ARTIFICIAL INTELLIGENCE

INTRODUCTION TO DEEP LEARNING

MUSTAFA ALDEMIR, INTEL TURKEY

ARTIFICIAL INTELLIGENCE

WHAT IS DEEP LEARNING GOOD FOR

DEEP LEARNING: EXAMPLES

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Images
Computer vision, Image classification, Traffic sign detection, Pedestrian detection, localization…

Sound
Speech recognition, Natual Language Processing, Translation, Content captioning, speaker identification

Text
Natual Language Processing, text classification; web search, spam, email filtering

CLASSIFICATION

-> Label the image

Person

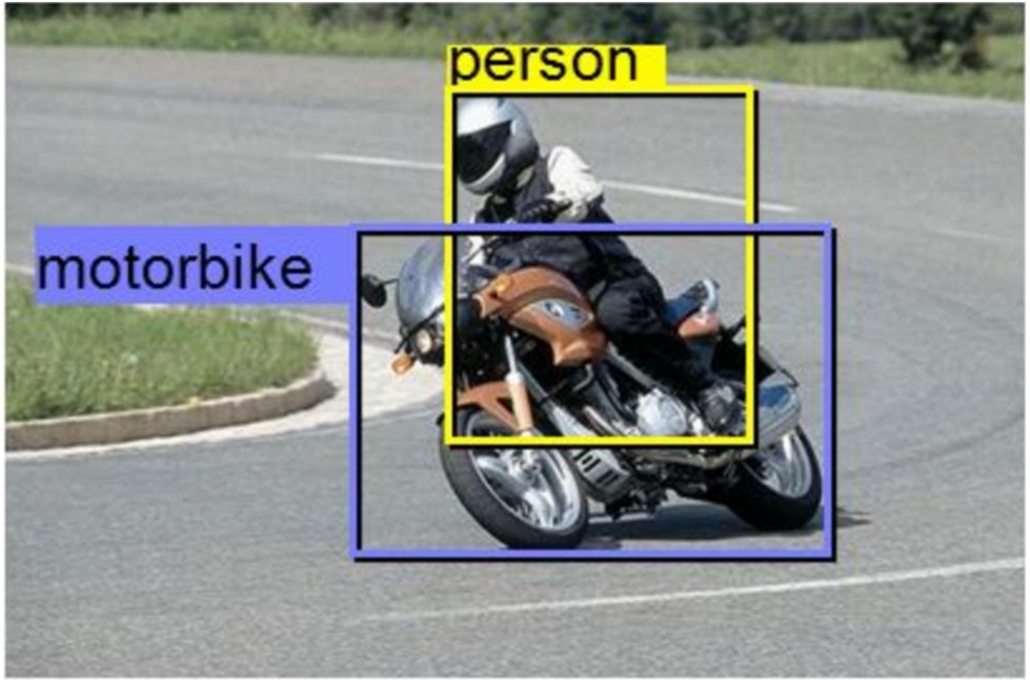
Motorcyclist

Bike

https://people.eecs.berkeley.edu/~jhoffman/talks/lsda-baylearn2014.pdf

DETECTION

-> Detect and label



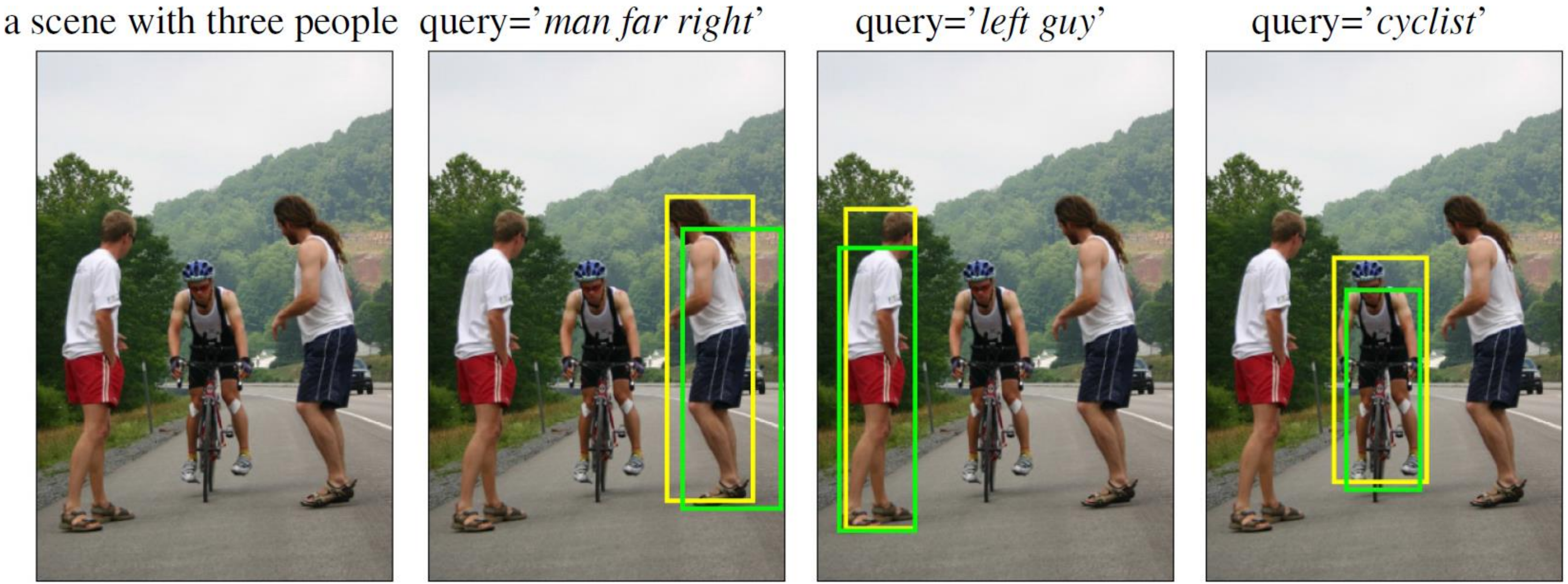
https://people.eecs.berkeley.edu/~jhoffman/talks/lsda-baylearn2014.pdf

SEMANTIC SEGMENTATION

-> Label every pixel

https://people.eecs.berkeley.edu/~jhoffman/talks/lsda-baylearn2014.pdf

NATURAL LANGUAGE OBJECT RETRIEVAL



http://arxiv.org/pdf/1511.04164v3.pdf

SPEECH RECOGNITION

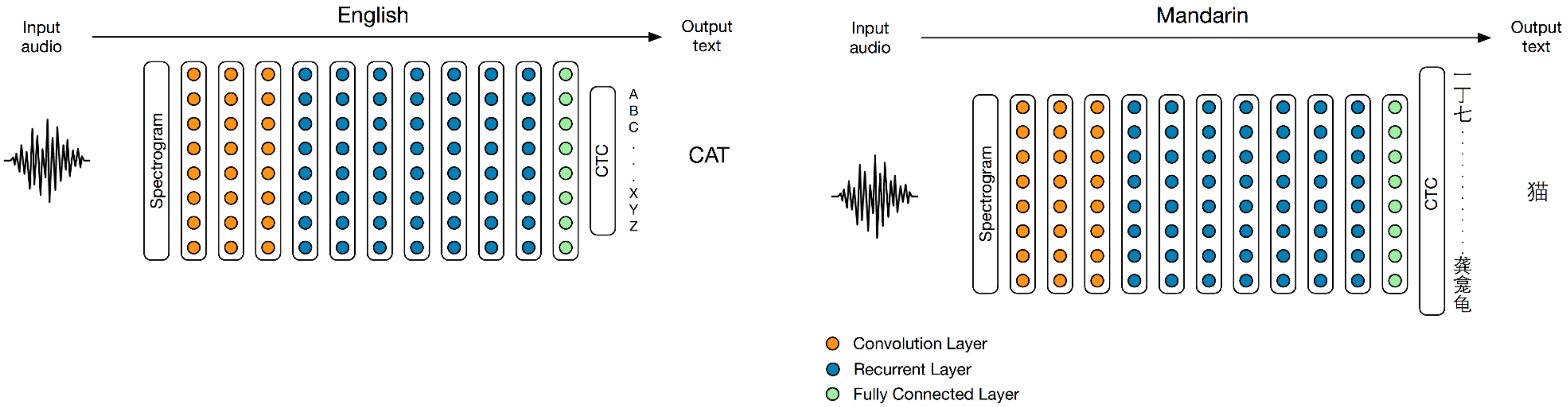


IMAGE / VIDEO CAPTIONING

Describes without errors	Describes with minor errors	Somewhat related to the image

A person riding a motorcycle on a dirt road.

Two dogs play in the grass.

A skateboarder does a trick on a ramp.

A group of young people playing a game of frisbee.

Two hockey players are fighting over the puck.

A little girl in a pink hat is blowing bubbles.

RECURRENT NEURAL NETWORKS

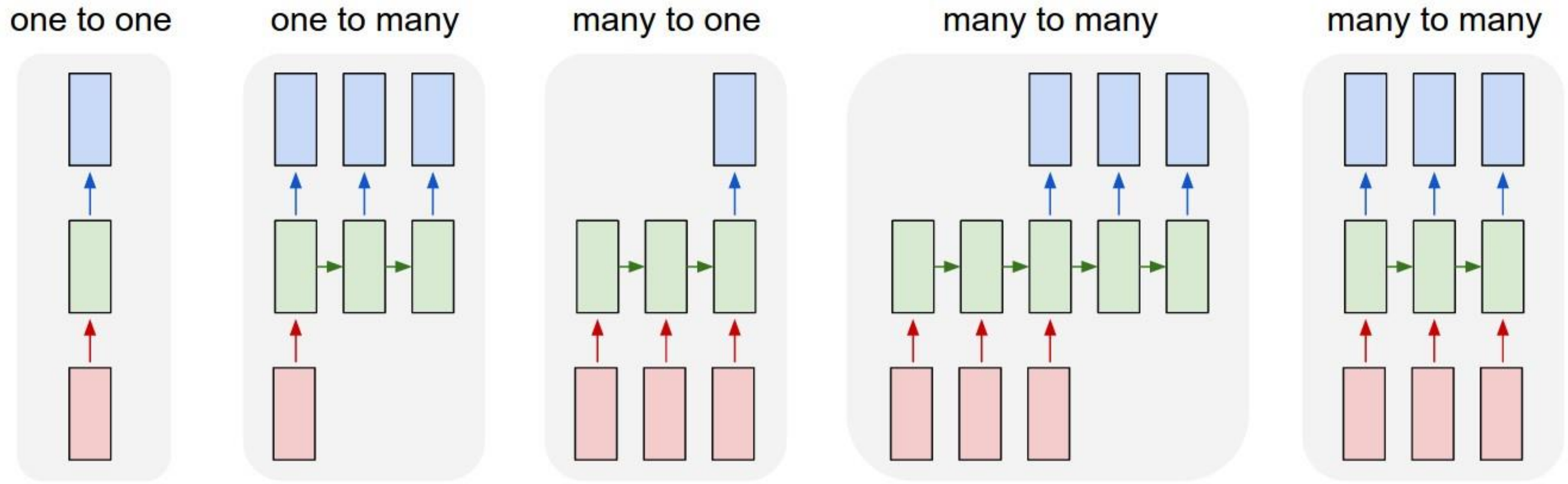


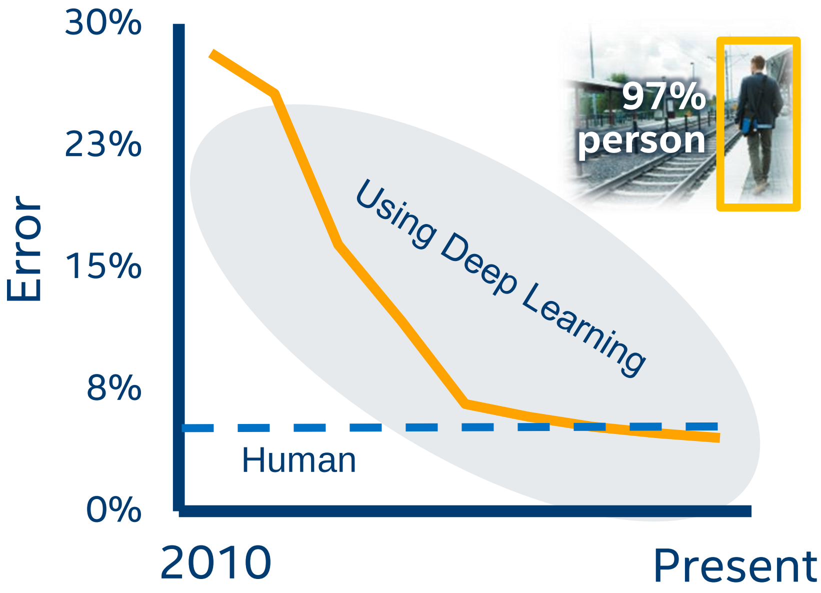
Image Classification | Image Captioning | Sentiment Analysis | Google Translate | Video Frame Classification

Red: input; Blue: output, Green: RNN state

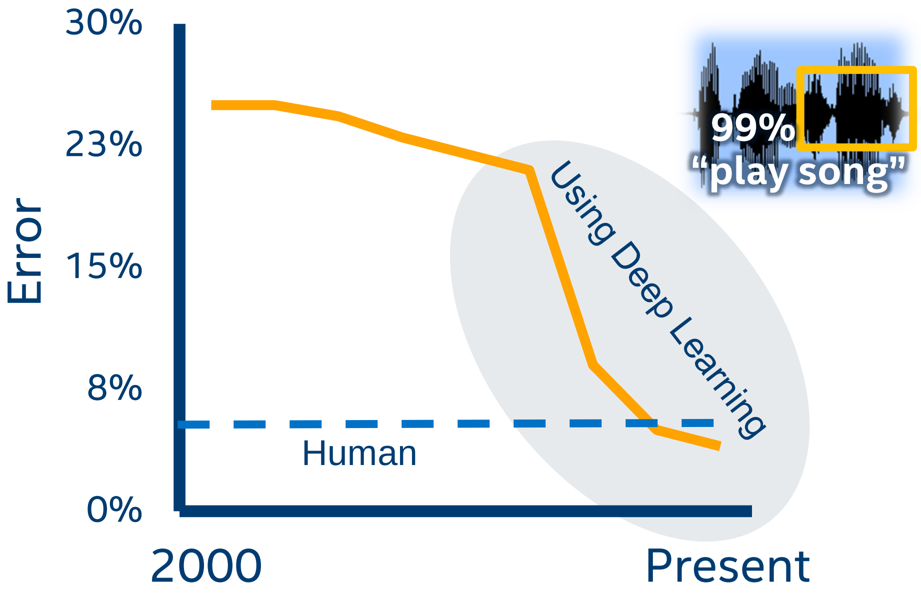
GENERATIVE ADVERSARIAL NETWORKS

DEEP LEARNING BREAKTHROUGHS

IMAGE RECOGNITION



SPEECH RECOGNITION

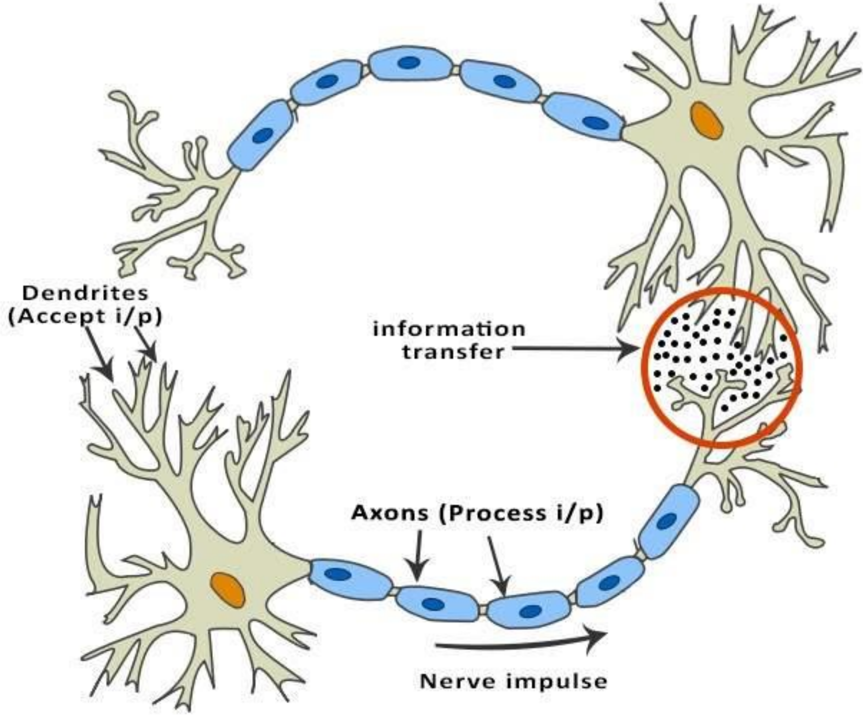


These and more are enabling new & improved applications

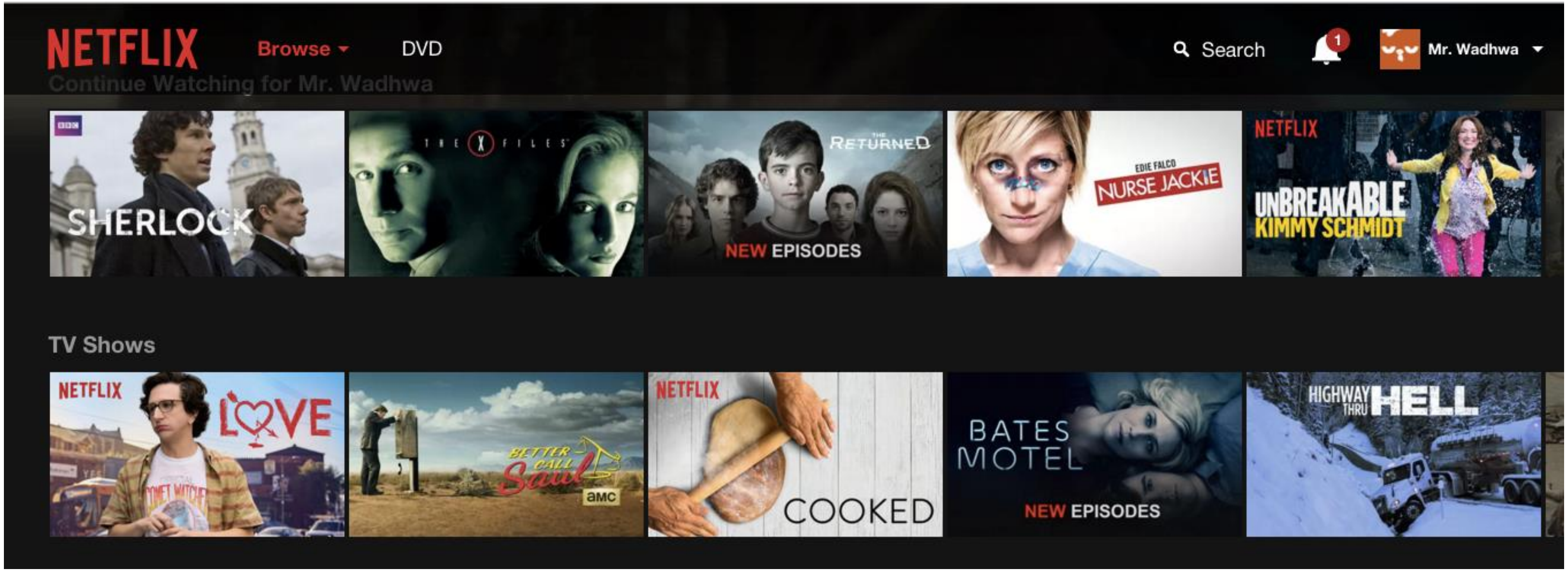
ARTIFICIAL INTELLIGENCE

NEURAL NETWORKS

INSPIRED BY HUMAN BRAIN



CASE STUDY: RECOMMENDATION SYSTEMS

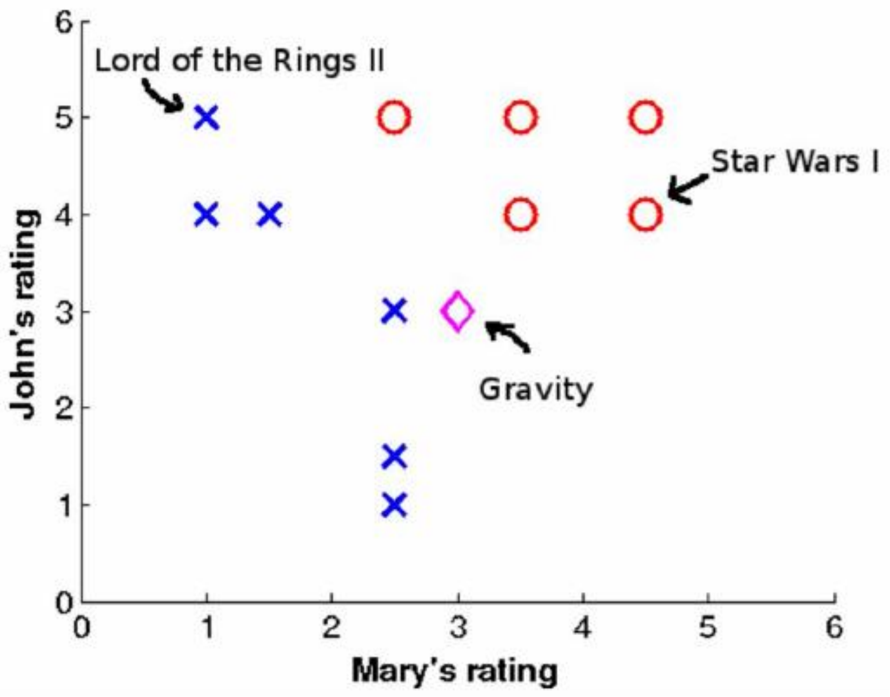


Will Nancy like Gravity?

Let's ask close friends Mary and John, who already watched it and rated between 1-5.

NETFLIX

Movie	Mary's Rating	John's Rating	Does Nancy like?
Lord of the Rings 2	1	5	No
…	…	…	…
Star Wars 1	4.5	4	Yes
Gravity	3	3	?


Lord of the Rings II
Star Wars I
Gravity
John's rating
Mary's rating
0
1
2
3
4
5
6

A decision function can be as simple as weighted linear combination of friends:

$$h_{\theta,b} = \theta_1 x_1 + \theta_2 x_2 + b$$

$$h_{\theta,b} = \theta^T x + b$$

- Labels: “I like it” -> 1 “I don’t like it” -> 0
- Inputs: Mary’s rating, John’s rating

ACTIVATION FUNCTION

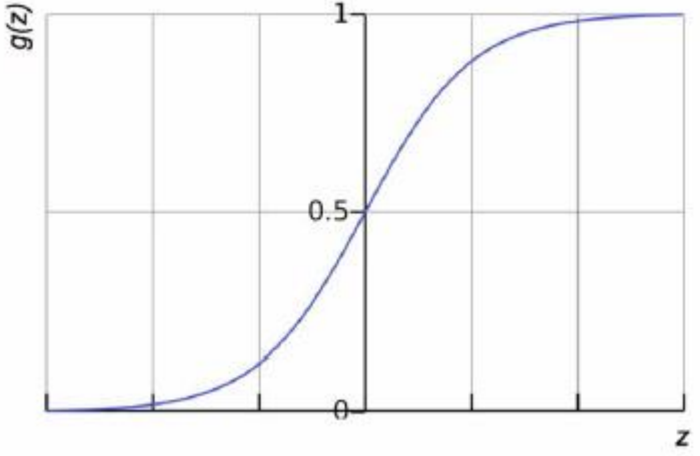
This function has a problem. Its values are unbounded.

We want its output to be in the range of 0 and 1.

$$h_{\theta,b} = g(\theta^T x + b),$$

where $g(z)$ is sigmoid function.

$$g(z) = \frac{1}{1 + exp(-z)}$$



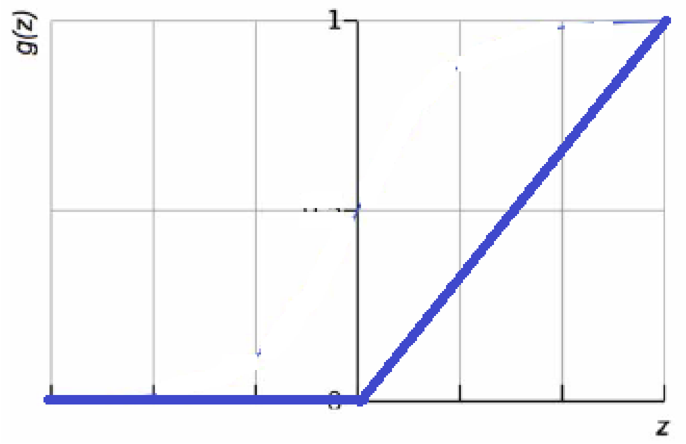
ACTIVATION FUNCTION

This function has a problem. Its values are unbounded.

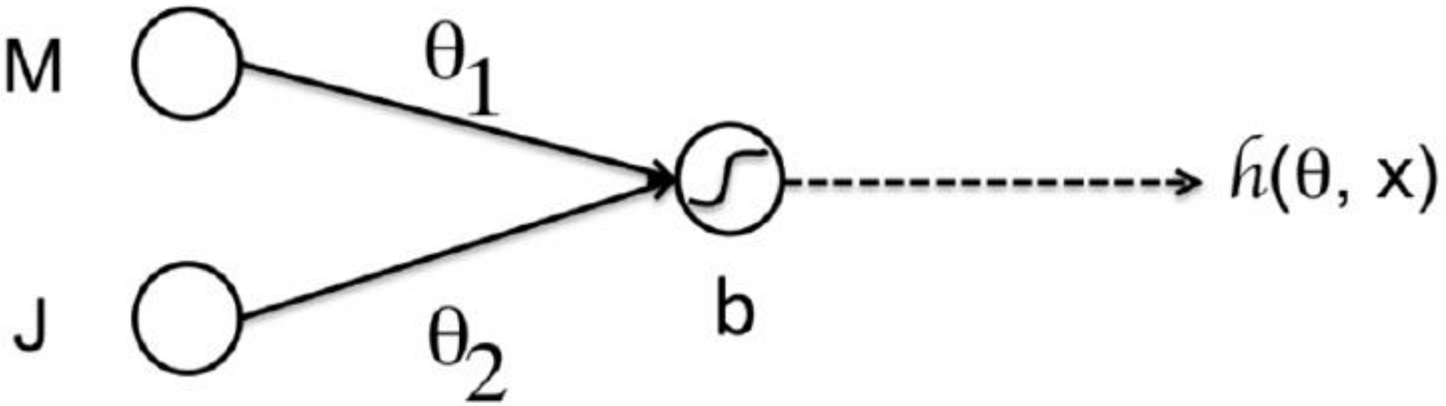
We want its output to be in the range of 0 and 1.

ReLU (Rectified Linear Unit)

f(x) = max(0,x)



ANOTHER WAY OF REPRESENTING THE MODEL



LEARN FROM DATA

We will use the past data to learn θ, b to approximate y. In particular, we want to obtain θ, b such that:

$h_{\theta,b}\left(x^{(1)}\right) \approx y^{(1)}$ where $x^{(1)}$ is my friend's ratings for 1st movie.

$h_{\theta,b}\left(x^{(2)}\right) \approx y^{(2)}$ where $x^{(2)}$ is my friend's ratings for 2nd movie.

…

$h_{\theta,b}\left(x^{(m)}\right) \approx y^{(m)}$ where $x^{(m)}$ is my friend's ratings for mth movie.

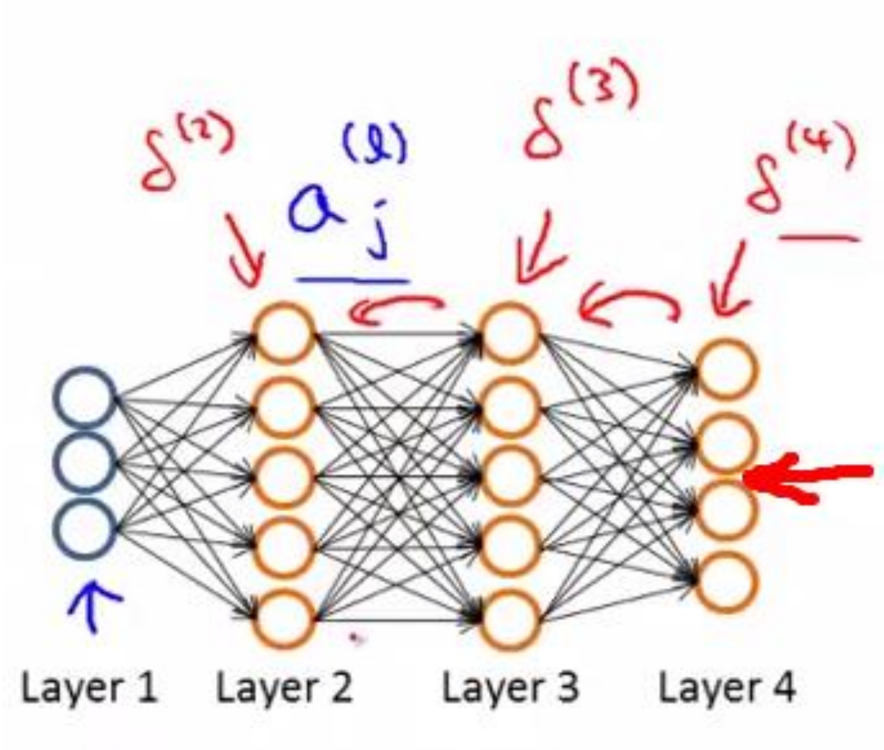
COST FUNCTION

To find values of θ and b we can minimize the following ***cost function***:

$$J(\theta, b) = \left(h_{\theta,b}\left(x^{(1)}\right) - y^{(1)}\right)^2 + \left(h_{\theta,b}\left(x^{(2)}\right) - y^{(2)}\right)^2 + \ \ldots + \left(h_{\theta,b}\left(x^{(m)}\right) - y^{(m)}\right)^2$$

$$J(\theta, b) = \sum_{i=1}^{m} \left(h_{\theta,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

BACKPROPOGATION



STOCHASTIC GRADIENT DESCENT

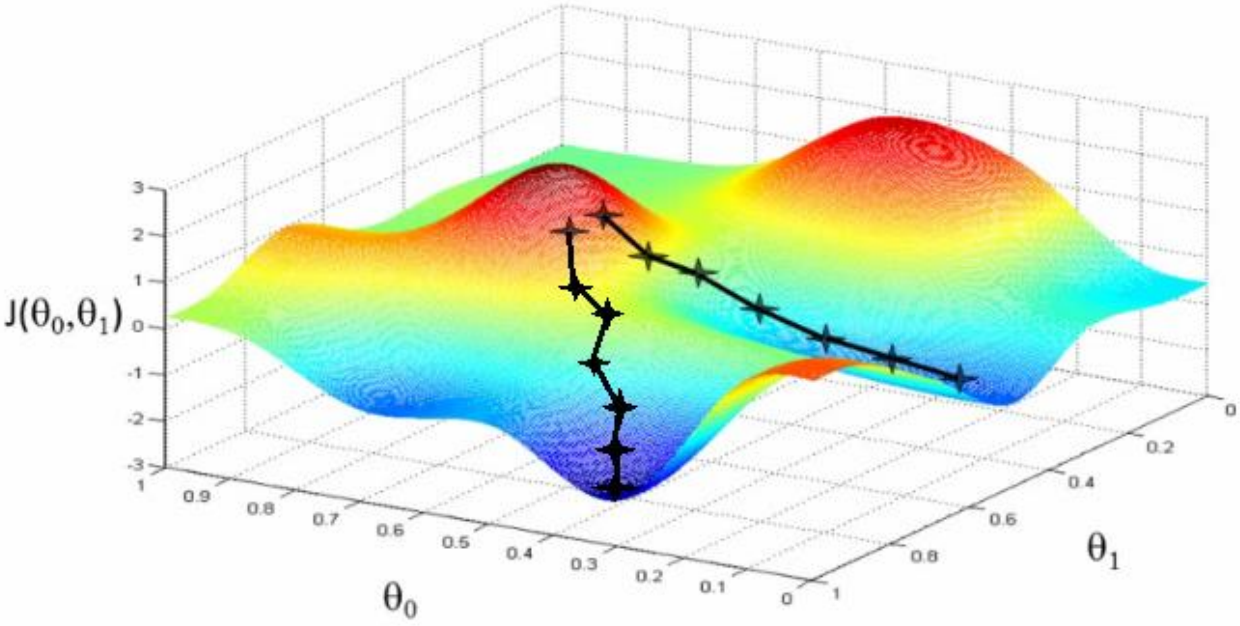
Use Stochastic Gradient Descent (SGD):

$$\theta_1 = \theta_1 - \alpha\Delta\theta_1$$

$$\theta_2 = \theta_2 - \alpha\Delta\theta_2$$

$$b = b - \alpha\Delta b$$

STOCHASTIC GRADIENT DESCENT

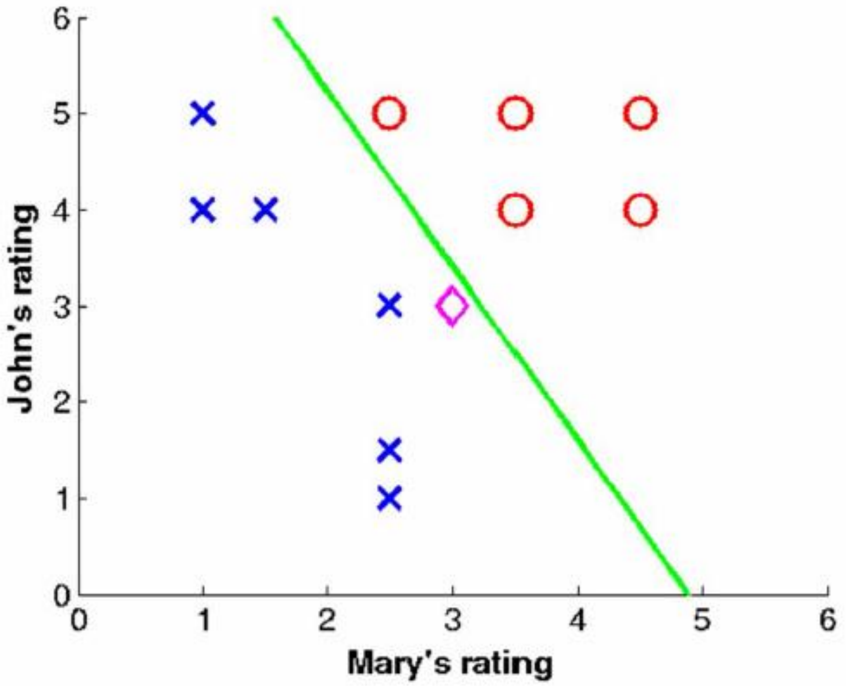


STEPS

1. Initialize the parameters θ and b at random
2. Pick a random example $\{x^{(i)}, y^{(i)}\}$
3. Compute the partial derivatives of θ_1, θ_2, b
4. Update parameters using:

$$\theta_1 = \theta_1 - \alpha\Delta\theta_1$$
$$\theta_2 = \theta_2 - \alpha\Delta\theta_2$$
$$b = b - \alpha\Delta b$$

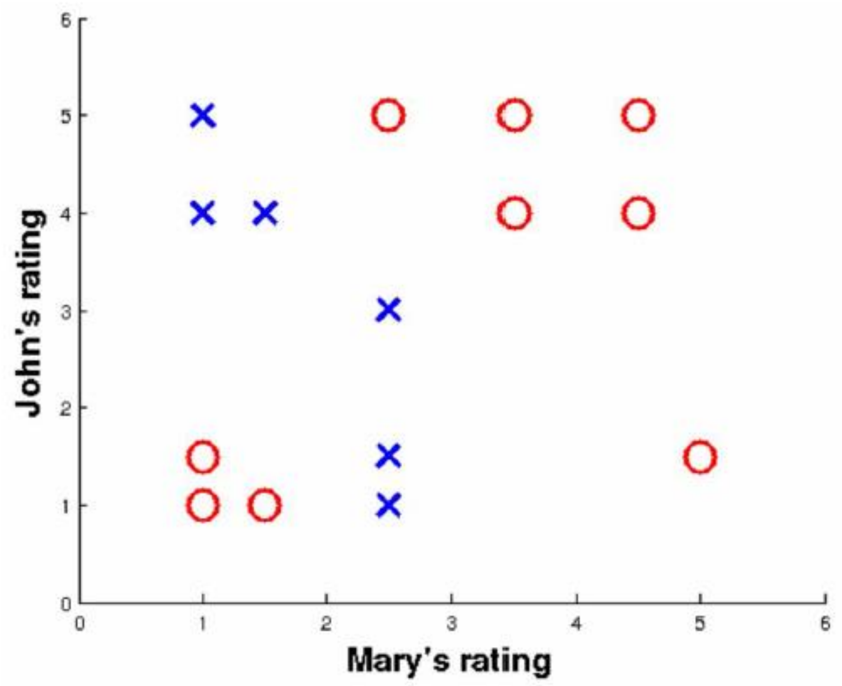
Stop it when parameters don't change much, or after a certain number of iterations.



Gravity movie is slightly on the "don't watch" side.

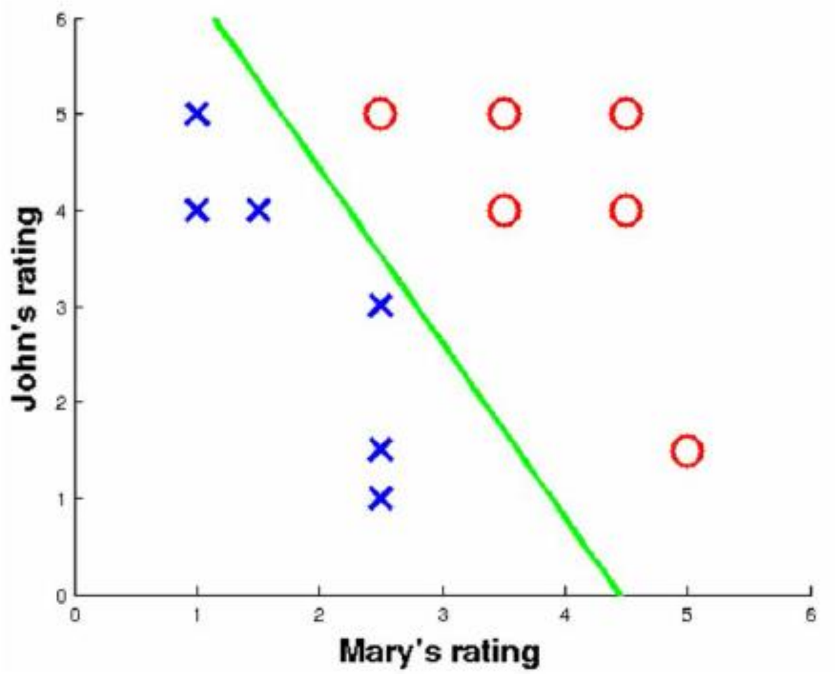
With this data set, it seems like "not watching it" makes more sense.

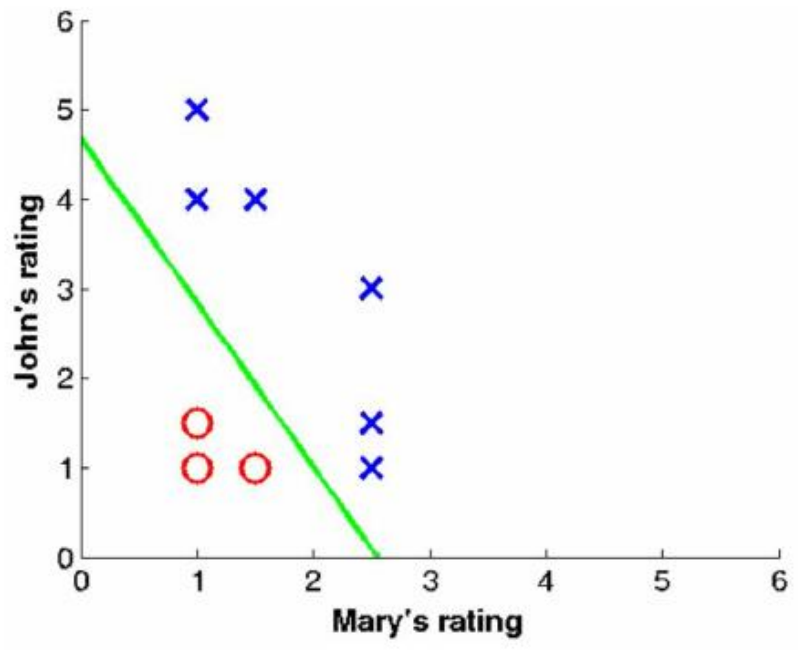
NETFLIX



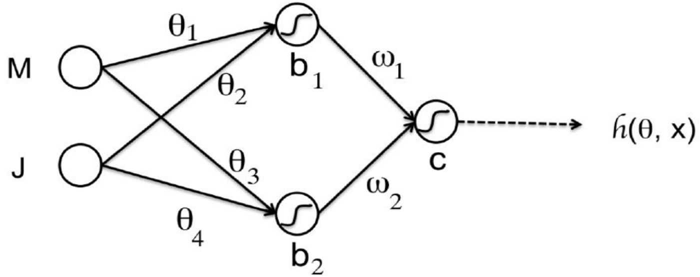
Nancy likes some of the movies both Mary and John rated poorly.

How can I have a linear decision boundary separate these?

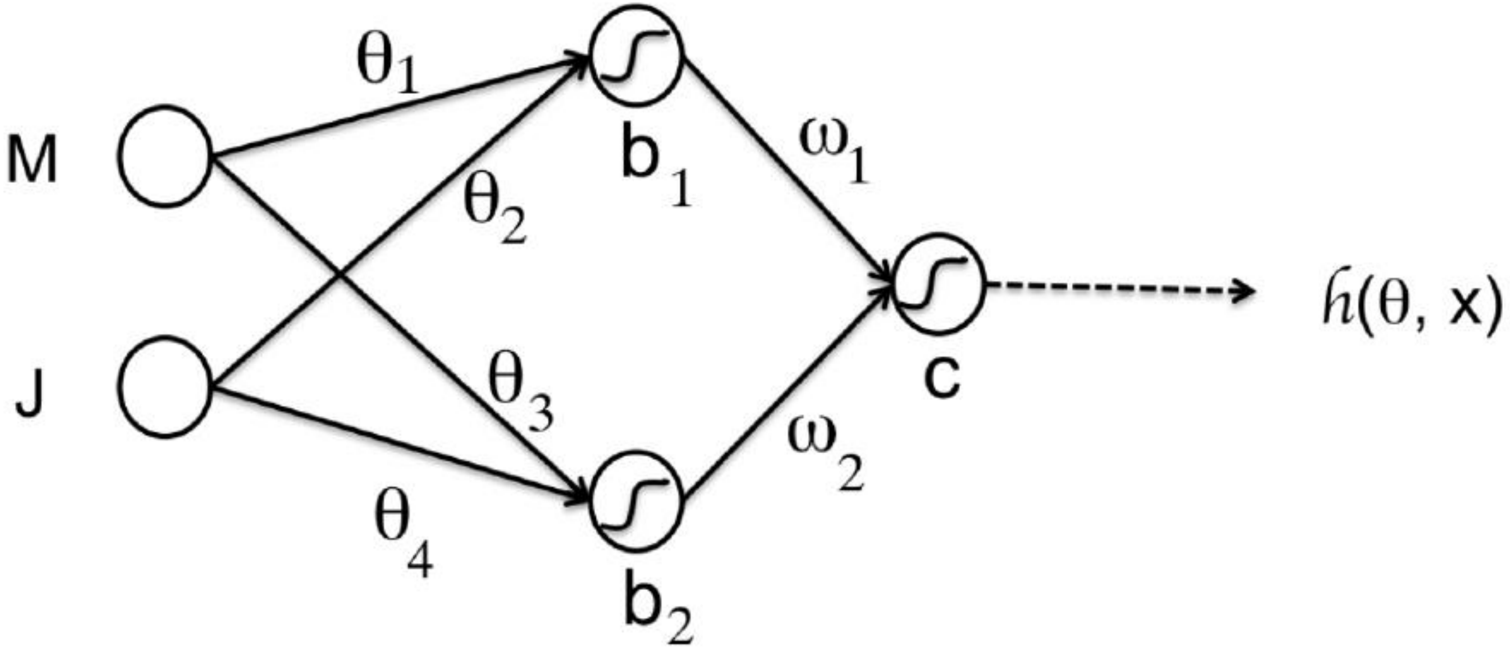

John's rating
Mary's rating
0
1
2
3
4
5
6


John's rating
Mary's rating
0
1
2
3
4
5
6

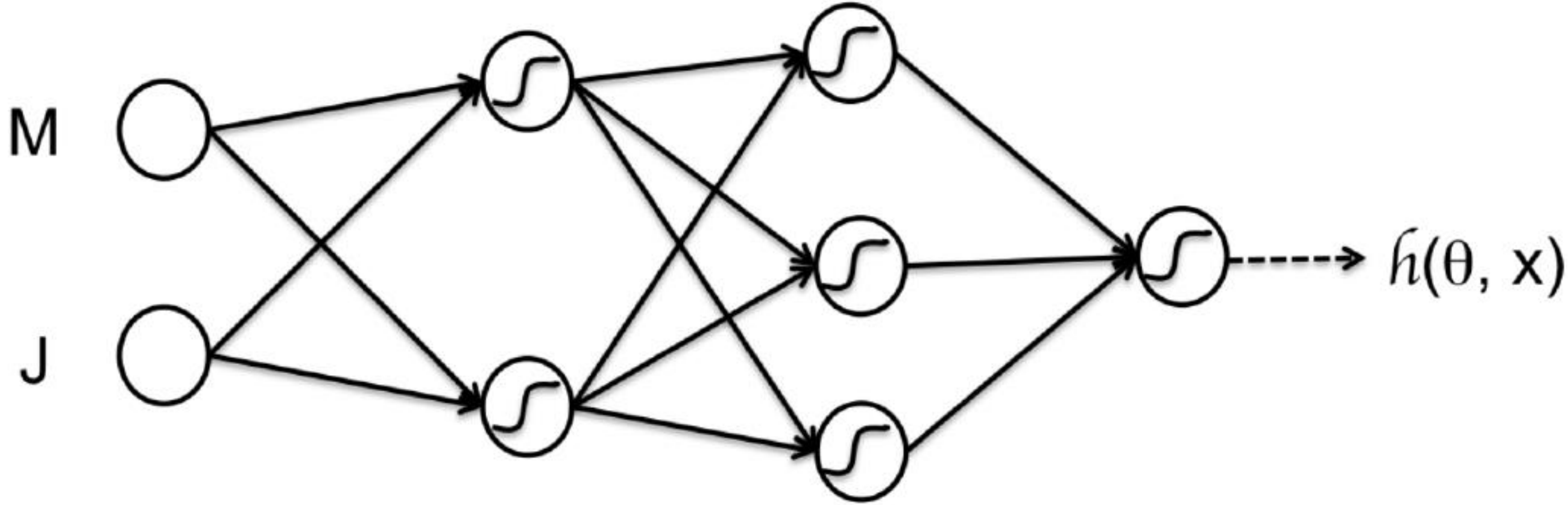
Movie	Output by decision function *h1*	Output by decision function *h2*	Does Nancy like?
Lord of the Rings 2	$h_1(x^{(1)})$	$h_2(x^{(1)})$	No
…	…	…	…
Star Wars 1	$h_1(x^{(n)})$	$h_2(x^{(n)})$	Yes
Gravity	$h_1(x^{(n+1)})$	$h_2(x^{(n+1)})$	?



THIS IS THE NEURAL NETWORK

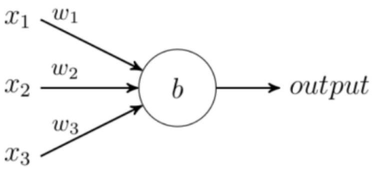


A DEEPER NEURAL NETWORK



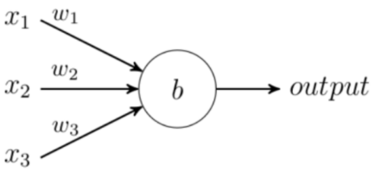
DEEP LEARNING: BASIC STRUCTURE

BASIC SINGLE NEURON



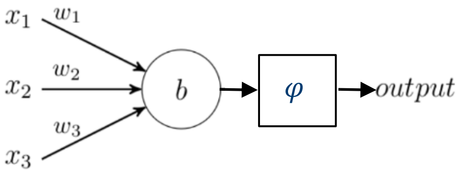
DEEP LEARNING: BASIC STRUCTURE

BASIC SINGLE NEURON



$$u_n = \sum_{j=1}^{m} w_{nj} x_j$$

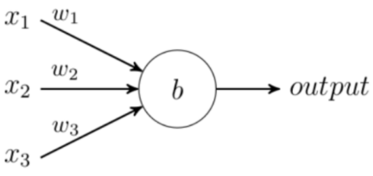
SINGLE NEURON WITH ACTIVATION



$\varphi \longrightarrow$ Activation function

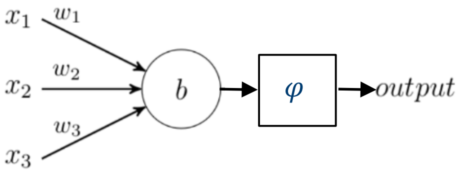
DEEP LEARNING: BASIC STRUCTURE

BASIC SINGLE NEURON



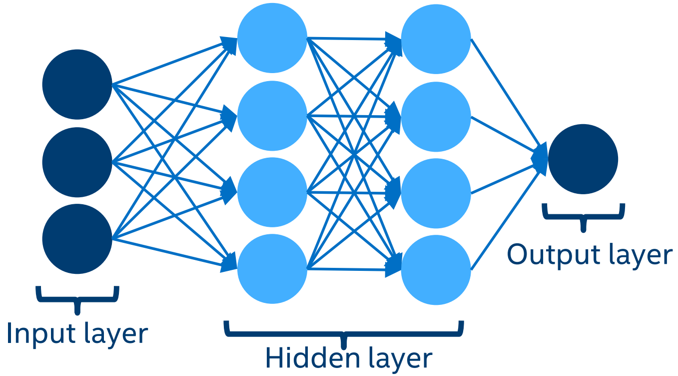
$$u_n = \sum_{j=1}^{m} w_{nj} x_j$$

SINGLE NEURON WITH ACTIVATION



$\varphi \longrightarrow$ Activation function

BASIC STRUCTURE WITH TWO HIDDEN LAYERS



KEYWORDS

- Training / Testing percentage
- Overfitting – Underfitting
- Topology
- Training Algorithms
- Learning Rate
- Batch Size

CLASSICAL MACHINE LEARNING VS DEEP LEARNING

CLASSIC ML

Using optimized functions or algorithms to extract insights from data

Training Data*

Algorithms
- Random Forest
- Support Vector Machines
- Regression
- Naïve Bayes
- Hidden Markov
- K-Means Clustering
- Ensemble Methods
- More…

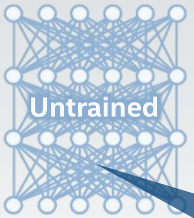
Inference, Clustering, or Classification

New Data*

*Note: not all classic machine learning functions require training

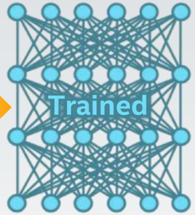
DEEP LEARNING

Using massive labeled data sets to train deep (neural) graphs that can make inferences about new data



CNN, RNN, RBM..

New Data



Step 1: Training

Hours to Days in Cloud

Use massive labeled dataset (e.g. 10M tagged images) to iteratively adjust weighting of neural network connections

Step 2: Inference

Real-Time at Edge/Cloud

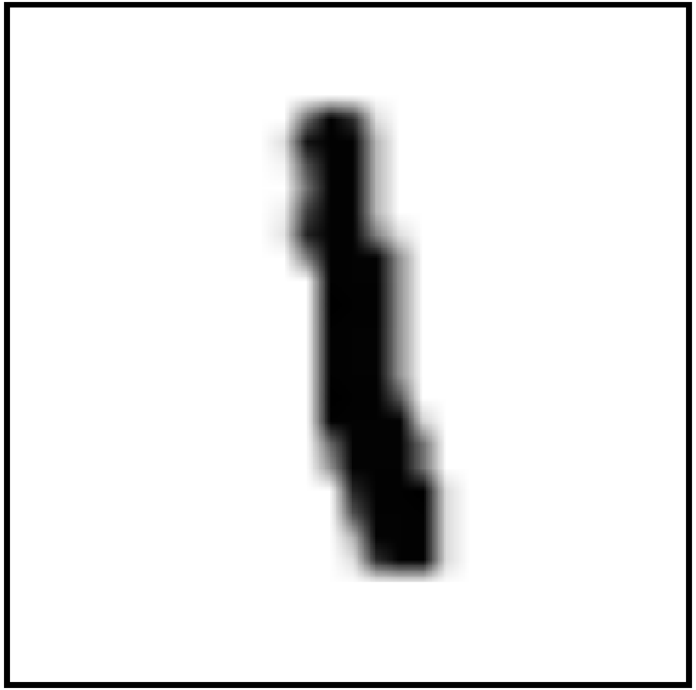
Form inference about new input data (e.g. a photo) using trained neural network

ARTIFICIAL INTELLIGENCE

NAIVE APPROACH

USE CASE: HANDWRITTEN DIGITS (MNIST)

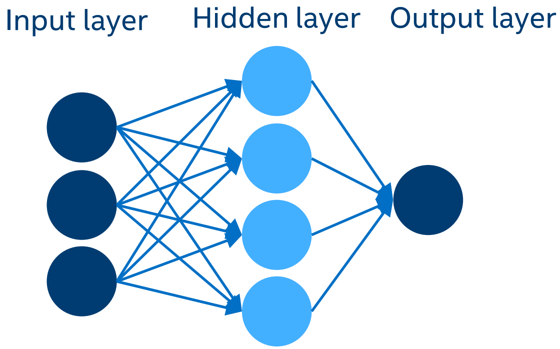
USE CASE: HANDWRITTEN DIGITS (MNIST)



$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

USE CASE: HANDWRITTEN DIGITS (MNIST)

MNIST DATASET
28x28 Pixels



= 28x28 pixels
= 784 units

= 100 units
(variable)

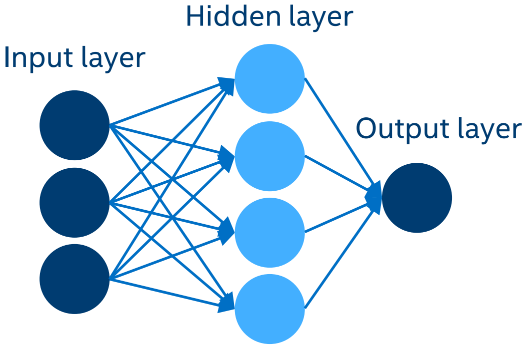
= 10 units (0 – 9)

TOTAL PARAMETERS

$W_{input \rightarrow hidden}$	784 x 100
b_{hidden}	100
$W_{hidden \rightarrow output}$	100 x 10
B_{output}	10

$$u_n = \sum_{j=1}^{m} w_{nj} x_j$$

TRAINING

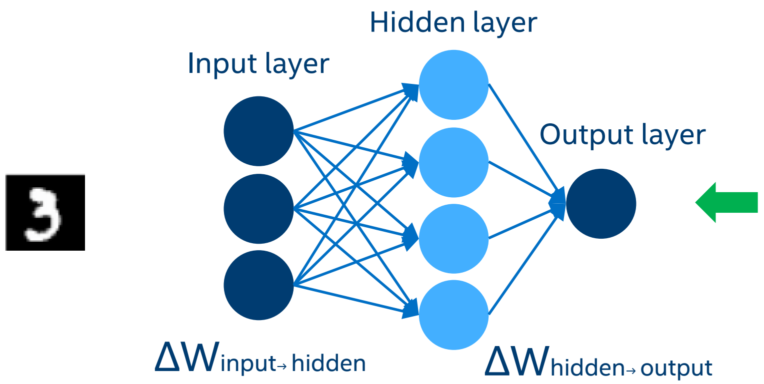


1) Initialize weights
2) Forward pass
3) Calculate cost
4) Backward pass
5) Update weights

Output	Ground Truth
0.2	0.0
0.0	0.0
0.5	1
0.0	0.0
0.1	0.0
0.4	0.0
0.2	0.0
0.0	0.0
0.1	0.0
0.0	0.0

Cost function =
C(output, truth)

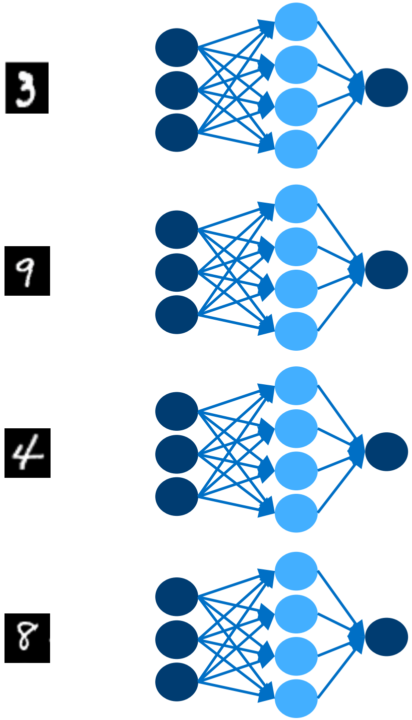
TRAINING: BACKPROPAGATION

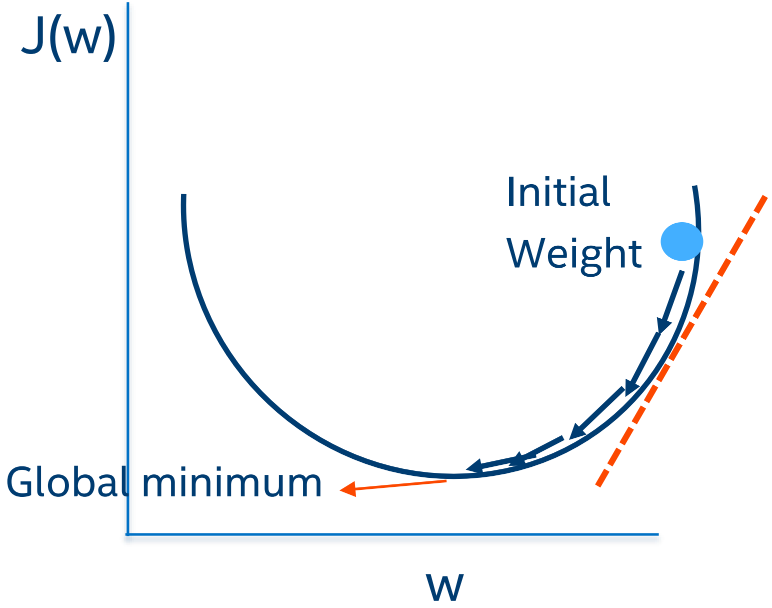


Cost function =

C(output, truth)

TRAINING: STOCHASTIC GRADIENT DESCENT





HANDS-ON WORK

Case Study: MNIST by Softmax Regression

USE CASE: HANDWRITTEN DIGITS (MNIST)

Softmax Regression

iPython notebook:

https://github.com/mstfldmr/IntelAIWorkshop/blob/master/SoftmaxRegression.ipynb

ARTIFICIAL INTELLIGENCE

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks (CNN)

Essentially neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

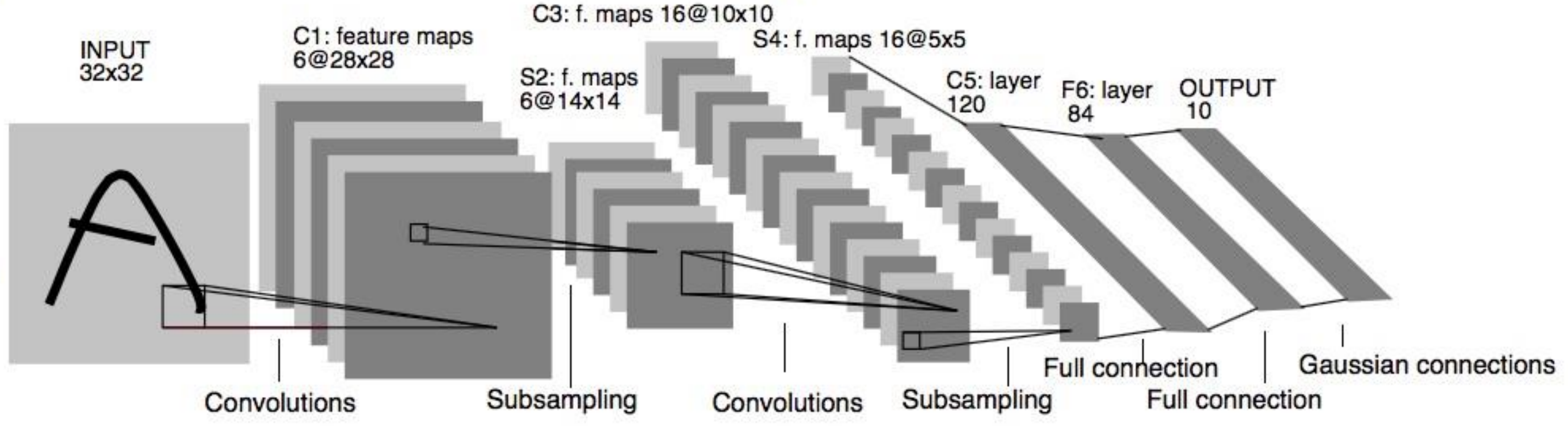
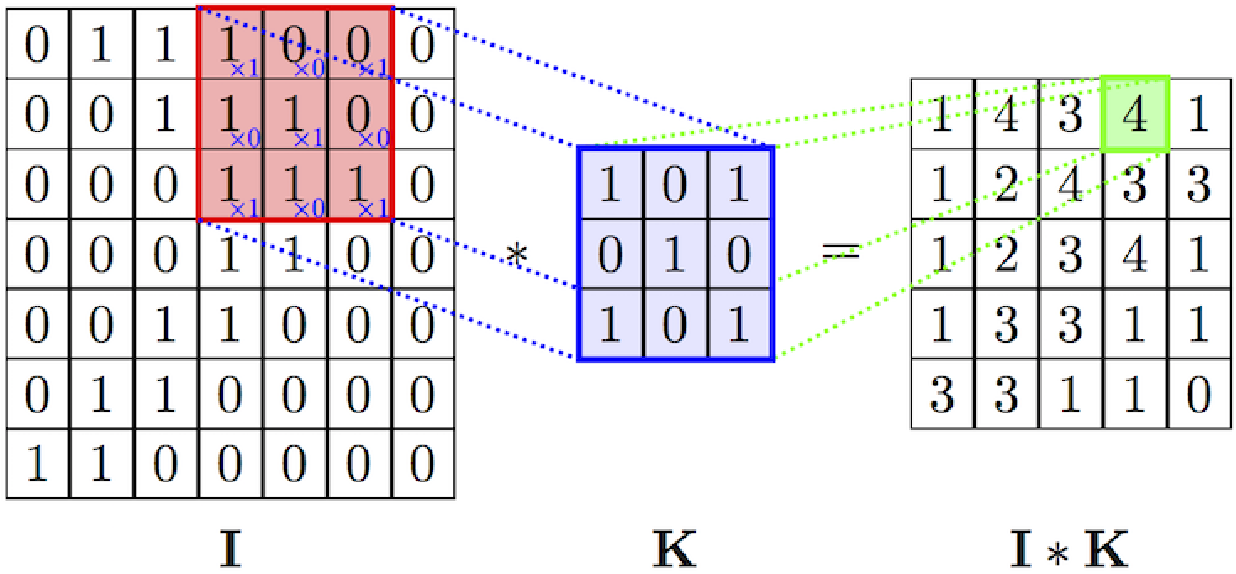


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CONVOLUTION



CONVOLUTION

$*$

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

IMAGE KERNELS

identity

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

left sobel

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

bottom sobel

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

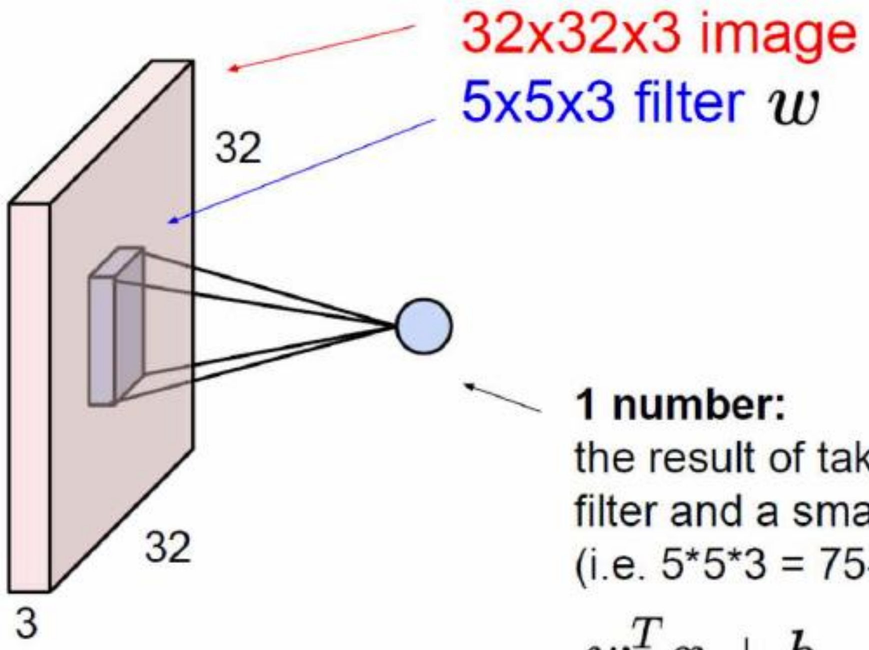
blur

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$

http://setosa.io/ev/image-kernels/

CONVOLUTION

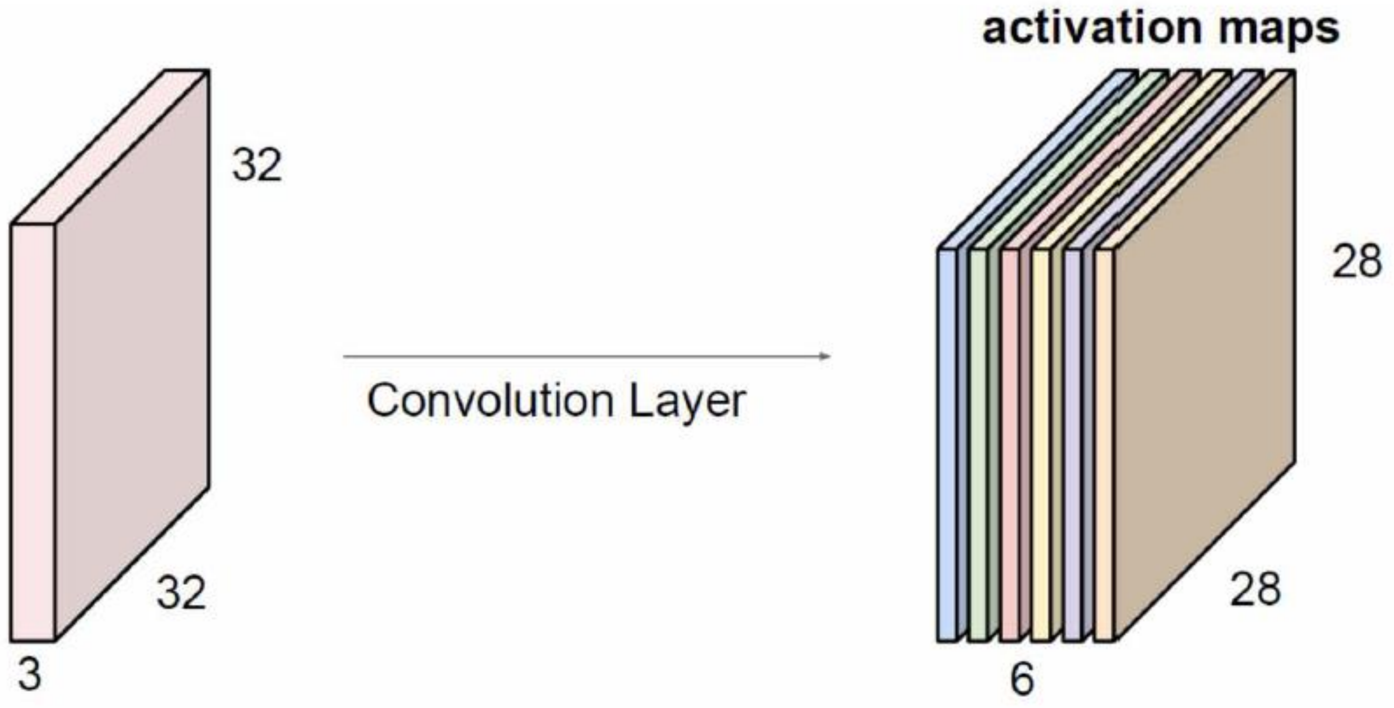
Convolution Layer



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

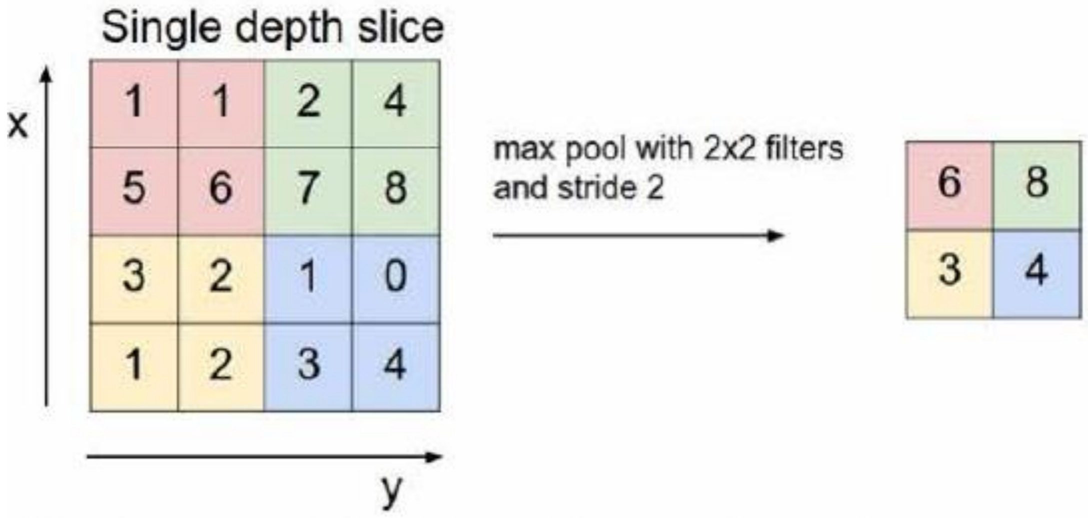
$$w^T x + b$$

Convolution Layer

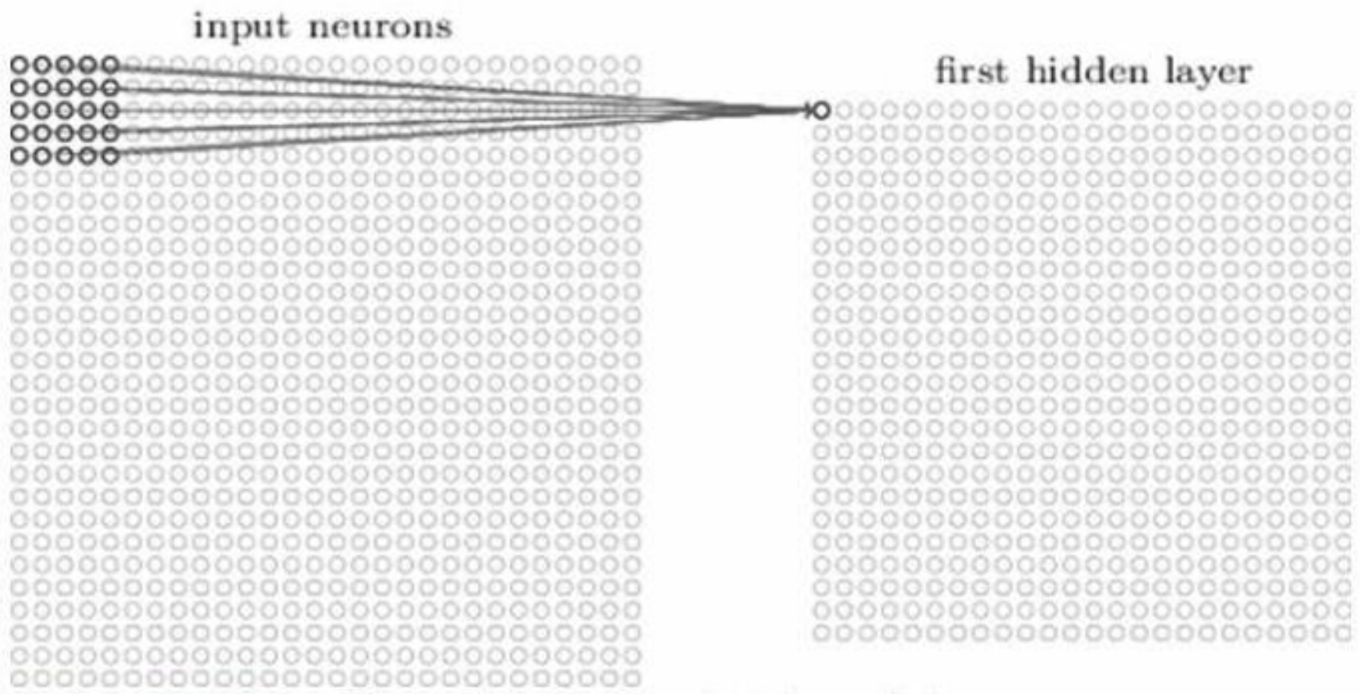


We stack these up to get a “new image” of size 28x28x6!

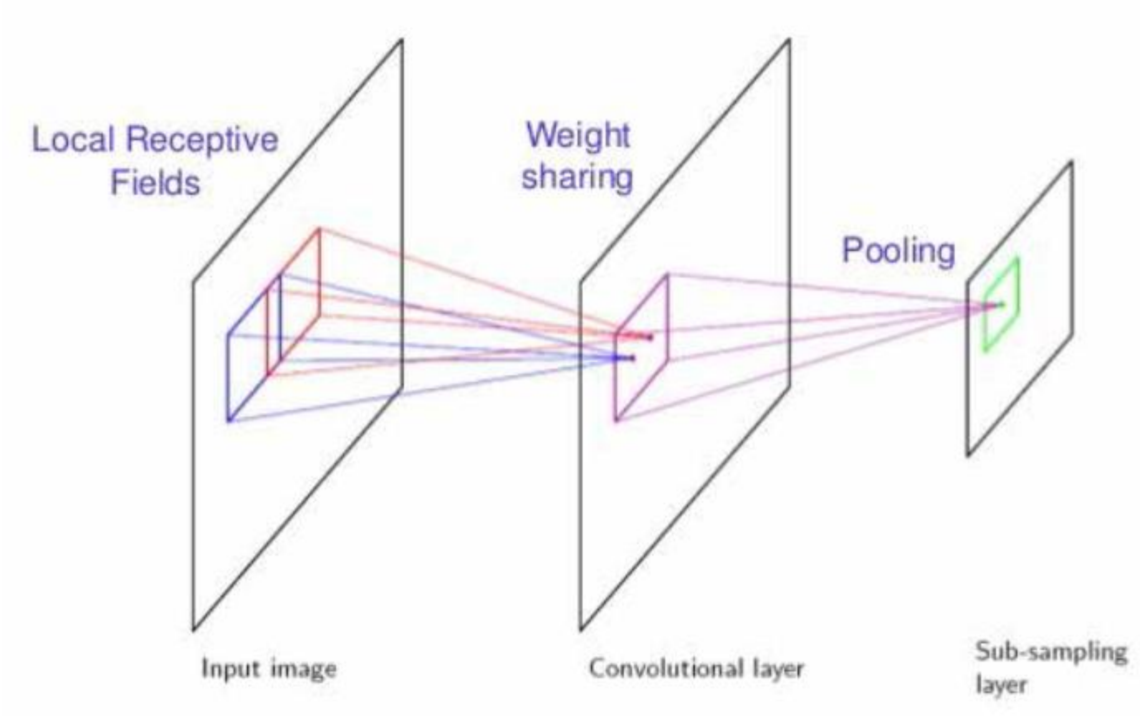
Max Pooling


Single depth slice
x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
max pool with 2x2 filters
and stride 2
6 8
3 4

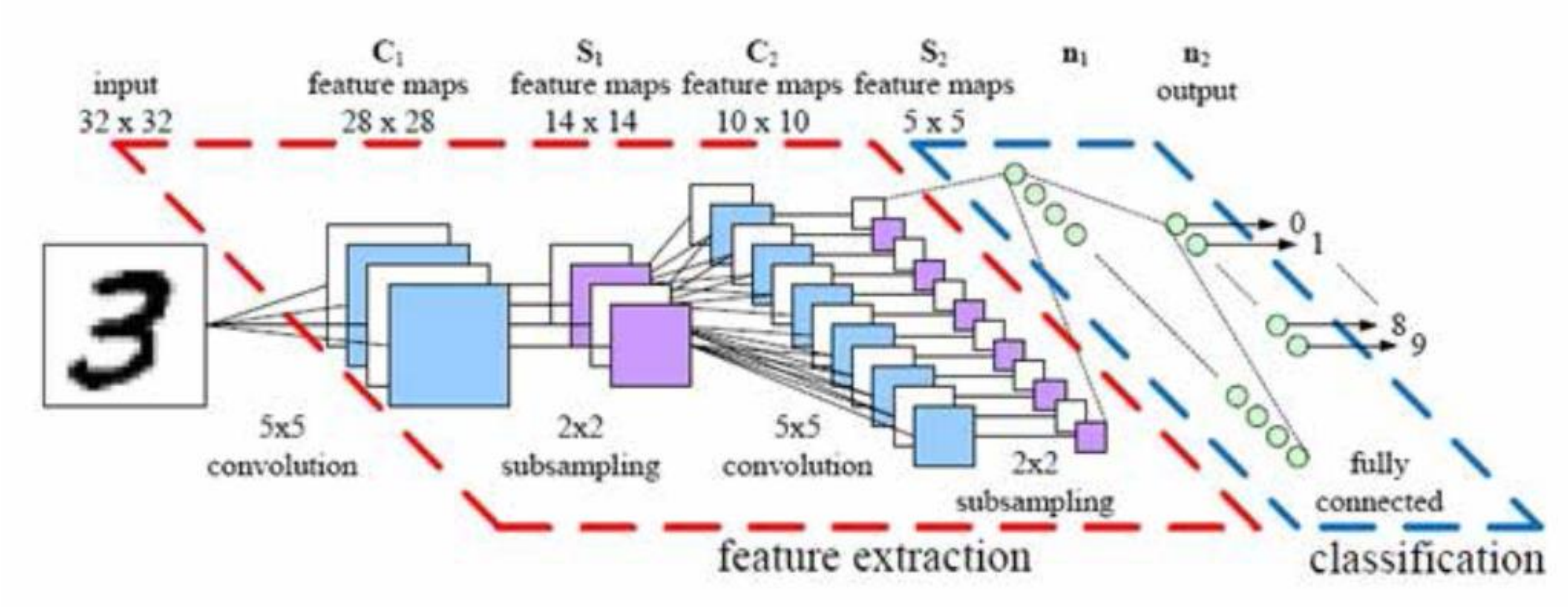
Simplification 1: Local Receptive Fields



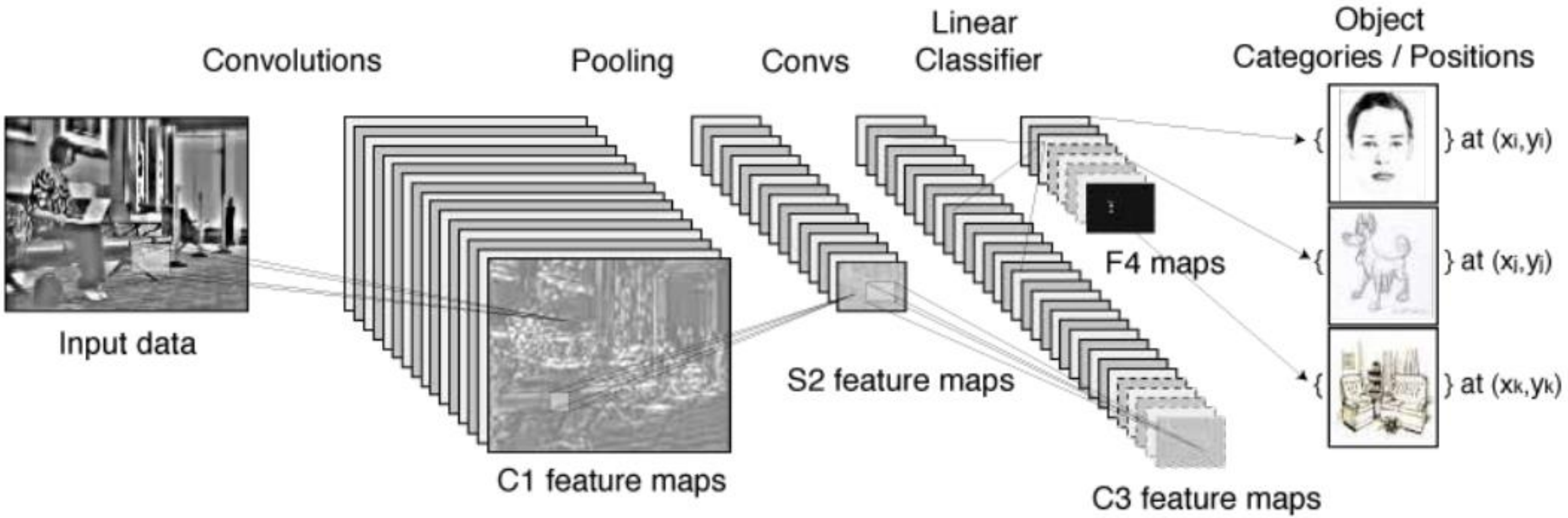
Simplification 2: Shared Weights



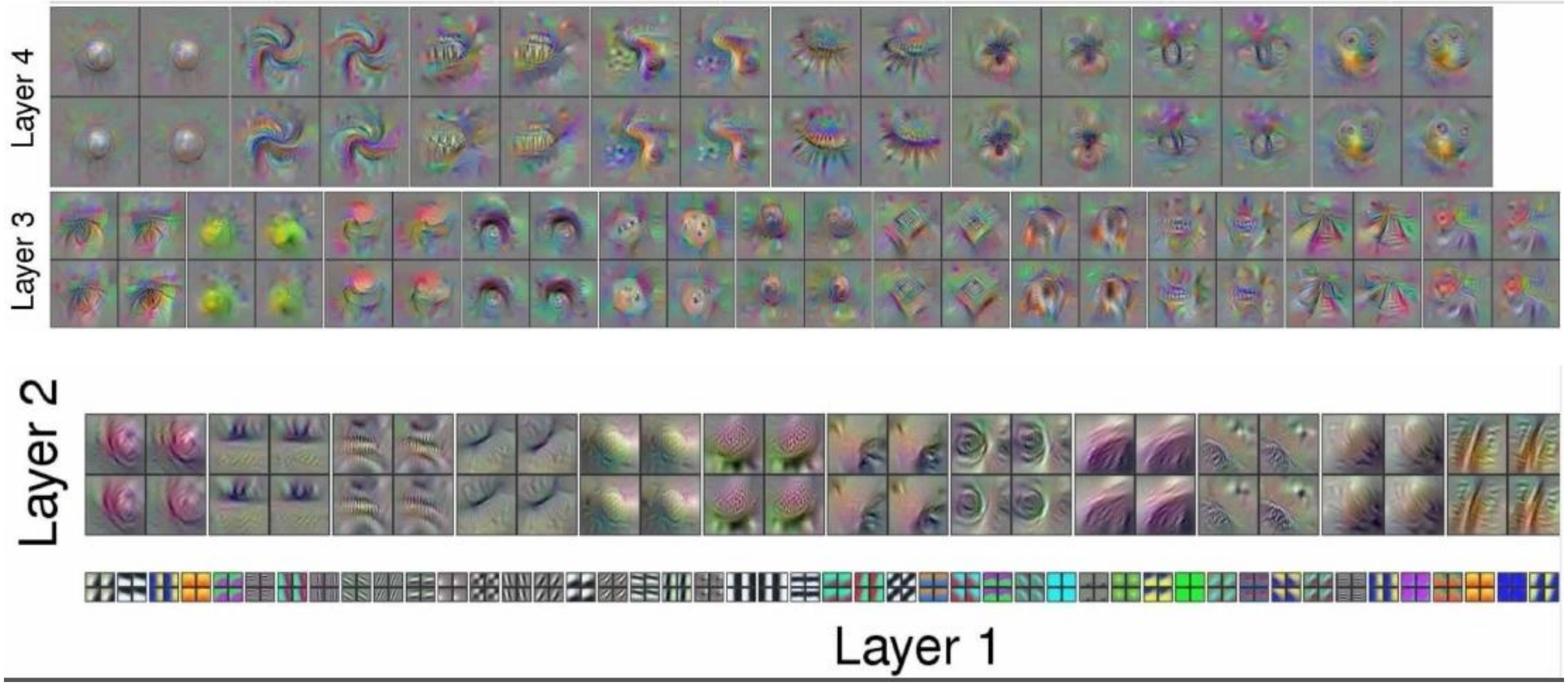
CNN Pipeline



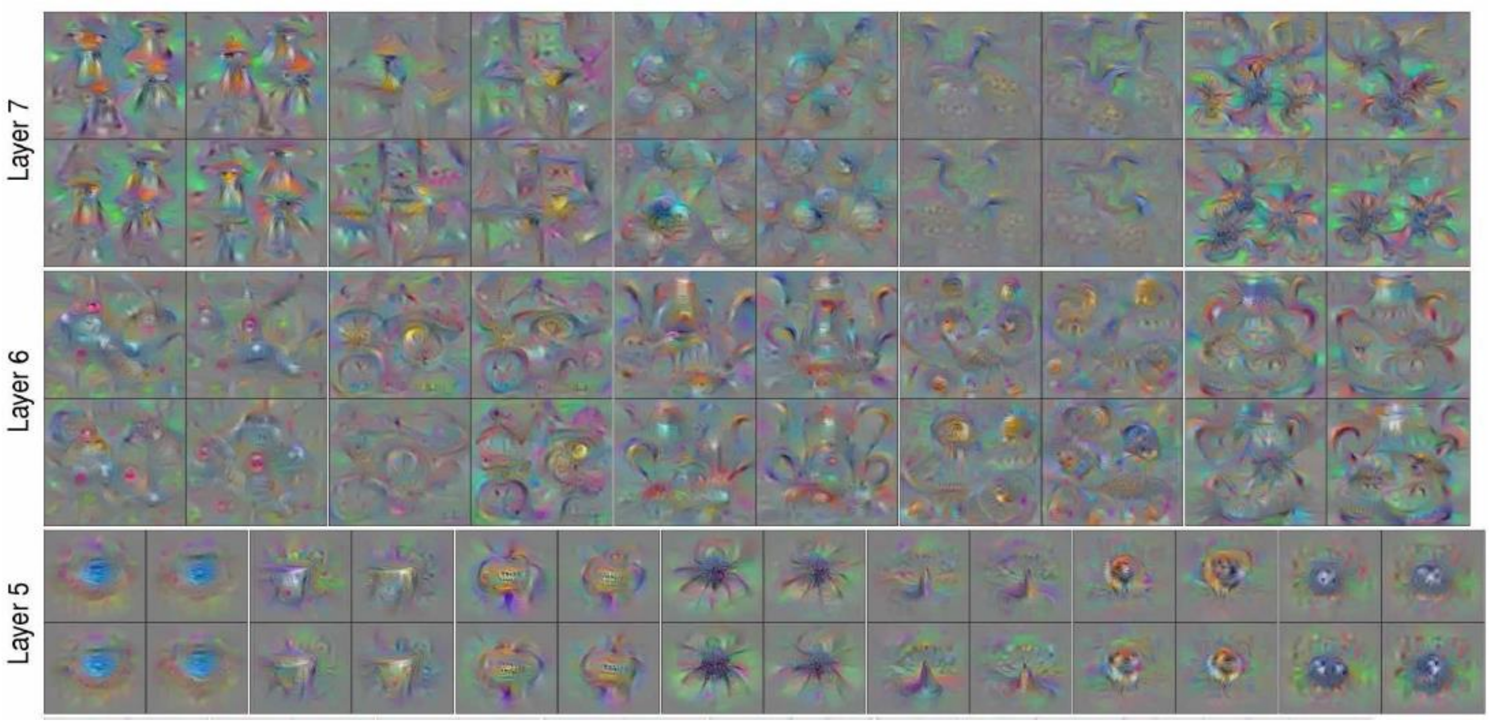
Example CNN Pipeline for Object Detection



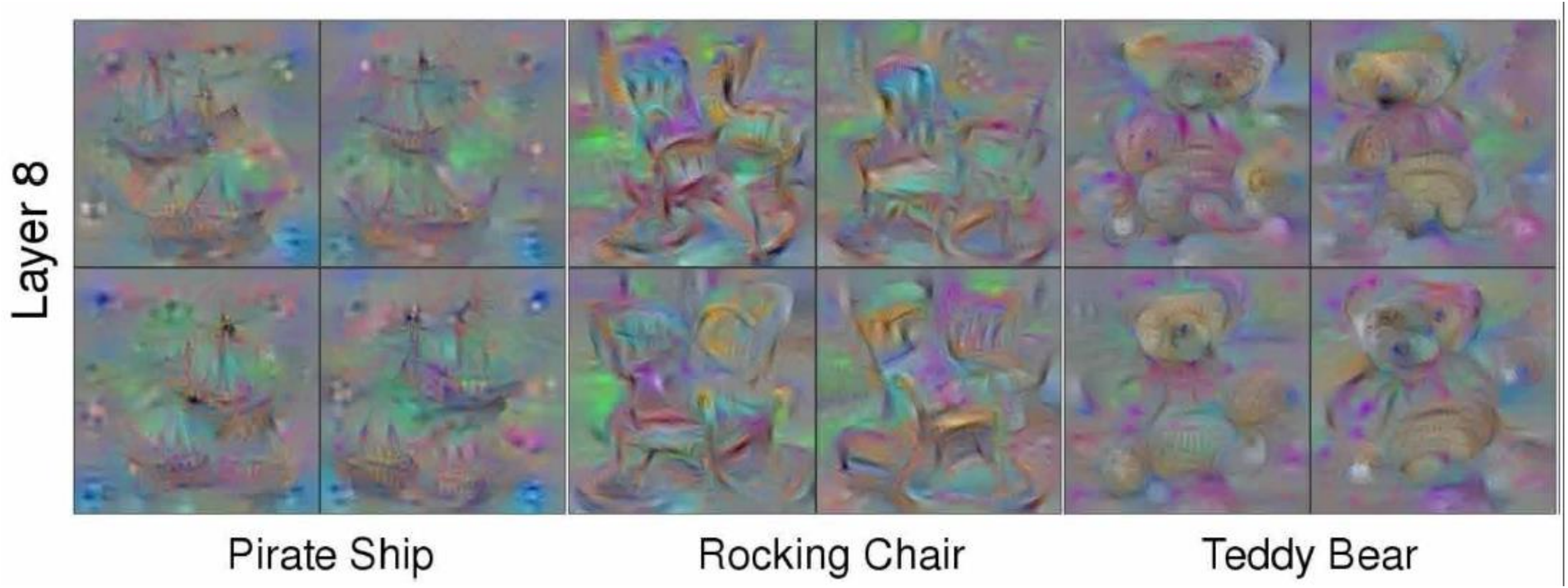
Visualizing Neurons



Visualizing Neurons



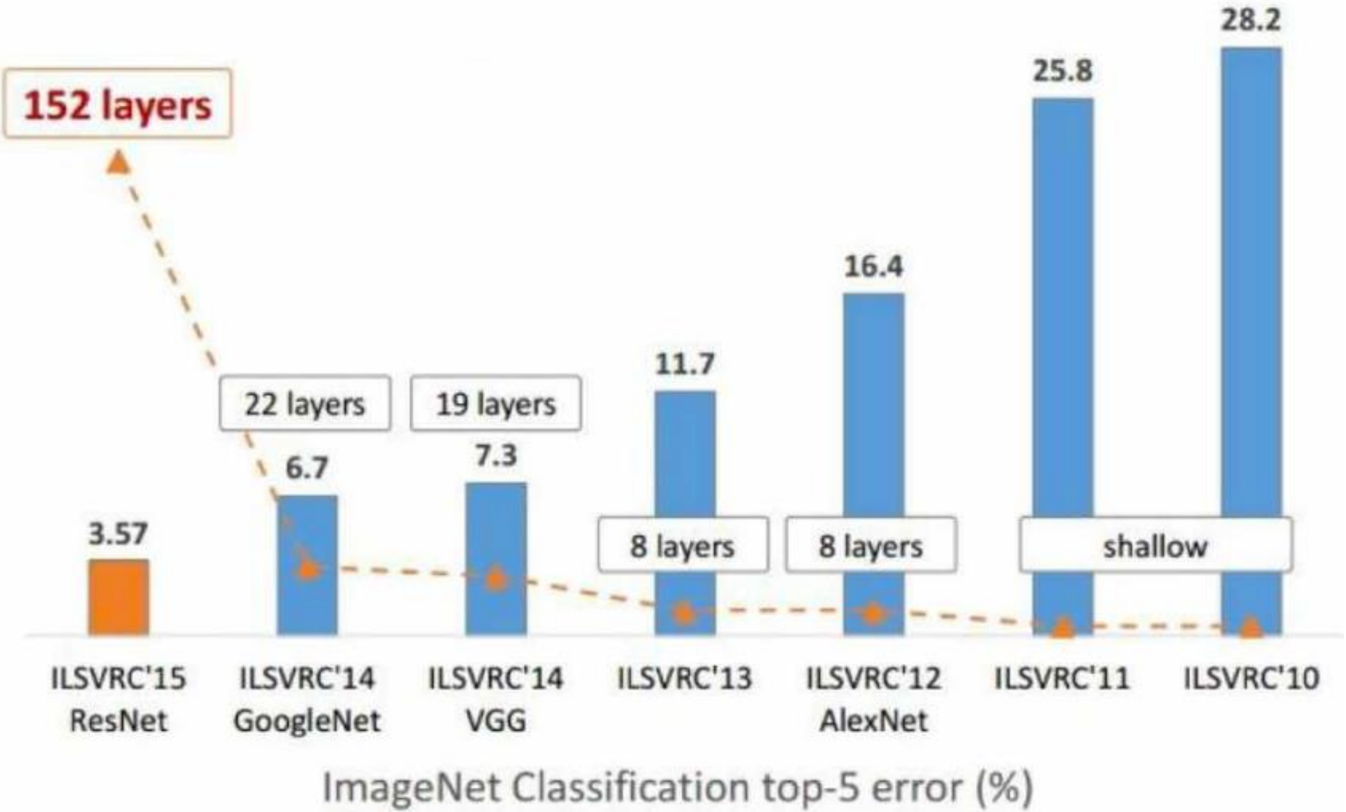
Visualizing Neurons



ImageNet Large Scale Visual Recognition Competition (ILSVRC)

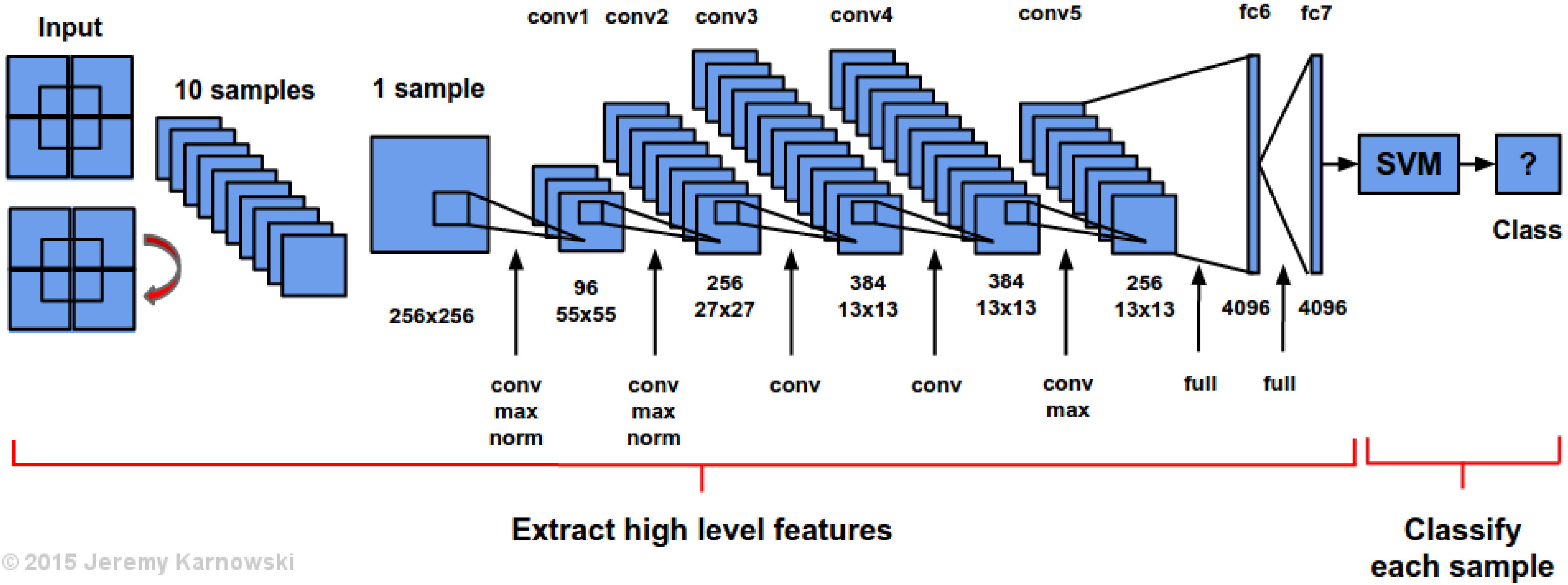
- ~1M images
- 1K object categories in the training set
- Task: What is the object in the image?
- Classify the image into one of 1000 categories
- Evaluation
  - Is one of the best 5 guesses is correct?
  - Human performance is around 5.1% error.

Evolution of Depth



AlexNet (2012)

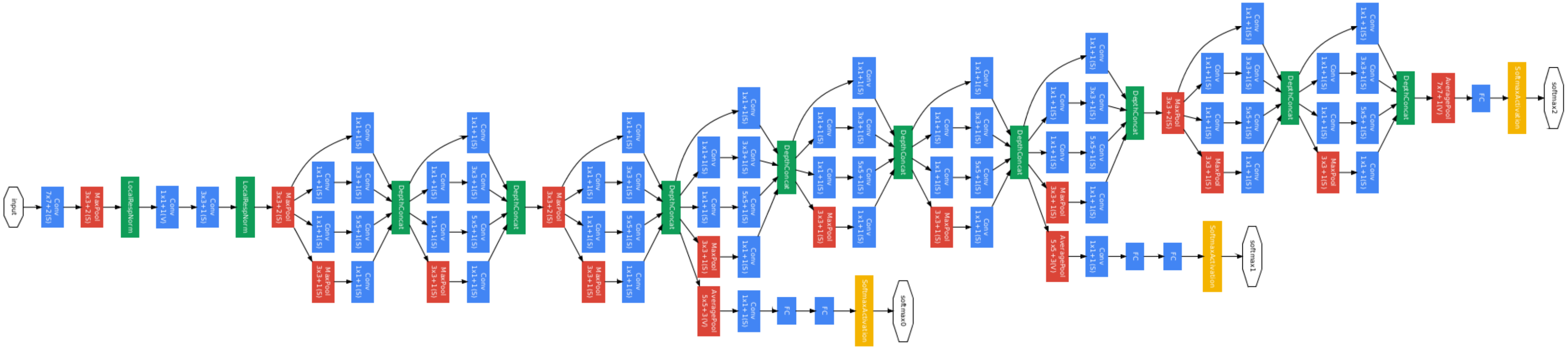
8 layers
16,4 error rate



GoogleNet (2014)

22 layers

6,7 error rate



ResNet (2015)

152 layers

3,57 error rate

HANDS-ON WORK

Case Study: MNIST with Convolutional Neural Networks

USE CASE: HANDWRITTEN DIGITS (MNIST)

Convolutional Neural Networks

iPython notebook:

https://github.com/mstfldmr/IntelAIWorkshop/blob/master/ConvolutionalNeuralNetwork.ipynb

HANDS-ON WORK

Case Study: CIFAR10 with Convolutional Neural Networks

USE CASE: HANDWRITTEN DIGITS (CIFAR-10)

Convolutional Neural Networks

iPython notebook:

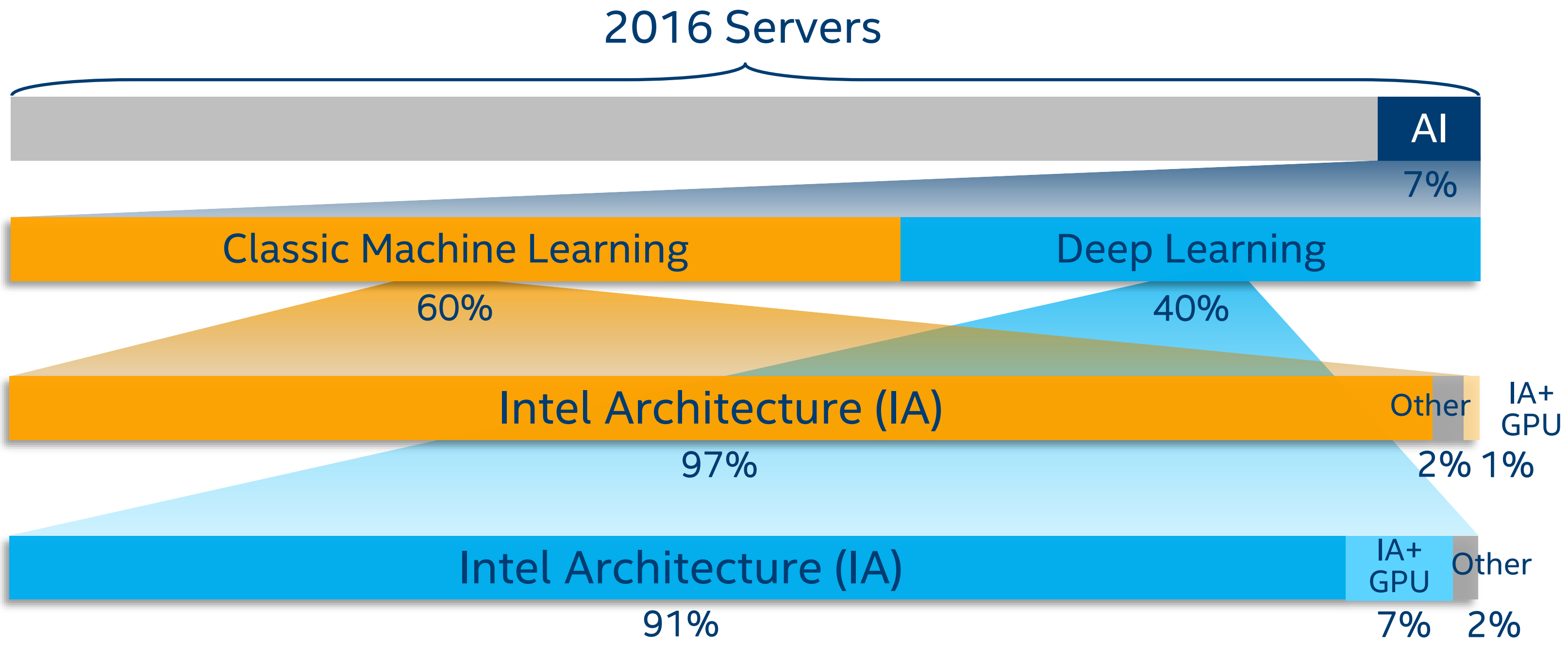
https://github.com/mstfldmr/IntelAIWorkshop/blob/master/CNN_CIFAR10_Keras.ipynb

https://github.com/mstfldmr/IntelAIWorkshop/blob/master/CNN_CIFAR10_Keras2.ipynb

ARTIFICIAL INTELLIGENCE

INTEL® AI PORTFOLIO

AI IS THE FASTEST GROWING DATA CENTER WORKLOAD



INTEL® NERVANA™ AI ACADEMY

Hone Your Skills and Build the Future of AI

Benefit from expert-led trainings, hands-on workshops, exclusive remote access, and more.

Gain access to the latest libraries, frameworks, tools and technologies from Intel to accelerate your AI project.

Collaborate with industry luminaries, developers, students, and Intel engineers.

software.intel.com/ai/academy

INTEL AI PORTFOLIO

EXPERIENCES

TOOLS

Intel® Deep Learning SDK

Intel® Computer Vision SDK

Movidius Neural Compute Stick

saffron TECHNOLOGY an intel company

FRAMEWORKS

Apache Spark MLlib BigDL

TensorFlow

mxnet

theano

neon

torch

Caffe

E2E Tool

LIBRARIES

python Intel Dist

Intel® DAAL

Intel® Nervana™ Graph*

Intel® MKL

MKL-DNN

Intel® MLSL

Movidius MvTensor Library

Associative Memory Base

HARDWARE

intel ATOM inside | intel QUARK inside | intel CORE i7 inside | intel XEON inside | intel XEON PHI inside | intel ARRIA 10 inside | Lake Crest*

Compute

Memory & Storage

Networking

intel REALSENSE TECHNOLOGY

Movidius an intel company

Visual Intelligence

UNLEASH FULL POTENTIAL

*Coming 2017

LIBRARIES, FRAMEWORKS & TOOLS

	Intel® Math Kernel Library: Intel® MKL	Intel® Math Kernel Library: MKL-DNN	Intel® MLSL	Intel® Data Analytics Acceleration Library (DAAL)	python Intel® Distribution	neon TensorFlow mxnet … Open Source Frameworks	Intel Deep Learning SDK	Intel® Computer Vision SDK
High Level Overview	Computation primitives; high performance math primitives granting low level of control	Computation primitives; free open source DNN functions for high-velocity integration with deep learning frameworks	Communication primitives; building blocks to scale deep learning framework performance over a cluster	Broad data analytics acceleration object oriented library supporting distributed ML at the algorithm level	Most popular and fastest growing language for machine learning	Toolkits driven by academia and industry for training machine learning algorithms	Accelerate deep learning model design, training and deployment	Toolkit to develop & deploying vision-oriented solutions that harness the full performance of Intel CPUs and SOC accelerators
Primary Audience	Consumed by developers of higher level libraries and Applications	Consumed by developers of the next generation of deep learning frameworks	Deep learning framework developers and optimizers	Wider Data Analytics and ML audience, Algorithm level development for all stages of data analytics	Application Developers and Data Scientists	Machine Learning App Developers, Researchers and Data Scientists.	Application Developers and Data Scientists	Developers who create vision-oriented solutions
Example Usage	Framework developers call matrix multiplication, convolution functions	New framework with functions developers call for max CPU performance	Framework developer calls functions to distribute Caffe training compute across an Intel® Xeon Phi™ cluster	Call distributed alternating least squares algorithm for a recommendation system	Call scikit-learn k-means function for credit card fraud detection	Script and train a convolution neural network for image recognition	Deep Learning training and model creation, with optimization for deployment on constrained end device	Use deep learning to do pedestrian detection

Find out more at http://software.intel.com/ai

INTEL DISTRIBUTION FOR PYTHON

Advancing Python Performance Closer to Native Speeds

python™

For developers using the most popular and fastest growing programming language for AI

Easy, Out-of-the-box Access to High Performance Python

- Prebuilt, optimized for numerical computing, data analytics, HPC
- Drop in replacement for your existing Python (no code changes required)

Drive Performance with Multiple Optimization Techniques

- Accelerated NumPy/SciPy/Scikit-Learn with Intel® MKL
- Data analytics with pyDAAL, enhanced thread scheduling with TBB, Jupyter* Notebook interface, Numba, Cython
- Scale easily with optimized MPI4Py and Jupyter notebooks

Faster Access to Latest Optimizations for Intel Architecture

- Distribution and individual optimized packages available through conda and Anaconda Cloud
- Optimizations upstreamed back to main Python trunk

software.intel.com/intel-distribution-for-python

INTEL® MKL-DNN

Math Kernel Library for Deep Neural Networks

For developers of deep learning frameworks featuring optimized performance on Intel hardware

Distribution Details

- Open Source
- Apache 2.0 License
- Common DNN APIs across all Intel hardware.
- Rapid release cycles, iterated with the DL community, to best support industry framework integration.
- Highly vectorized & threaded for maximal performance, based on the popular Intel® MKL library.

BETA Now Available!

github.com/01org/mkl-dnn

Direct 2D Convolution | Local response normalization (LRN) | Rectified linear unit neuron activation (ReLU) | Maximum pooling | Inner product

INTEL® MACHINE LEARNING SCALING LIBRARY (MLSL)

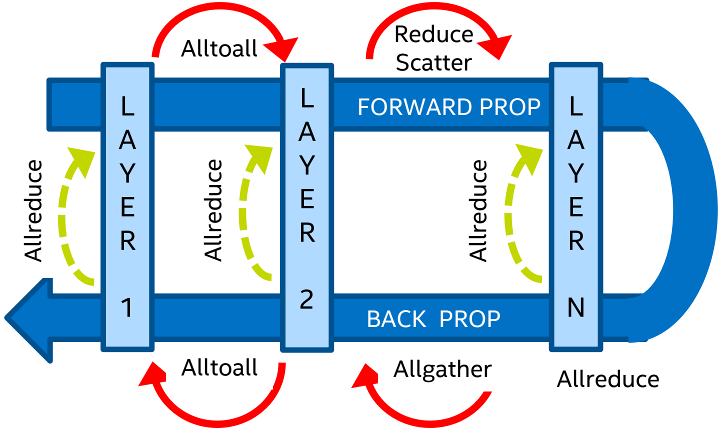
Scaling Deep Learning to 32 Nodes and Beyond

BETA Now Available!

For maximum deep learning scale-out performance on Intel® architecture

Deep learning abstraction of message-passing implementation

- Built on top of MPI; allows other communication libraries to be used as well
- Optimized to drive scalability of communication patterns
- Works across various interconnects: Intel® Omni-Path Architecture, InfiniBand, and Ethernet
- Common API to support Deep Learning frameworks (Caffe, Theano, Torch etc.)



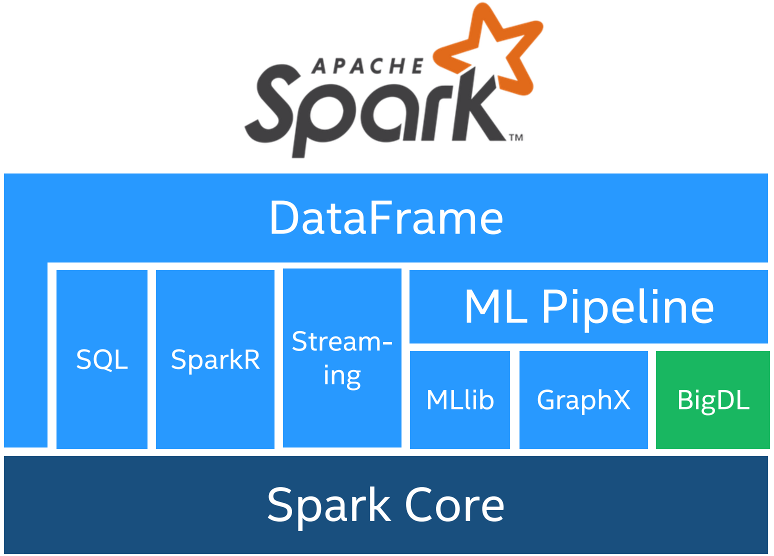
github.com/01org/MLSL/releases

BIGDL

Bringing Deep Learning to Big Data

For developers looking to run deep learning on Hadoop/Spark due to familiarity or analytics use

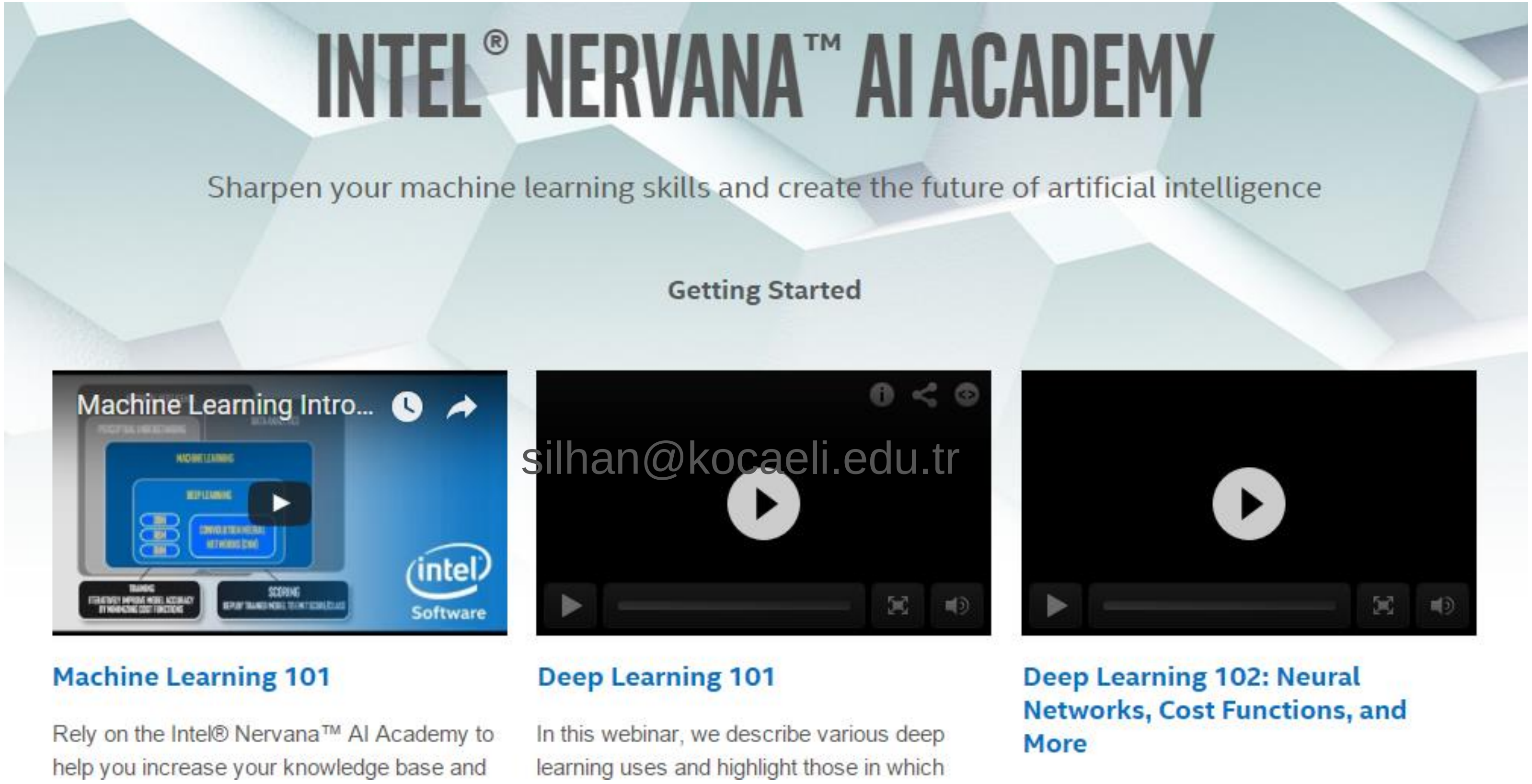
- **Open Sourced** Deep Learning Library for Apache Spark*
- **Make Deep learning more Accessible** to Big data users and data scientists.
- **Feature Parity** with popular DL frameworks like Caffe, Torch, Tensorflow etc.
- **Easy Customer and Developer Experience**
  - Run Deep learning Applications as Standard Spark programs;
  - Run on top of existing Spark/Hadoop clusters (No Cluster change)
- **High Performance** powered by Intel MKL and Multi-threaded programming.
- **Efficient Scale out** leveraging Spark architecture.



github.com/intel-analytics/BigDL

ARTIFICIAL INTELLIGENCE

NEXT STEPS...



https://software.intel.com/ai/academy

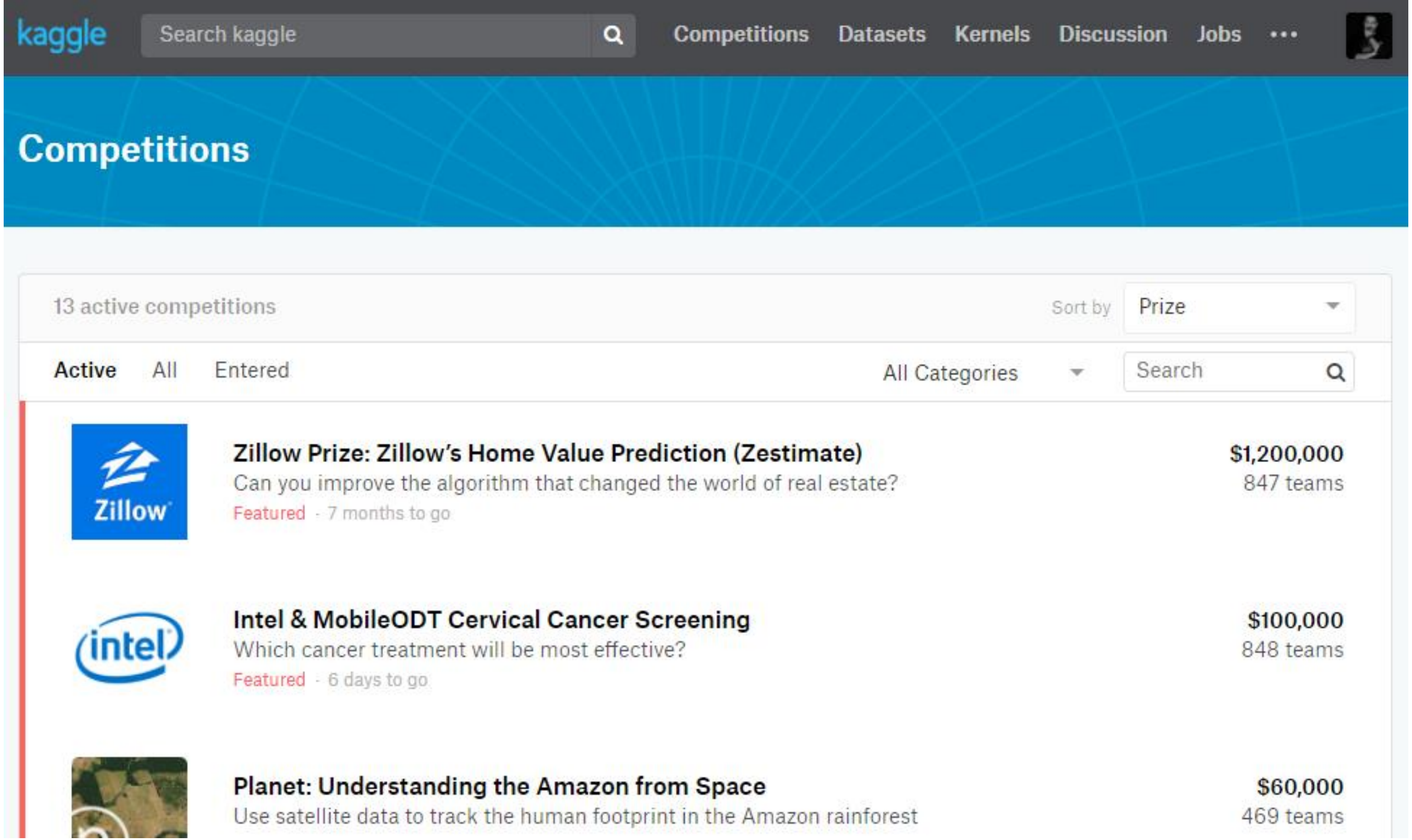
ONLINE COURSES

- Stanford Convolutional Neural Networks for Visual Recognition: http://cs231n.github.io
- MIT Machine Learning and Neural Networks: http://www.ai.mit.edu/courses/6.891-f99
- MIT Self Driving Cars: https://selfdrivingcars.mit.edu

- Deep Learning AI: https://www.deeplearning.ai
- Coursera: https://www.coursera.org
- Udacity: https://www.udacity.com
- Udemy: https://www.udemy.com
- …

TURKISH CONTENT

- Deep Learning Türkiye: https://github.com/deeplearningturkiye/turkce-yapay-zeka-kaynaklari



https://www.kaggle.com

ARTIFICIAL INTELLIGENCE

Q&A

Software

STUDENT DEVELOPER PROGRAM