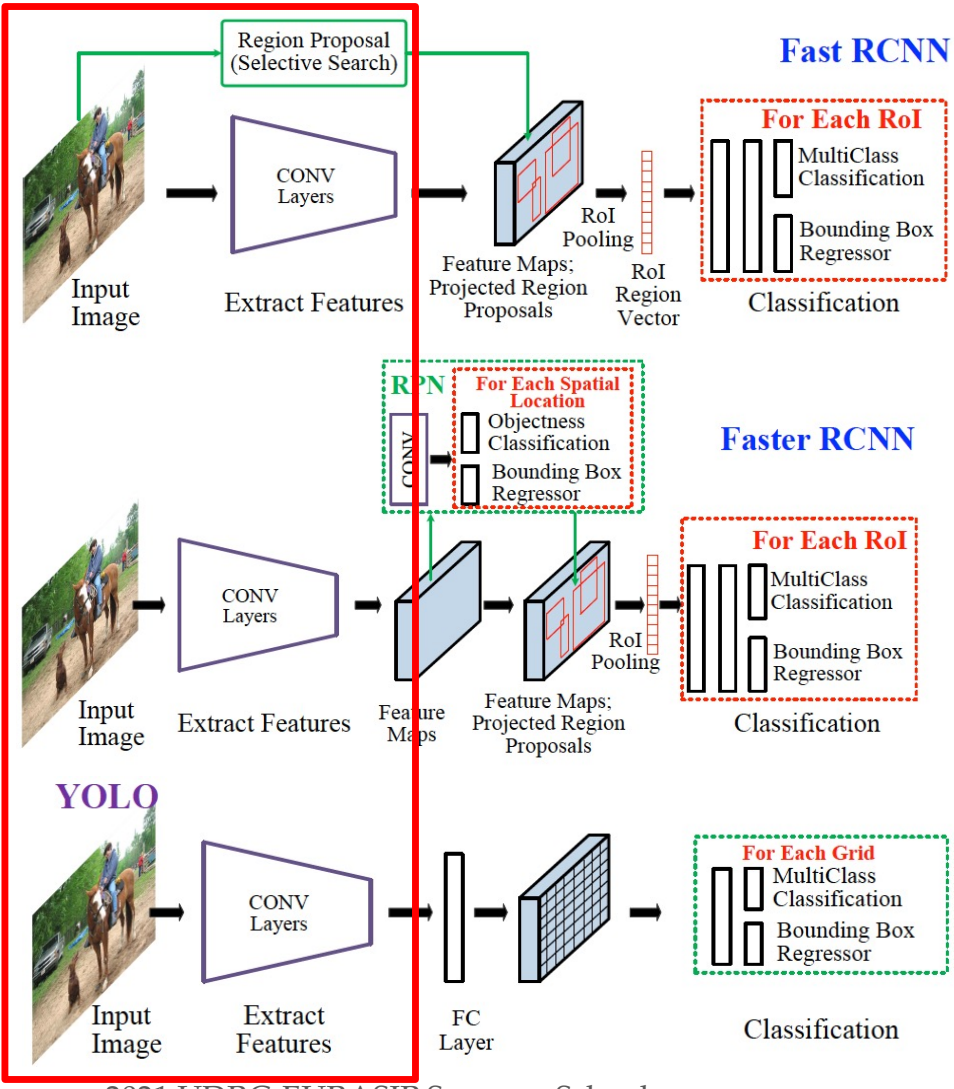
Deep Learning Applications

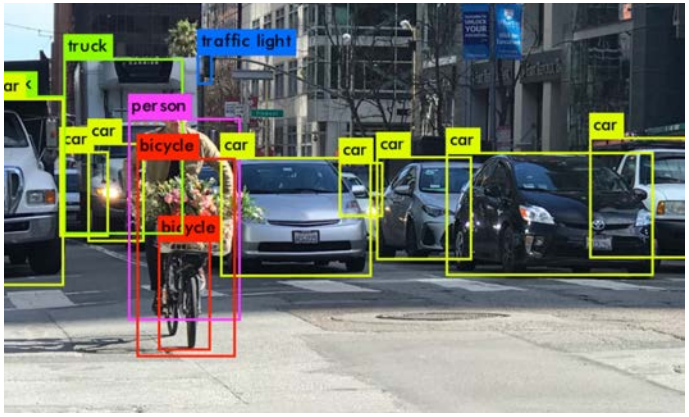
Deep Learning Applications

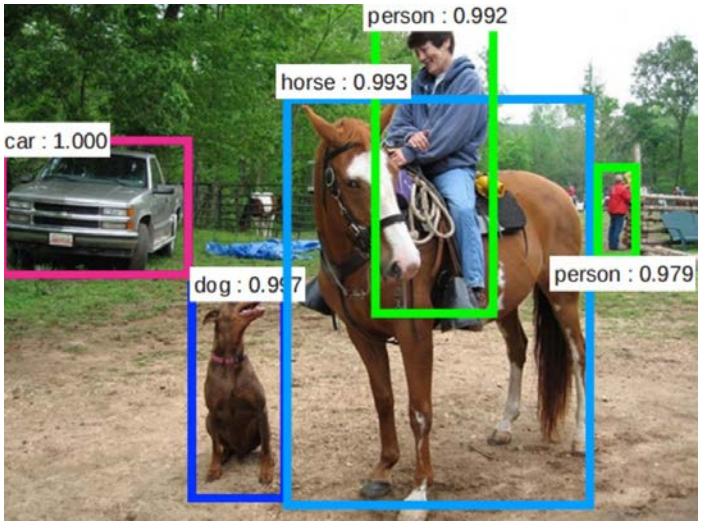


Object Detection and Recognition

Vision based: RCNN, Fast RCNN, Faster RCNN, YOLO, SSD,…..

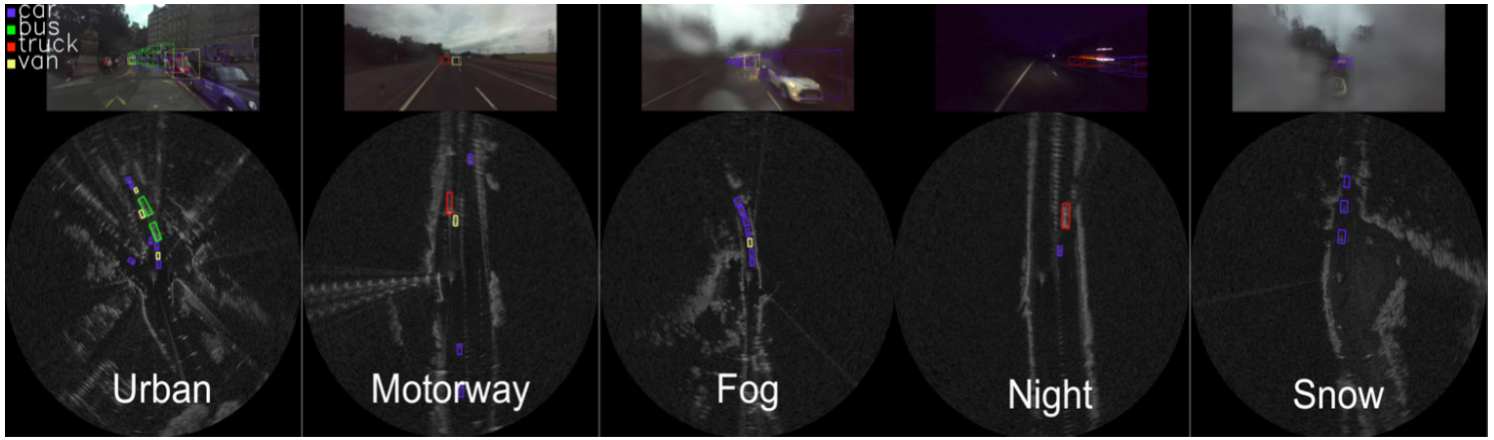




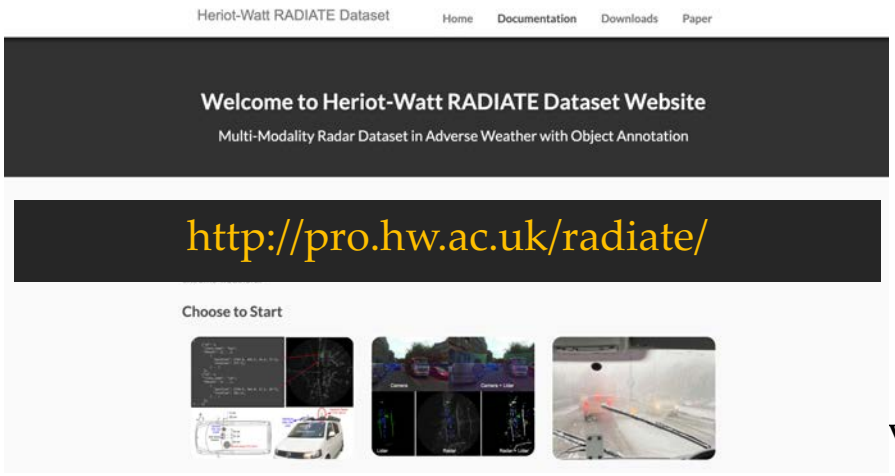


Object Detection and Recognition

Radar and polarized sensor based object detection and recognition



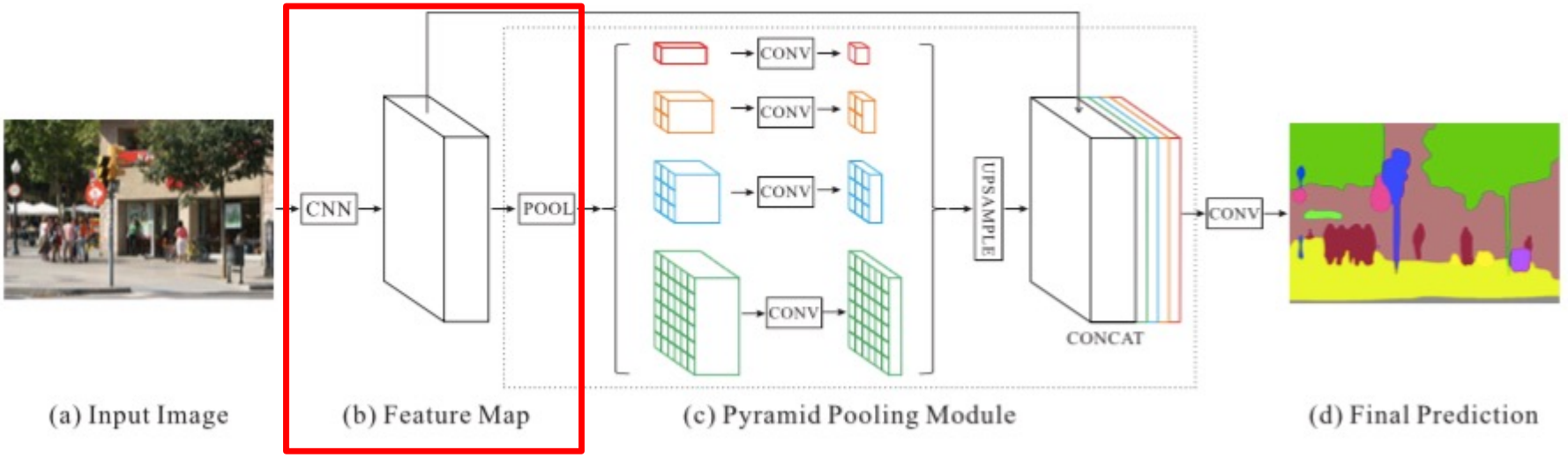
Vehicle detection and recognition using radar [Sheeny 2021]



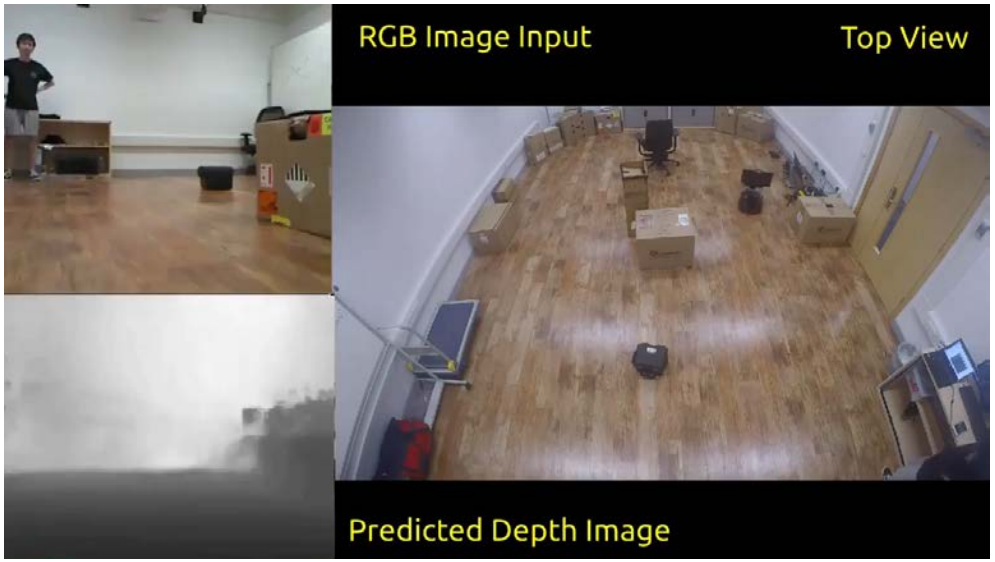
vehicle detection using polarised infrared sensors [Sheeny 2018]

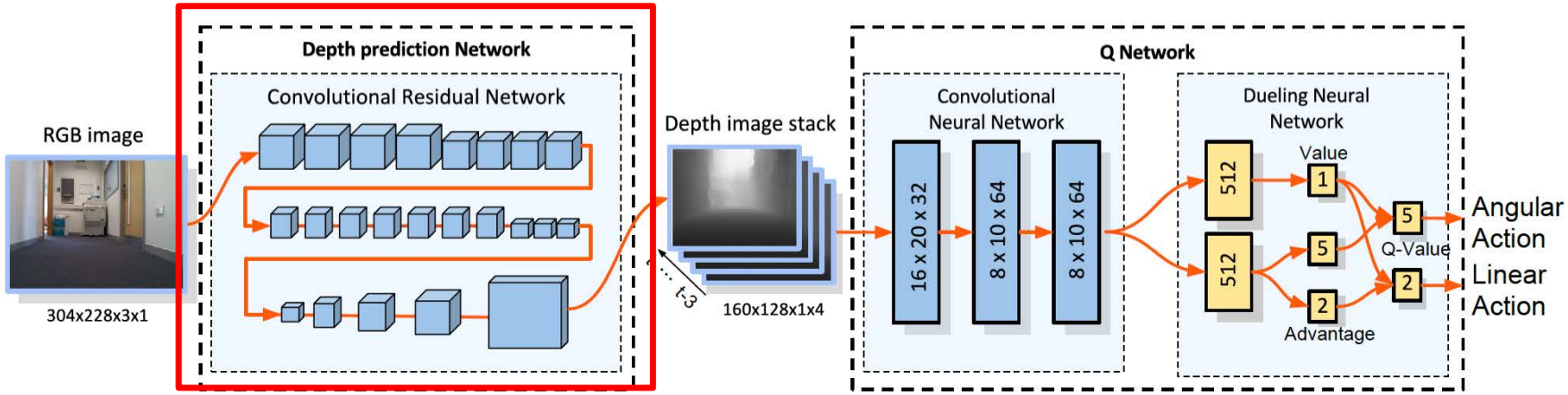
Semantic Segmentation

FCN, SegNet, RefineNet, PSPNet, …..



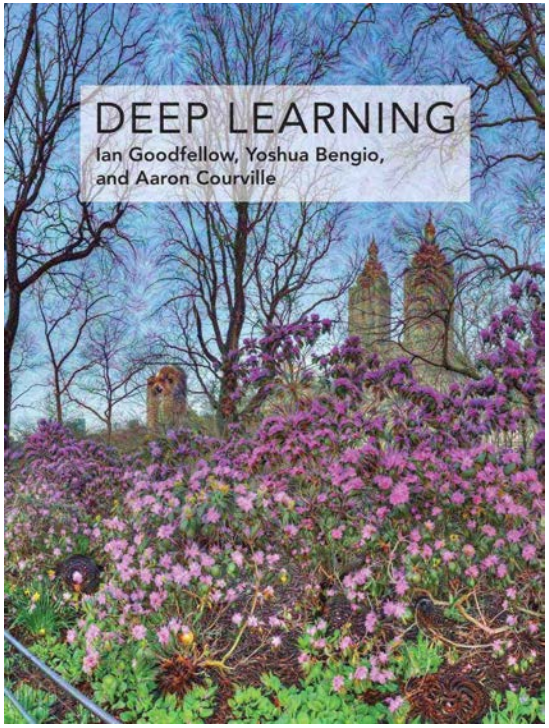
Robot Navigation





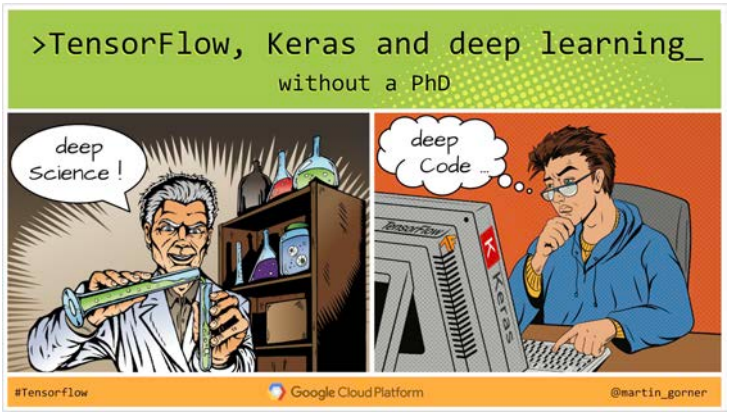
Extra Resource

Book



Online Codes and Practice

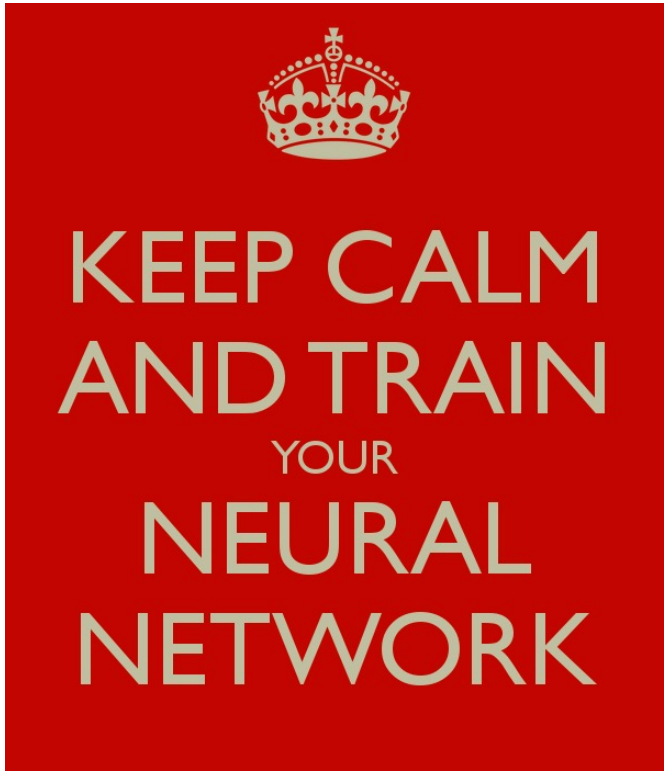
- https://codelabs.developers.google.com/
- TensorFlow, Keras and deep learning, without a PhD



- Build and deploy a custom object detection model with TensorFlow Lite (Android)

Summary

- Deep Learning is a powerful tool
- Leveraging pre-trained models for feature extraction whenever possible (similar modality with smaller dataset)
- **Learning representation** is the key for Deep Learning



HERIOT WATT UNIVERSITY

[dstl]

EPSRC
Engineering and Physical Sciences Research Council

Thank you for your attention!

Dr. Sen Wang
s.wang@hw.ac.uk