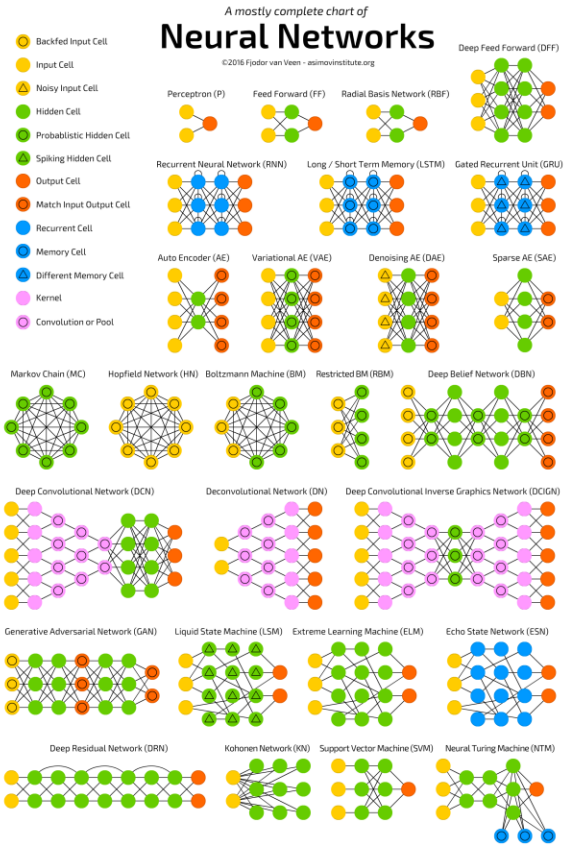
TDS3651
Visual Information Processing



Deep Learning Foundations for Computer Vision (Part 1)

Lecture 11

Faculty of Computing and Informatics
Multimedia University
created by Lai-Kuan, Wong
modified by Yuen Peng, Loh

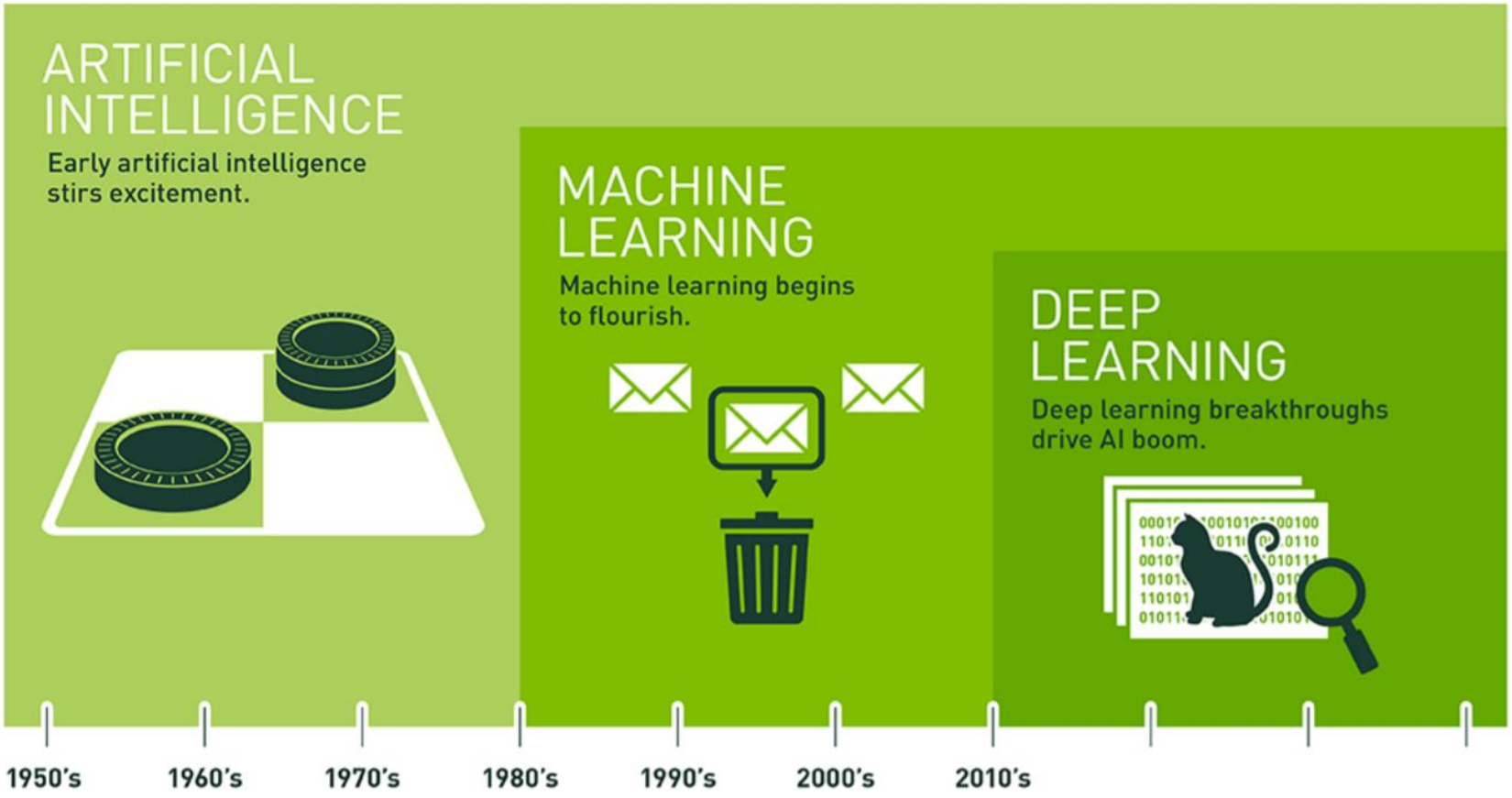
Course Outline

- Machine Learning Foundation
  - Machine Learning Basics
  - Terminologies
  - Pipeline
- Deep learning and CNN Overview
  - Neural Networks
  - Convolutional Neural Networks (CNN)
    - CNN Layers, Architectures, and Transfer Learning

Machine Learning Basics

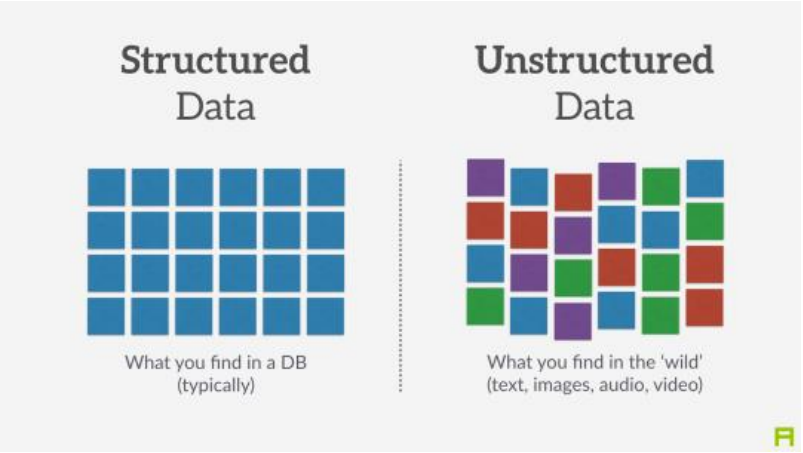
What is Machine Learning?

- Giving computers the ability to learn from data



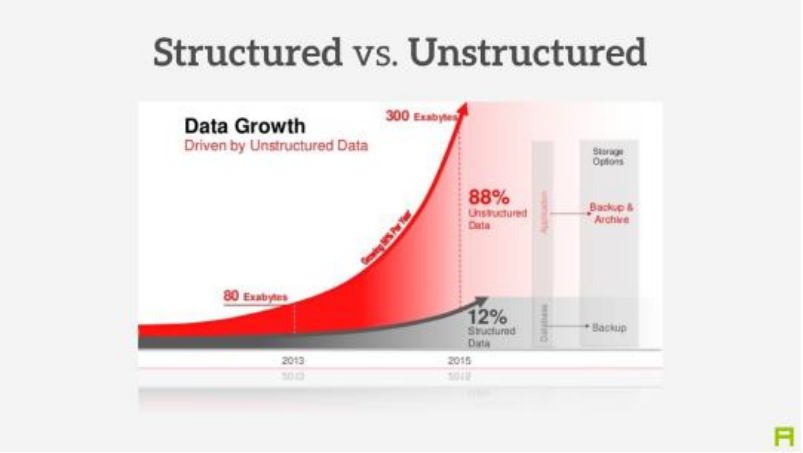
Data for Machine Learning

- Today: Large amounts of data everywhere
- Data
  - Structured
  - Unstructured
- Machine Learning
  - Data-driven
  - Capture knowledge in data to improve model performance



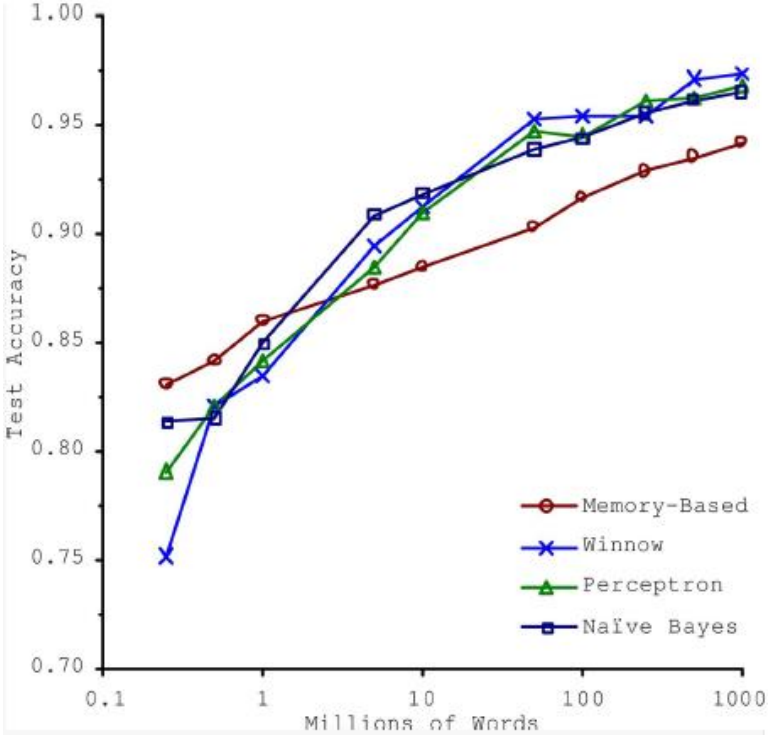
Strict format

Very limited indication of data type



Data for Machine Learning

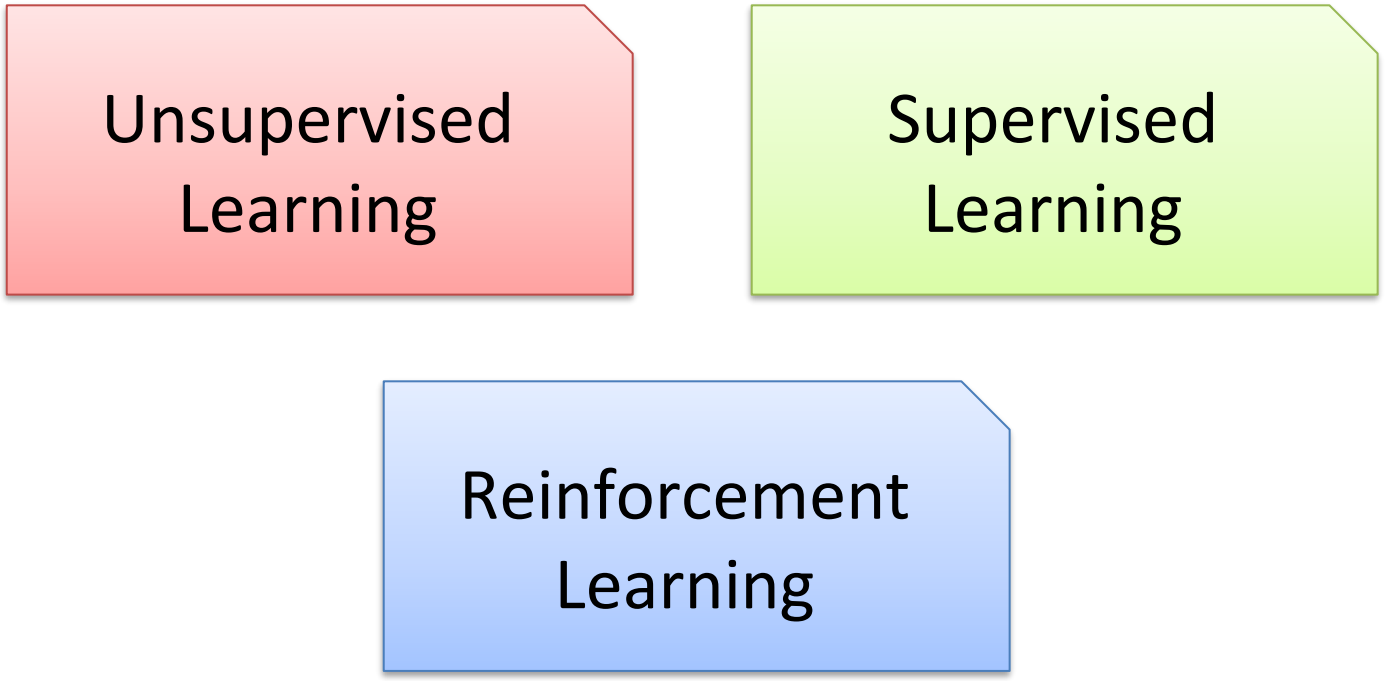
- "The Unreasonable Effectiveness of Data", by Microsoft researchers Michele Banko and Eric Brill
  - The authors said, "these results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development."



Halevy, A., Norvig, P., & Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2), 8-12.

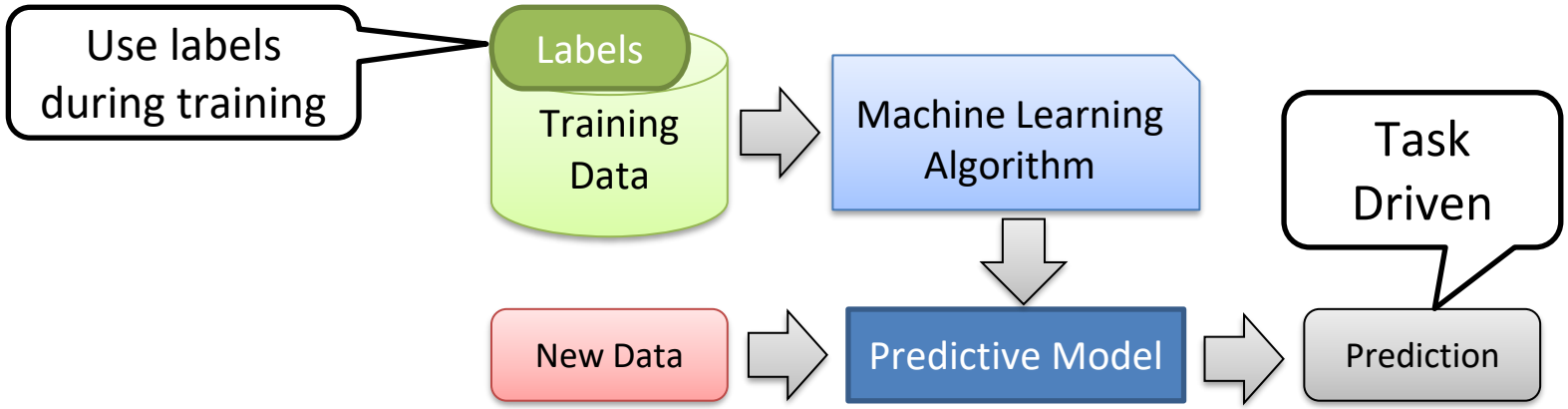
Machine Learning Paradigms

- 3 common types of Machine Learning



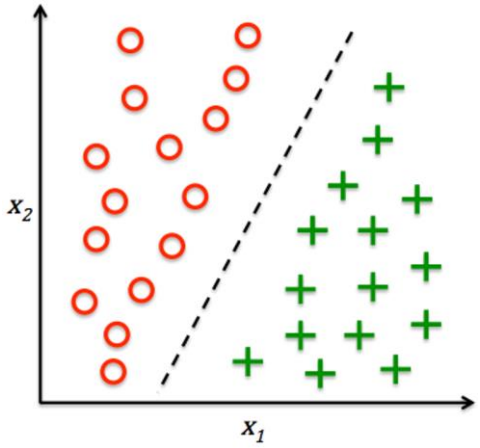
Supervised Learning

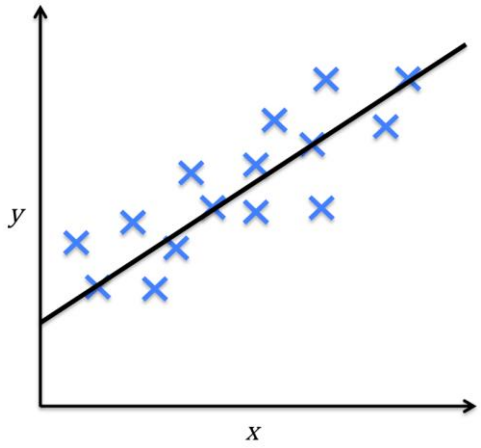
- Making predictions on unseen/future data



Classification

Finds a boundary that best separates the classes





Regression

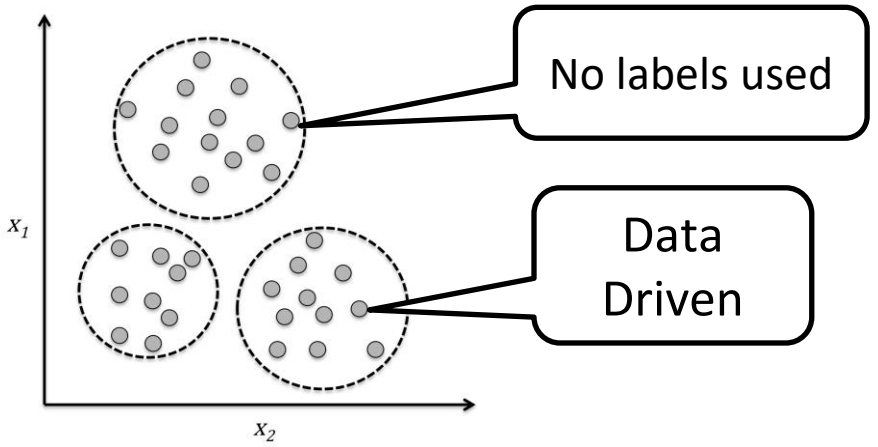
Fit a function that best describes the data

Unsupervised Learning

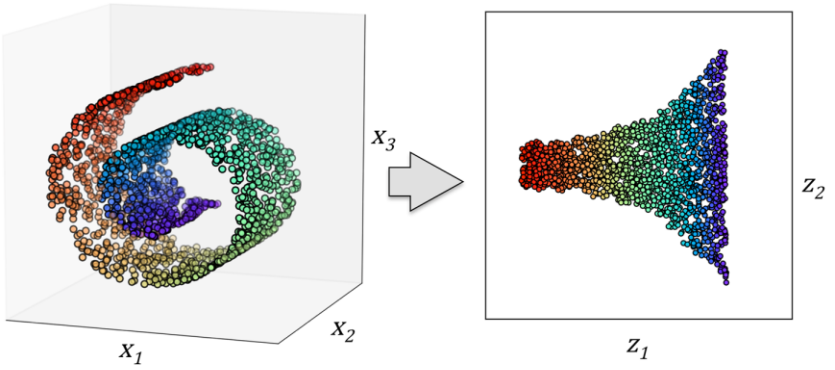
- Discover hidden structure of data without guidance of outcome

Clustering

Organizing data into meaningful sub-groups



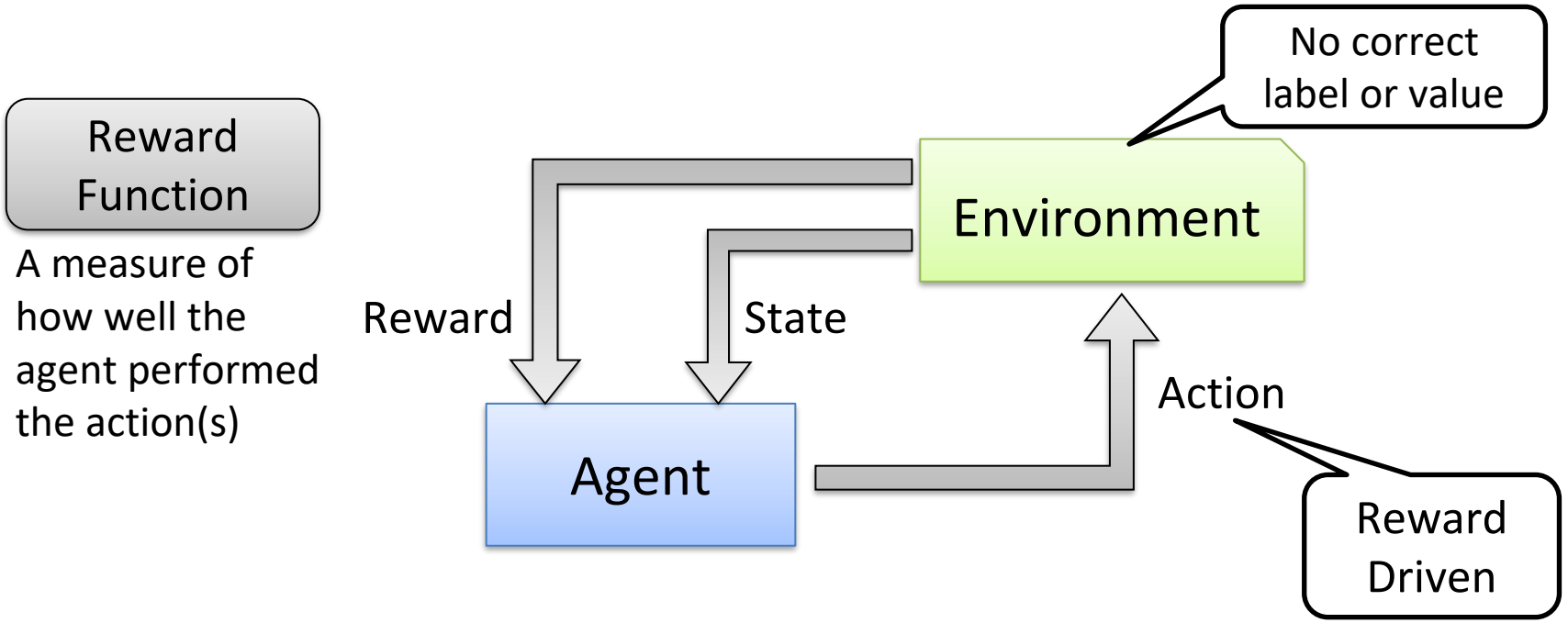
- Dimensionality reduction for data compression



Feature processing, data compression, data visualization

Reinforcement Learning

- System that improves performance based on interactions with the environment



Basic Terminology and Notations

Example:
Iris Dataset

4 features
3 classes
150 samples

Samples
(instances,
observations)

Features
(attributes, measurements, dimensions)

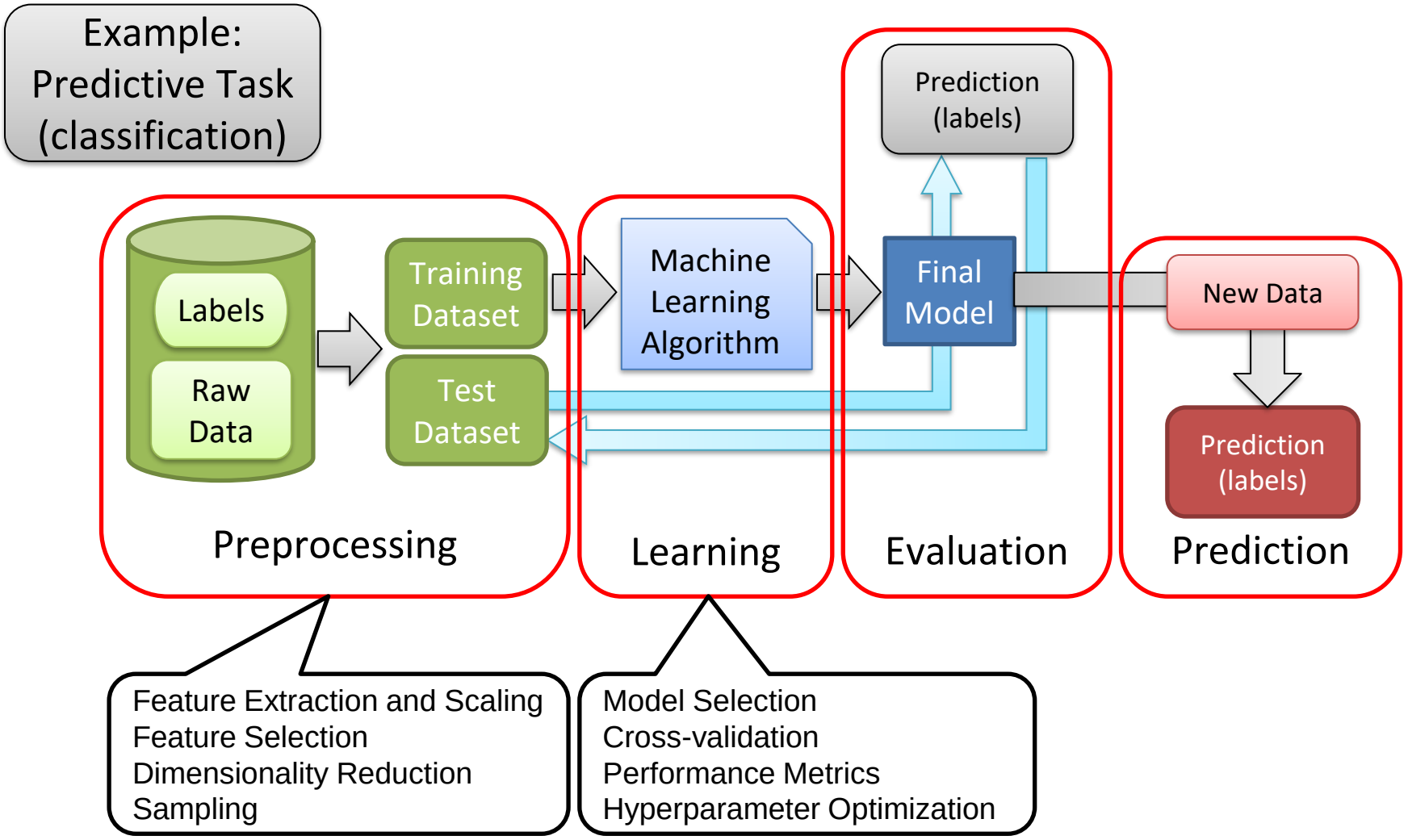
Class labels
(targets)

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris - Setosa
4.9	3.0	1.4	0.2	Iris - Setosa
7.0	3.2	4.7	1.4	Iris - Versicolor
6.4	3.2	4.5	1.5	Iris - Versicolor
6.4	3.2	5.3	2.3	Iris - Verginica
7.9	3.8	6.4	2.0	Iris - Verginica

- Data expressed in standard matrix/vector notation
- Superscript (i) refer to the i-th training sample
- Subscript (j) refer to the j-th feature dimension

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

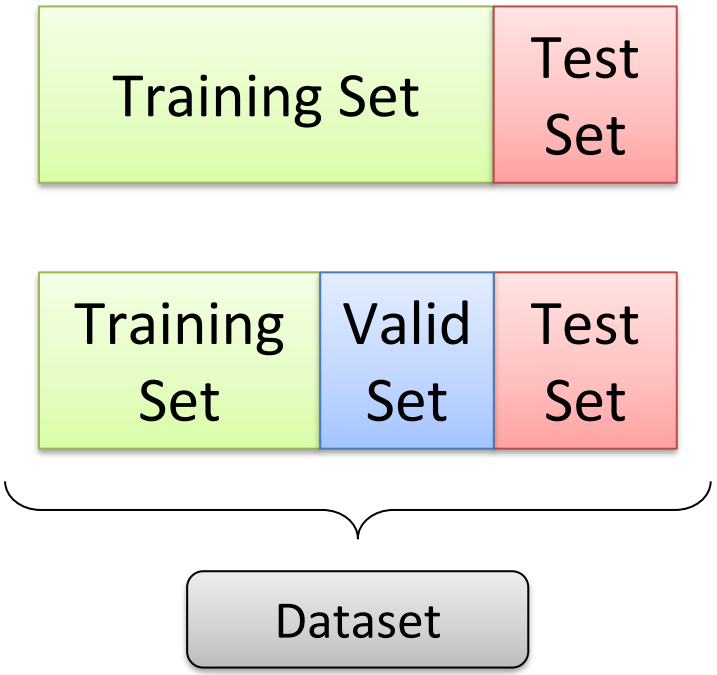
Machine Learning Pipeline



Preprocessing

- Gather the required data
- Raw data is not well formed, not suitable for machine to learn
- Data cleaning
  - Fill missing data, remove outliers
- Normalization of features (usually needed)
- Features may be highly correlated and redundant
  - Requires dimensionality reduction

Partitioning Data



- Data needs to be set up for a model to learn, and then later to be evaluated
- Divide the dataset into separate training and test sets
  - Training set optimizes the machine learning model
  - Test set is used to evaluate the model with unseen data
  - Validation set is used (sometimes) for tuning hyperparameters

Learning and Evaluation

- How well does a learned model generalize from the data it was trained on to a new test set?

Training Set (labels known)

Test Set (labels unknown)

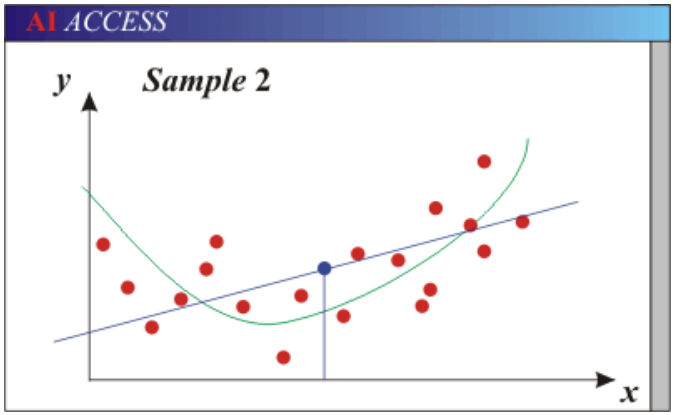
Slide credit: L. Lazebnik

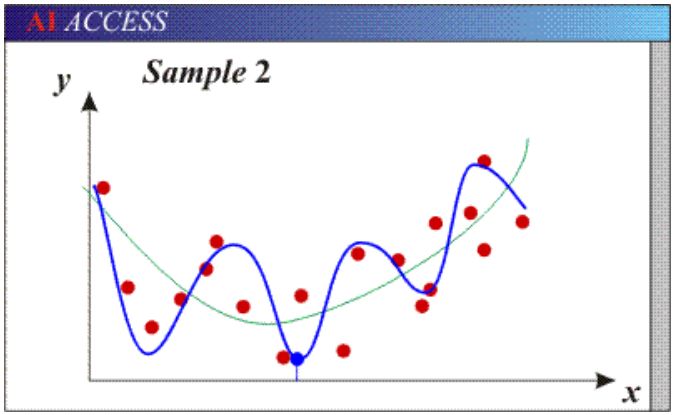
Generalization: Error Components

- Bias: How much the average model over all training sets differ from the true model?
  - Error due to inaccurate assumptions/simplifications made by the model
- Variance: How much the models trained on different datasets differ from each other?
- Underfitting: Model is too simple to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high validation/test error
- Overfitting: Model is too complex and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high validation/test error

Slide credit: L. Lazebnik

Bias-Variance Trade-off





- Models with too few parameters are inaccurate because of a large bias (not enough flexibility)
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample)

$$E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

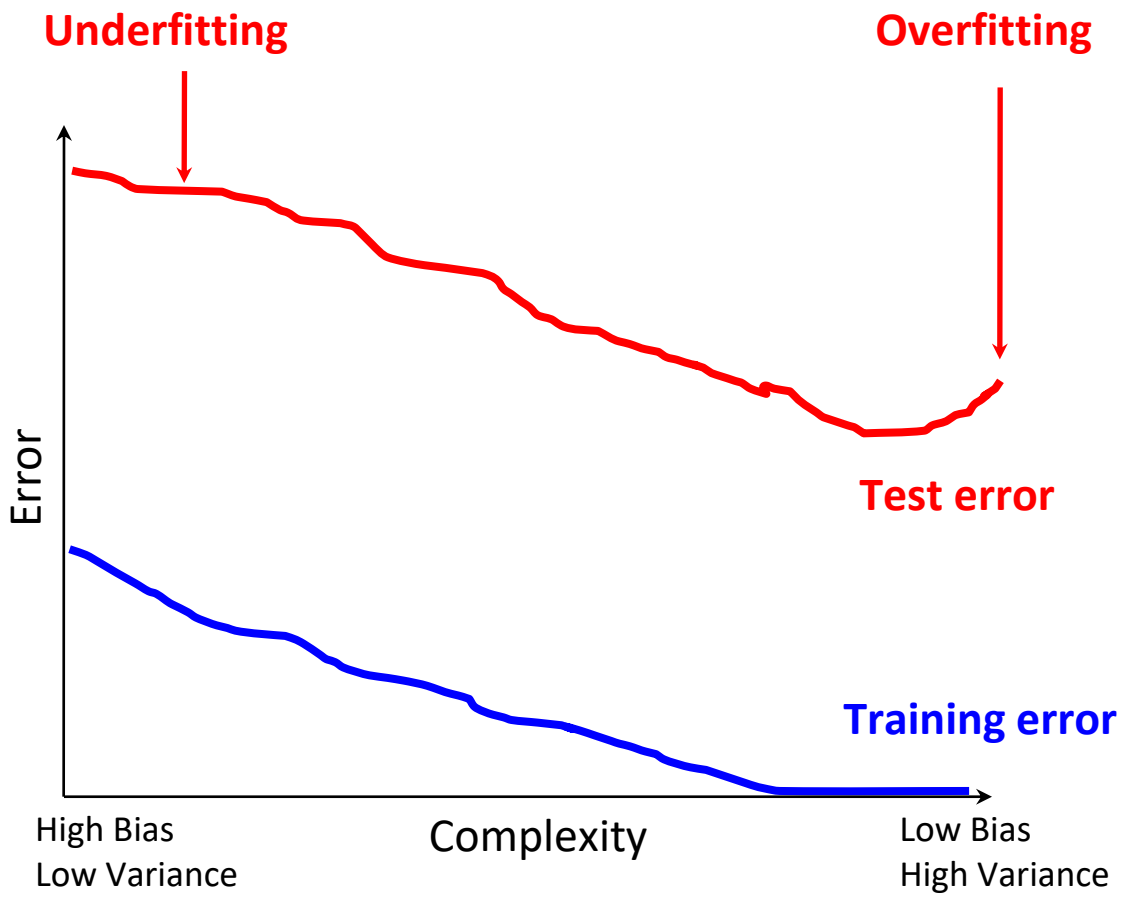
Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

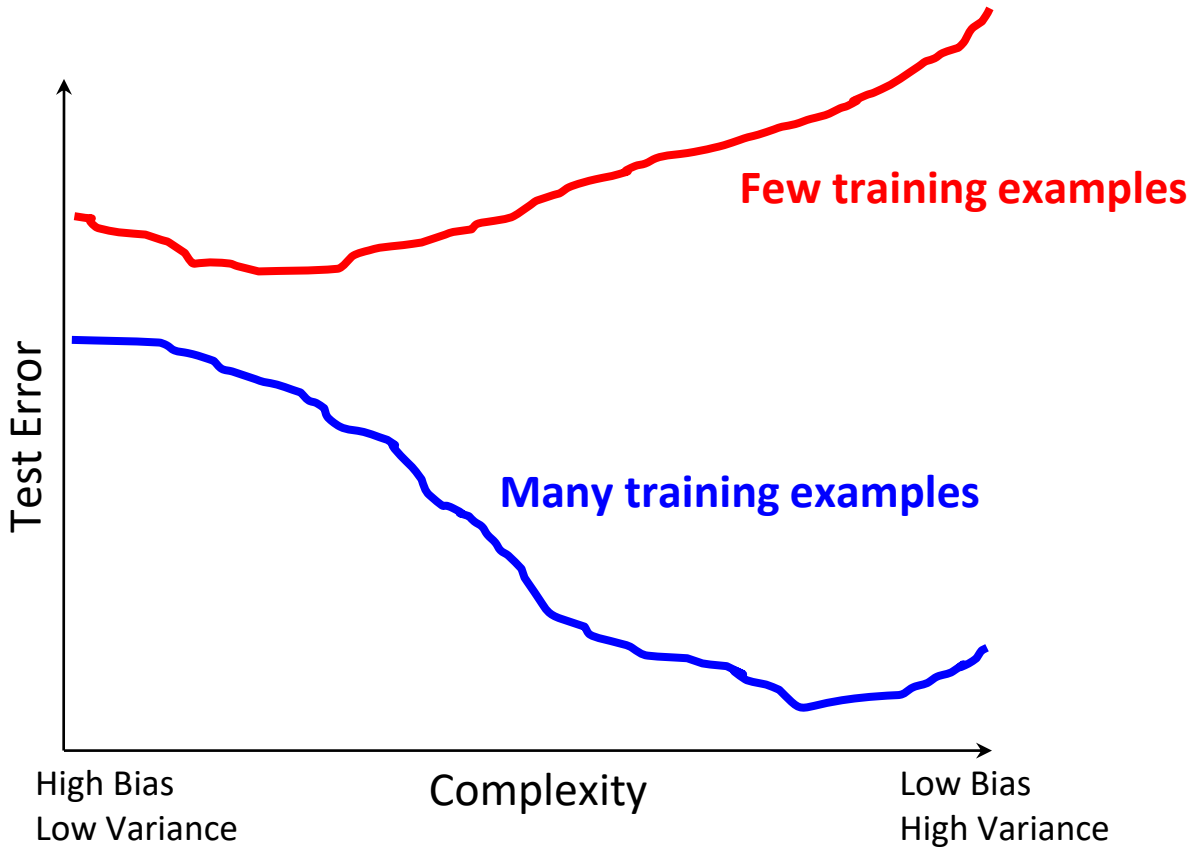
Slide credit: D. Hoiem

Bias-Variance Trade-off



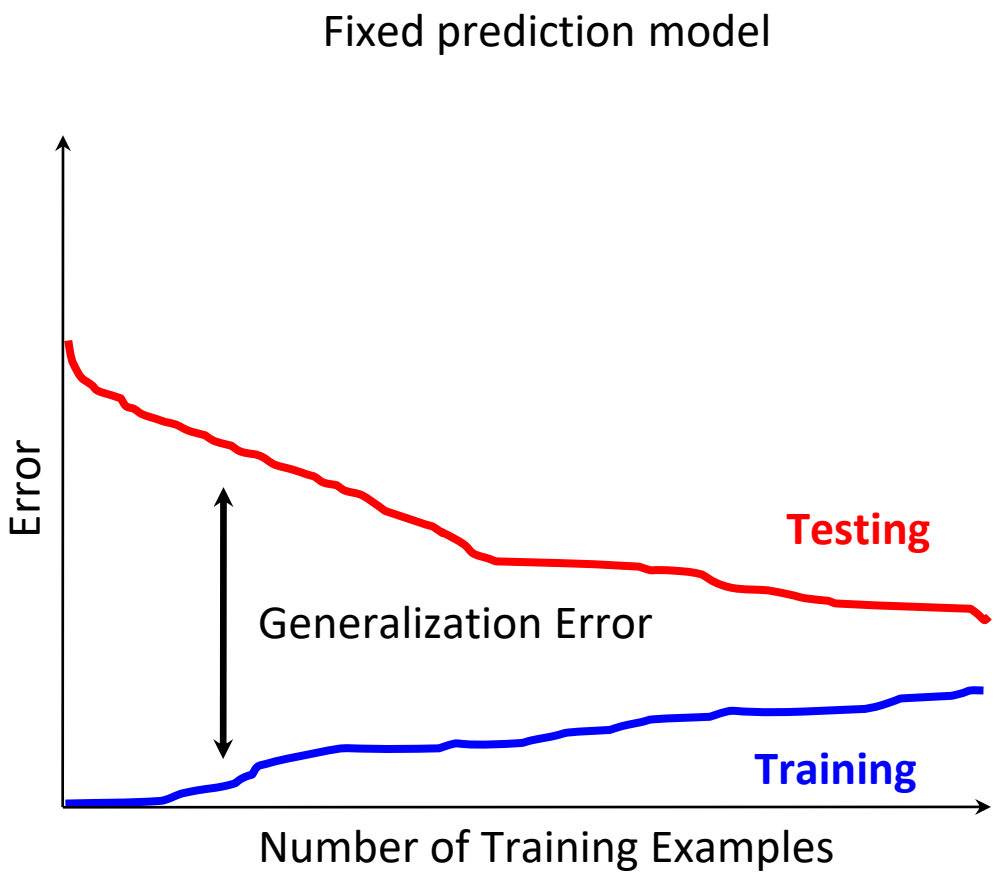
Slide credit: D. Hoiem

Bias-Variance Trade-off



Slide credit: D. Hoiem

Effect of Training Size



Slide credit: D. Hoiem

Learning and Evaluation

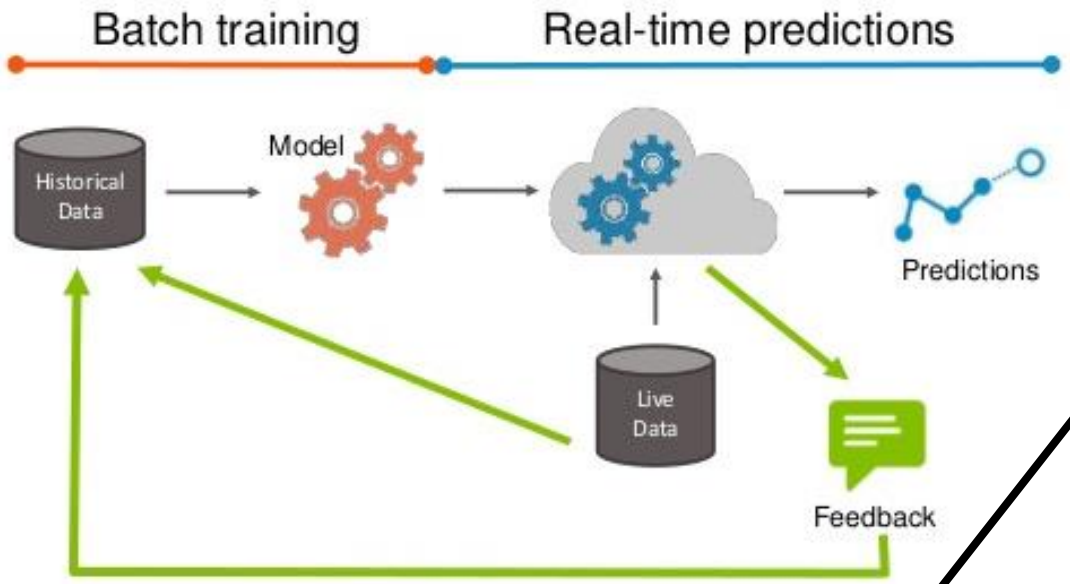
- "No Free Lunch" Theorem
  - Each algorithm has its biases, no model enjoys superiority if we don't make assumptions
- Cross-validation – validation subsets used to estimate generalization performance of model

- Decide on how to measure performance of model
  - Accuracy, precision, recall, mean square error, etc.

Deployment

- After selecting and training model, use test data to estimate generalization error (test error)
- Deploy model to predict new, future data
  - Important to apply similar procedures (feature normalization, dimensionality reduction) to new data

Machine Learning in Production



On-going predictions and update model with “new” data

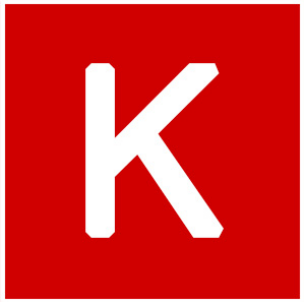
Lifecycle of ML in Production

Python Packages for ML

- **Scikit-Learn**
  - Open source project containing a good number of state-of-the-art ML algorithms
  - Probably the most popular ML library in Python
  - Good documentation, active user community
  - Works seamlessly with a number of other libraries (numpy, pandas)

- **Tensorflow**
  - Open source deep learning library developed by Google
  - Easy numerical computation for large size data
  - Automatic derivatives computation
  - Able to deploy to multiple CPUs/GPUs
  - Powerful data visualization toolkit (TensorBoard)

- **Keras**
  - High-level neural networks API, written in Python
  - Models described in Python code, no separate model config files
  - Allows for easy and fast prototyping
  - Compatible with a number of backend libraries (Tensorflow, Theano, CNTK)



ML for Image Classification

- Traditional machine learning requires feature engineering

Traditional Machine Learning Flow

Example: Iris Dataset

4 features
3 classes
150 samples

Samples (instances, observations)

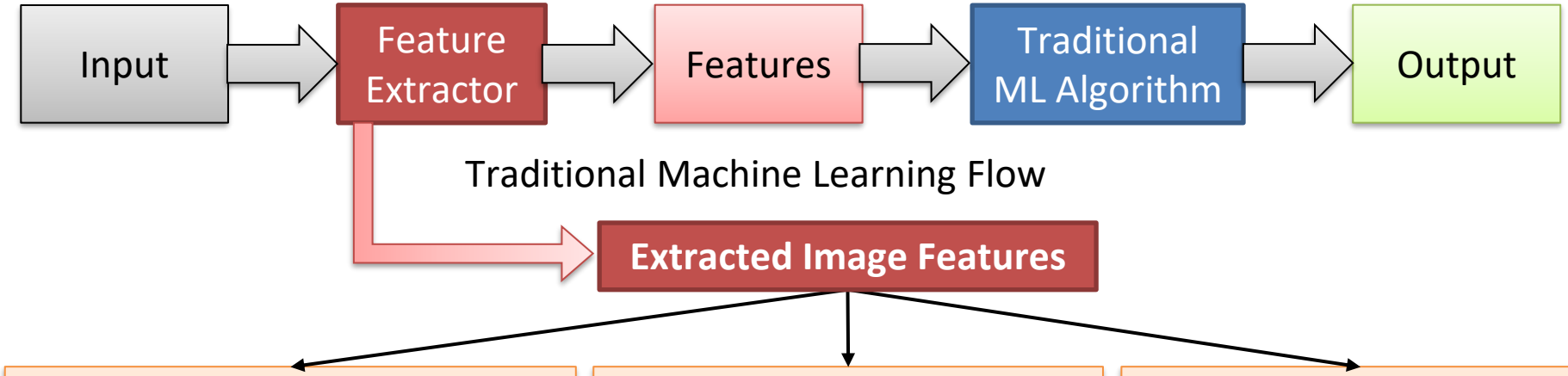
Features (attributes, measurements, dimensions)

Class labels (targets)

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris - Setosa
4.9	3.0	1.4	0.2	Iris - Setosa
7.0	3.2	4.7	1.4	Iris - Versicolor
6.4	3.2	4.5	1.5	Iris - Versicolor
6.4	3.2	5.3	2.3	Iris - Verginica
7.9	3.8	6.4	2.0	Iris - Verginica

ML for Image Classification

- Traditional machine learning requires feature engineering

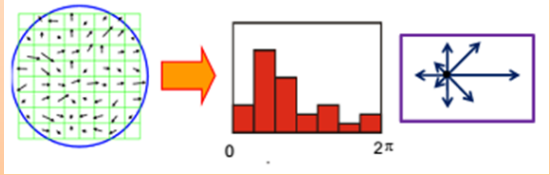


Handcrafted Features

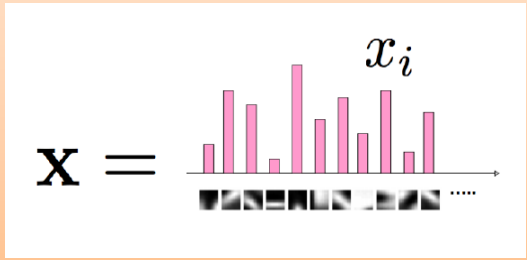
	f	Features
Global features	f_1	Contrast
	f_2	Hue count
	f_3	Saturation
	f_4	Color harmony (Hue)
	f_5	Color harmony (Sat)
	f_6	Color harmony (Val)
	f_7	Global Sharpness
	f_8	Global Brightness

	f	Features
Subject features	f_9	Subject Sharpness
	f_{10}	Subject Brightness
	f_{11}	Spatial composition (Portraiture)
	f_{12}	Spatial composition (Macro)
	f_{13}	Bounding Box
Contrasting features	f_{14}	Contrasting Sharpness
	f_{15}	Contrasting Brightness
	f_{16}	Subject size ratio
	f_{17}	Complementary color

Generic Descriptors (e.g. SIFT/SURF/HOG)

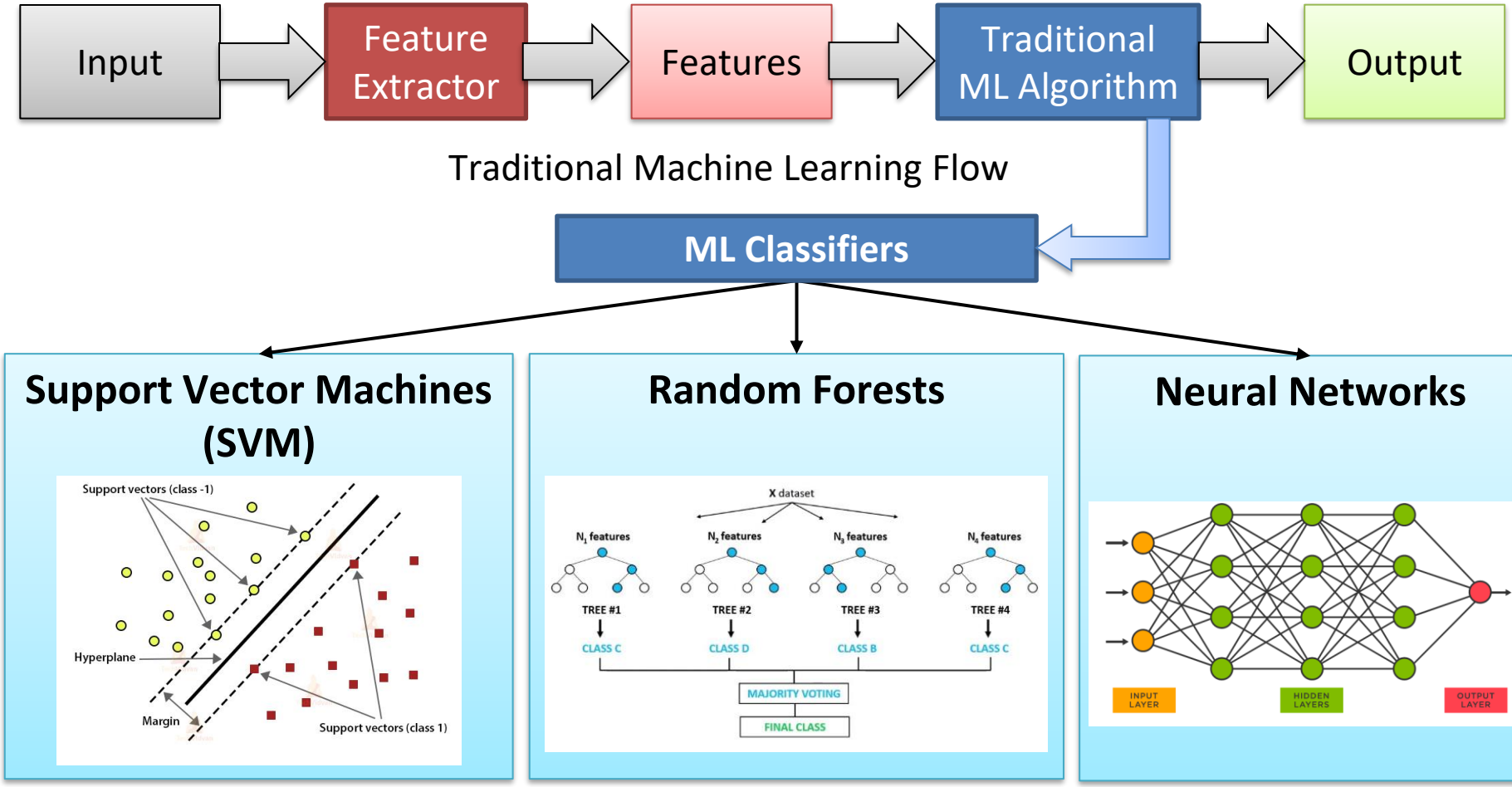


Histogram occurrences of visual words



ML for Image Classification

- Traditional machine learning requires feature engineering

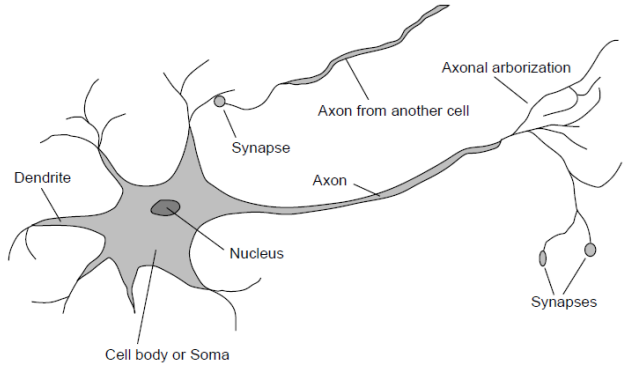


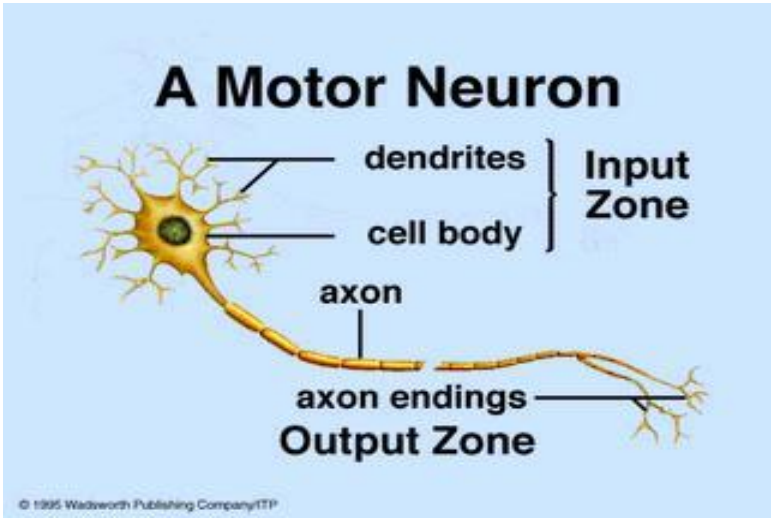
Neural Network

Biological Neuron and Neural Network

- A human biological neuron has three types of main components; **dendrites**, **soma** (or cell body) and **axon**.
- Dendrites **receives signals** from other neurons.
- The soma, **sums the incoming signals**. When **sufficient input** is received, the **cell "fires"** to transmit the signal over to its axon
- Axon **sends signals** to other cells.

10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
Signals are noisy "spike trains" of electrical potential





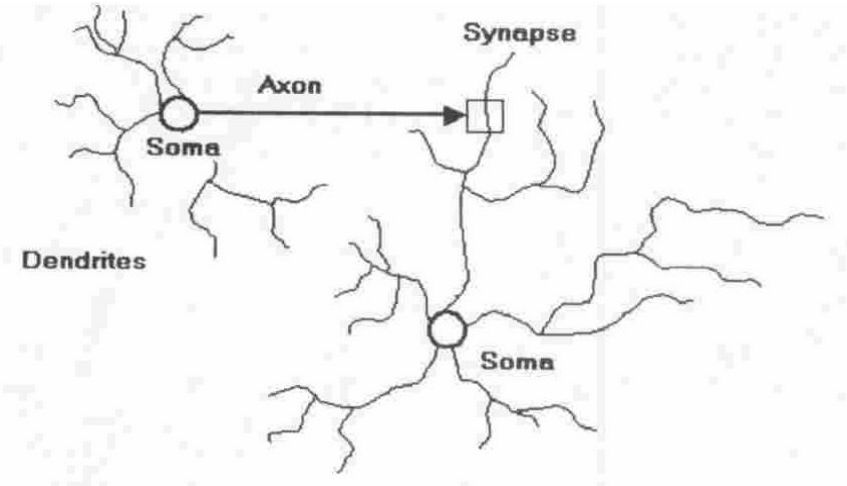
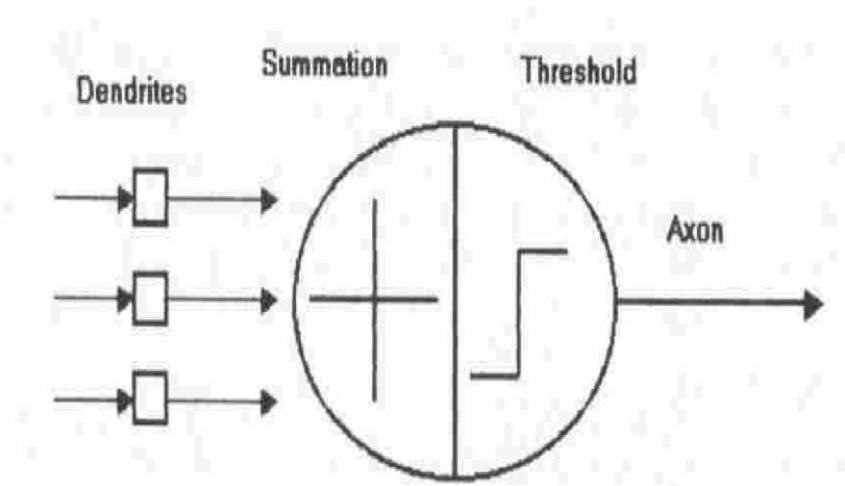
Artificial Neural Network

"Data processing system consisting of a large number of simple, highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain"

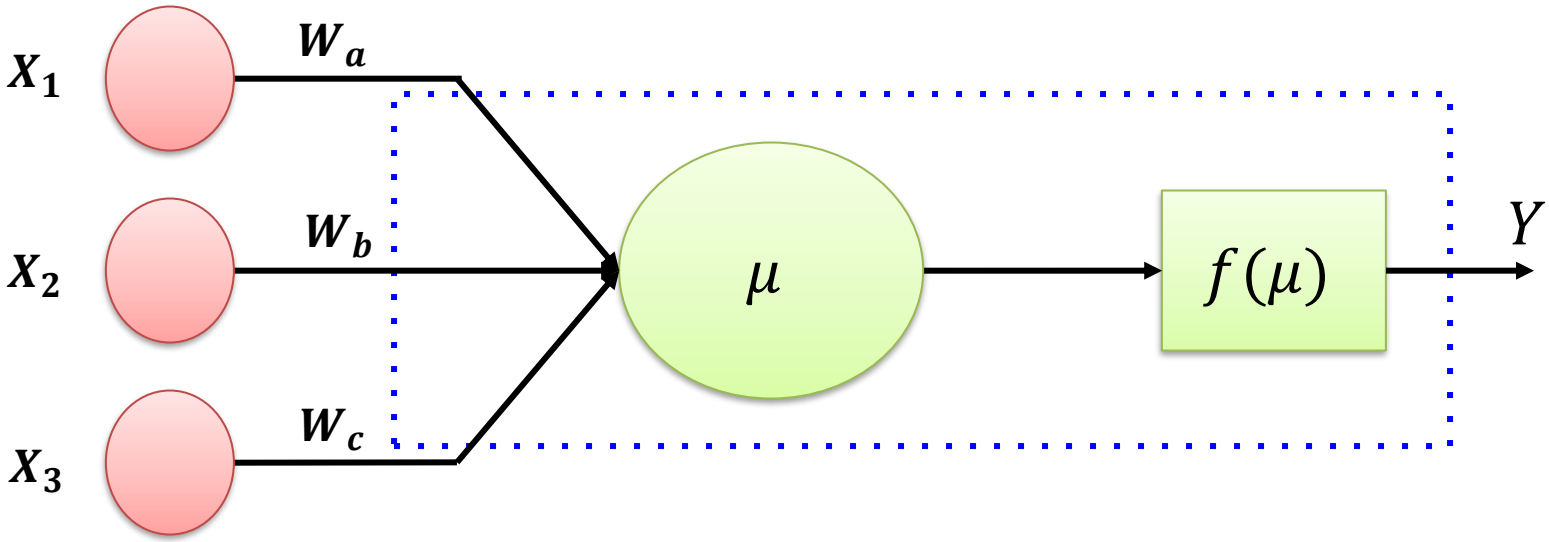
(Tsoukalas & Uhrig, 1997).

- Neural network: information processing paradigm inspired by biological nervous systems, such as our brain
- Structure: large number of highly interconnected processing elements (neurons) working together
- Like people, they learn from experience (by example)

Artificial Neural Network

Biological Neuron	Artificial Neuron
	
Four basic components of a human biological neuron	The components of a basic artificial neuron

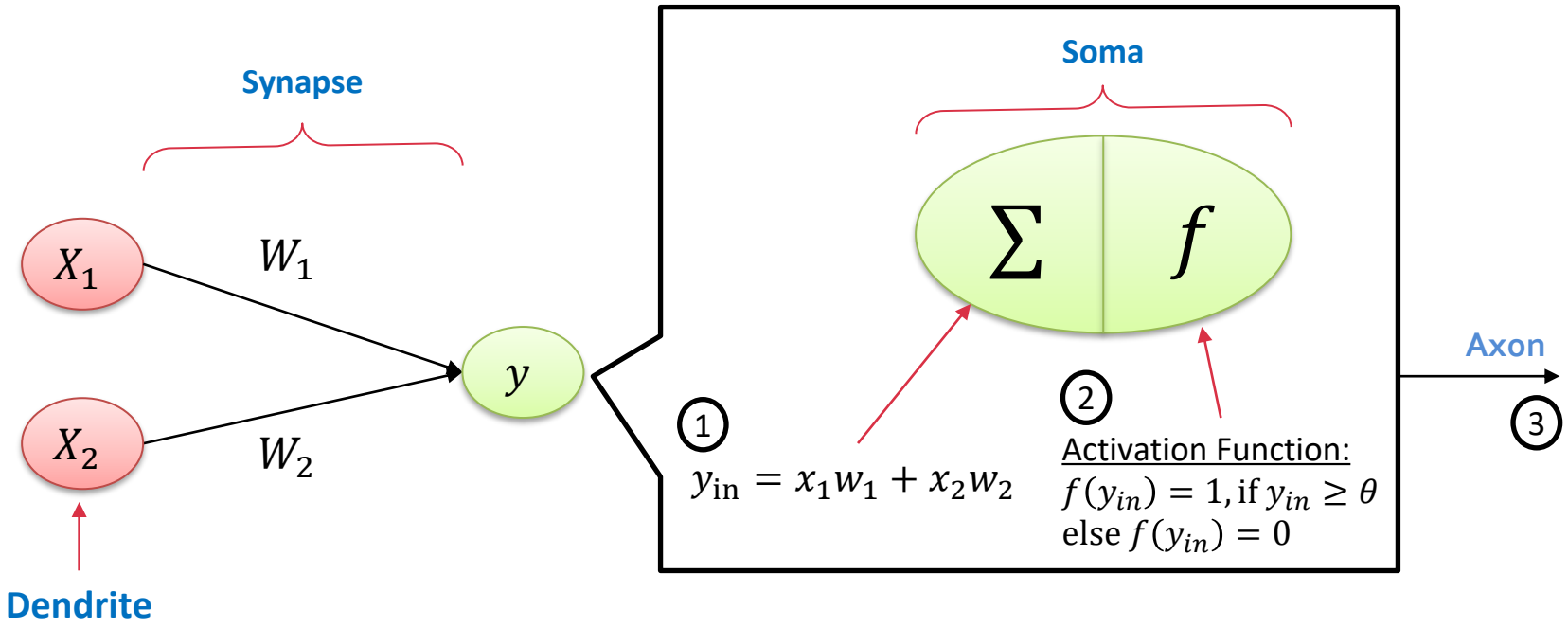
Model of a Neuron



Input units — Connection weights — Summing function — Activation function

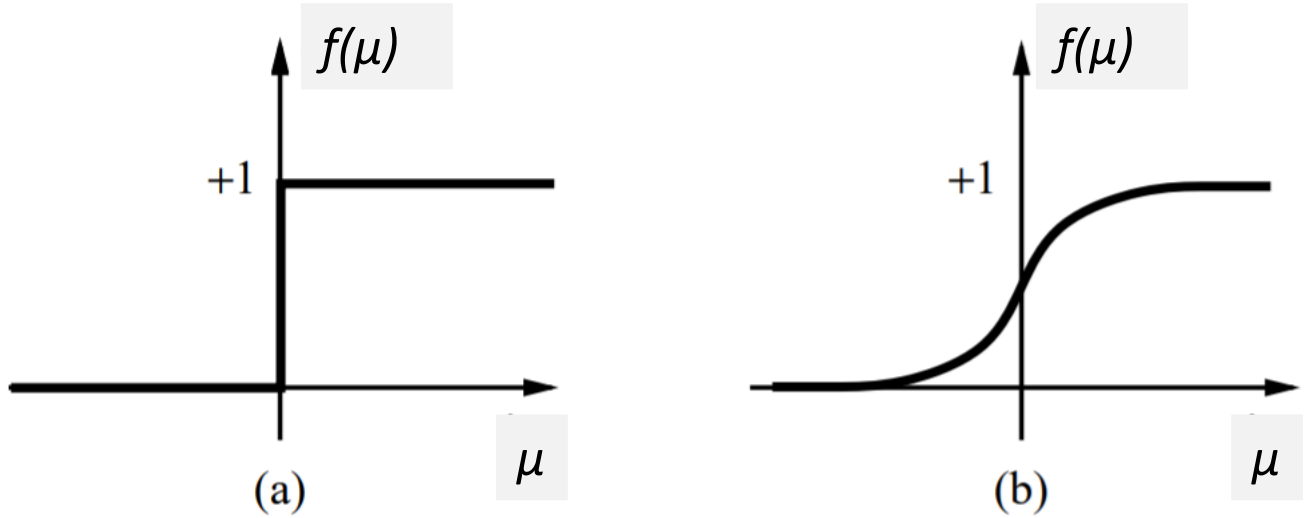
(dendrite) **(synapse)** **(soma)** **(axon)**

Model of a Neuron



1. A neuron receives input, determines the strength or the weight of the input, calculates the total weighted input
2. If the total weighted input greater than or equal the threshold value, the neuron will produce the output, and if the total weighted input less than the threshold value, no output will be produced
3. The value is in the range of 0 and 1

Activation Functions

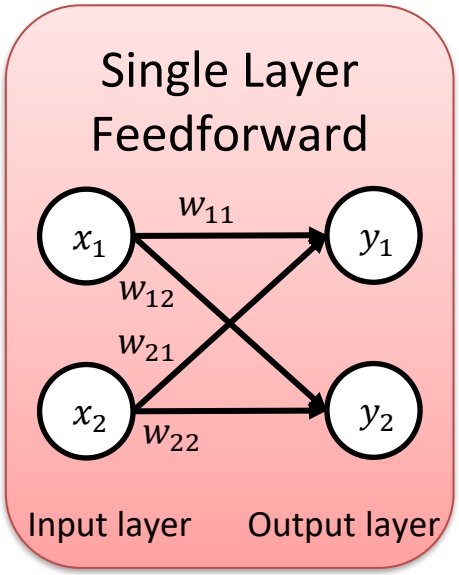


- (a) is a step function or threshold function
- (b) is a sigmoid function: $\frac{1}{1+e^{-x}}$

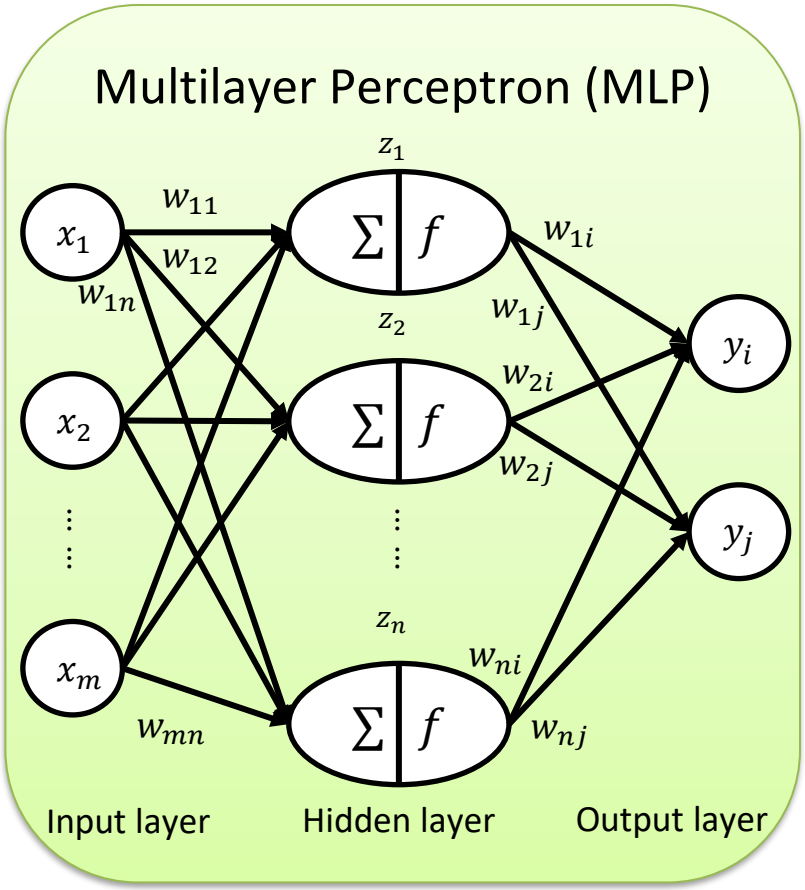
Changing the bias weight $W_{0,i}$ moves the threshold location

Architecture

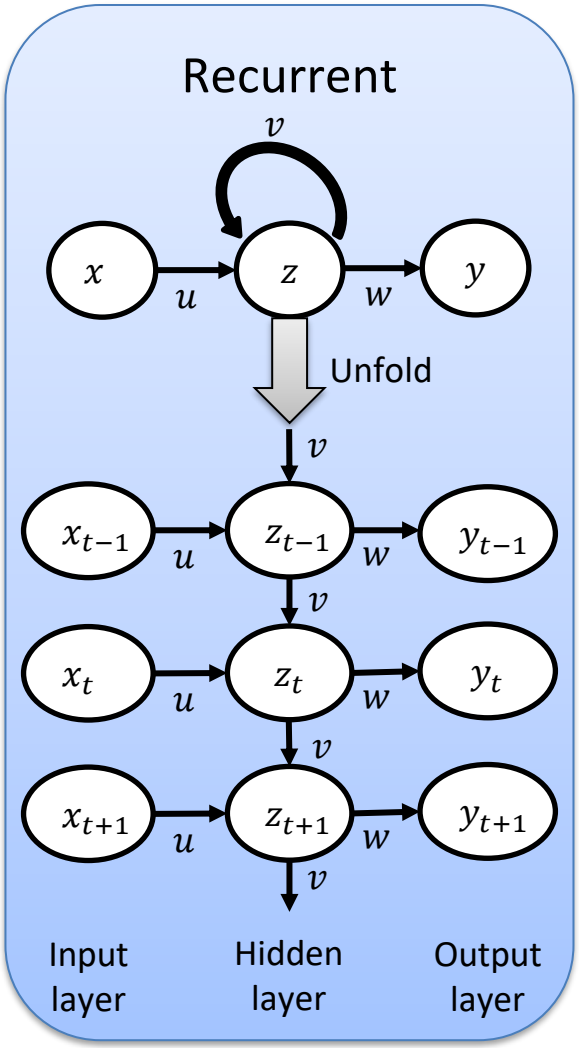
- Patterns of connections between neurons



Examples:
ADALINE, AM,
Hopfield, LVQ,
Perceptron, SOFM



Examples:
CCN, GRNN, MADALINE, MLFF with
BP, Neocognitron, RBF, RCE



Advantages of NN

- Non-linearity
  - It can model non-linear systems
- Input-output mapping
  - It can derive a relationship between a set of input & output responses
- Adaptivity
  - The ability to learn allows the network to adapt to changes in the surrounding environment
- Evidential response
  - It can provide a confidence level to a given solution

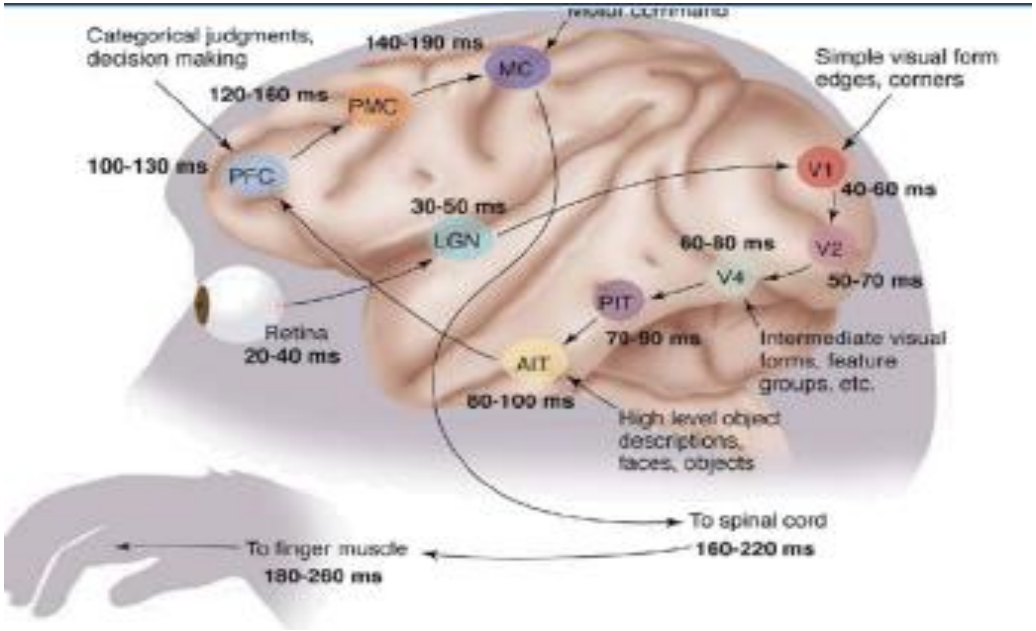
Deep Learning and CNN Basics

What is Deep Learning?

- Part of Machine Learning field of learning representations of data
- Utilizes learning algorithms that derive meaning out of data by using a hierarchy of multiple layers
- Mimics the neural networks of our brain

Inspired by the Brain

- First hierarchy of neurons that receives information in the visual cortex are sensitive to specific edges
- Brain regions further down the pipeline are sensitive to more complex structures such as faces



Brief History

1958 Perceptron

1974 Backpropagation

Convolution Neural Networks for Handwritten Recognition

1998

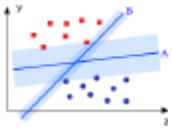
Google Brain Project on 16k Cores

2012

awkward silence (AI Winter)

1969
Perceptron criticized

1995
SVM reigns



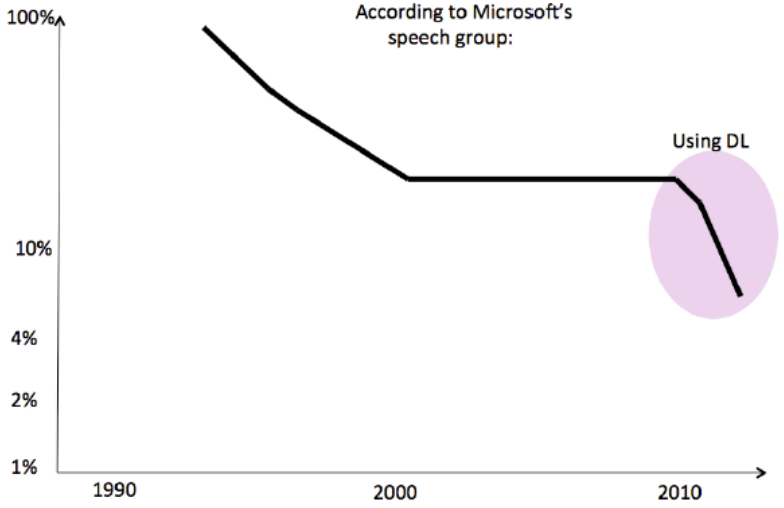
2006
Restricted Boltzmann Machine

2012
AlexNet wins ImageNet
IMAGENET

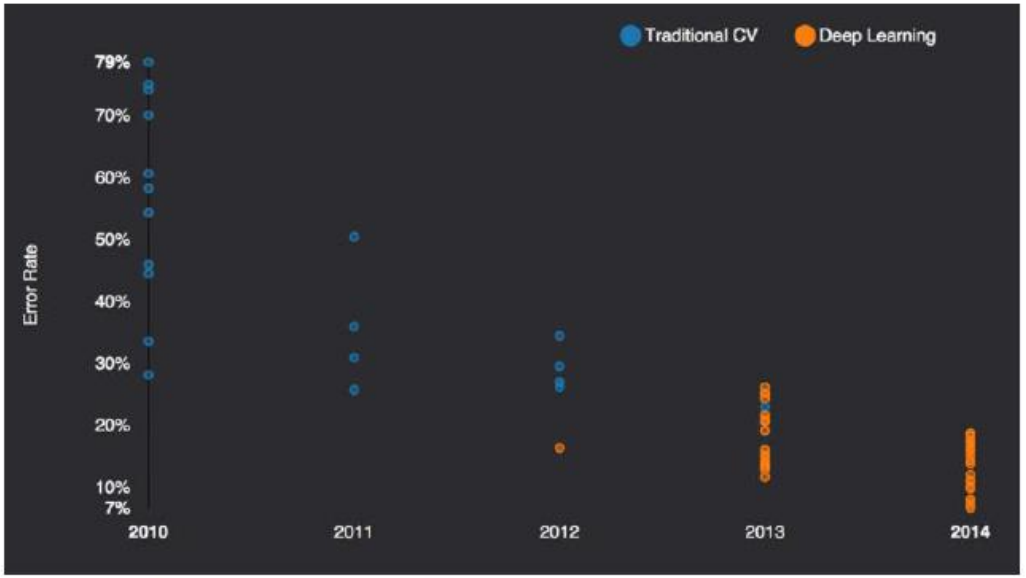
Big Data
(Digitalization)

Computation
(Moore's Law, GPUs)

Algorithmic Progress



Deep Learning in Speech Recognition



ImageNet: The "computer vision World Cup"

Deep Learning Experts

Geoffrey E. **Hinton**
Chief Scientific Advisor, *Vector Institute*
Vice President and Engineering Fellow, *Google*
Emeritus Distinguished Professor of Computer Science, *University of Toronto*

Yann **LeCun**
Director of AI Research at *Facebook*
Silver Professor of Data Science, Computer Science, Neural Science, and Electrical Engineering, *New York University*

Yoshua **Bengio**
Full Professor in the Department of Computer Science and Operations Research, *Université de Montreal*
Founder and Scientific Director of *Mila* – Quebec Artificial Intelligence Institute
Scientific Director of *IVADO* – Artificial Intelligence Research and Transfer Institute

Andrew Yan-Tak **Ng**
Adjunct Professor at Computer Science Department, *Stanford University*
Chairman and Co-Founder of *Coursera*
Founder of *DeepLearning.AI*
Founder & CEO of *Landing AI*

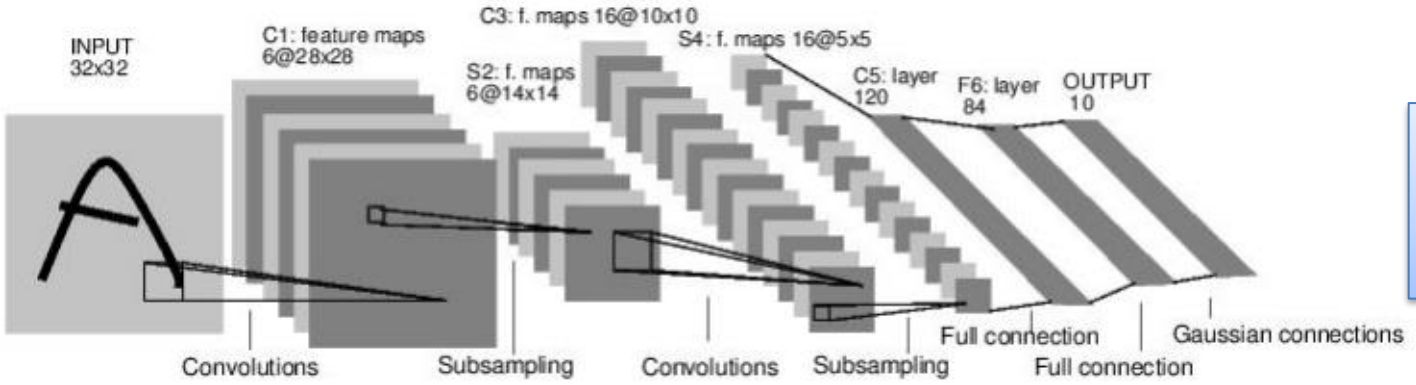
Fei-Fei **Li**
Sequoia Professor in the Computer Science Department, *Stanford University*
Denning Co-Director, *Stanford Human-Centered AI Institute (HAI)*
Inventor of *ImageNet* and the ImageNet Challenge

Deep Learning vs. Tradition

- Deep learning does not require feature engineering anymore

Traditional Machine Learning Flow

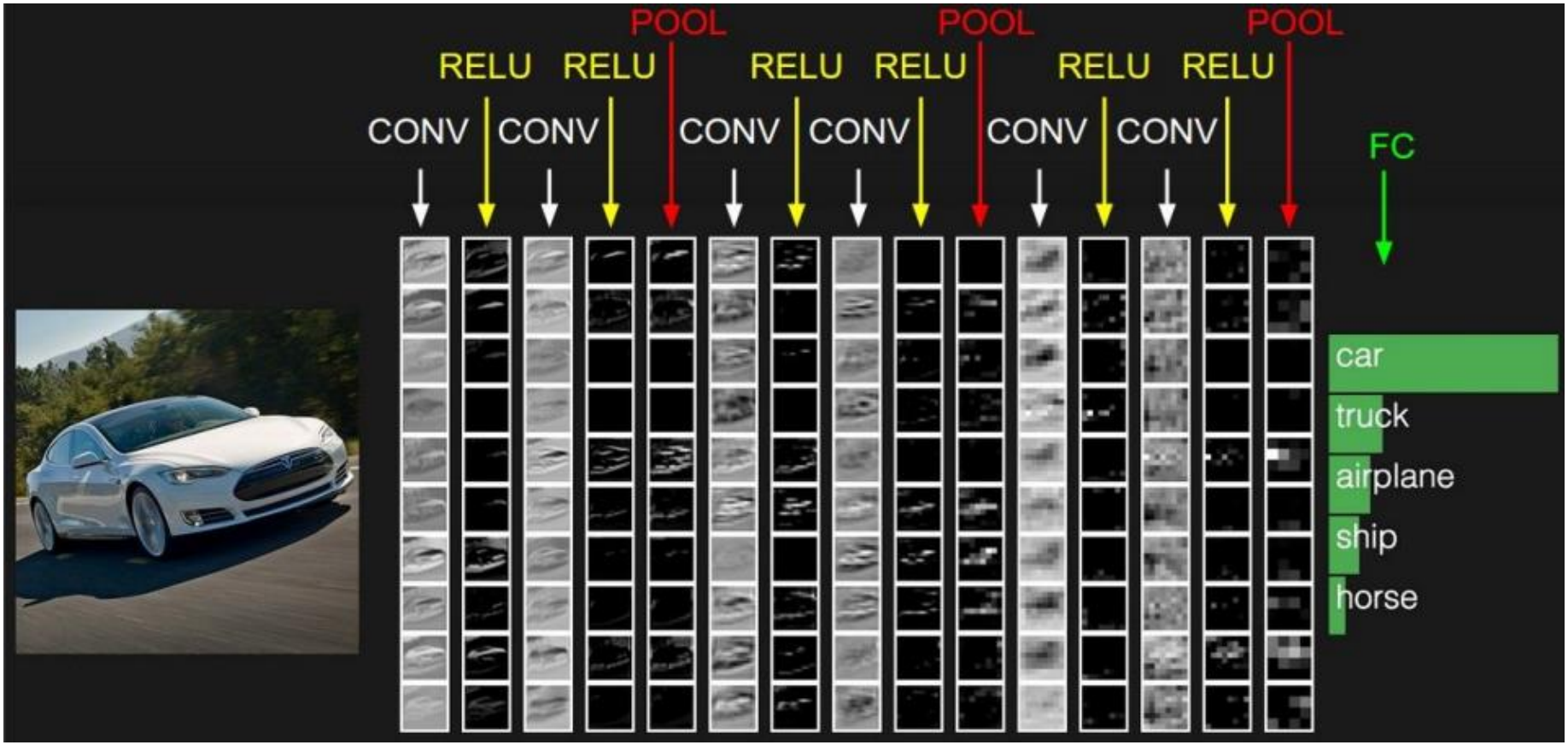
Deep Learning Flow



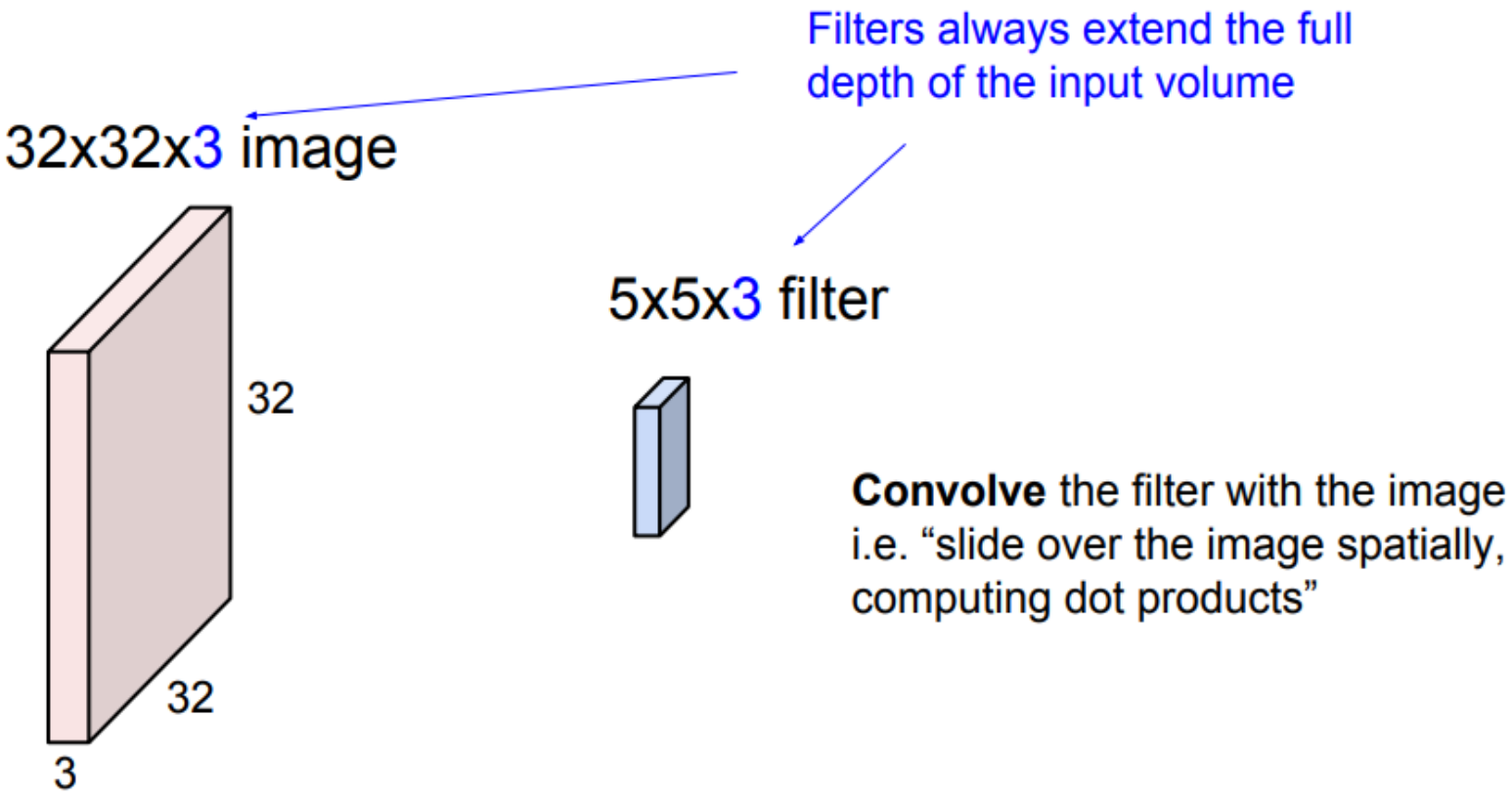
Convolutional Neural Network (CNN)

LeNet – Yann LeCun et. al., 1998

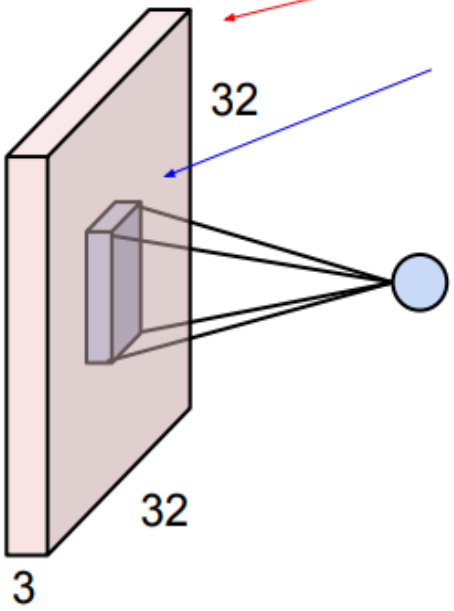
Convolutional Neural Network



Convolution Layer



Convolution Layer



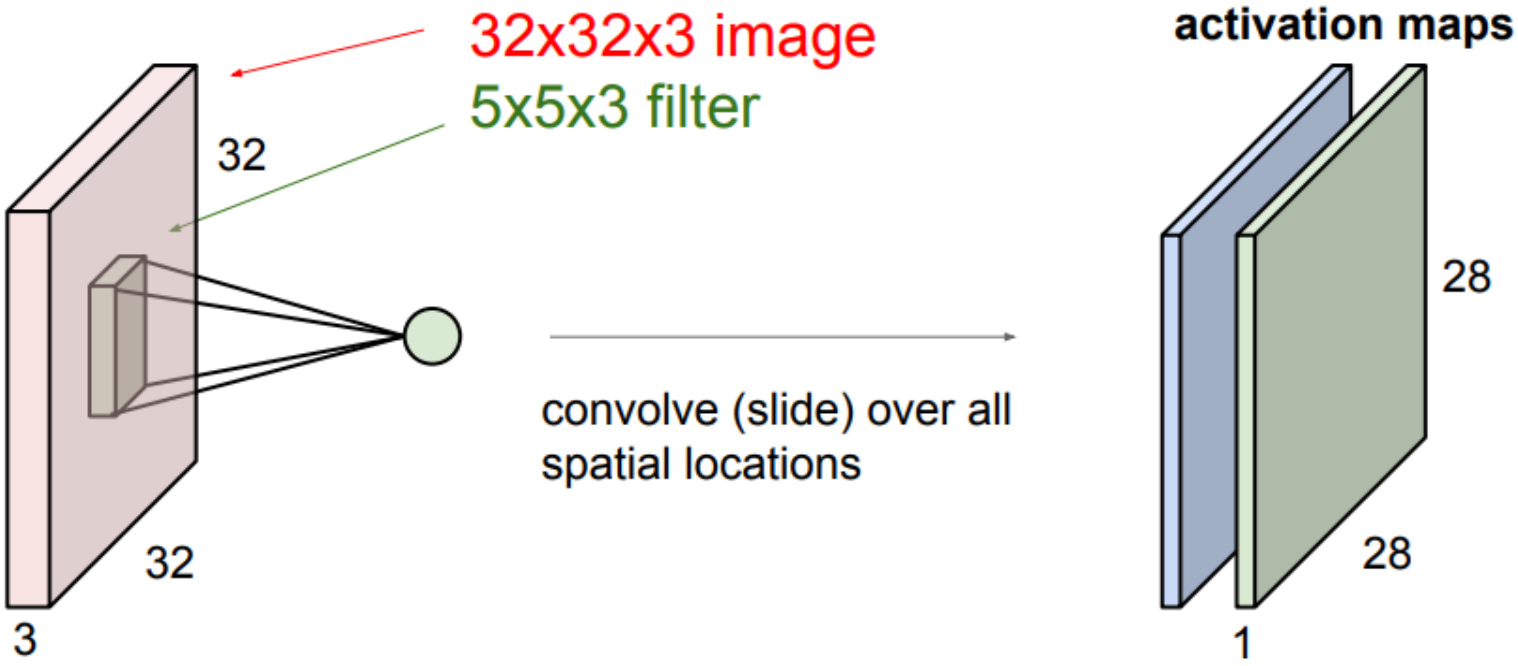
32x32x3 image

5x5x3 filter w

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)

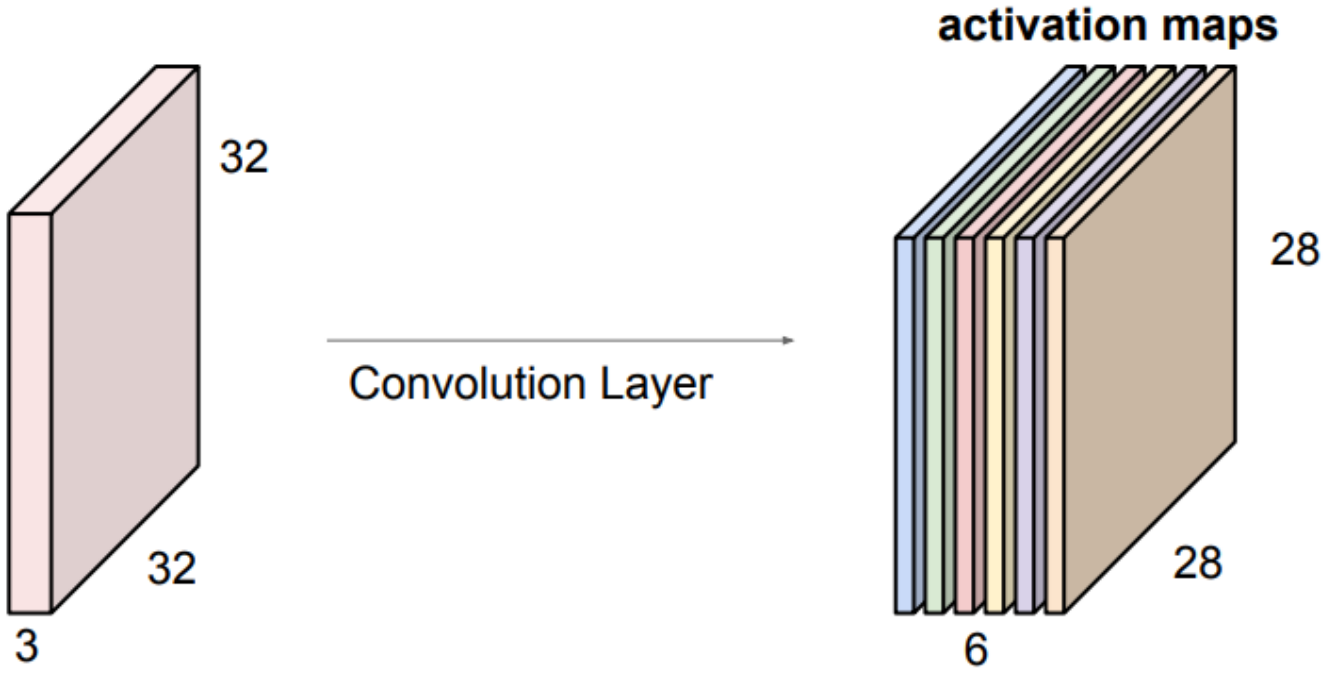
$$w^T x + b$$

Convolution Layer



Activation Maps

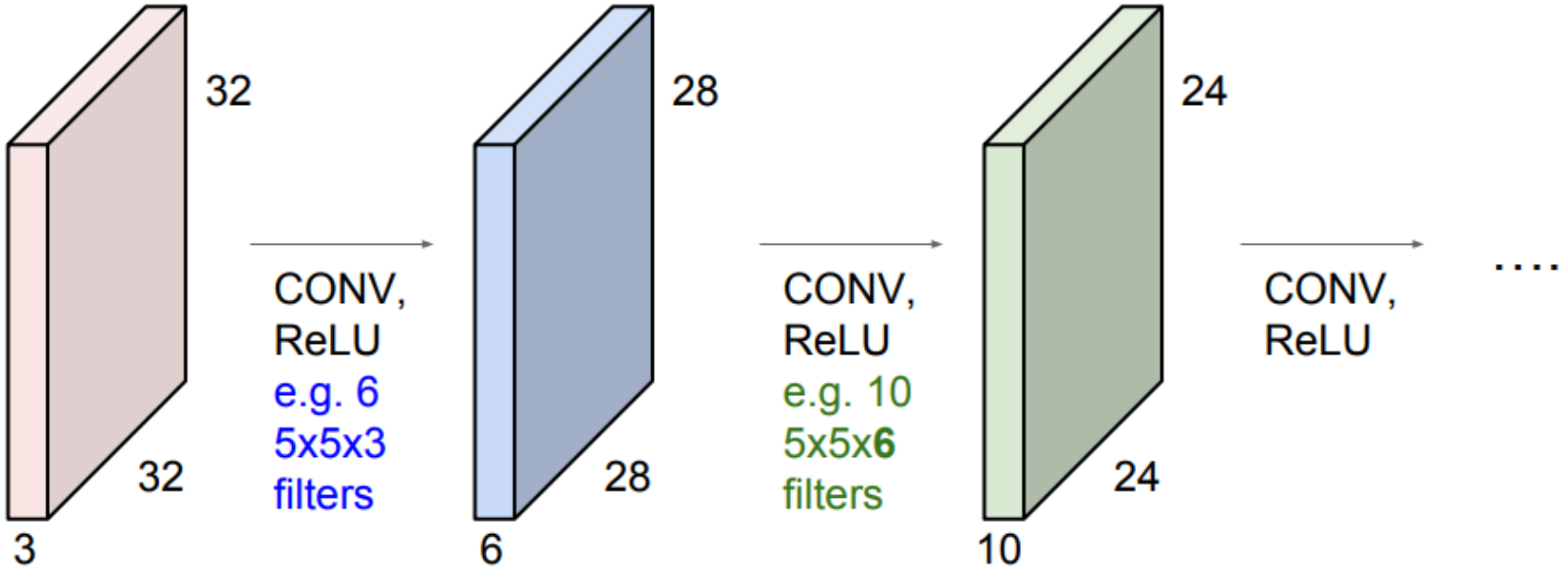
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



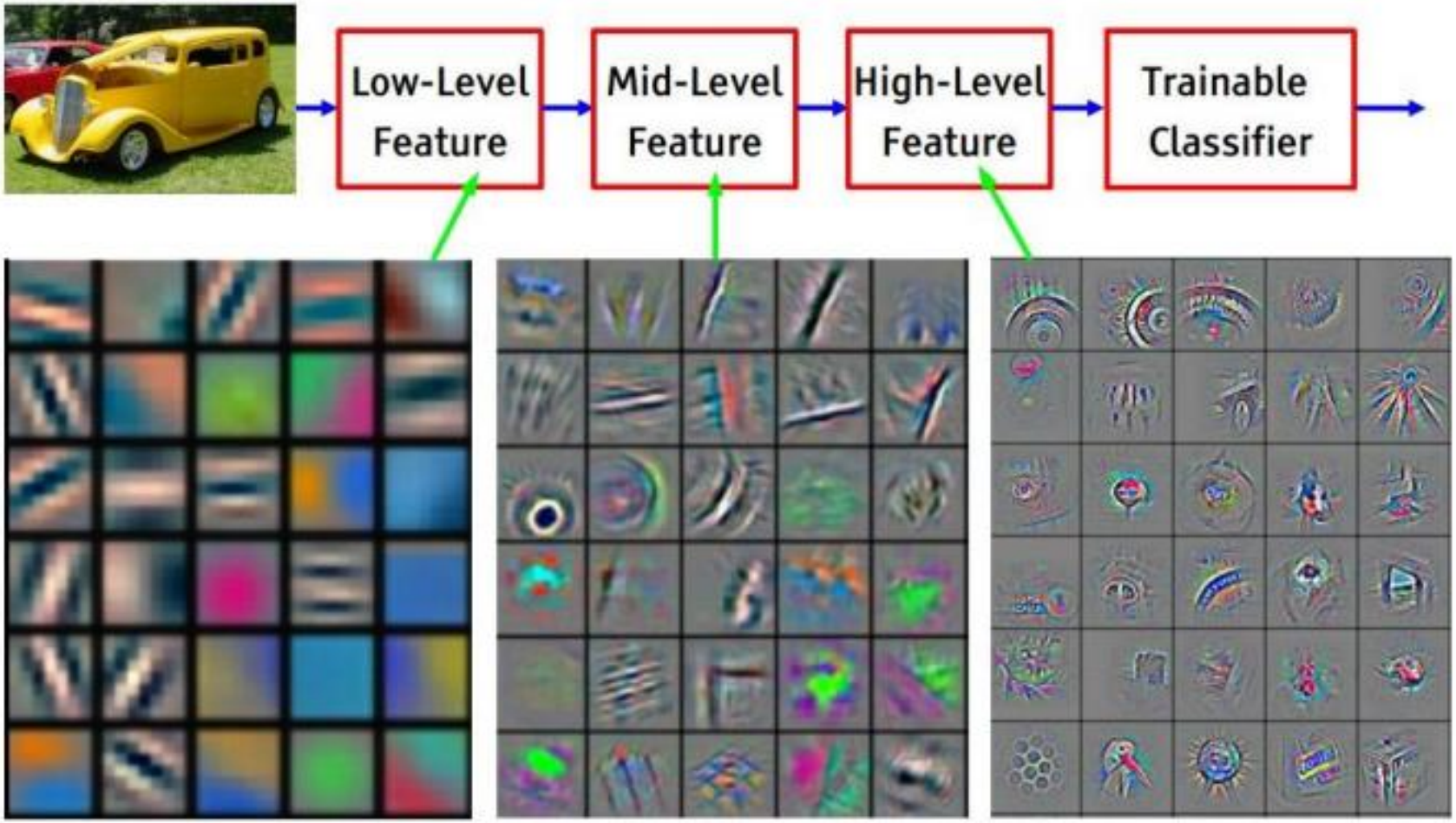
We stack these up to get a "new image" of size 28x28x6!

Sequence of Convolution Layers

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

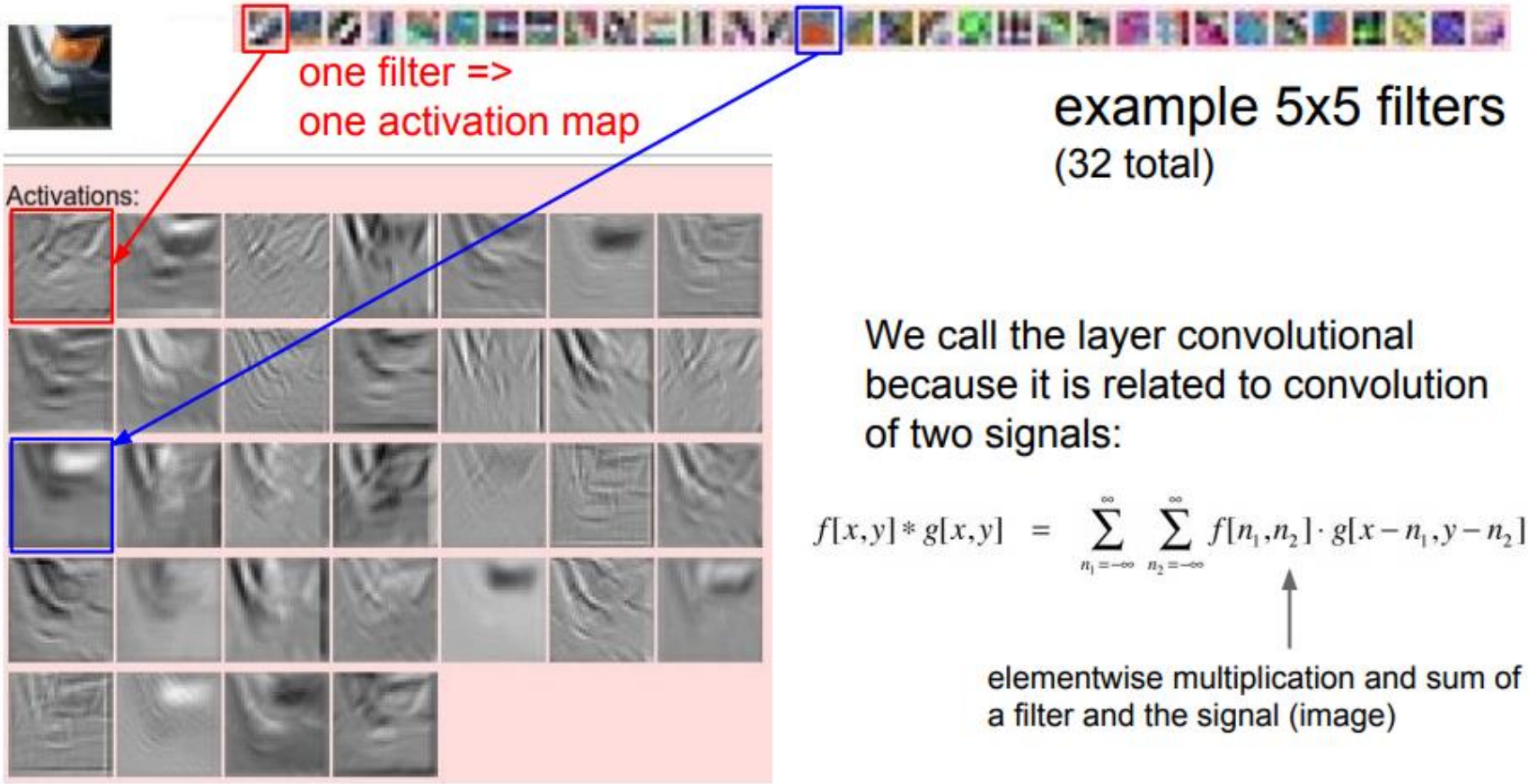


Hierarchy of Features



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Visualizing Activations



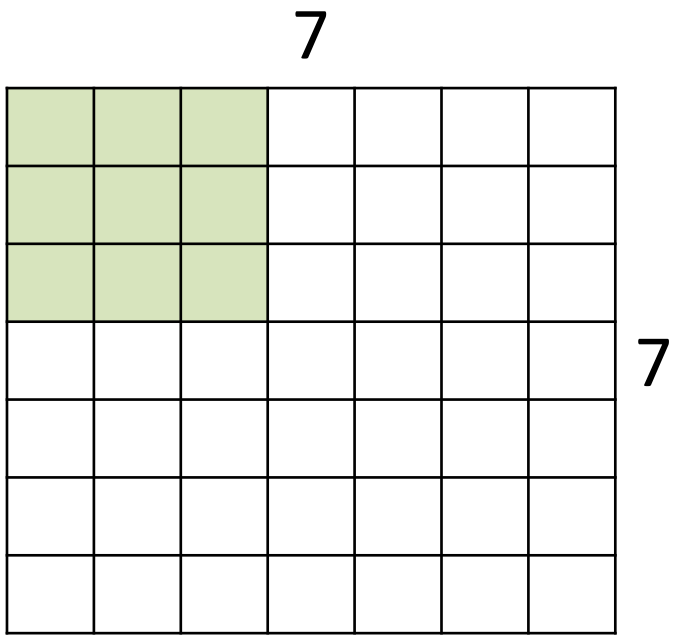
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

Strides

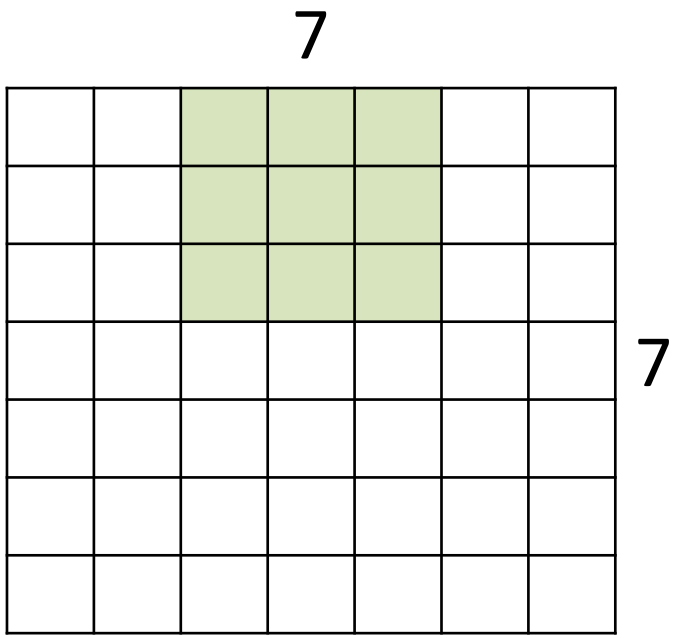
- A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Strides

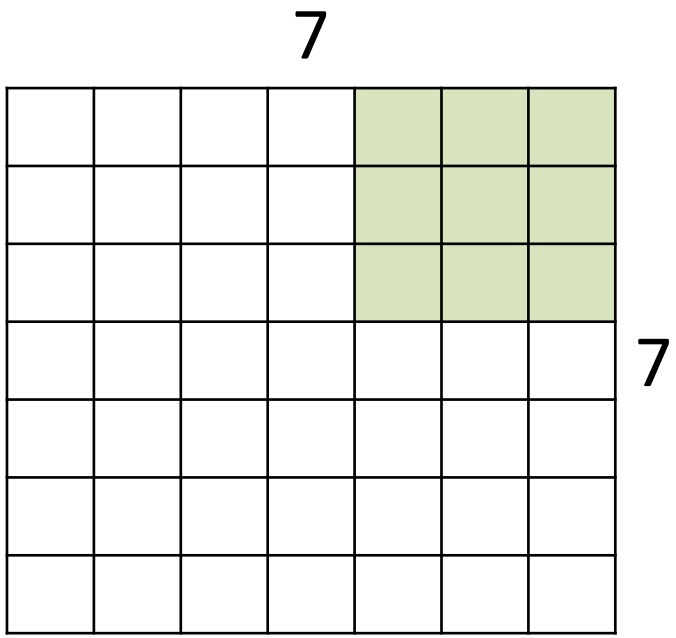
- A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Strides

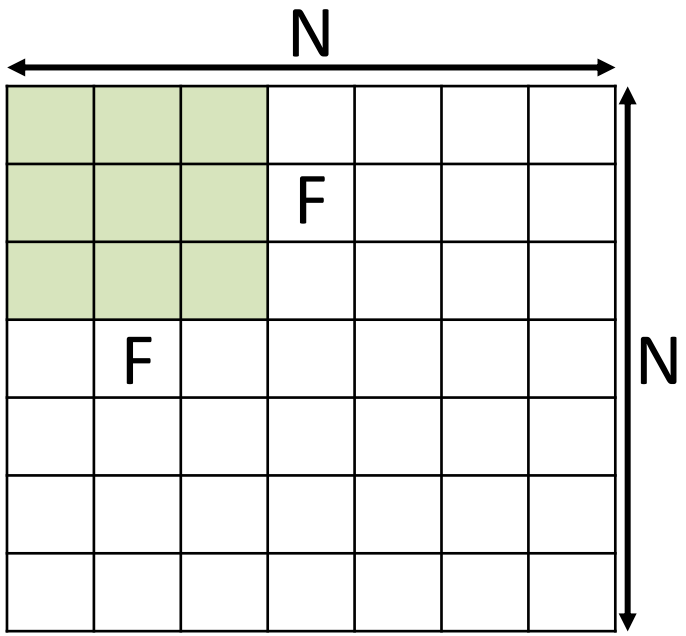
- A closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output

Strides

- Not all values can fit



Output size:

(N – F) / stride + 1

e.g. N = 7, F = 3:

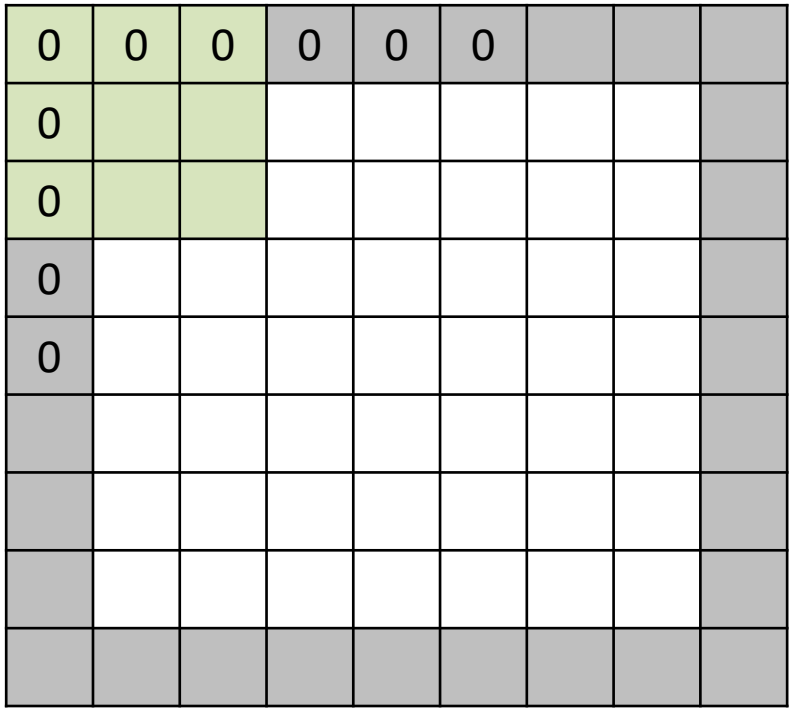
stride 1 => (7 – 3)/1 + 1 = 5

stride 2 => (7 – 3)/2 + 1 = 3

stride 3 => (7 – 3)/3 + 1 = 2.33

Padding

- In practice: Common to zero pad the border



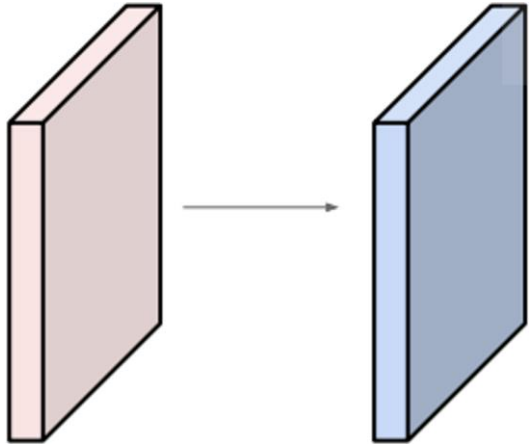
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border
=> What is the output?

7x7 output

In general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2.
(Preserve spatial size)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Revision

- Example:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size? 32x32x10

Number of parameters in this layer?

(5x5x3+1) x 10

Setting Convolution Layer

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

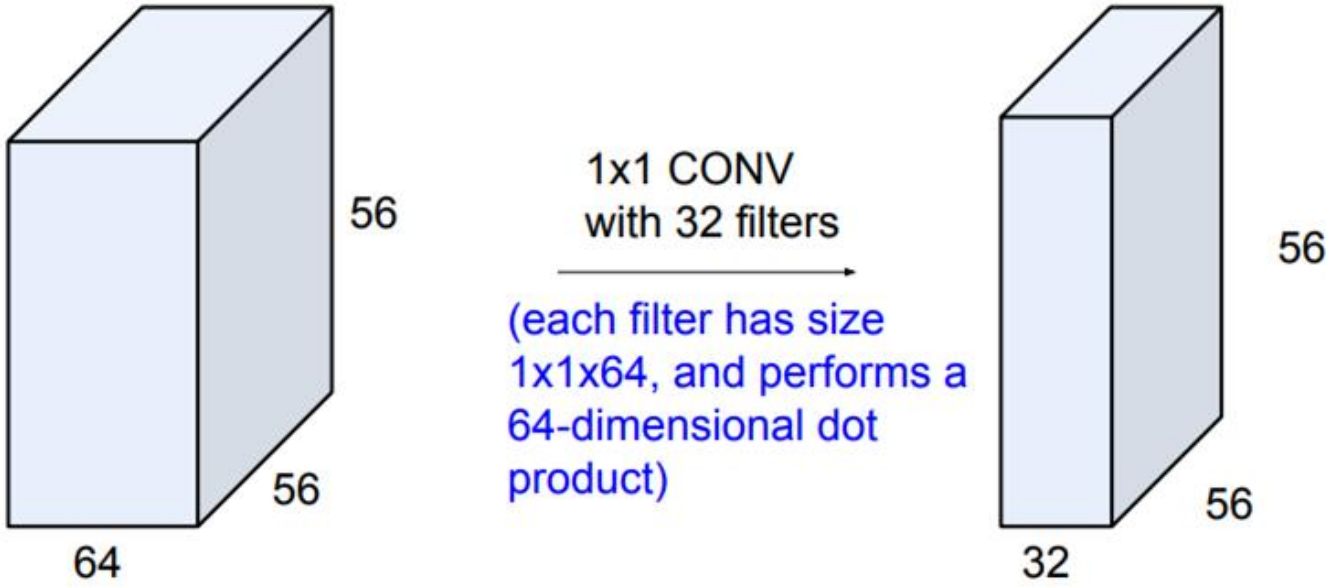
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

From Karpathy's notes

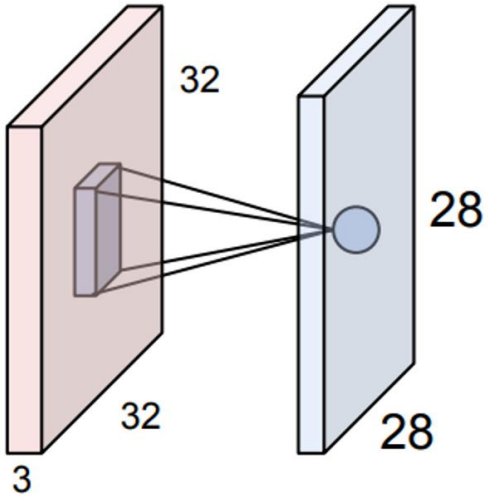
1x1 Convolution

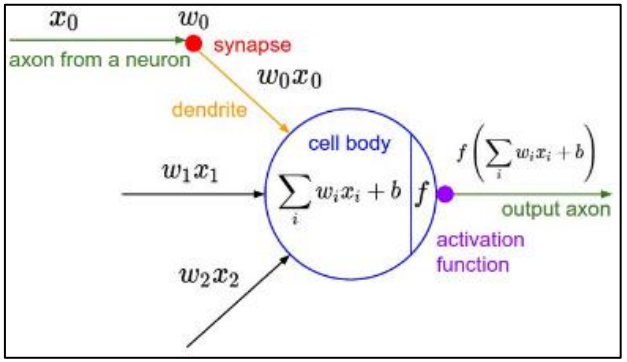
- Usually for direct mapping



Convs are Neural Connections

- The brain/neuron view of CONV Layer





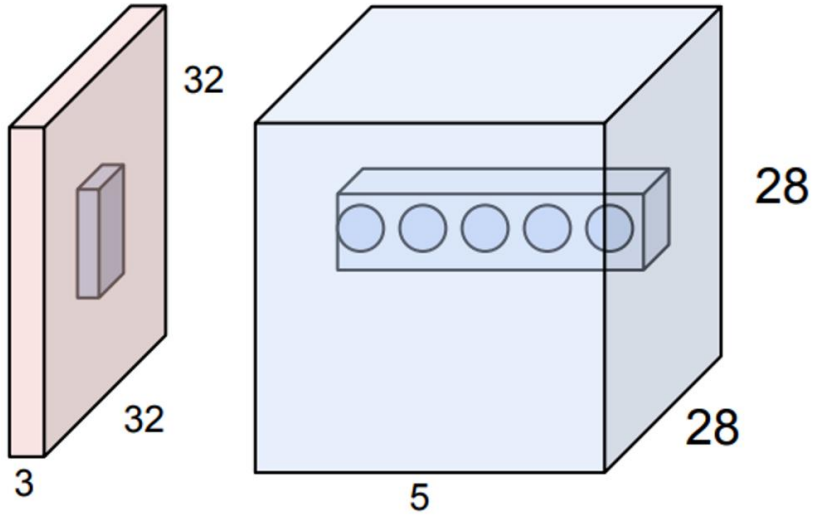
An activation map is a 28x28 sheet of neuron outputs:

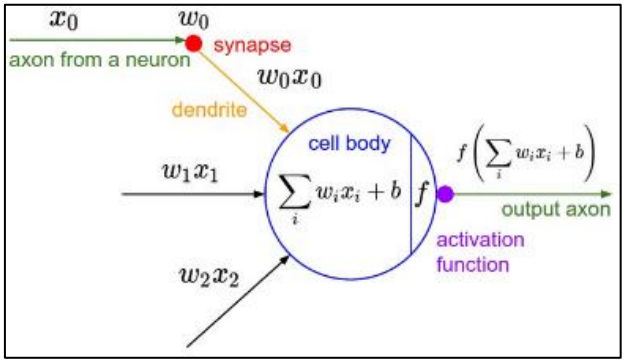
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" => "5x5 receptive field for each neuron"

Convs are Neural Connections

- The brain/neuron view of CONV Layer



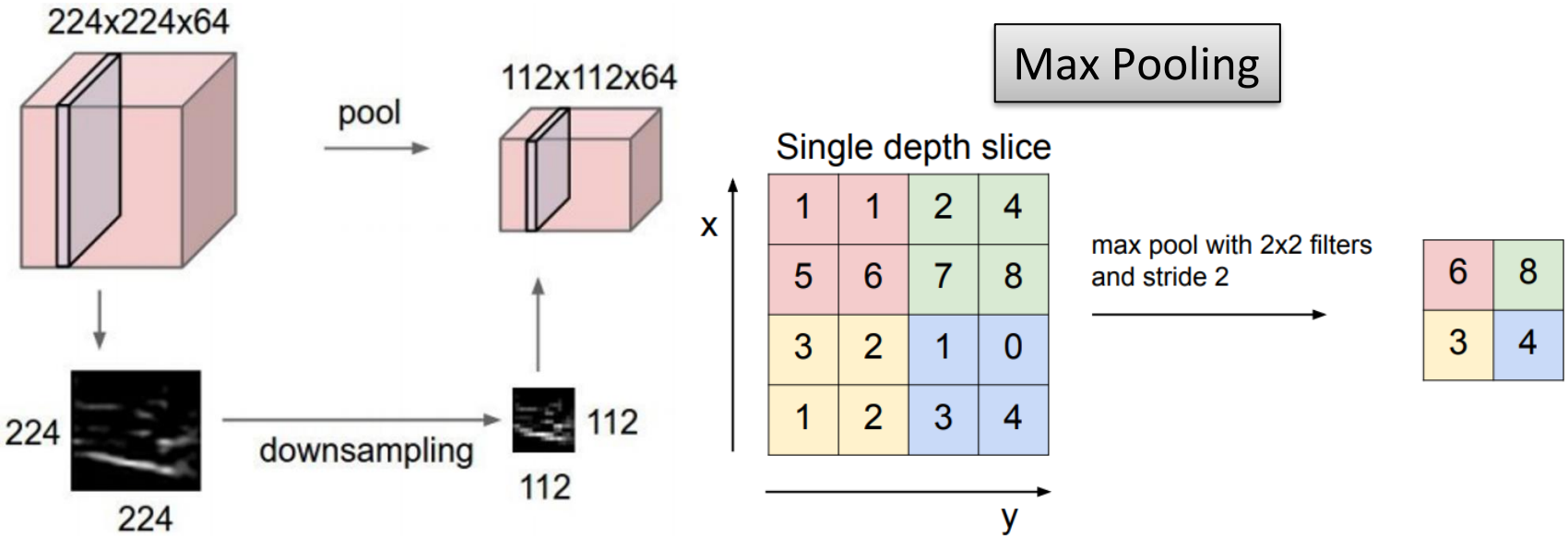


E.g. with filters, CONV layer consists of neurons arrange in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

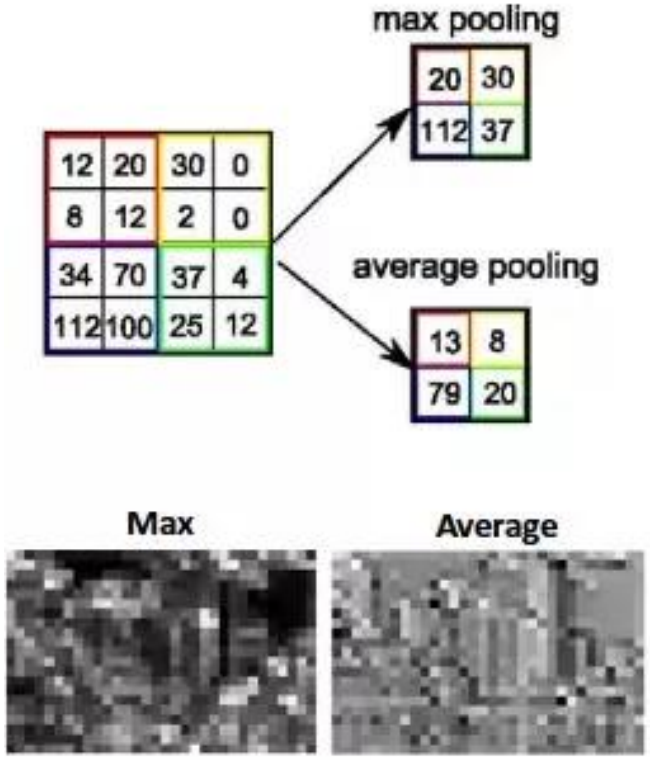
Pooling Layer

- Makes the representations smaller and more manageable
- Operates over each activation map independently:



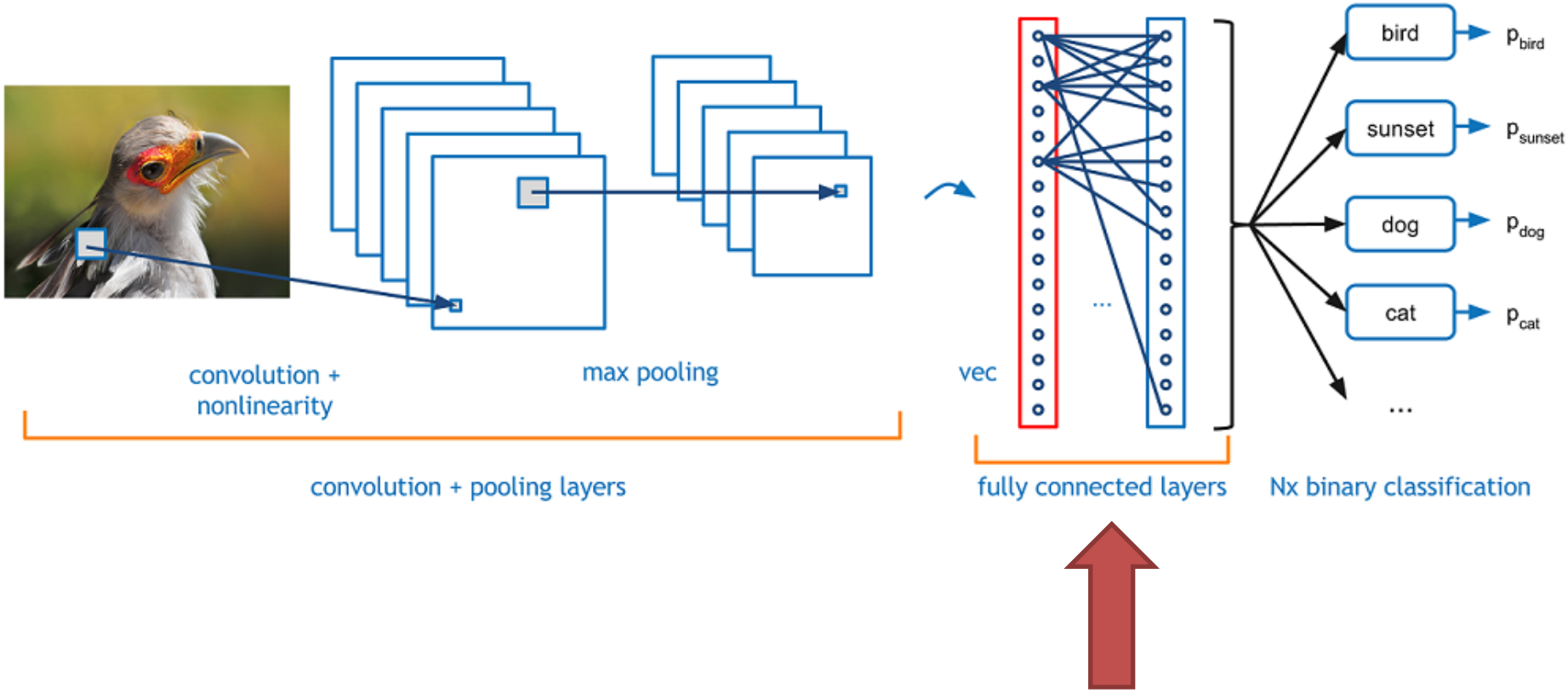
Motivation of Pooling

- To reduce variance and computation complexity
  - E.g. 2x2 max/average pooling reduces 75% data
- To extract low level features from neighborhood

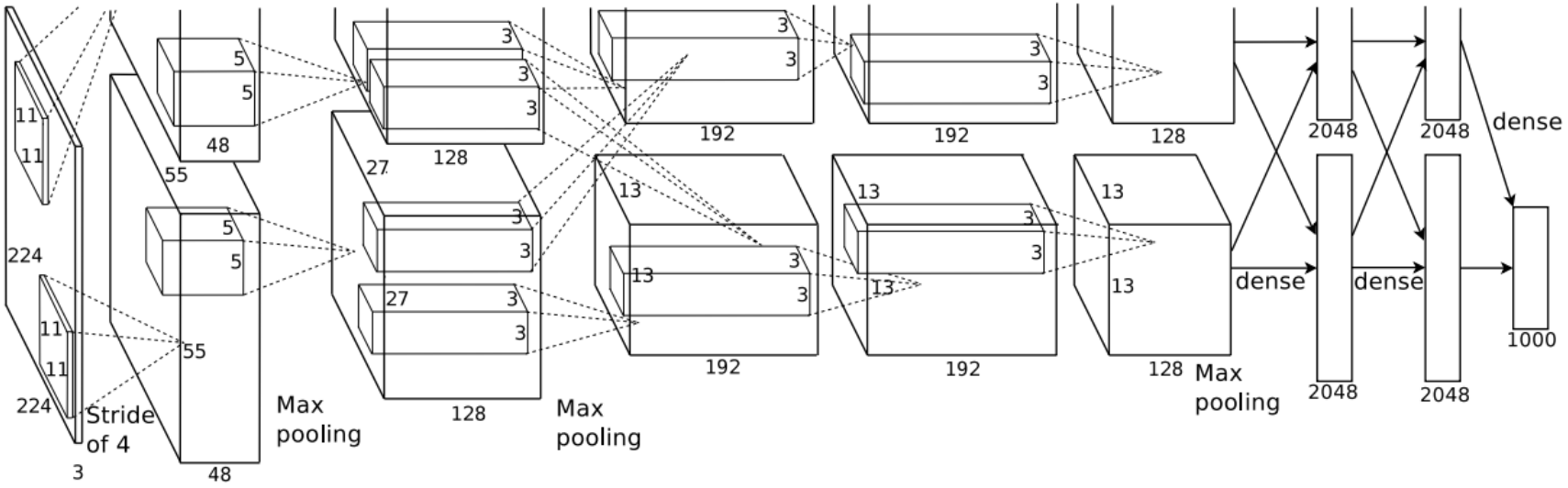


Fully Connected (FC) Layer

- The “classifier” part of the network



Case Study: AlexNet

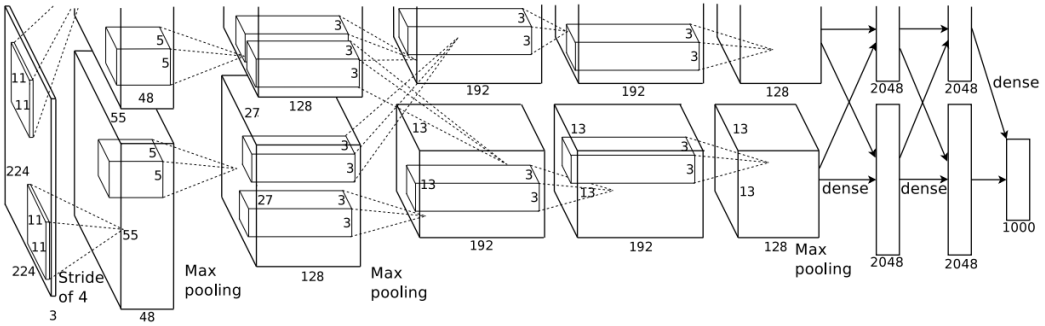


- Input: 227x227x3 image
- First layer (CONV1): 96 11x11 filters applied at stride 4
  - What is the output volume size? (227 – 11)/4 + 1 = 55
  - What is the total number of parameters in this layer? (11x11x3+1) x 96 = 34,944
- After CONV1: 55x55x96
- Second layer (POOL1): 3x3 filters applied at stride 2
  - Output volume: 27x27x96
  - Parameters: 0

Output size:
(N – F) / stride + 1

[*Krizhevsky et al. 2012*]

Case Study: AlexNet



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55X5596] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

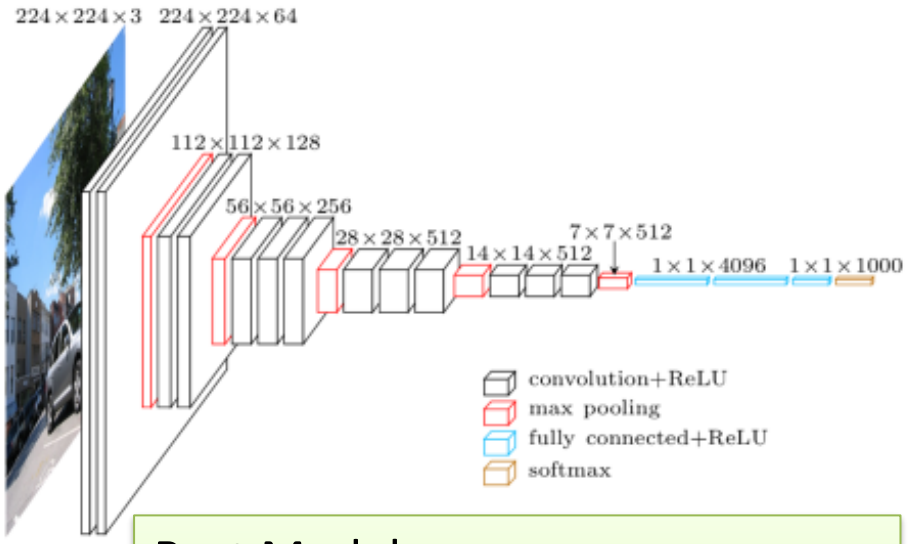
[1000] FC8: 1000 neurons

Details/Retrospectives:

- First use of ReLU
- Used Norm layers (not common anymore)
- Heavy data augmentation
- Dropout 0.5
- Batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% => 15.4%

[*Krizhevsky et al. 2012*]

Case Study: VGGNet

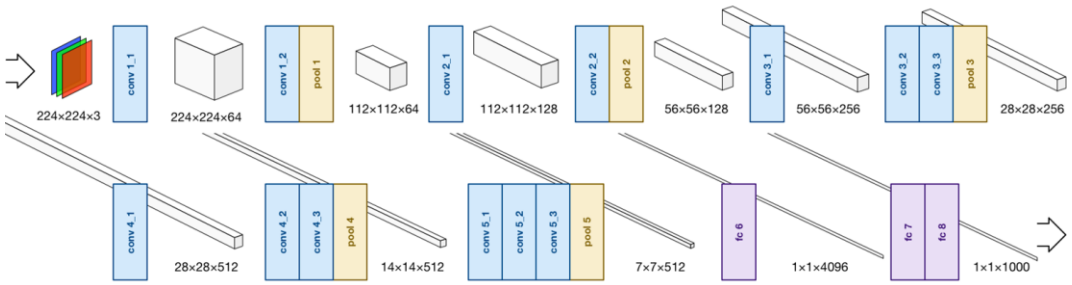


Best Model:
Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2
11.2% top 5 error in ILSVRC 2013 =>
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144



[*Simonyan and Zisserman, 2014*]

Case Study: VGGNet

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

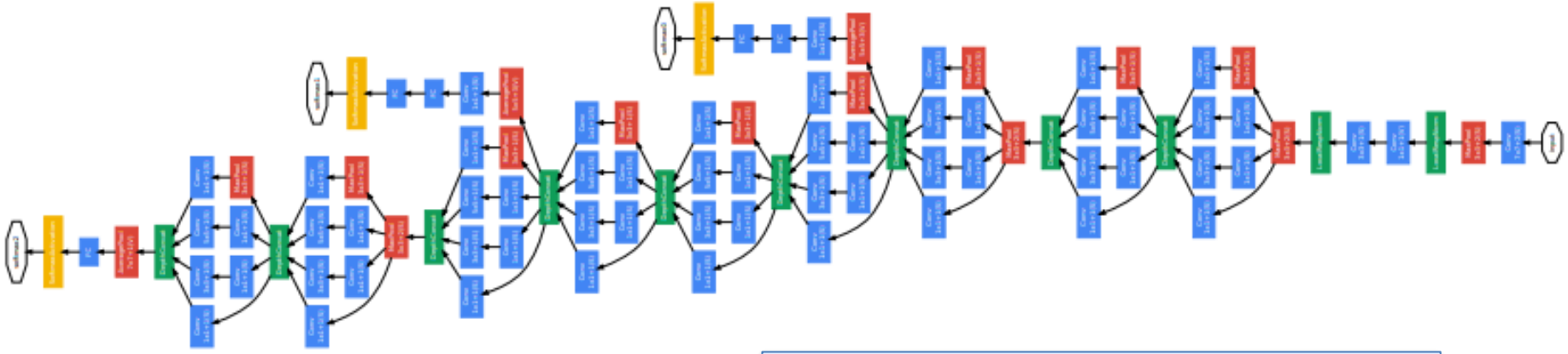
Note:

Most memory is in early CONV

Most params are in late FC

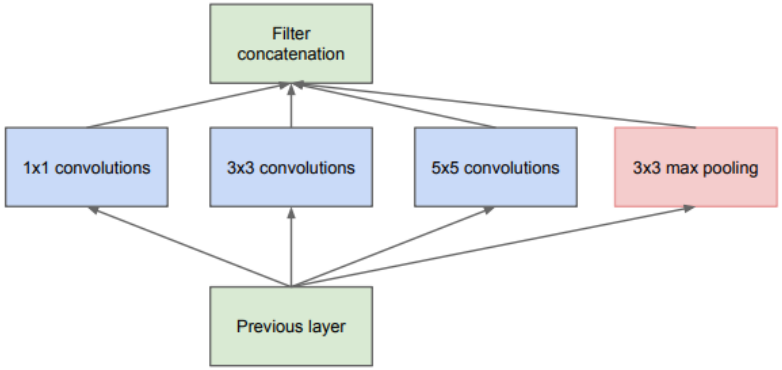
[*Simonyan and Zisserman, 2014*]

Case Study: GoogLeNet

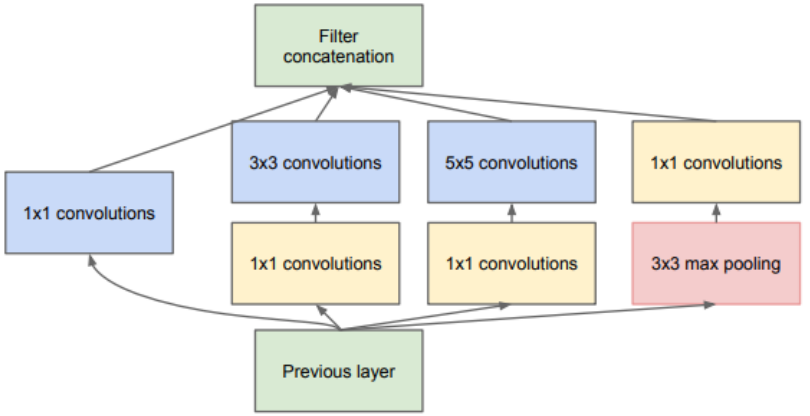


ILSVRC 2014 winner (6.7% top 5 error)

Inception Modules:



Naïve version



With dimension reduction

[*Szegedy et al., 2014*]

Case Study: GoogLeNet

Notable features:

Only 5 million parameters
(Removes FC layers completely)

Compared to AlexNet:

- 12x less parameters
- 2x more compute
- 6.67% (vs. 16.4%) error

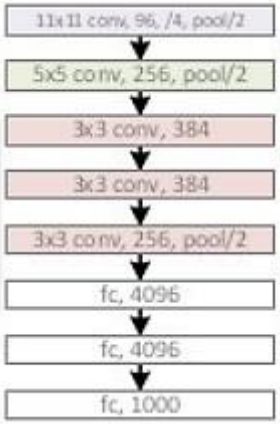
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

[*Szegedy et al., 2014*]

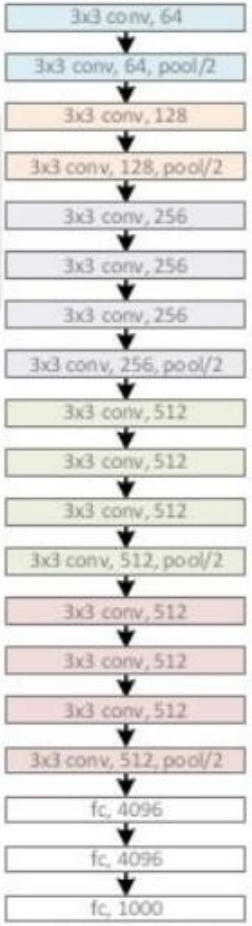
Case Study: ResNet

- Revolution of Depth

AlexNet
8 layers
(ILSVRC 2012)



VGGNet
19 layers
(ILSVRC 2014)



GoogLeNet
22 layers
(ILSVRC 2014)

Case Study: ResNet

- Revolution of Depth

AlexNet
8 layers
(ILSVRC 2012)

VGGNet
19 layers
(ILSVRC 2014)

ResNet
152 layers
(ILSVRC 2015)

ILSVRC 2015 winner (3.6% top 5 error)
2-3 weeks of training on 8 GPU machine

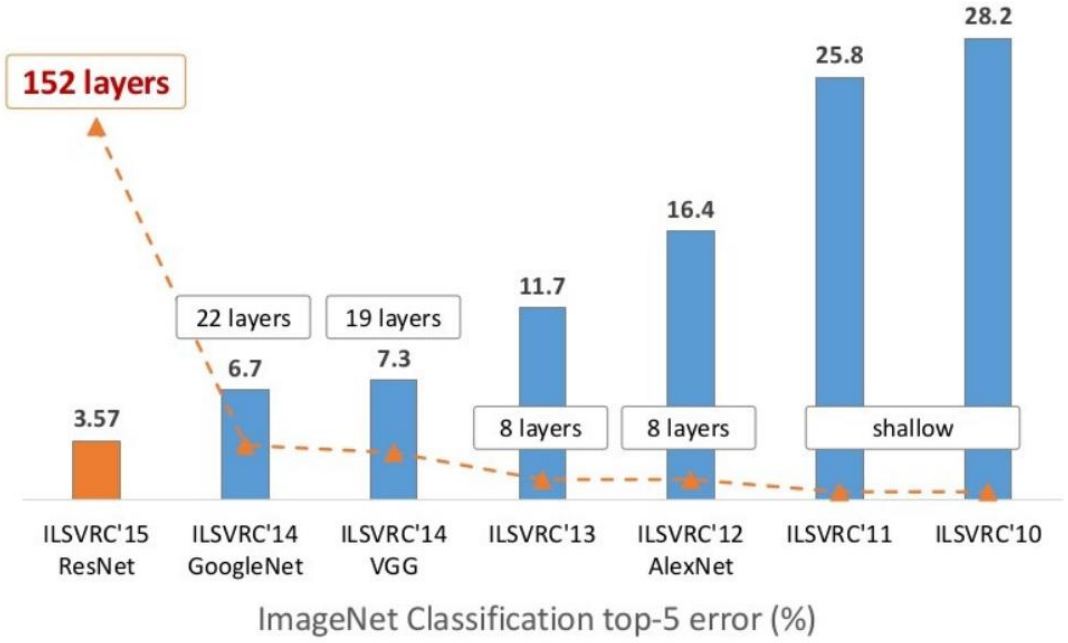
At runtime:
Faster than a VGGNet
(even though it has 8x more layers)

[*He et al., 2015*]

Case Study: ResNet

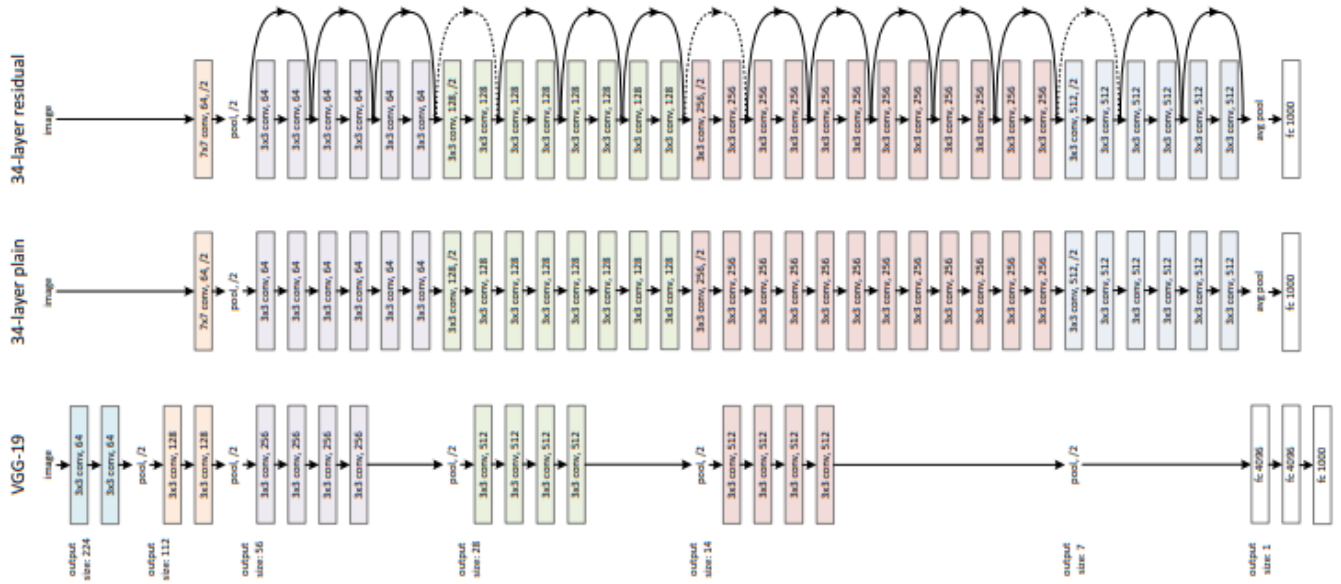
MSRA @ ILSVRC & COCO 2015 Competitions

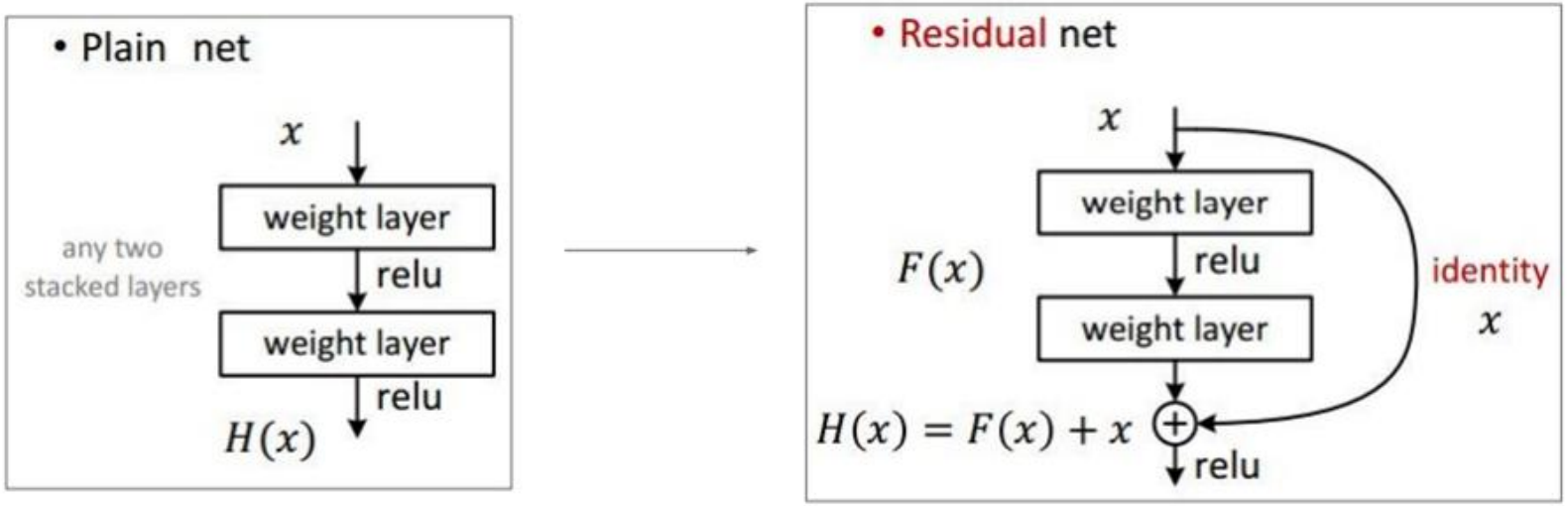
- 1st places in all five main tracks
  - ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd



[*He et al., 2015,* https://www.youtube.com/watch?v=1PGLj-uKT1w]

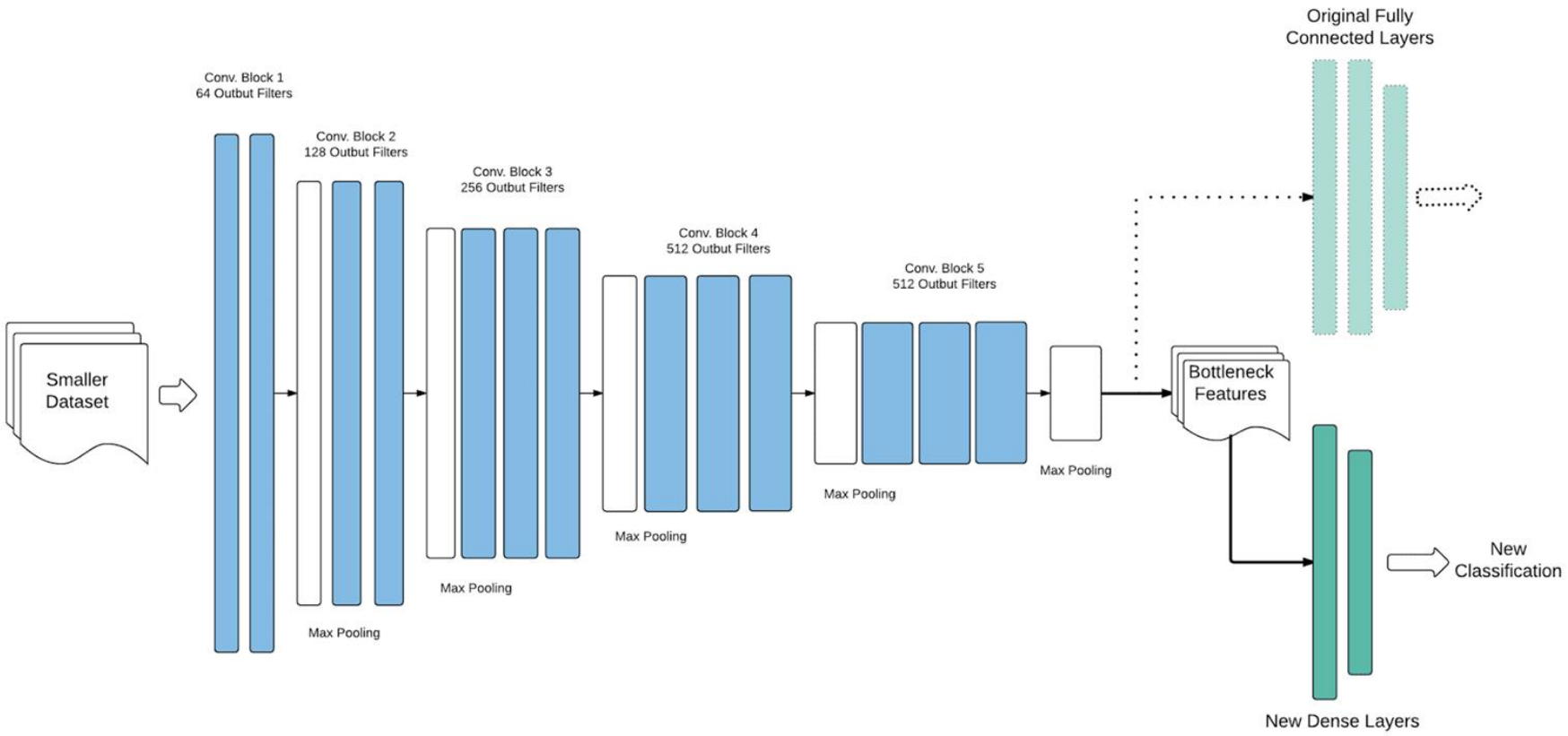
Case Study: ResNet





Transfer Learning

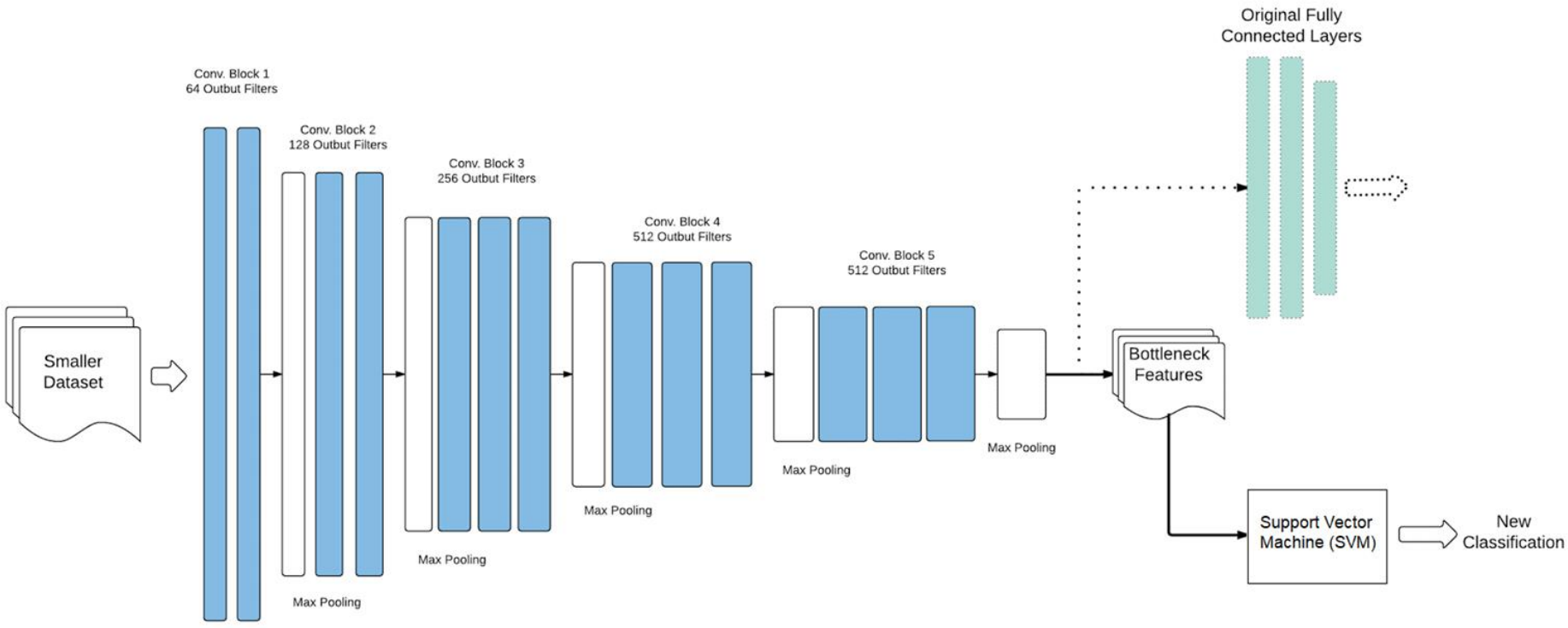
- Use the bottleneck features of a pre-trained network
- Fine-tune top layers of a pre-trained network or create new top layers



Reference: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

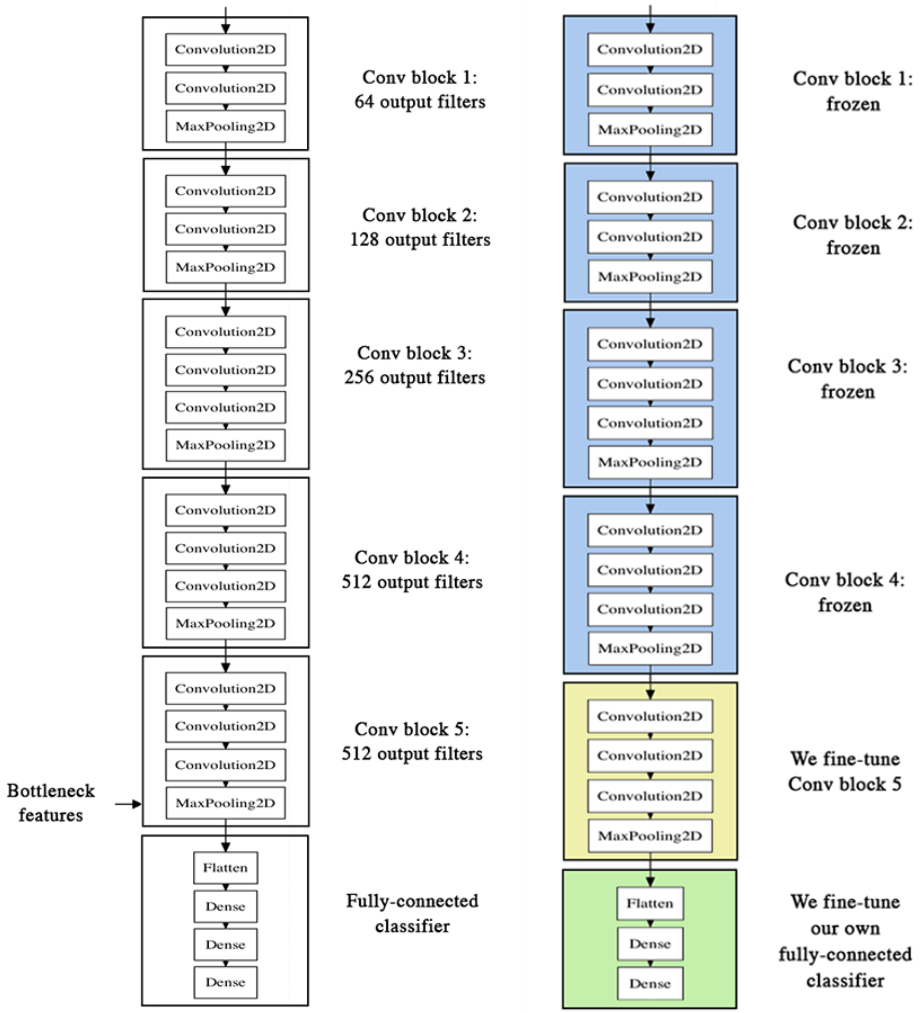
Transfer Learning

- Use the bottleneck features of a pre-trained network
- Pass the bottleneck features to traditional ML algorithm for prediction

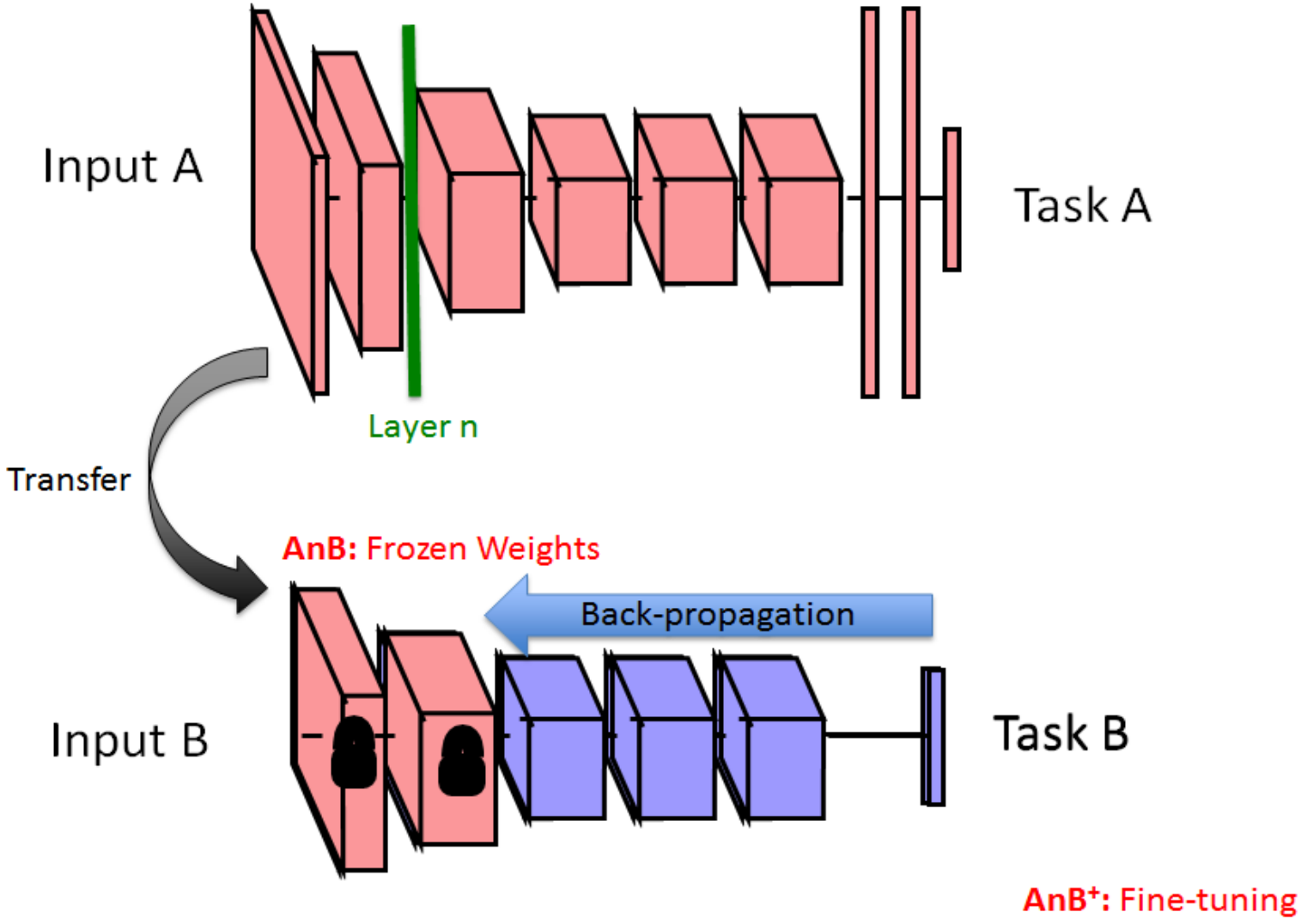


Transfer Learning

- Unfreeze the pre-trained model and let it adapt more to the task at hand.
- Making some fine adjustments to further improve performance – more commonly known as Fine Tuning.



Transfer Learning



Summary

- Machine Learning Pipeline
  - Data partitioning
  - Feature extraction
  - Generalization
- Deep Learning
  - Neural Networks
  - CNNs stack together Conv, Pool, and FC layers
  - Trend: Deeper networks, smaller filters
  - Trend: Get rid of Pooling/FC layers

Further Studies and Resources

- Deep Learning Courses
  - deeplearning.ai: https://www.deeplearning.ai/
  - Stanford: http://cs231n.stanford.edu/
  - MIT: https://deeplearning.mit.edu/
  - Coursera: https://www.coursera.org/specializations/deep-learning
- YouTube Lectures
  - Assoc. Prof. Lee Hung-Yi, National Taiwan University (NTU): https://www.youtube.com/@HungyiLeeNTU/playlists
- Autonomous Driving
  - Stanford Autonomous Driving Team: http://driving.stanford.edu/
- Visualization of deep features: https://github.com/jacobgil/keras-grad-cam