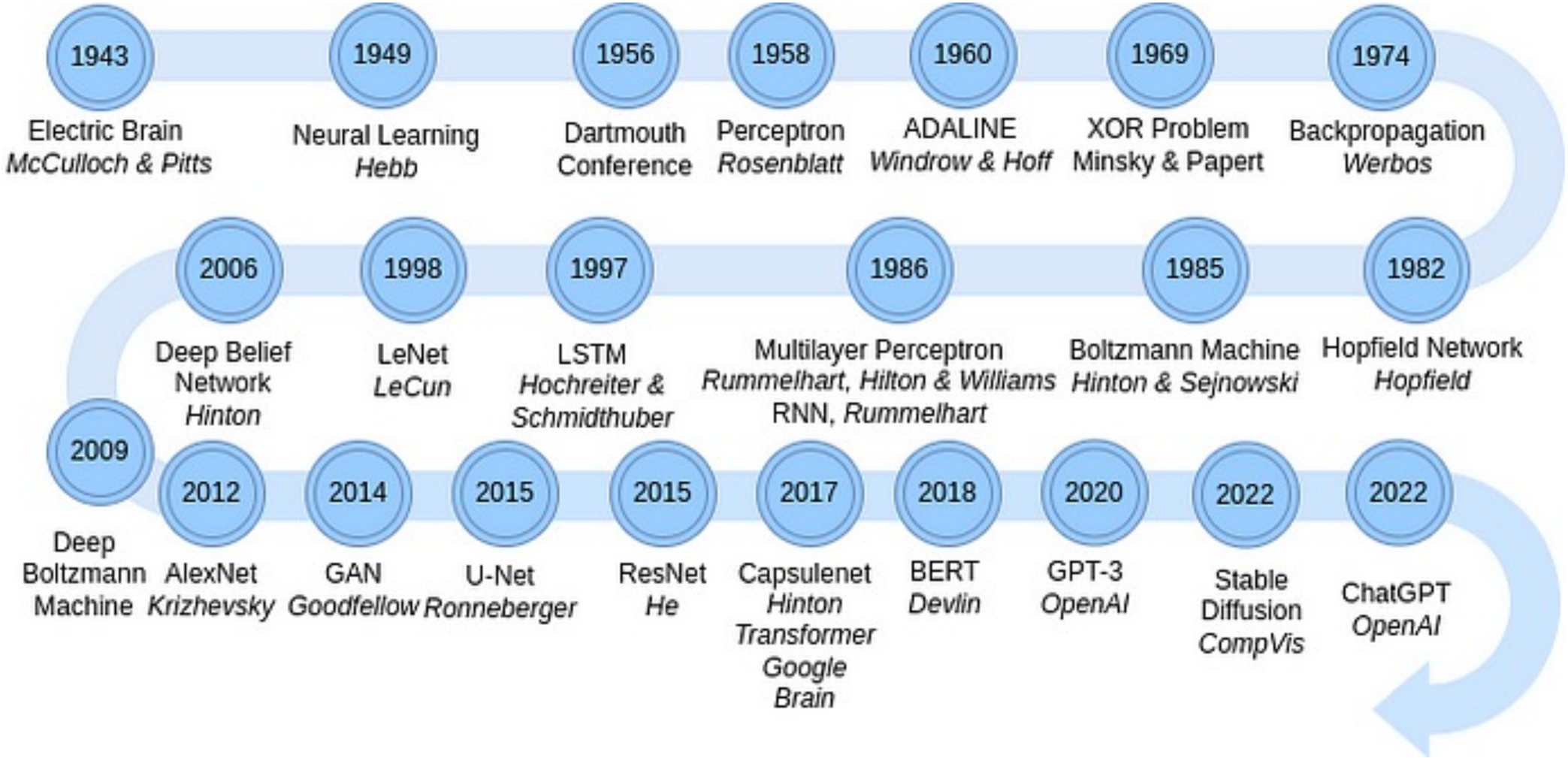
Deep Learning
Convolutional Neural Networks

CS 334: Machine Learning

Slides adapted from Jinho Choi, Stuart Russell, Fei-Fei Li, Andrej Karpathy, Justin Johnson, John Buillinaria, and Kyunghyun Cho

Deep learning: a brief history



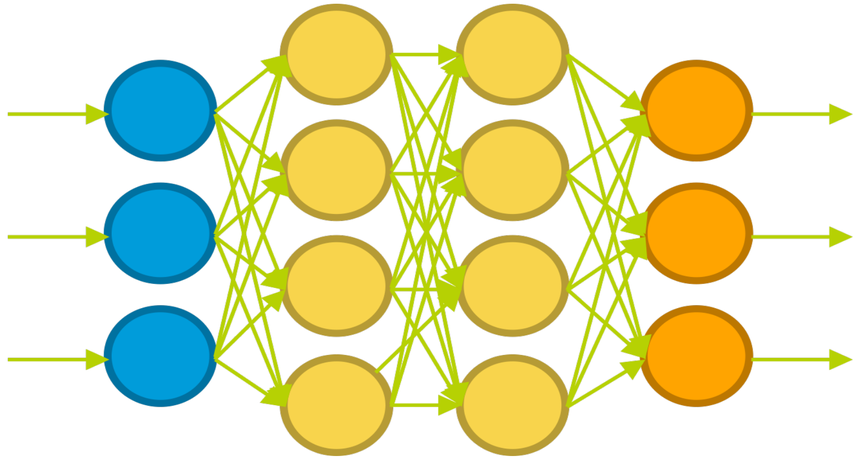
https://pub.towardsai.net/a-brief-history-of-neural-nets-472107bc2c9c

Deep learning

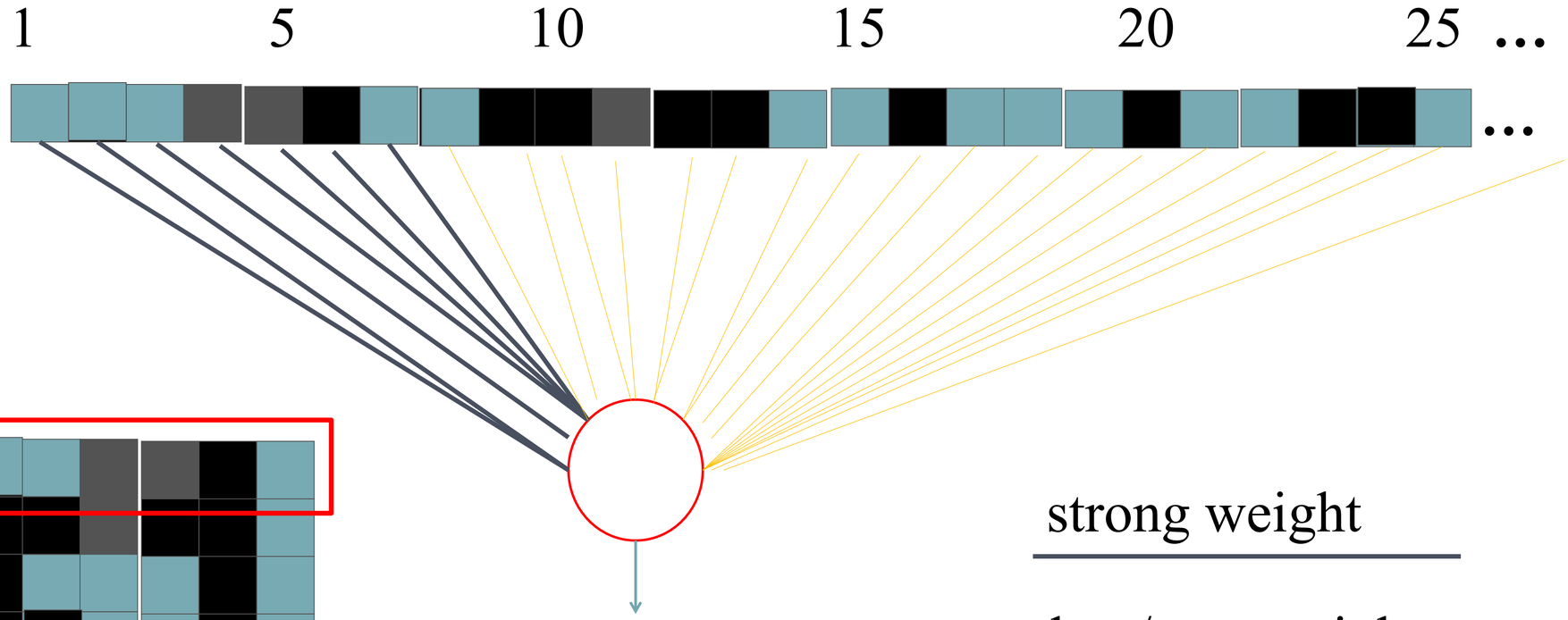
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Networks (CNN) – image data
- Recurrent Neural Networks (RNN) and Transformers – sequences

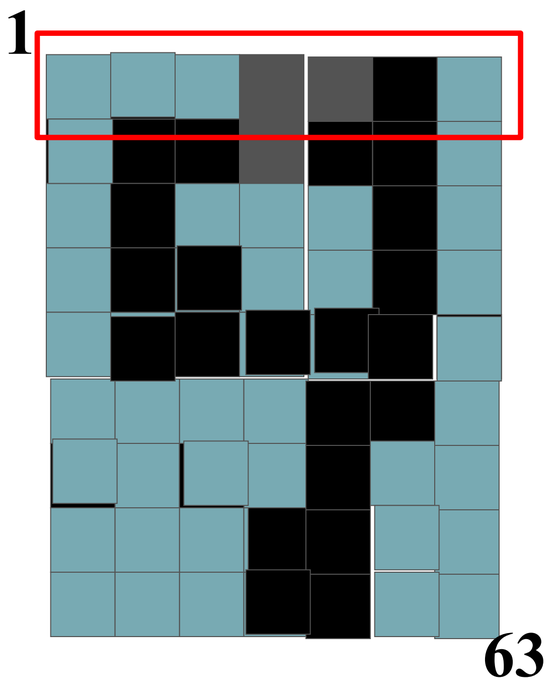
Review: Neural Networks

- Based on assumption that a computational architecture similar to brain would duplicate its abilities
- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Layering many neurons can create a complex model



REVIEW: WHAT DOES THIS UNIT DETECT?



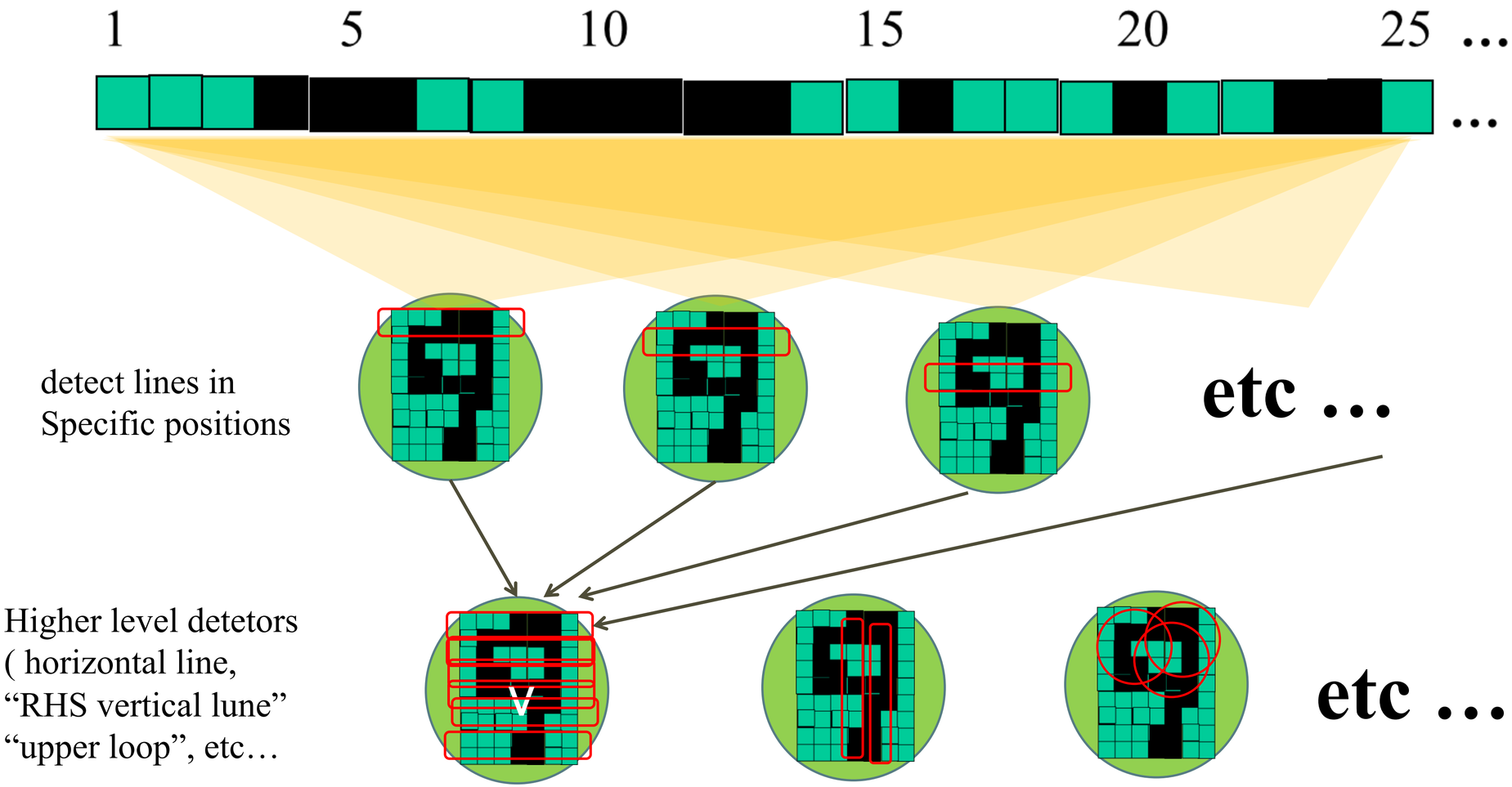


strong weight

low/zero weight

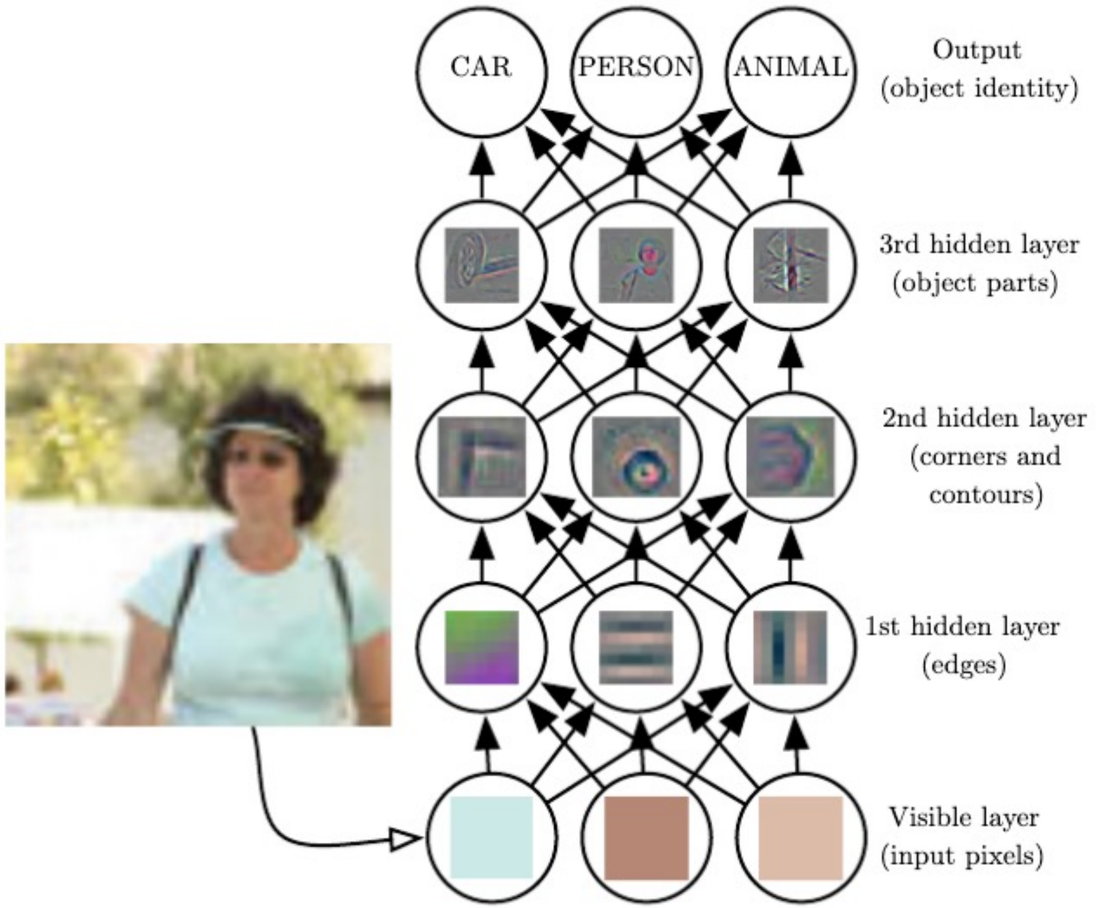
it will send strong signal for a horizontal line in the top row, ignoring everywhere else

REVIEW: HIERARCHICAL FEATURE LEARNING



Deep Learning: Overview

- Form of representation learning
- Aimed at learning feature hierarchies
- Features from higher levels of the hierarchy are formed by lower level features
- Each hidden layer allows for more complex features of input

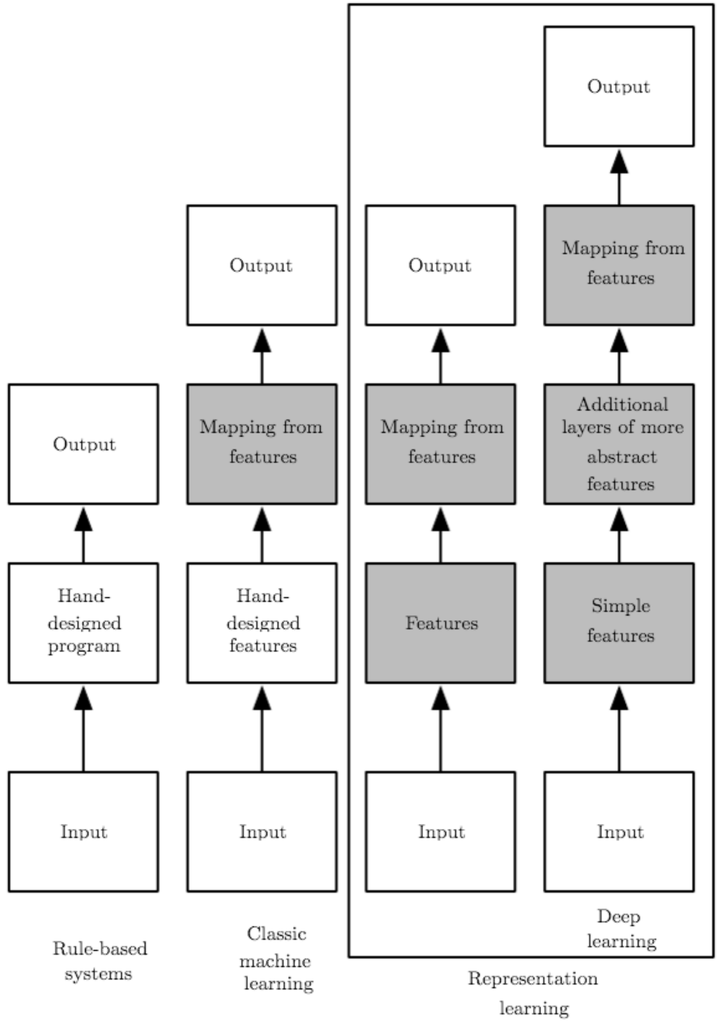


http://www.deeplearningbook.org/contents/intro.html
Figure 2

Deep Learning: The Promised Land

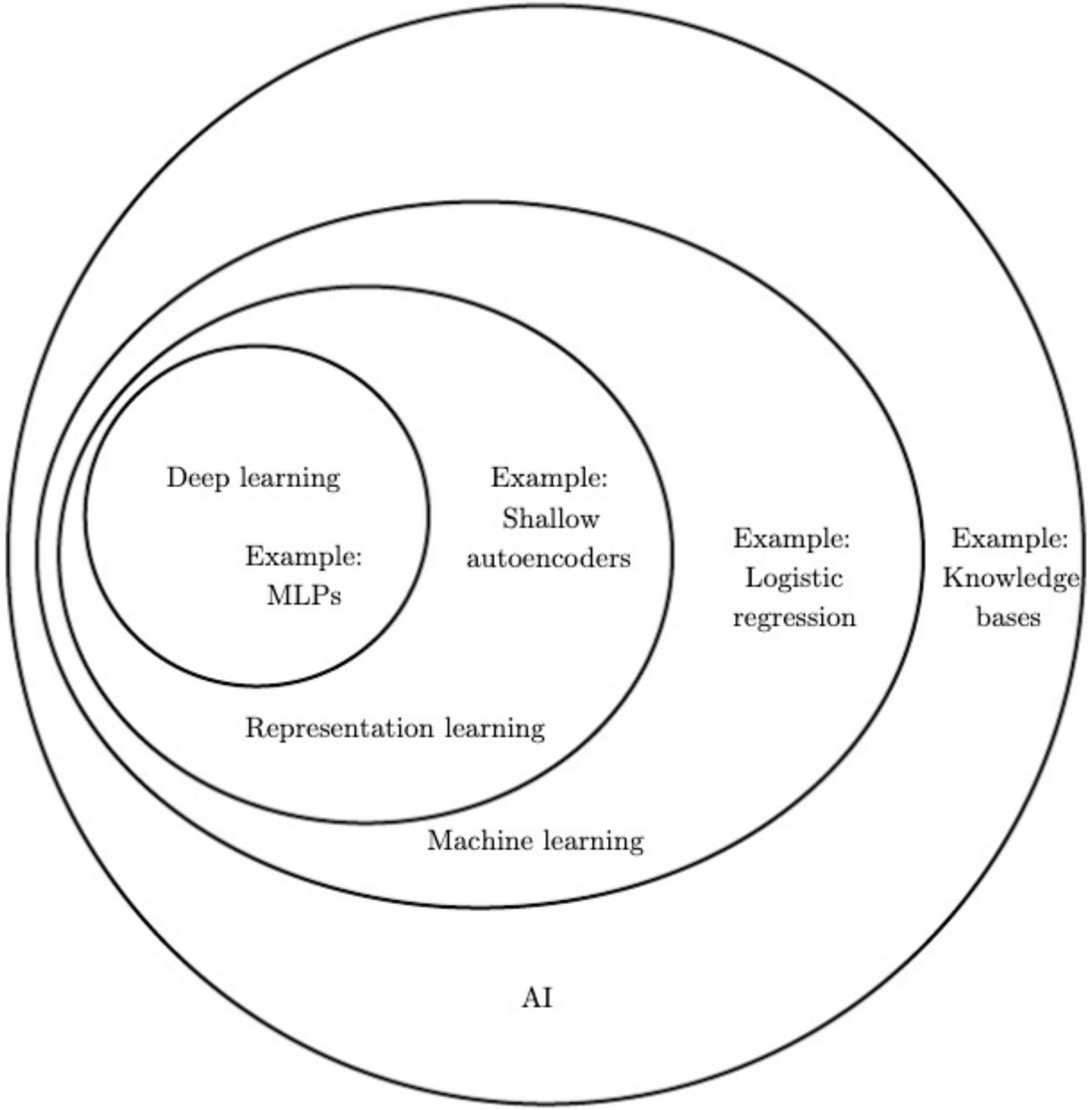
Automatic feature discovery

- Hidden layers discover semantically meaningful concepts
- Features learned without need for seeing exponentially large number of configuration of other features
- Expressiveness of deep networks



http://www.deeplearningbook.org/contents/intro.html

Deep learning

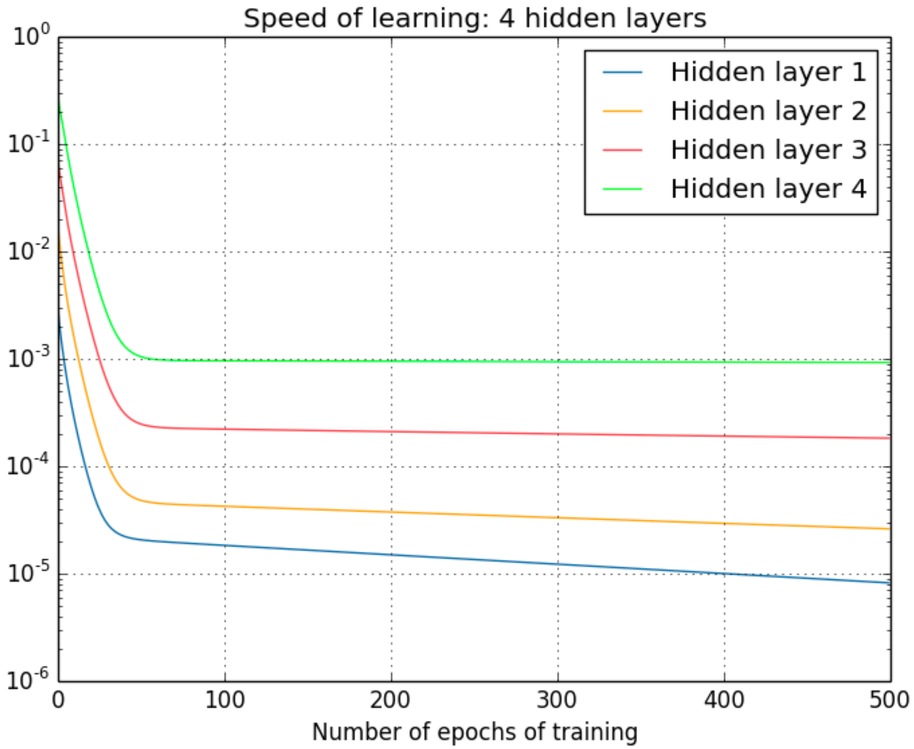


Deep learning

- Multi-Layer Perceptron (MLP)
- **Convolutional Neural Networks (CNN)**
- Recurrent Neural Networks (RNN)

Obstacles to Deep MLPs

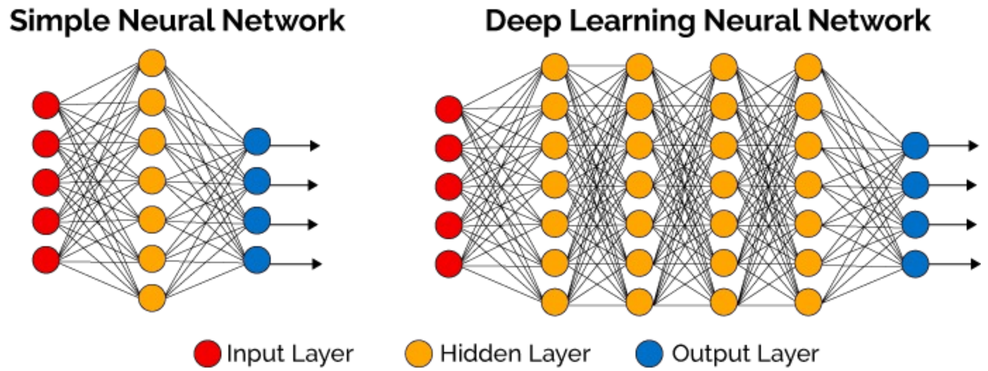
- Requires lots of labeled training data
- Computationally extremely expensive
  - Vanishing & unstable gradients
  - Training can be slow and get stuck in local minimum



http://neuralnetworksanddeeplearning.com/chap5.html

Obstacles to Deep MLPs

- Difficult to tune
  - Choice of architecture (layers + activation function)
  - Learning algorithm
  - Hyperparameters

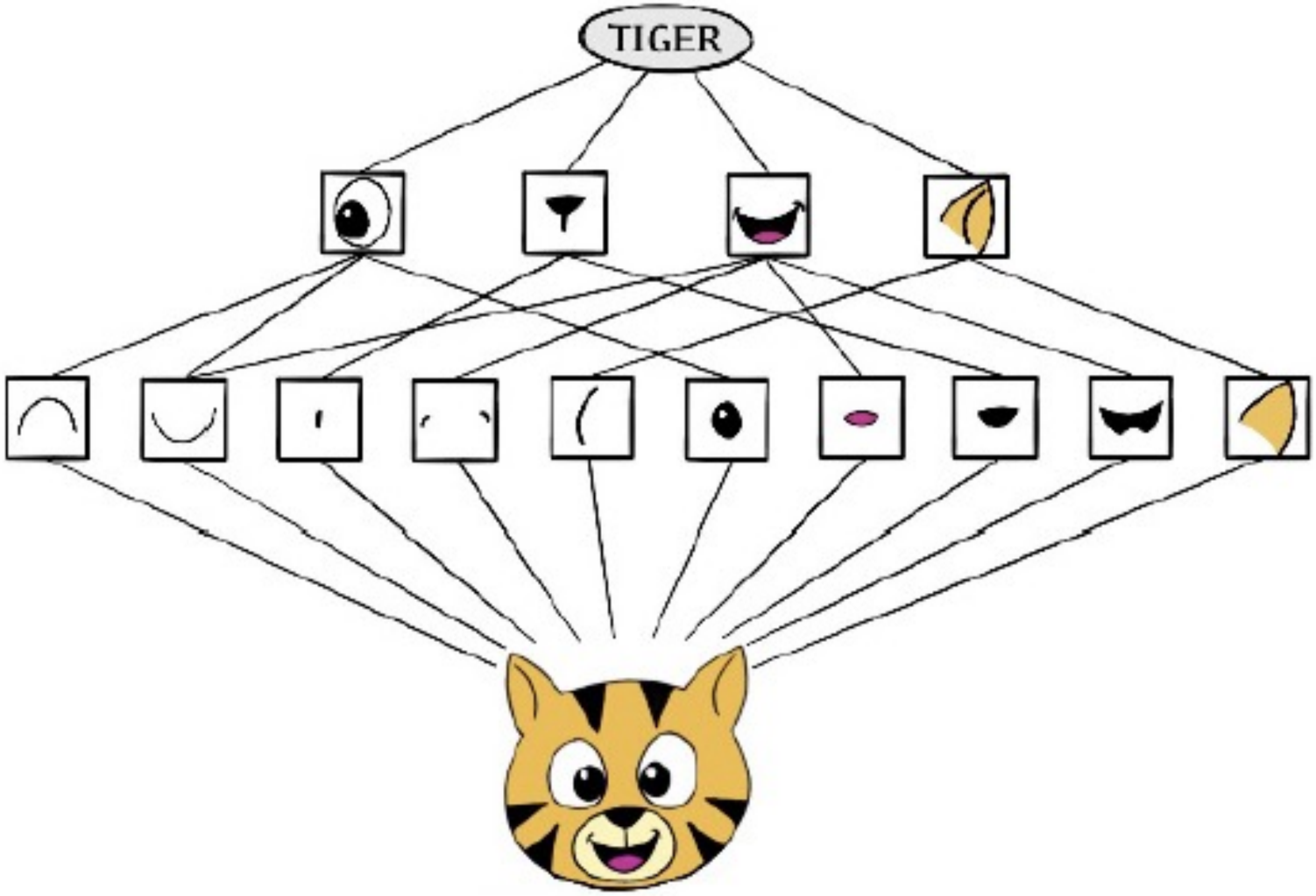


https://hackernoon.com/log-analytics-with-deep-learning-and-machine-learning-20a1891ff70e

Example: Images

- Fully connected network requires a vast number of parameters
- Think about a color image
  - 200 x 200 image x 3 color channels = 120,000 values
  - Single fully connected layer would require $(200 \times 200 \times 3)^2$ = 14,400,000,000 weights!
  - Bias-variance tradeoff?

Hierarchical Learning



Convolutional Neural Networks

- Specialized neural network for processing grid-like data, powerful model for image and language processing
- Convolutional layers
  - Learn small/local patterns
- Pooling layers
  - Select prominent patterns

CNN: Four Main Layers

- **Convolutional layer — output neurons that are connected to local regions in the input**
- ReLU layer — elementwise activation function
- **Pooling layer — perform a downsampling operation along the spatial dimensions**
- Fully-connected layer — same as regular neural networks

CNN: Example

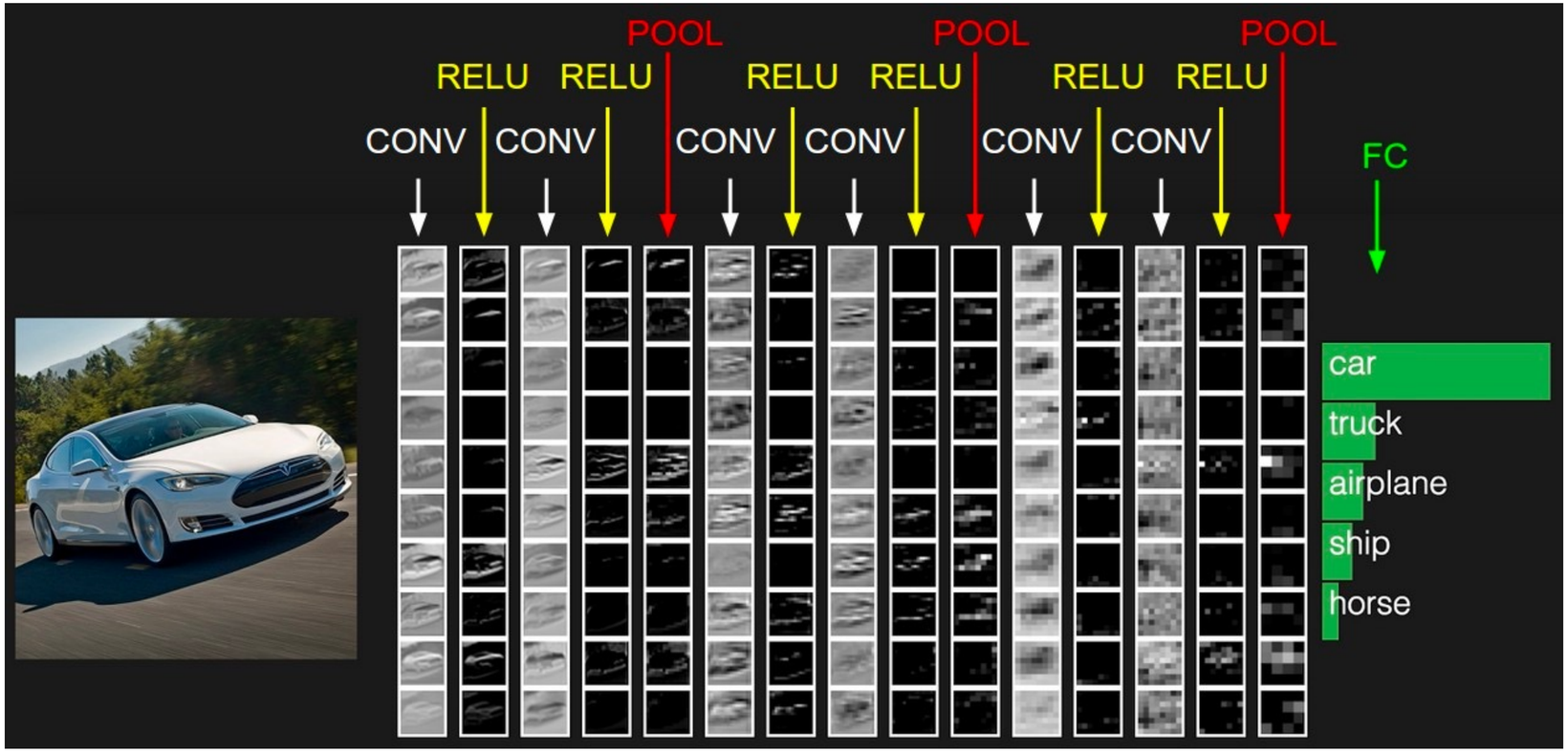

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Image Processing: Kernels

- Kernel (convolution matrix, mask) is a grid of weights “overlaid” on an image, centered on one pixel
- Each weight multiplied with pixel underneath it
- Output over centered pixel is weighted sum of the pixels

$$\sum_{p=1}^{P} w_p \text{pixel}_p$$

Example: 3x3 Kernel

Input

3	2	1
1	2	3
1	1	1

Kernel

-1	0	1
-2	0	2
-1	0	1

Output

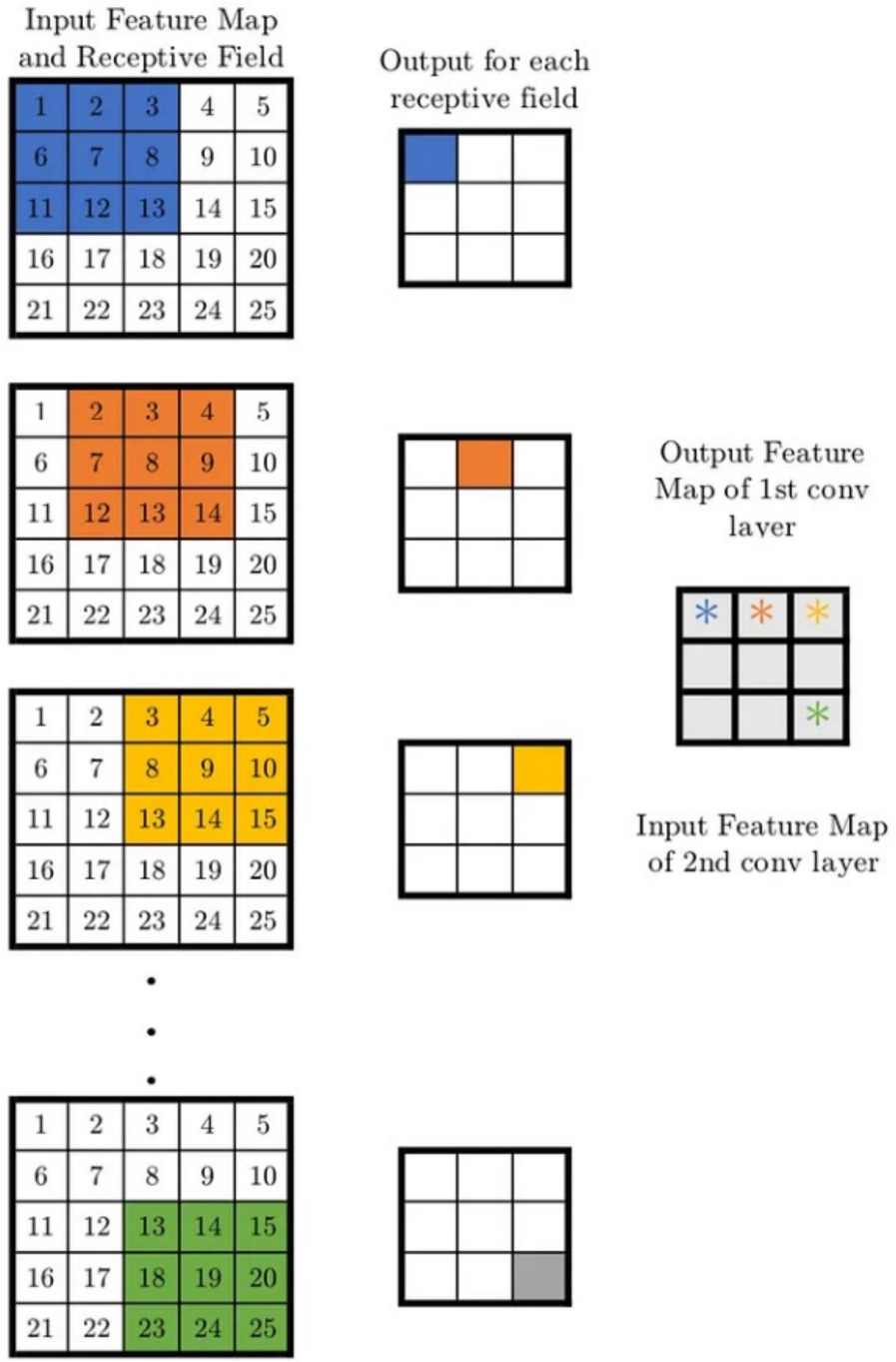
	2	

$$= (3 \cdot -1) + (2 \cdot 0) + (1 \cdot 1)$$
$$+(1 \cdot -2) + (2 \cdot 0) + (3 \cdot 2)$$
$$+(1 \cdot -1) + (1 \cdot 0) + (1 \cdot 1)$$

$$= -3 + 1 - 2 + 6 - 1 + 1 = 2$$

Convolution Filter

- Slide over image to compute an output for each 3*3 region



Convolution Filter

$$\text{Original Image} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$$

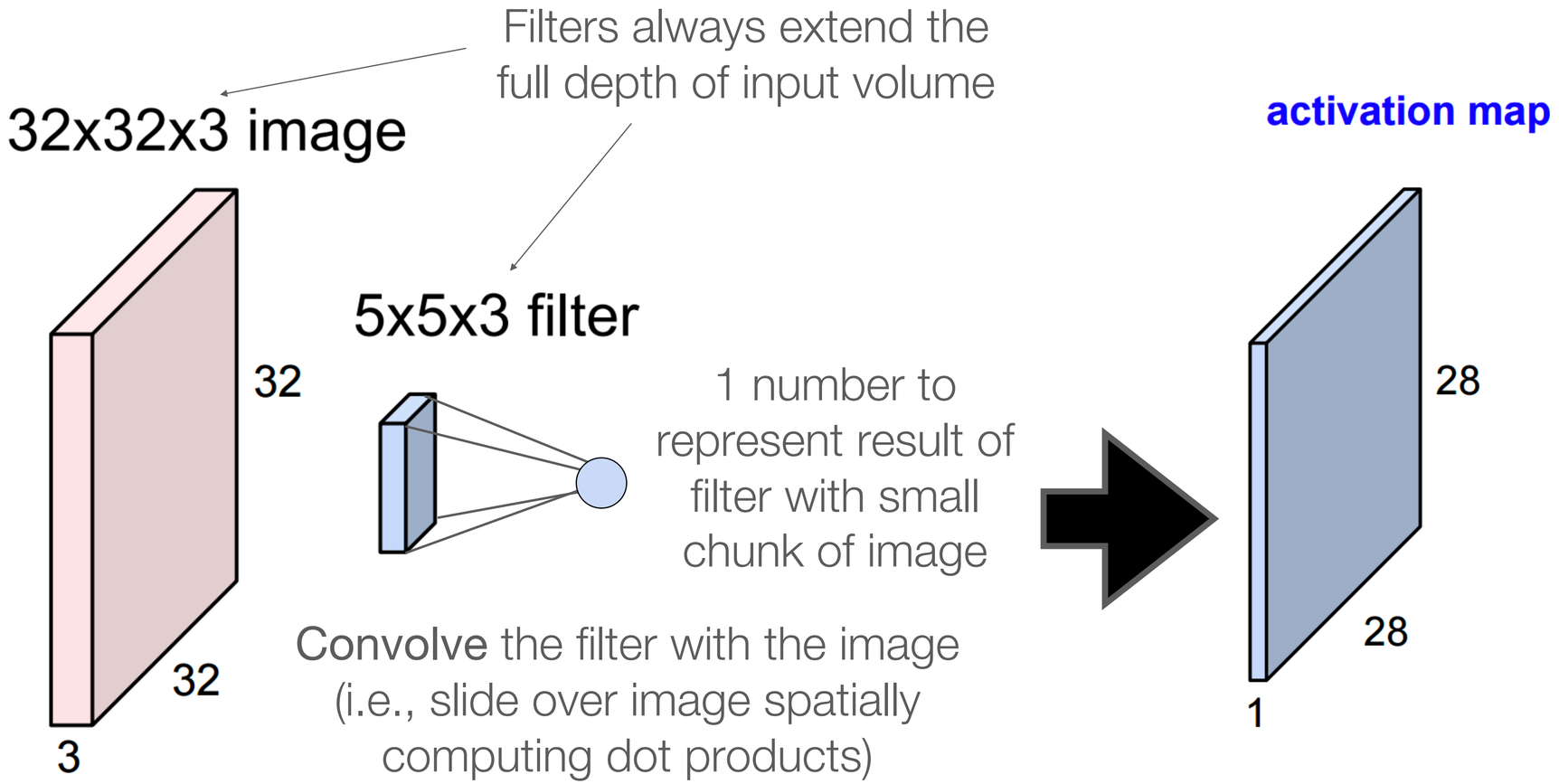
$$\text{Convolution Filter} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

$$\text{Convolved Image} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Convolution Settings: Depth

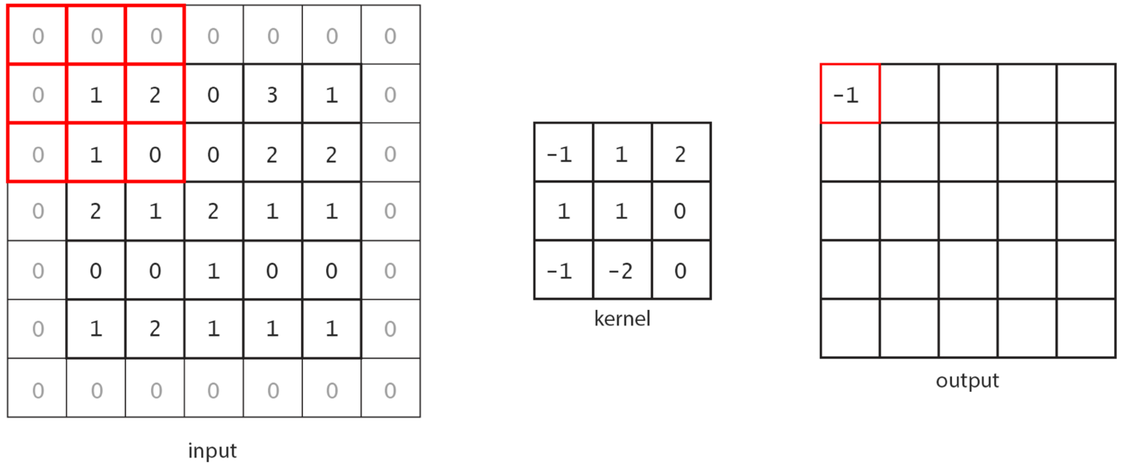
- Depth: Number of channels associated with each pixel location (# of filters)
- Kernel itself will have a depth of the same size as the input channels
  - RGB image — 3 channels
  - CMYK image — 4 channels
- Ex: 5x5 kernel on RGB image corresponds to 75 weights (5*5*3)

CNN: Convolution Layer



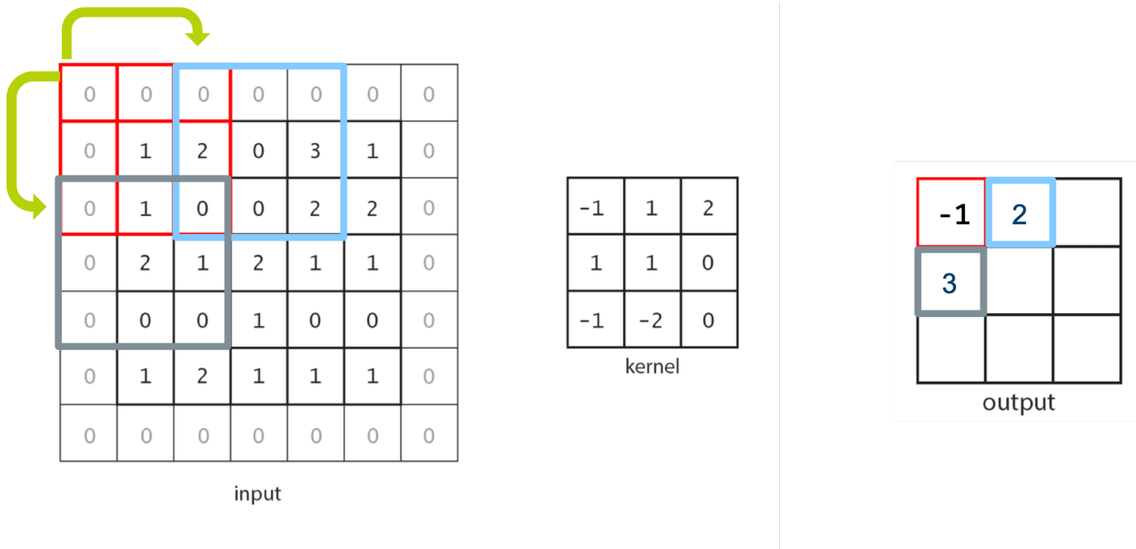
Convolution Settings: Padding

- Padding: Adds extra pixels around the frame
  - Takes care of the edges so they will be used as “center” pixels w/ kernels
  - Typically zero-padding

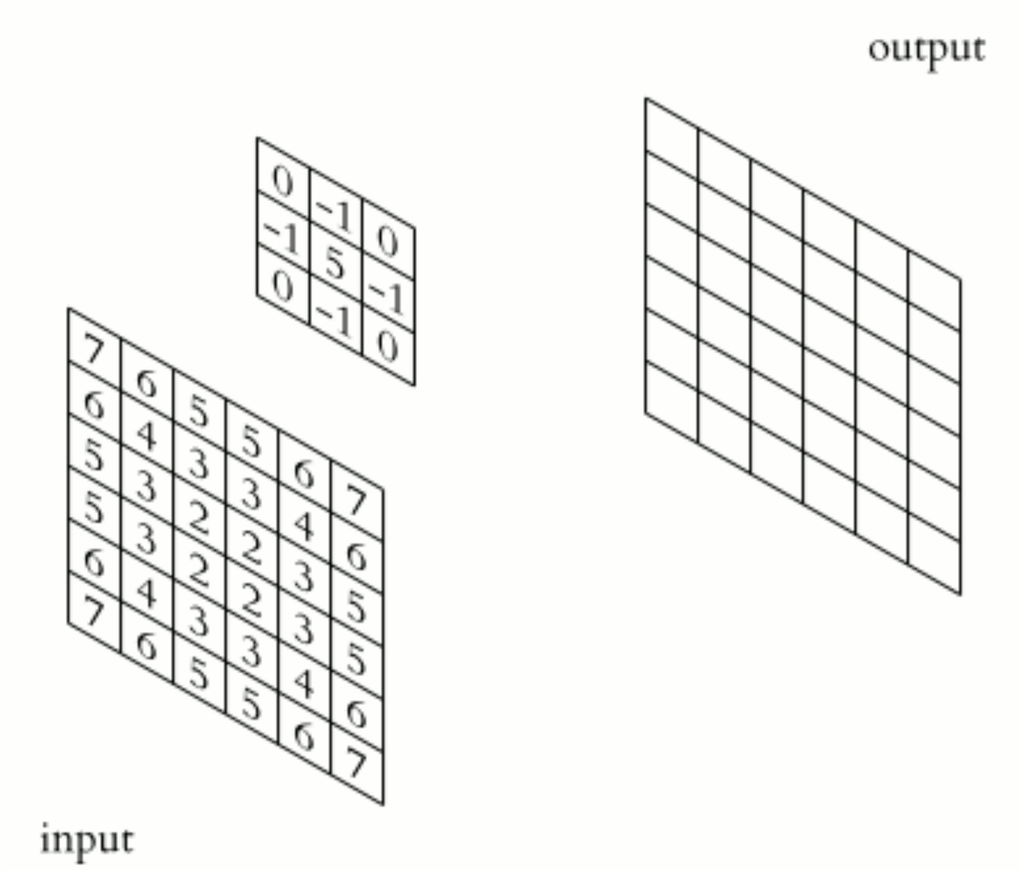


Convolution Settings: Stride

- Stride: Step size for the kernel
  - Specifies vertical and horizontal steps
  - Stride > 1 scales down the output dimension



Convolution



Multiple Kernels as Feature Detectors

Vertical Line Detector

-1	1	-1
-1	1	-1
-1	1	-1

Horizontal Line Detector

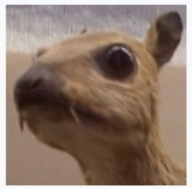
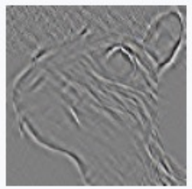
-1	-1	-1
1	1	1
-1	-1	-1

Corner Detector

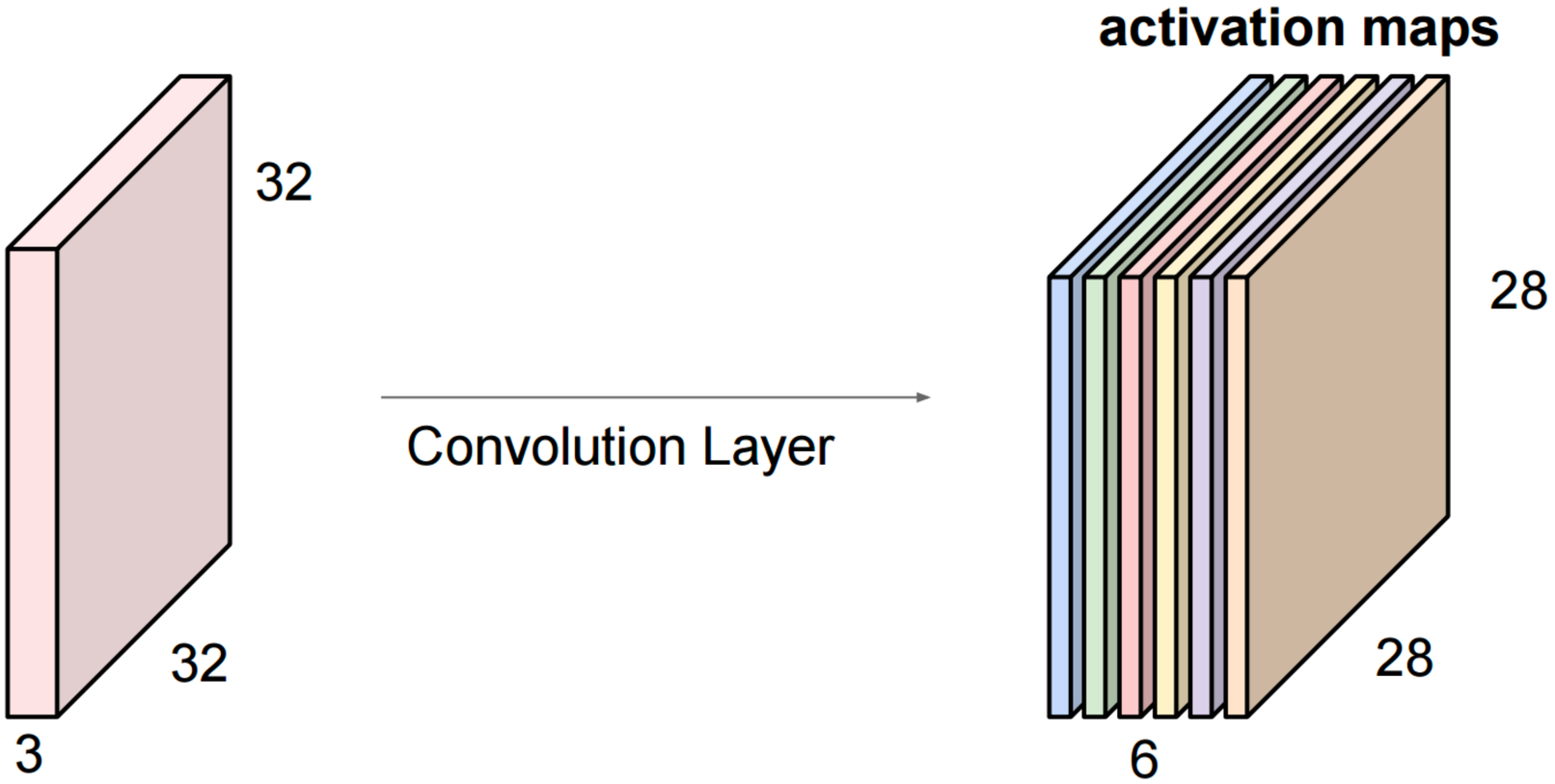
-1	-1	-1
-1	1	1
-1	1	1

Think of them as “local feature detectors”

Kernels

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

CNN: Convolution Layer



Stacking up multiple filters yields “new image”

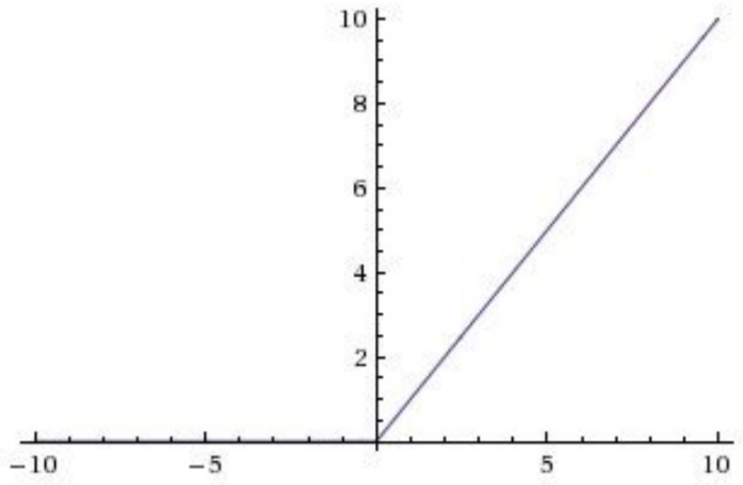
CNN: Convolution Filters

- Filters act as feature detectors from original image
- Network will learn filters that activate when they see some type of visual feature (e.g., edge of some orientation, blotch of some color, etc.)
- Only need to learn the weights of the filters

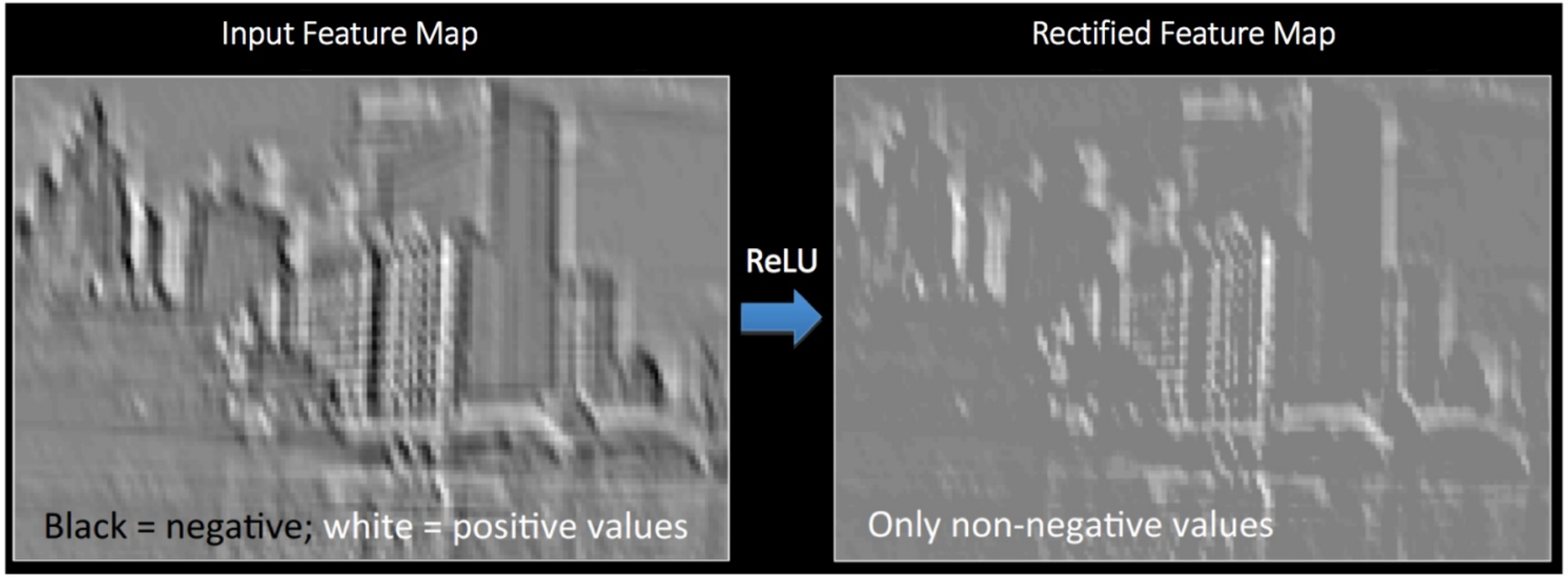
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Review: ReLU

- Does not saturate in positive region
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g., 6x)



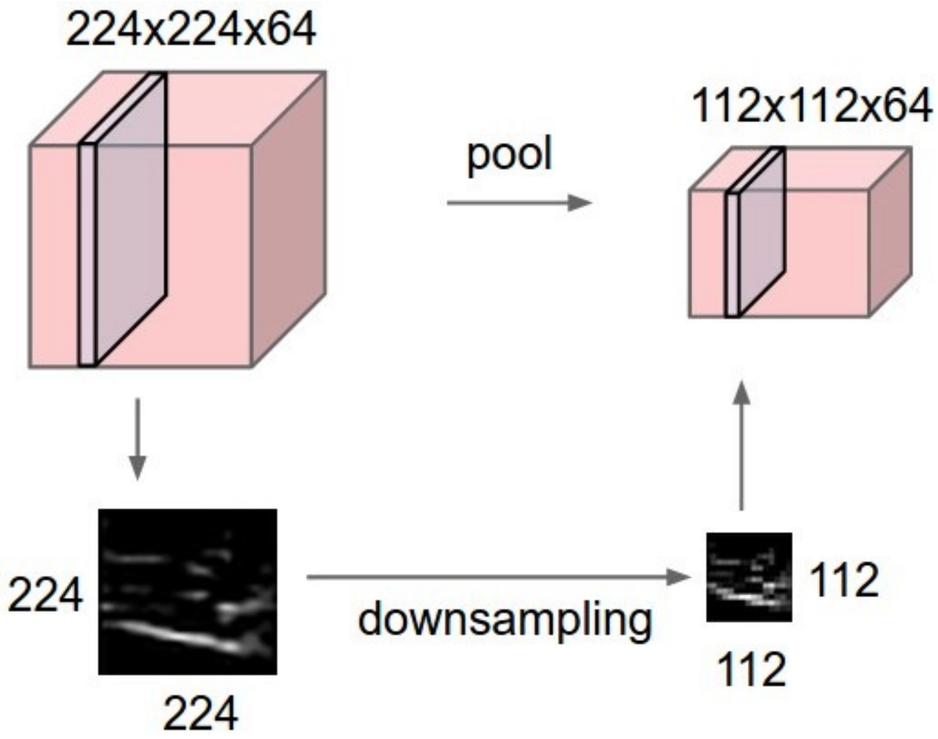
CNN: ReLU



Used after every convolution operation and introduces non-linearity

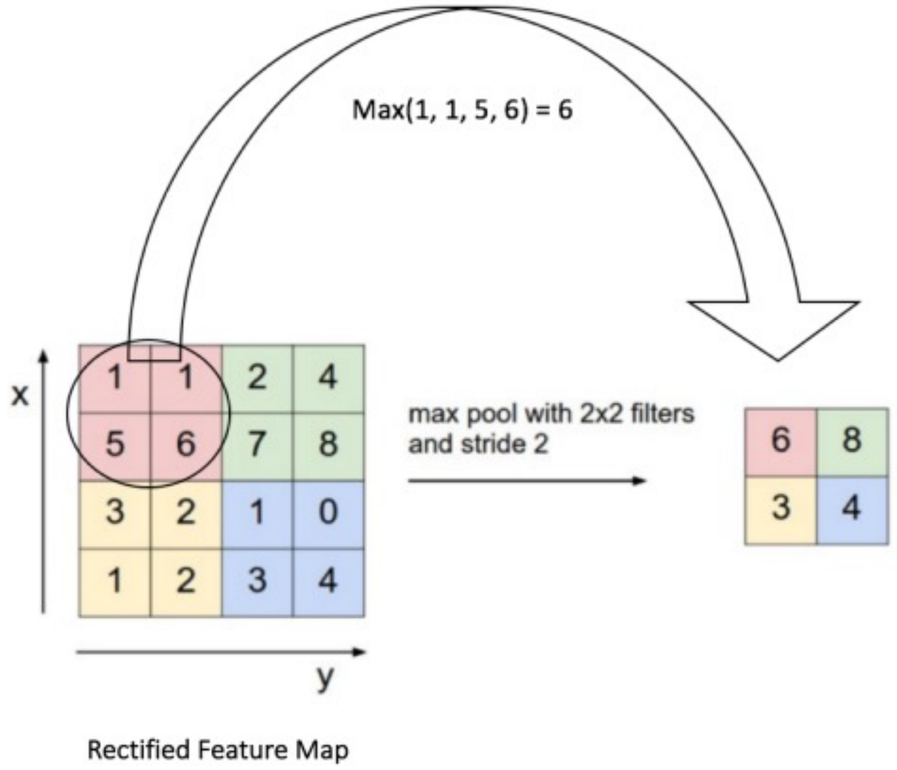
CNN: Pooling Layer

- Make representations smaller & more manageable
- Helps control overfitting
- Operates over each activation map independently
- Does not have parameters

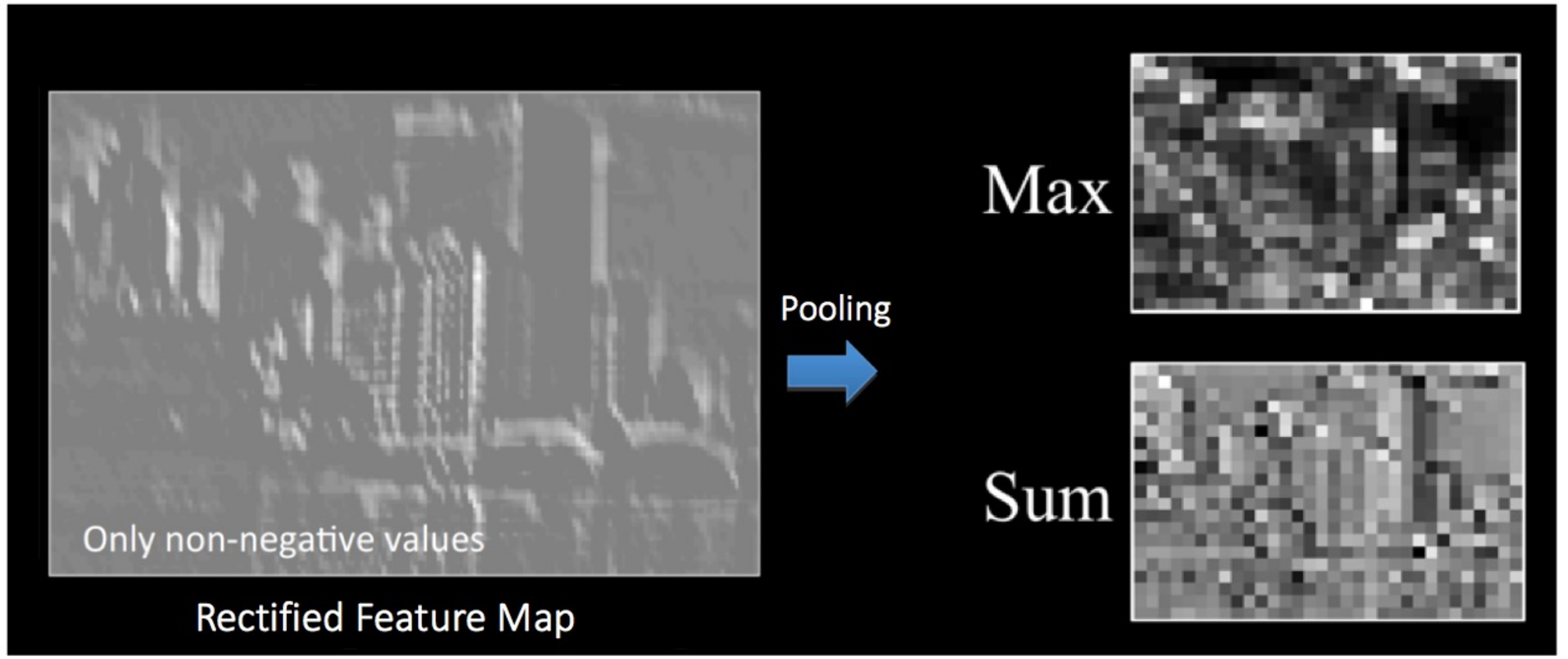


Pooling: Max-Pool

- For each distinct patch, represent it by the maximum (or average for average pooling)



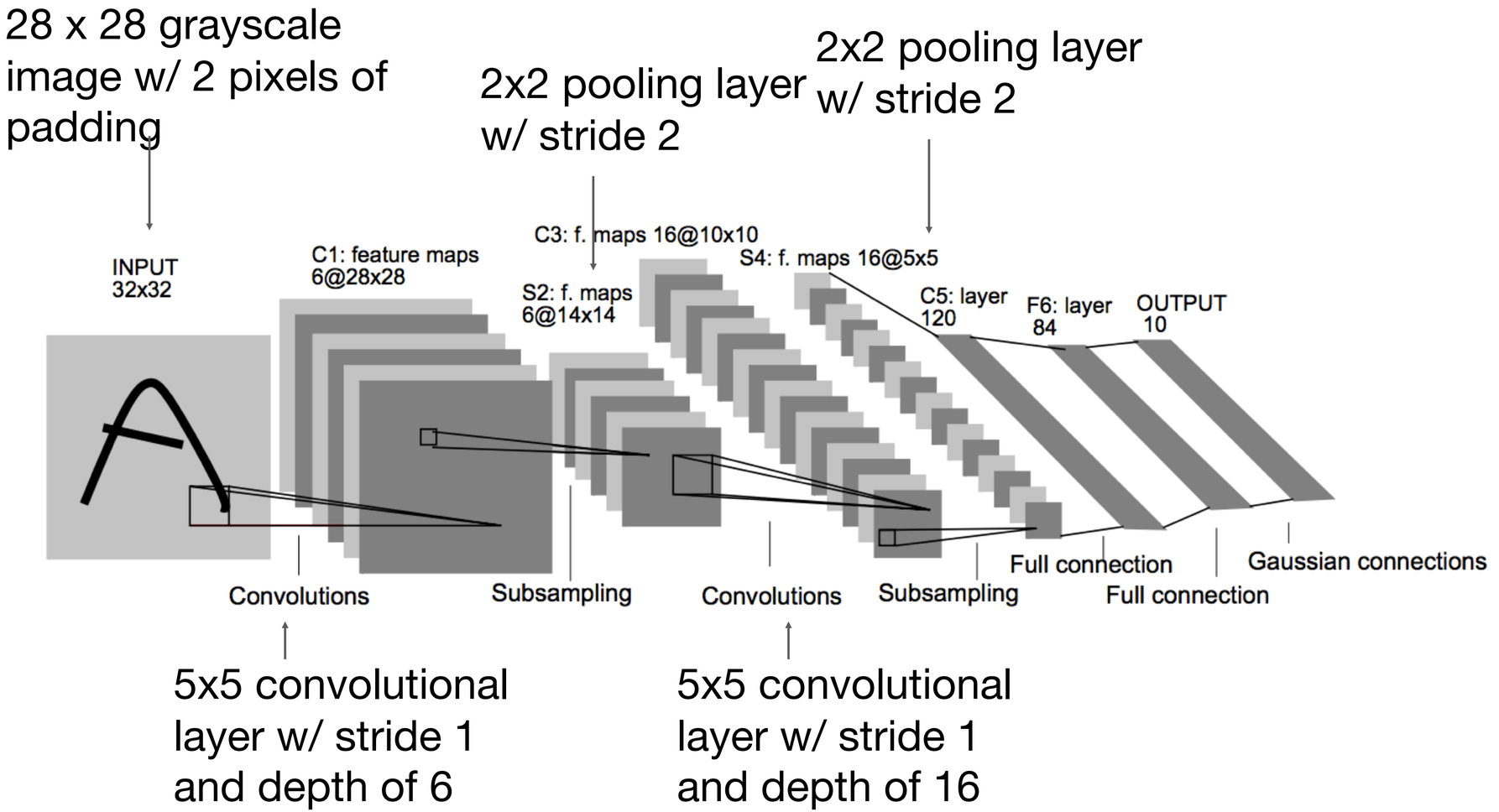
CNN: Pooling Example



CNN: Fully-Connected Layer

- Traditional MLP using softmax activation function
  - Generalization of logistic function to multi-class problem
  - Output probabilities for each class that sum to 1
- Output of convolutional and pooling layers represent high-level features

LeNet 5 [LeCun et al., 1998]



LeNet-5: Number of weights?

- C1: 6*1* 5*5 + 6 = 156
- C3: 16*6*5*5 + 16 = 2416
- C5: 120*16*5*5 + 120 = 48120
- FC1: 120*84 + 84 = 10164
- FC2: 84*10 + 10 = 850
- Total: = 61706

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

Less than a single FC layer with 80 neurons: (28 x 28) * 80 weights!

LeNet: MNIST Dataset Results

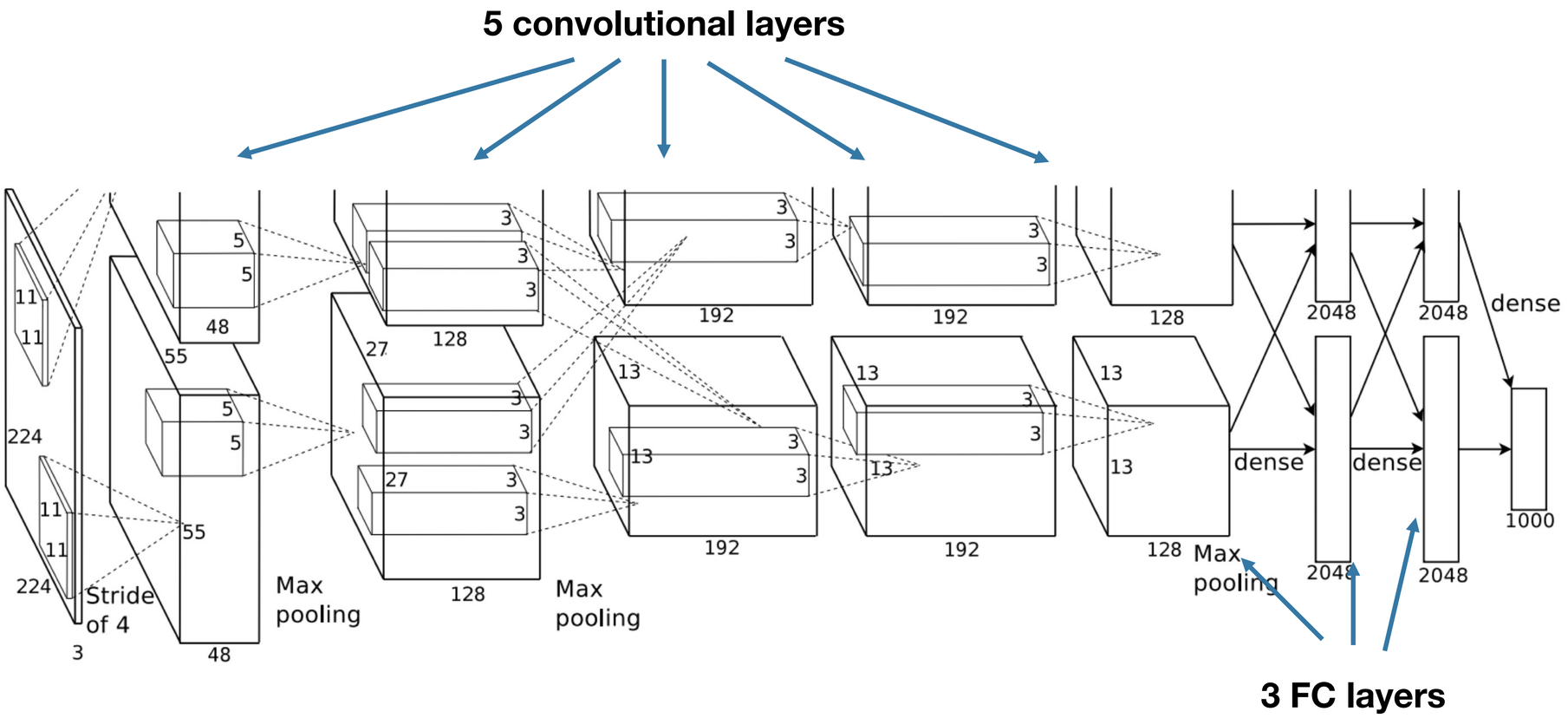
- Original dataset
  (60,000 images)
  - Test error = 0.95%
- Distorted dataset
  (540,000 artificial distortions + 60,000 images)
  - Test error = 0.8%

Misclassified examples

AlexNet

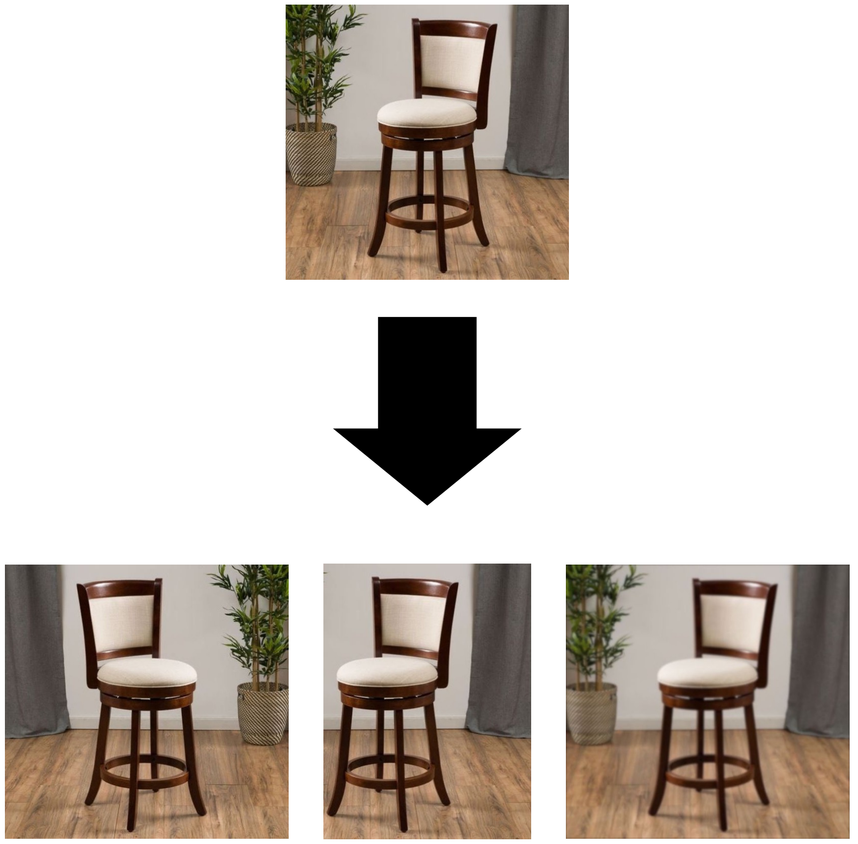
- Created in 2012 for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- Task: Predict correct label from among **1000** classes
- Dataset: ~1.2M images
- “Flash point” for modern ML
- Destroyed the competition — top 5 error rate of 15.4%, next best was 26.2%

AlexNet: Diagram

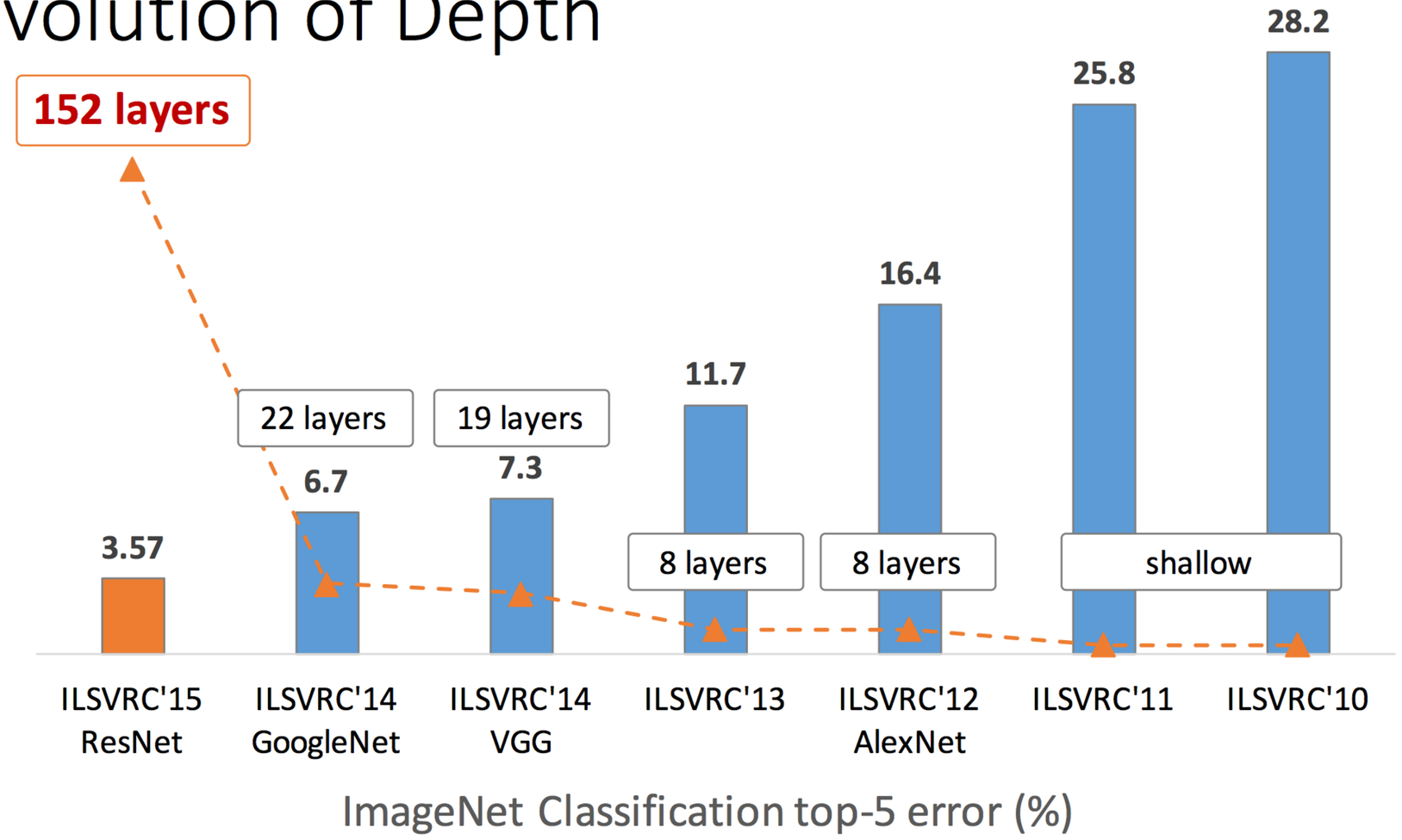


AlexNet: Details

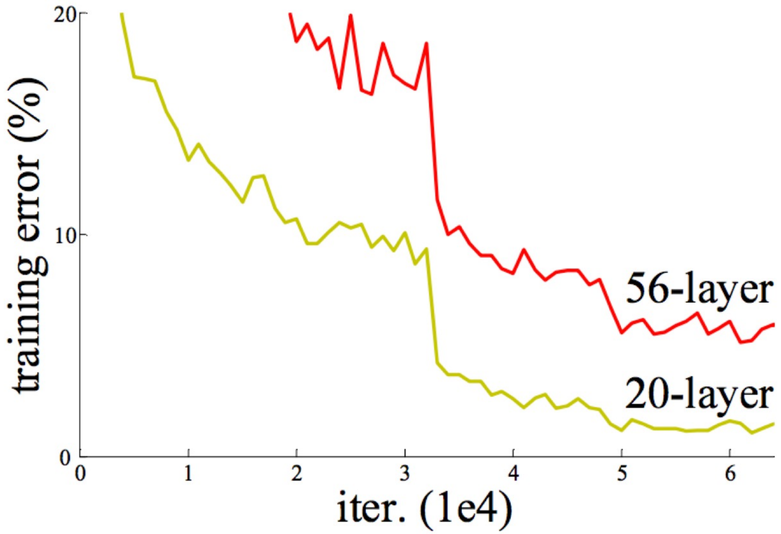
- Data augmentation for training — cropping, horizontal flipping, & other manipulations
- Basic template
  - Convolutions with ReLUs
  - Maxpool after convolutional layer
  - FCs at end

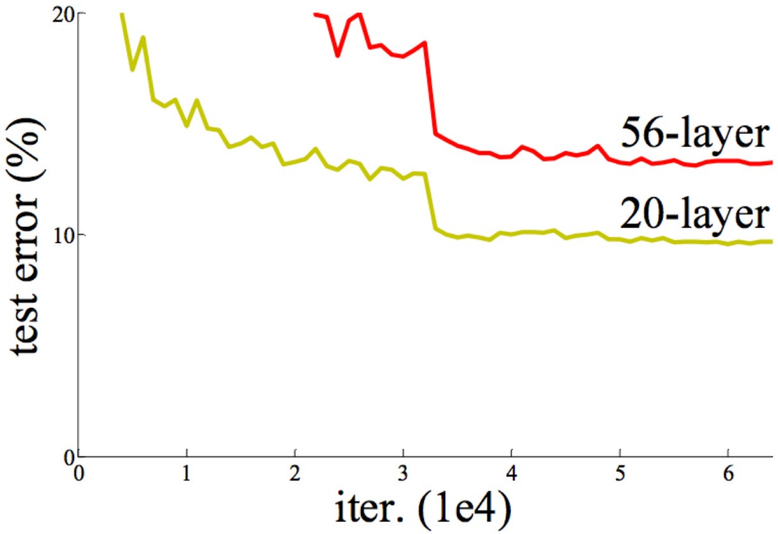


Revolution of Depth



ResNet: Motivation



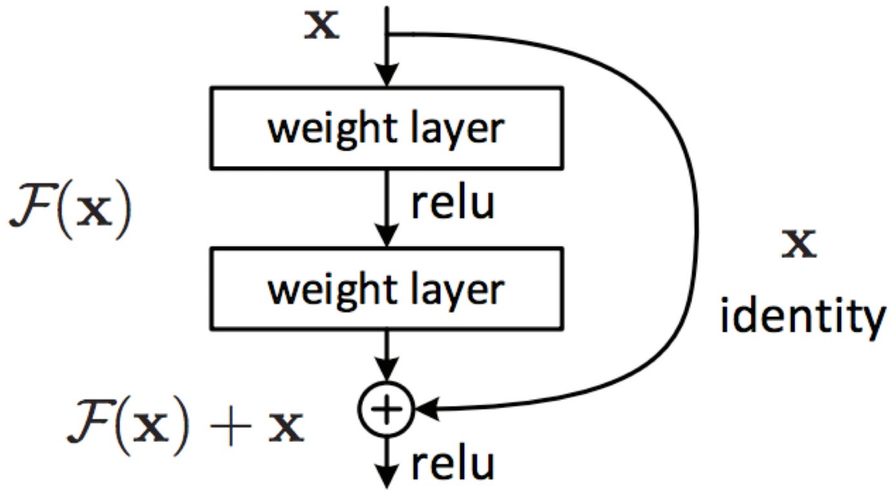


Shouldn't deeper networks perform better on training and test?!

https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

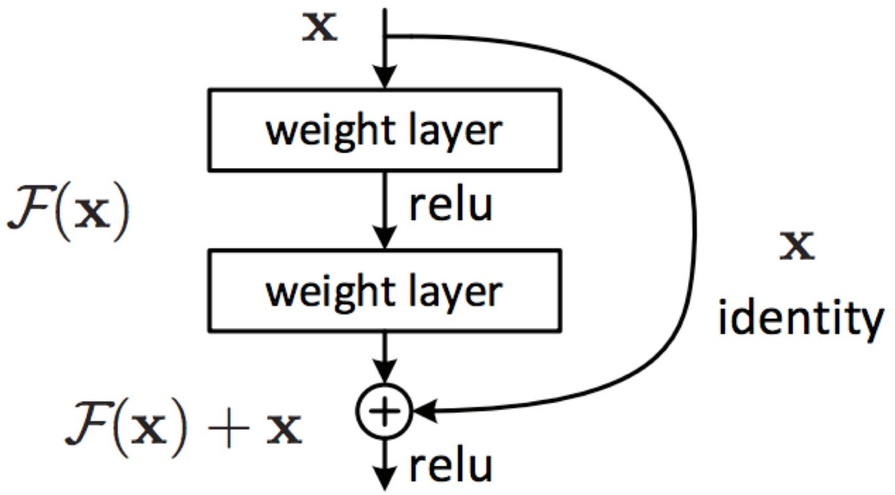
ResNet

- Main idea: Create “shortcut connections”
  - Add inputs from an earlier layer to output of current layer
  - Keep passing initial information to the next layer (in addition to transformed information)

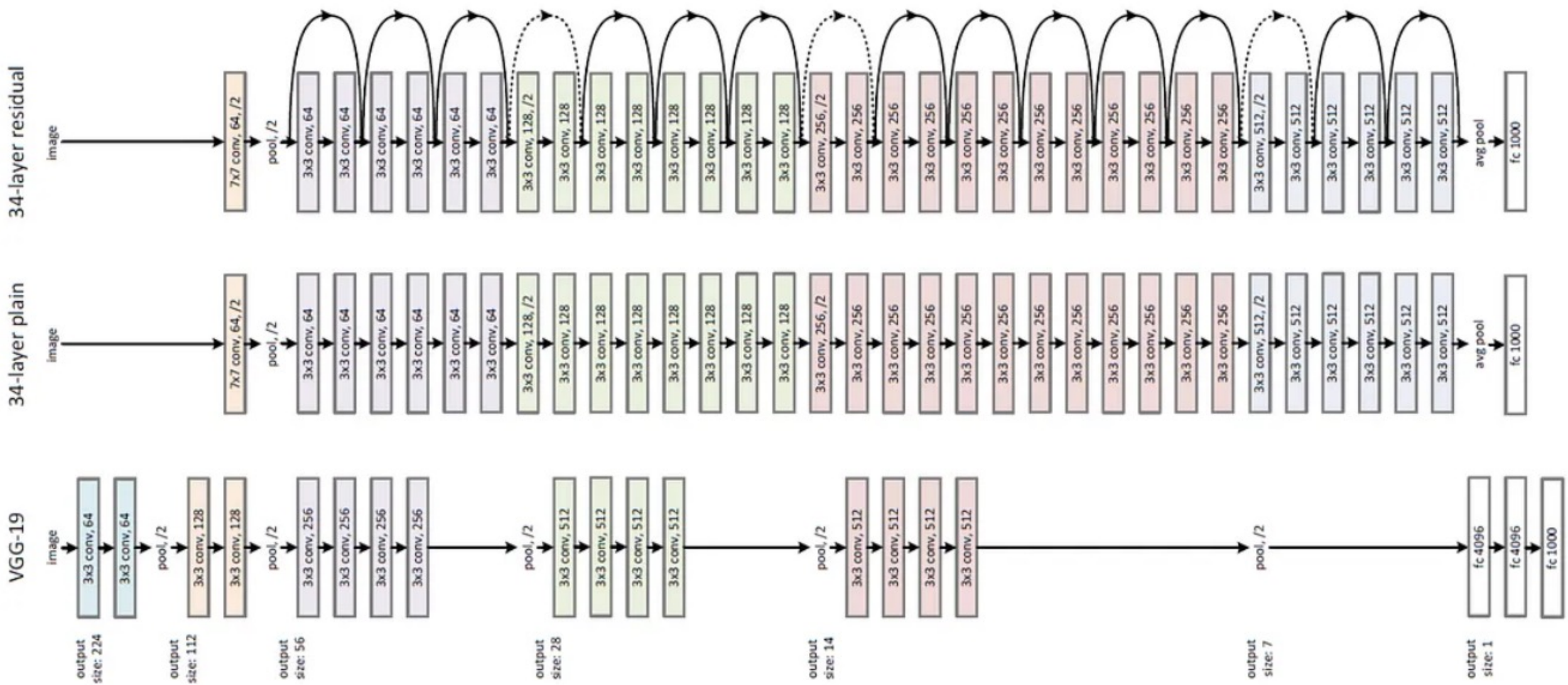


ResNet

- Doesn’t this look similar to boosting?
- Easier to optimize the residual mapping than the original unreferenced mapping

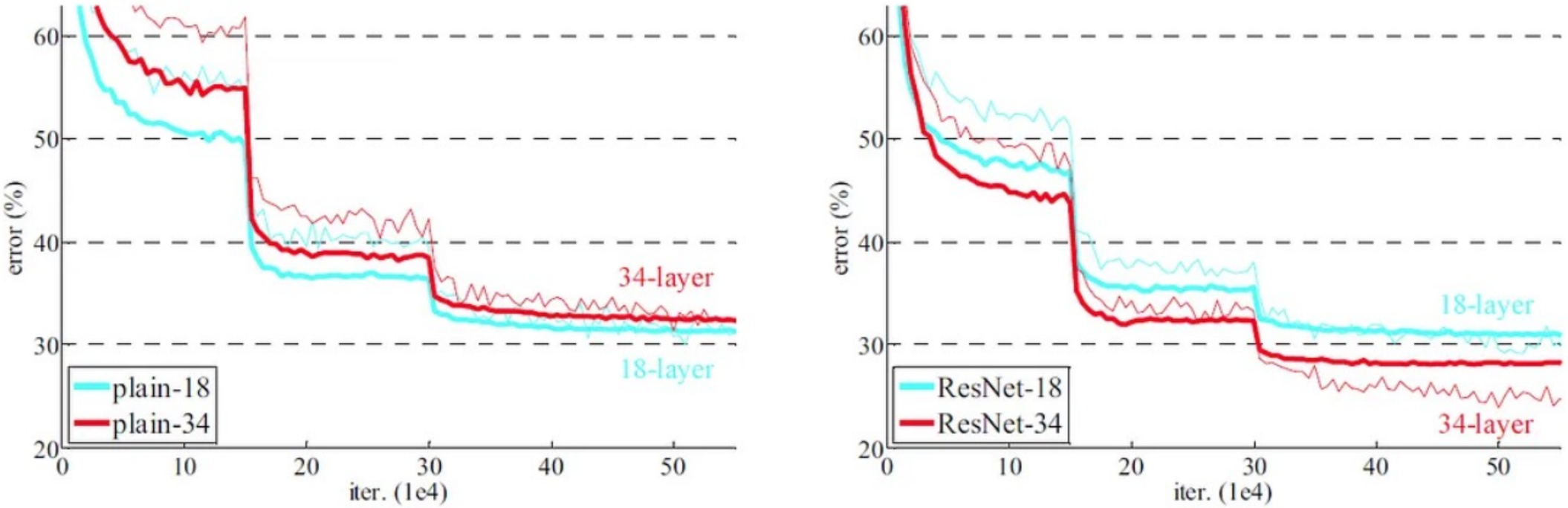


ResNet



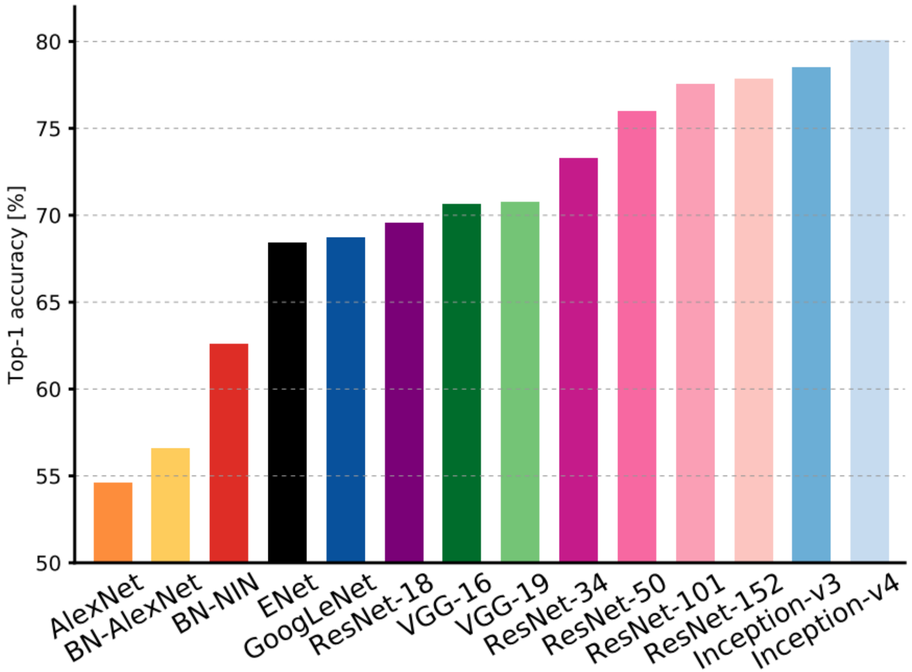
34-layer ResNet with Skip / Shortcut Connection (Top), 34-layer Plain Network (Middle), 19-layer VGG-19 (Bottom)

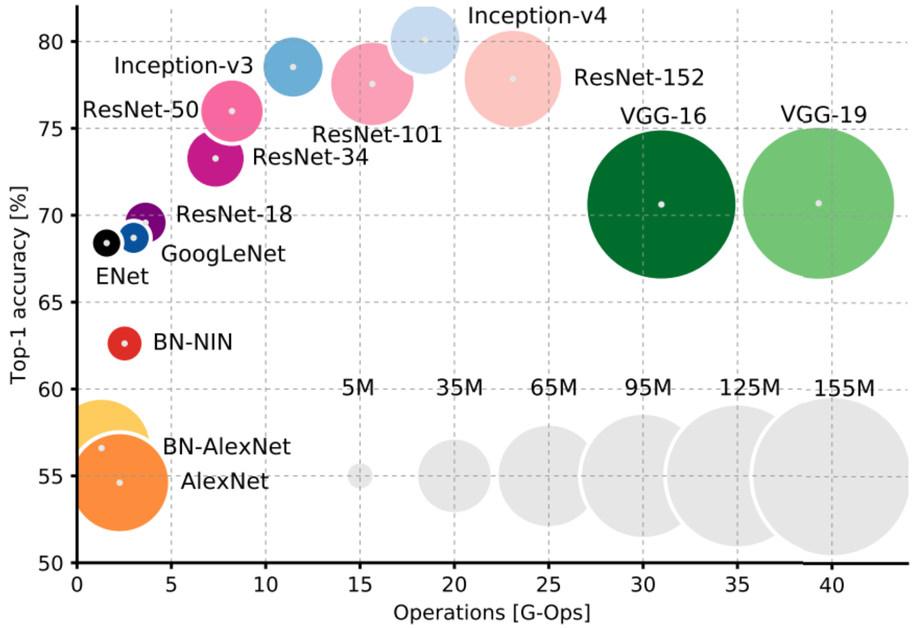
Plain Network vs. ResNet



Validation Error: 18-Layer and 34-Layer Plain Network (Left), 18-Layer and 34-Layer ResNet (right)

Complexity Comparison





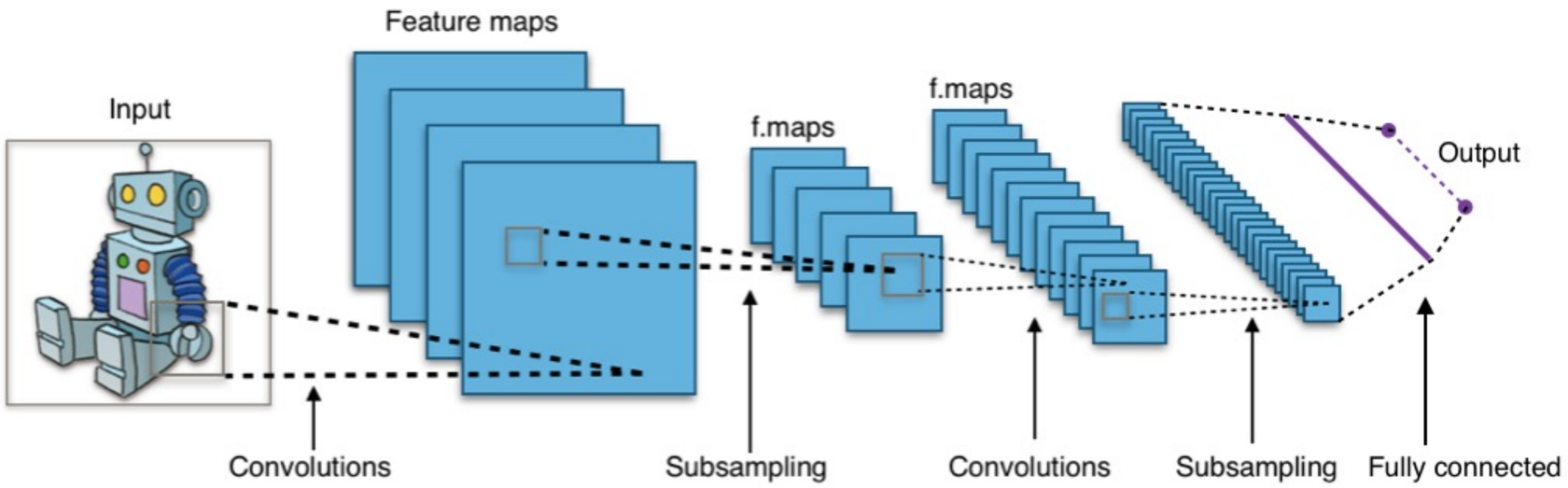
https://arxiv.org/pdf/1605.07678.pdf

Why is CNN Successful?

Compared to standard feedforward neural networks with similarly-sized (5-7) layers

- CNNS have much fewer connections and parameters —> easier to train
- Traditional fully-connected neural network is almost impossible to train when initialized randomly
- Theoretically-best performance is likely only slightly worse than vanilla neural networks

Summary: CNN



CNN was only successful deep network up to 2006, as anything past 3 layers was impossible to train

https://en.wikipedia.org/wiki/Convolutional_neural_network