

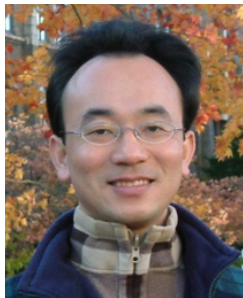


深度學習視覺辨識 (Deep Learning for Visual Recognition)

Time: 2018/01/25 (Thu) (9:00 -12:00, 13:00-16:00)

Place: 國立臺北護理健康大學 (台北市明德路365號) 親仁樓 B112

Host: 祝國忠 院長 (健康科技學院院長)



Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2018-01-25





戴敏育 博士 (Min-Yuh Day, Ph.D.)

淡江大學資管系專任助理教授

中央研究院資訊科學研究所訪問學人

國立台灣大學資訊管理博士

Publications Co-Chairs, IEEE/ACM International Conference on
Advances in Social Networks Analysis and Mining (ASONAM 2013-)

Program Co-Chair, IEEE International Workshop on
Empirical Methods for Recognizing Inference in Text (IEEE EM-RITE 2012-)

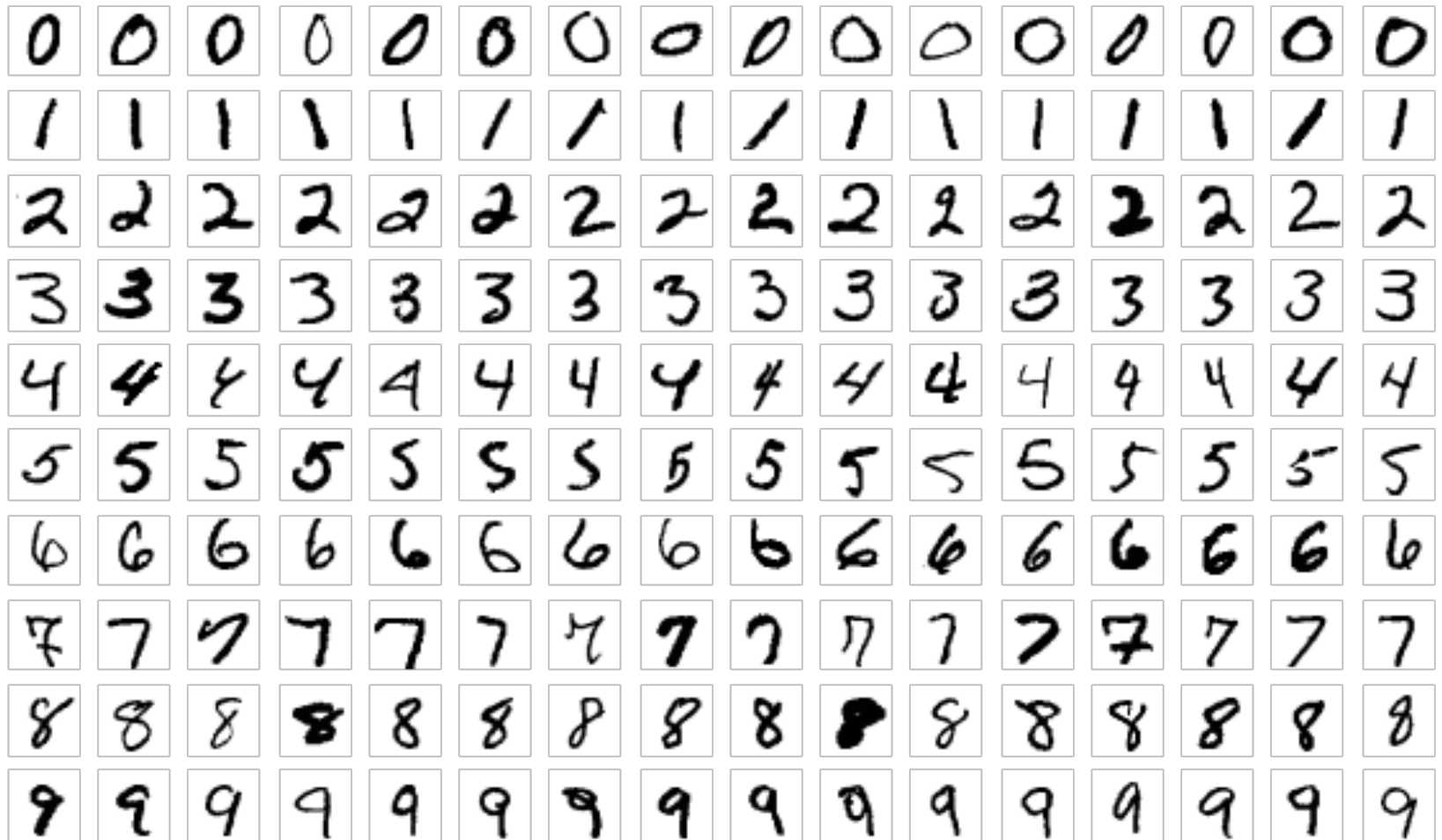
Workshop Chair, The IEEE International Conference on
Information Reuse and Integration (IEEE IRI)



Outline

- **AI, Machine Learning and Deep Learning**
- **Visual Recognition**
 - **Image Classification**
- **Deep Learning for Visual Recognition with TensorFlow and Keras**

MNIST

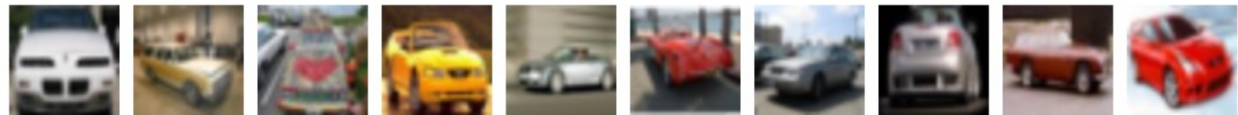


CIFAR-10 dataset

airplane



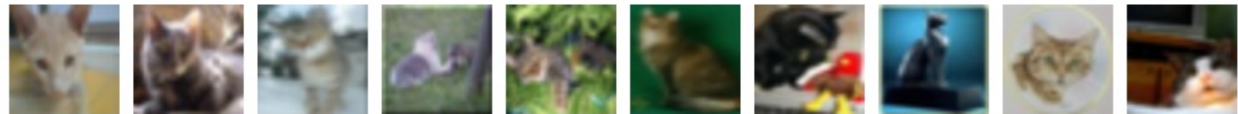
automobile



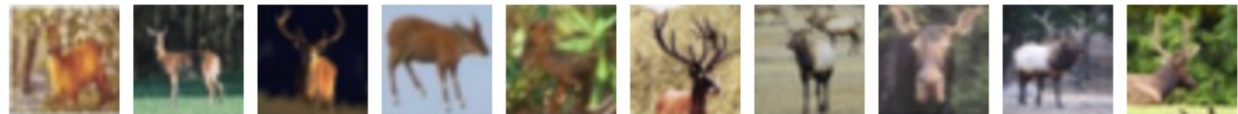
bird



cat



deer



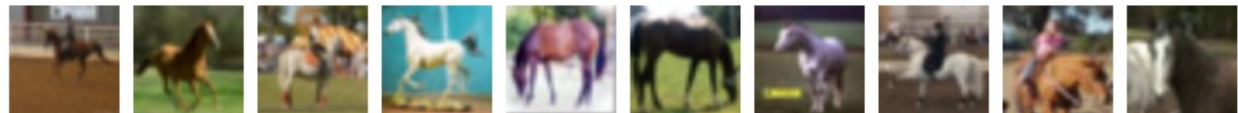
dog



frog



horse



ship



truck



Source: <https://www.cs.toronto.edu/~kriz/cifar.html>

Deep Learning Foundations: Neural Networks

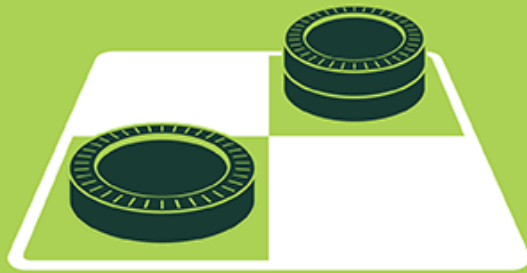
Keras: High-level API for **TensorFlow**

Artificial Intelligence

Machine Learning & Deep Learning

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.



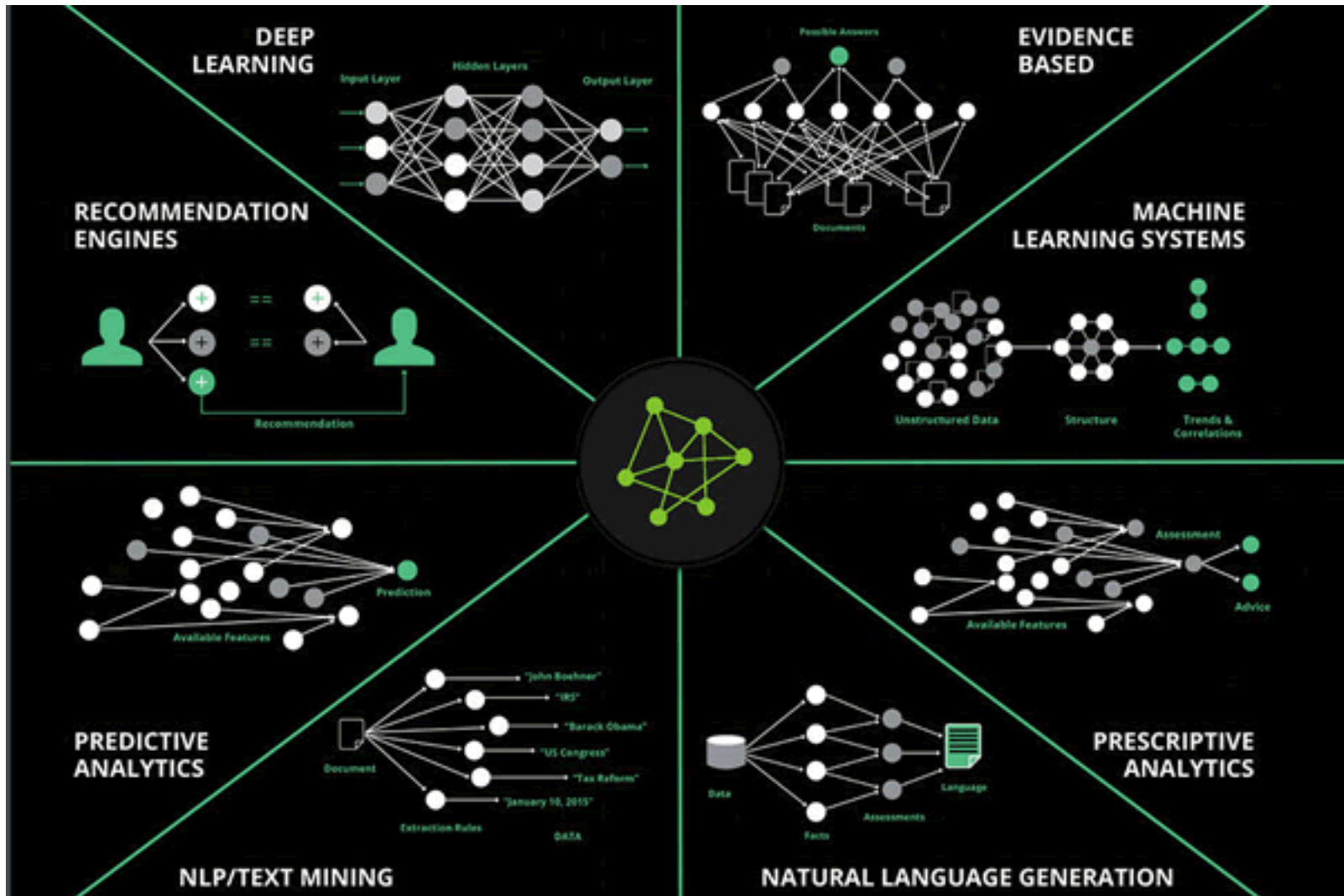
DEEP LEARNING

Deep learning breakthroughs drive AI boom.



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Artificial Intelligence (AI) is many things

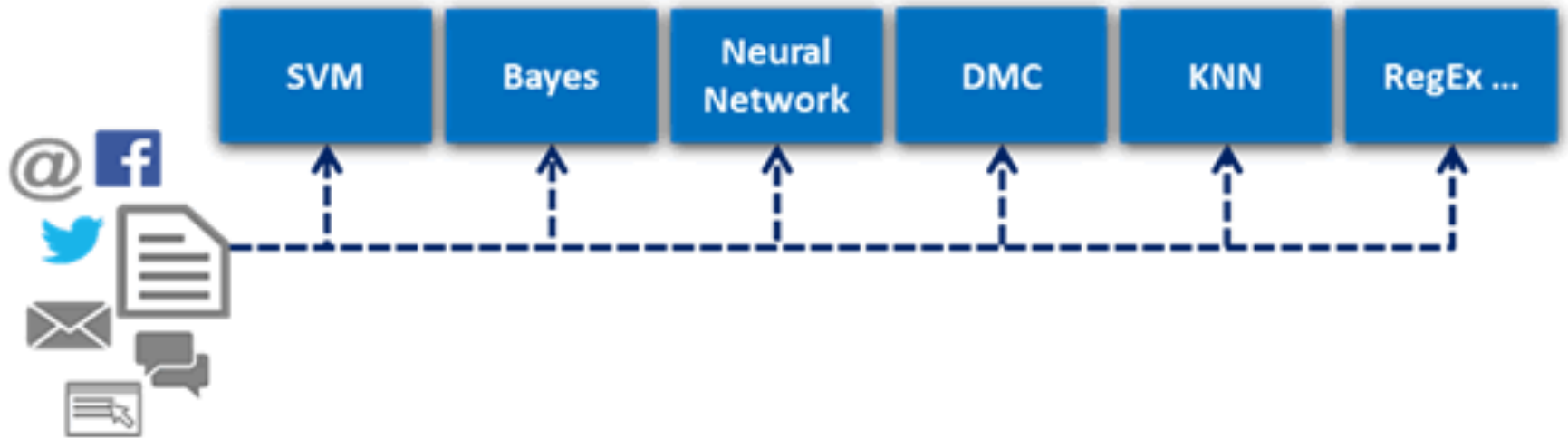


Ecosystem of AI

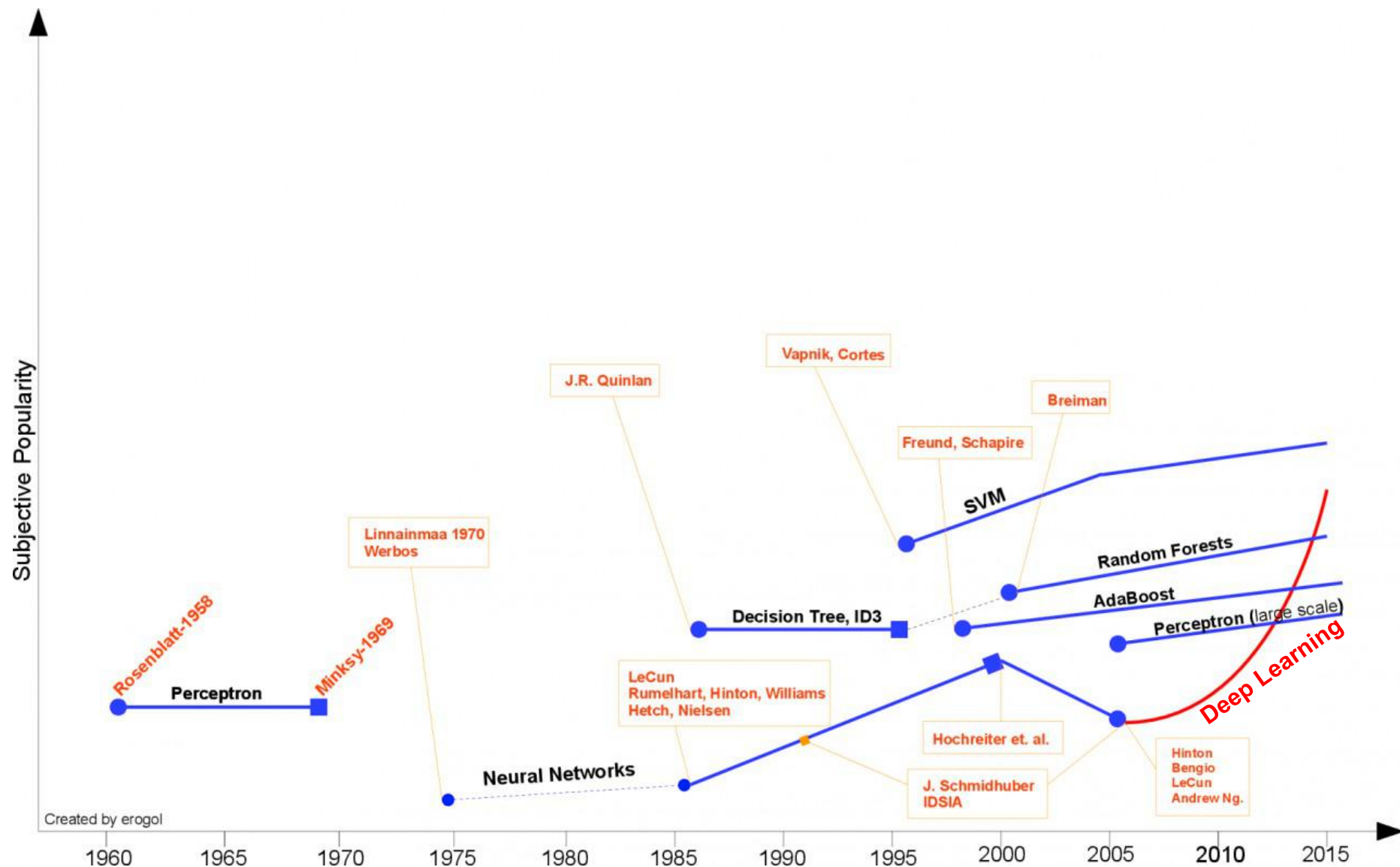
Source: <https://www.i-scoop.eu/artificial-intelligence-cognitive-computing/>

Artificial Intelligence (AI)

Intelligent Document Recognition algorithms



Deep Learning Evolution



Machine Learning Models

Deep Learning

Kernel

Association rules

Ensemble

Decision tree

Dimensionality reduction

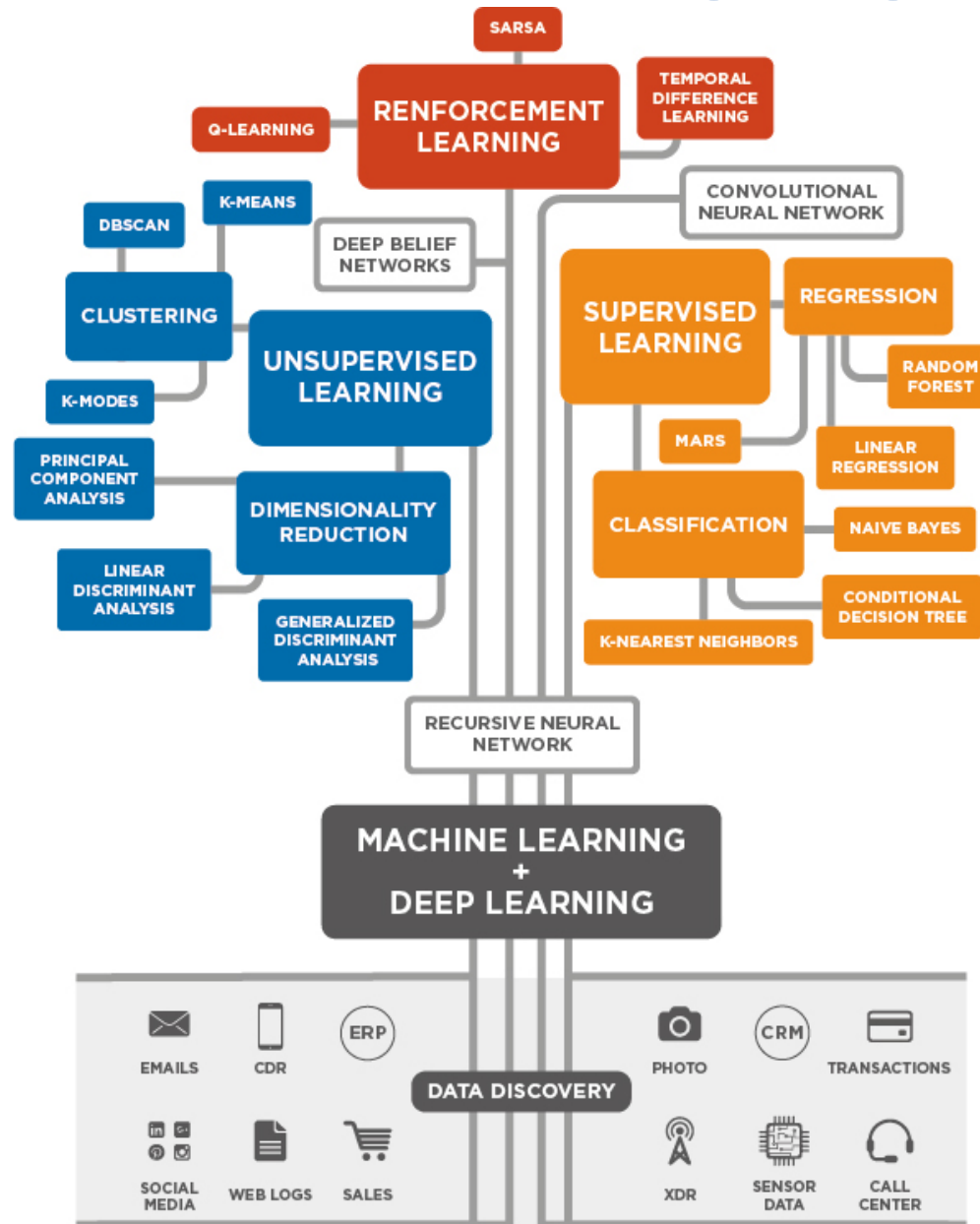
Clustering

Regression Analysis

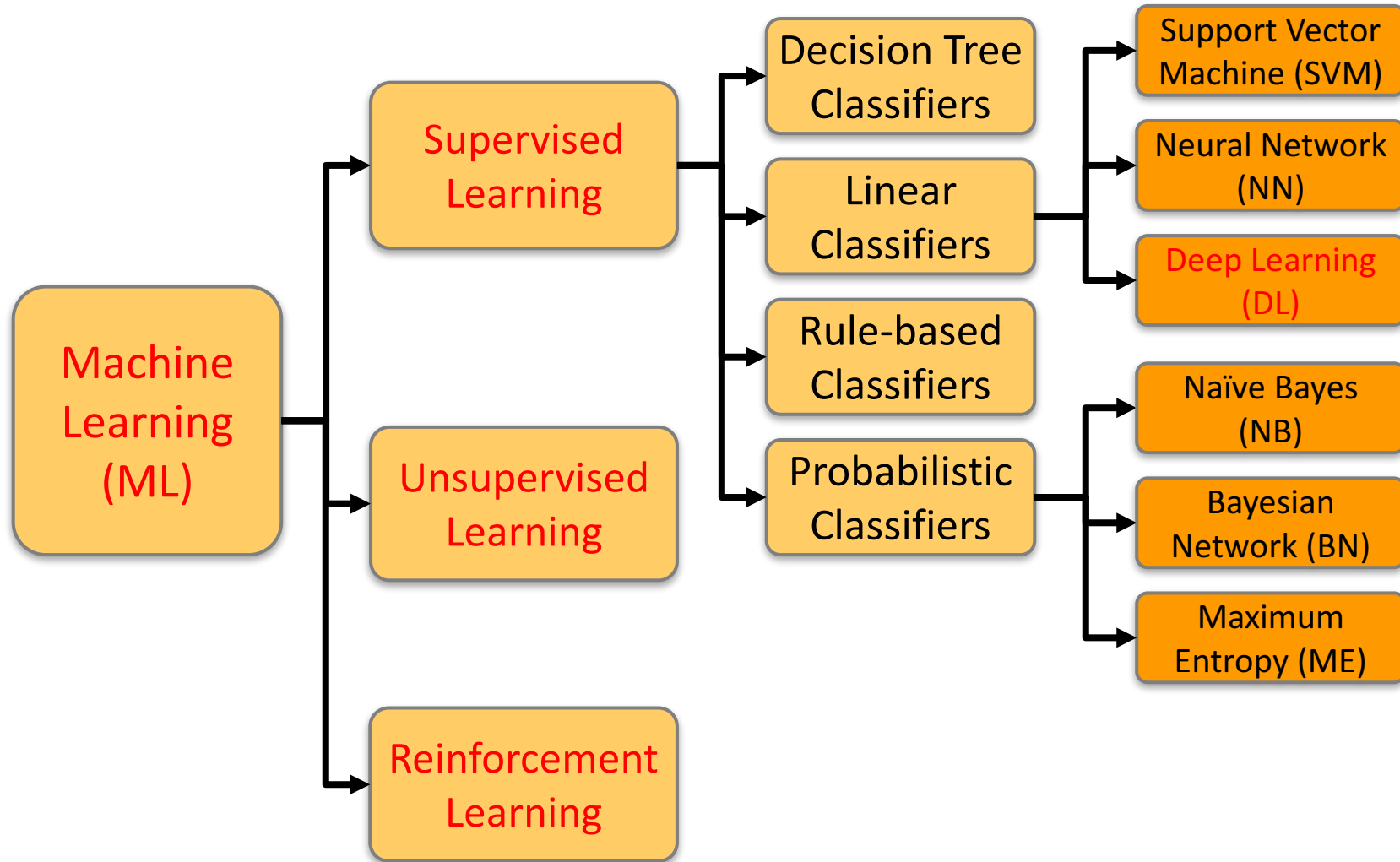
Bayesian

Instance based

3 Machine Learning Algorithms



Machine Learning (ML) / Deep Learning (DL)



Deep Learning and Deep Neural Networks

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

Nature 521, no. 7553 (2015): 436-
444.

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

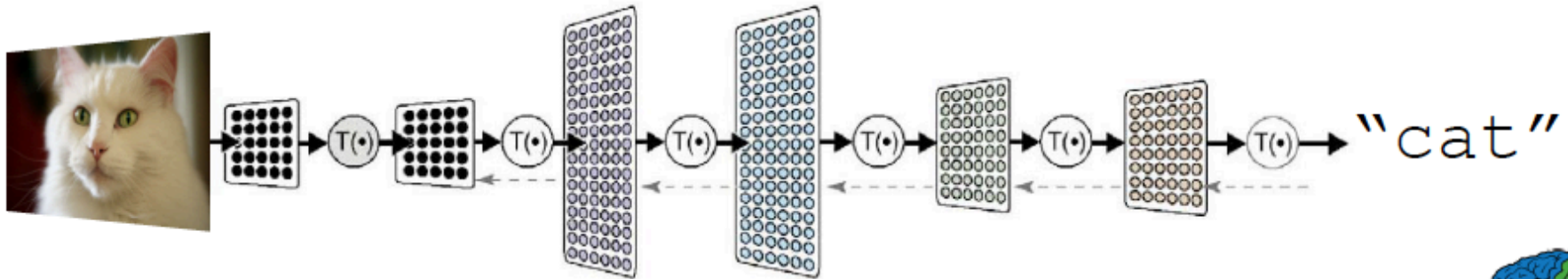
Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, con-

intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition^{1–4} and speech recognition^{5–7}, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules⁸, analysing particle accelerator data^{9,10}, reconstructing brain circuits¹¹, and predicting the effects of mutations in non-coding DNA on gene expression and disease^{12,13}. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding¹⁴, particularly topic classification, sentiment analysis, question answering¹⁵ and language translation^{16,17}.

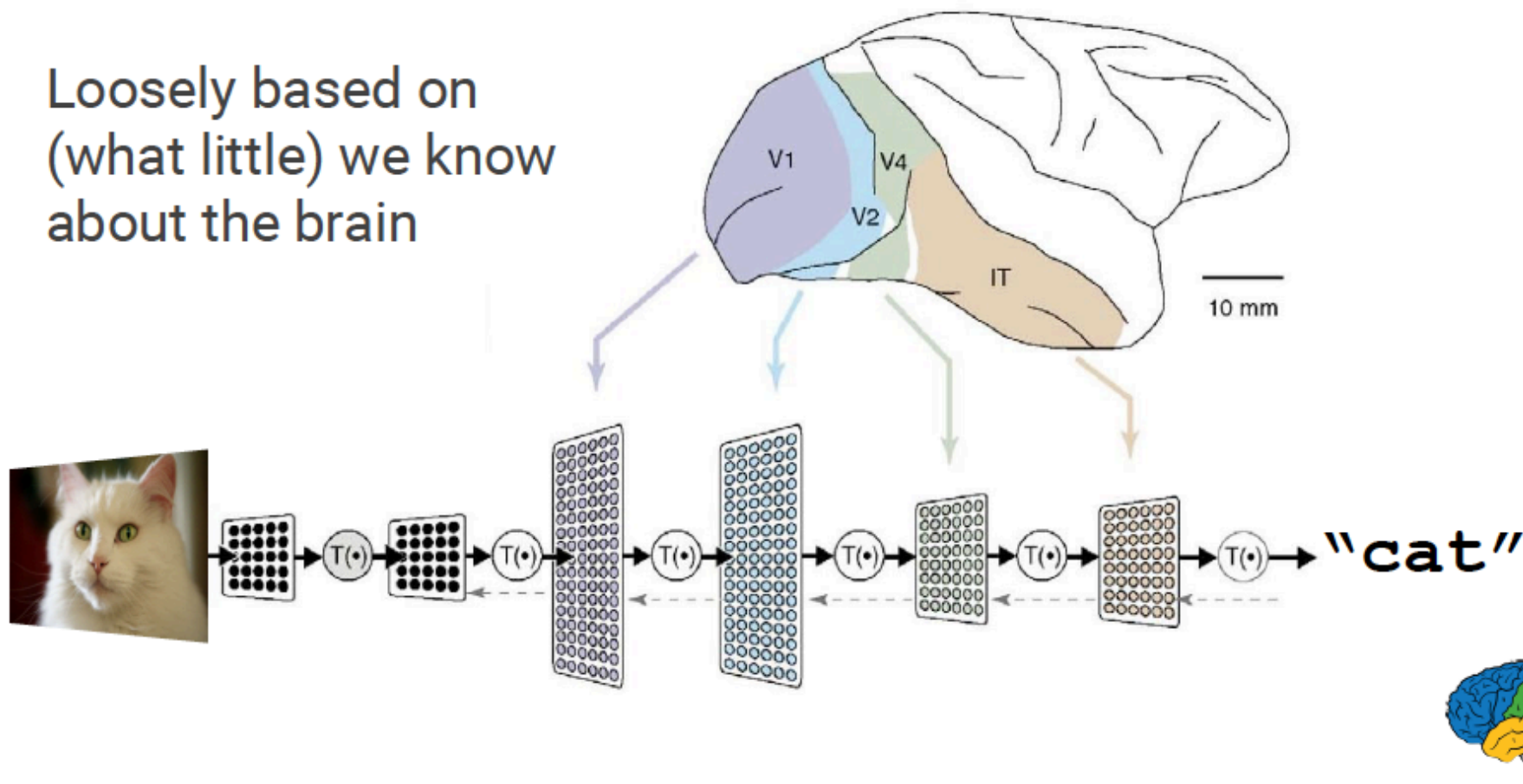
Deep Learning

- A powerful class of **machine learning** model
- **Modern reincarnation** of **artificial neural networks**
- Collection of simple, trainable mathematical functions
- Compatible with many variants of machine learning

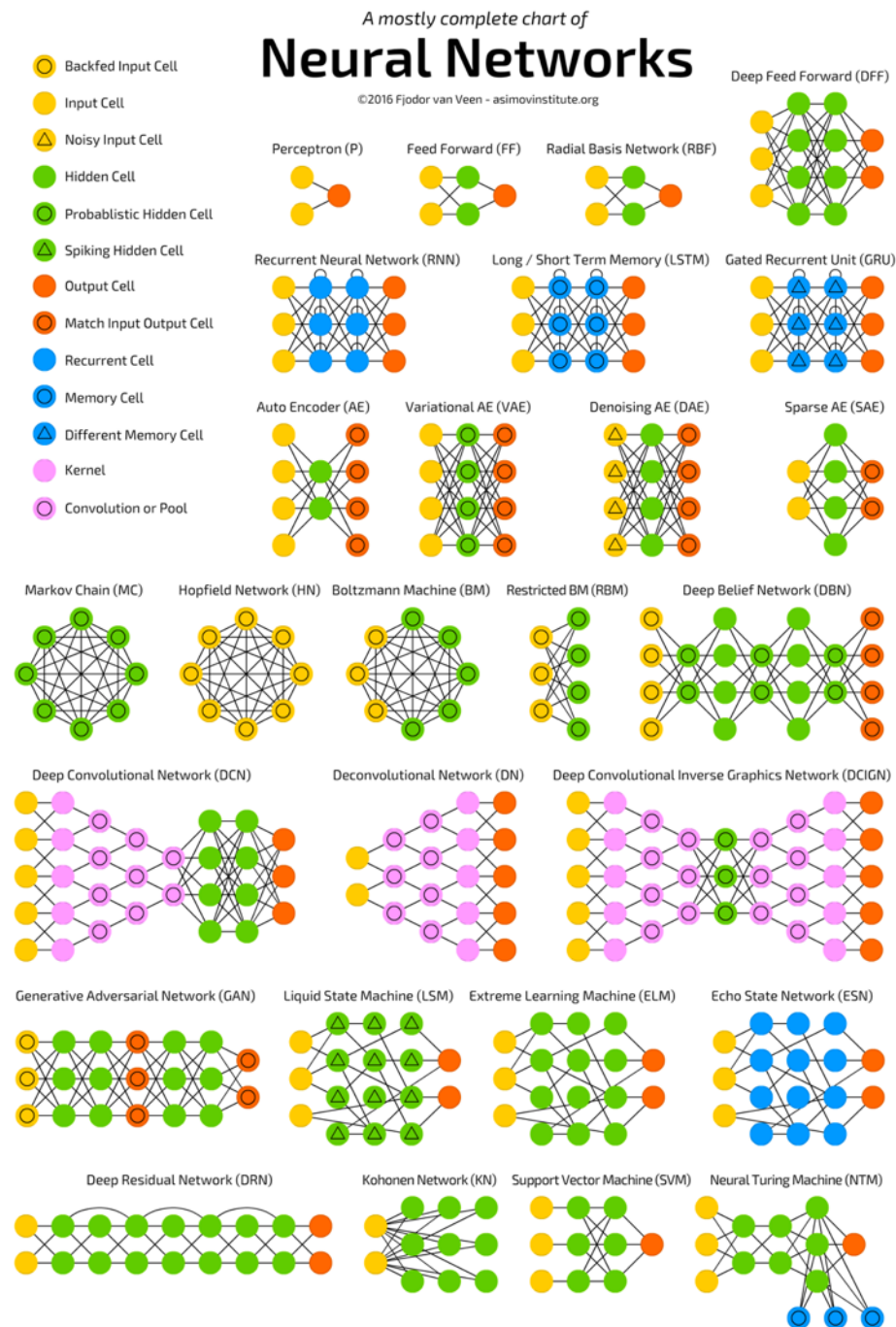


What is Deep Learning?

- Loosely based on (what little) we know about the brain



Neural Networks (NN)



Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probablistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

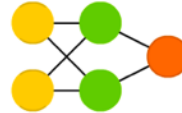
Deep Feed Forward (DFF)



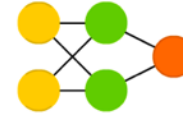
Perceptron (P)



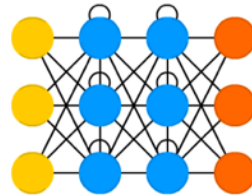
Feed Forward (FF)



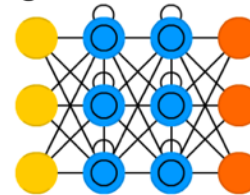
Radial Basis Network (RBF)



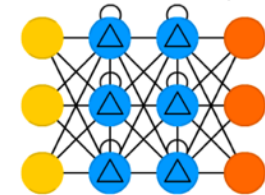
Recurrent Neural Network (RNN)



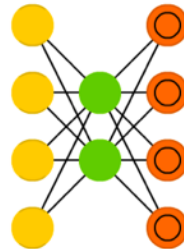
Long / Short Term Memory (LSTM)



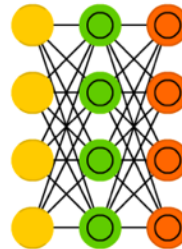
Gated Recurrent Unit (GRU)



Auto Encoder (AE)



Variational AE (VAE)



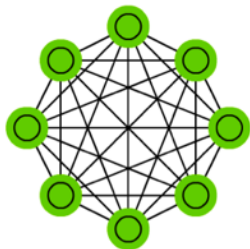
Denoising AE (DAE)



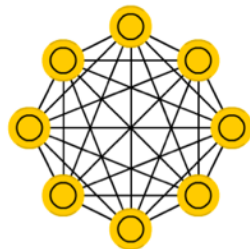
Sparse AE (SAE)



Markov Chain (MC)



Hopfield Network (HN)



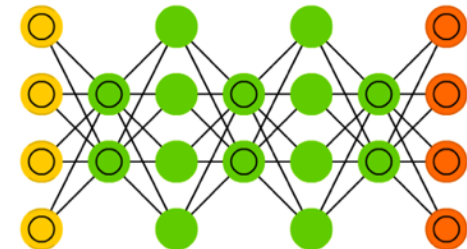
Boltzmann Machine (BM)



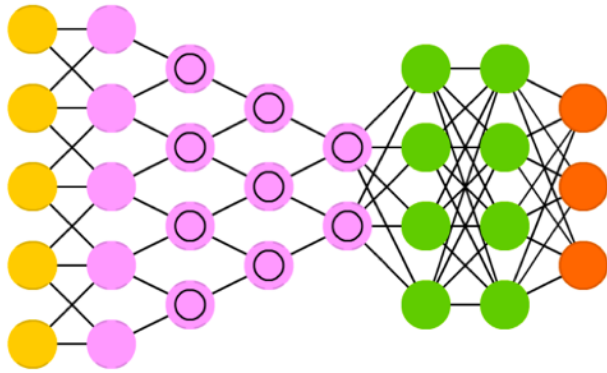
Restricted BM (RBM)



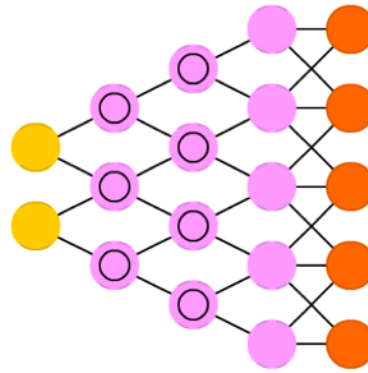
Deep Belief Network (DBN)



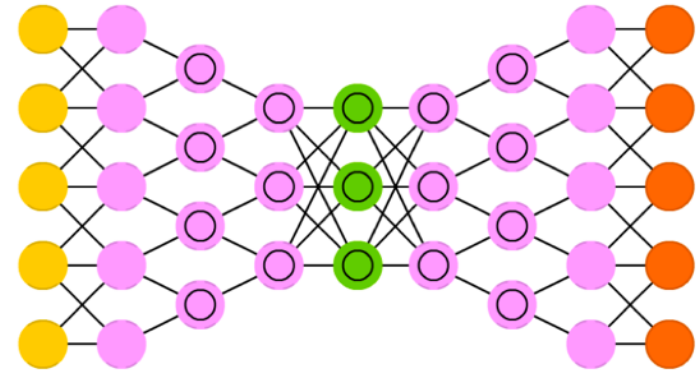
Deep Convolutional Network (DCN)



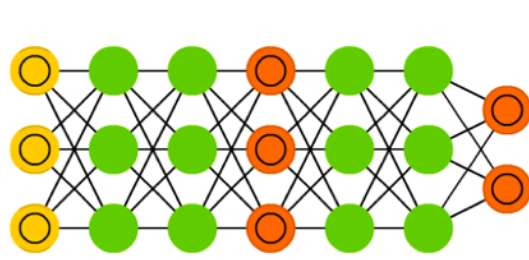
Deconvolutional Network (DN)



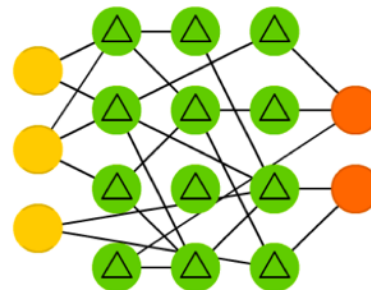
Deep Convolutional Inverse Graphics Network (DCIGN)



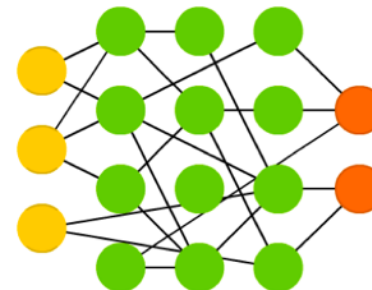
Generative Adversarial Network (GAN)



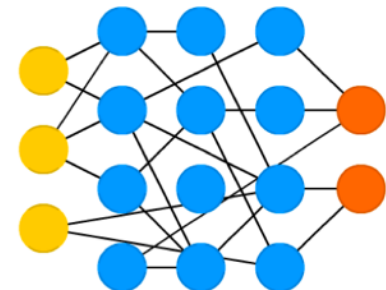
Liquid State Machine (LSM)



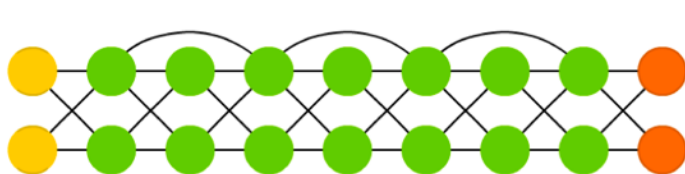
Extreme Learning Machine (ELM)



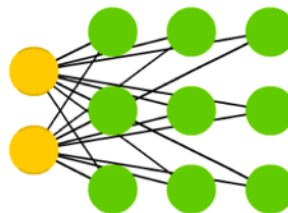
Echo State Network (ESN)



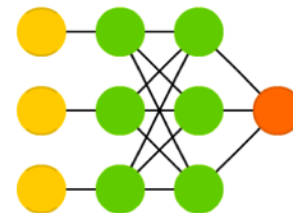
Deep Residual Network (DRN)



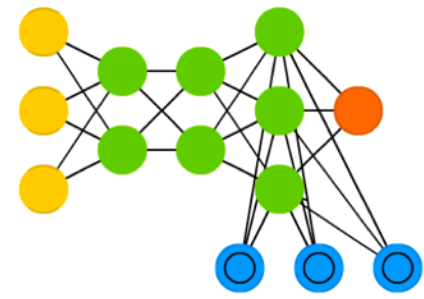
Kohonen Network (KN)



Support Vector Machine (SVM)

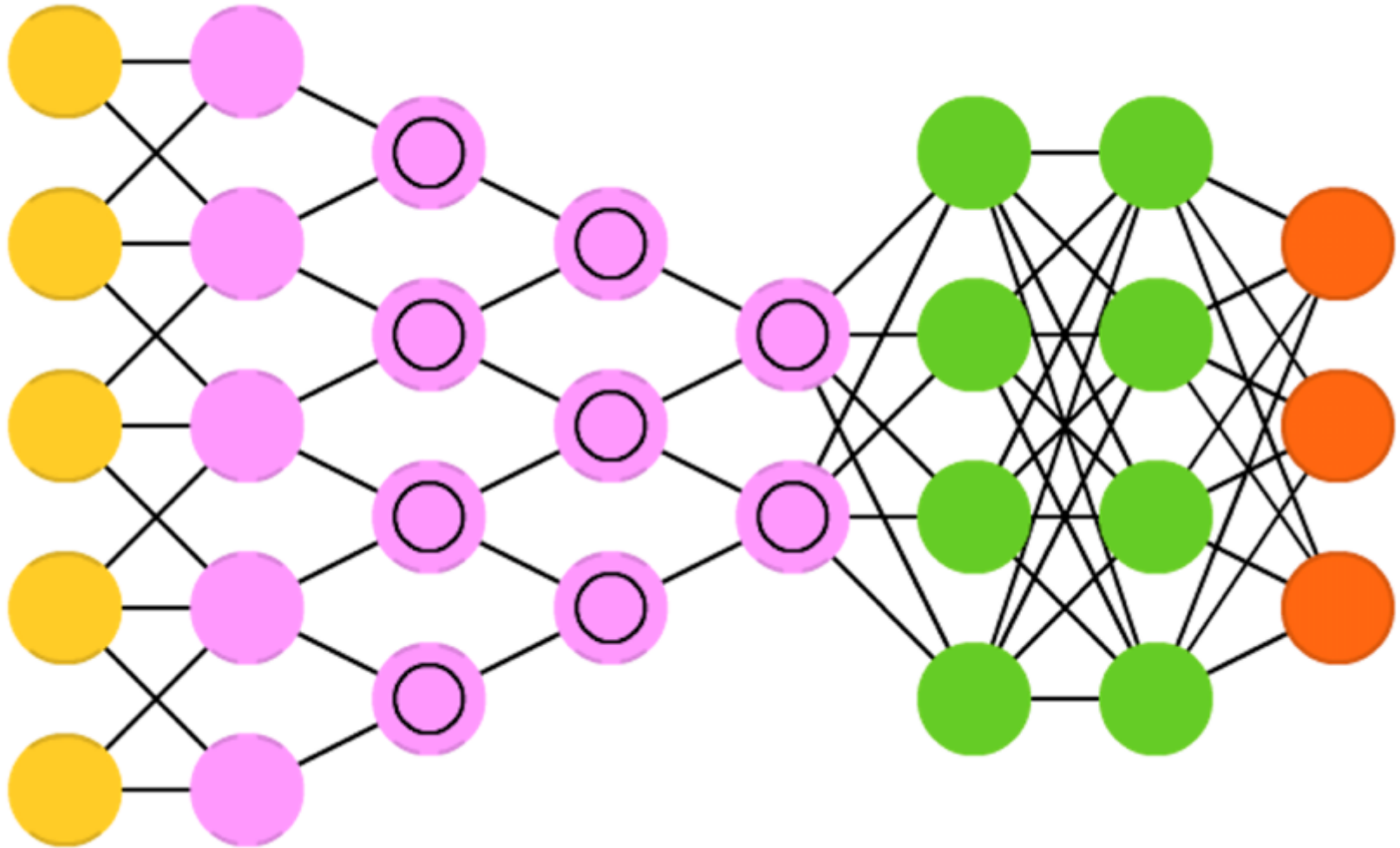


Neural Turing Machine (NTM)

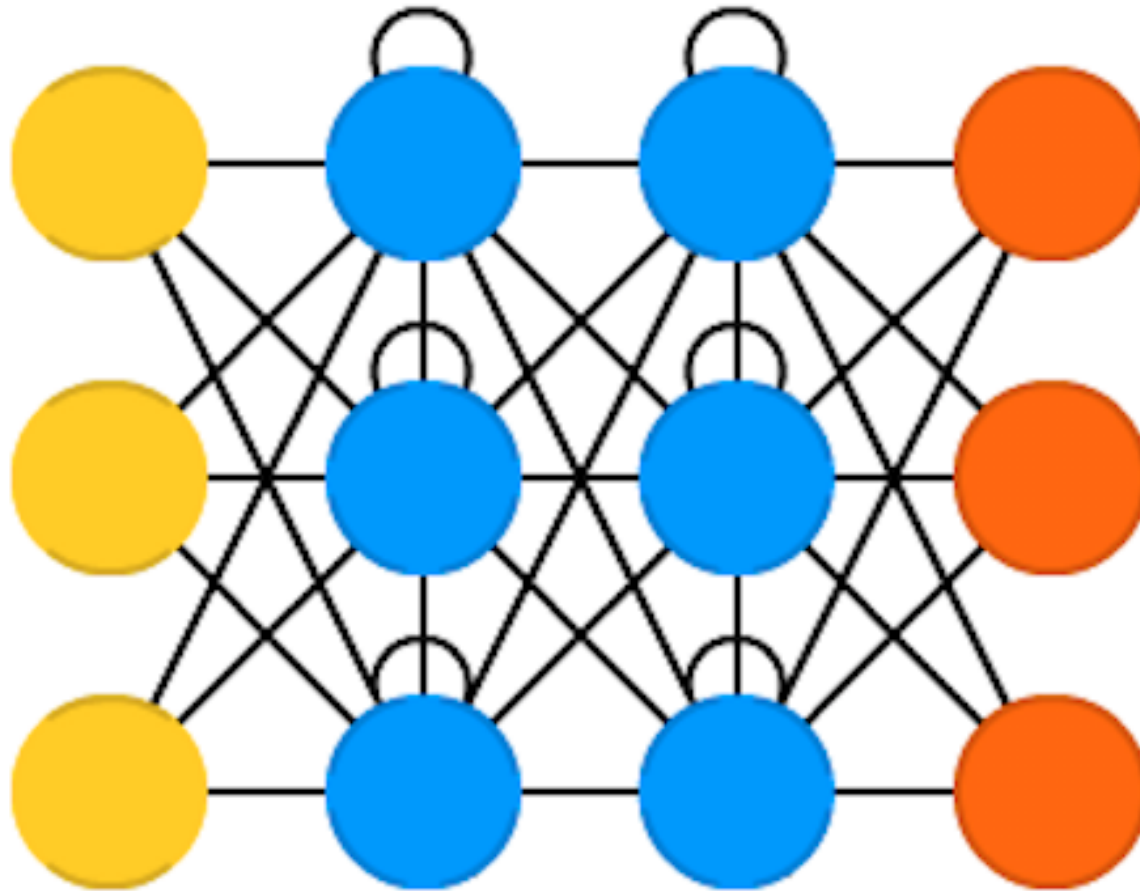


Convolutional Neural Networks

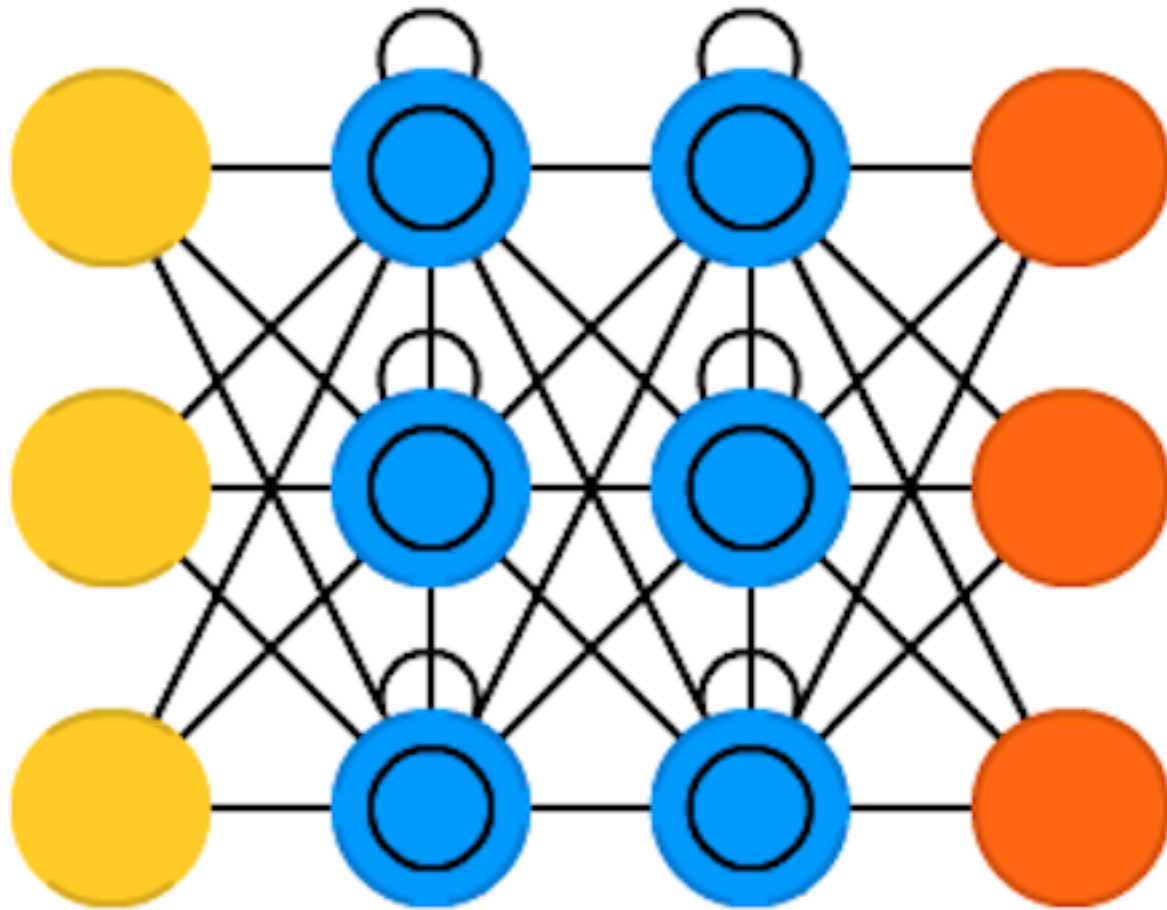
(CNN or Deep Convolutional Neural Networks, DCNN)



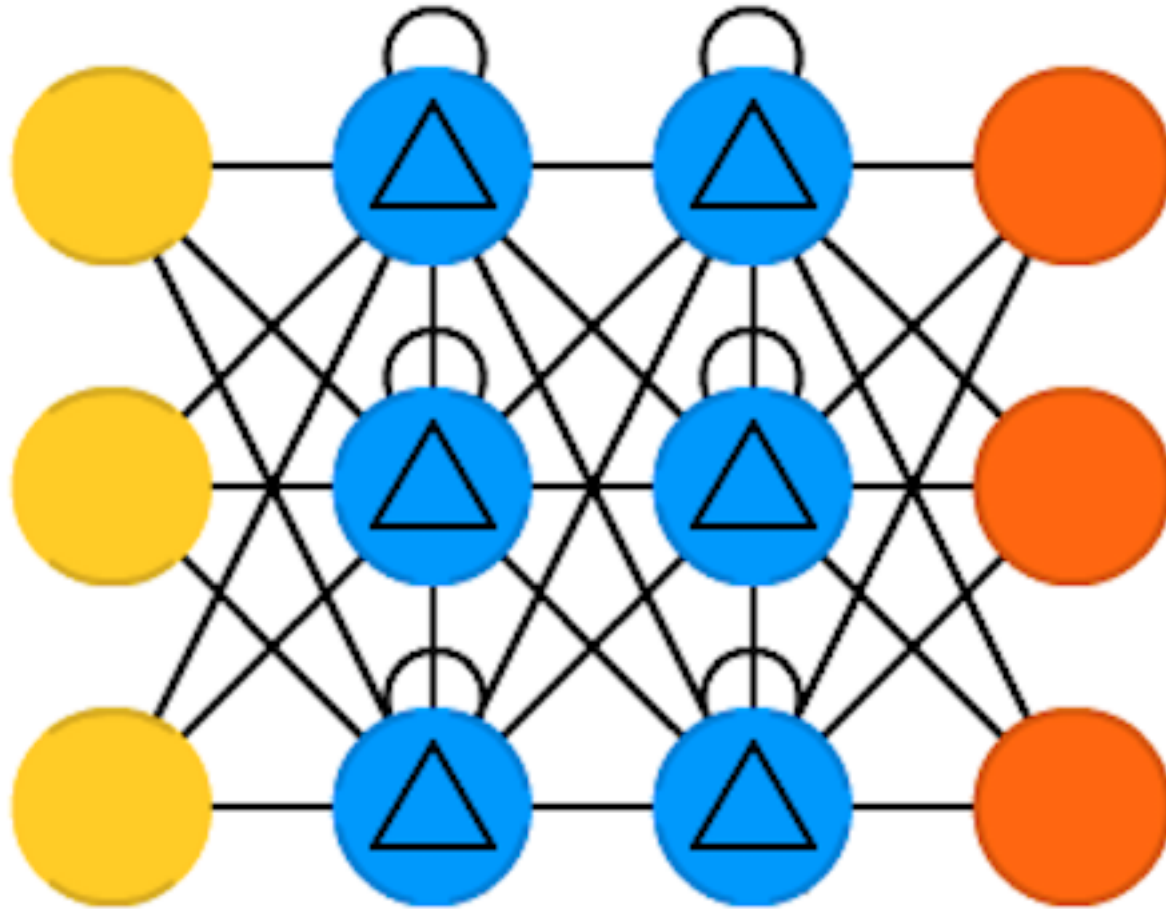
Recurrent Neural Networks (RNN)



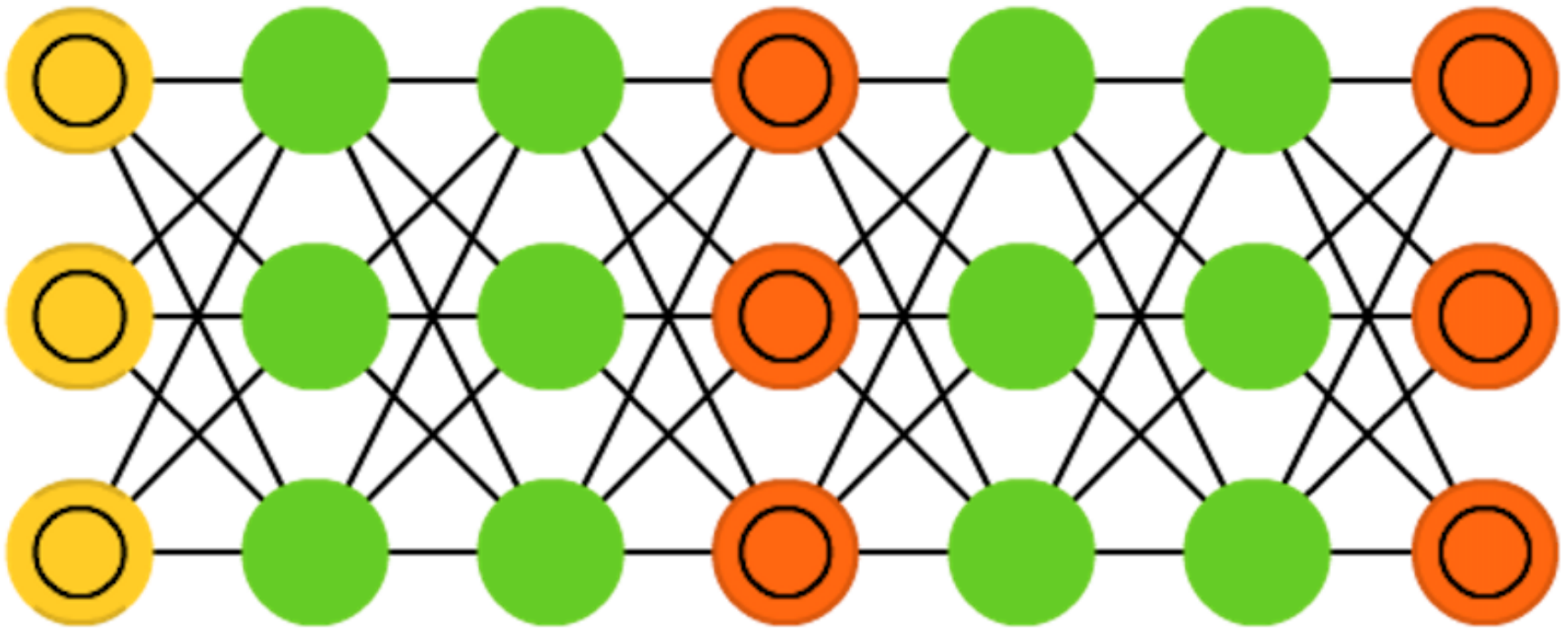
Long / Short Term Memory (LSTM)



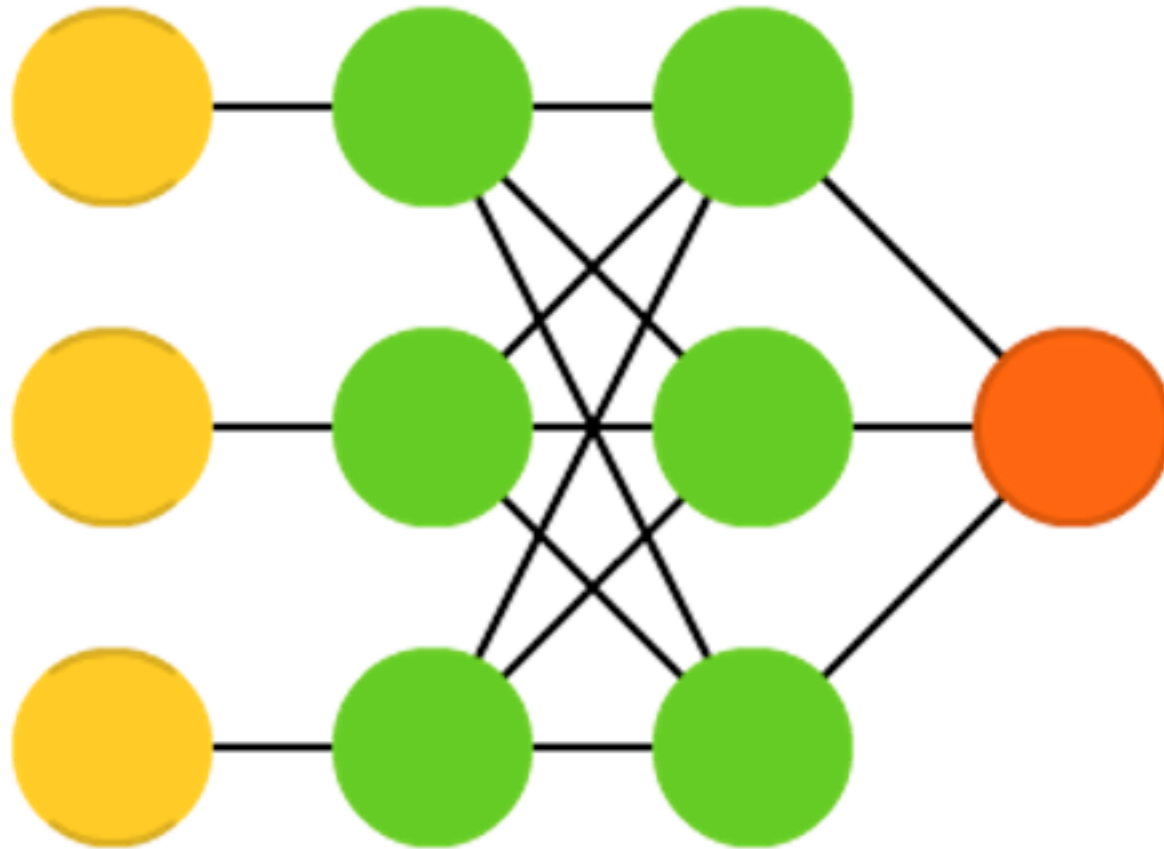
Gated Recurrent Units (GRU)



Generative Adversarial Networks (GAN)



Support Vector Machines (SVM)



Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

Source: <http://www.asimovinstitute.org/neural-network-zoo/>

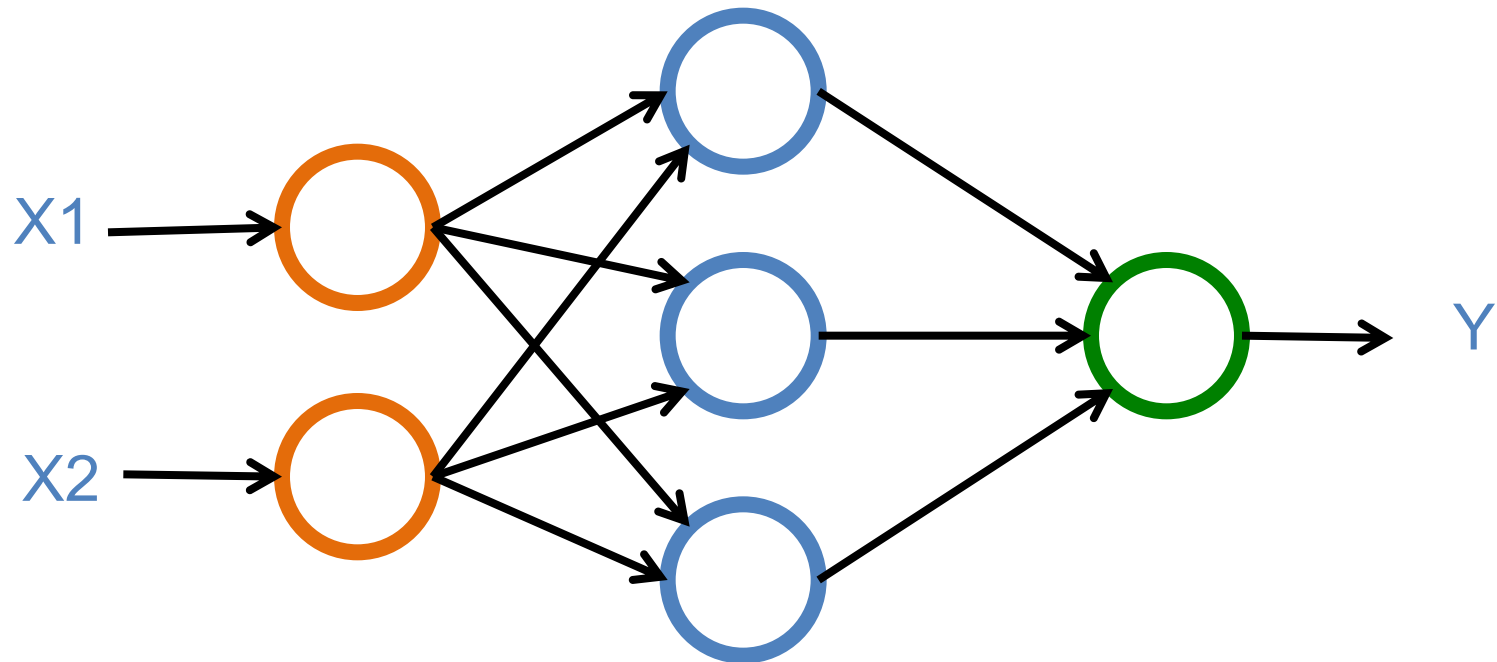
Neural networks (NN) 1960

Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



Multilayer Perceptrons (MLP) 1985

Support Vector Machine (SVM) 1995



Hinton presents the

Deep Belief Network (DBN)

**New interests in deep learning
and RBM**

State of the art MNIST

2005

Deep Recurrent Neural Network (RNN) 2009

Convolutional DBN 2010

Max-Pooling CDBN 2011

Deep Learning

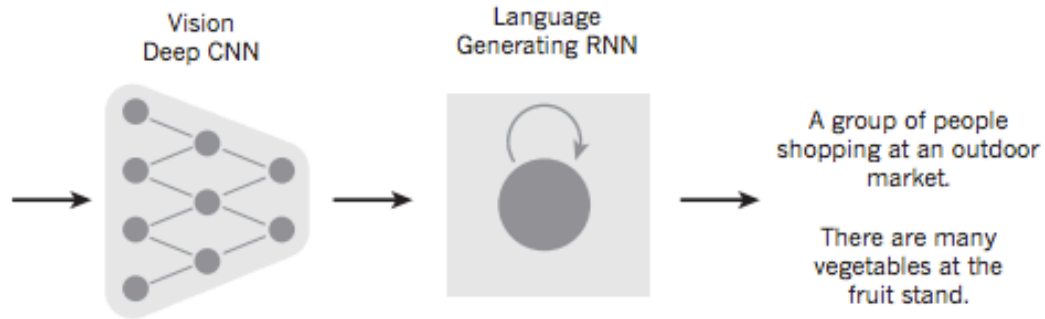
Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng

From image to text



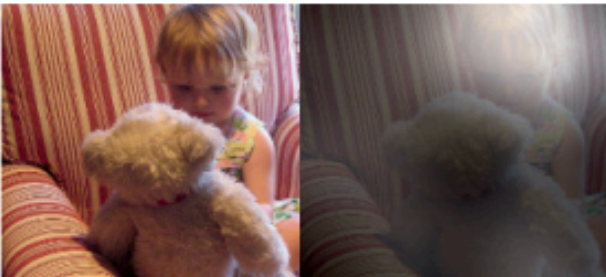
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.



A group of **people** sitting on a boat in the water.



A giraffe standing in a forest with **trees** in the background.

From image to text

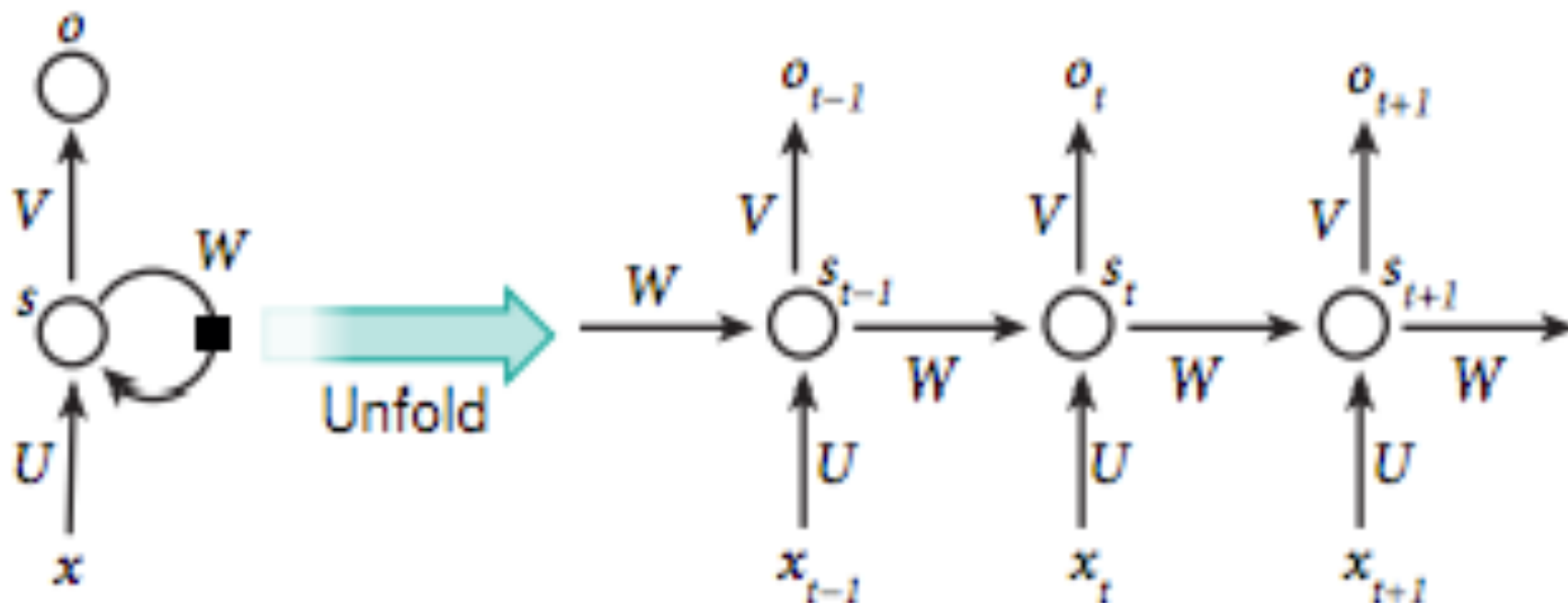
Image: deep convolution neural network (CNN)

Text: recurrent neural network (RNN)



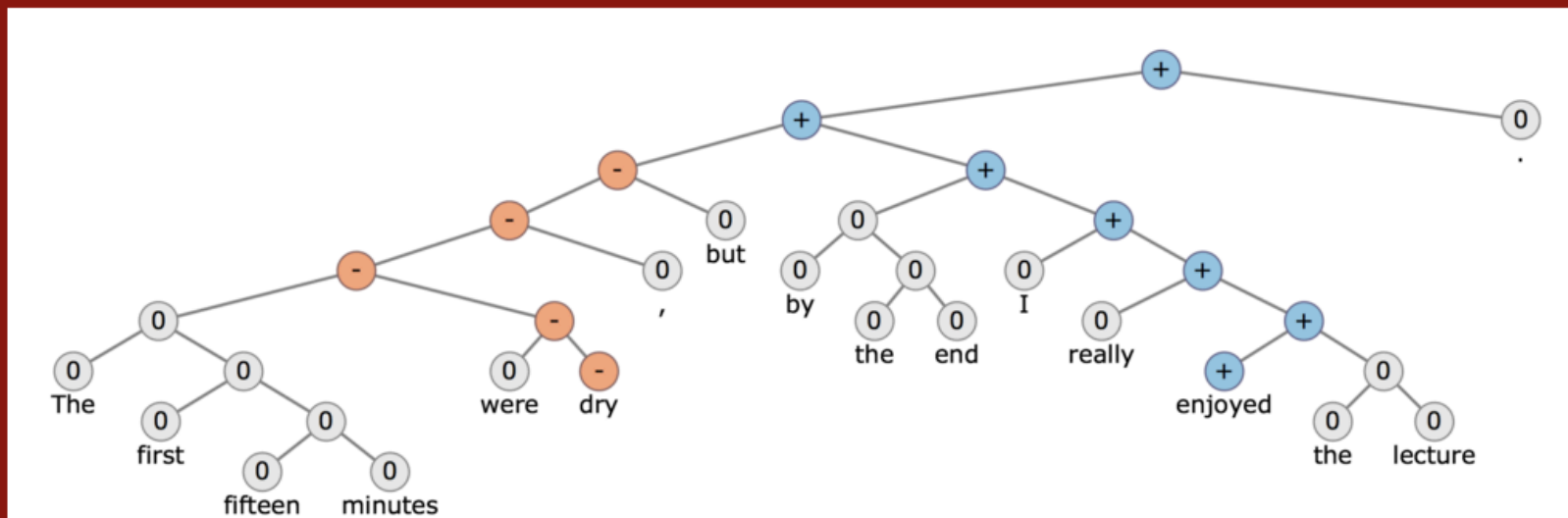
A group of **people** sitting on a boat in the water.

Recurrent Neural Network (RNN)



CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

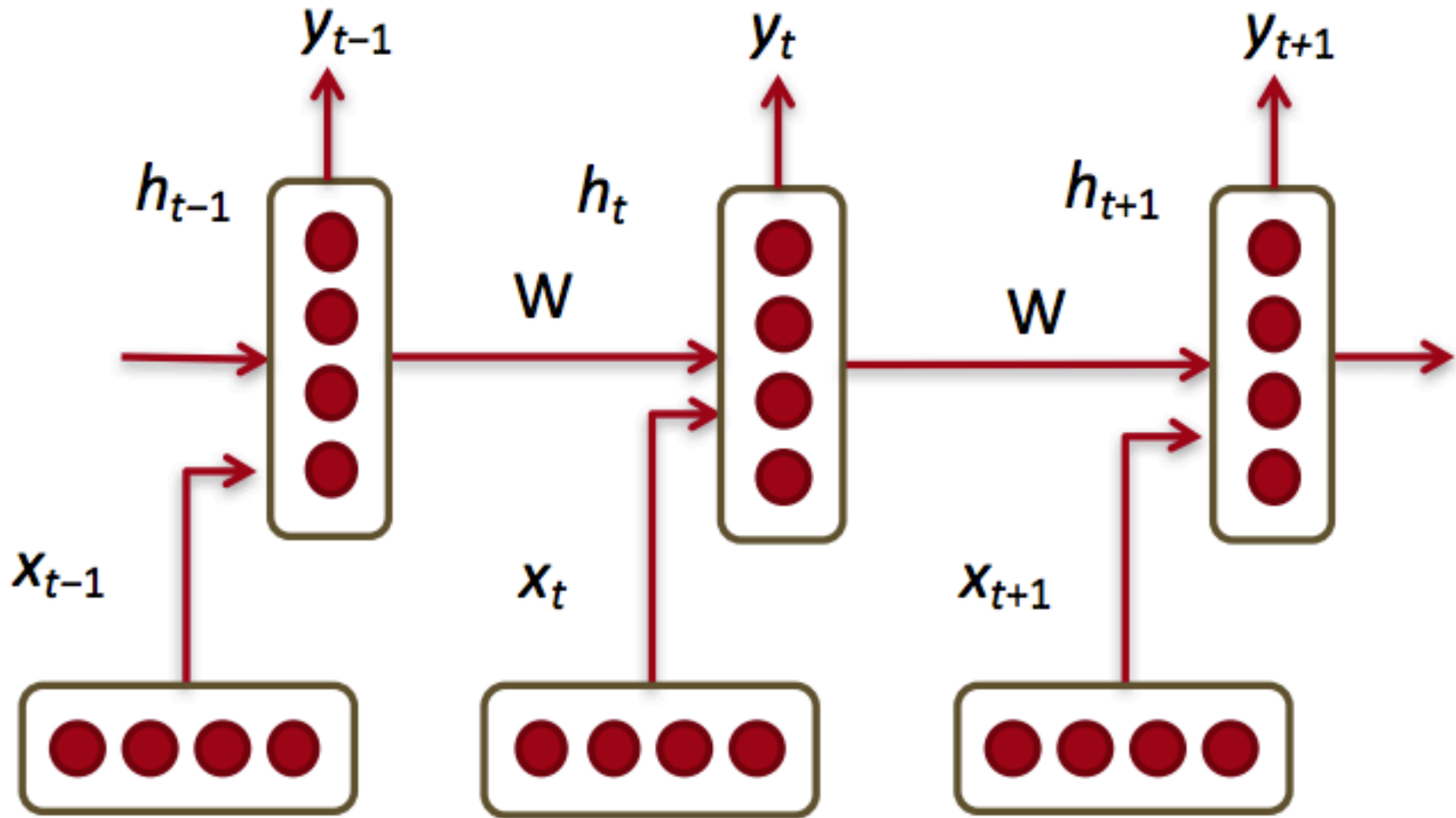


Course Description

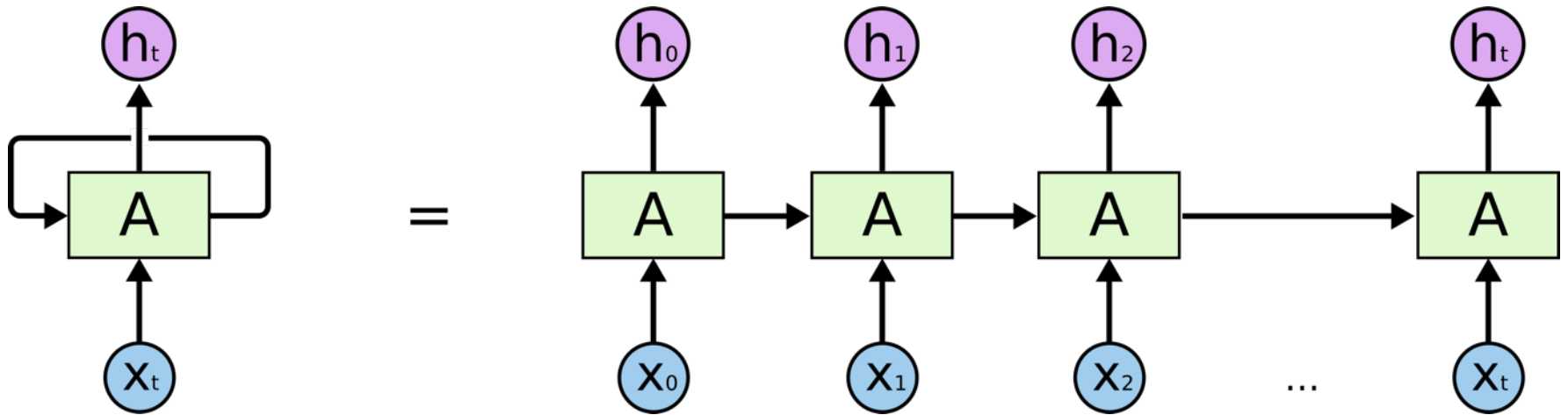
Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, emails, customer service, language translation, radiology reports, etc. There are a large variety of underlying tasks and machine learning models powering NLP applications. Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering. In this spring quarter course students will learn to implement, train, debug, visualize and invent their own neural network models. The course provides a deep excursion into cutting-edge research in deep learning applied to NLP. The final project will involve training a complex recurrent neural network and applying it to a large scale NLP problem. On the model side we will cover word vector representations,

<http://cs224d.stanford.edu/>

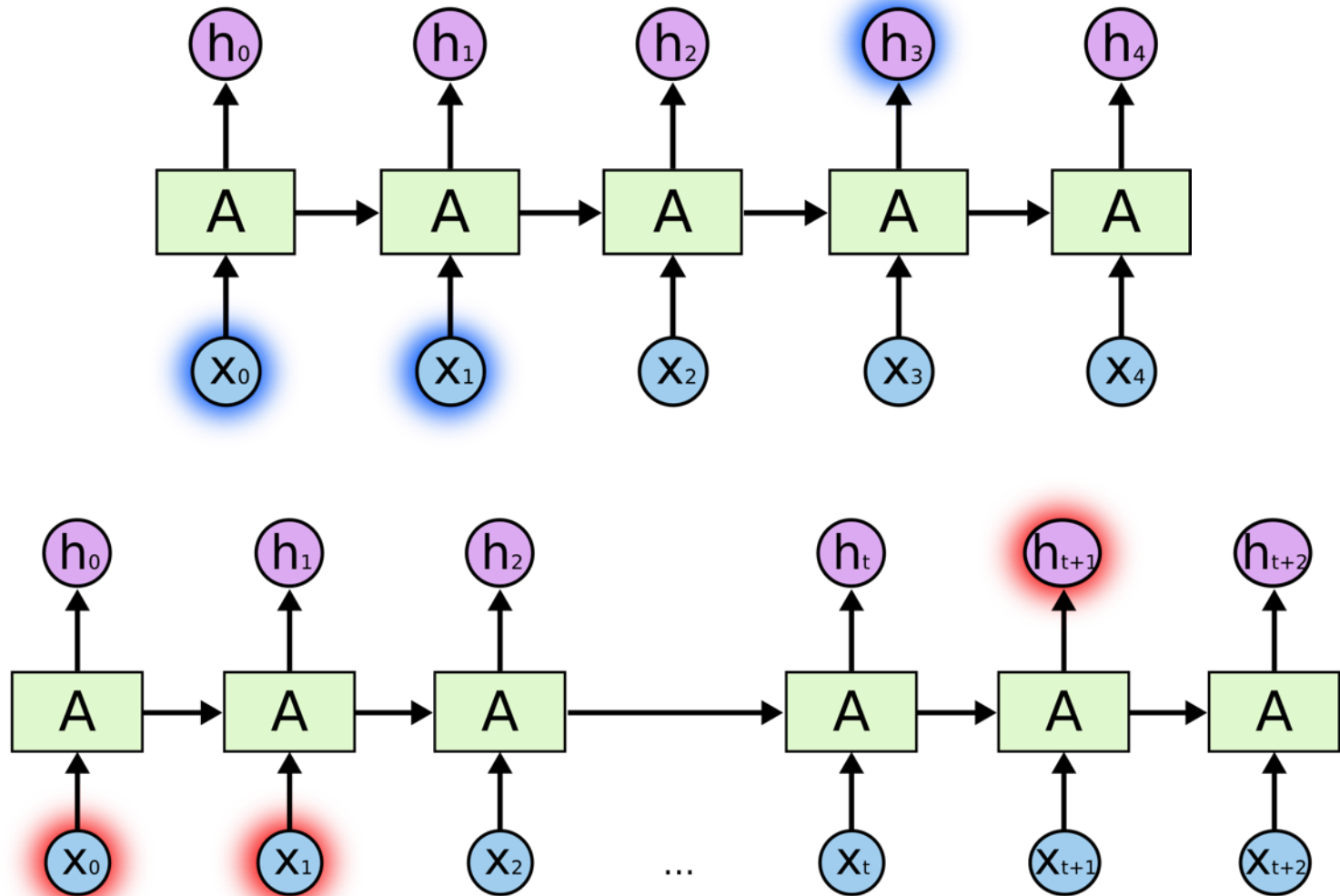
Recurrent Neural Networks (RNNs)



RNN

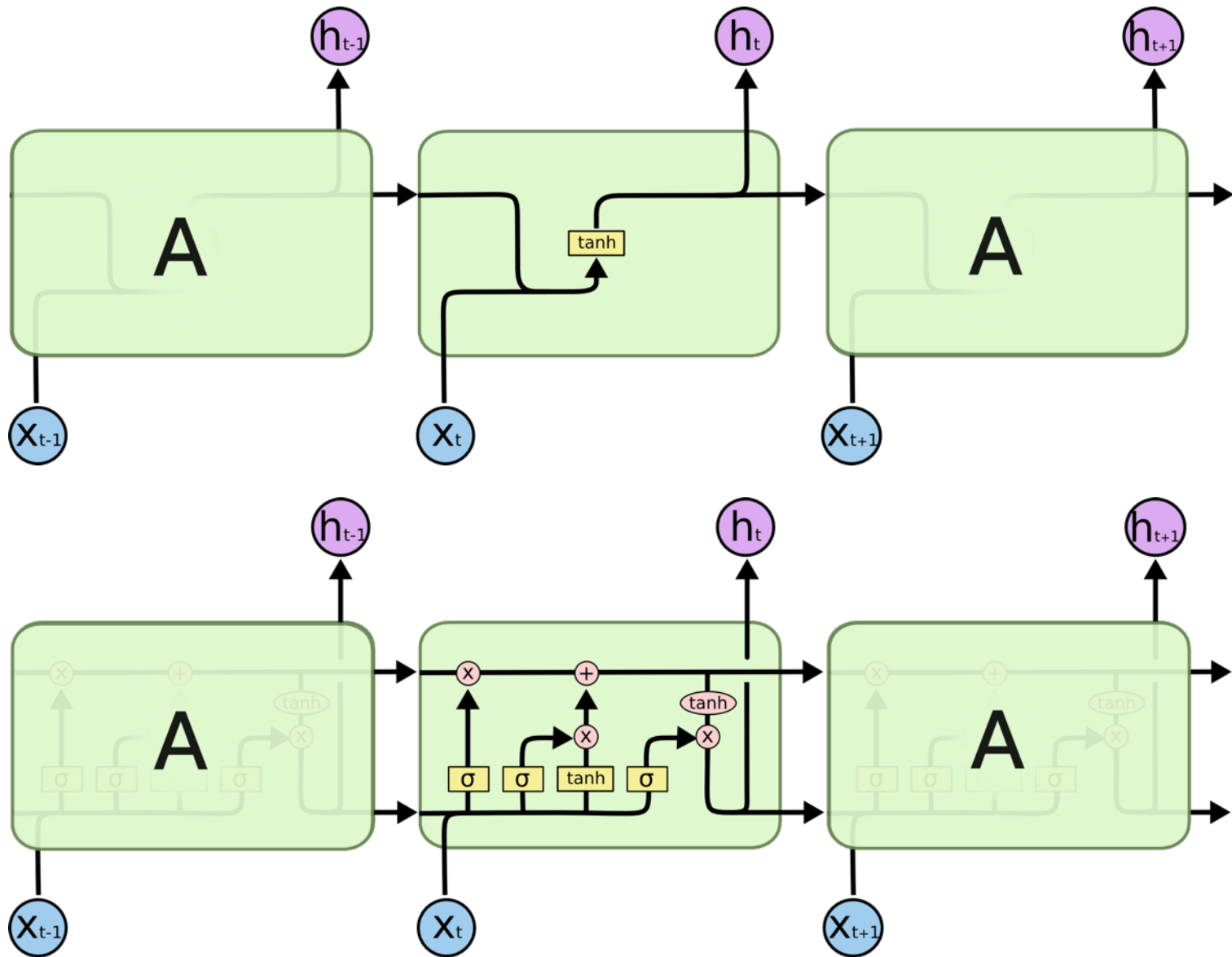


RNN long-term dependencies

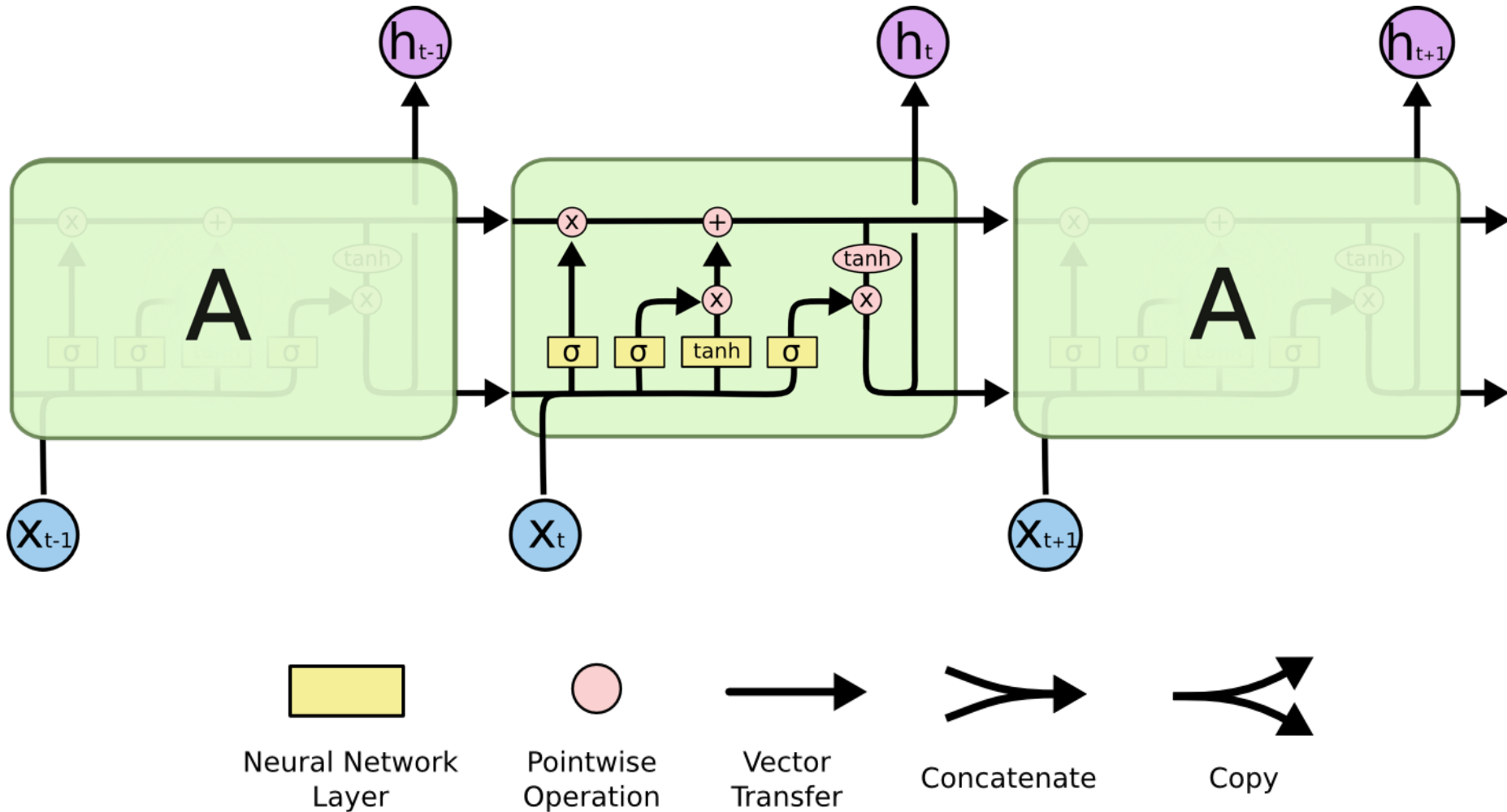


I grew up in France... I speak fluent French.

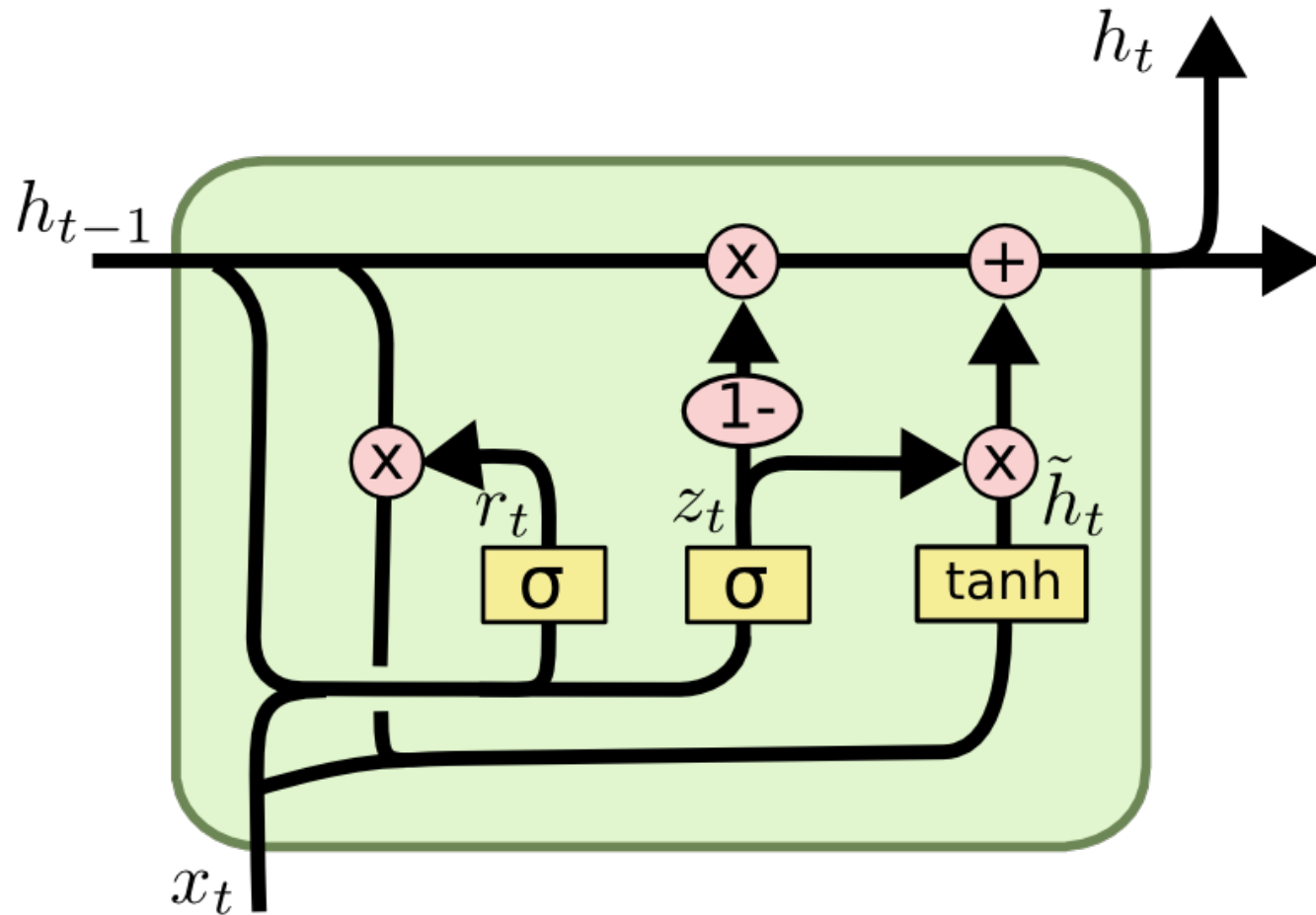
RNN LSTM



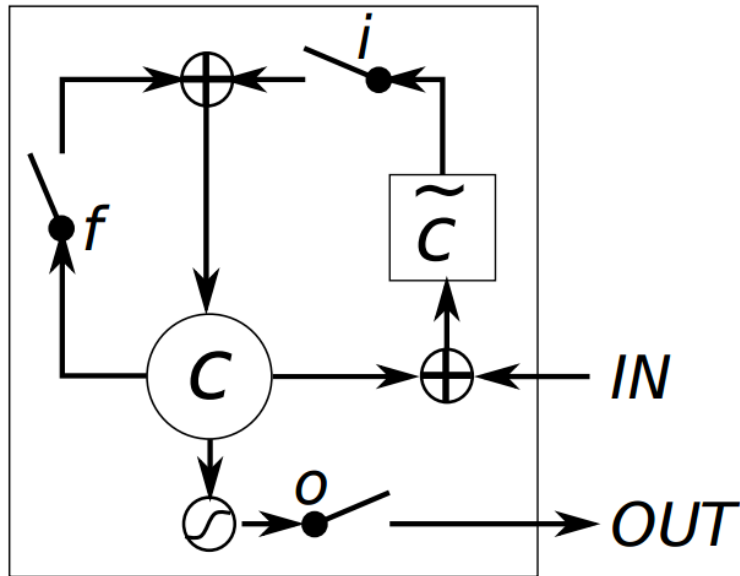
Long Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)

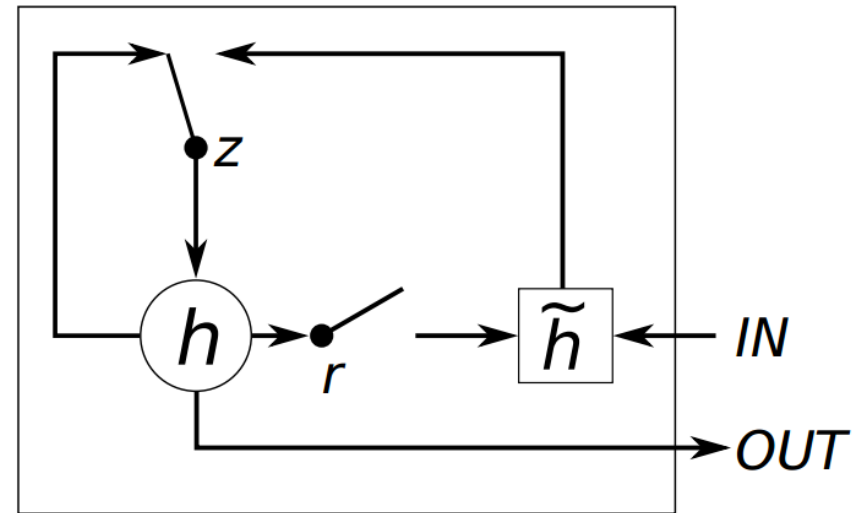


LSTM vs GRU



LSTM

i , f and o are the **input**, **forget** and **output** gates, respectively.
 c and $\tilde{c}$ denote the memory cell and the new memory cell content.

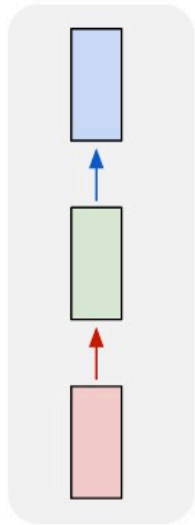


GRU

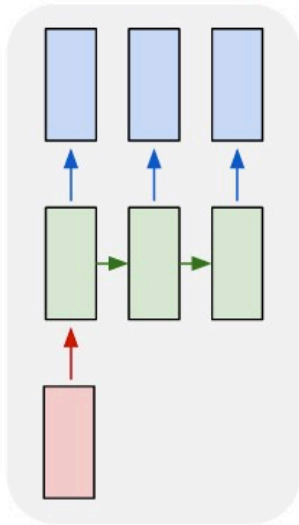
r and z are the **reset** and **update** gates, and h and $\tilde{h}$ are the activation and the candidate activation.

LSTM Recurrent Neural Network

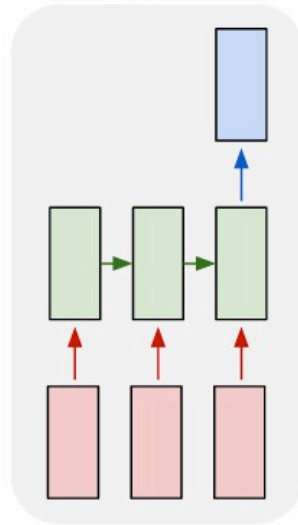
one to one



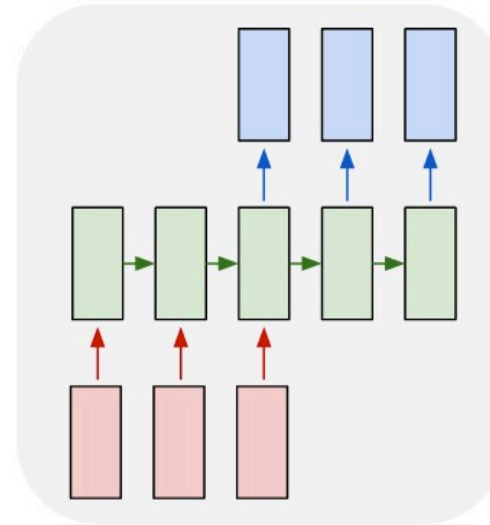
one to many



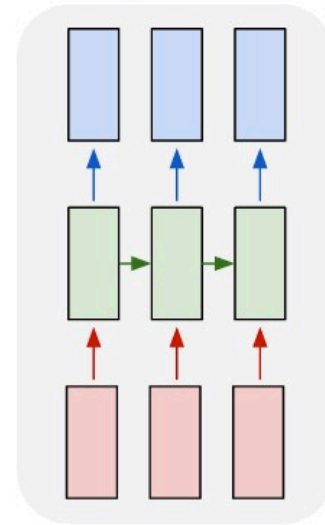
many to one



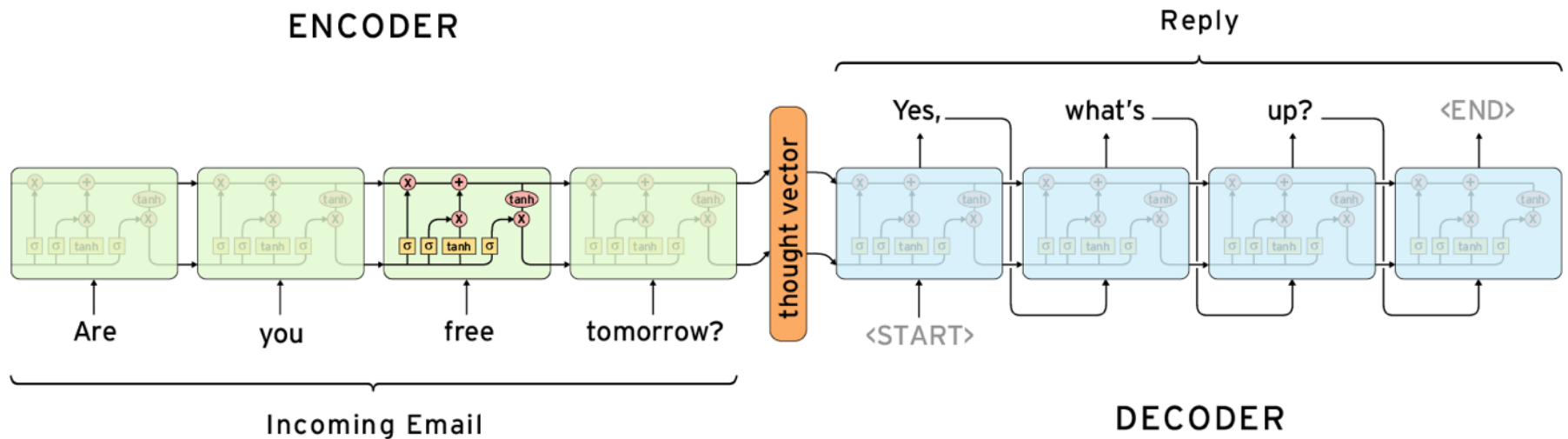
many to many



many to many



The Sequence to Sequence model (seq2seq)

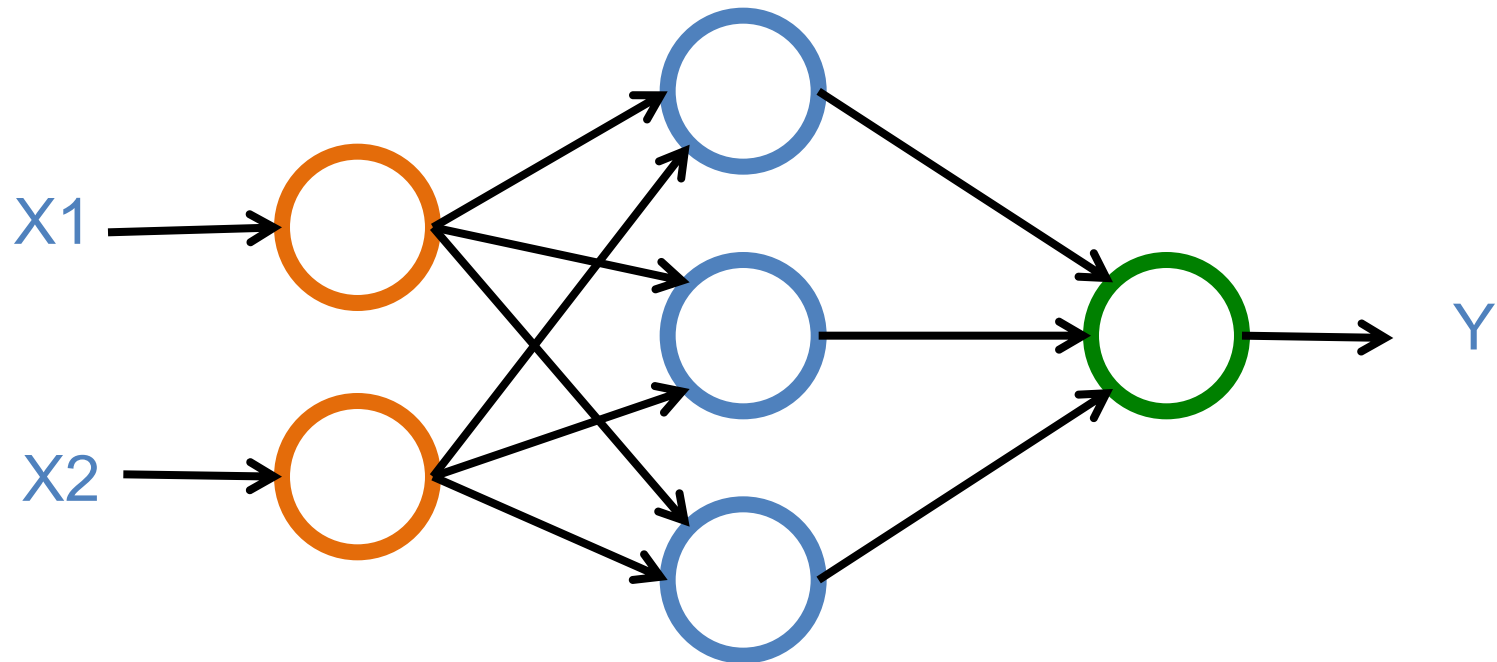


Neural Networks

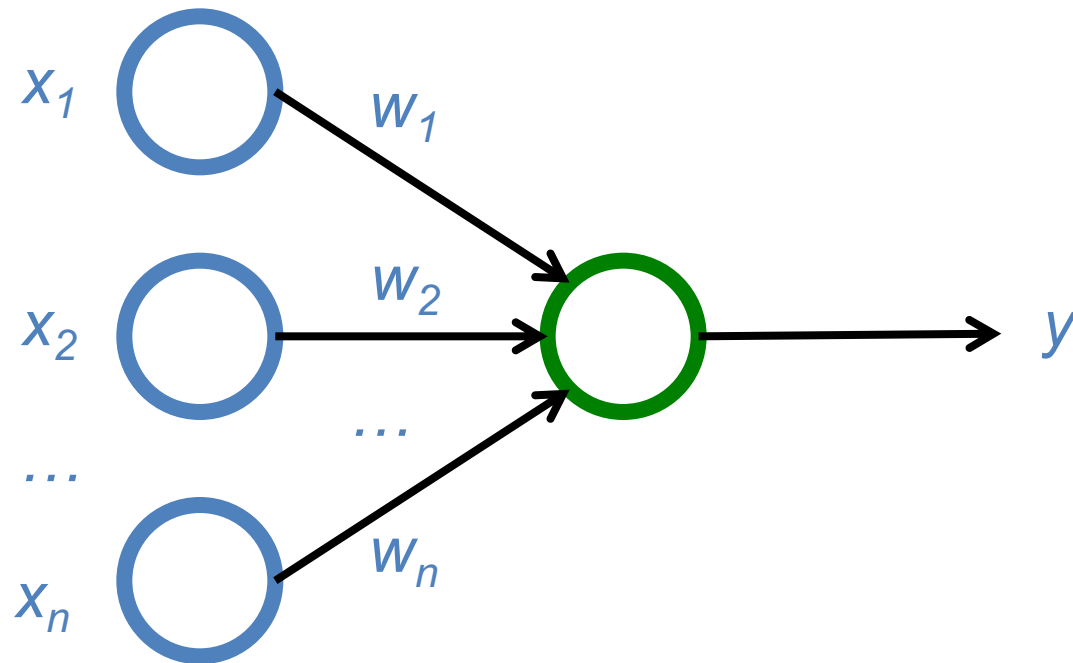
Input Layer
(X)

Hidden Layer
(H)

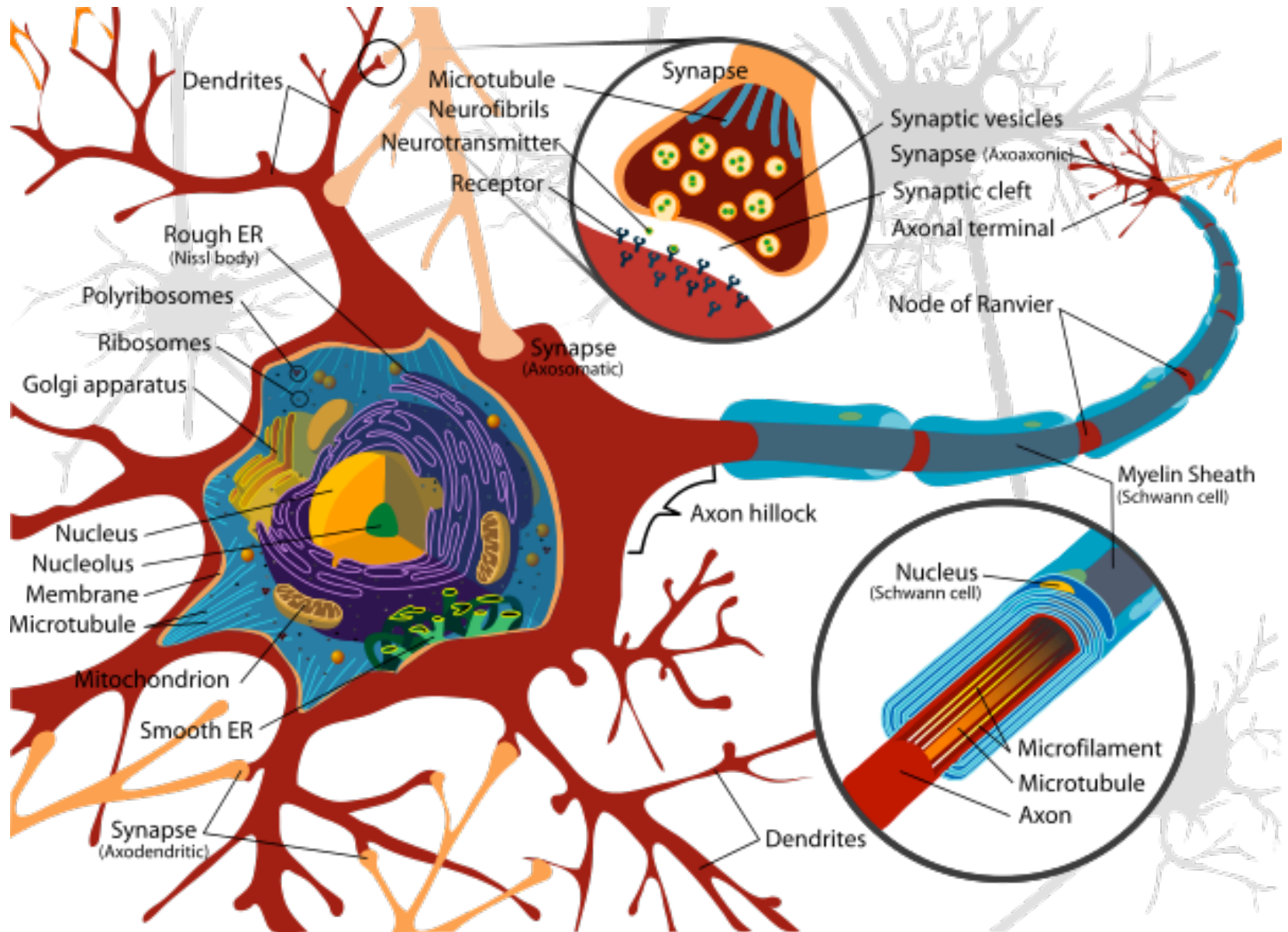
Output Layer
(Y)



The Neuron

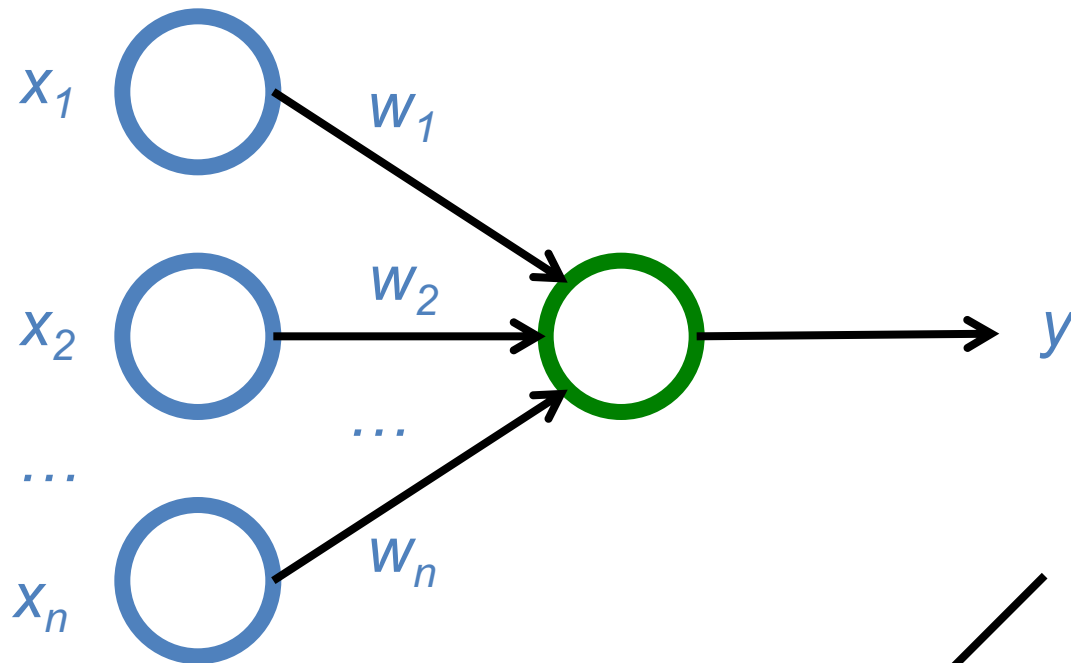


Neuron and Synapse



The Neuron

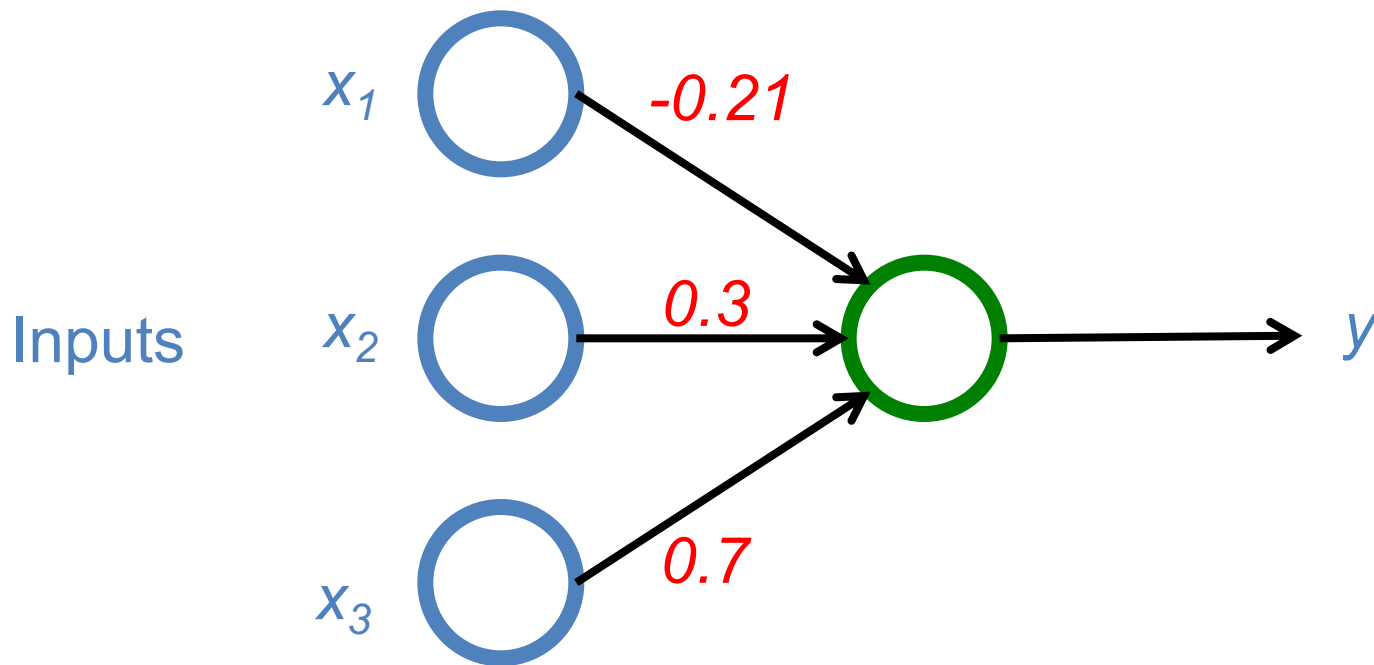
$$y = F\left(\sum_i w_i x_i\right)$$



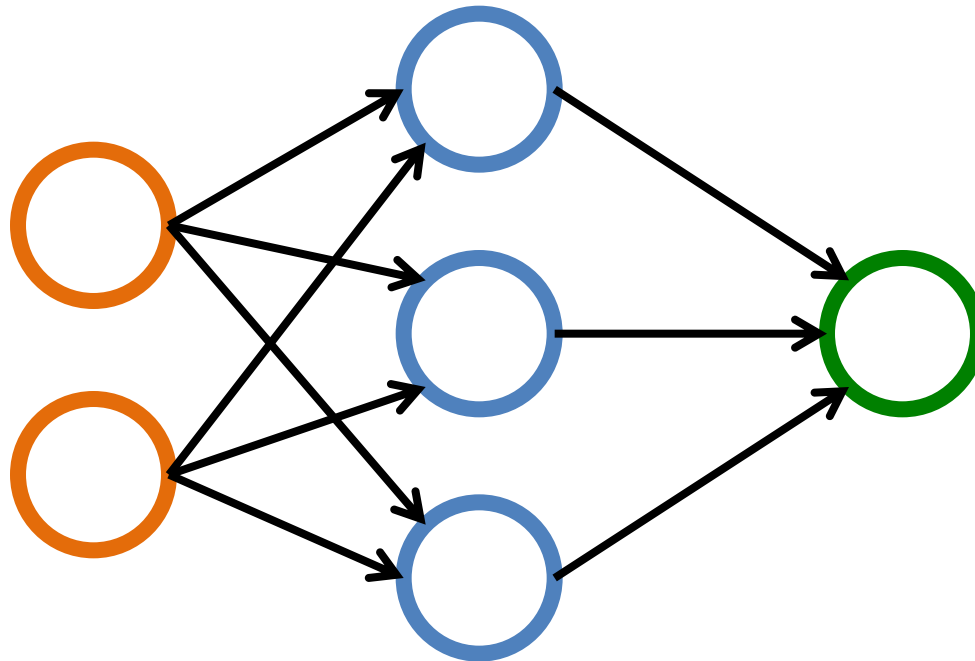
$$F(x) = \max(0, x)$$

$$y = \max (0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

Weights



Neural Networks

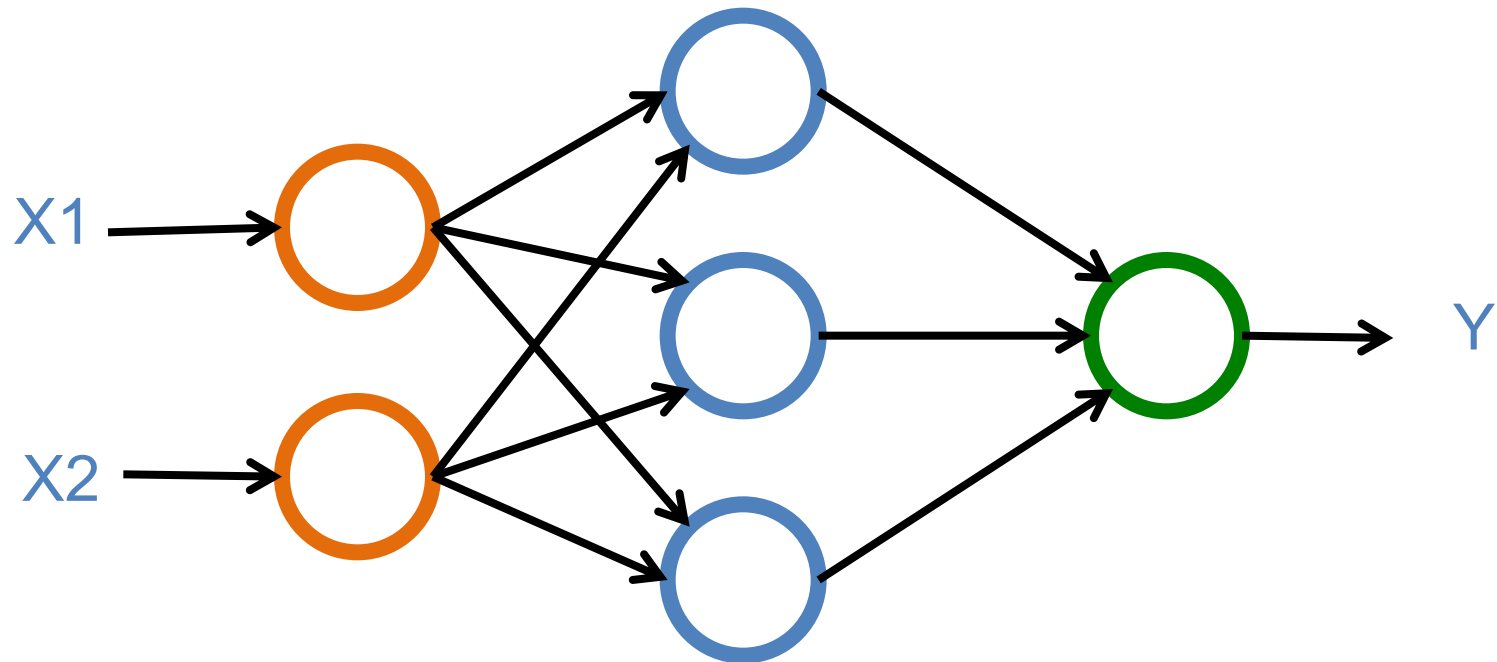


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



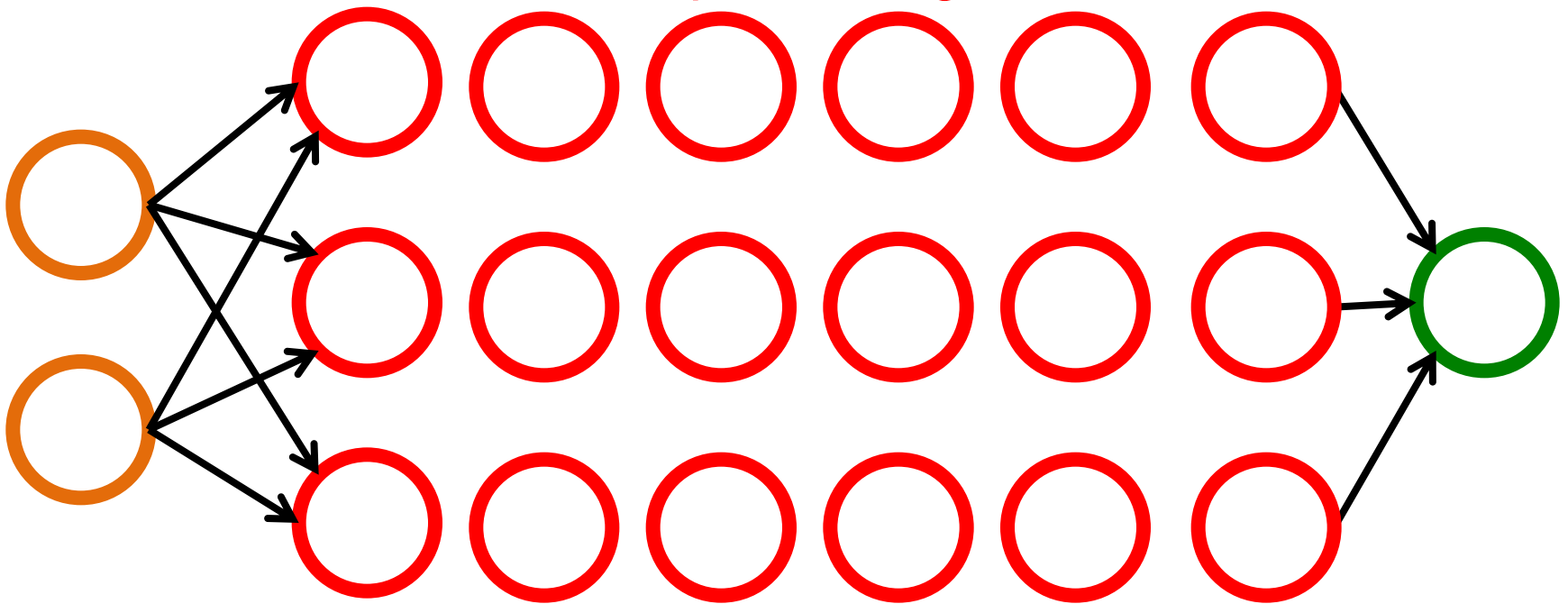
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

Output Layer
(Y)

Deep Neural Networks
Deep Learning

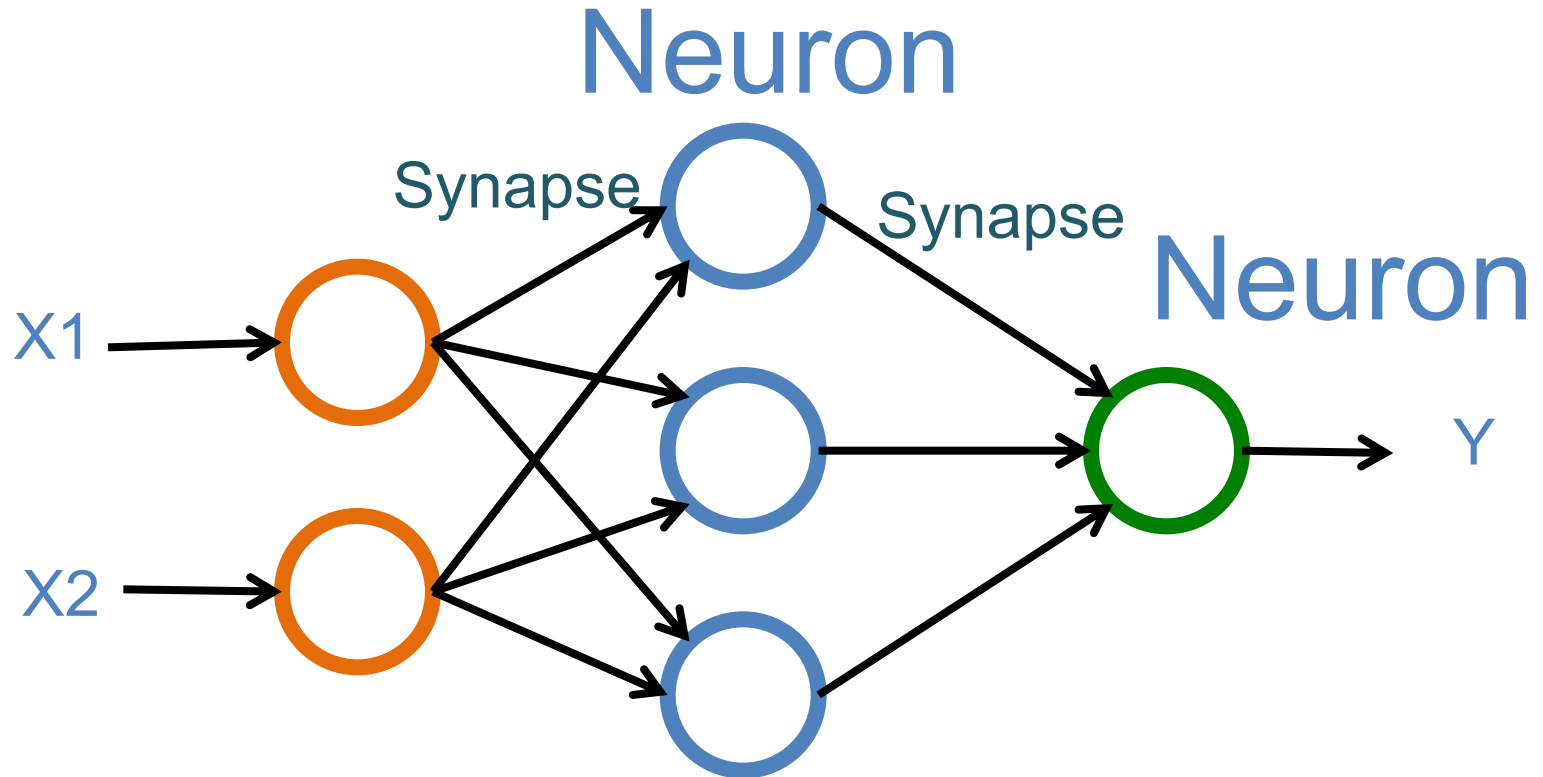


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

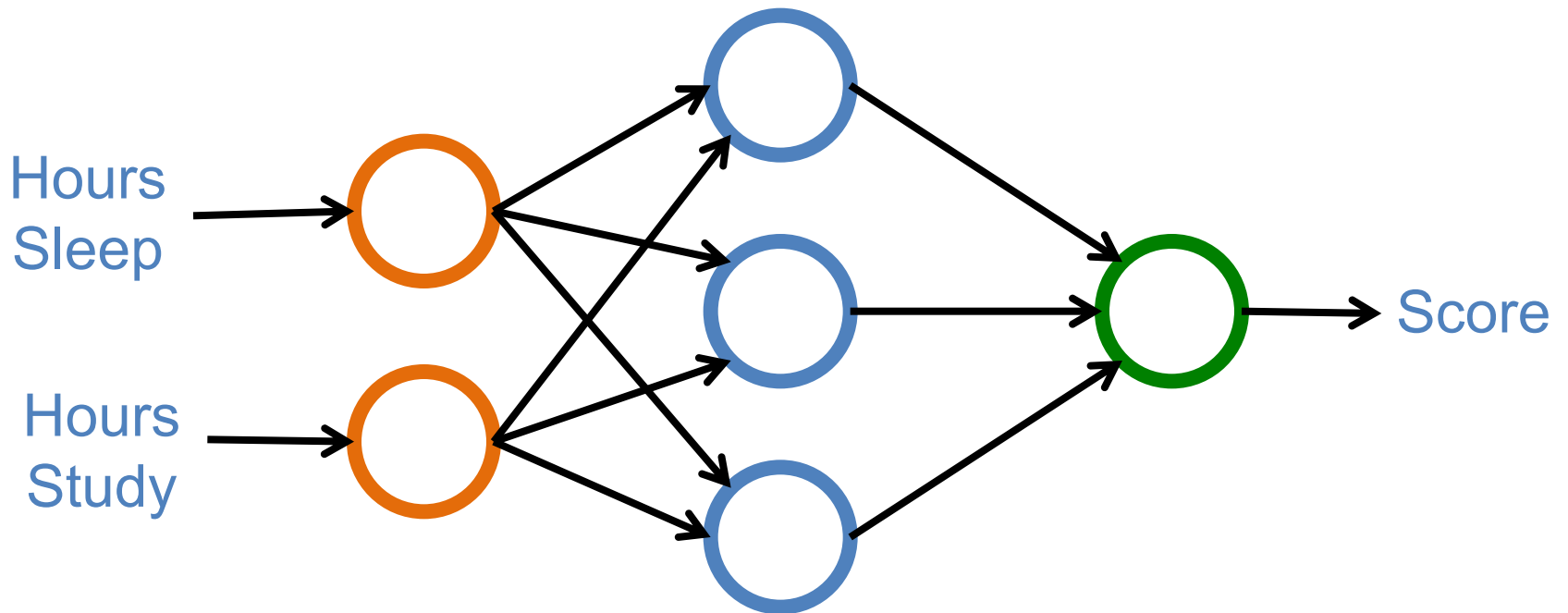


Neural Networks

**Input Layer
(X)**

**Hidden Layer
(H)**

**Output Layer
(Y)**

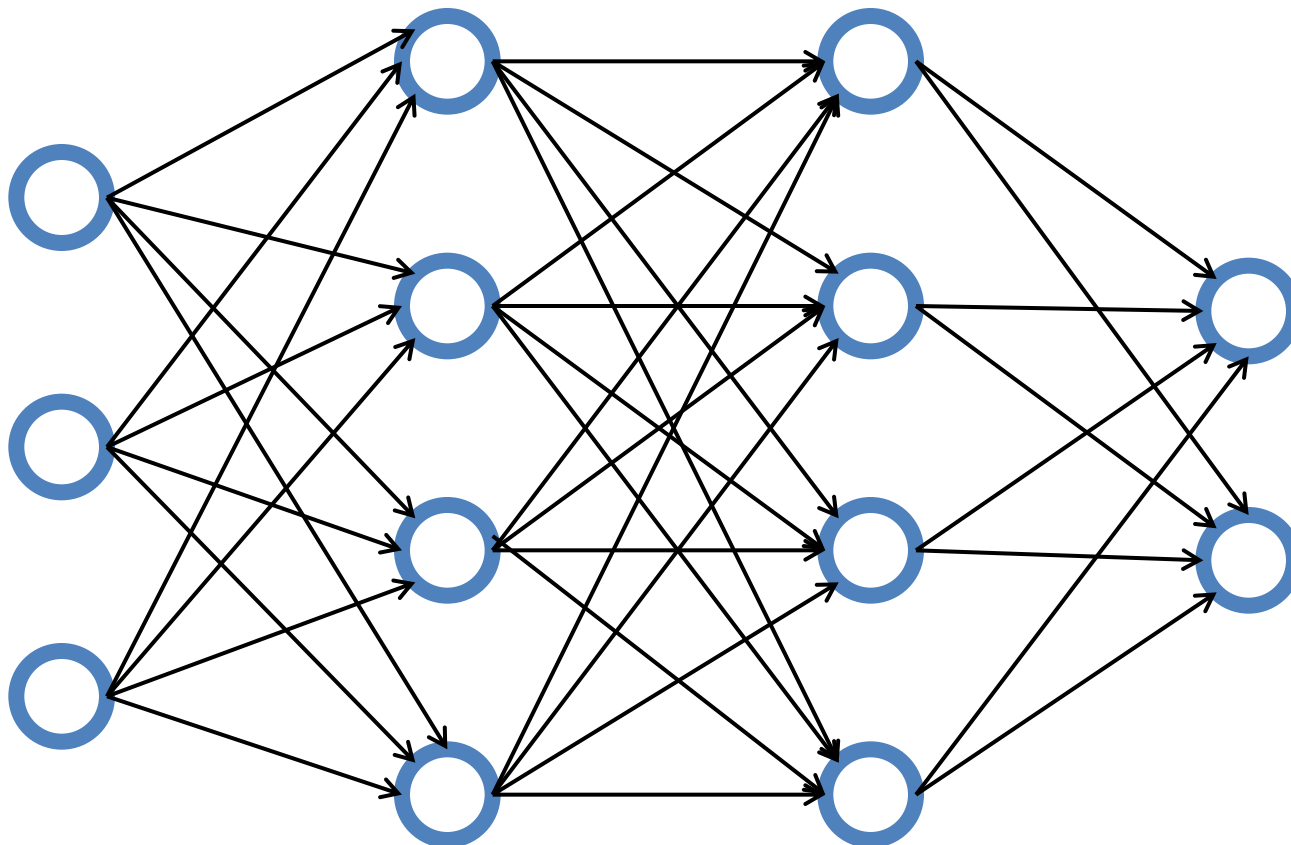


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

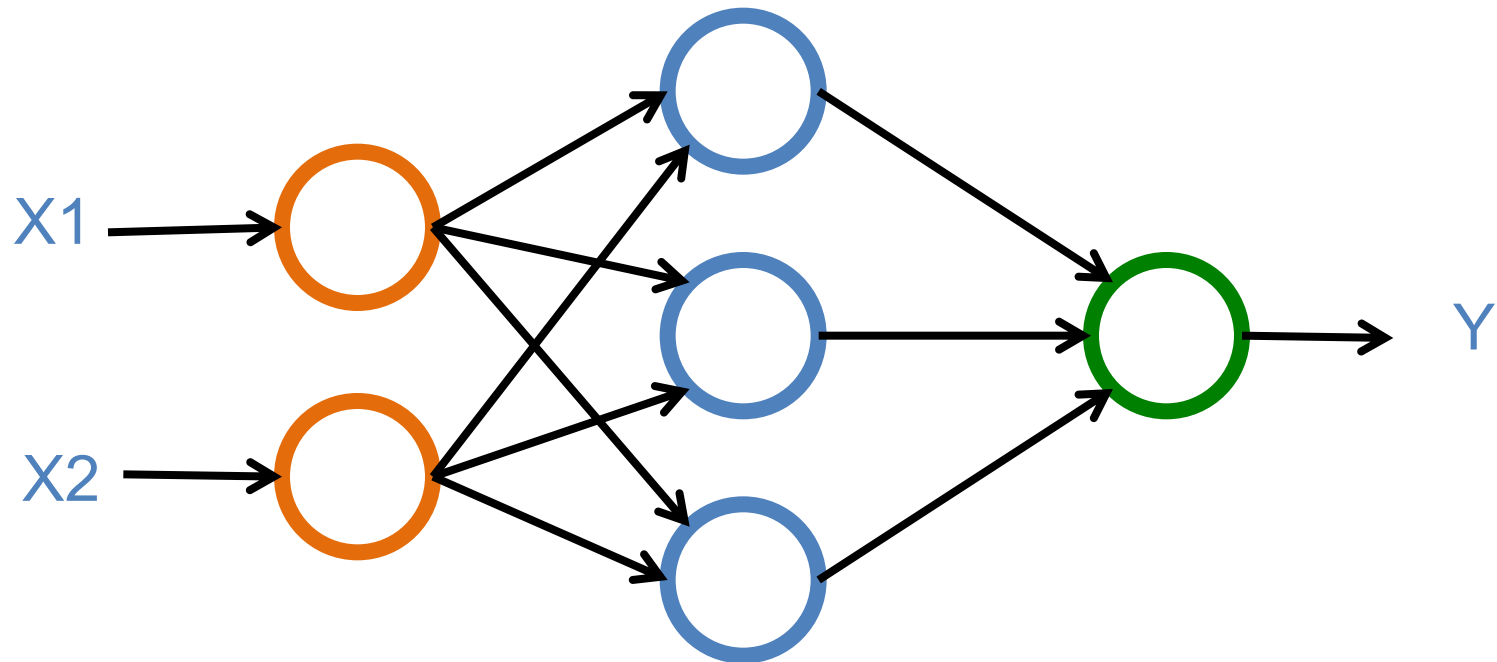


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



X		Y
Hours Sleep	Hours Study	Score
3	5	75
5	1	82
10	2	93
8	3	?

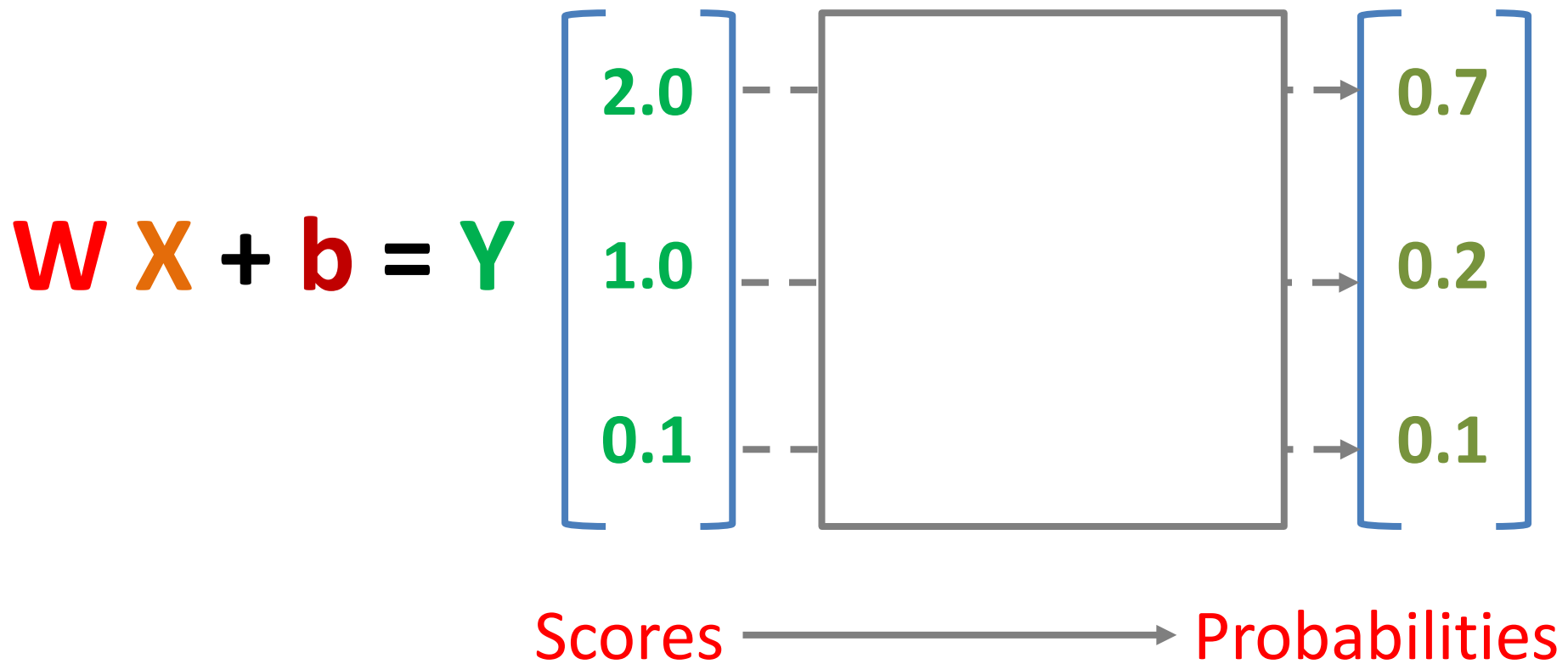
	X		Y
	Hours Sleep	Hours Study	Score
Training	3	5	75
	5	1	82
	10	2	93
Testing	8	3	?

$$Y = W X + b$$

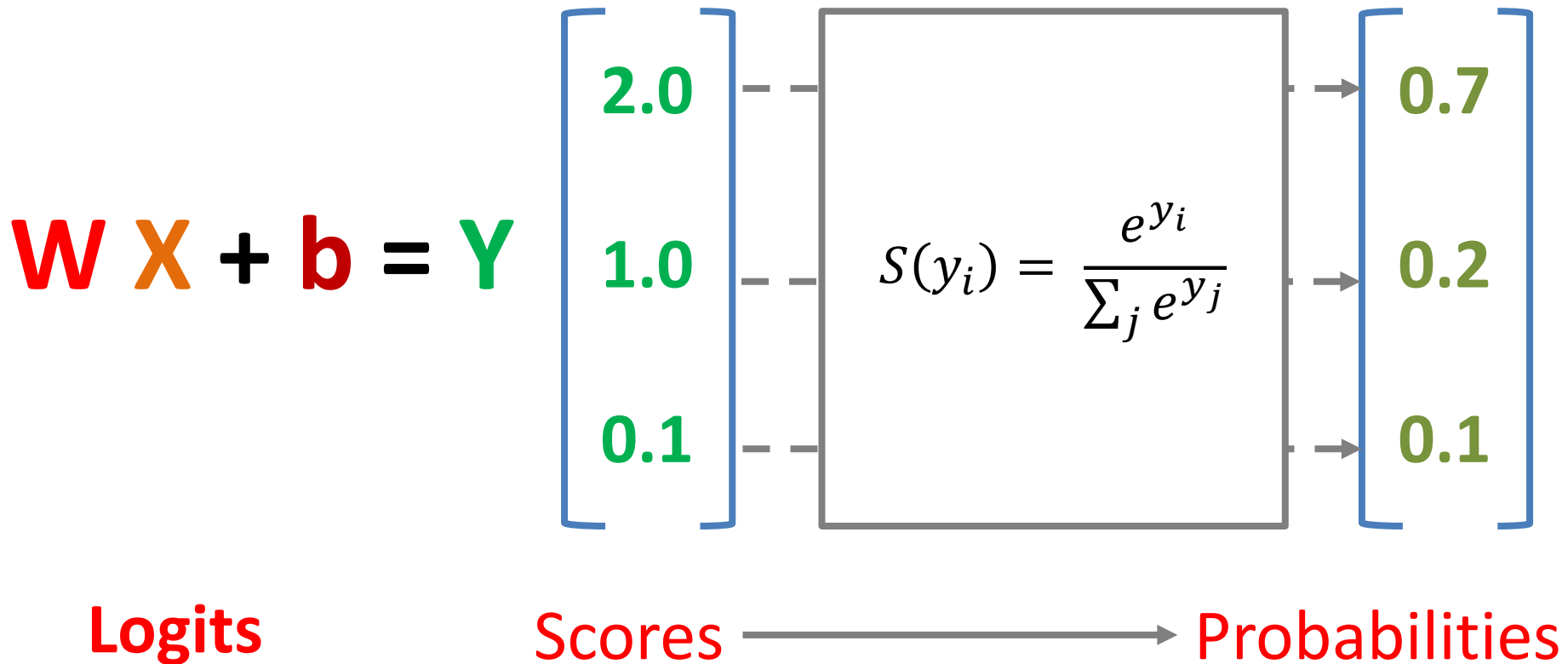
The diagram illustrates the linear equation $Y = WX + b$ with the following components and annotations:

- Output:** A green label above the variable Y , with a downward arrow pointing to it.
- input:** An orange label above the variable X , with a downward arrow pointing to it.
- Weights:** A red label below the variable W , with an upward arrow pointing to it.
- bias:** A red label below the variable b , with an upward arrow pointing to it.
- Trained:** A red label at the bottom center, with two arrows pointing upwards to the labels "Weights" and "bias".

The equation itself is $Y = WX + b$, where Y is green, W is red, X is orange, and b is red. The equals sign and plus sign are black.



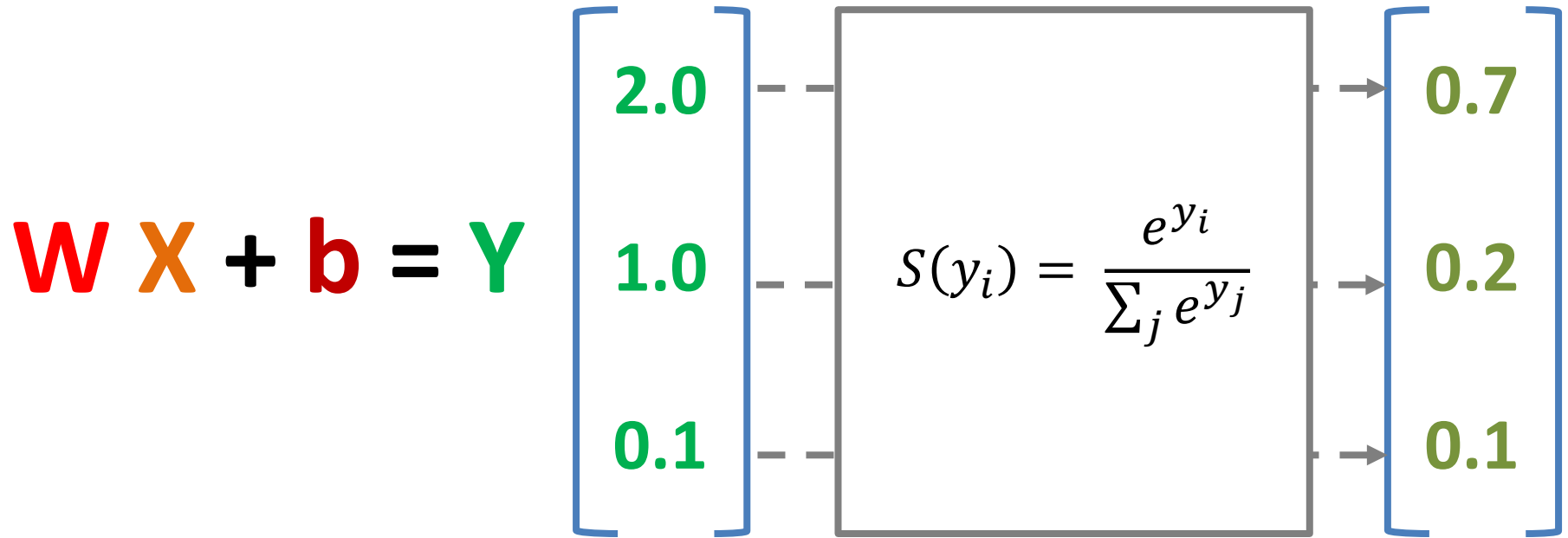
SoftMAX



$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1$$



Logits

Scores

Probabilities

Training a Network
=
Minimize the Cost Function

Training a Network

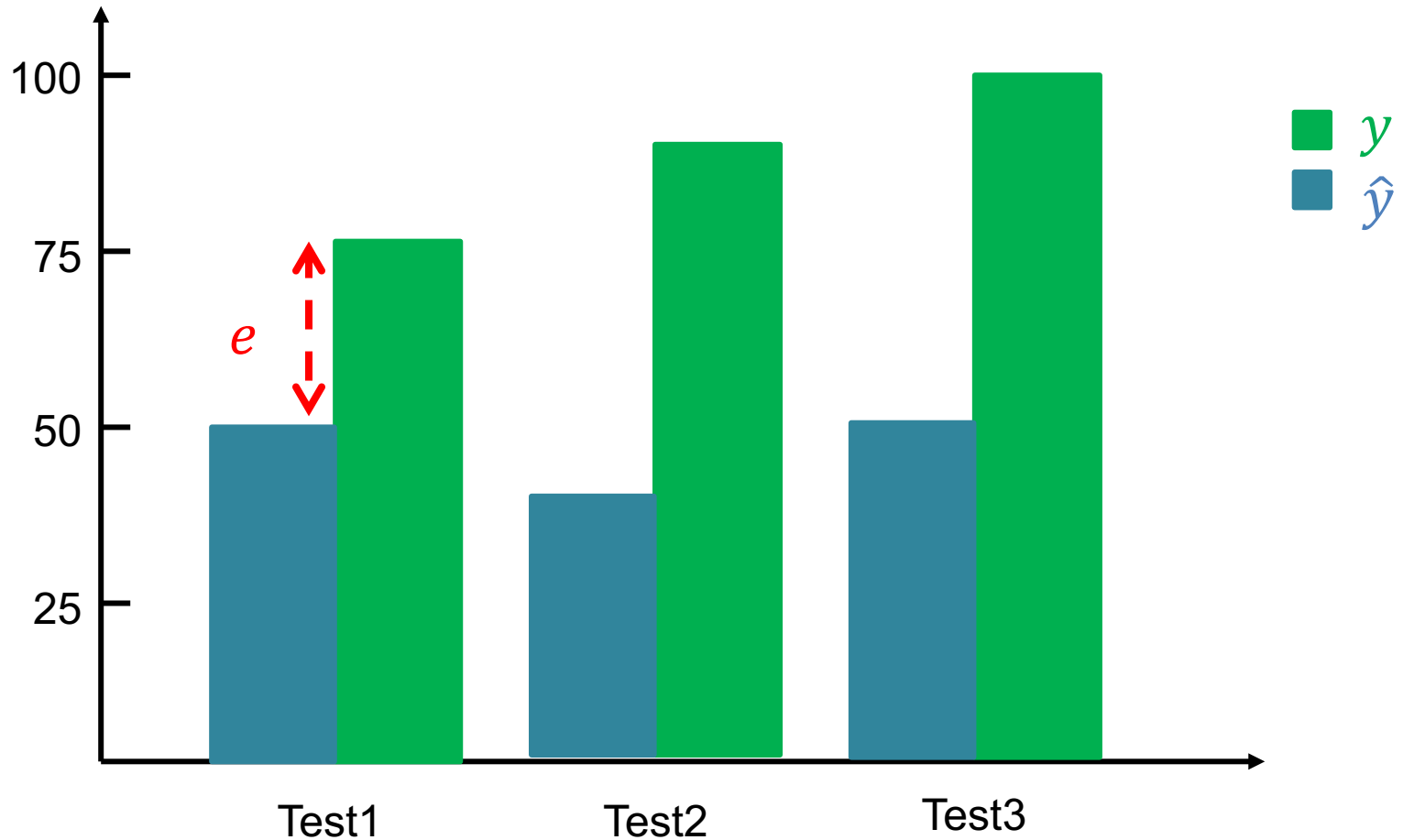
=

Minimize the **Cost** Function

Minimize the **Loss** Function

$$\text{Error} = \text{Predict } Y - \text{Actual } Y$$

Error : Cost : Loss



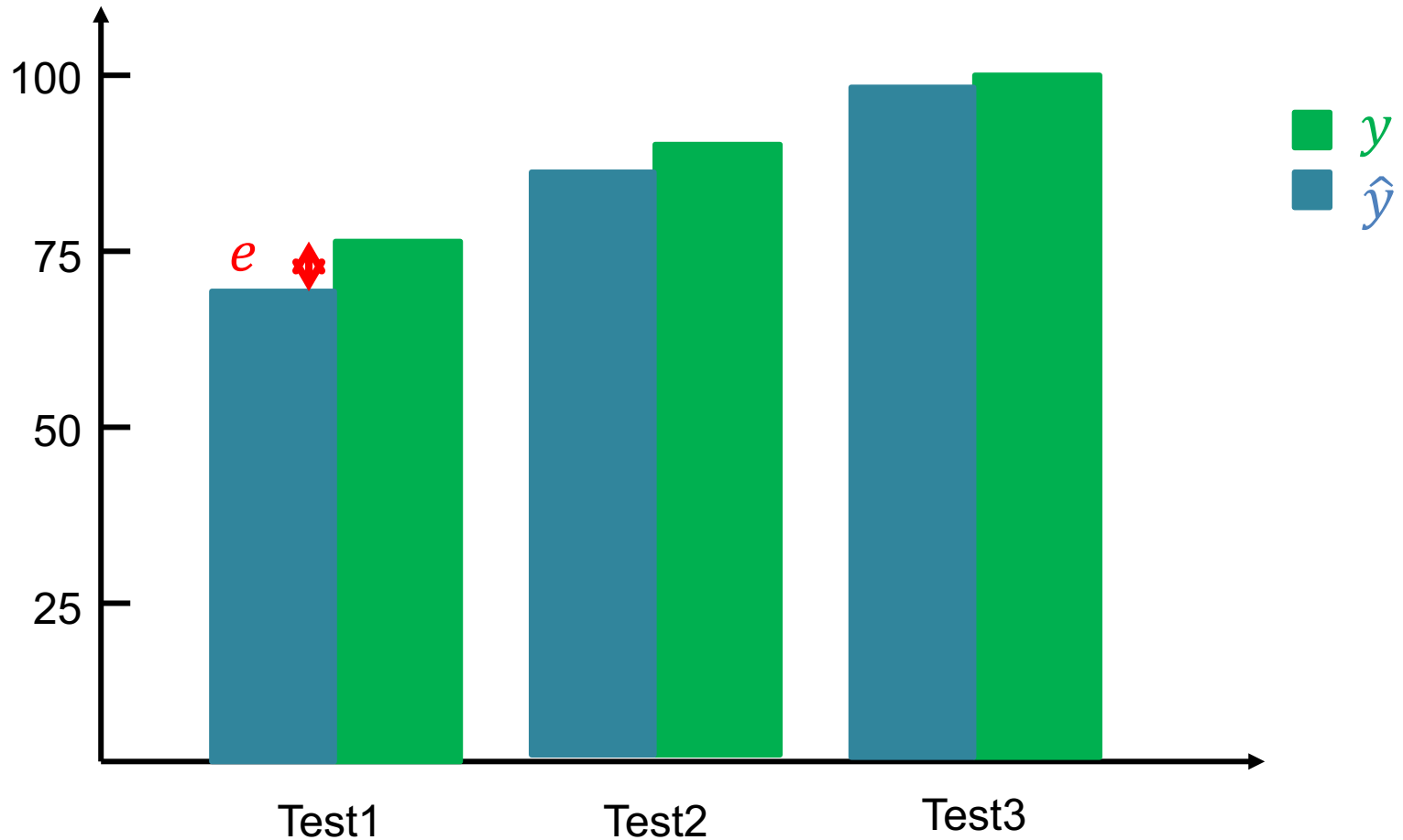
$$\text{Error} = \text{Predict } Y - \text{Actual } Y$$

Error : Cost : Loss



$$\text{Error} = \text{Predict } Y - \text{Actual } Y$$

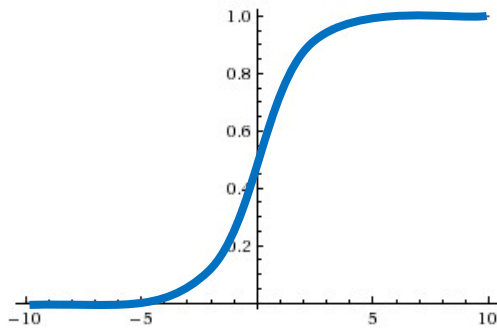
Error : Cost : Loss



Activation Functions

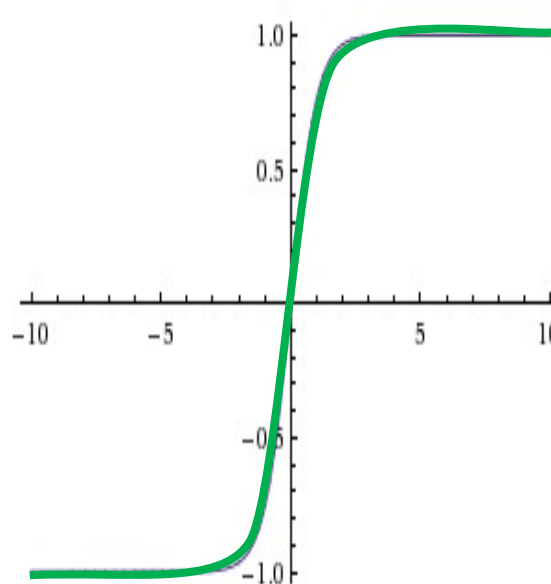
Activation Functions

Sigmoid



[0, 1]

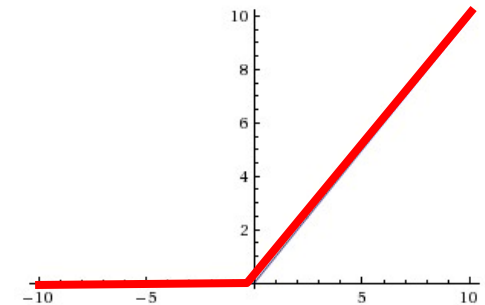
TanH



[-1, 1]

ReLU

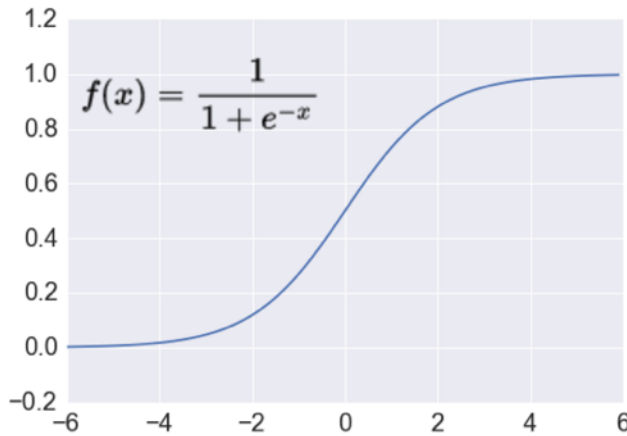
(Rectified Linear Unit)



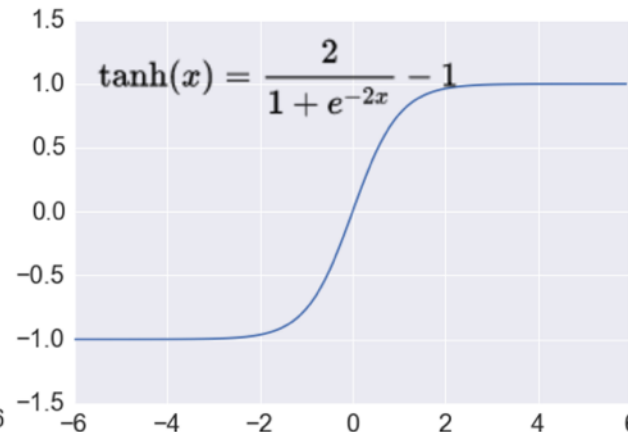
$f(x) = \max(0, x)$

Activation Functions

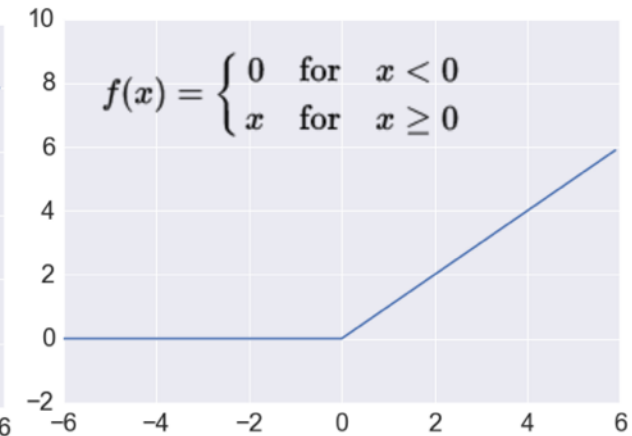
Sigmoid



TanH



ReLU



Loss Function

Binary Classification: 2 Class

**Activation Function:
Sigmoid**

**Loss Function:
Binary Cross-Entropy**

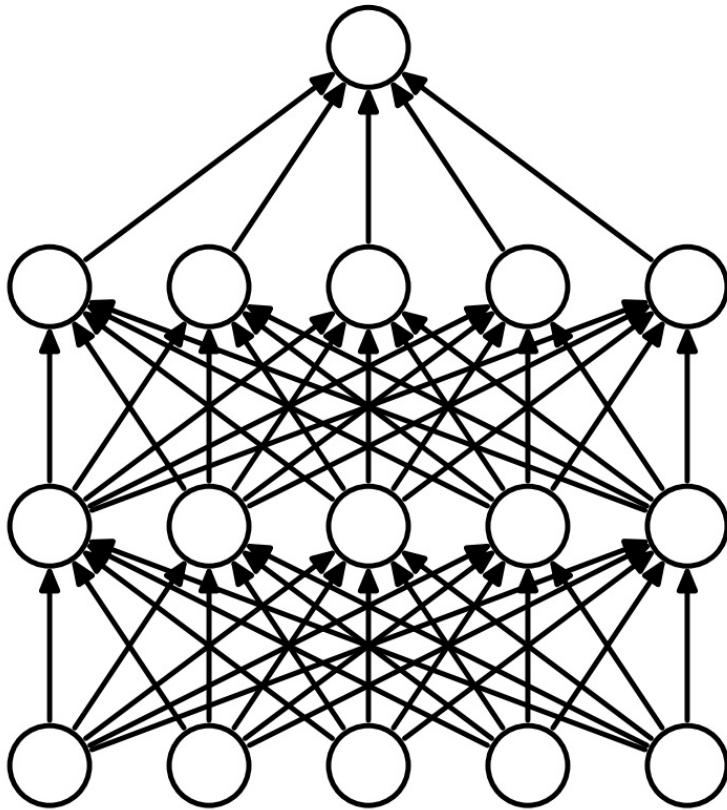
Multiple Classification: 10 Class

**Activation Function:
SoftMAX**

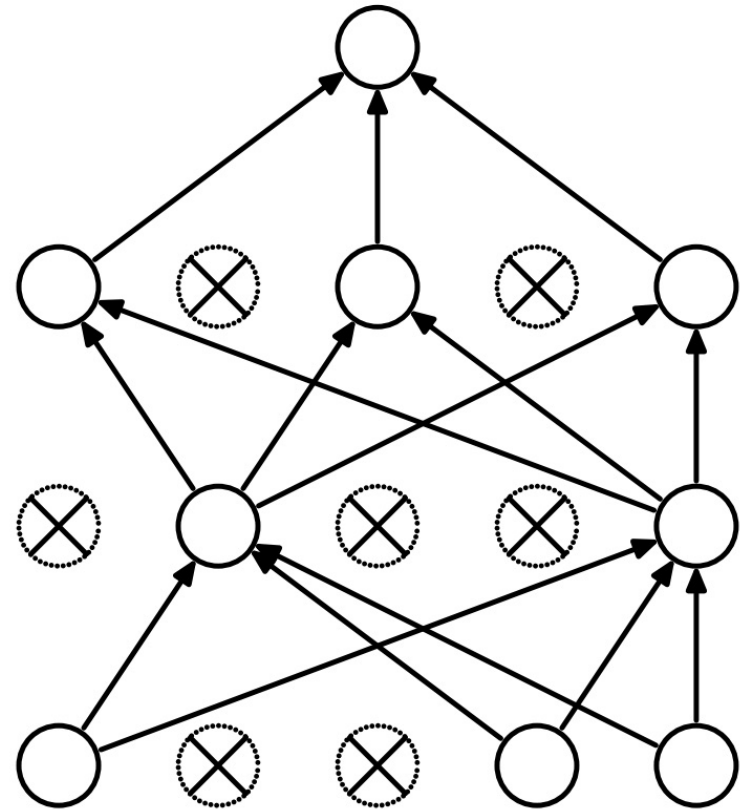
**Loss Function:
Categorical Cross-Entropy**

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net



(b) After applying dropout.

Source: Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.

"Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15, no. 1 (2014): 1929-1958.

Learning Algorithm

While not done:

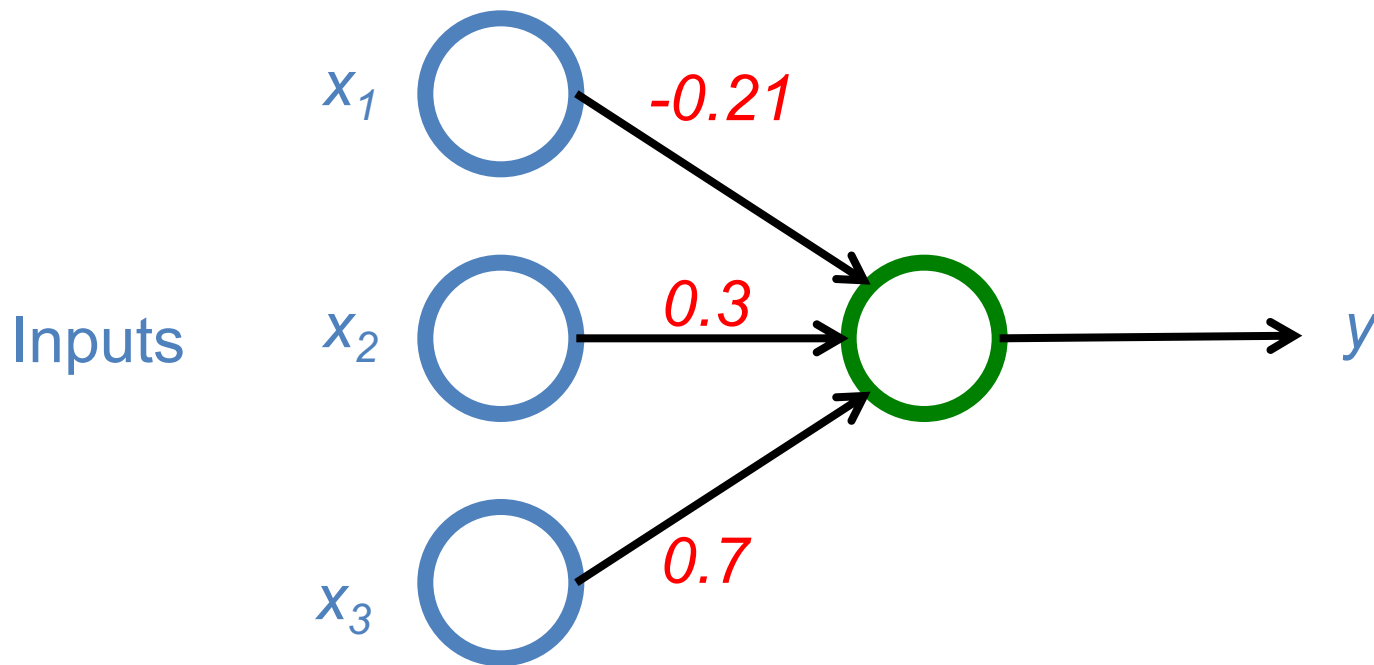
- Pick a random training example “(input, label)”

- Run neural network on “input”

- Adjust weights on edges to make output closer to “label”

$$y = \max (0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

Weights

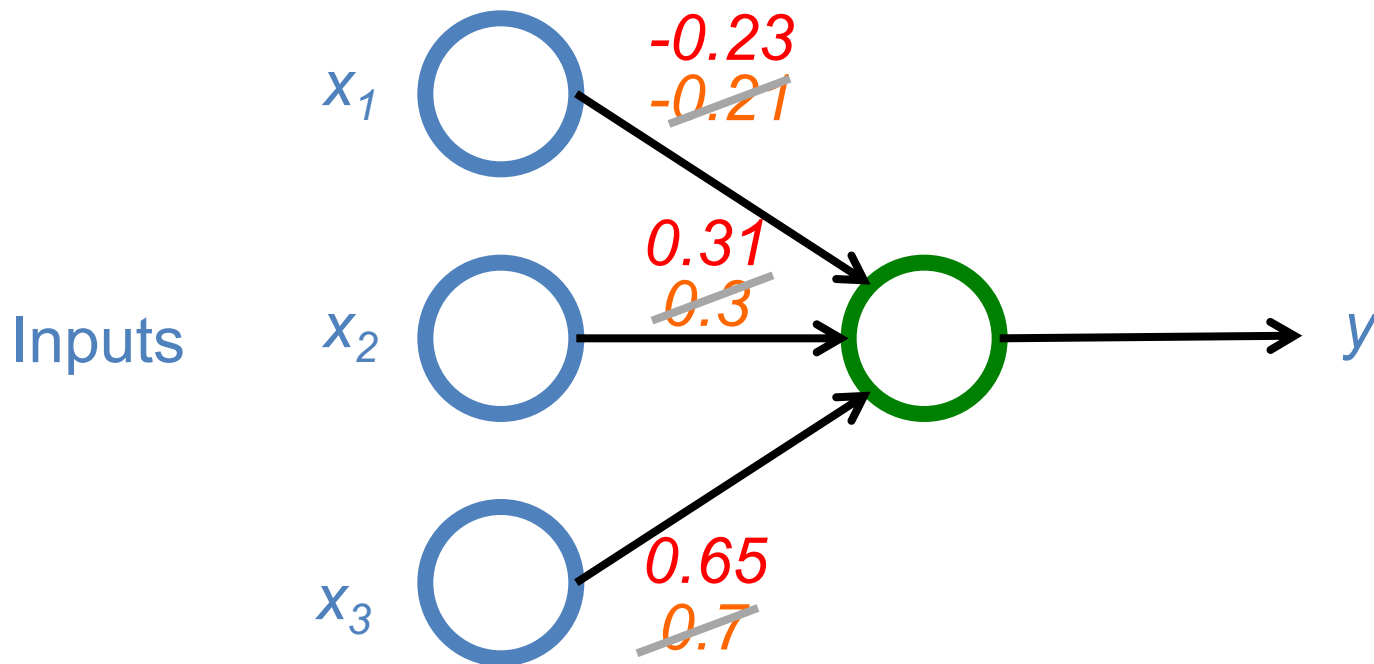


Next time:

$$y = \max (0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3)$$

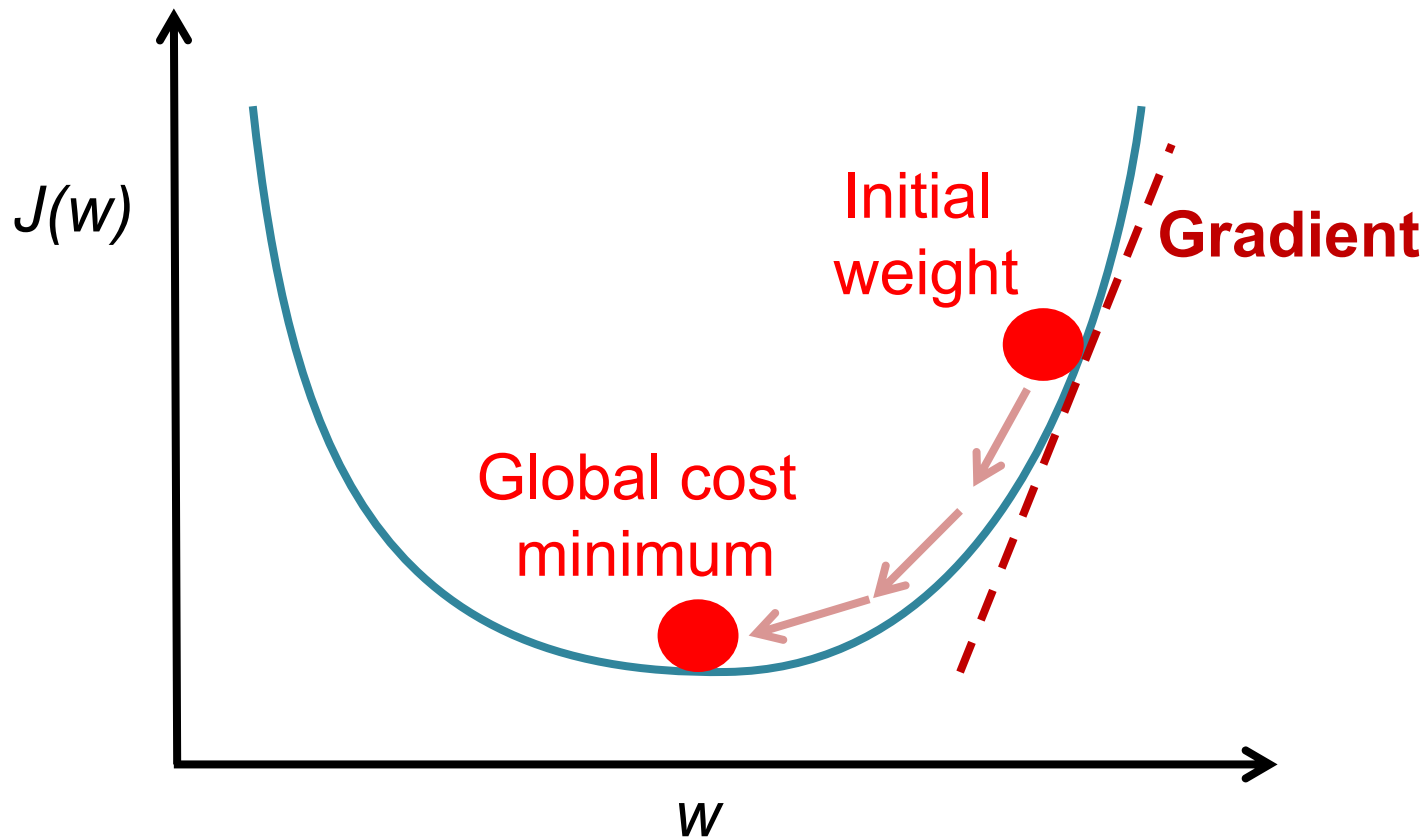
~~$$y = \max (0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$~~

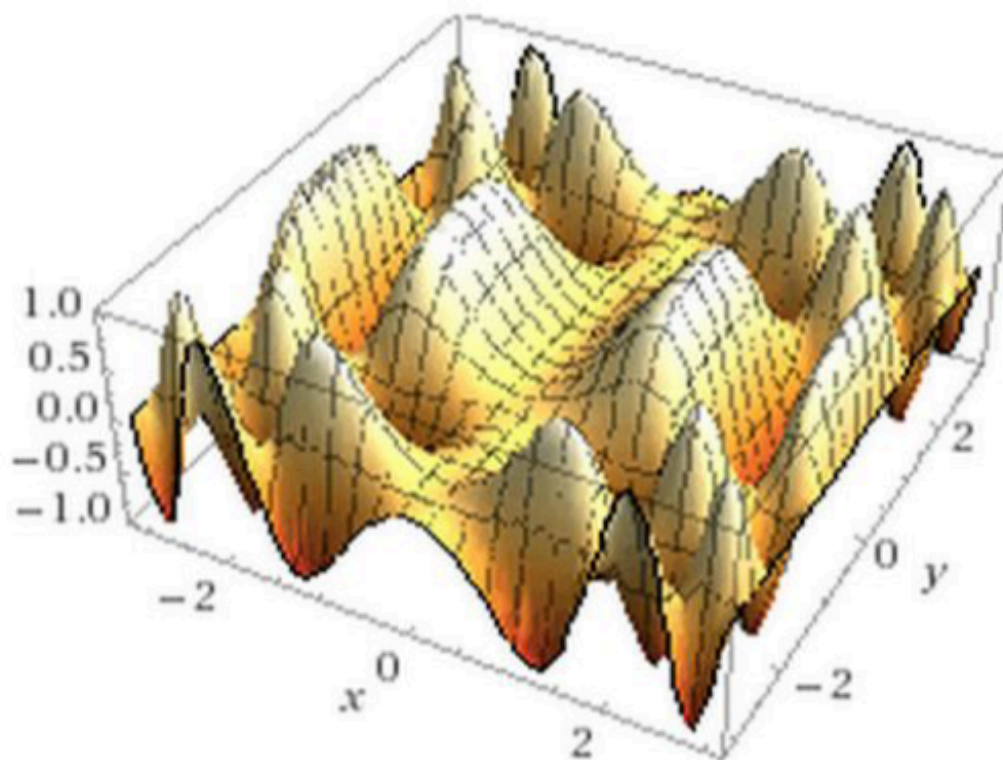
Weights



Optimizer:

Stochastic Gradient Descent (SGD)





This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

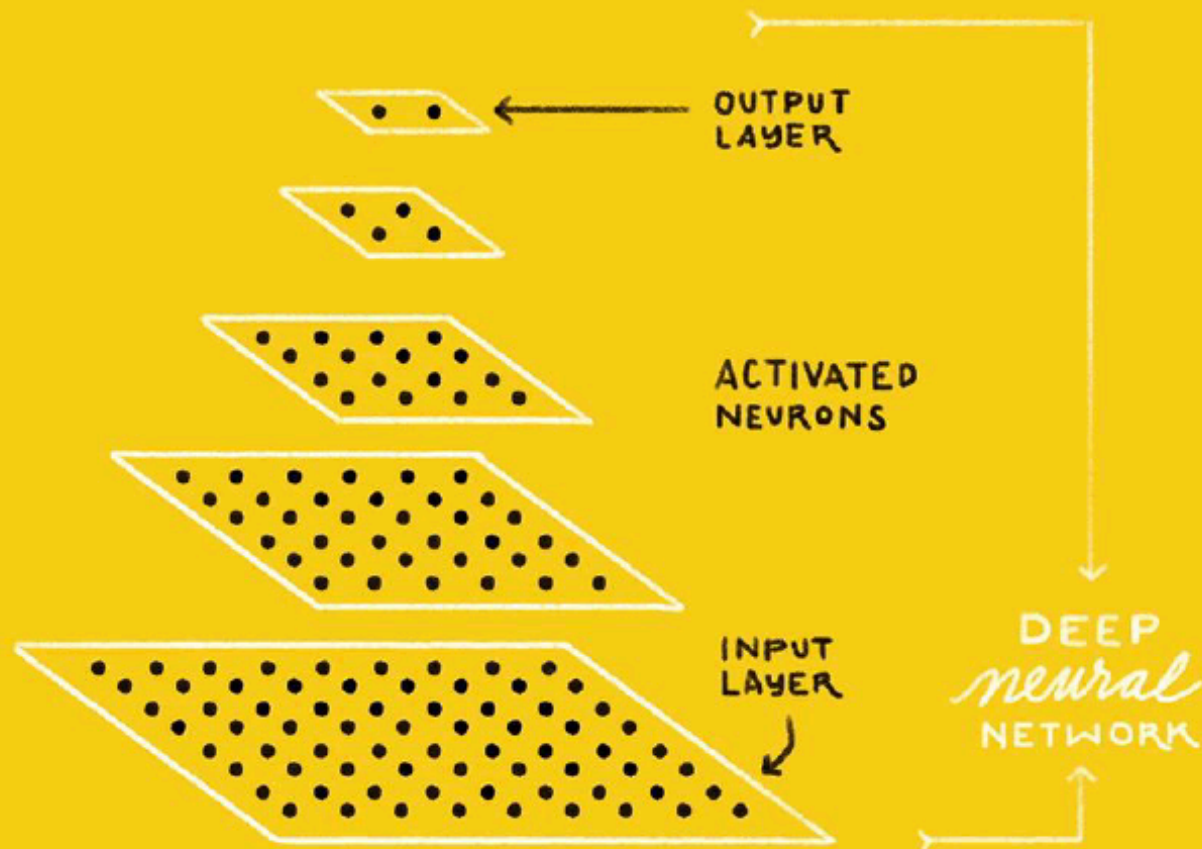
Visual Recognition

Image Classification

IS THIS A
CAT or DOG?

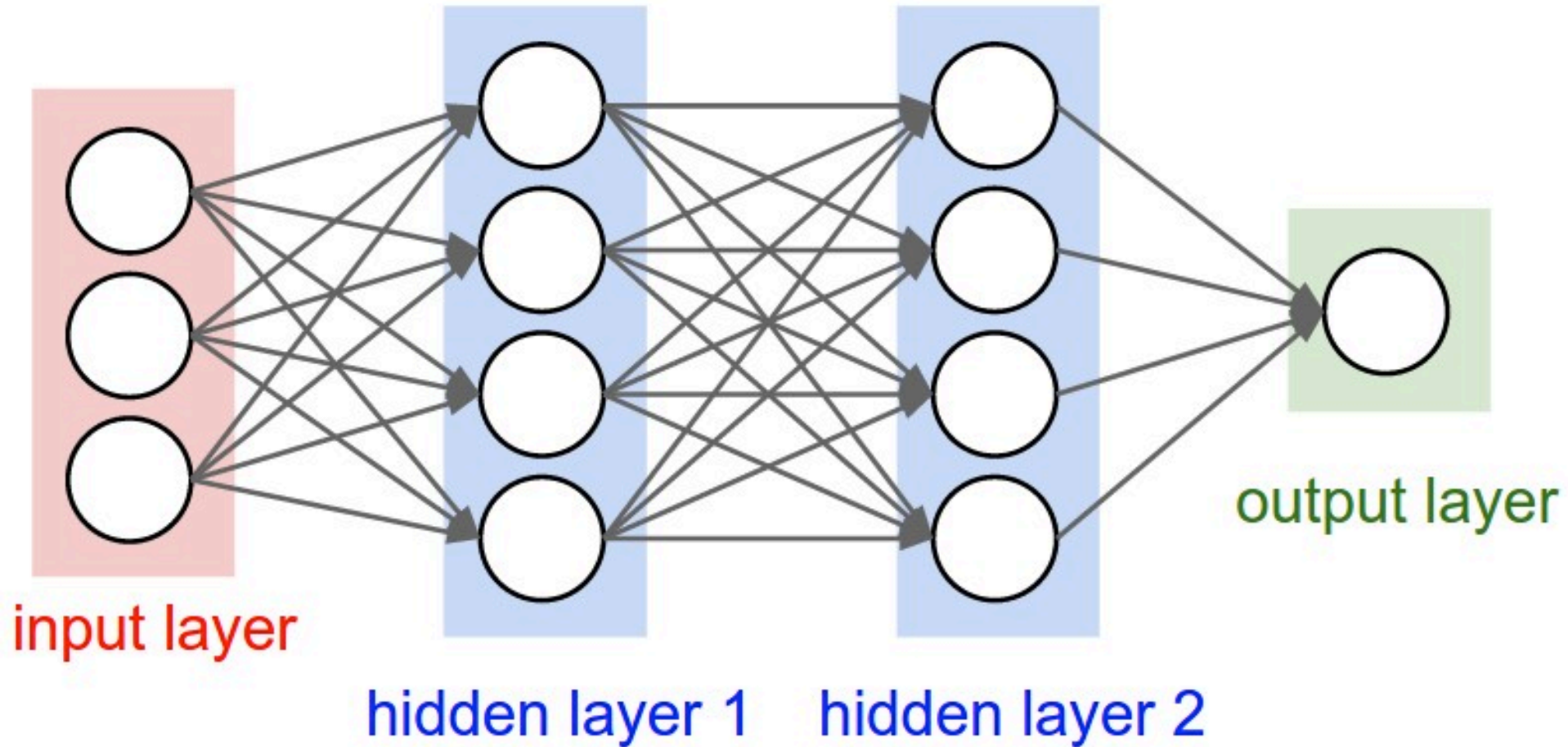


CAT DOG

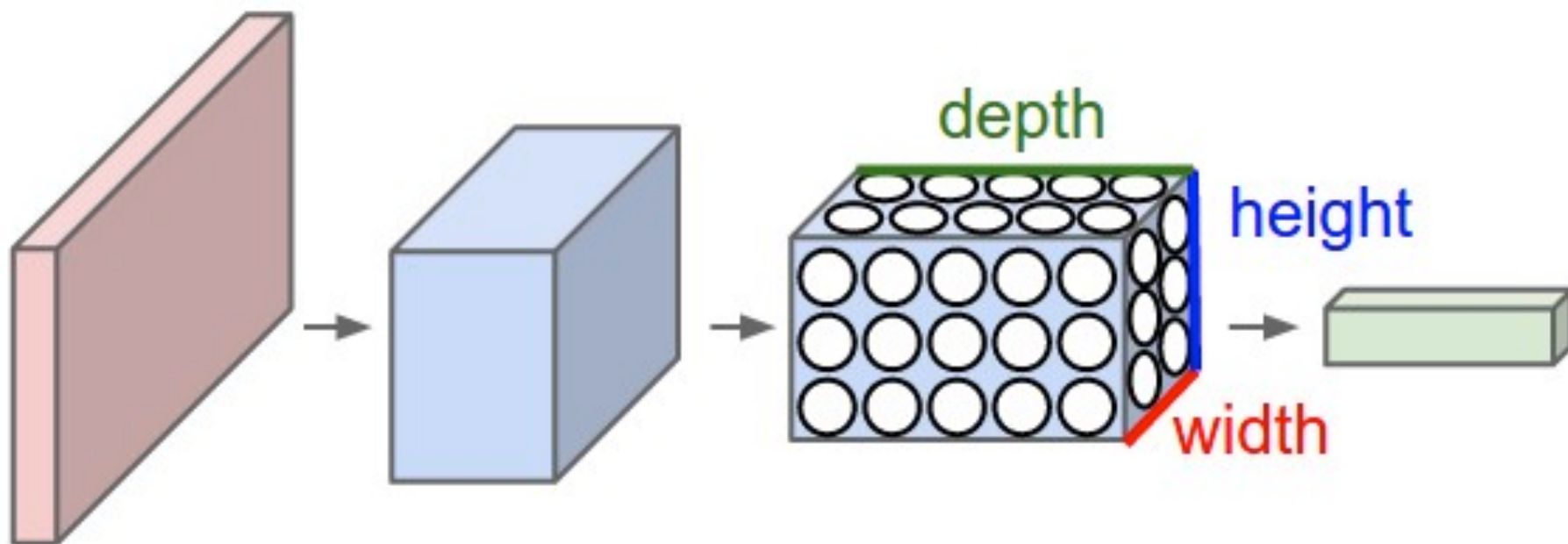


Convolutional Neural Networks (CNNs / ConvNets)

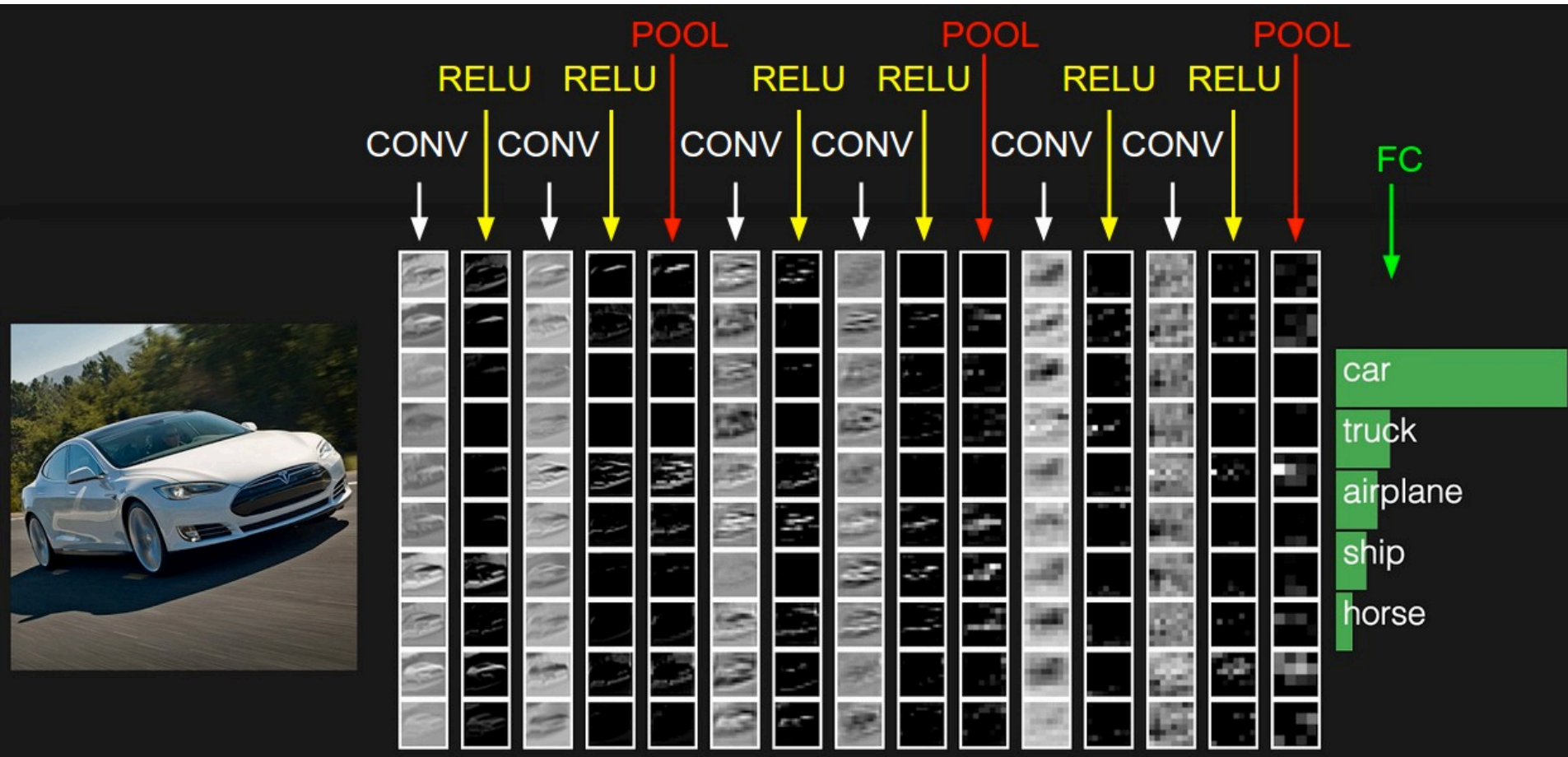
A regular 3-layer Neural Network



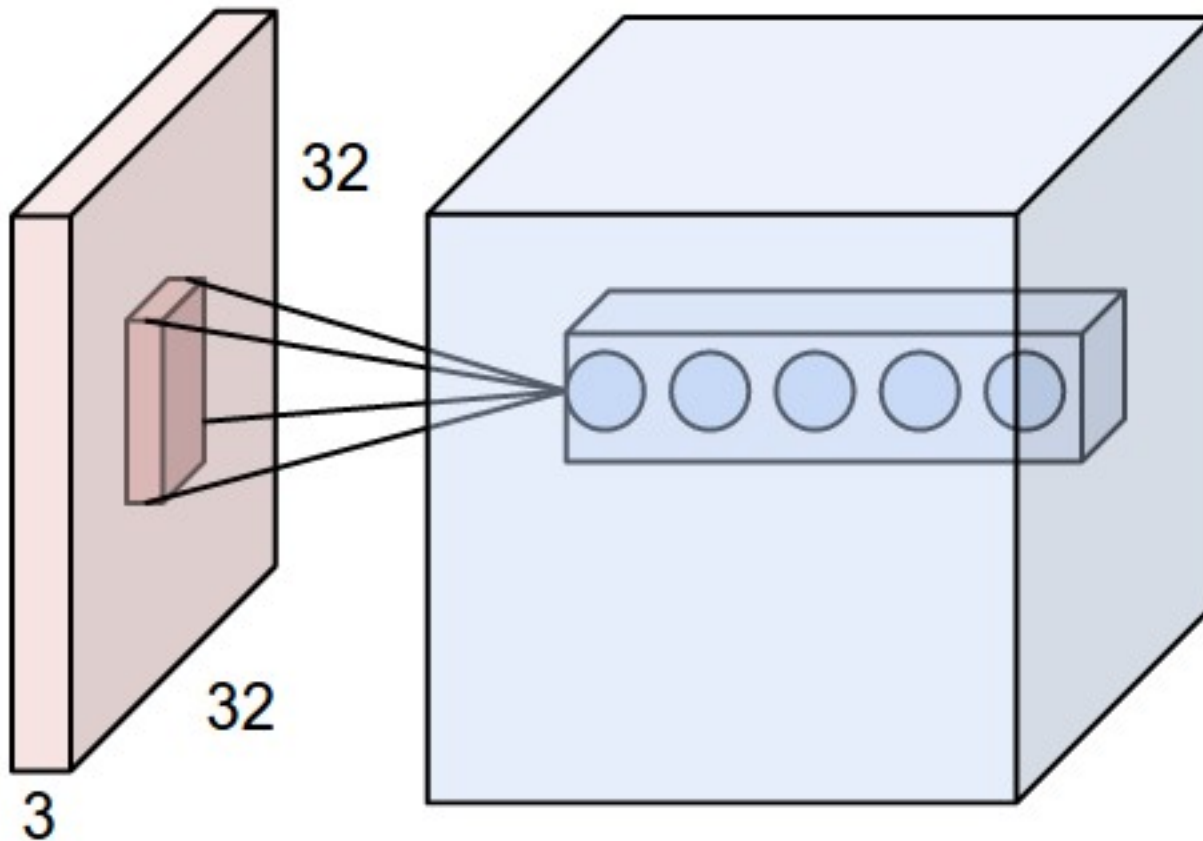
A ConvNet arranges its neurons in three dimensions (width, height, depth)



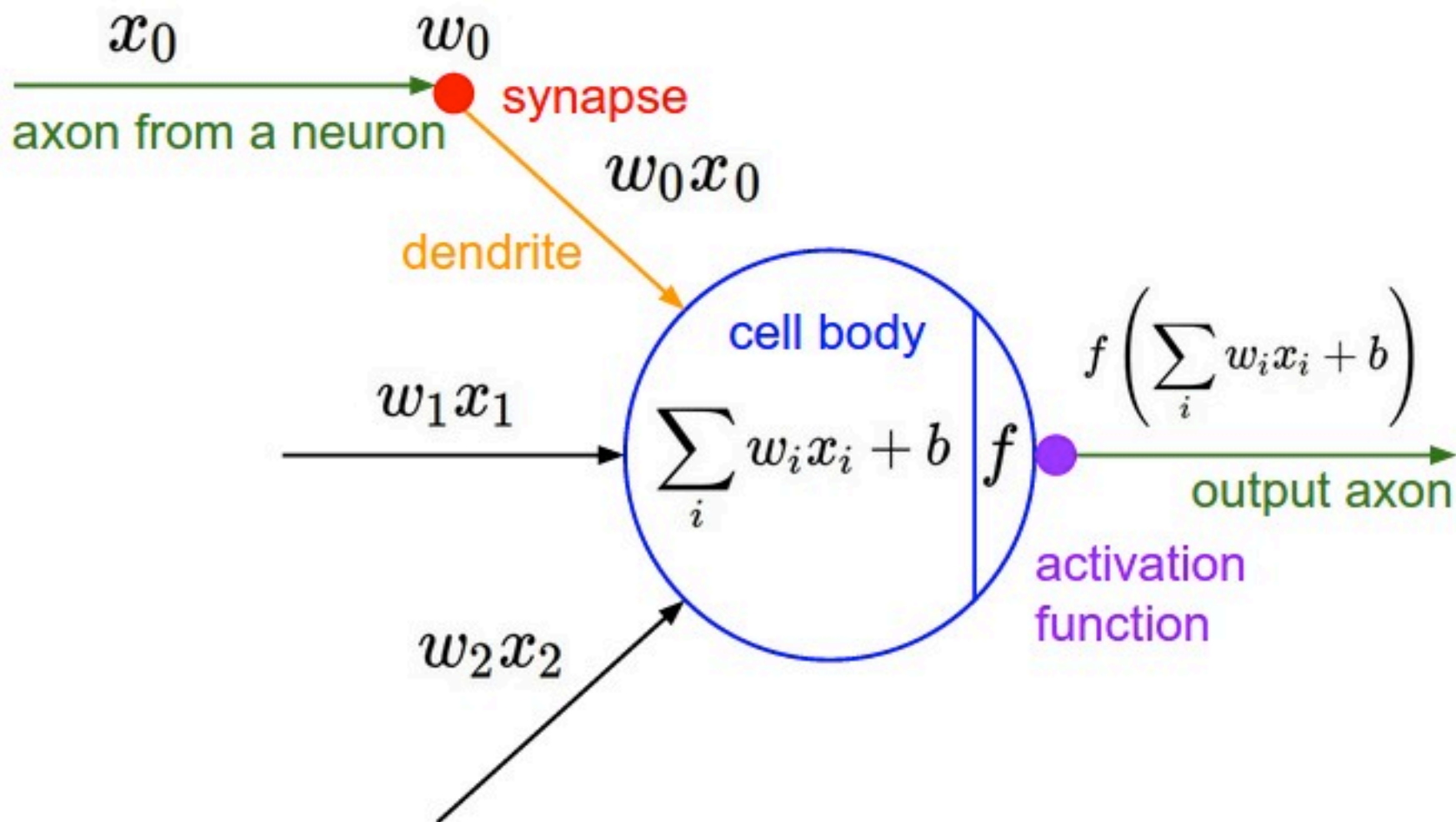
The activations of an example ConvNet architecture.



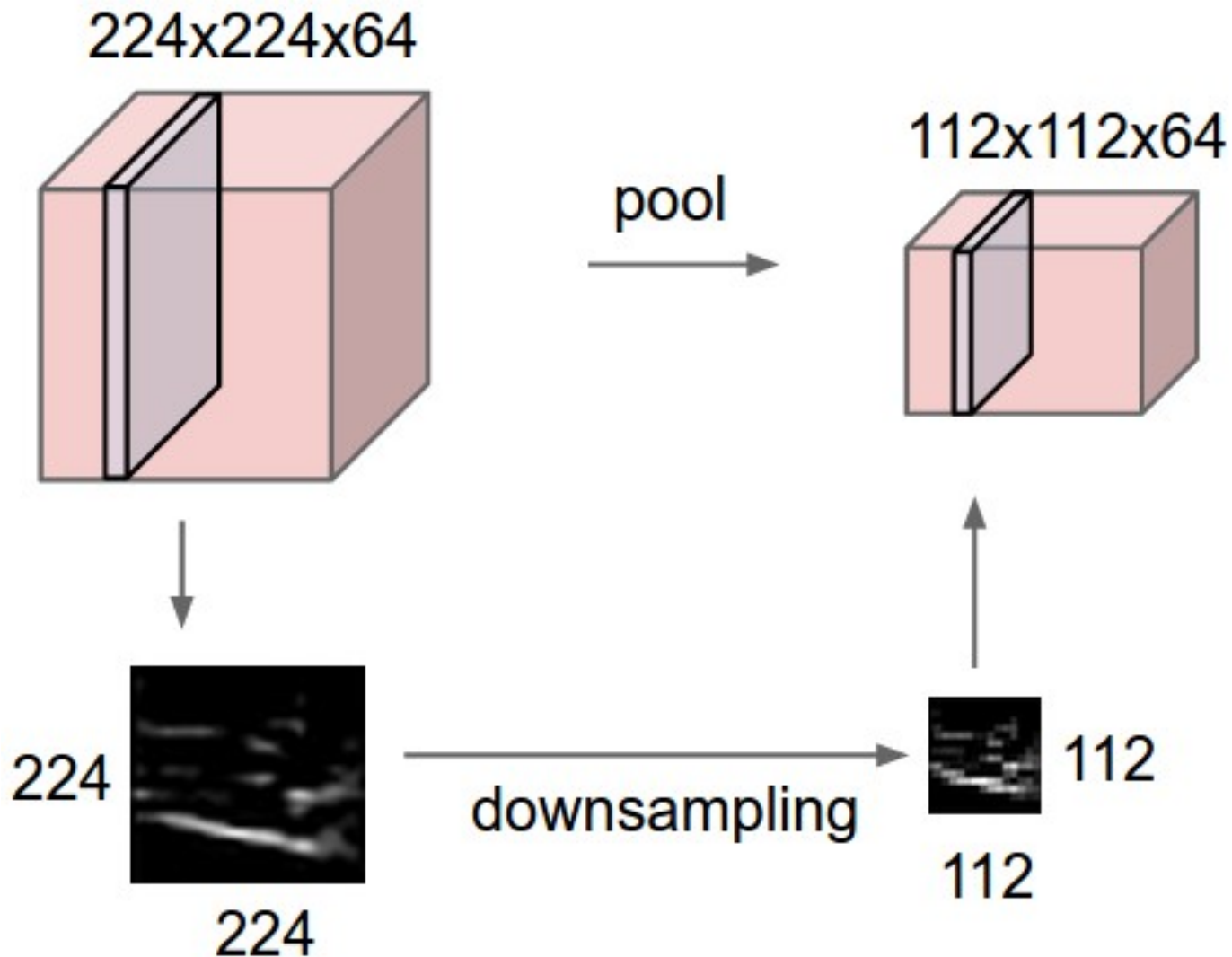
ConvNets



ConvNets



ConvNets



ConvNets

max pooling

Single depth slice

x

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

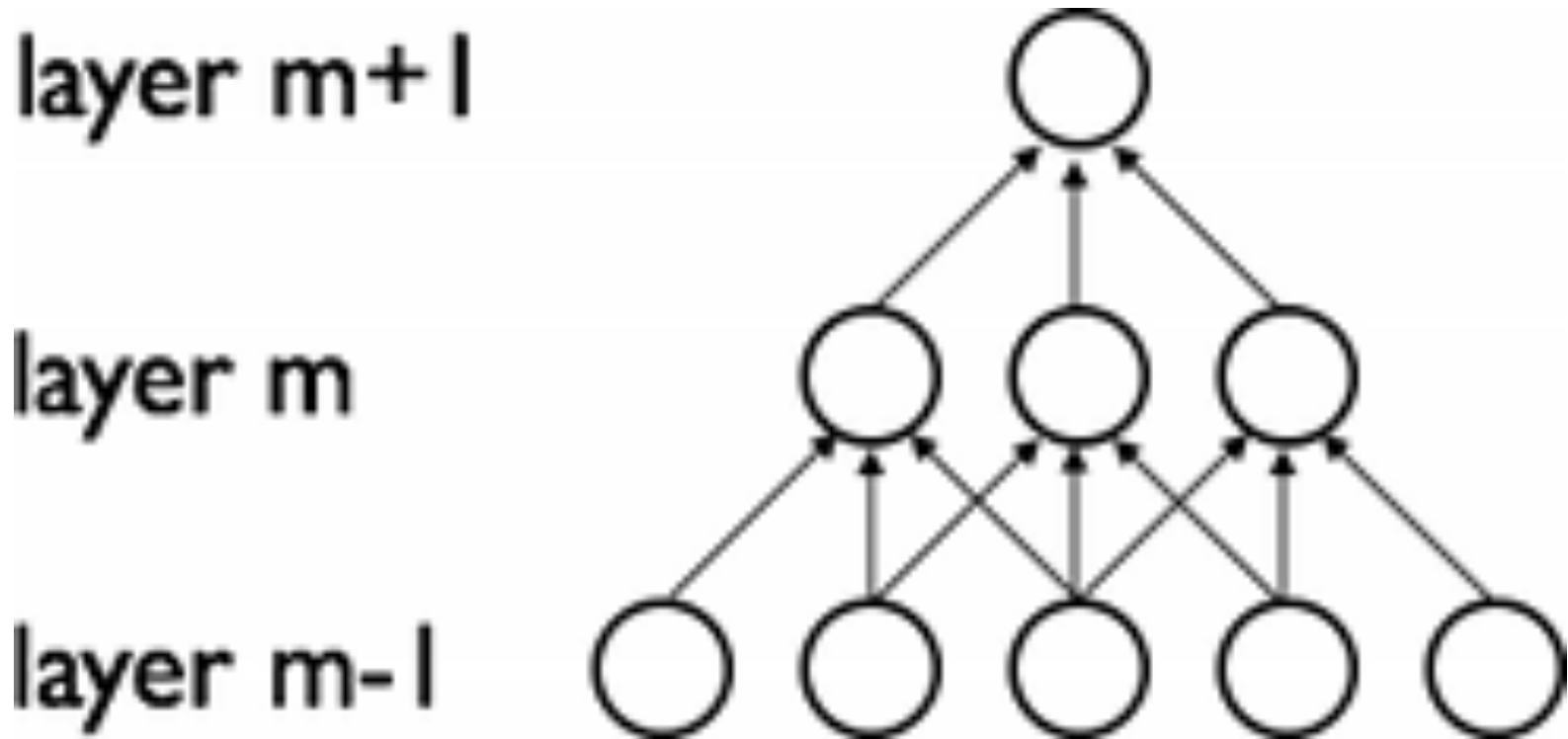
y

max pool with 2x2 filters
and stride 2



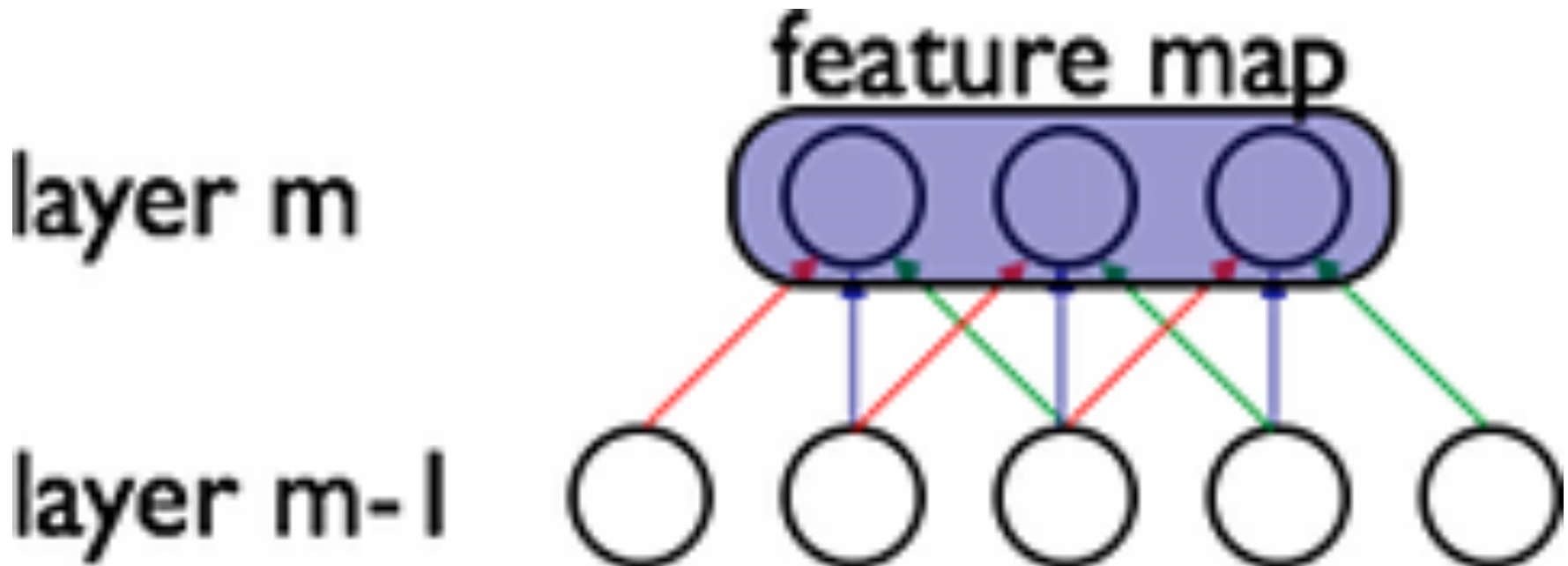
6	8
3	4

Convolutional Neural Networks (CNN) (LeNet) Sparse Connectivity



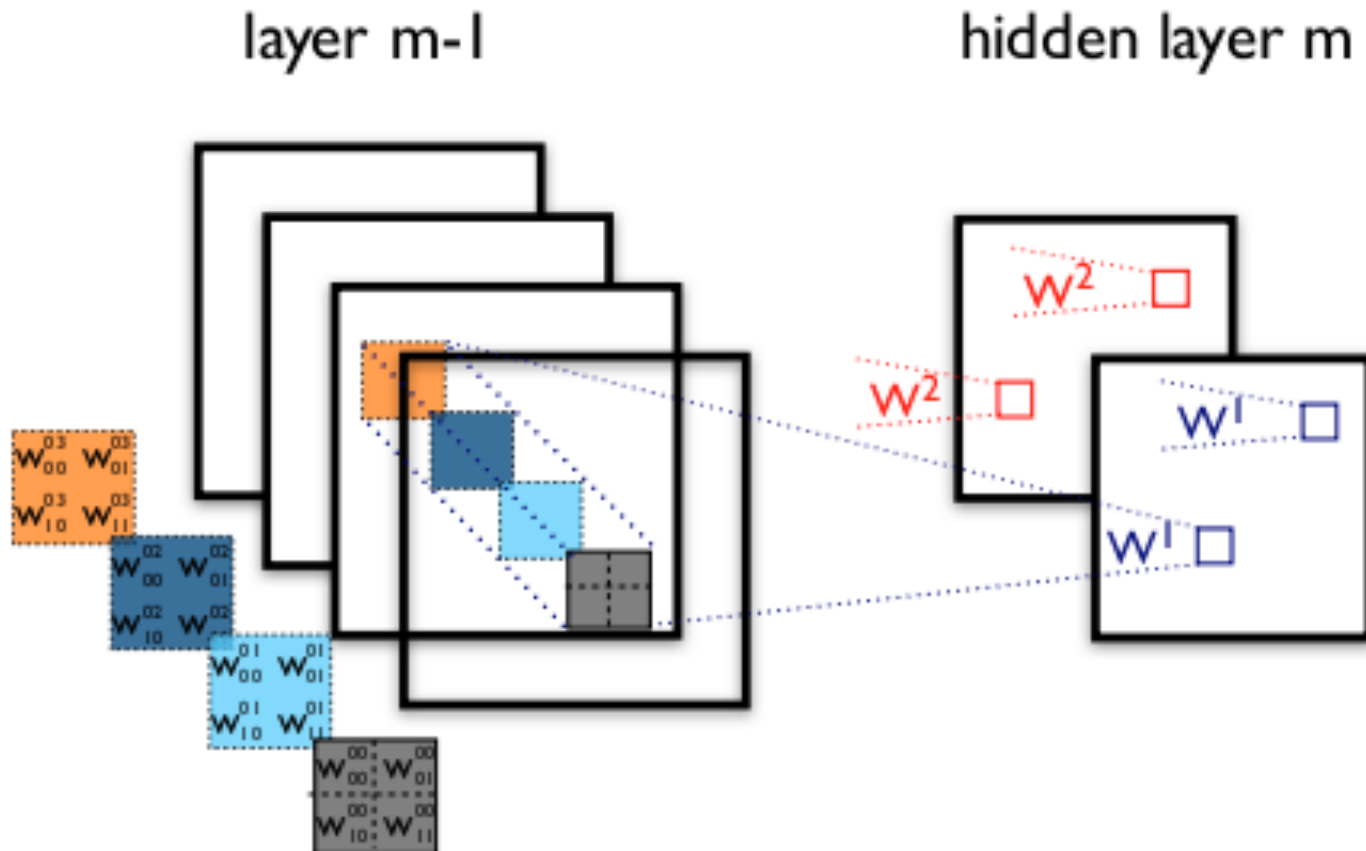
Convolutional Neural Networks (CNN) (LeNet)

Shared Weights

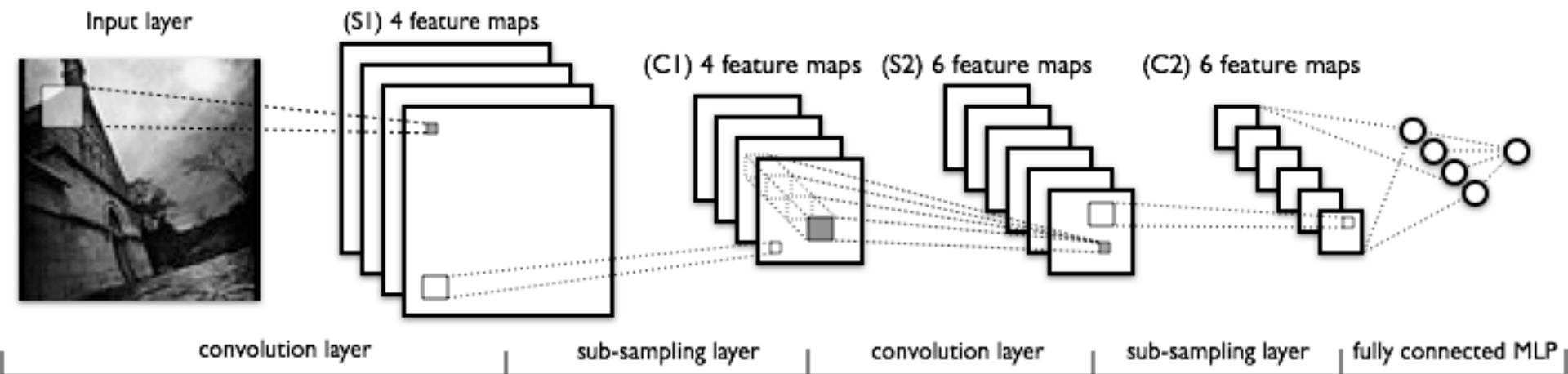


Convolutional Neural Networks (CNN) (LeNet)

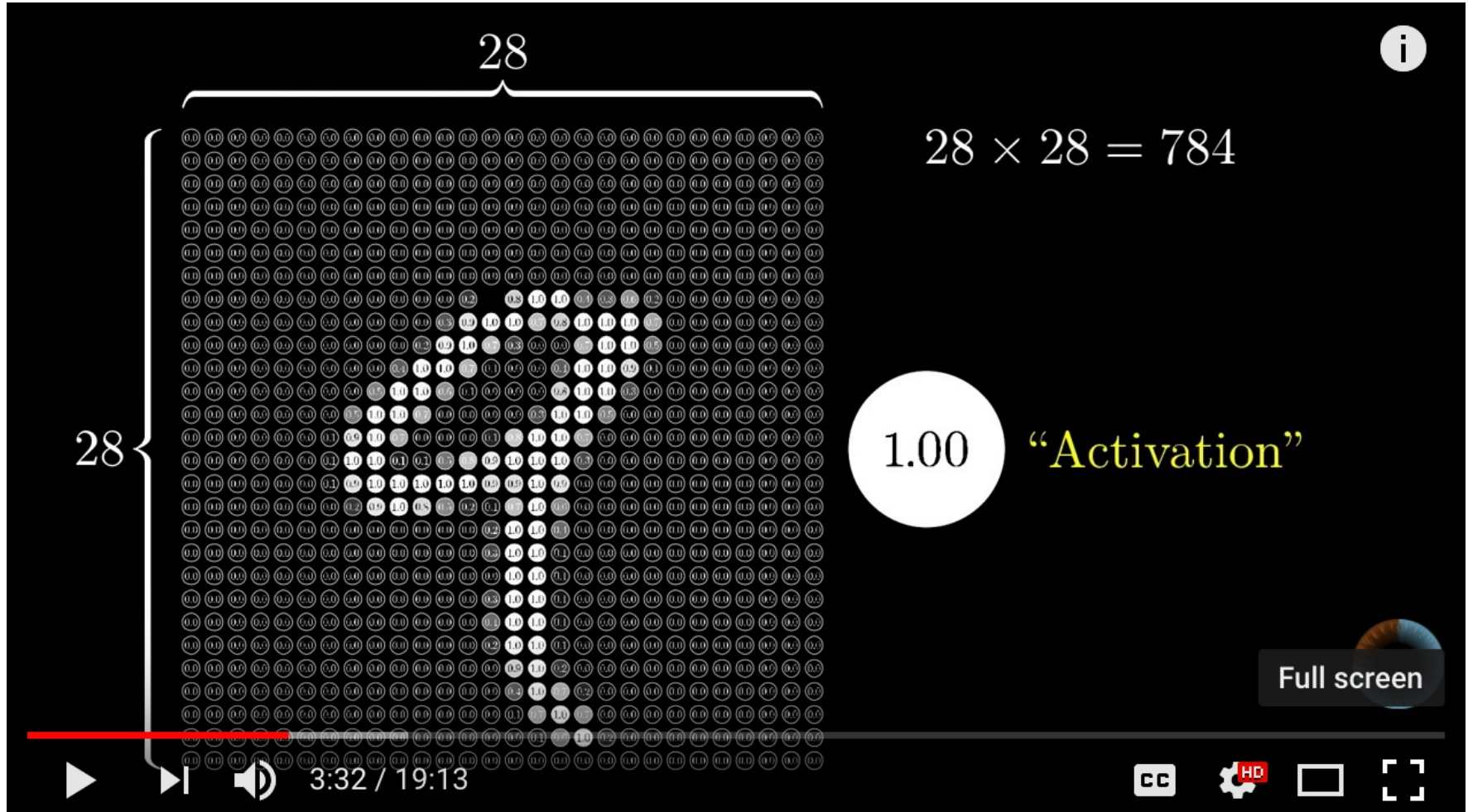
example of a convolutional layer



Convolutional Neural Networks (CNN) (LeNet)



Neural Network and Deep Learning



Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning,

<https://www.youtube.com/watch?v=aircAruvnKk>

Gradient Descent

how neural networks learn

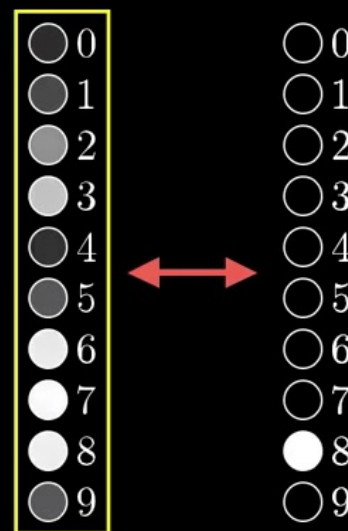
Average cost of
all training data...

Cost of

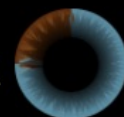


$$\left\{ \begin{array}{l} (0.18 - 0.00)^2 + \\ (0.29 - 0.00)^2 + \\ (0.58 - 0.00)^2 + \\ (0.77 - 0.00)^2 + \\ (0.20 - 0.00)^2 + \\ (0.36 - 0.00)^2 + \\ (0.93 - 0.00)^2 + \\ (1.00 - 0.00)^2 + \\ (0.95 - 1.00)^2 + \\ (0.35 - 0.00)^2 \end{array} \right.$$

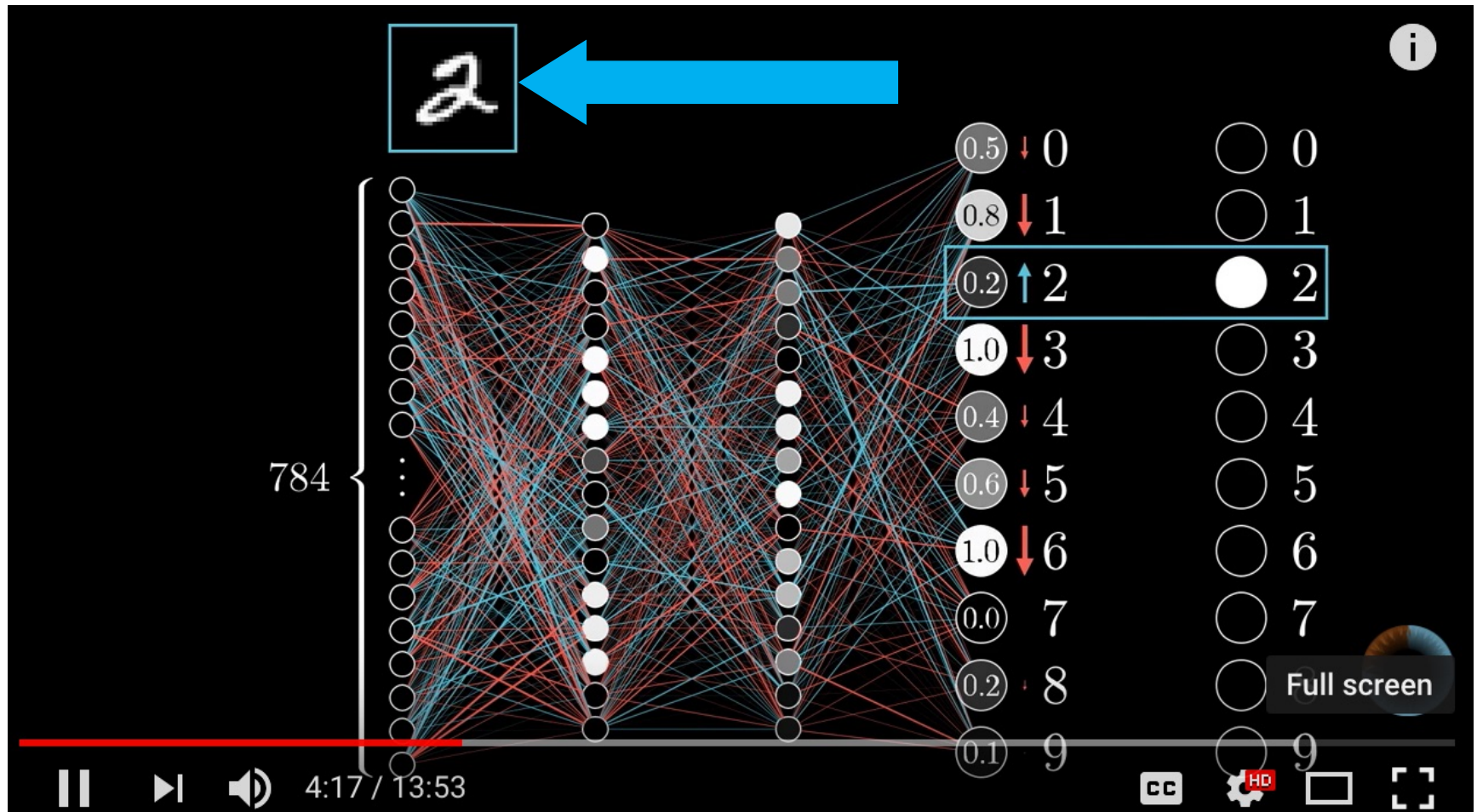
What's the "cost" ⁱ
of this difference?



Utter trash



Backpropagation



Learning Algorithm

While not done:

- Pick a random training example “(input, label)”

- Run neural network on “input”

- Adjust weights on edges to make output closer to “label”

Deep Learning with TensorFlow

Deep Learning Libraries: Tensorflow and Keras

Deep learning libraries: GitHub activity from February 11 to April 12, 2017

new contributors from 2017-02-11 to 2017-04-12			new forks from 2017-02-11 to 2017-04-12		
#1:	131	tensorflow/tensorflow	#1:	4192	tensorflow/tensorflow
#2:	63	fchollet/keras	#2:	991	fchollet/keras
#3:	51	pytorch/pytorch	#3:	810	BVLC/caffe
#4:	49	dmlc/mxnet	#4:	517	deeplearning4j/deeplearning4j
#5:	18	Theano/Theano	#5:	414	dmlc/mxnet
#6:	11	BVLC/caffe	#6:	307	pytorch/pytorch
#7:	11	Microsoft/CNTK	#7:	244	Microsoft/CNTK
#8:	9	tflearn/tflearn	#8:	211	tflearn/tflearn
#9:	9	pfnet/chainer	#9:	134	torch/torch7
#10:	8	torch/torch7	#10:	131	Theano/Theano
#11:	5	deeplearning4j/deeplearning4j	#11:	116	baidu/paddle
#12:	4	NVIDIA/DIGITS	#12:	88	NVIDIA/DIGITS
#13:	3	baidu/paddle	#13:	55	pfnet/chainer

new issues from 2017-02-11 to 2017-04-12			aggregate activity from 2017-02-11 to 2017-04-12		
#1:	1175	tensorflow/tensorflow	#1:	36.64	tensorflow/tensorflow
#2:	568	fchollet/keras	#2:	12.52	fchollet/keras
#3:	499	dmlc/mxnet	#3:	8.53	dmlc/mxnet
#4:	286	pytorch/pytorch	#4:	6.09	BVLC/caffe
#5:	257	Microsoft/CNTK	#5:	5.92	pytorch/pytorch
#6:	239	deeplearning4j/deeplearning4j	#6:	5.12	deeplearning4j/deeplearning4j
#7:	219	baidu/paddle	#7:	4.12	Microsoft/CNTK
#8:	173	Theano/Theano	#8:	2.93	Theano/Theano
#9:	171	BVLC/caffe	#9:	2.86	baidu/paddle
#10:	112	NVIDIA/DIGITS	#10:	2.17	tflearn/tflearn
#11:	84	tflearn/tflearn	#11:	1.68	NVIDIA/DIGITS
#12:	57	pfnet/chainer	#12:	1.38	torch/torch7
#13:	47	torch/torch7	#13:	1.12	pfnet/chainer

Deep Learning Software

- Keras
 - Deep Learning library for TensorFlow, CNTK
- Tensorflow
 - TensorFlow™ is an open source software library for numerical computation using data flow graphs.
- CNTK
 - Computational Network Toolkit by Microsoft Research
- PyTorch
 - Tensors and Dynamic neural networks in Python with strong GPU acceleration

Keras

Secure | <https://keras.io>

 Keras Documentation

Search docs

Home

Keras: The Python Deep Learning library

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Switching from TensorFlow to CNTK or Theano

Support

Why this name, Keras?

Why use Keras

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

 GitHub

Next »

Docs » Home

 Edit on GitHub

Keras: The Python Deep Learning library



You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of **TensorFlow**, **CNTK**, or **Theano**. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at [Keras.io](https://keras.io).

Keras is compatible with: **Python 2.7-3.6**.

<http://keras.io/>

Tensorflow

TensorFlow™

Install

Develop

API r1.4

Deploy

Extend

Community

Versions



Search

GITHUB

An open-source software library
for Machine Intelligence

GET STARTED



Eager Execution

We're announcing eager execution, an imperative, define-by-run interface to TensorFlow. Check out the README to get started today.



TensorFlow 1.4 has arrived!

We're excited to announce the release of TensorFlow 1.4! Check out the release notes for all the latest.



Announcing TensorFlow Lite

Learn more about TensorFlow's lightweight solution for mobile and embedded devices.

<https://www.tensorflow.org/>

PyTorch

[Get Started](#)[About](#)[Blog](#)[Support](#)[Discuss](#)[Docs](#)

Fork me on GitHub

Tensors and Dynamic neural networks in Python
with strong GPU acceleration.

PyTorch is a deep learning framework that puts Python first.

We are in an early-release Beta. Expect some adventures.

[Learn More](#)



Keras



Keras

- Keras is a **high-level neural networks API**
- Written in Python and capable of running on top of **TensorFlow, CNTK, or Theano**.
- It was developed with a focus on enabling fast experimentation.
- Being able to go from idea to result with the least possible delay is key to doing good research.

Install Keras

- Step 1. Install backend engines: **Tensorflow**
 - Installing TensorFlow on **Ubuntu**
 - Installing TensorFlow on **macOS**
 - Installing TensorFlow on **Windows**
- Step 2. Install **Keras**
 - sudo pip install keras
 - pip install keras

TensorFlow Installation



Install

Installing TensorFlow

Installing TensorFlow on Ubuntu

Installing TensorFlow on macOS

Installing TensorFlow on Windows

Installing TensorFlow from Sources

Transitioning to TensorFlow 1.0

Installing TensorFlow for Java

Installing TensorFlow for Go

Installing TensorFlow for C

Installing TensorFlow

We've built and tested TensorFlow on the following 64-bit laptop/desktop operating systems:

- MacOS X 10.11 (El Capitan) or later.
- Ubuntu 14.04 or later
- Windows 7 or later.

Although you might be able to install TensorFlow on other laptop or desktop systems, we only support (and only fix issues in) the preceding configurations.

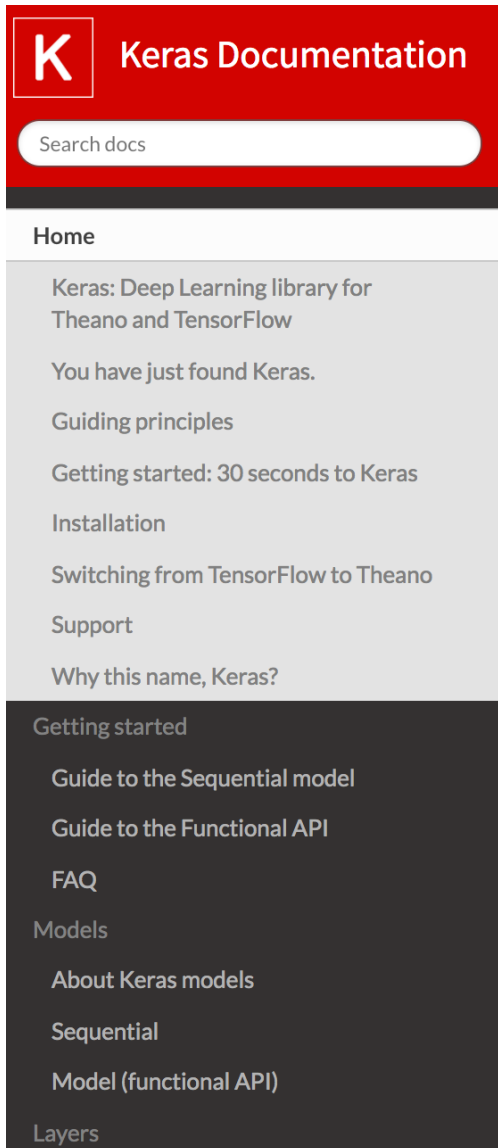
The following guides explain how to install a version of TensorFlow that enables you to write applications in Python:

- [Installing TensorFlow on Ubuntu](#)
- [Installing TensorFlow on macOS](#)
- [Installing TensorFlow on Windows](#)
- [Installing TensorFlow from Sources](#)

Many aspects of the Python TensorFlow API changed from version 0.n to 1.0. The following guide explains how to migrate older TensorFlow applications to Version 1.0:

<https://www.tensorflow.org/install/>

Keras Installation



Installation

Keras uses the following dependencies:

- numpy, scipy
- yaml
- HDF5 and h5py (optional, required if you use model saving/loading functions)
- Optional but recommended if you use CNNs: cuDNN.

When using the TensorFlow backend:

- TensorFlow
 - See installation instructions.

When using the Theano backend:

- Theano
 - See installation instructions.

To install Keras, `cd` to the Keras folder and run the install command:

```
sudo python setup.py install
```

You can also install Keras from PyPI:

```
sudo pip install keras
```

<https://keras.io/#installation>



TensorFlow

Google TensorFlow

TensorFlow™

GET STARTED TUTORIALS HOW TO API RESOURCES ABOUT

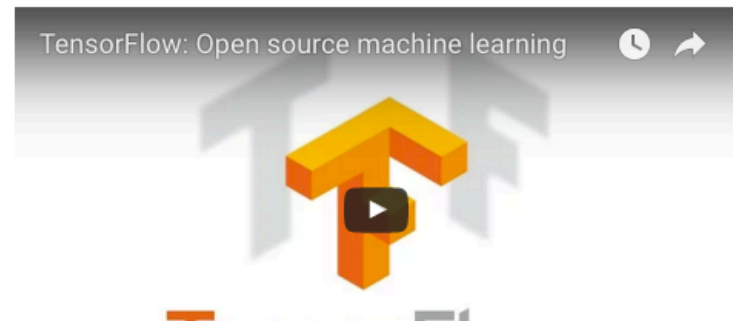
Fork me on GitHub

TensorFlow is an Open Source Software
Library for Machine Intelligence

GET STARTED

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.



<https://www.tensorflow.org/>

TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Iterations
000,582

Learning rate

0.03

Activation

Tanh

Regularization

None

Regularization rate

0

Problem type

Classification

DATA

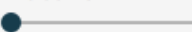
Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



INPUT

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$



3 HIDDEN LAYERS



4 neurons



2 neurons



2 neurons

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$

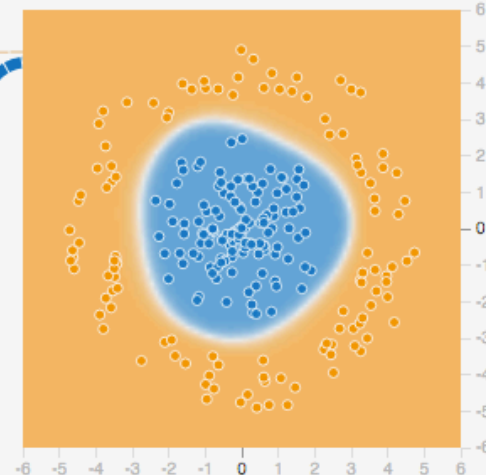
This is the output from one **neuron**. Hover to see it larger.

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

OUTPUT

Test loss 0.000

Training loss 0.000



TensorFlow
is an
Open Source
Software Library
for
Machine Intelligence

numerical computation using data flow graphs

Tensor

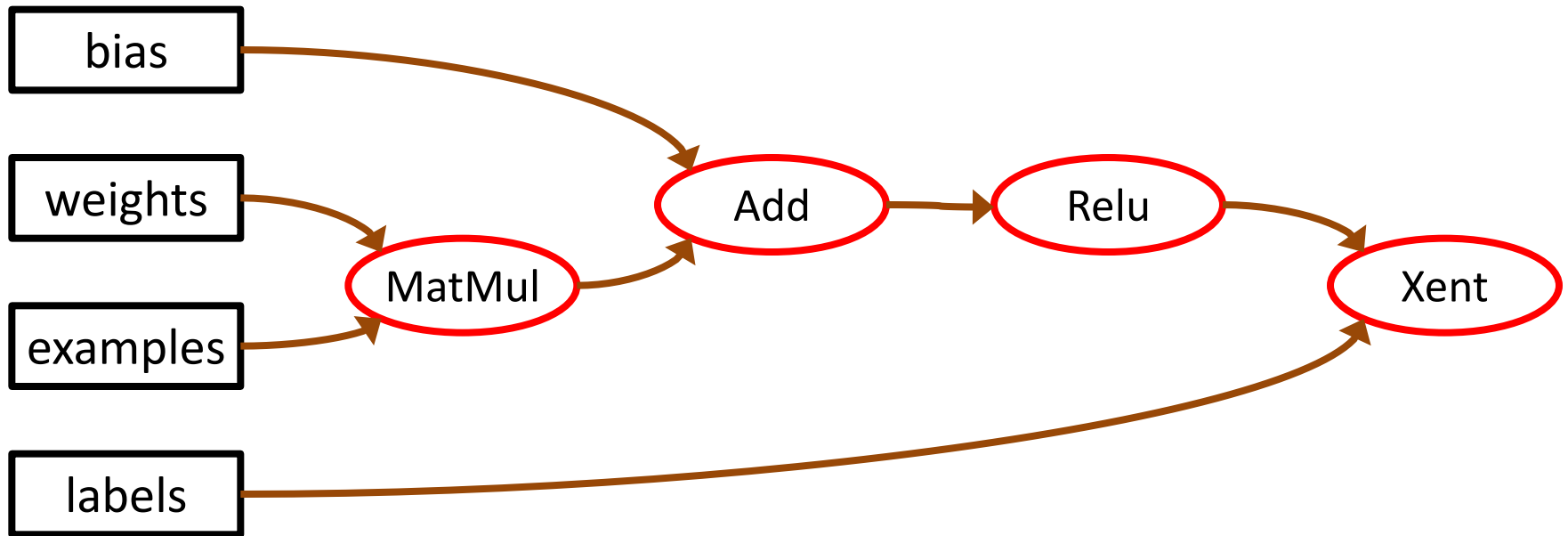
- 3
 - # a rank 0 tensor; this is a **scalar** with shape []
- [1., 2., 3.]
 - # a rank 1 tensor; this is a **vector** with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
 - # a rank 2 tensor; a **matrix** with shape [2, 3]
- [[[1., 2., 3.]], [[7., 8., 9.]]]
 - # a rank 3 **tensor** with shape [2, 1, 3]

Nodes:
mathematical operations

edges:
multidimensional data arrays
(tensors)
communicated between nodes

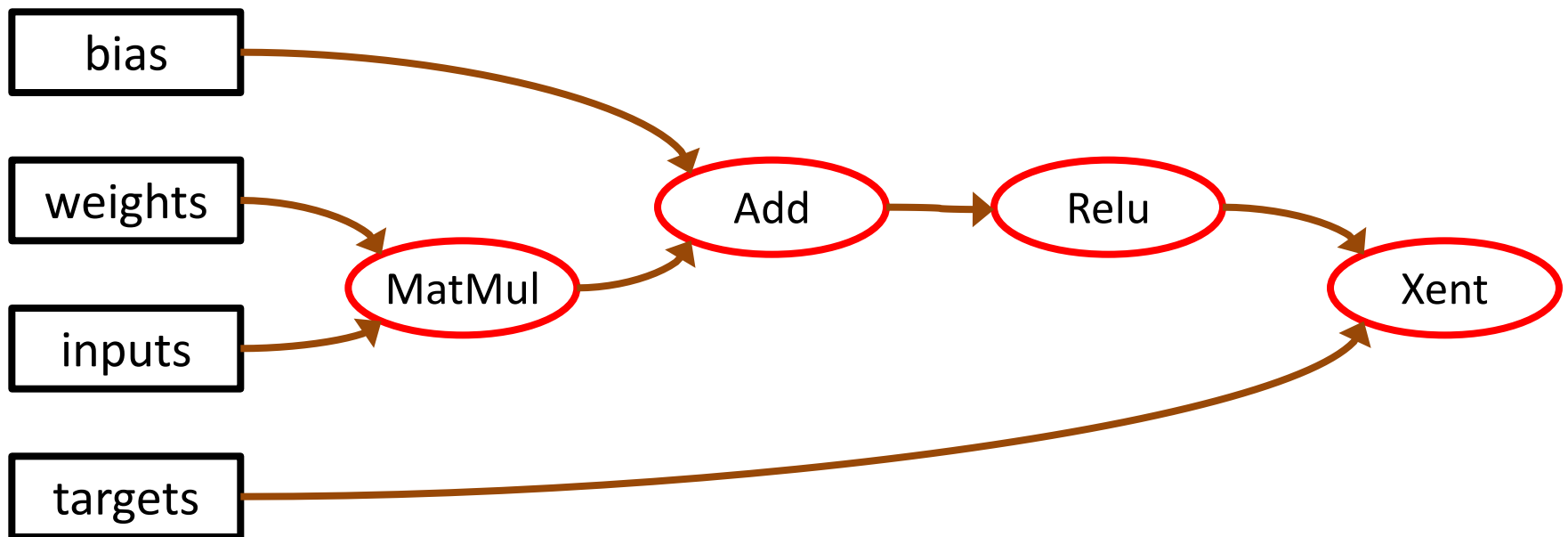
Computation is a Dataflow Graph

Graph of **Nodes**,
also called **Operations** or **ops**.

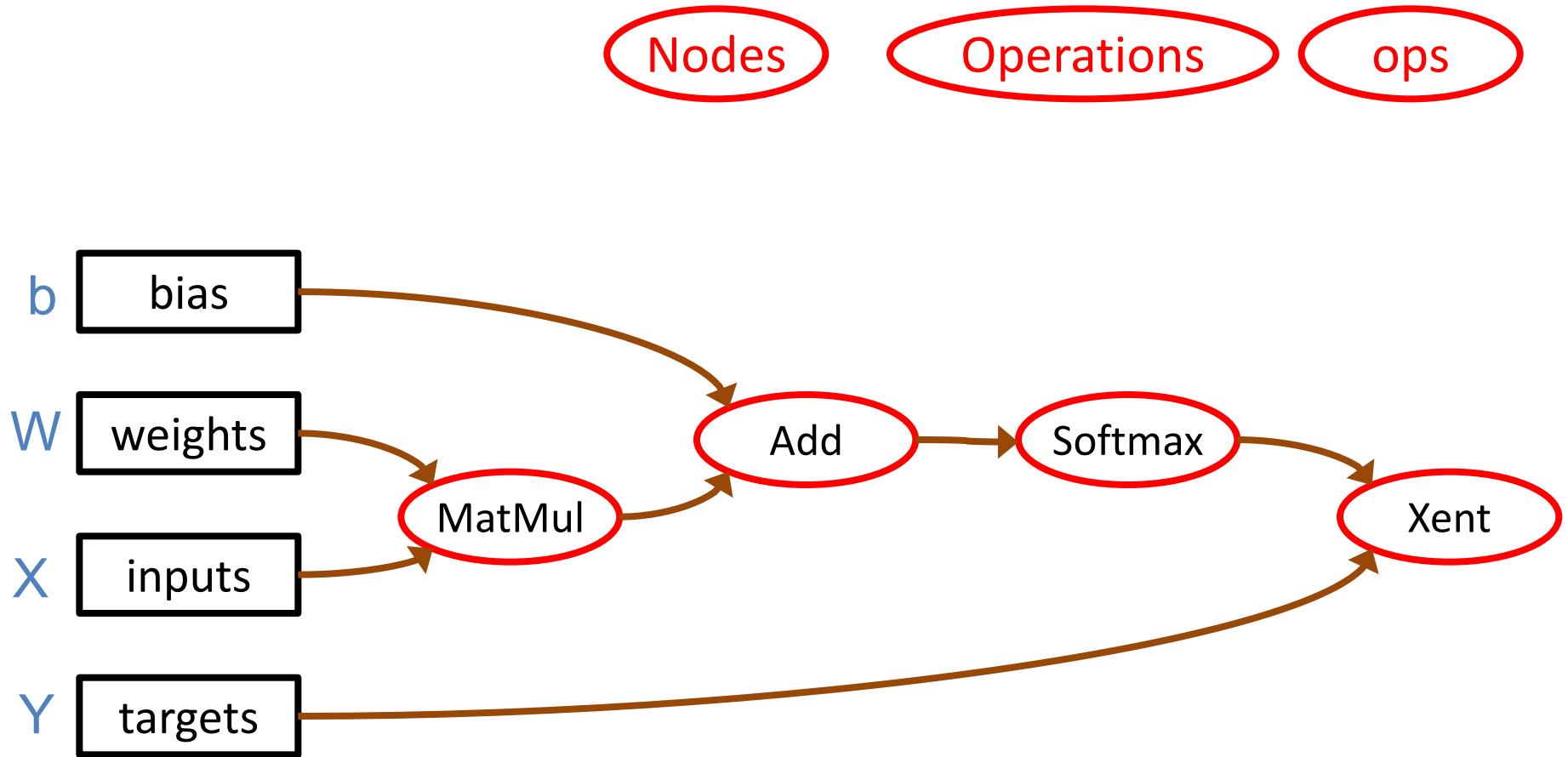


Computation is a Dataflow Graph

Edges are N-dimensional arrays: **Tensors**



Logistic Regression as Dataflow Graph

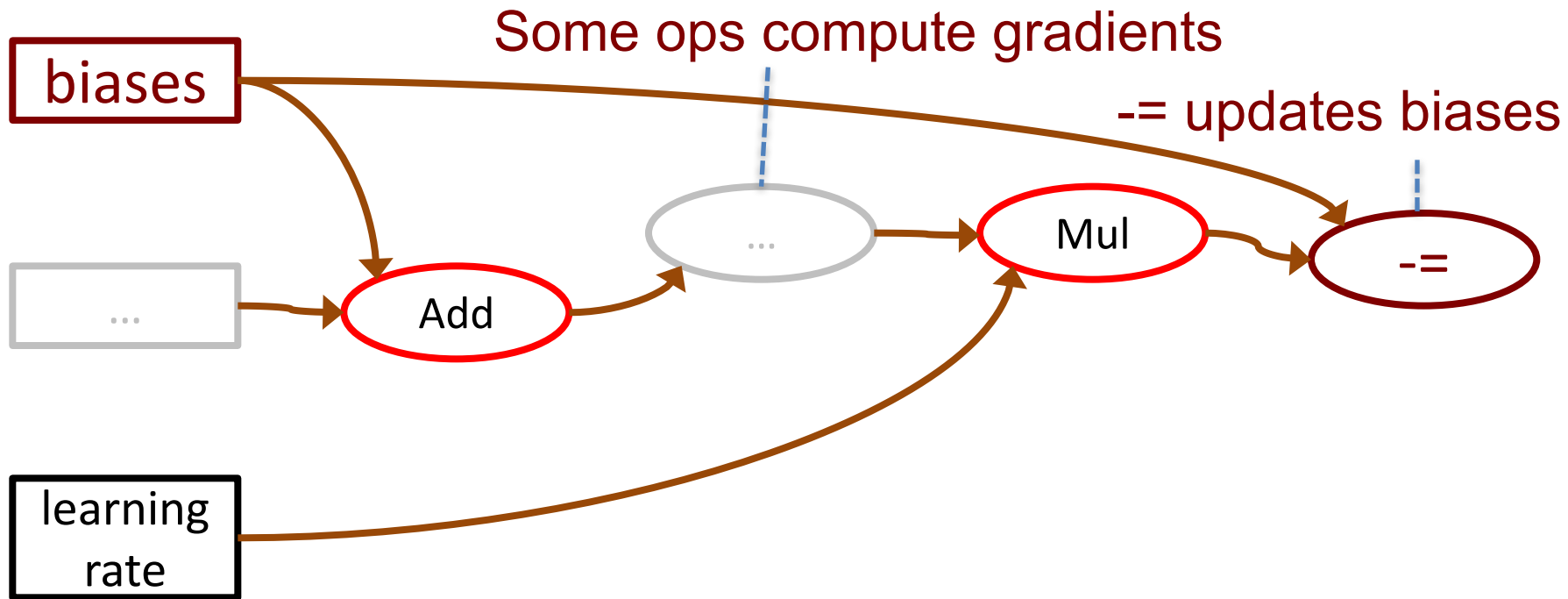


Edges are N-dimensional arrays: **Tensors**

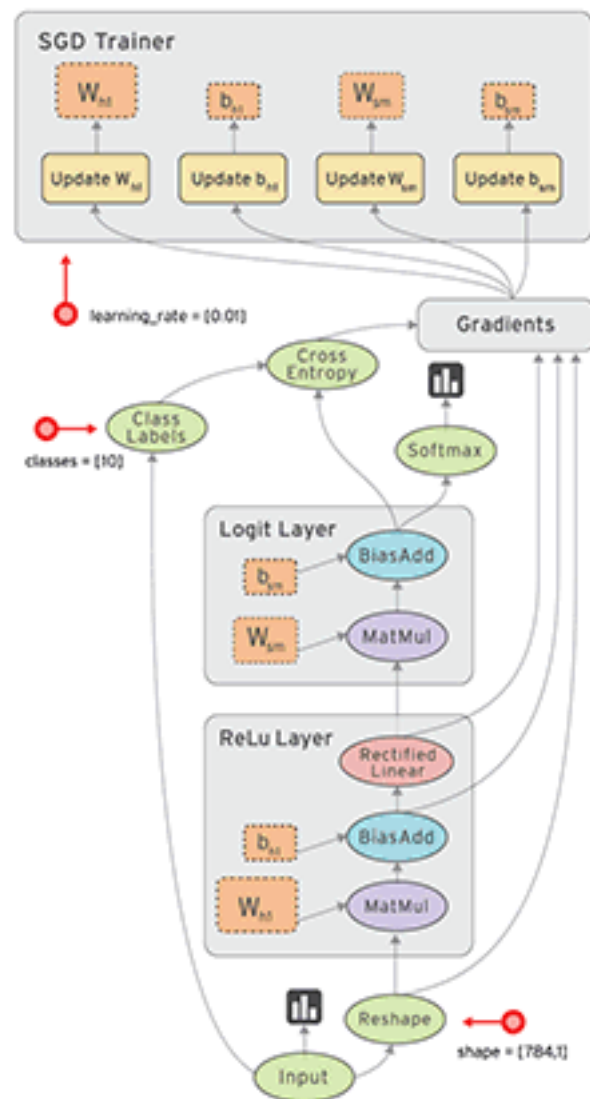
Computation is a Dataflow Graph

with state

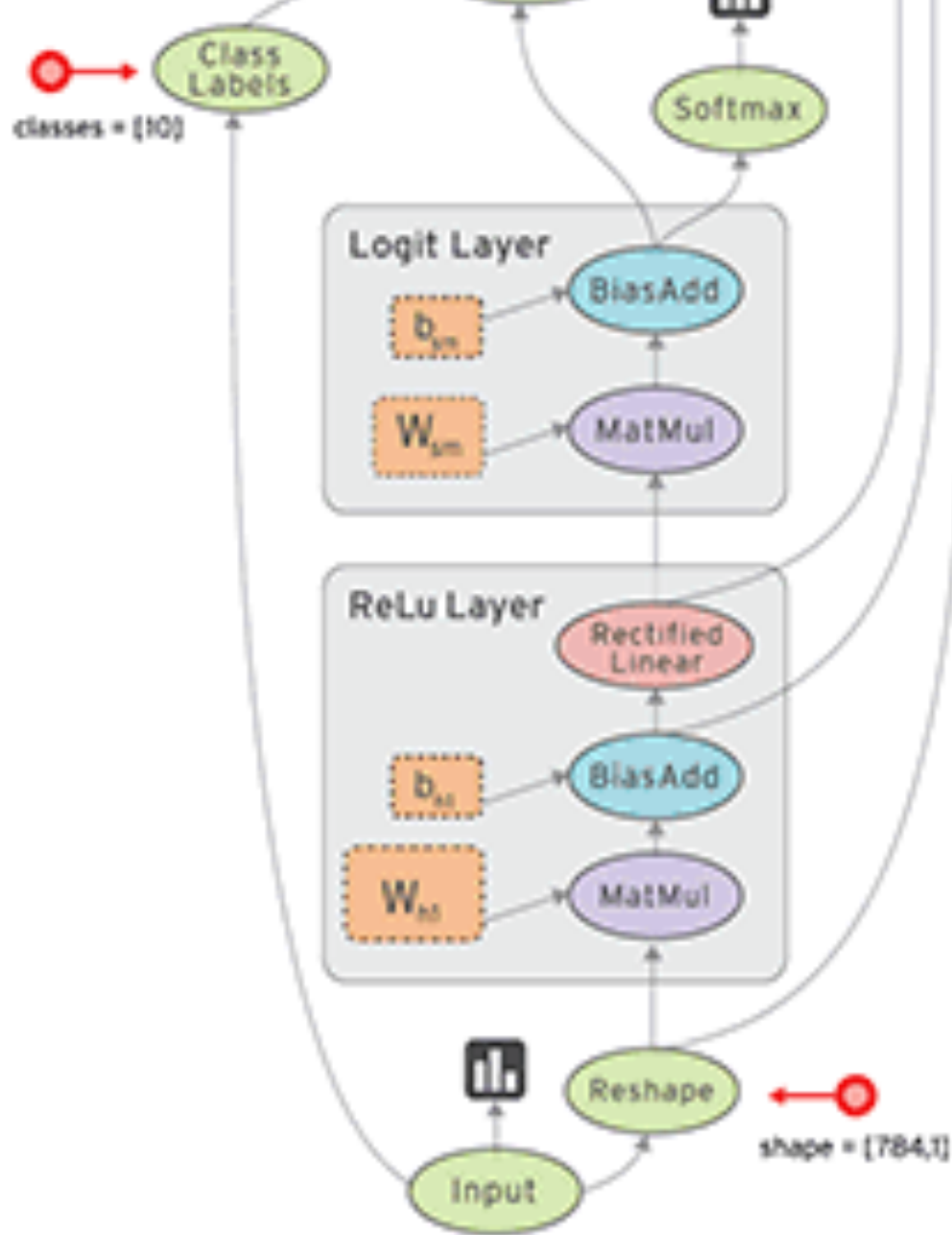
'Biases' is a variable



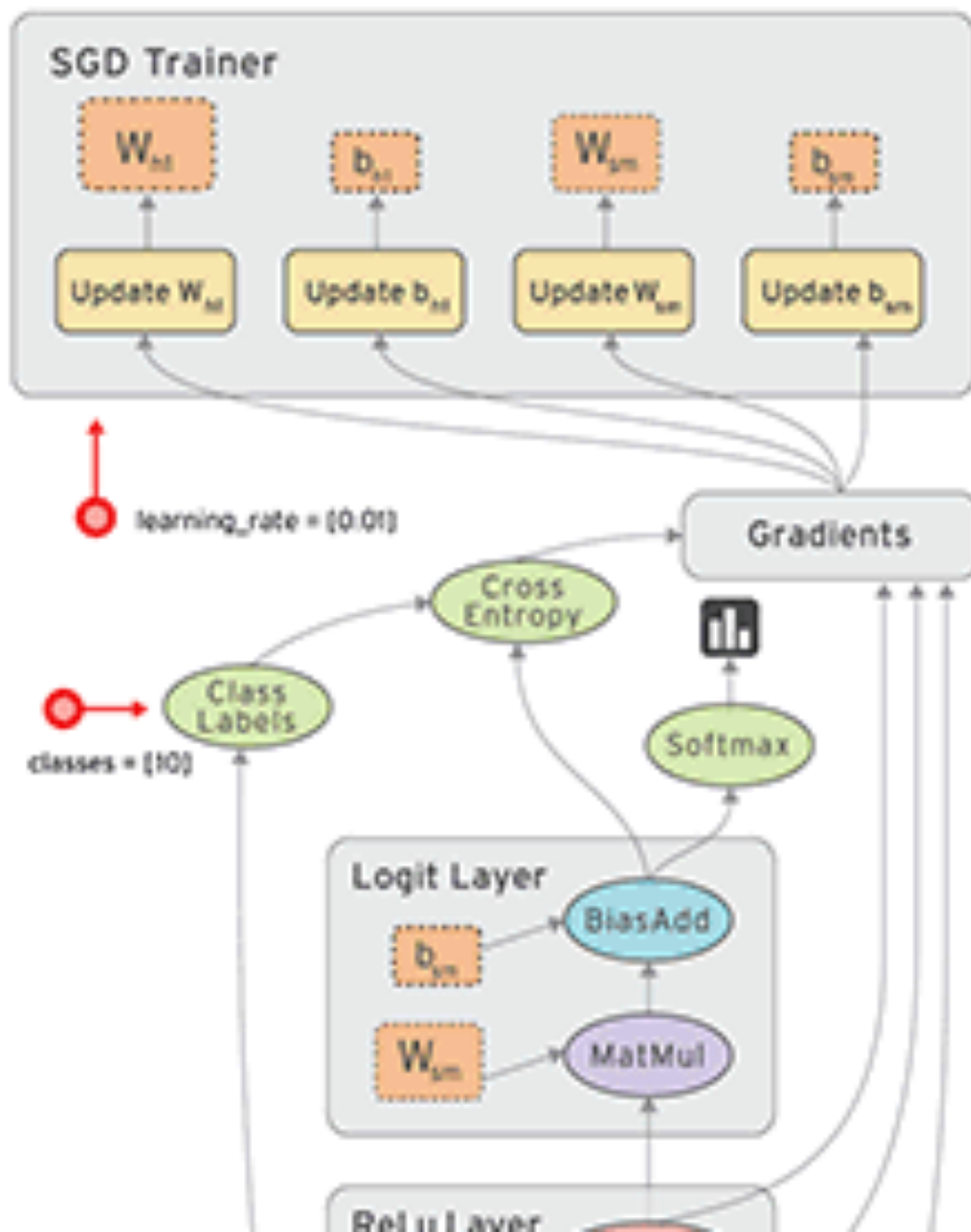
Data Flow Graph



Data Flow Graph



Data Flow Graph





Python

Download Anaconda



[Log In](#) [Get Support](#) [Search](#) [Contact](#)

[PRODUCTS](#) [COMMUNITY](#) [CONSULTING](#) [TRAINING](#) [ABOUT](#) [RESOURCES](#)

**DOWNLOAD
ANACONDA NOW**

Download for



GET SUPERPOWERS WITH **ANACONDA**

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Which version should I download and install?

With Anaconda you can run multiple versions of Python in isolated environments, so choose the download with the Python version that you use more often, as that will be your default Python version.

<https://www.continuum.io/downloads>

Download Anaconda Python 3.6

Download for Windows

Download for macOS

Download for Linux

Anaconda 4.3.1

For macOS

macOS 10.12.2 users: To prevent permissions problems, we recommend that you upgrade to macOS 10.12.3 or later before installing Anaconda.

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

[Changelog](#)

Graphical Installer

1. Download the graphical installer
2. Double-click the downloaded **.pkg** file and follow the instructions

Command Line Installer

1. Download the command-line installer
2. Optional: Verify data integrity with [MD5 or SHA-256](#) [More info](#)
3. In your terminal window type one of the below and follow the instructions:
Python 3.6 version

Python 3.6 version

GRAPHICAL INSTALLER (424M)

COMMAND-LINE INSTALLER (363M)

64-Bit

Python 2.7 version

GRAPHICAL INSTALLER (419M)

COMMAND-LINE INSTALLER (358M)

64-Bit

GET ANACONDA SUPPORT

OS X Anaconda Python 3.6

Installation

Command Line Installer

Download the command-line installer

In your terminal window type one of the below and follow the instructions:

Python 3.6 version

```
bash Anaconda3-4.3.1-MacOSX-x86_64.sh
```

Python 2.7 version

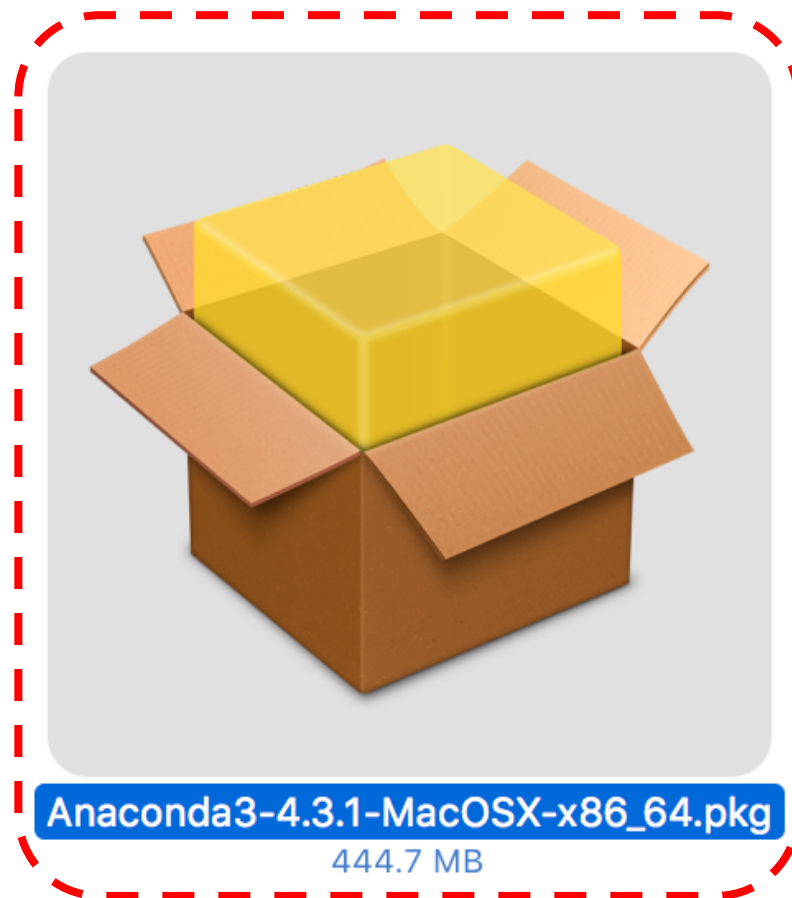
```
bash Anaconda2-4.3.1-MacOSX-x86_64.sh
```

<https://www.continuum.io/downloads>

OS X Anaconda 3 - 4.3.1

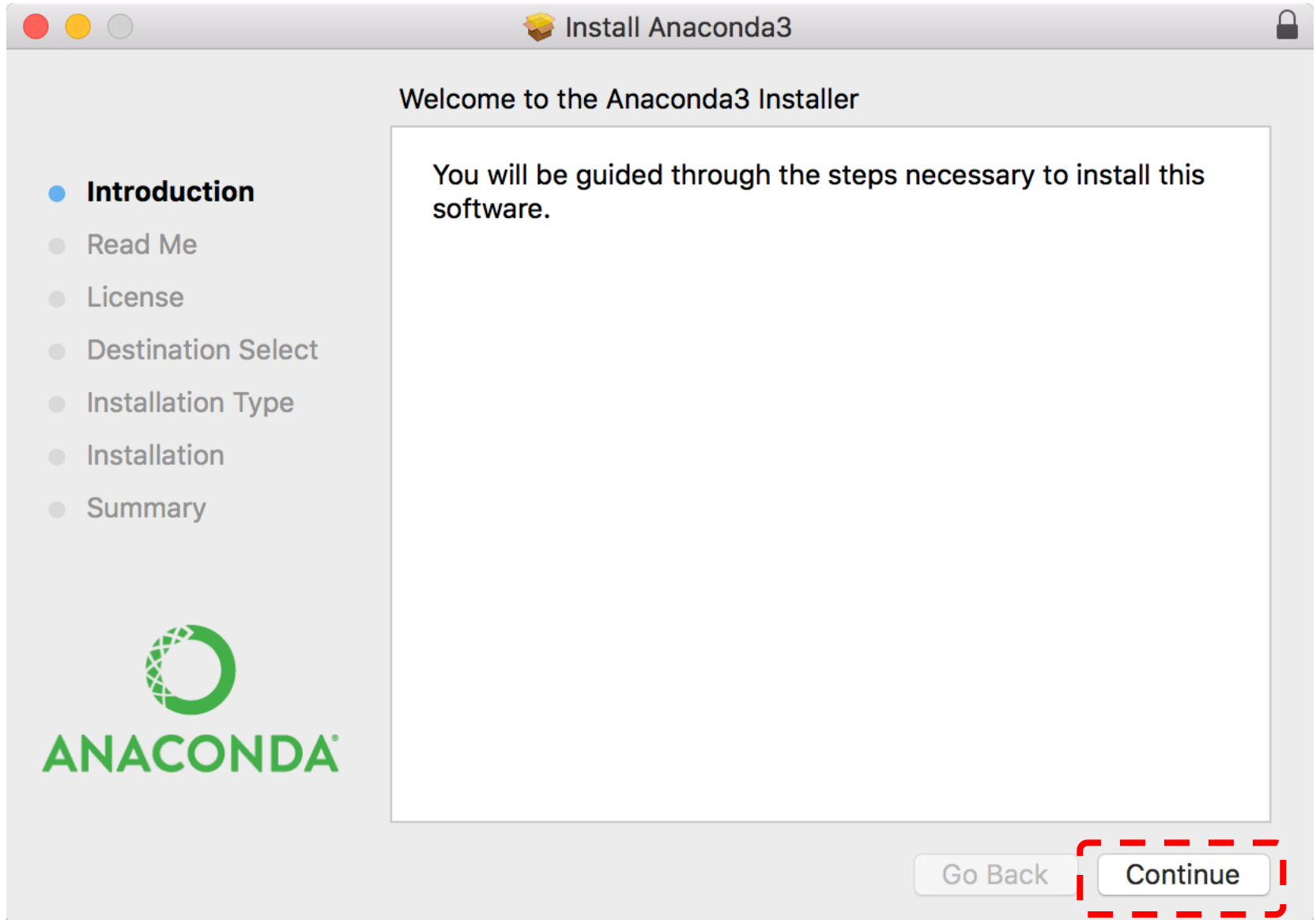
Python 3.6 Installation

Anaconda3-4.3.1-MacOSX-x86_64.pkg

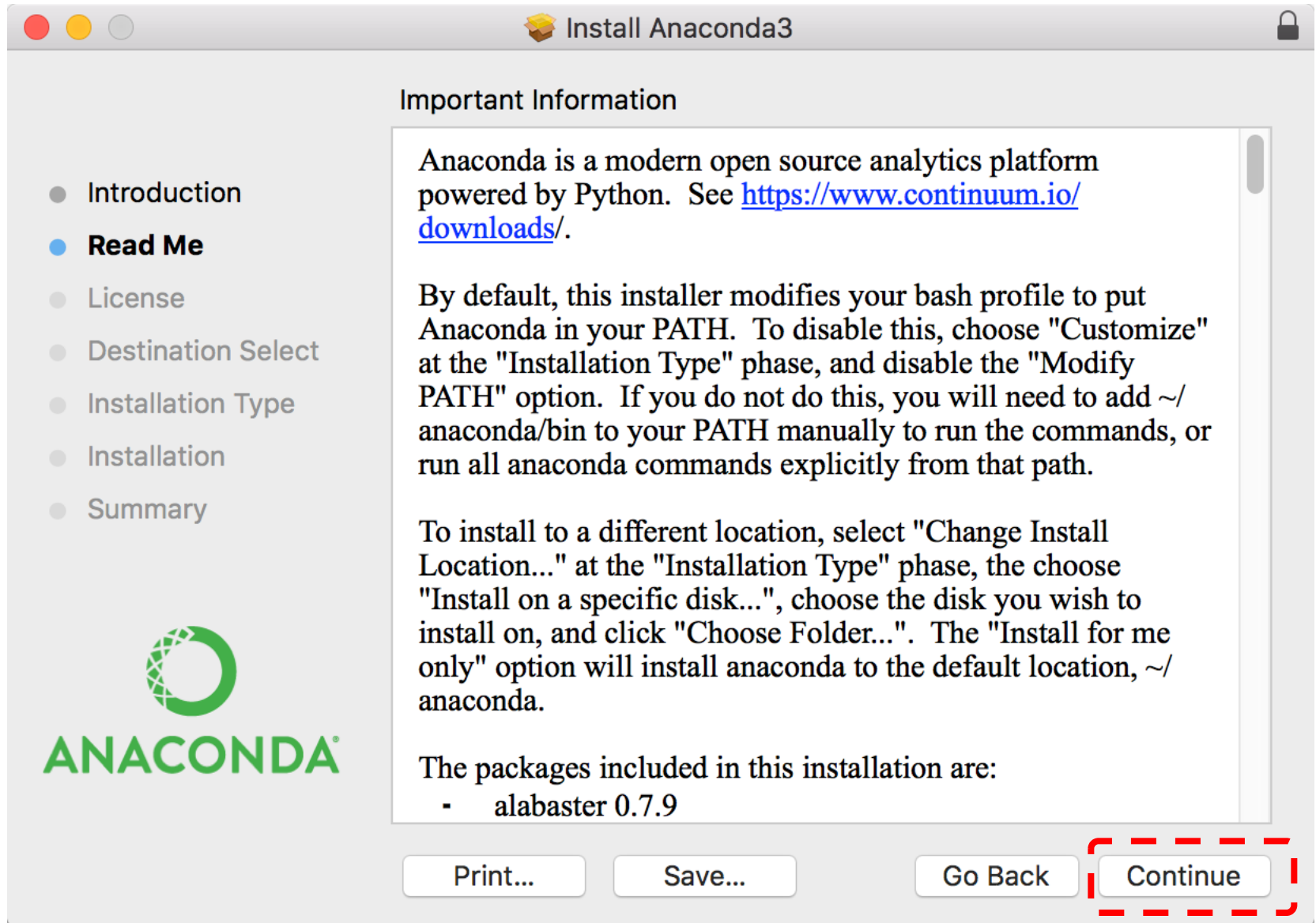


Installer package

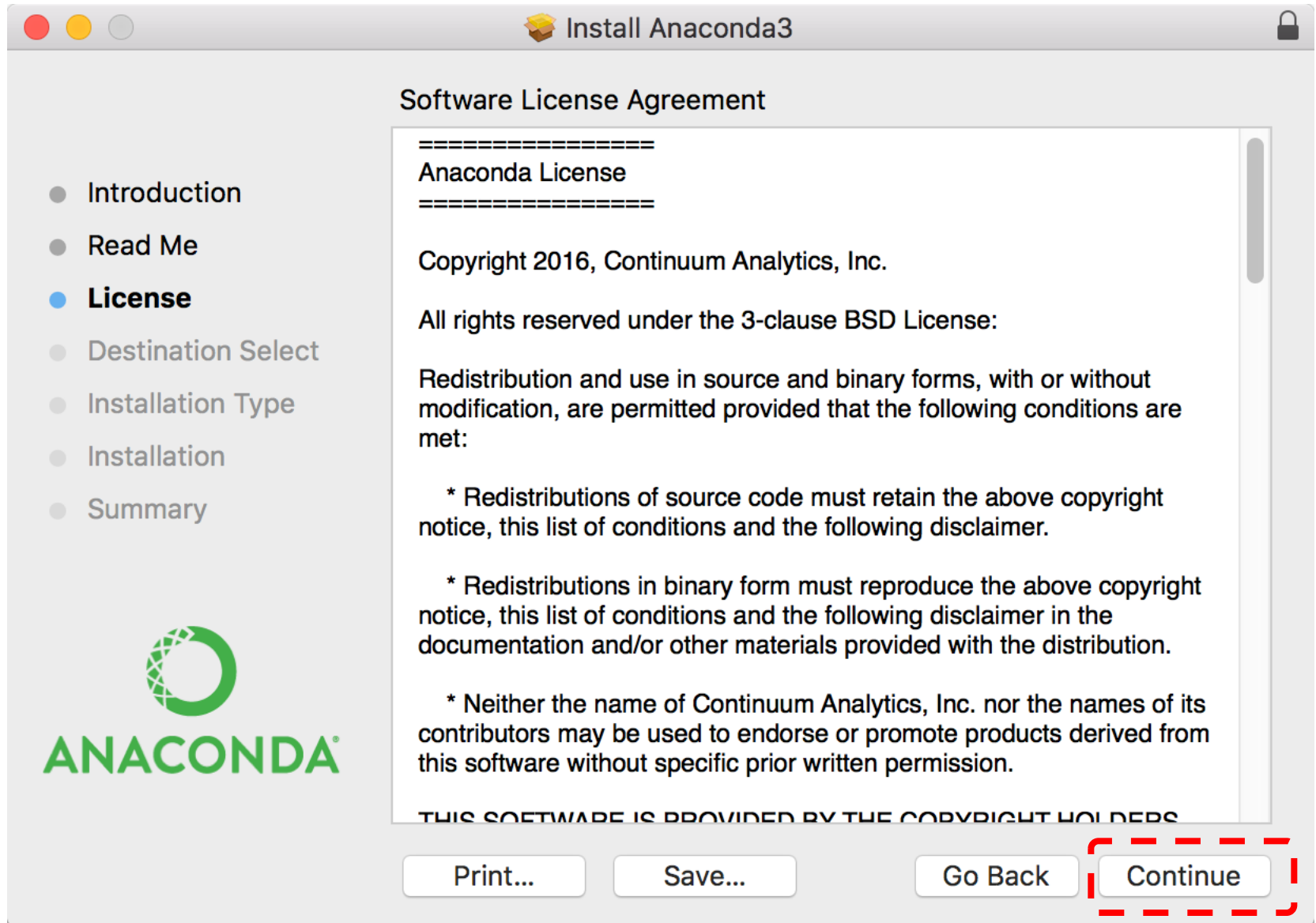
Install Anaconda 3



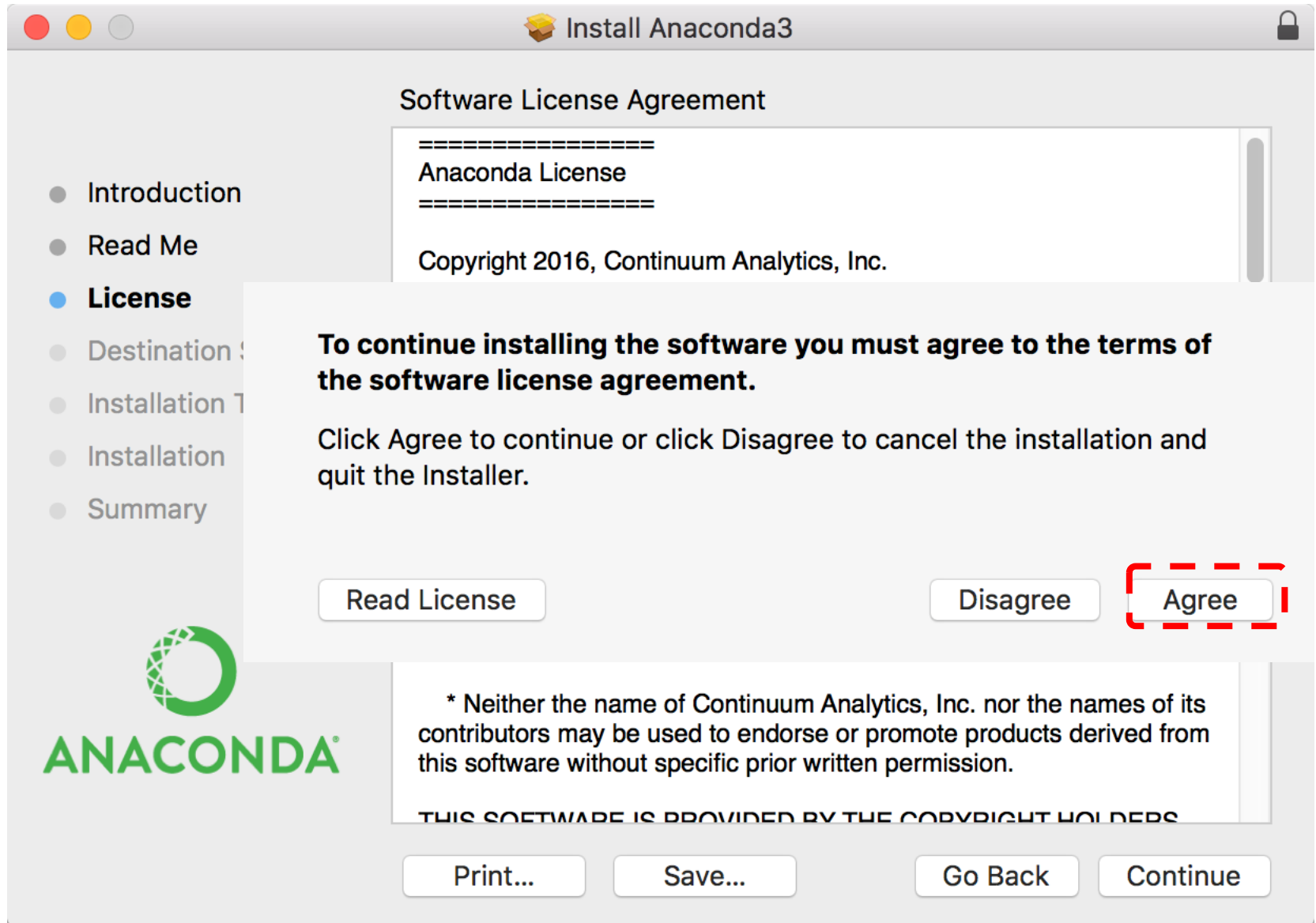
Install Anaconda 3



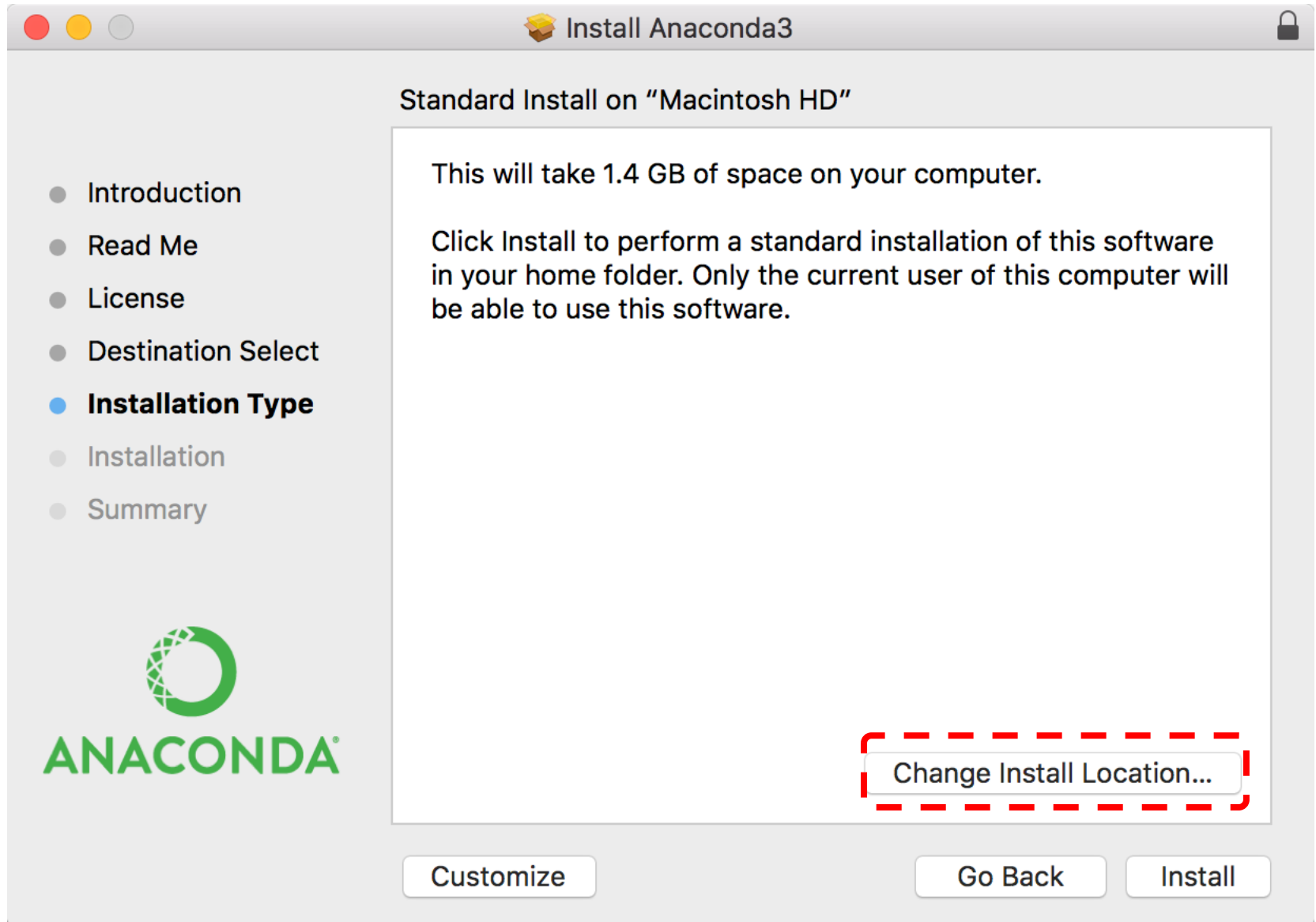
Install Anaconda 3



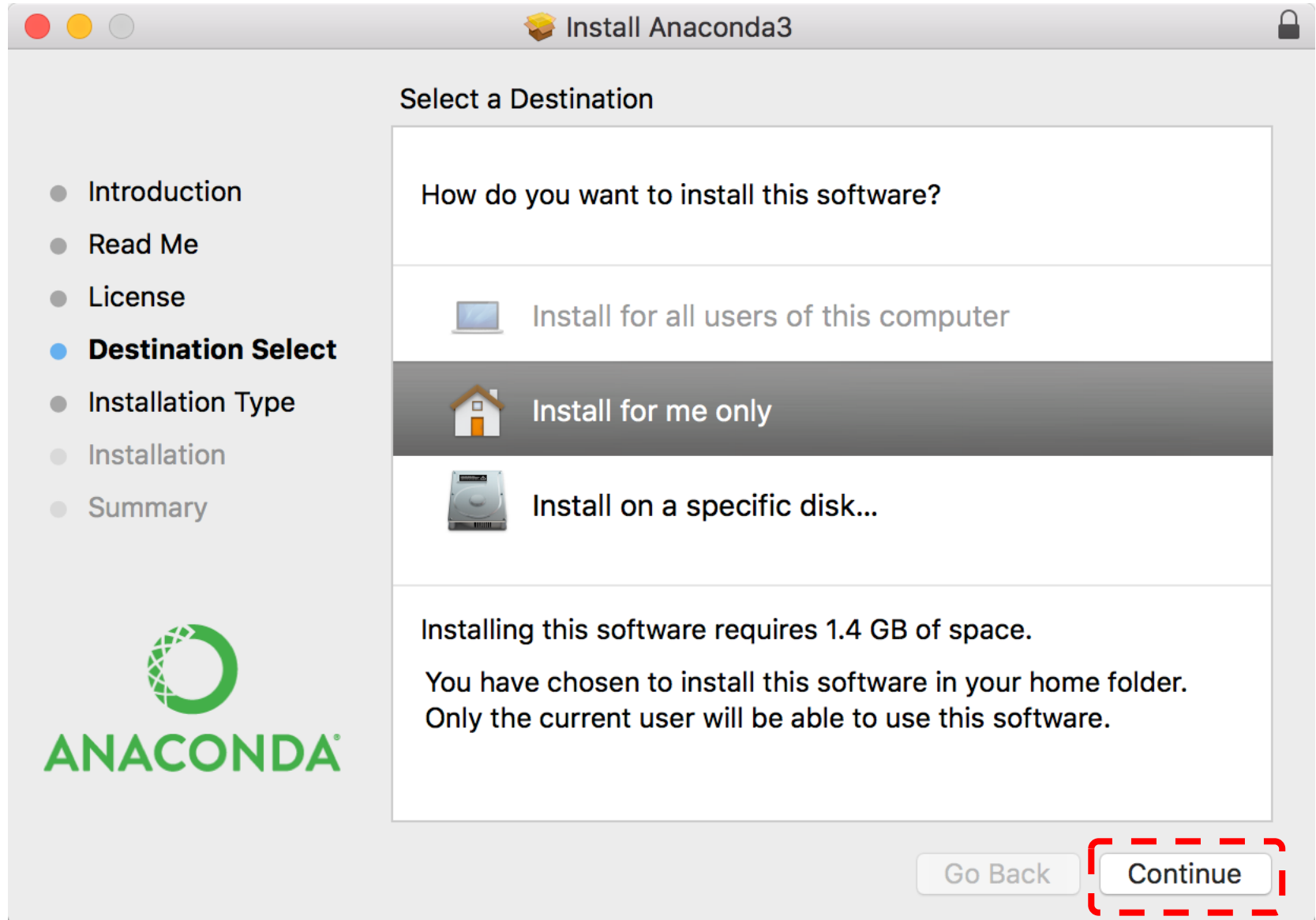
Install Anaconda 3



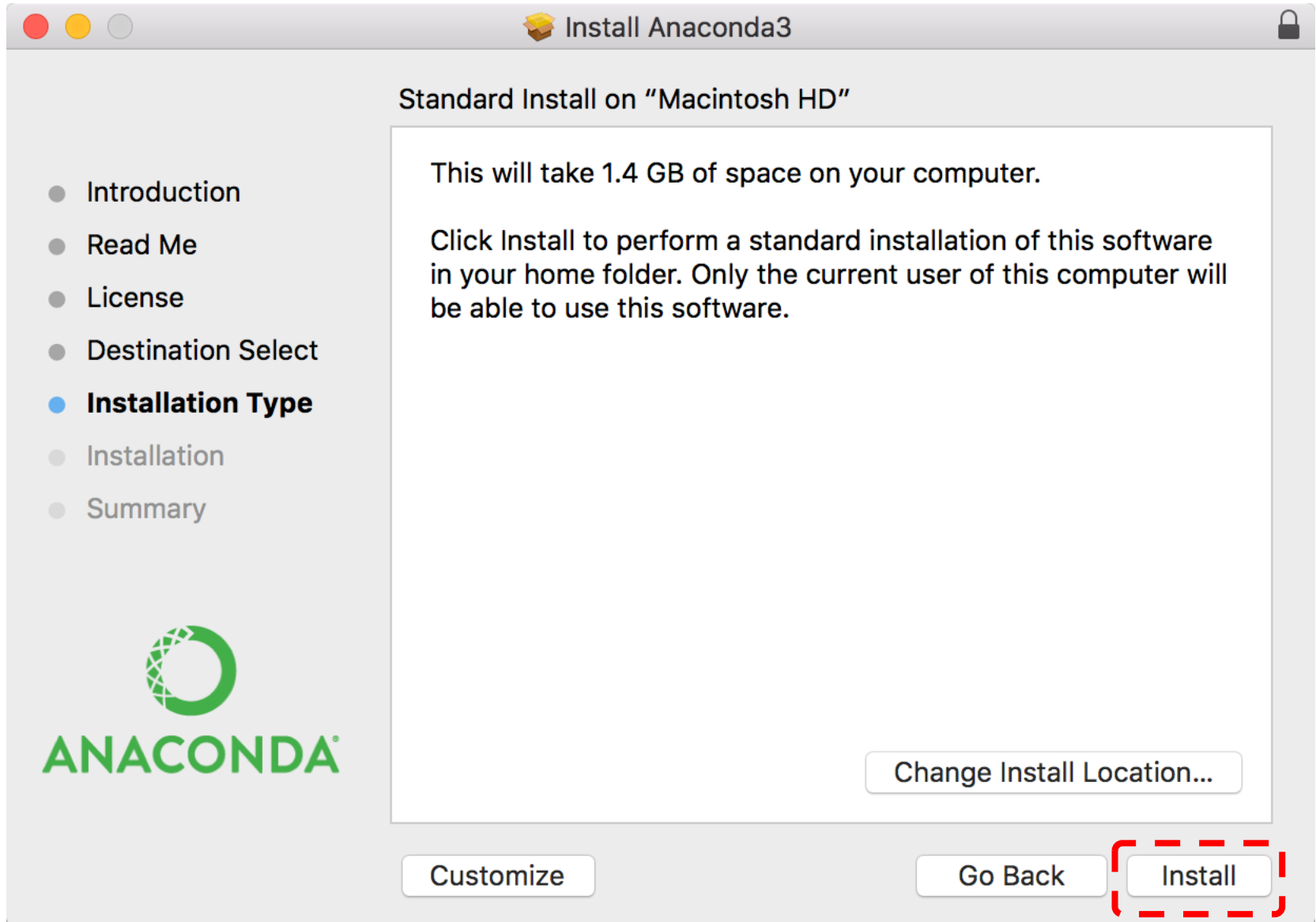
Install Anaconda 3



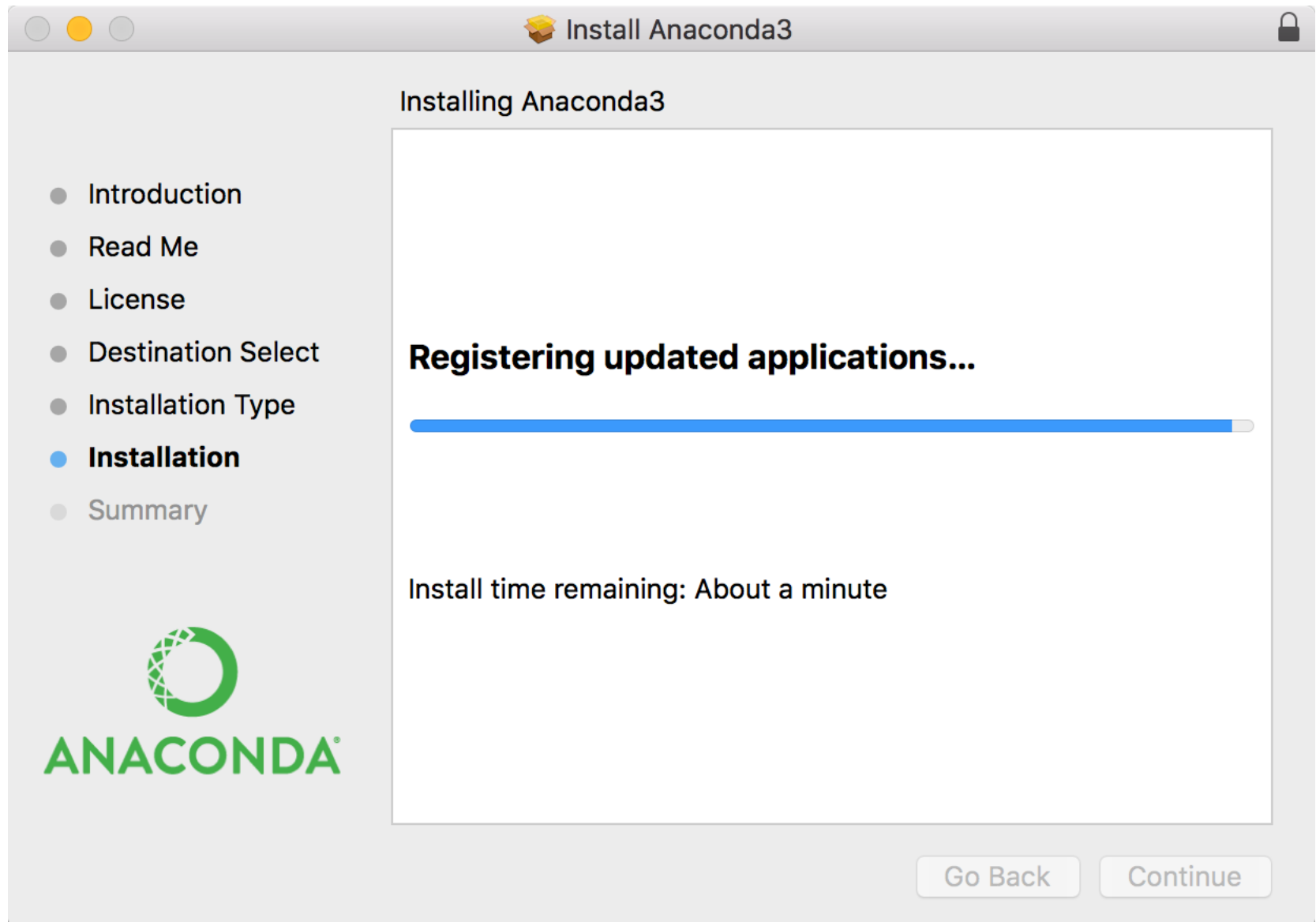
Install Anaconda 3



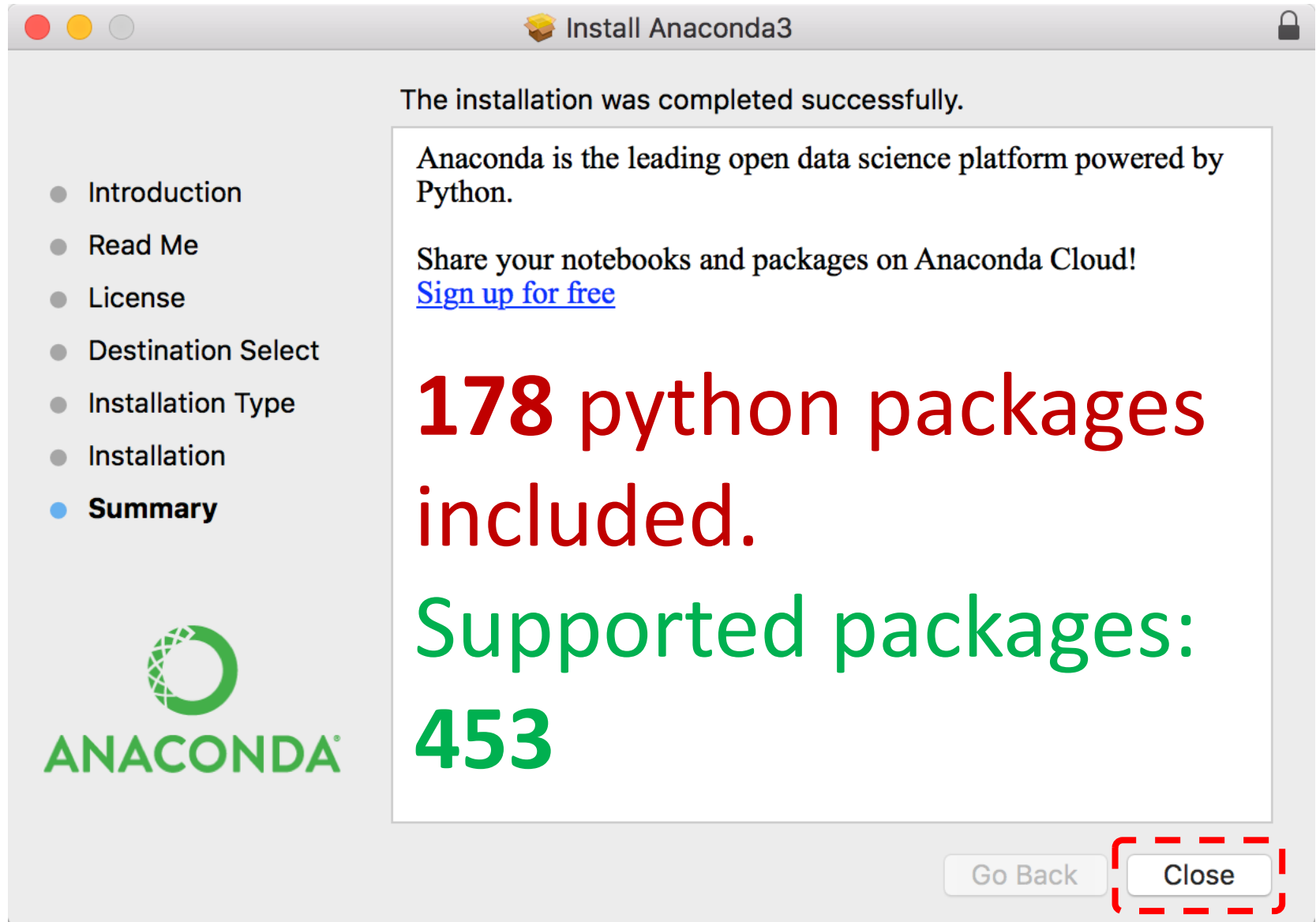
Install Anaconda 3



Install Anaconda 3



Install Anaconda 3

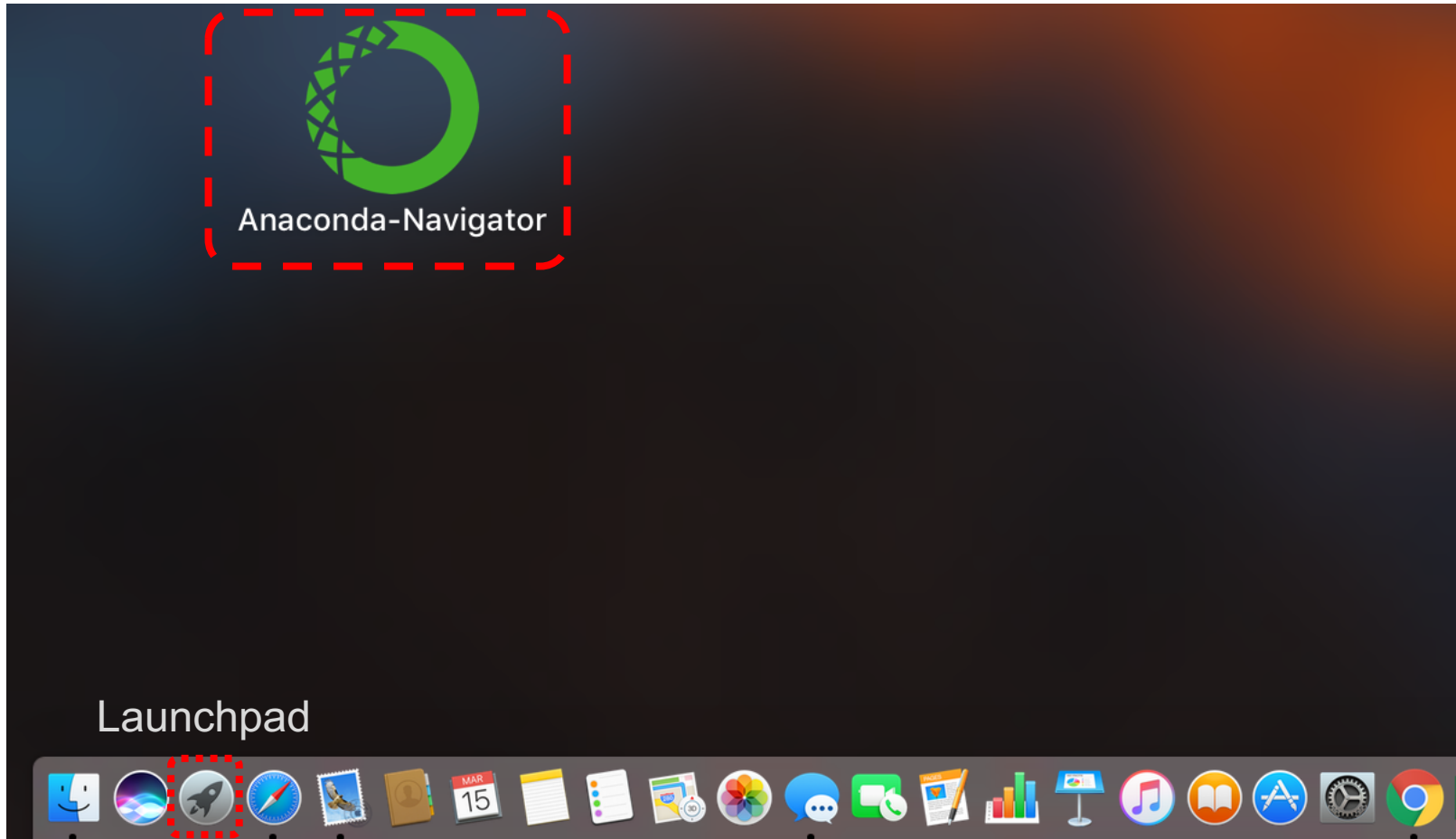


Install Anaconda 3

1	_license	1.1	51	heapdict	1.0.0	101	partd	0.3.7	151	sip	4.18	py36_0
2	alabaster	0.7.9	52	icu	54.1	102	path.py	10.0	152	six	1.10.0	py36_0
3	anaconda	4.3.1	53	idna	2.2	103	pathlib2	2.2.0	153	snowballstemmer	1.2.1	py36_0
4	anaconda-client	1.6.0	54	imagesize	0.7.1	104	patsy	0.4.1	154	sockjs-tornado	1.0.3	py36_0
5	anaconda-navigator	1.5.0	55	ipykernel	4.5.2	105	pep8	1.7.0	155	sphinx	1.5.1	py36_0
6	anaconda-project	0.4.1	56	ipython	5.1.0	106	pexpect	4.2.1	156	spyder	3.1.2	py36_0
7	appnope	0.1.0	57	ipython_genutils	0.1.0	107	pickleshare	0.7.4	157	sqlalchemy	1.1.5	py36_0
8	appscript	1.0.1	58	ipywidgets	5.2.2	108	pillow	4.0.0	158	sqlite	3.13.0	0
9	astroid	1.4.9	59	isort	4.2.5	109	pip	9.0.1	159	statsmodels	0.6.1	np111py36_1
10	astropy	1.3	60	itsdangerous	0.24	110	ply	3.9	160	sympy	1.0	py36_0
11	babel	2.3.4	61	jbig	2.1	111	prompt_toolkit	1.0.9	161	terminado	0.6	py36_0
12	backports	1.0	62	jdcal	1.3	112	psutil	5.0.1	162	tk	8.5.18	0
13	beautifulsoup4	4.5.3	63	jedi	0.9.0	113	ptyprocess	0.5.1	163	toolz	0.8.2	py36_0
14	bitarray	0.8.1	64	jinja2	2.9.4	114	py	1.4.32	164	tornado	4.4.2	py36_0
15	blaze	0.10.1	65	jpeg	9b	115	pyasn1	0.1.9	165	traitlets	4.3.1	py36_0
16	bokeh	0.12.4	66	jsonschema	2.5.1	116	pycosat	0.6.1	166	unicodesv	0.14.1	py36_0
17	boto	2.45.0	67	jupyter	1.0.0	117	pycparser	2.17	167	wcwidth	0.1.7	py36_0
18	bottleneck	1.2.0	68	jupyter_client	4.4.0	118	pycrypto	2.6.1	168	werkzeug	0.11.15	py36_0
19	cffi	1.9.1	69	jupyter_console	5.0.0	119	pycurl	7.43.0	169	wheel	0.29.0	py36_0
20	chardet	2.3.0	70	jupyter_core	4.2.1	120	pyflakes	1.5.0	170	widgetsnbextension	1.2.6	py36_0
21	chest	0.2.3	71	lazy-object-proxy	1.2.2	121	pygments	2.1.3	171	wrapit	1.10.8	py36_0
22	click	6.7	72	libiconv	1.14	122	pylint	1.6.4	172	xldr	1.0.0	py36_0
23	cloudpickle	0.2.2	73	libpng	1.6.27	123	pyopenssl	16.2.0	173	xlswriter	0.9.6	py36_0
24	clyent	1.2.2	74	libtiff	4.0.6	124	pyparsing	2.1.4	174	xlwings	0.10.2	py36_0
25	colorama	0.3.7	75	libxml2	2.9.4	125	pyqt	5.6.0	175	xlwt	1.2.0	py36_0
26	conda	4.3.14	76	libxslt	1.1.29	126	pytables	3.3.0	176	xz	5.2.2	1
27	conda-env	2.6.0	77	llvmlite	0.15.0	127	pytest	3.0.5	177	yaml	0.1.6	0
28	configobj	5.0.6	78	locket	0.2.0	128	python	3.6.0	178	zlib	1.2.8	3
29	contextlib2	0.5.4	79	lxml	3.7.2	129	python-dateutil	2.6.0		py36_0		
30	cryptography	1.7.1	80	markupsafe	0.23	130	python.app	1.2		py36_4		
31	curl	7.52.1	81	matplotlib	2.0.0	131	pytz	2016.10		py36_0		
32	cycler	0.10.0	82	mistune	0.7.3	132	pyyaml	3.12		py36_0		
33	cython	0.25.2	83	mkl	2017.0.1	133	pymzq	16.0.2		py36_0		
34	cytoolz	0.8.2	84	mkl-service	1.1.2	134	qt	5.6.2		0		
35	dask	0.13.0	85	mpmath	0.19	135	qtawesome	0.4.3		py36_0		
36	datashape	0.5.4	86	multipledispatch	0.4.9	136	qtconsole	4.2.1		py36_1		
37	decorator	4.0.11	87	nbconvert	4.2.0	137	qtpy	1.2.1		py36_0		
38	dill	0.2.5	88	nbformat	4.2.0	138	readline	6.2		2		
39	docutils	0.13.1	89	networkx	1.11	139	redis	3.2.0		0		
40	entrypoints	0.2.2	90	nttk	3.2.2	140	redis-py	2.10.5		py36_0		
41	et_xmlfile	1.0.1	91	nose	1.3.7	141	requests	2.12.4		py36_0		
42	fastcache	1.0.2	92	notebook	4.3.1	142	rope	0.9.4		py36_1		
43	flask	0.12	93	numba	0.30.1	143	ruamel_yaml	0.11.14		py36_1		
44	flask-cors	3.0.2	94	numexpr	2.6.1	144	scikit-image	0.12.3		np111py36_1		
45	freetype	2.5.5	95	numpy	1.11.3	145	scikit-learn	0.18.1		np111py36_1		
46	get_terminal_size	1.0.0	96	numpydoc	0.6.0	146	scipy	0.18.1		np111py36_1		
47	gevent	1.2.1	97	odo	0.5.0	147	seaborn	0.7.1		py36_0		
48	greenlet	0.4.11	98	openpyxl	2.4.1	148	setuptools	27.2.0		py36_0		
49	h5py	2.6.0	99	openssl	1.0.2k	149	simplegeneric	0.8.1		py36_1		
50	hdf5	1.8.17	100	pandas	0.19.2	150	singledispatch	3.4.0.3		py36_0		

178
python
packages
included.

Anaconda-Navigator



Anaconda-Navigator

The screenshot displays the Anaconda Navigator application window. The title bar reads "Anaconda Navigator". The top navigation bar includes the "ANACONDA NAVIGATOR" logo, an "Upgrade Now" button, and a "Sign in to Anaconda Cloud" button. A left sidebar contains links to Home, Environments, Projects (beta), Learning, and Community, along with buttons for Documentation, Developer Blog, and Feedback, and social media icons for Twitter, YouTube, and GitHub. The main content area shows a grid of applications available on the "root" channel. A central dialog box is open, displaying a welcome message and a checkbox for helping improve Anaconda, which is checked. The "Ok, and don't show again" button is highlighted with a red dashed border. The background applications visible include jupyter notebook (4.3.1), spyder (3.1.2), anaconda-fusion (1.0.2), glueviz (0.9.1), and rstudio (1.0.136).

ANACONDA NAVIGATOR

Upgrade Now Sign in to Anaconda Cloud

Home Environments Projects (beta) Learning Community

Documentation Developer Blog Feedback

Twitter YouTube GitHub

Applications on root Channels Refresh

jupyter notebook 4.3.1
Web-based, interactive computing environment. Edit and run human-readable code while describing the data.

Launch

spyder 3.1.2
Python Development powerful Python IDE with debugging, interactive testing, and introspection features.

Launch

anaconda-fusion 1.0.2
Integration between Excel ® and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel.

Install

glueviz 0.9.1
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install

rstudio 1.0.136
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install

ANACONDA NAVIGATOR

Thanks for installing Anaconda!

Anaconda Navigator helps you easily start important Python applications and manage the packages in your local Anaconda installation. It also connects you to online resources for learning and engaging with the Python, SciPy, and PyData community.

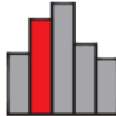
To help us improve Anaconda Navigator, fix bugs, and make it even easier for everyone to use Python, we gather anonymized usage information, just like most web browsers and mobile apps.

To opt out of this, please uncheck below (You can always change this setting in the Preferences menu).

☒ Yes, I'd like to help improve Anaconda.

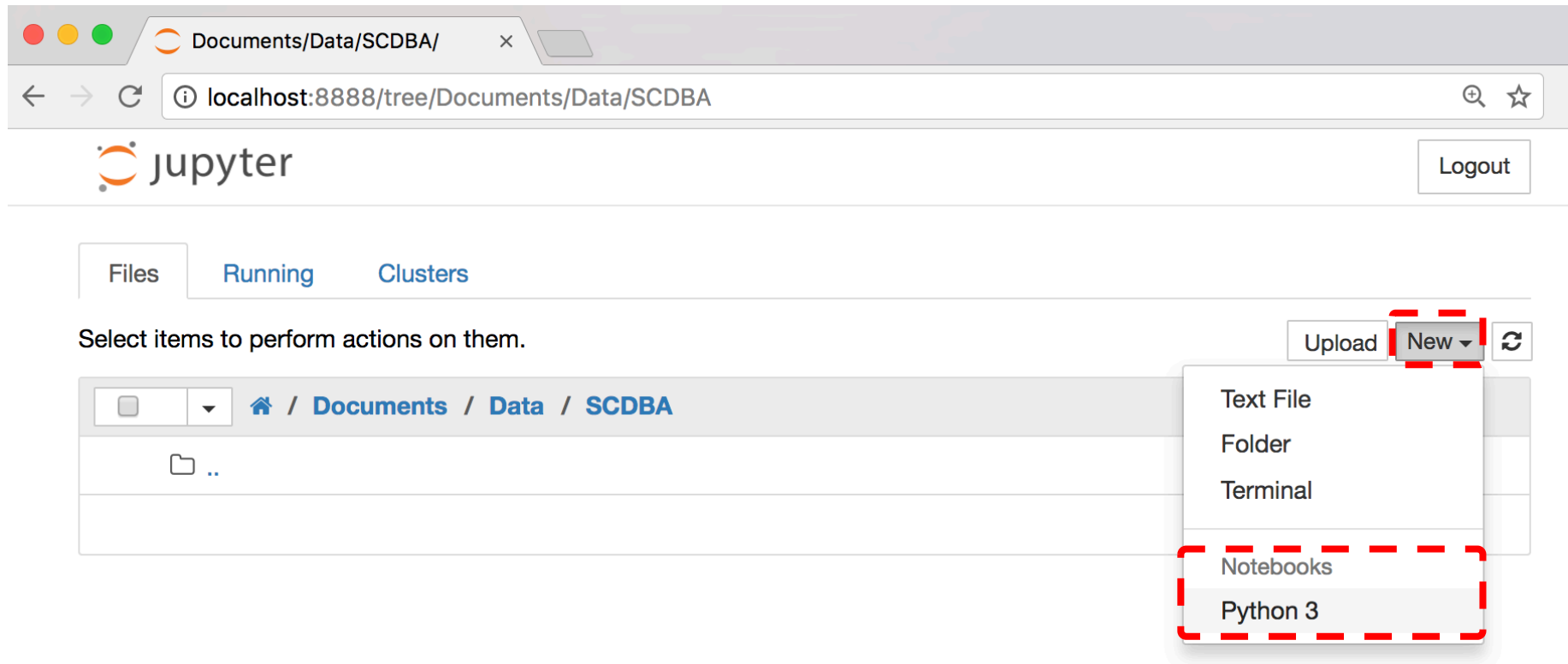
Ok Ok, and don't show again

Jupyter Notebook

 jupyter notebook 4.3.1 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch	 qtconsole 4.2.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch	 spyder 3.1.2 Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features Launch
 anaconda-fusion 1.0.2 Integration between Excel® and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel Install	 glueviz 0.9.1 Multidimensional data visualization across files. Explore relationships within and among related datasets. Install	 rstudio 1.0.136 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. Install

Jupyter Notebook

New Python 3



The screenshot shows the Jupyter Notebook web interface in a browser. The address bar indicates the URL is `localhost:8888/tree/Documents/Data/SCDBA`. The Jupyter logo and a 'Logout' button are at the top right. Below the navigation tabs (Files, Running, Clusters), there is a prompt 'Select items to perform actions on them.' and a file browser view showing the path `/ Documents / Data / SCDBA`. In the top right of the file browser, there are 'Upload' and 'New' buttons. The 'New' button is open, showing a dropdown menu with options: 'Text File', 'Folder', 'Terminal', 'Notebooks', and 'Python 3'. The 'Python 3' option is highlighted with a red dashed border, indicating it is the selected action.

Documents/Data/SCDBA/

localhost:8888/tree/Documents/Data/SCDBA

jupyter Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

Text File
Folder
Terminal
Notebooks
Python 3

```
print("hello, world")
```

The screenshot displays a web browser window with two tabs: 'Documents/Data/SCDBA/' and 'HelloWorld'. The address bar shows the URL 'localhost:8888/notebooks/Documents/Data/SCDBA/HelloWorld.ipynb'. The Jupyter interface includes a header with the 'jupyter' logo, the notebook title 'HelloWorld (autosaved)', a 'Logout' button, and a Python 3 kernel selector. A menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, moving up/down, running (highlighted with a red dashed box), and interrupting the kernel. The main area shows a code cell with the input 'In [1]: print("hello, world")' and the output 'hello, world'.

```
In [1]: print("hello, world")  
hello, world
```



```
from platform import python_version
print("Python Version:", python_version())
```

The screenshot shows a Jupyter Notebook browser interface. The address bar indicates the notebook is located at `localhost:8888/notebooks/Documents/Data/SCDBA/HelloWorld.ipynb`. The notebook title is "HelloWorld (autosaved)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu contains icons for saving, adding, deleting, copying, pasting, and running code. The "Run" button, represented by a right-pointing arrow, is highlighted with a red dashed box. Below the toolbar, there are two code cells. The first cell, labeled "In [1]:", contains the code `print("hello, world")` and has executed, showing the output "hello, world". The second cell, labeled "In [2]:", contains the code `from platform import python_version` followed by `print("Python Version:", python_version())` and has also executed, showing the output "Python Version: 3.6.0".

```
In [1]: print("hello, world")
hello, world

In [2]: from platform import python_version
print("Python Version:", python_version())
Python Version: 3.6.0
```

Create Python Environments with Anaconda

- Python 3.6
- Python 3.5
 - Python 3.5.3
 - Python 3.5.2
- Python 2.7

Anaconda Create New Python 3.5 Environment (py35)

The screenshot shows the Anaconda Navigator application window. The title bar reads "Anaconda Navigator". The main interface has a sidebar on the left with navigation options: Home, Environments (highlighted with a red dashed box), Projects (beta), Learning, and Community. Below these are links for Documentation, Developer Blog, and Feedback, and social media icons for Twitter, YouTube, and GitHub. The main panel displays the "Environments" tab, showing a list of installed packages. A "Create new environment" dialog box is open in the center, with the following details:

- Environment name:
- ☒ Python ☐ R
- Python version:
- Buttons: Cancel, Create

The background table lists installed packages with their names, descriptions, and versions. The "Create" button in the dialog and the "Environments" tab in the sidebar are highlighted with red dashed boxes. The text "py35 Python 3.5" is overlaid in red on the left side of the dialog.

Name	T	Description	Version
✓ _license	○		1.1
✓ _labstack	○	Configurable, python 2+3 compatible sphinx theme	0.7.9
			4.3.1
		and line client library	1.6.0
		table project directories	0.4.1
			0.1.0
			1.0.1
✓ astroid	○	Abstract syntax tree for python with inference support	1.4.9
✓ astropy	○	Community-developed python library for astronomy	1.3
✓ babel	○		2.3.4
✓ backports	○		1.0
✓ backports.shutil-get-terminal-size	○		1.0.0
✓ BeautifulSoup4	○	Python library designed for screen-scraping	4.5.3

186 packages available (root)

Anaconda Create New Python 2.7 Environment (py27)

The screenshot displays the Anaconda Navigator application window. The left sidebar contains navigation links: Home, Environments (highlighted with a red dashed box), Projects (beta), Learning, and Community. Below these are links for Documentation, Developer Blog, and Feedback, along with social media icons for Twitter, YouTube, and GitHub. The main panel is divided into two sections. The top section, also highlighted with a red dashed box, shows a list of environments: 'root' and 'py35'. The bottom section, also highlighted with a red dashed box, shows a list of installed packages for the 'py35' environment, including openssl, pip, python, readline, setuptools, sqlite, tk, wheel, xz, and zlib. A 'Create new environment' dialog box is open in the foreground, also highlighted with a red dashed box. It contains the following fields and options:

- Environment name:
- ☒ Python ☐ R
- Python version:
- Buttons: Cancel, Create

At the bottom of the main panel, it states '10 packages available (/Users/imyday/anaconda/envs/py35)'.

Verify that conda is installed, check current conda version

- **conda --version**
- Update conda to the current version
 - **conda update conda**

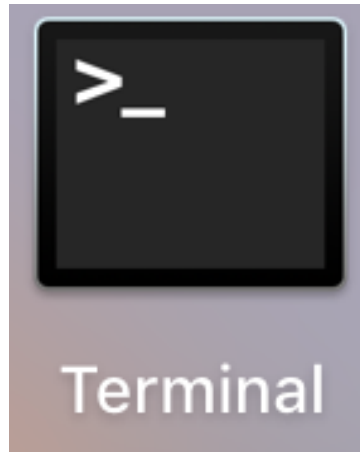
Check current conda version

Check current python version

Check conda environments

- **conda --version**
- **python --version**
- **conda info --envs**

Terminal



🔍 terminal|



terminal

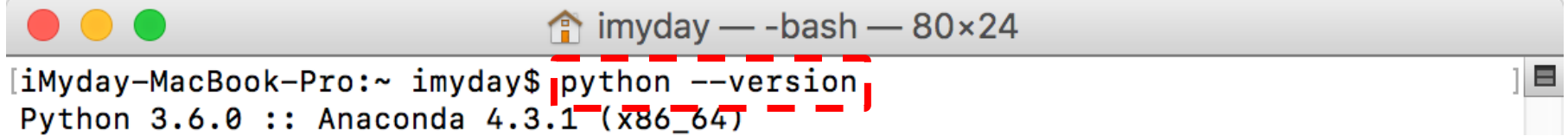
conda list

imyday — -bash — 80×24

```
[iMyday-MacBook-Pro:~ imyday$ conda list]
# packages in environment at /Users/imyday/anaconda:
```

```
#
_license                1.1                py36_1
alabaster                0.7.9                py36_0
anaconda                 4.3.1                np111py36_0
anaconda-client          1.6.0                py36_0
anaconda-navigator       1.5.0                py36_0
anaconda-project         0.4.1                py36_0
appnope                  0.1.0                py36_0
appscript                1.0.1                py36_0
astroid                  1.4.9                py36_0
astropy                  1.3                  np111py36_0
babel                    2.3.4                py36_0
backports                1.0                  py36_0
beautifulsoup4           4.5.3                py36_0
bitarray                 0.8.1                py36_0
blaze                    0.10.1               py36_0
bokeh                    0.12.4               py36_0
boto                     2.45.0               py36_0
bottleneck               1.2.0                np111py36_0
cffi                     1.9.1                py36_0
chardet                  2.3.0                py36_0
chest                    0.2.3                py36_0
```


python --version

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text 'imyday — -bash — 80x24' in the center, and a close button on the right. The terminal content shows the command 'python --version' being entered at the prompt 'iMyday-MacBook-Pro:~ imyday\$'. The output 'Python 3.6.0 :: Anaconda 4.3.1 (x86_64)' is displayed on the line below. The command and its output are highlighted with red dashed boxes. A vertical scrollbar is visible on the right side of the terminal window.

```
iMyday-MacBook-Pro:~ imyday$ python --version  
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
```

conda --version

```
imyday — -bash — 80x24
[iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
[iMyday-MacBook-Pro:~ imyday$ conda --version
conda 4.3.14
[iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
root          * /Users/imyday/anaconda

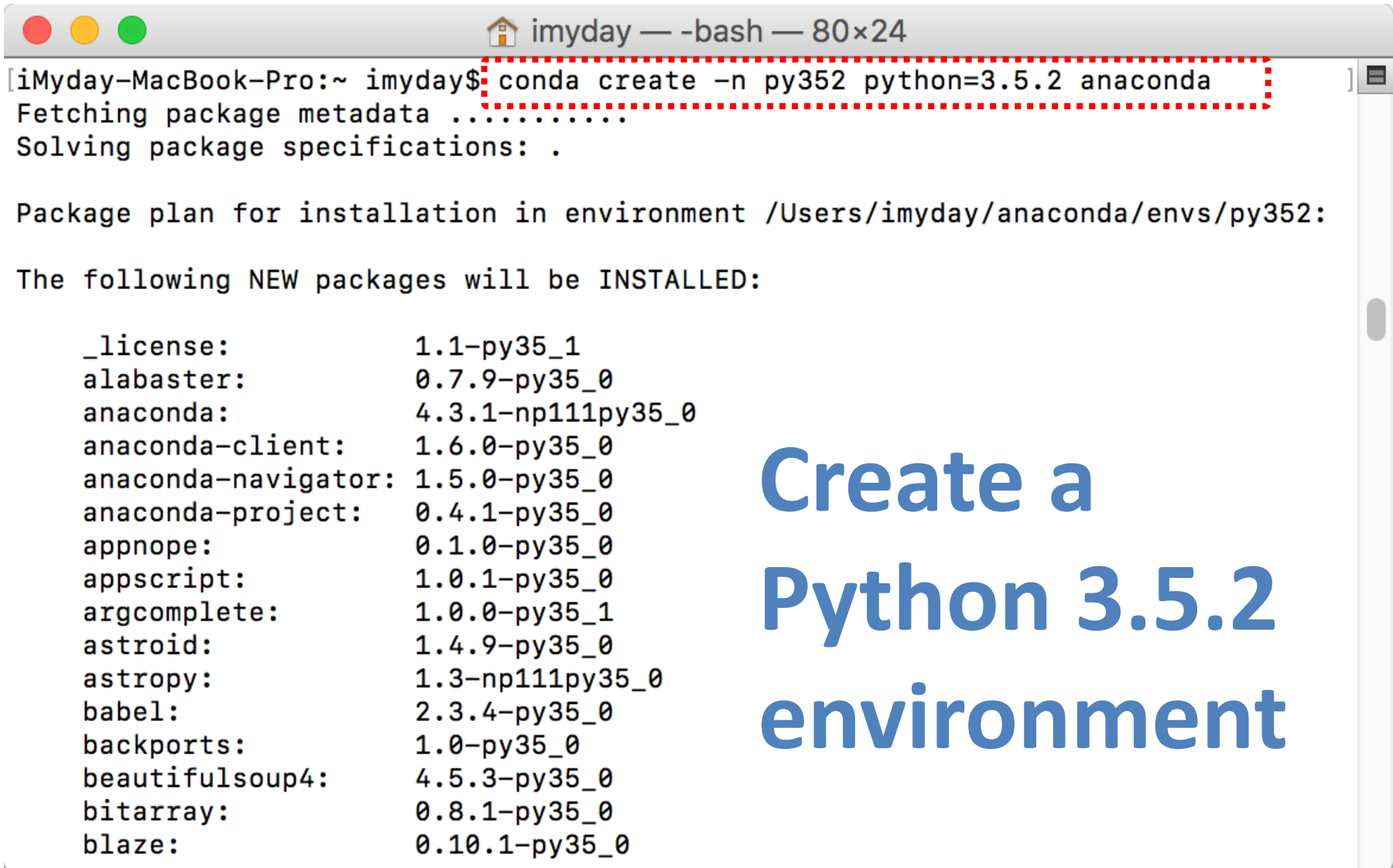
[iMyday-MacBook-Pro:~ imyday$ source activate py35
(py35) iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.5.3 :: Continuum Analytics, Inc.
(py35) iMyday-MacBook-Pro:~ imyday$ conda --version
conda 4.3.14
(py35) iMyday-MacBook-Pro:~ imyday$ source deactivate py35
[iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
root          * /Users/imyday/anaconda
```

python --version
conda --version
conda info --envs

source activate py35

source deactivate py35

```
conda create -n py352 python=3.5.2 anaconda
```



```
imyday — -bash — 80x24
[iMyday-MacBook-Pro:~ imyday$ conda create -n py352 python=3.5.2 anaconda
Fetching package metadata .....
Solving package specifications: .

Package plan for installation in environment /Users/imyday/anaconda/envs/py352:

The following NEW packages will be INSTALLED:

 _license:                1.1-py35_1
 alabaster:                0.7.9-py35_0
 anaconda:                4.3.1-np111py35_0
 anaconda-client:         1.6.0-py35_0
 anaconda-navigator:      1.5.0-py35_0
 anaconda-project:        0.4.1-py35_0
 appnope:                 0.1.0-py35_0
 appscript:               1.0.1-py35_0
 argcomplete:             1.0.0-py35_1
 astroid:                 1.4.9-py35_0
 astropy:                 1.3-np111py35_0
 babel:                   2.3.4-py35_0
 backports:               1.0-py35_0
 beautifulsoup4:          4.5.3-py35_0
 bitarray:                0.8.1-py35_0
 blaze:                   0.10.1-py35_0
```

Create a
Python 3.5.2
environment

```
conda create -n py352 python=3.5.2 anaconda
```

```

pyopenssl-16.2 100% |#####| Time: 0:00:00 1.40 MB/s
scikit-image-0 100% |#####| Time: 0:00:17 1.05 MB/s
seaborn-0.7.1- 100% |#####| Time: 0:00:00 1.05 MB/s
statsmodels-0. 100% |#####| Time: 0:00:04 1.06 MB/s
anaconda-navig 100% |#####| Time: 0:00:04 1.05 MB/s
blaze-0.10.1-p 100% |#####| Time: 0:00:00 1.05 MB/s
ipykernel-4.5. 100% |#####| Time: 0:00:00 1.21 MB/s
nbconvert-4.2. 100% |#####| Time: 0:00:00 1.22 MB/s
jupyter_consol 100% |#####| Time: 0:00:00 2.74 MB/s
notebook-4.3.1 100% |#####| Time: 0:00:05 1.05 MB/s
qtconsole-4.2. 100% |#####| Time: 0:00:00 1.03 MB/s
spyder-3.1.2-p 100% |#####| Time: 0:00:03 1.06 MB/s
widgetsnbexten 100% |#####| Time: 0:00:01 1.05 MB/s
ipywidgets-5.2 100% |#####| Time: 0:00:00 1.08 MB/s
jupyter-1.0.0- 100% |#####| Time: 0:00:00 2.53 MB/s
anaconda-4.3.1 100% |#####| Time: 0:00:00 4.49 MB/s
#
# To activate this environment, use:
# > source activate py352
#
# To deactivate this environment, use:
# > source deactivate py352
#
```

```
source activate py352
```

conda info --envs

```

[imyday-MacBook-Pro:~ imyday$ conda info --envs]
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
py352        /Users/imyday/anaconda/envs/py352
root          * /Users/imyday/anaconda

[imyday-MacBook-Pro:~ imyday$ python --version]
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)

[imyday-MacBook-Pro:~ imyday$ source activate py352]
(py352) imyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
py352        * /Users/imyday/anaconda/envs/py352
root          /Users/imyday/anaconda

[(py352) imyday-MacBook-Pro:~ imyday$ python --version]
Python 3.5.2 :: Anaconda 4.3.1 (x86_64)
(py352) imyday-MacBook-Pro:~ imyday$
```

TensorFlow

```
conda info --envs
```

```
conda --version
```

```
python --version
```

```
conda list
```

```
conda create -n tensorflow python=3.5
```

```
source activate tensorflow
```

```
activate tensorflow
```

```
sudo pip install tensorflow
```

```
pip install tensorflow
```

```
sudo pip install keras
```

```
pip install keras
```

```
pip install ipython[all]
```

```
jupyter notebook
```


`pip install tensorflow`

```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: six>=1.10.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Collecting protobuf>=3.2.0 (from tensorflow)
  Downloading protobuf-3.2.0-py2.py3-none-any.whl (360kB)
    100% |████████████████████████████████████████| 368kB 453kB/s
Requirement already satisfied: werkzeug>=0.11.10 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: numpy>=1.11.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: setuptools in ./anaconda/lib/python3.6/site-packages/setuptools-27.2.0-py3.6.egg (from protobuf>=3.2.0->tensorflow)
Installing collected packages: protobuf, tensorflow
Successfully installed protobuf-3.2.0 tensorflow-1.1.0
bash-3.2$
```

TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Iterations
000,582

Learning rate

0.03

Activation

Tanh

Regularization

None

Regularization rate

0

Problem type

Classification

DATA

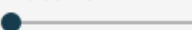
Which dataset do you want to use?



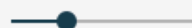
Ratio of training to test data: 50%



Noise: 0



Batch size: 10



INPUT

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$



3 HIDDEN LAYERS



4 neurons



2 neurons



2 neurons

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$

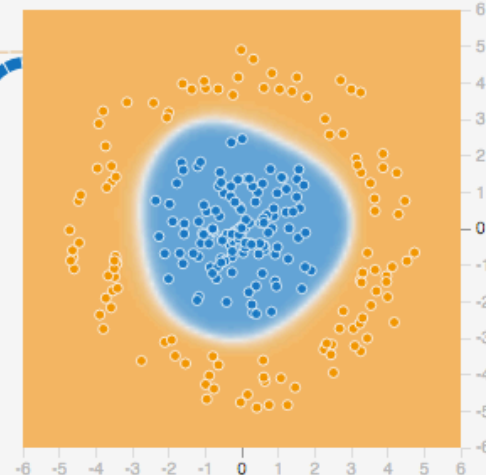
The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

OUTPUT

Test loss 0.000

Training loss 0.000



TensorBoard

TensorBoard

EVENTS

IMAGES

GRAPHS

HISTOGRAMS



Fit to screen



Download PNG

Run train
(1)

Session runs (0)

Upload

Color ☒ Structure
☐ Device
color: same substructure
gray: unique substructure

Graph (* = expandable)



Namespace*



OpNode



Unconnected series*



Connected series*



Constant



Summary



Dataflow edge

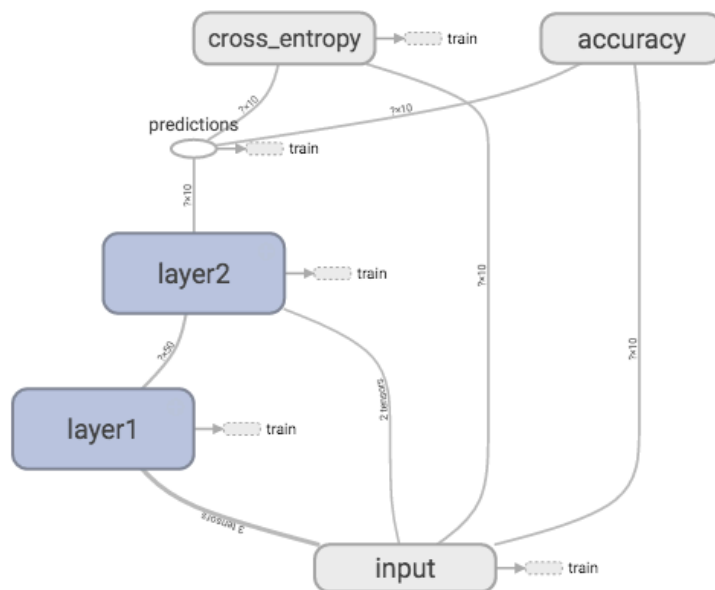


Control dependency edge

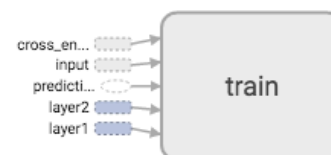


Reference edge

Main Graph



Auxiliary nodes



Getting Started with TensorFlow

Getting Started

[Getting Started With TensorFlow](#)

MNIST For ML Beginners

Deep MNIST for Experts

TensorFlow Mechanics 101

tf.estimator Quickstart

Building Input Functions with tf.estimator

TensorBoard: Visualizing Learning

TensorBoard: Graph Visualization

TensorBoard Histogram Dashboard

TensorFlow Versions

Getting Started With TensorFlow

This guide gets you started programming in TensorFlow. Before using this guide, [install TensorFlow](#). To get the most out of this guide, you should know the following:

- How to program in Python.
- At least a little bit about arrays.
- Ideally, something about machine learning. However, if you know little or nothing about machine learning, then this is still the first guide you should read.

TensorFlow provides multiple APIs. The lowest level API--TensorFlow Core-- provides you with complete programming control. We recommend TensorFlow Core for machine learning researchers and others who require fine levels of control over their models. The higher level APIs are built on top of TensorFlow Core. These higher level APIs are typically easier to learn and use than TensorFlow Core. In addition, the higher level APIs make repetitive tasks easier and more consistent between different users. A high-level API like tf.estimator helps you manage data sets, estimators, training and inference.

This guide begins with a tutorial on TensorFlow Core. Later, we demonstrate how to implement the same model in tf.estimator. Knowing TensorFlow Core principles will give you a great mental model of how things are working internally when you use the more compact higher level API.

Contents

TensorFlow Core tutorial

 Importing TensorFlow

 The Computational Graph

tf.train API

 Complete program

tf.estimator

 Basic usage

 A custom model

Next steps

Try your first TensorFlow

\$ **python**

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

Hello TensorFlow

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
sess.run(hello)
```

```
b'Hello, TensorFlow!'
```

tf.Session()
sess.run()

```
import tensorflow as tf
sess = tf.Session()
a = tf.constant(10)
b = tf.constant(32)
sess.run(a+b)
```

42

Linear Regression Model

```
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-0.3], dtype=tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W*x + b
y = tf.placeholder(tf.float32)

# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

# training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x: x_train, y: y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x: x_train, y: y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```

W: [-0.9999969] b: [0.99999082] loss: 5.69997e-11

tf.estimator

```
import numpy as np
import tensorflow as tf

feature_columns = [tf.feature_column.numeric_column("x", shape=[1])]

estimator = tf.estimator.LinearRegressor(feature_columns=feature_columns)

x_train = np.array([1., 2., 3., 4.])
y_train = np.array([0., -1., -2., -3.])
x_eval = np.array([2., 5., 8., 1.])
y_eval = np.array([-1.01, -4.1, -7, 0.])
input_fn = tf.estimator.inputs.numpy_input_fn(
    {"x": x_train}, y_train, batch_size=4, num_epochs=None, shuffle=True)
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    {"x": x_train}, y_train, batch_size=4, num_epochs=1000, shuffle=False)
eval_input_fn = tf.estimator.inputs.numpy_input_fn(
    {"x": x_eval}, y_eval, batch_size=4, num_epochs=1000, shuffle=False)

estimator.train(input_fn=input_fn, steps=1000)

train_metrics = estimator.evaluate(input_fn=train_input_fn)
eval_metrics = estimator.evaluate(input_fn=eval_input_fn)
print("train metrics: %r"% train_metrics)
print("eval metrics: %r"% eval_metrics)
```

```
train metrics: {'average_loss': 2.7210228e-07, 'loss': 1.0884091e-06, 'global_step': 1000}
eval metrics: {'average_loss': 0.0025725411, 'loss': 0.010290165, 'global_step': 1000}
```

TensorFlow and Deep Learning

TensorFlow and Deep Learning

1 Overview

Preparation: Install

2 TensorFlow, get the sample code

3 Theory: train a neural network

4 Theory: a 1-layer neural network

5 Theory: gradient descent

6 Lab: let's jump into the code

7 Lab: adding layers

8 Lab: special care for deep networks

9 Lab: learning rate decay

10 Lab: dropout, overfitting

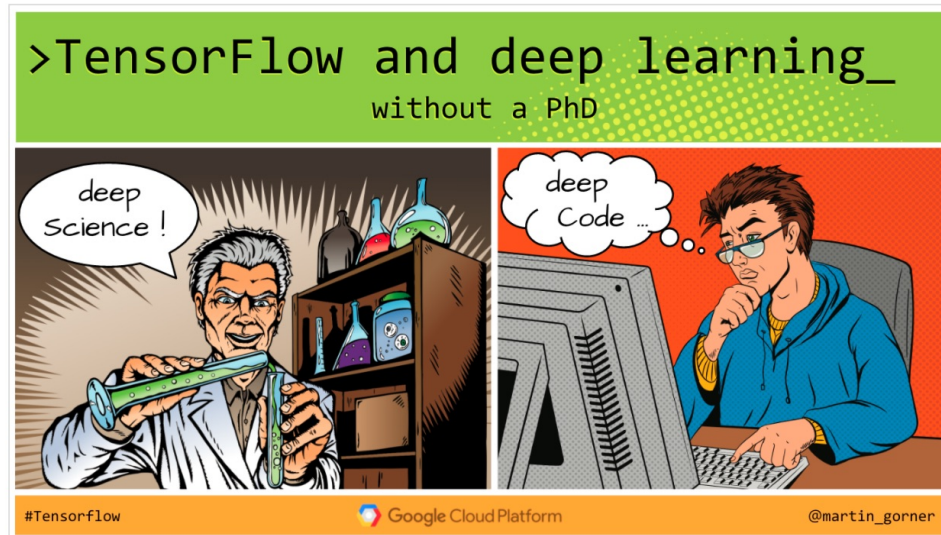
11 Theory: convolutional networks

Did you find a mistake? [Please file a bug.](#)

← TensorFlow and deep learning, without a PhD

🕒 149 min remaining

1. Overview



In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

This codelab uses the [MNIST](#) dataset, a collection of 60,000 labeled digits that has kept generations of PhDs busy for almost two decades. You will solve the problem with less than 100 lines of Python / TensorFlow code.

What you'll learn

TensorFlow MNIST Tutorial



Features Business Explore Pricing

This repository

Search

[Sign in](#) or [Sign up](#)

[martin-gorner](#) / [tensorflow-mnist-tutorial](#)

Watch

50

Star

489

Fork

204

Code

Issues **6**

Pull requests **2**

Projects **0**

Pulse

Graphs

Sample code for "Tensorflow and deep learning, without a PhD" presentation and code lab.

102 commits

1 branch

0 releases

4 contributors

Apache-2.0

Branch: **master** ▼

[New pull request](#)

[Find file](#)

[Clone or download](#) ▼



martin-gorner committed on **GitHub** Update INSTALL.txt ...

Latest commit ed331aa 25 days ago

mlengine	added example using the Tensorflow high level layers API	26 days ago
.gitignore	small bug fix in batch norm	6 months ago
CONTRIBUTING.md	initial commit 2	4 months ago
INSTALL.txt	Update INSTALL.txt	25 days ago
LICENSE	Initial commit	a year ago
README.md	better image URL	3 months ago
mnist_1.0_softmax.py	global_variables_initializer used everywhere instead of initalize_al...	2 months ago
mnist_2.0_five_layers_sigmoid.py	Fix spacing in the network structure comment	a month ago
mnist_2.1_five_layers_relu_lrdecay...	Fix spacing in the network structure comment	a month ago

TensorFlow and Deep Learning

- What is a neural network and how to train it
- How to build a basic 1-layer neural network using TensorFlow
- How to add more layers
- Training tips and tricks: overfitting, dropout, learning rate decay ...
- How to troubleshoot deep neural networks
- How to build convolutional networks

TensorFlow MNIST Tutorial

```
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git
```

```
cd tensorflow-mnist-tutorial
```

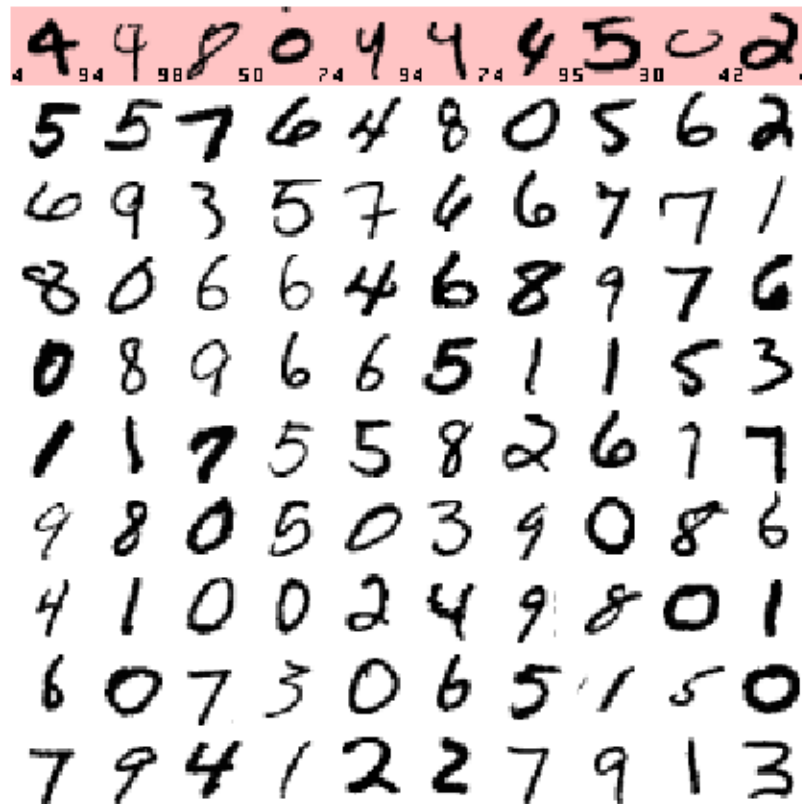
```
python3 mnist_1.0_softmax.py
```

```
python mnist_1.0_softmax.py
```

```
pythonw mnist_1.0_softmax.py
```

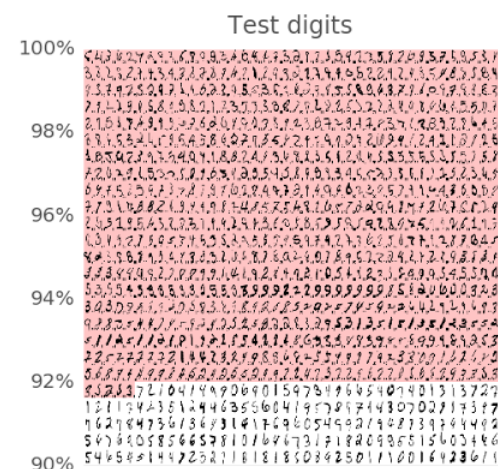
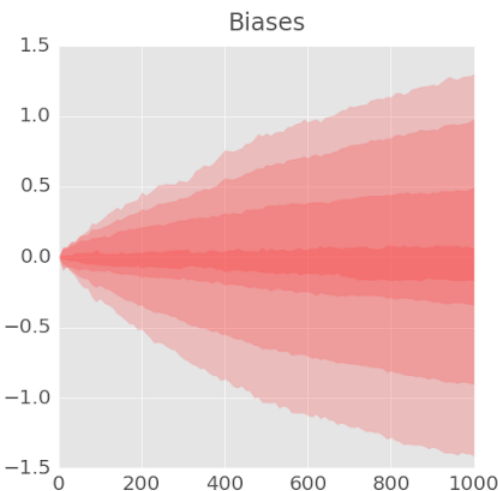
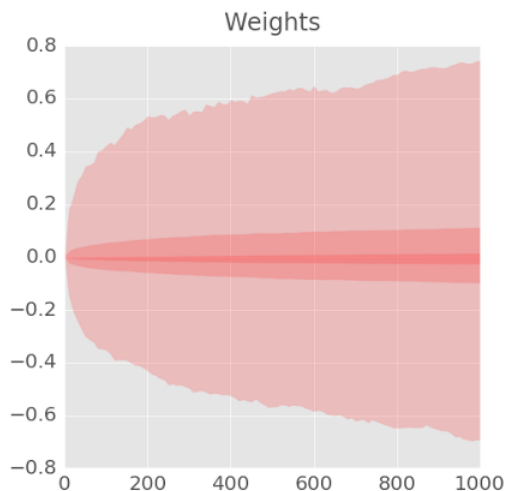
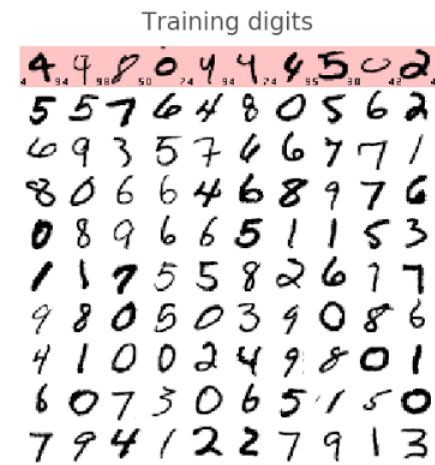
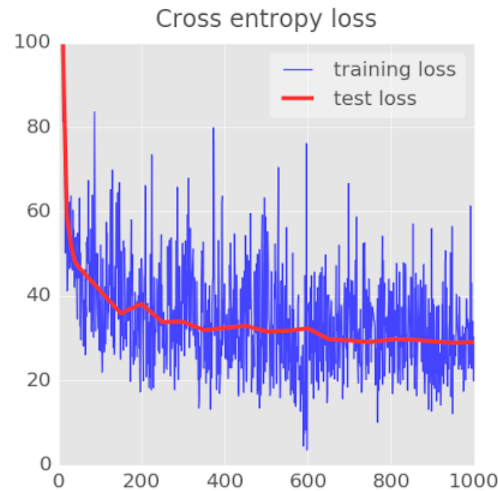
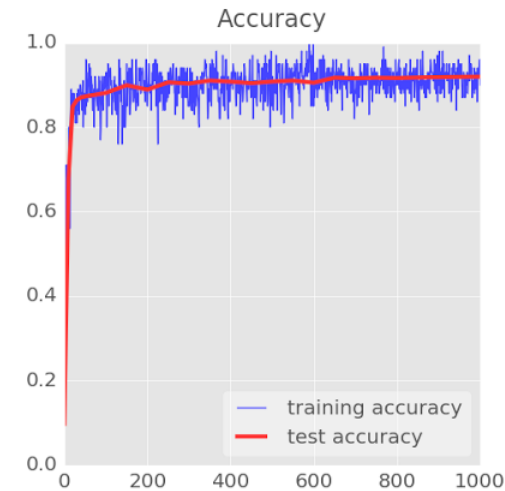
MNIST dataset: 60,000 labeled digits

Training digits



cd tensorflow-mnist-tutorial

python3 mnist_1.0_softmax.py

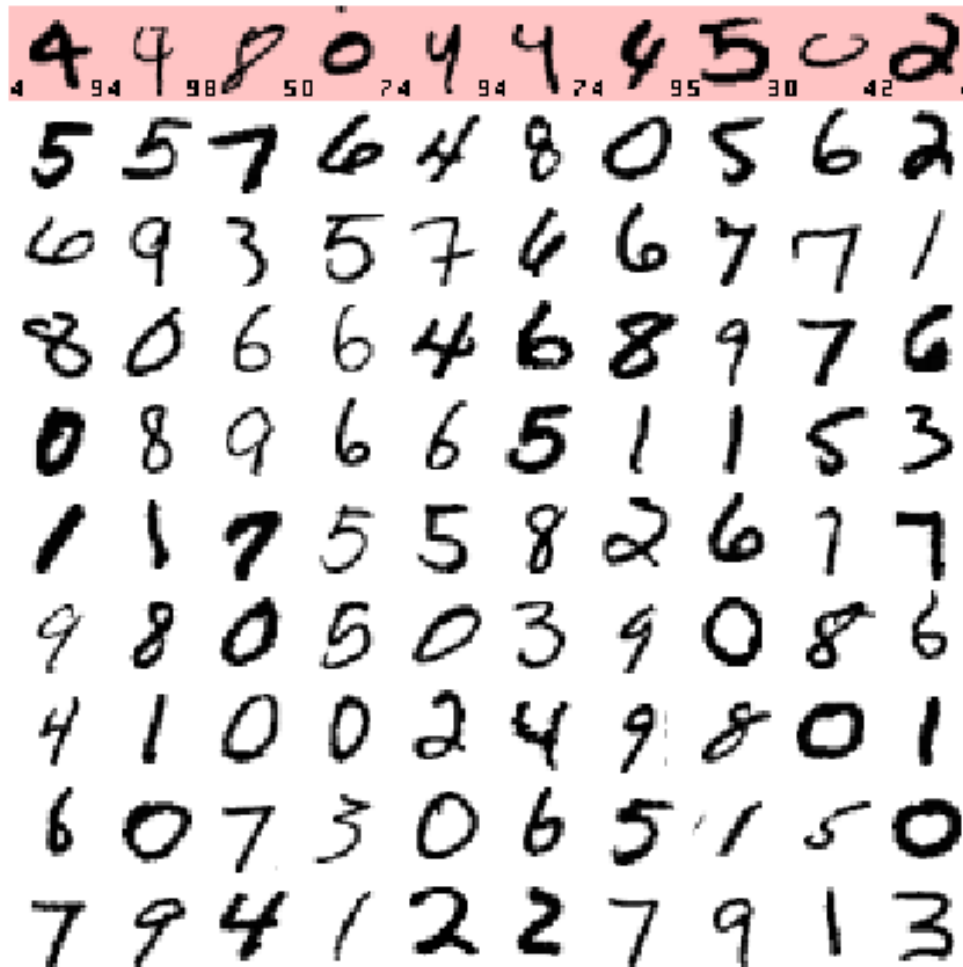


Train a Neural Network

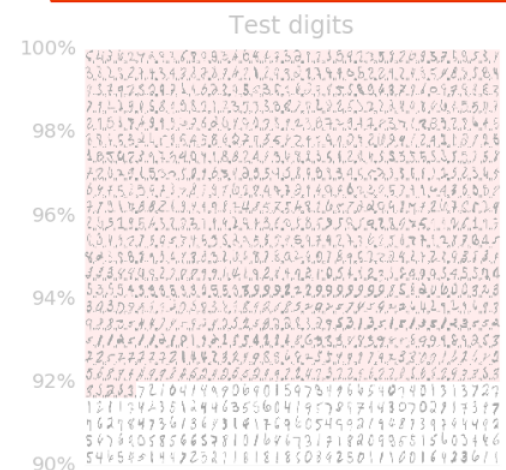
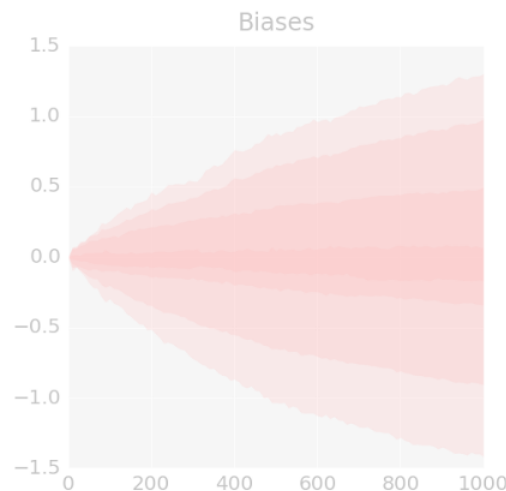
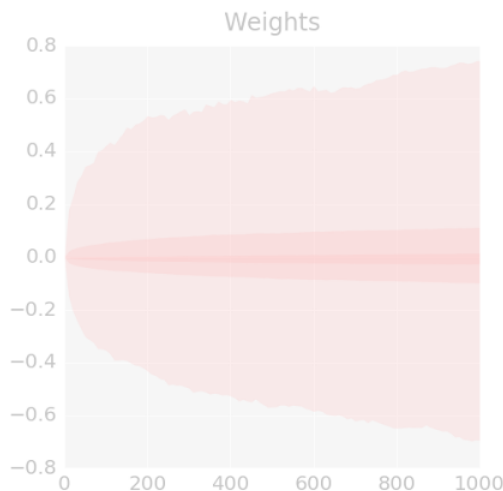
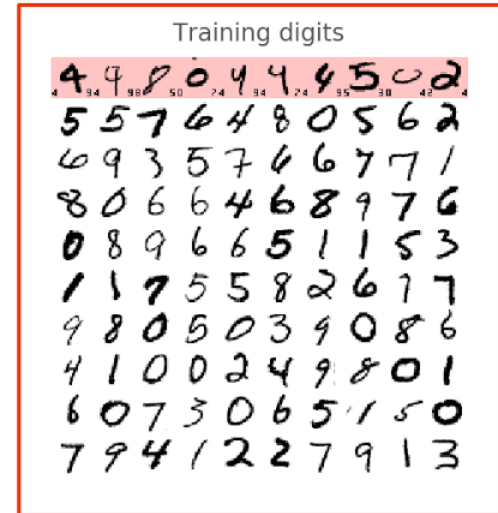
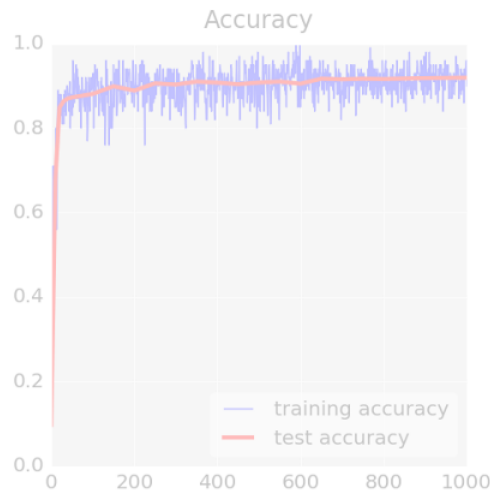
Training digits
updates to **weights** and **biases** =>
better recognition (loop)

Training digits

Training digits



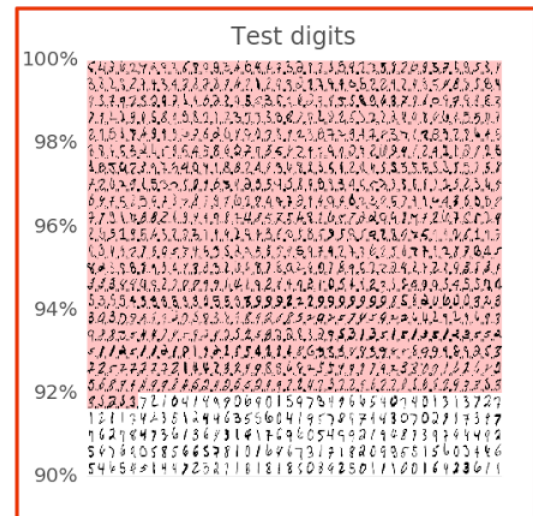
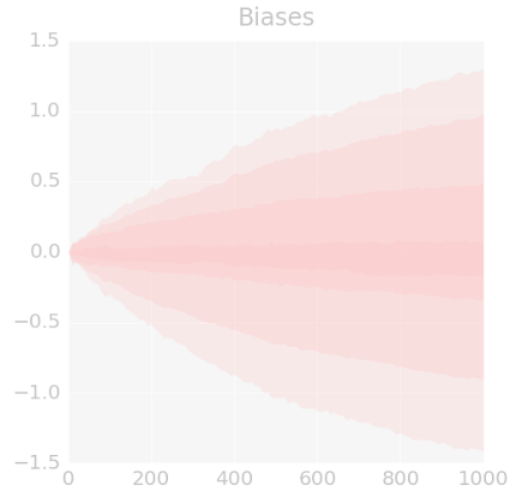
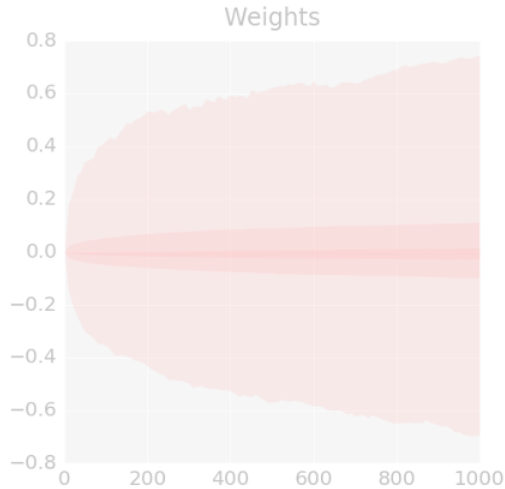
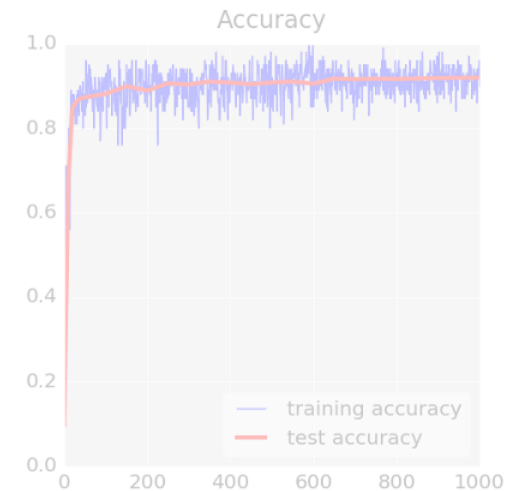
Training digits



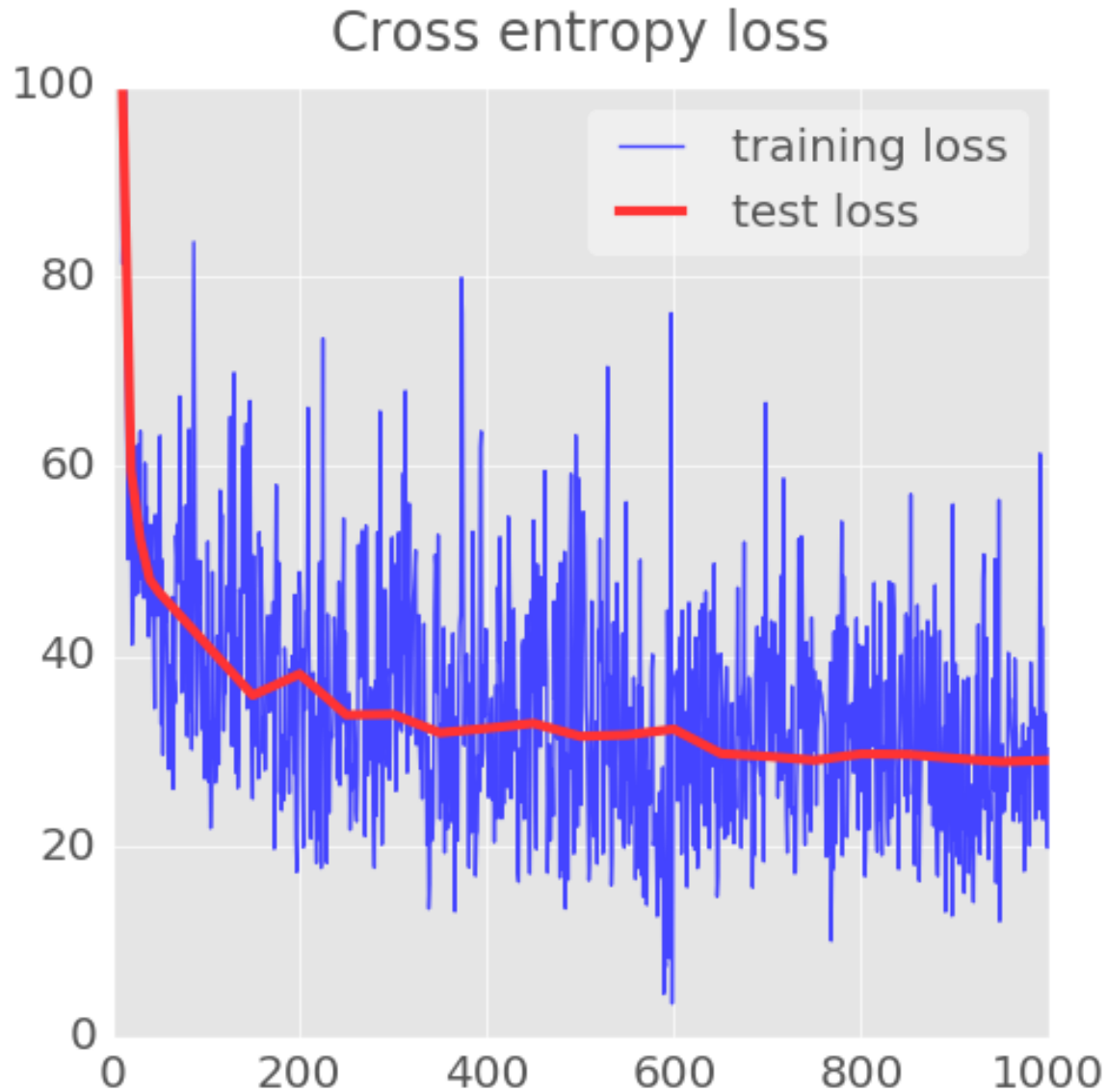
Test digits



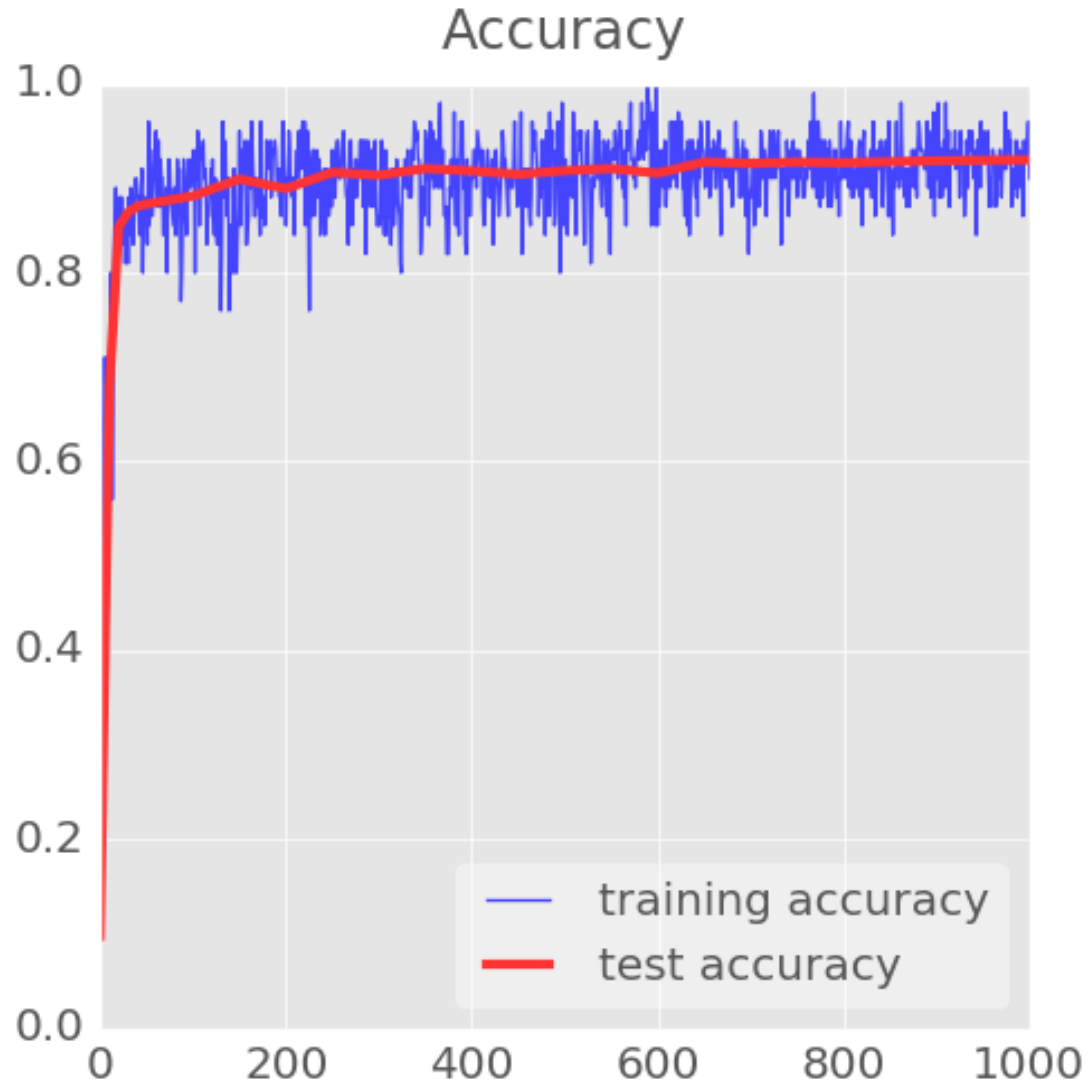
Test digits



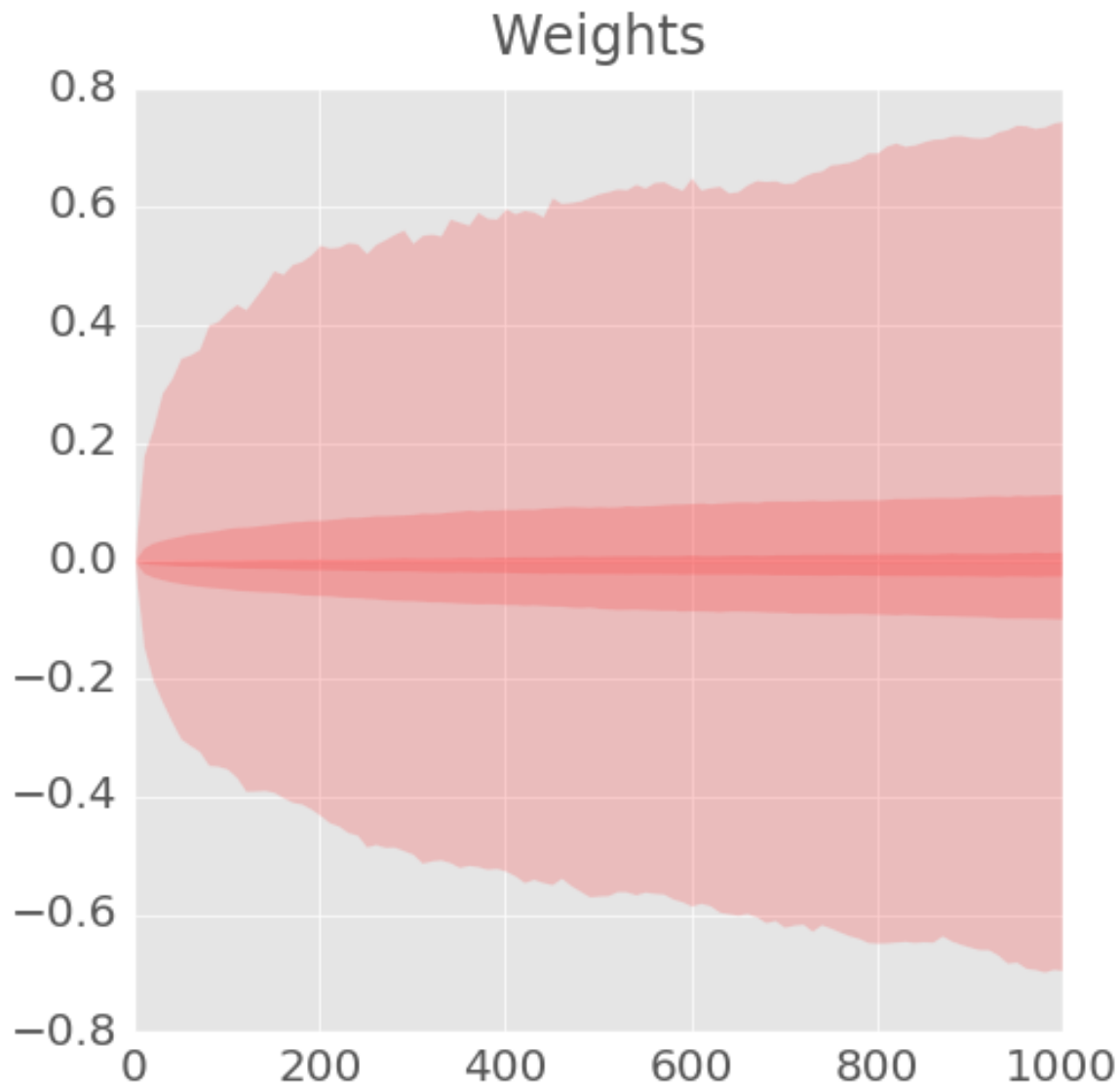
Cross entropy loss



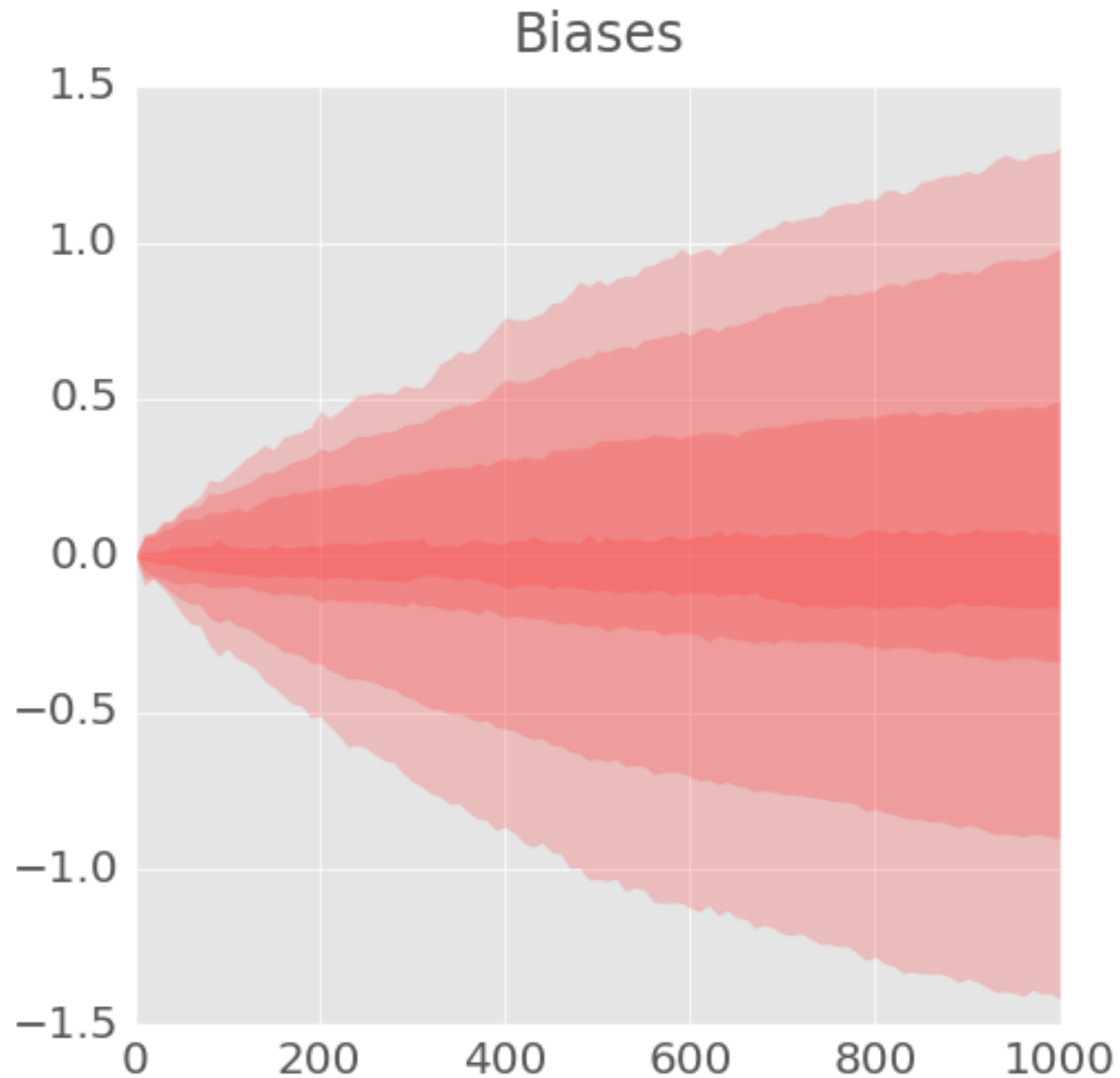
Accuracy



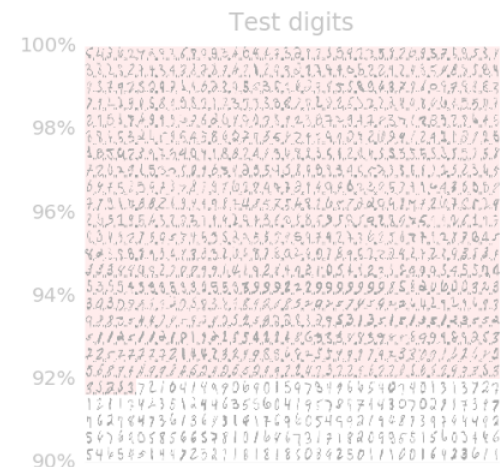
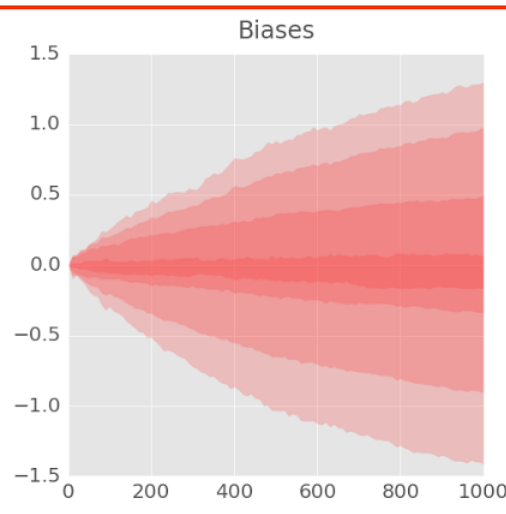
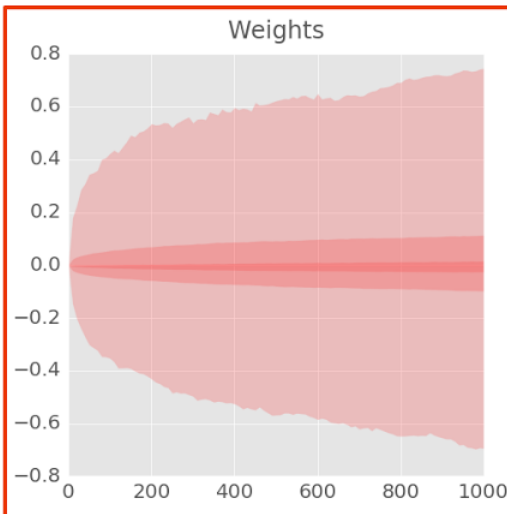
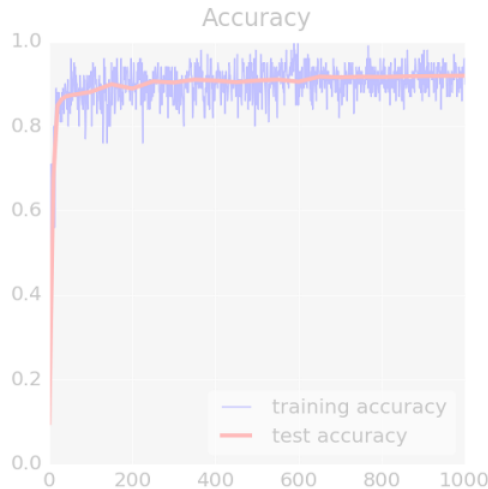
Weights



Biases



Weights and Biases

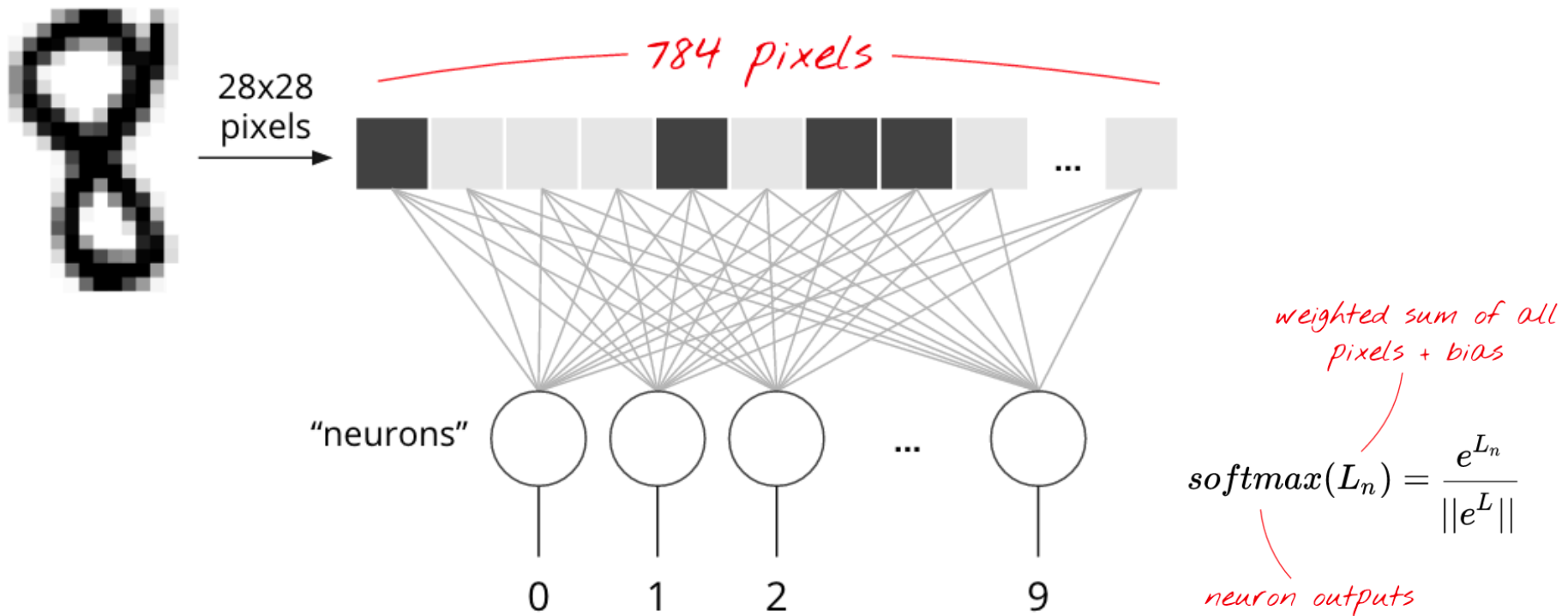


Cookbook

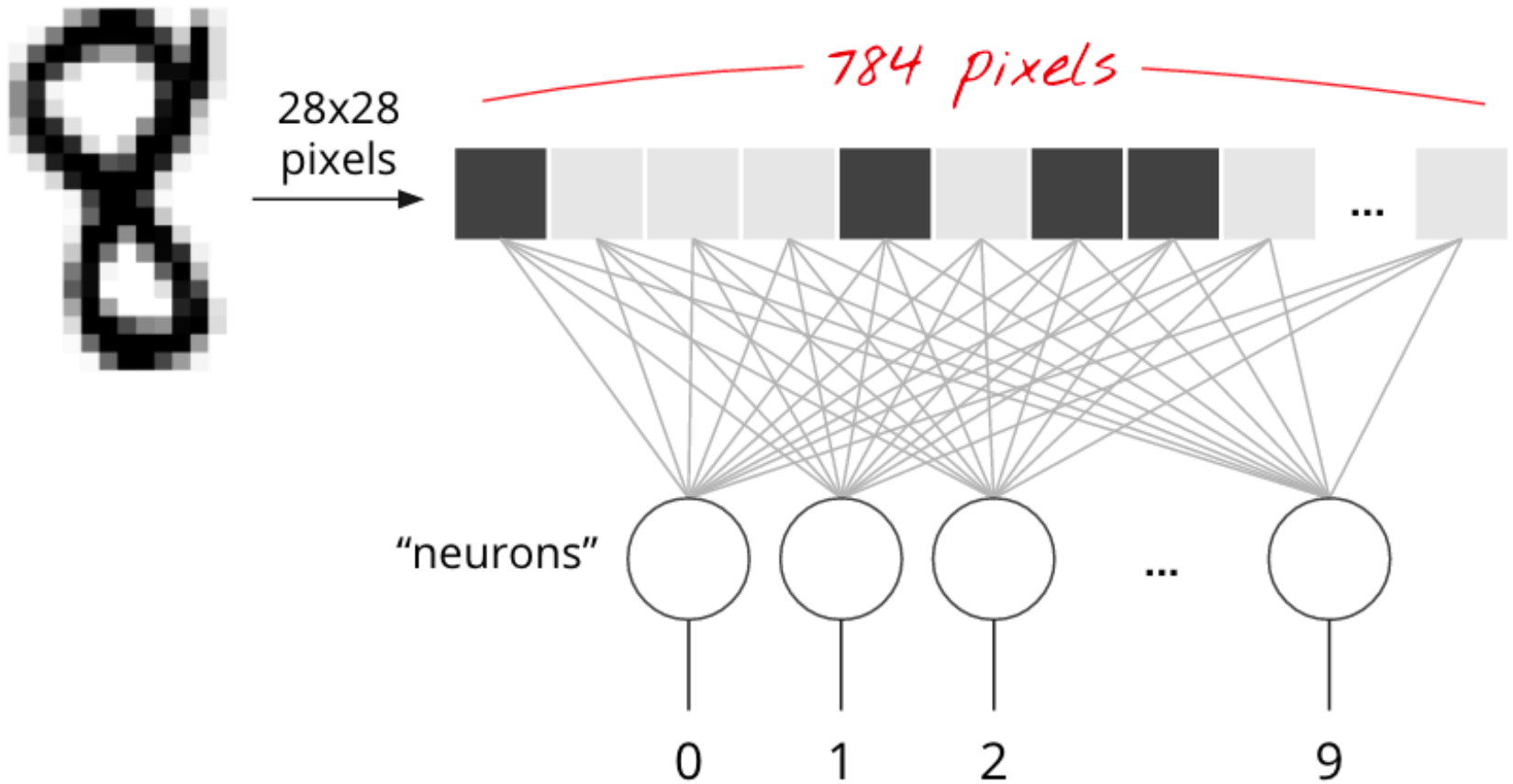
Softmax
Cross-entropy
Mini-batch



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification

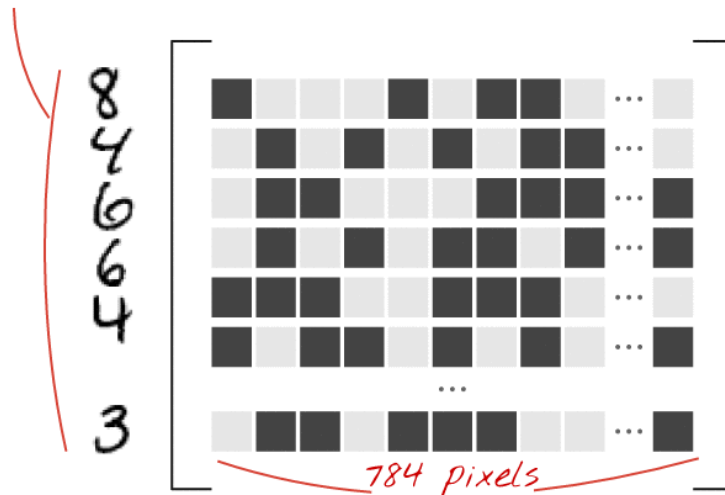
*weighted sum of all
pixels + bias*

$$\text{softmax}(L_n) = \frac{e^{L_n}}{||e^L||}$$

neuron outputs

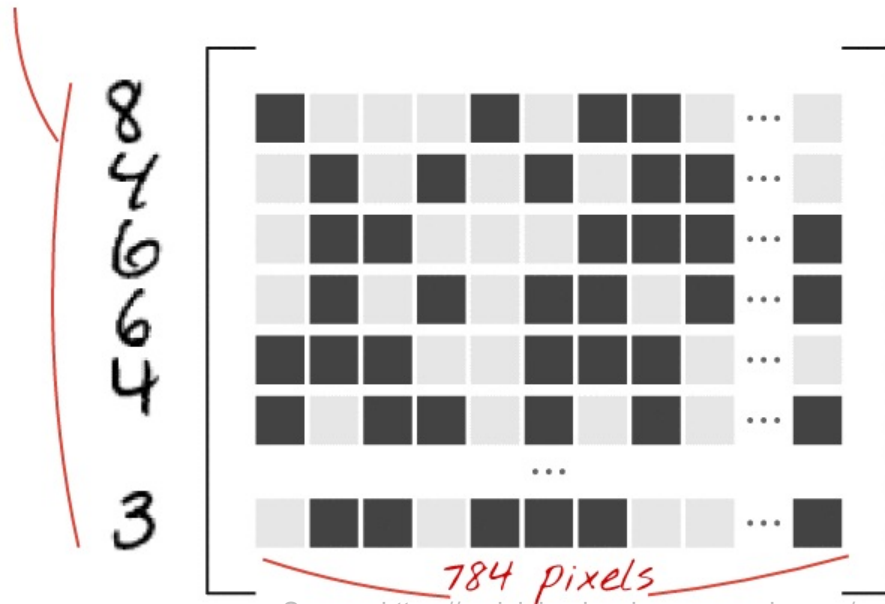
In Matrix notation, 100 images at a time

*X: 100 images,
one per line,
flattened*



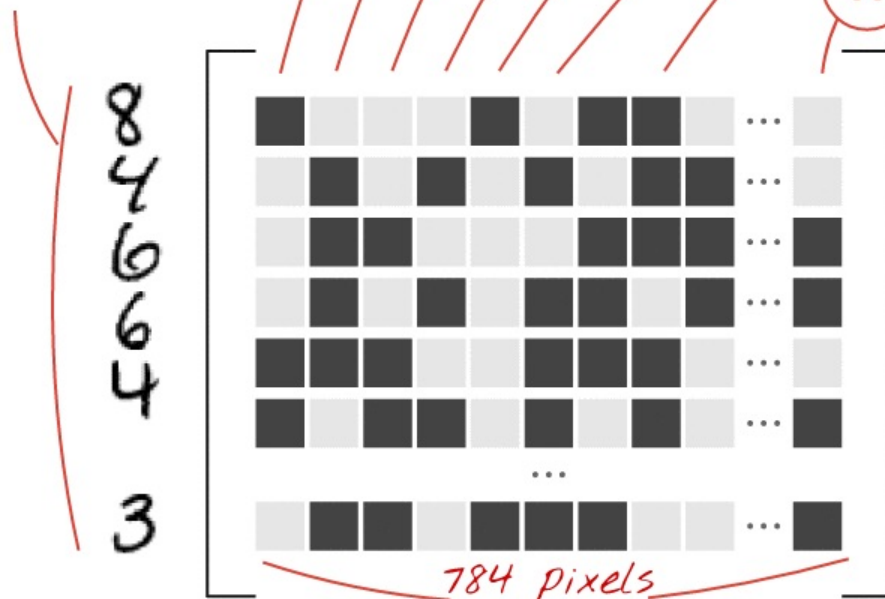
<i>10 columns</i>										<i>784 lines</i>
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	...	$W_{0,9}$					
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	...	$W_{1,9}$					
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	...	$W_{2,9}$					
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	...	$W_{3,9}$					
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	...	$W_{4,9}$					
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	...	$W_{5,9}$					
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	...	$W_{6,9}$					
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	...	$W_{7,9}$					
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	...	$W_{8,9}$					
...										
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	...	$W_{783,9}$						

*X: 100 images,
one per line,
flattened*



10 columns													
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	...	$W_{0,9}$	784 lines							
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	...	$W_{1,9}$								
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	...	$W_{2,9}$								
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	...	$W_{3,9}$								
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	...	$W_{4,9}$								
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	...	$W_{5,9}$								
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	...	$W_{6,9}$								
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	...	$W_{7,9}$								
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	...	$W_{8,9}$								
...													
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	...	$W_{783,9}$									

*X: 100 images,
one per line,
flattened*



10 columns

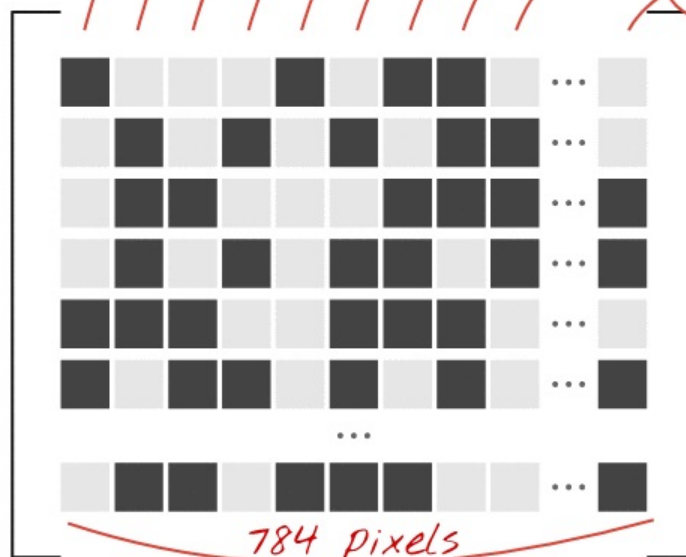
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	...	$W_{8,9}$
...					
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	...		$W_{783,9}$

784 lines

$L_{0,0}$

*X: 100 images,
one per line,
flattened*

8
4
6
6
4
3



10 columns

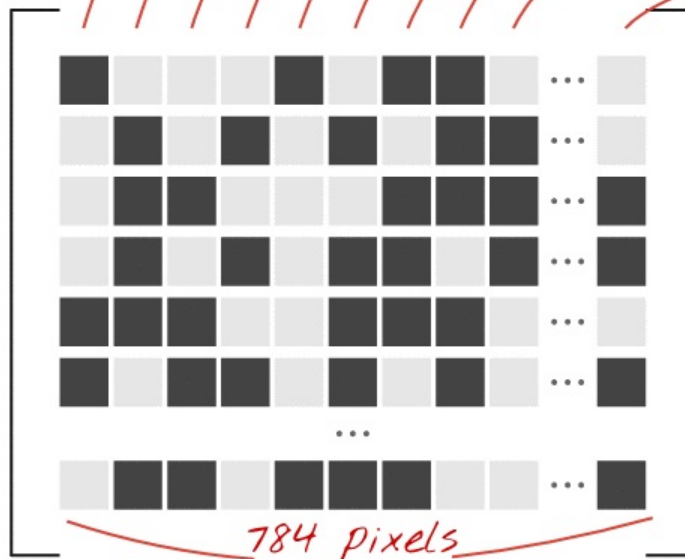
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	...	$W_{8,9}$
...	...	...	...	...	...
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	...	...	$W_{783,9}$

784 lines

$L_{0,0}$ $L_{0,1}$

*X: 100 images,
one per line,
flattened*

8
4
6
6
4
3



10 columns

$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	...	$W_{8,9}$
...	...	...	...	...	...
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	...	...	$W_{783,9}$

784 lines

$L_{0,0}$	$L_{0,1}$	$L_{0,2}$	$L_{0,3}$	...	$L_{0,9}$
$L_{1,0}$	$L_{1,1}$	$L_{1,2}$	$L_{1,3}$	...	$L_{1,9}$
$L_{2,0}$	$L_{2,1}$	$L_{2,2}$	$L_{2,3}$	...	$L_{2,9}$
$L_{3,0}$	$L_{3,1}$	$L_{3,2}$	$L_{3,3}$	...	$L_{3,9}$
$L_{4,0}$	$L_{4,1}$	$L_{4,2}$	$L_{4,3}$	...	$L_{4,9}$
...	...	...	...	...	...
$L_{99,0}$	$L_{99,1}$	$L_{99,2}$	...	...	$L_{99,9}$

What are "**weights**" and "**biases**" ?

How is the "**cross-entropy**"
computed ?

How exactly does the
training algorithm work ?

$$Y = f(X)$$

Predictions

$Y[100, 10]$

Images

$X[100, 784]$

Weights

$W[784, 10]$

Biases

$b[10]$

$$Y = \text{softmax}(X.W + b)$$

applied line
by line

matrix multiply

broadcast
on all lines

tensor shapes in []

$$Y = \text{tf.nn.softmax}(\text{tf.matmul}(X, W) + b)$$

TensorFlow (Python) Softmax

Predictions:

Y[100, 10]

tensor shapes: X[100, 784] W[784, 10] b[10]

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

matrix multiply

broadcast
on all lines

Cross Entropy

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

actual probabilities, "one-hot" encoded

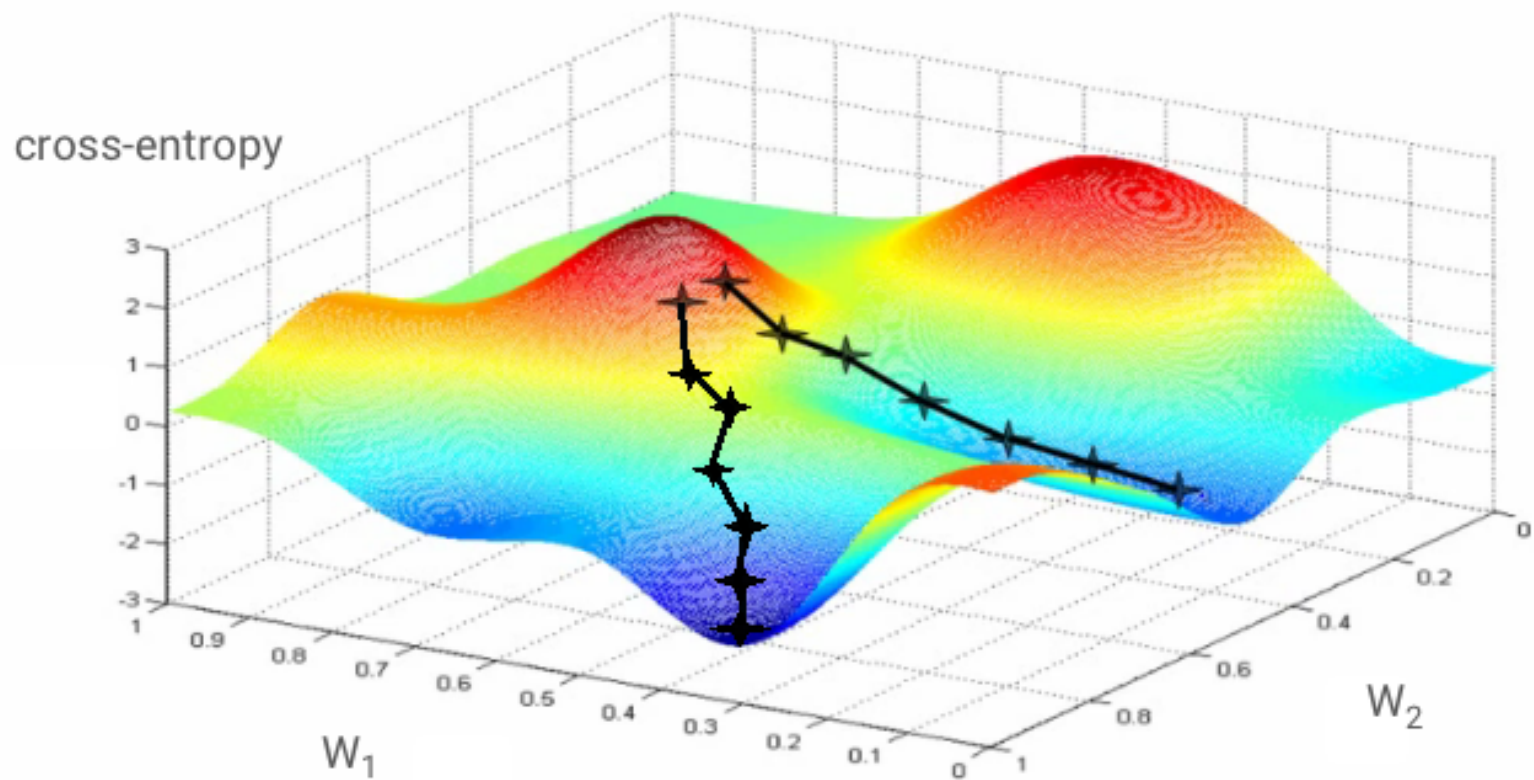
Cross entropy: $-\sum Y'_i \cdot \log(Y_i)$

computed probabilities

0.1	0.2	0.1	0.3	0.2	0.1	0.9	0.2	0.1	0.1
0	1	2	3	4	5	6	7	8	9

this is a "6"

Minimizing Cross Entropy (Minimizing Loss)



Training Loop

Training digits and labels

=> loss function

=> gradient (partial derivatives)

=> steepest descent

=> update weights and biases

=> repeat with next mini-batch of training images and labels

**"mini-batches":
100 images and labels**

```
import tensorflow as tf
```

mnist_1.0_softmax.py

```
import tensorflow as tf
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
```

mnist_1.0_softmax.py

```
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)
# placeholder for correct labels
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))
# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y, 1), tf.argmax(Y_, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

mnist_1.0_softmax.py

```
sess = tf.Session()
sess.run(init)

for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

    # train
    sess.run(train_step, feed_dict=train_data)
```

mnist_1.0_softmax.py

```
# success ?
```

```
a,c = sess.run([accuracy, cross_entropy],  
feed_dict=train_data)
```

```
# success on test data ?
```

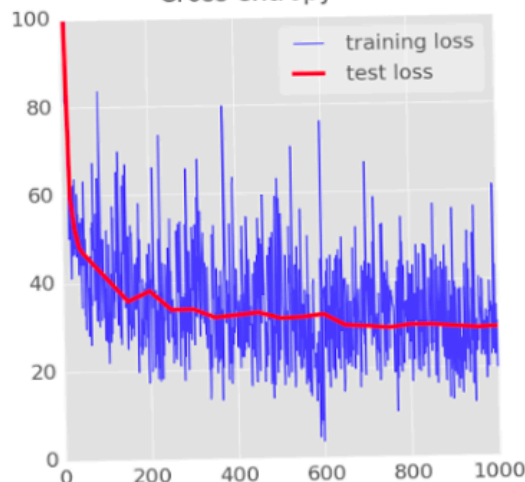
```
test_data={X: mnist.test.images, Y_: mnist.test.labels}  
a,c = sess.run([accuracy, cross_entropy], feed=test_data)
```

mnist_1.0_softmax.py

Accuracy



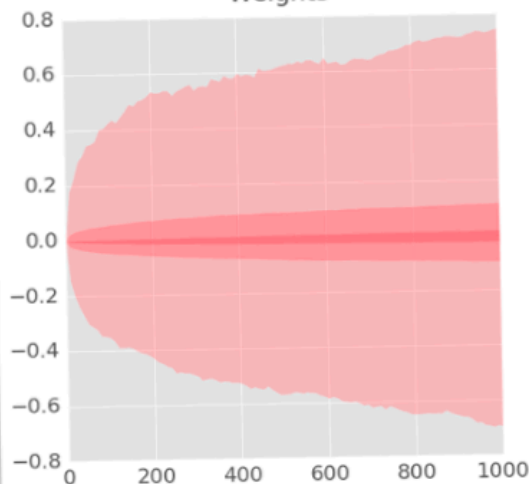
Cross entropy loss



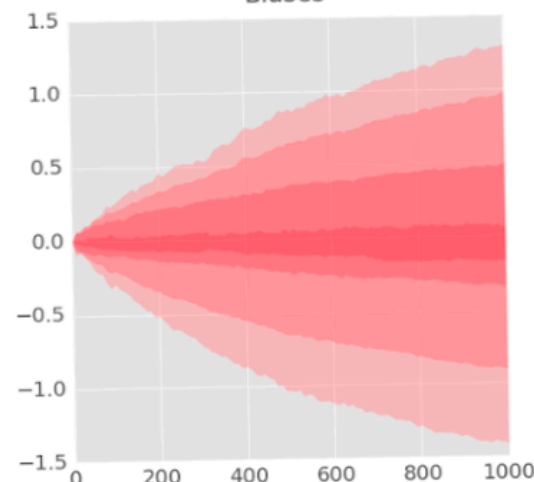
Training digits

4 4 8 0 4 4 4 5 0 2
5 5 7 6 4 8 0 5 6 2
6 9 3 5 7 6 6 7 7 1
8 0 6 6 4 6 8 9 7 6
0 8 9 6 6 5 1 1 5 3
1 1 7 5 5 8 2 6 7 7
9 8 0 5 0 3 9 0 8 6
4 1 0 0 2 4 9 8 0 1
6 0 7 3 0 6 5 1 5 0
7 9 4 1 2 2 7 9 1 3

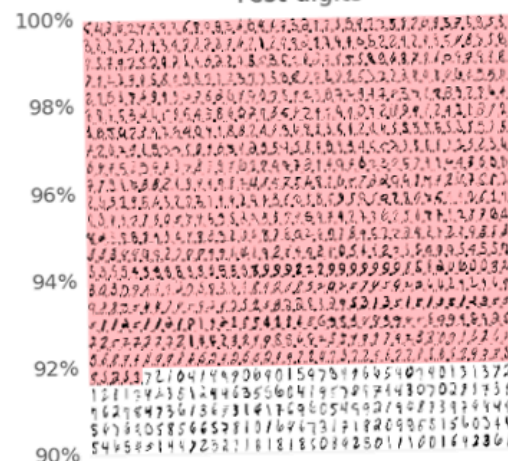
Weights



Biases



Test digits

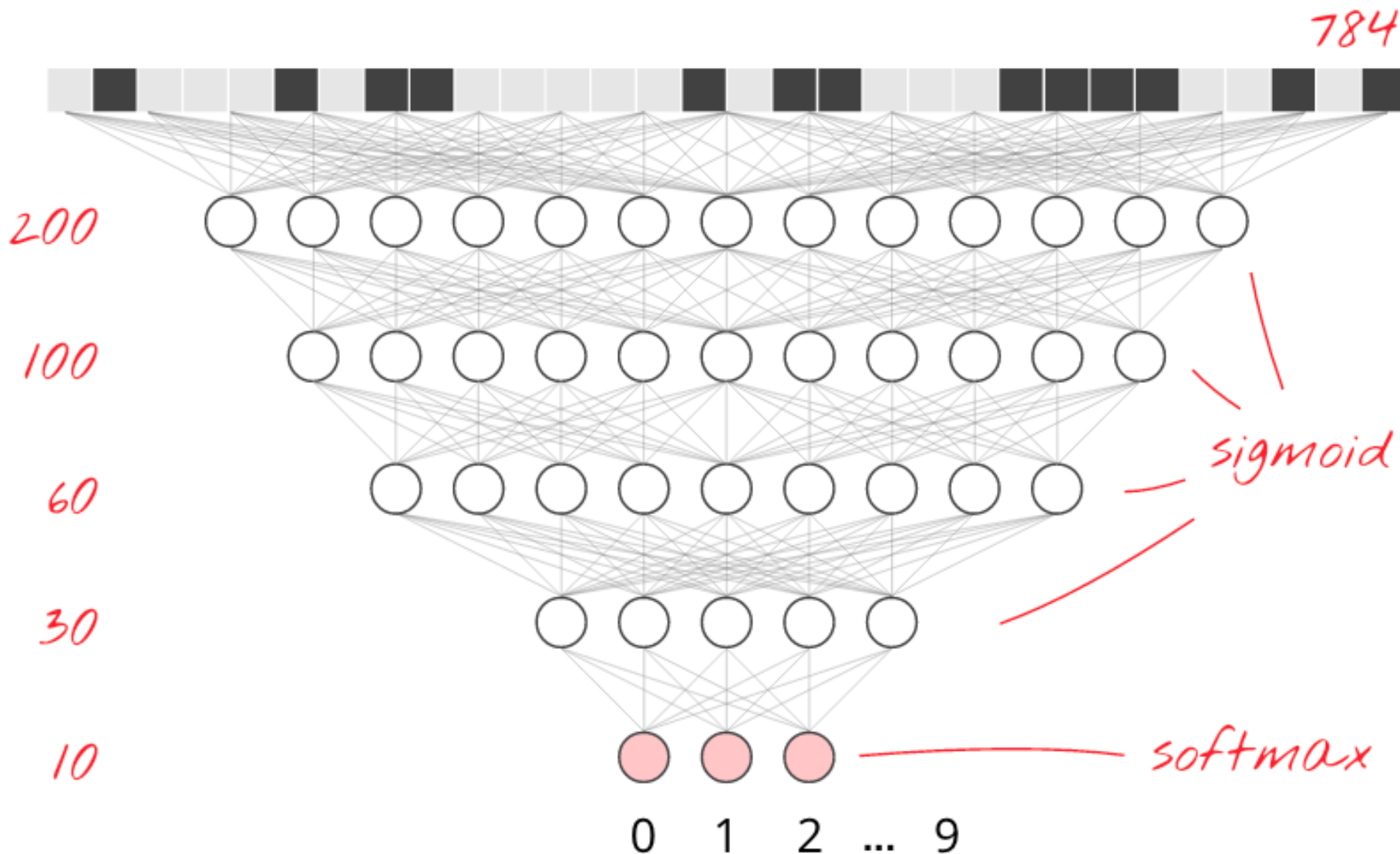


92!
😊

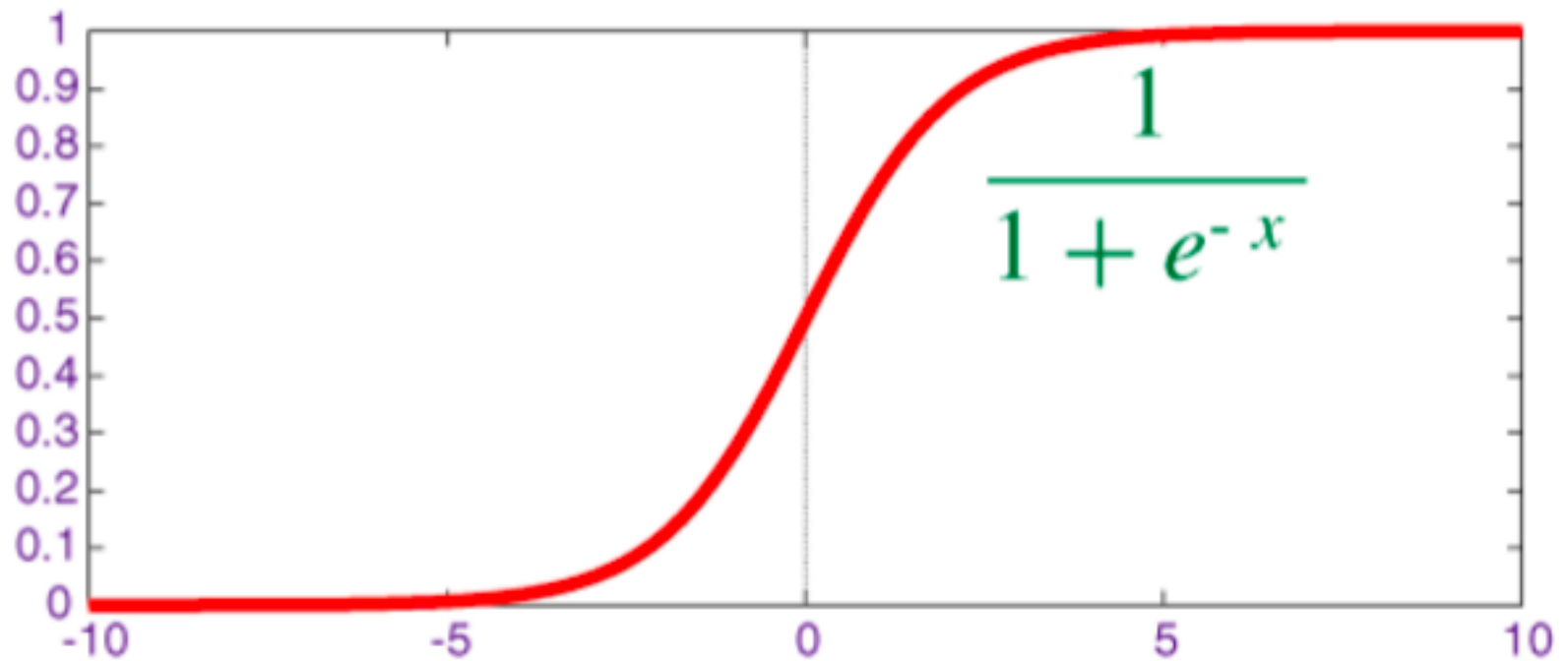
Deep Learning



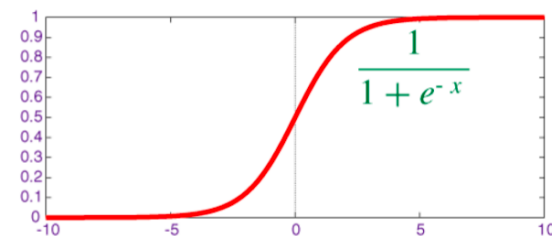
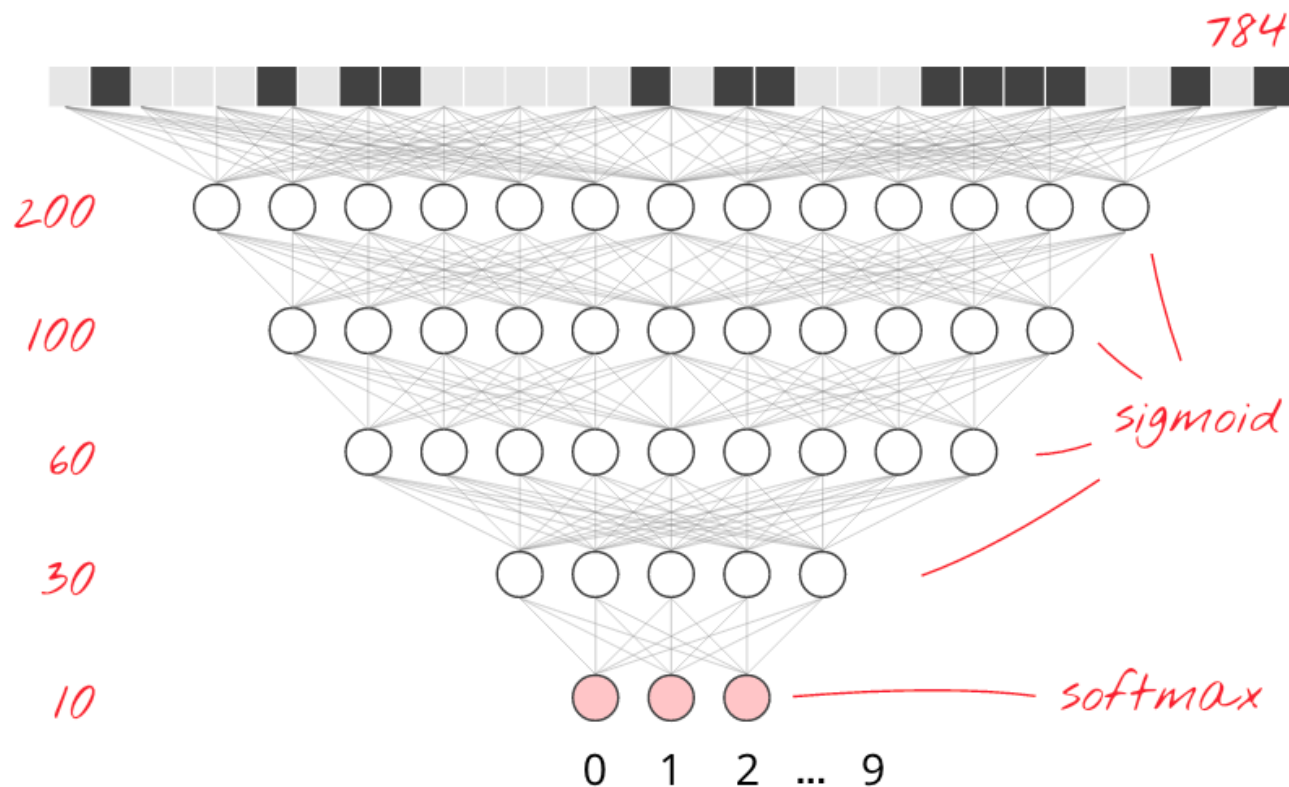
5 fully-connected layers



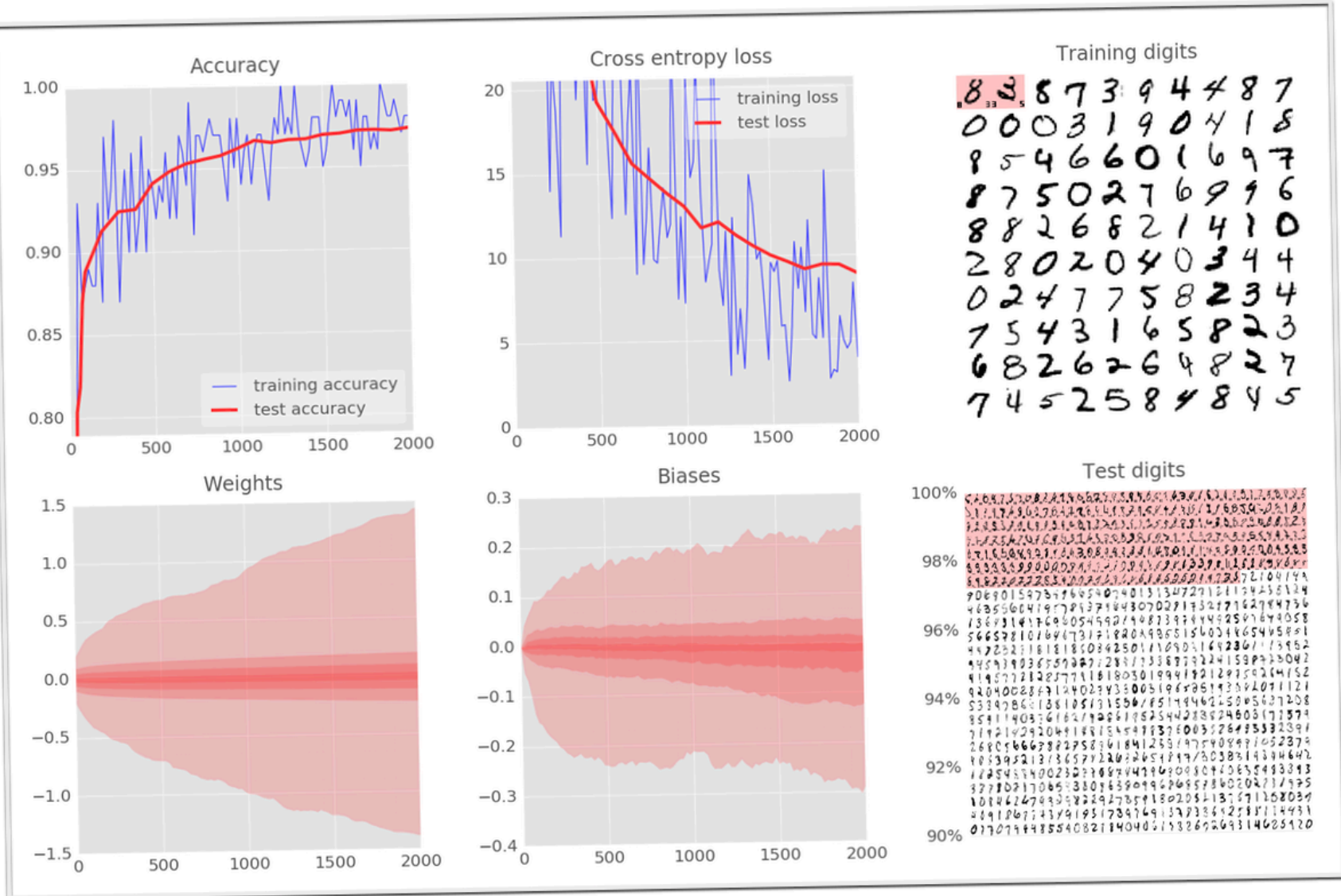
Sigmoid



5 fully-connected layers



TensorFlow MNIST Tutorial

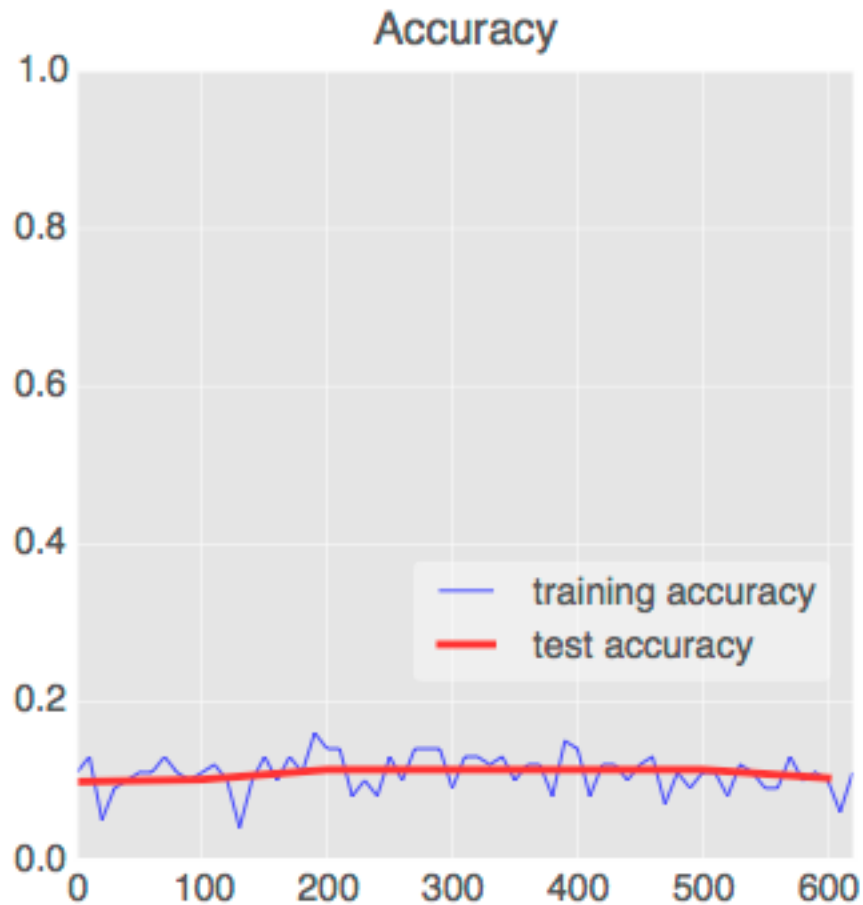


97%
😊

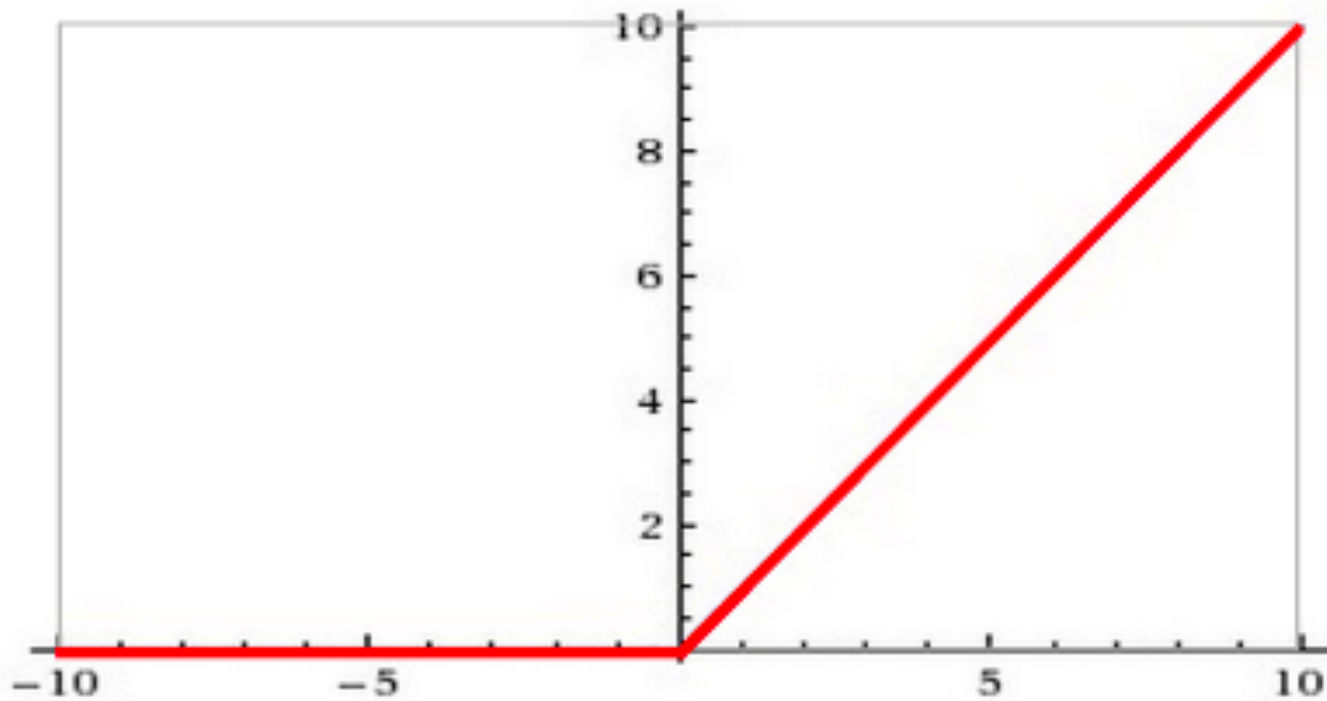
ReLU



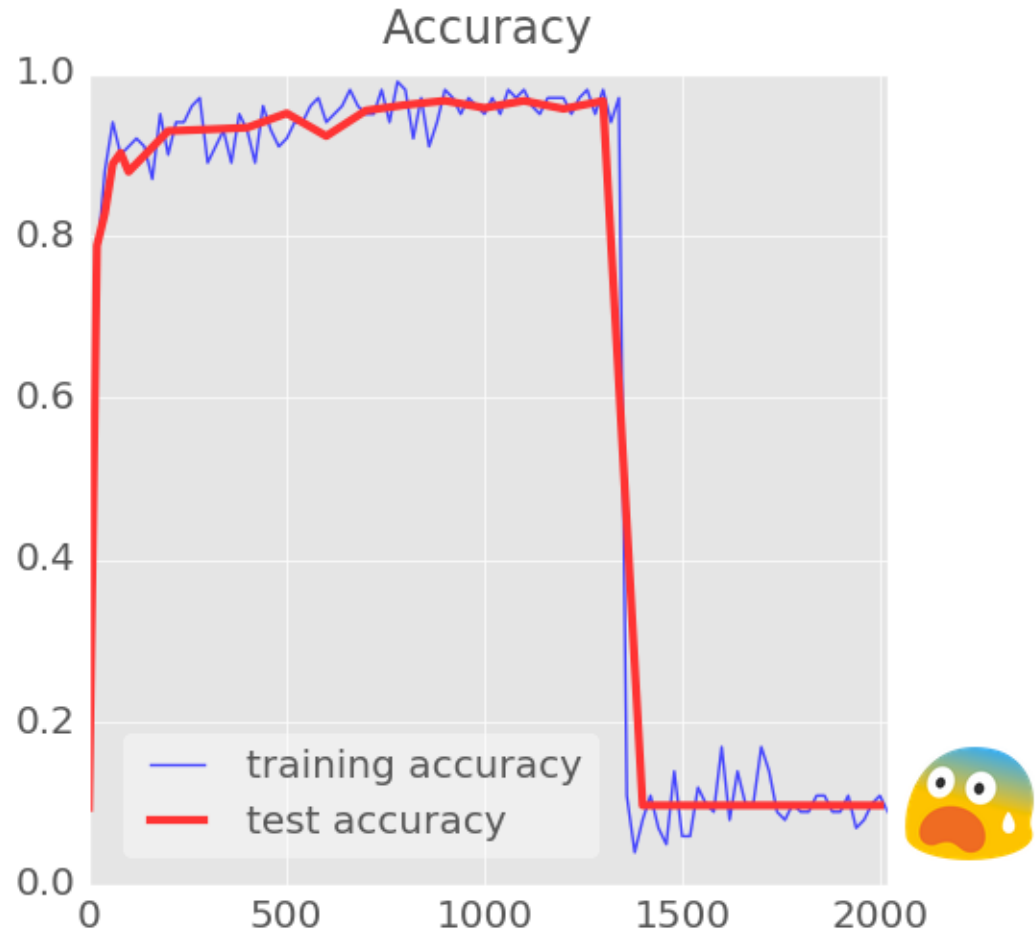
TensorFlow MNIST Tutorial



ReLU



TensorFlow MNIST Tutorial



Learning Rate

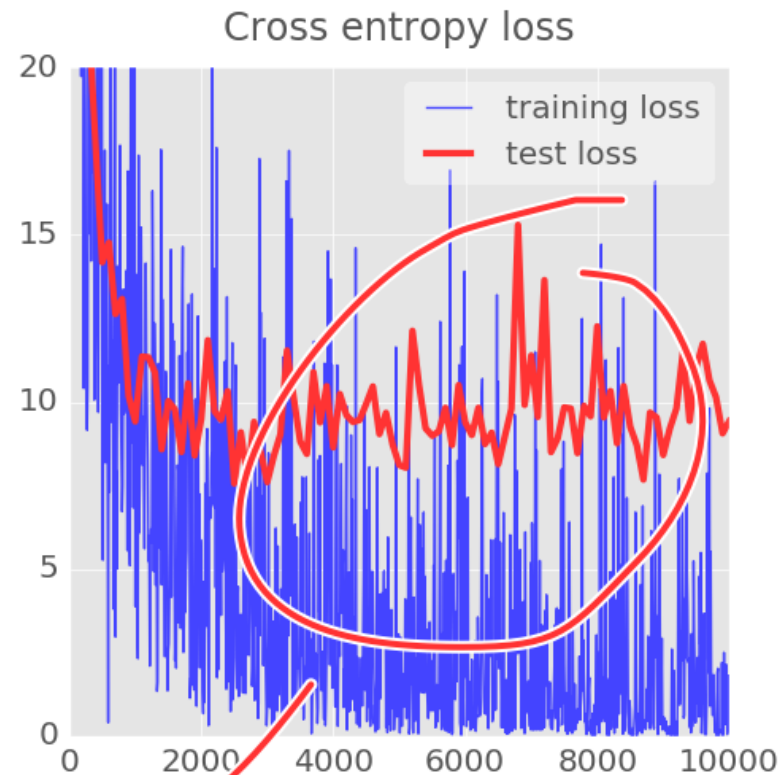
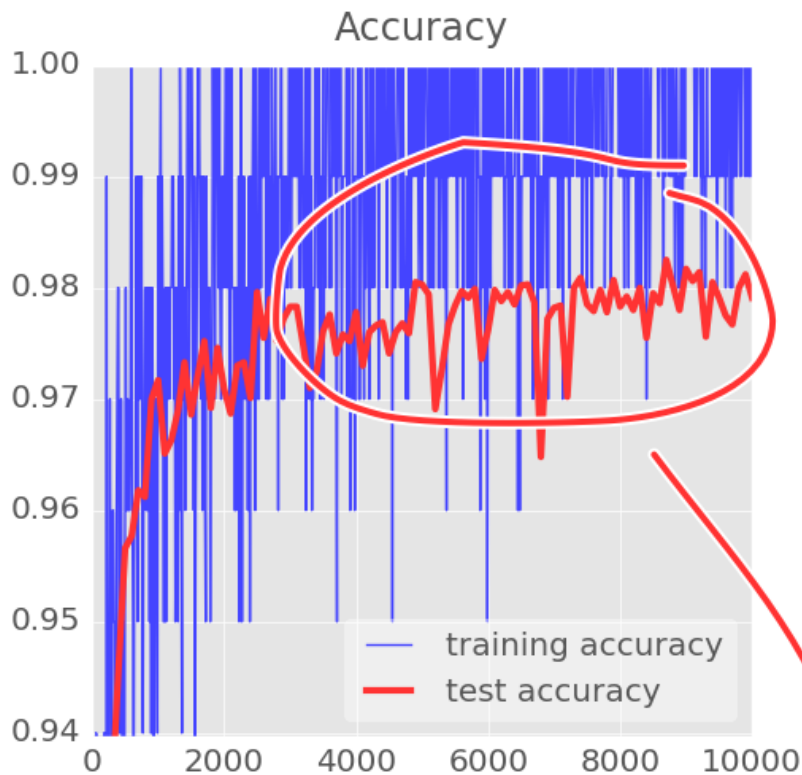
Slow down...

Learning
rate decay



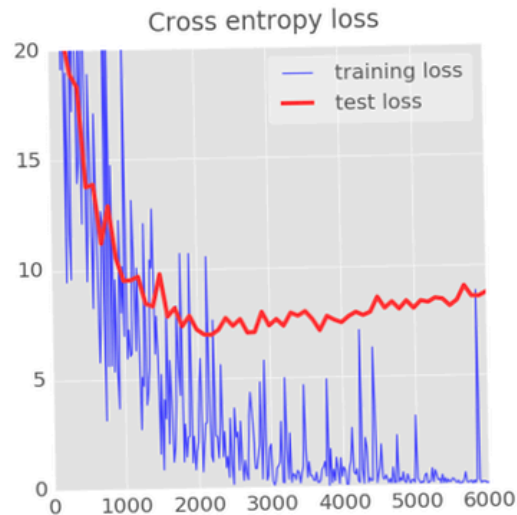
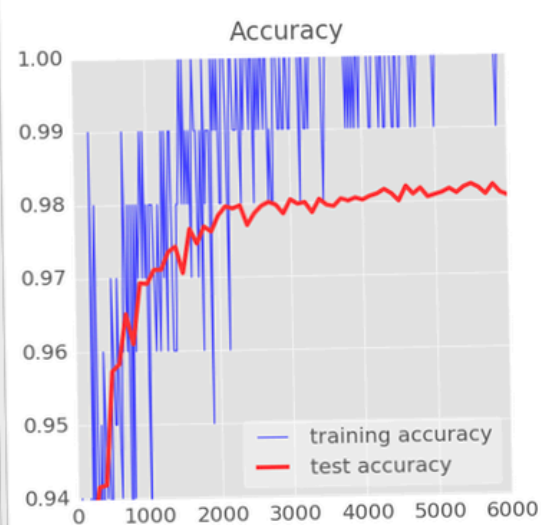
TensorFlow MNIST Tutorial

LR =
0.003



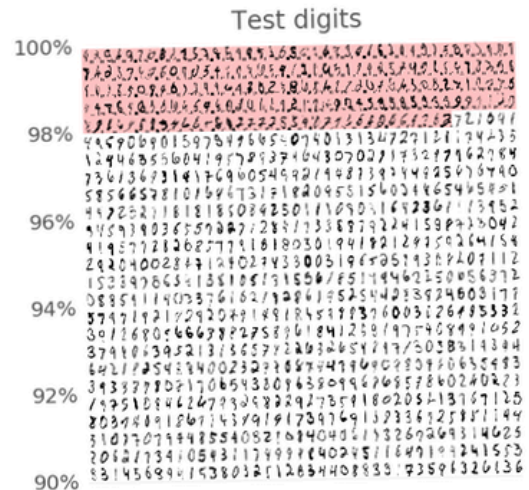
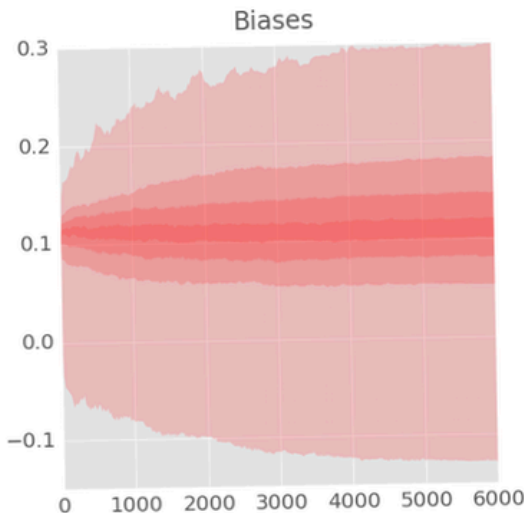
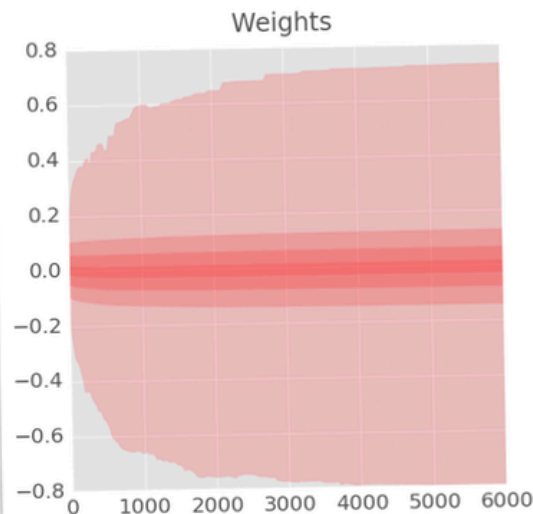
yuck!

TensorFlow MNIST Tutorial



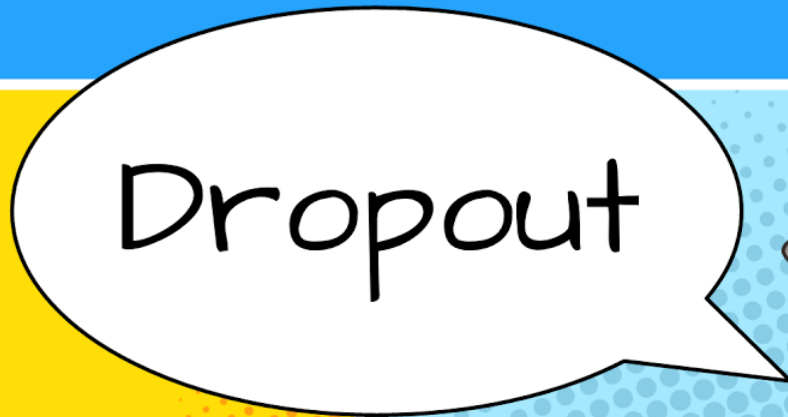
Training digits

5806282865
8441679945
0787807107
5575583859
6624086751
3910540501
6483737811
1688026570
0667548645
1872501529

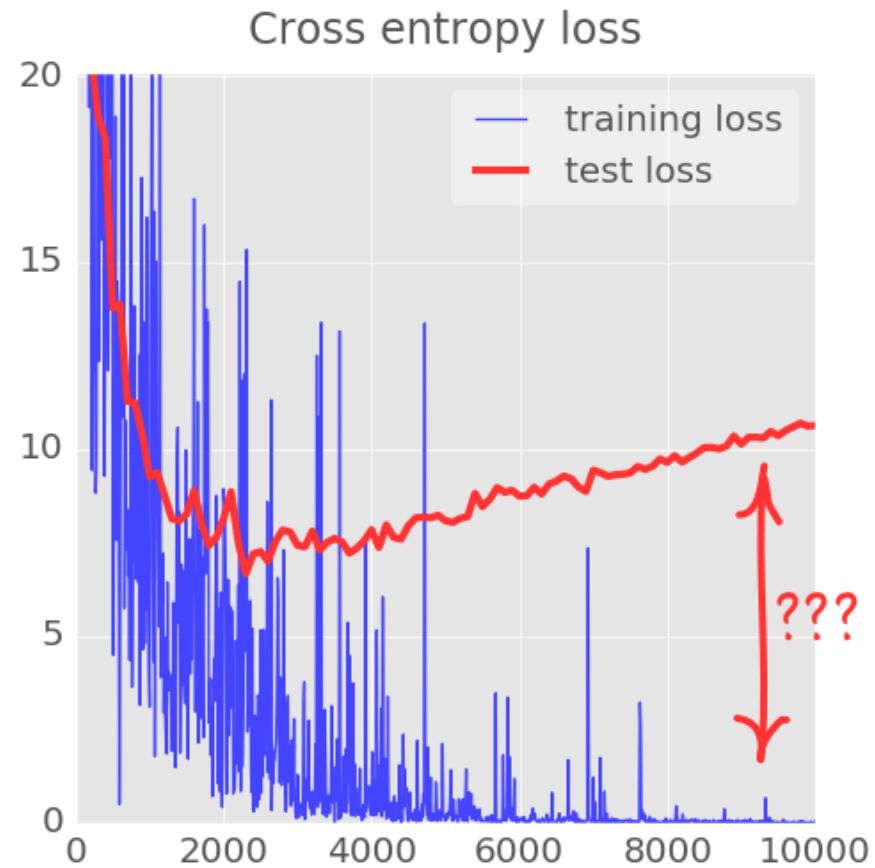
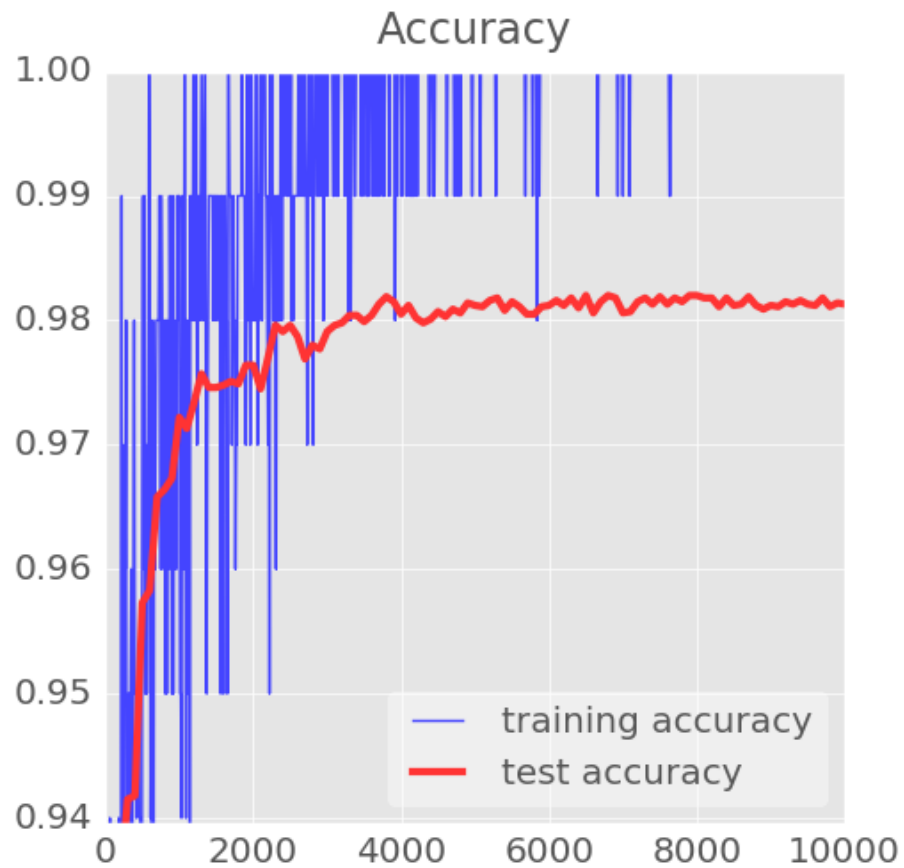


98%
😊

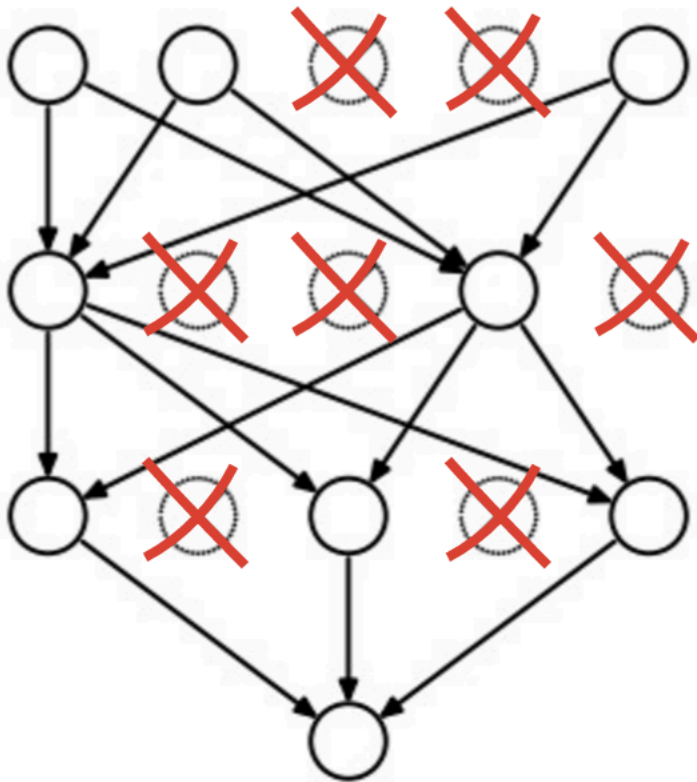
Dropout



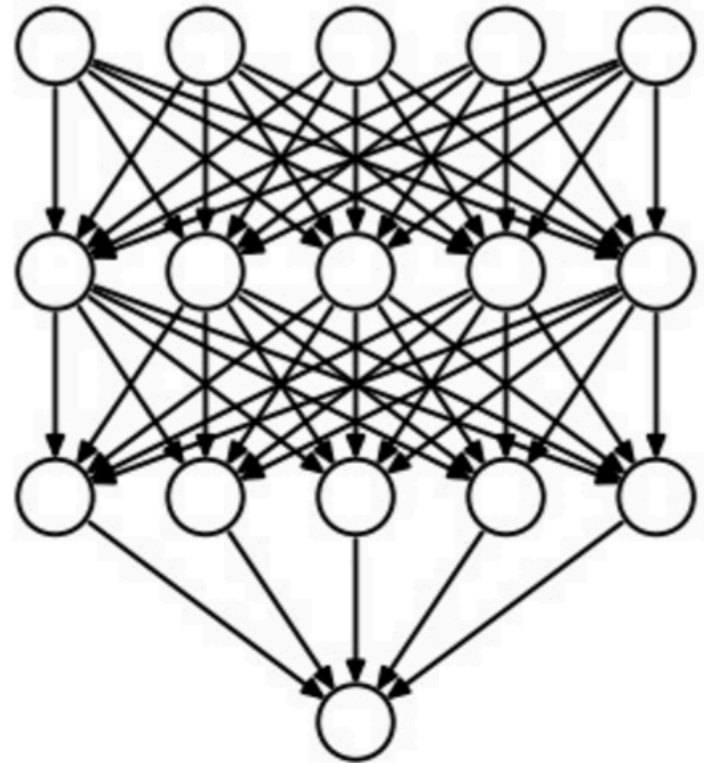
Overfitting



Dropout

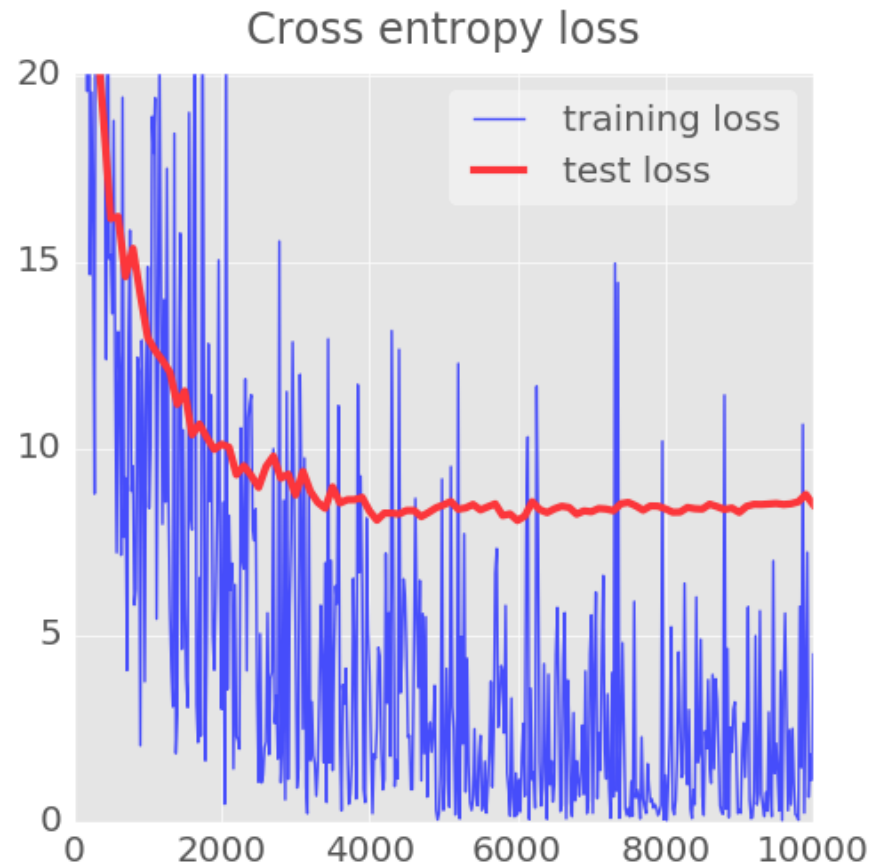
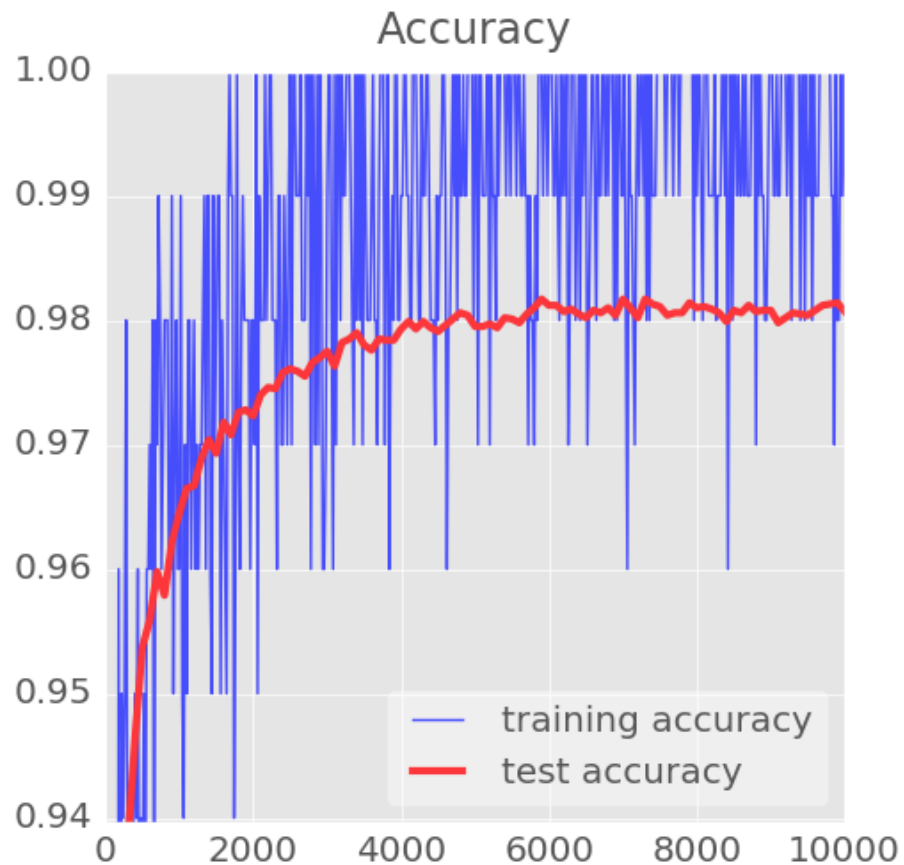


Training
 $p_{\text{keep}} = 0.75$

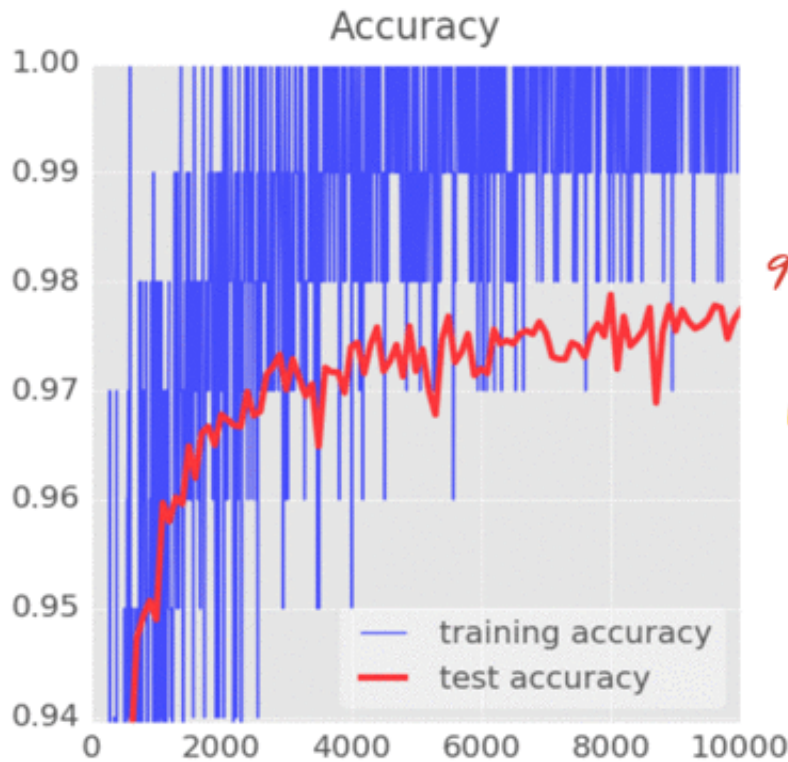


Test
 $p_{\text{keep}} = 1.0$

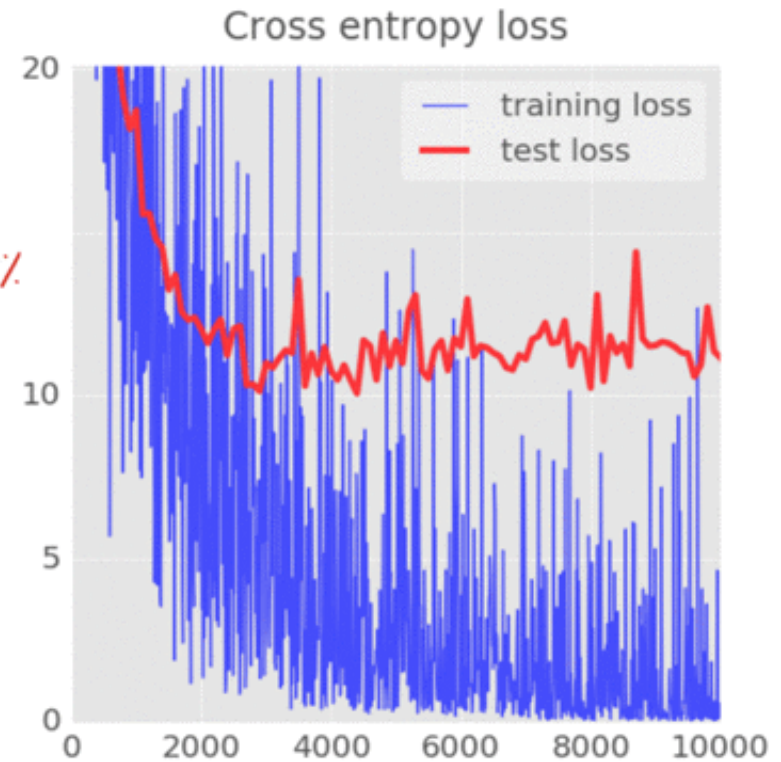
TensorFlow MNIST Tutorial



TensorFlow MNIST Tutorial



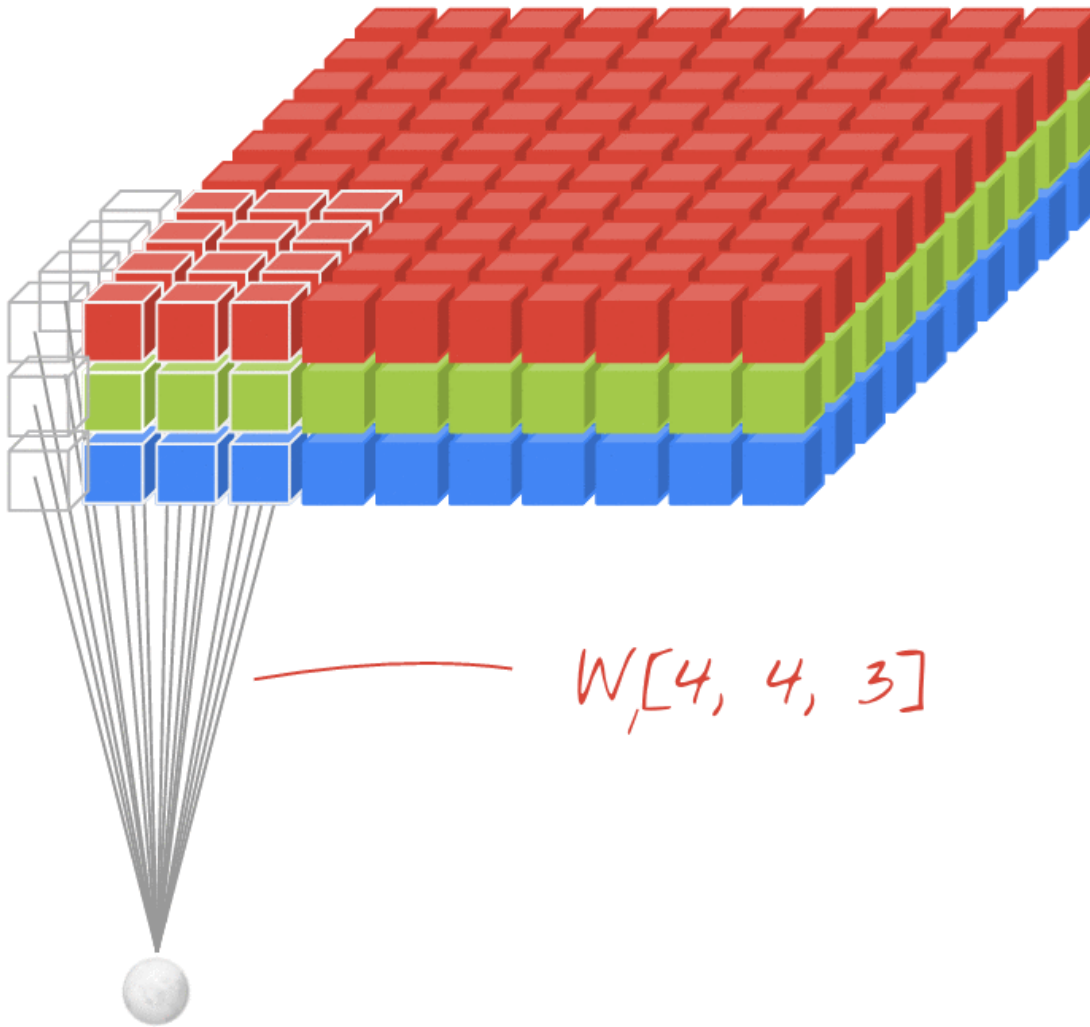
97.9%



Overfitting

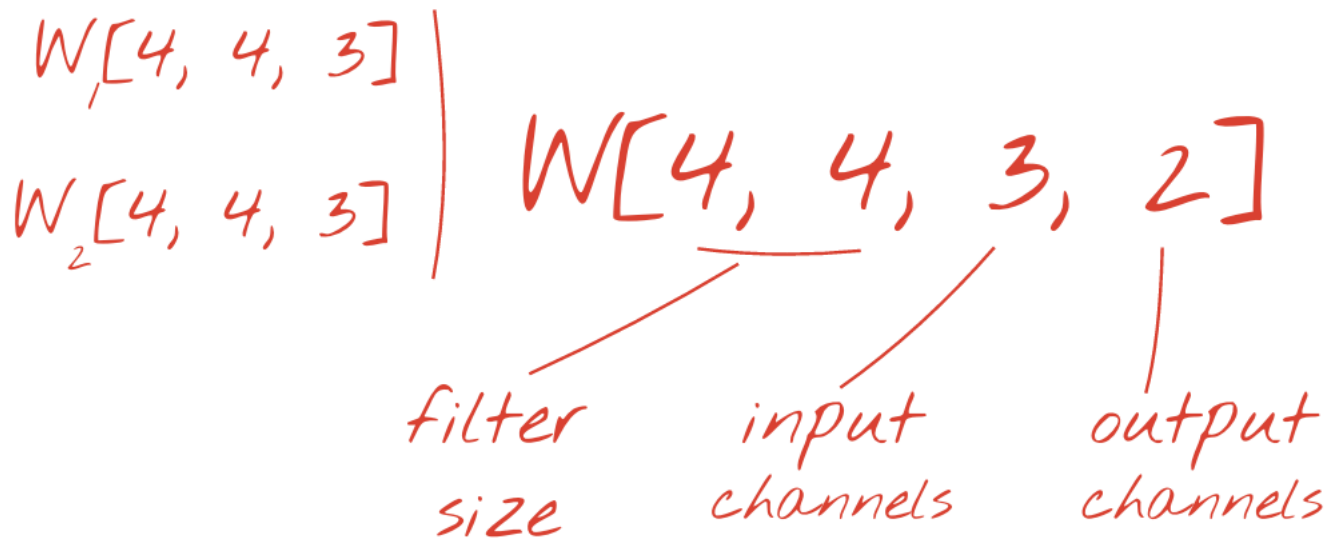


Convolutional Layer

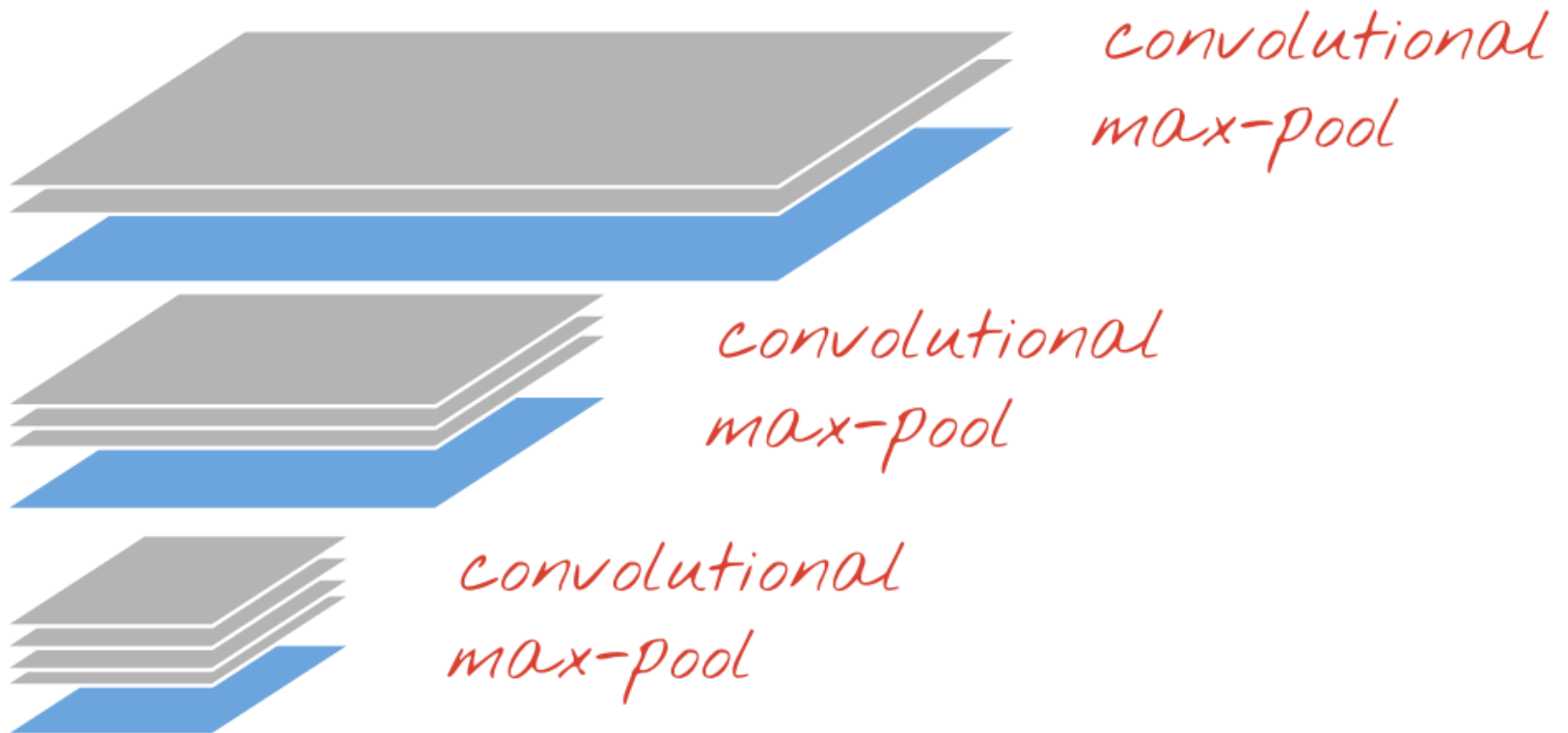


$W[4, 4, 3]$

Convolutional Layer



Convolutional Max-Pool



All Convolutional

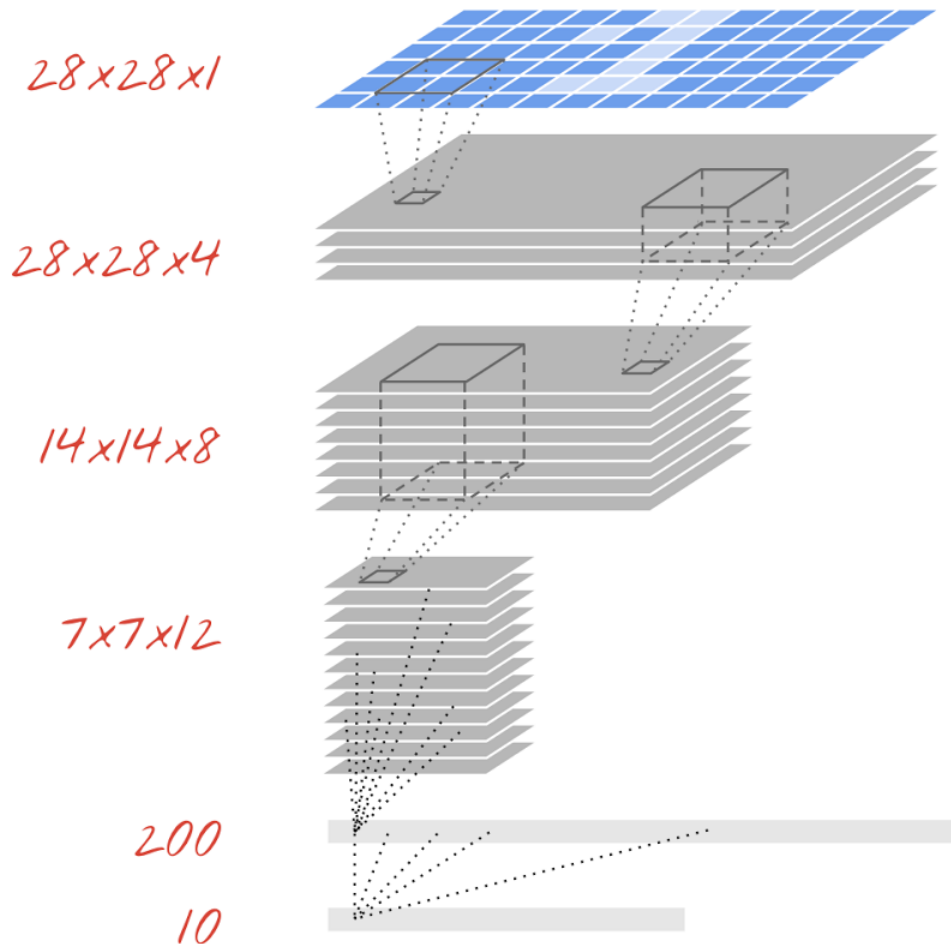
Hacker's tip



ALL
Convolutional

Bigger Convolutional Neural Network

+ biases on
all layers



convolutional layer, 4 channels
 $W1[5, 5, 1, 4]$ stride 1

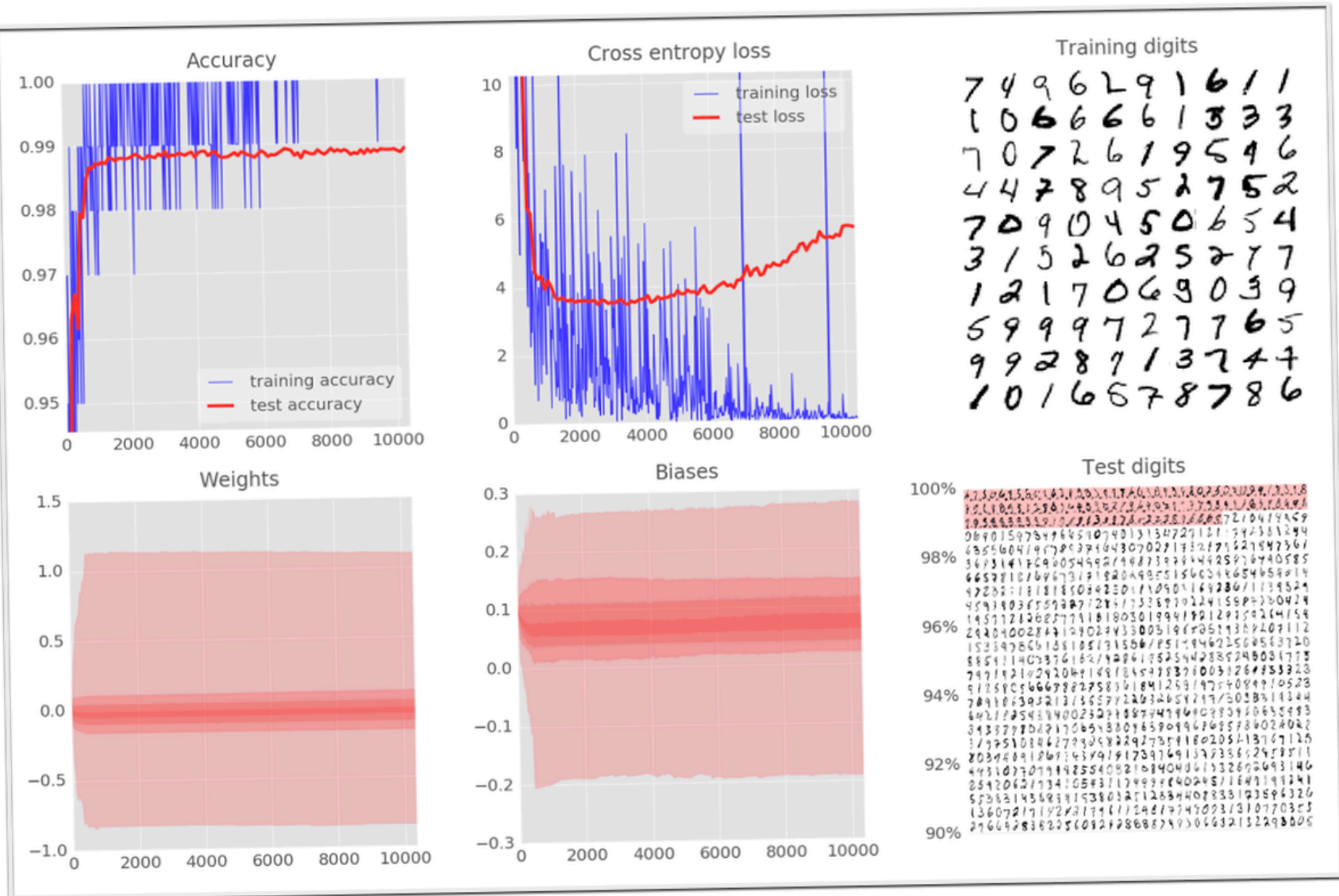
convolutional layer, 8 channels
 $W2[4, 4, 4, 8]$ stride 2

convolutional layer, 12 channels
 $W3[4, 4, 8, 12]$ stride 2

fully connected layer
 $W4[7 \times 7 \times 12, 200]$

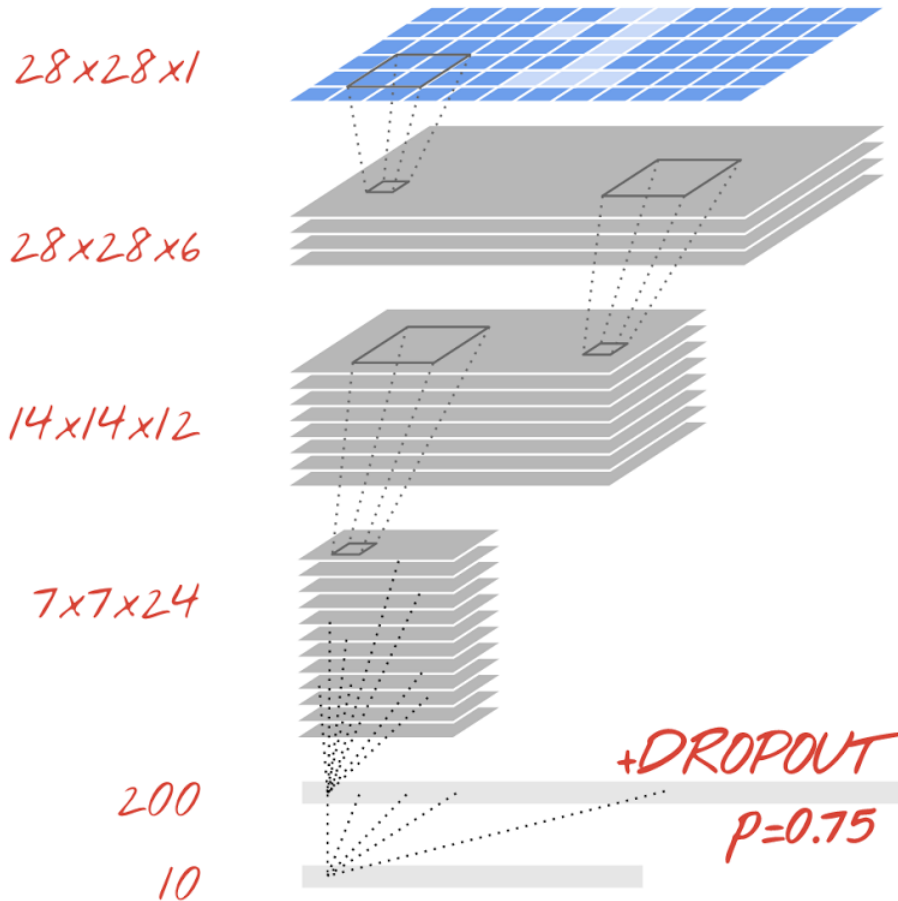
softmax readout layer
 $W5[200, 10]$

Bigger Convolutional Neural Network



Bigger Convolutional Neural Network + Dropout

+ biases on
all layers



convolutional layer, 6 channels
 $W1[6, 6, 1, 6]$ stride 1

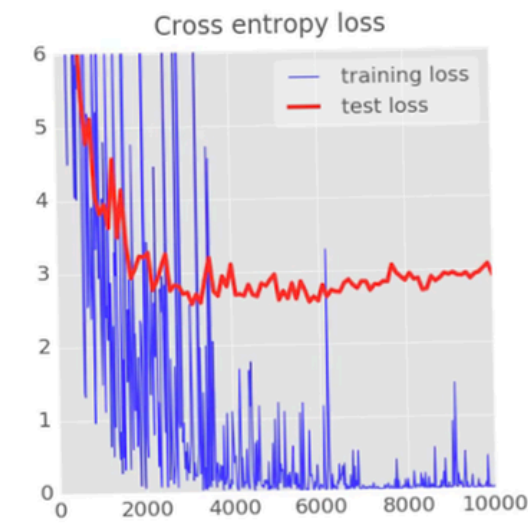
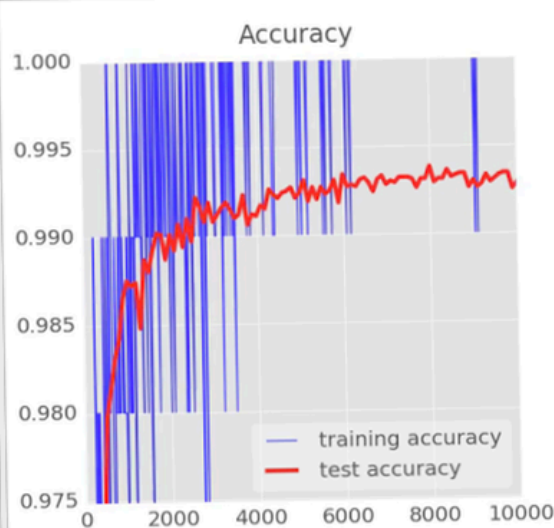
convolutional layer, 12 channels
 $W2[5, 5, 6, 12]$ stride 2

convolutional layer, 24 channels
 $W3[4, 4, 12, 24]$ stride 2

fully connected layer
 $W4[7 \times 7 \times 24, 200]$

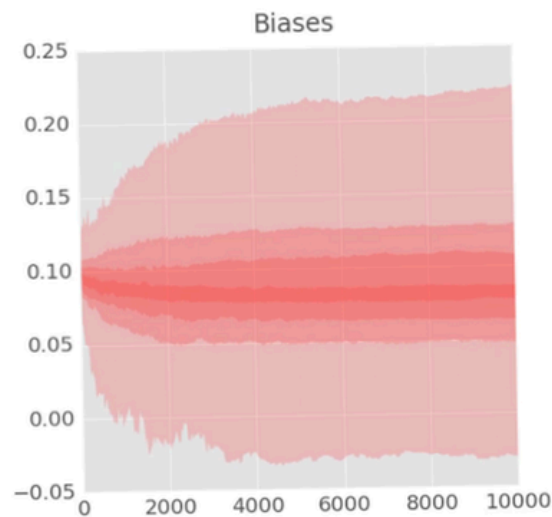
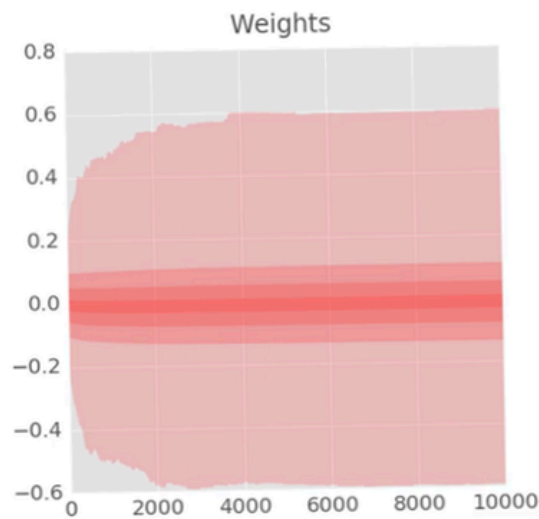
softmax readout layer
 $W5[200, 10]$

TensorFlow MNIST Tutorial



Training digits

9 4 1 9 2 1 6 9 4 9
0 1 6 1 5 3 8 1 9 3
9 6 8 9 9 0 1 6 2 8
0 1 2 6 3 3 1 1 3 8
9 3 2 4 5 2 0 0 0 9
3 3 4 9 5 0 5 6 1 4
7 9 7 3 6 4 1 9 5 2
9 0 1 8 0 4 5 6 8 2
6 3 0 1 0 0 5 7 2 5
1 2 4 4 2 8 5 6 7 0



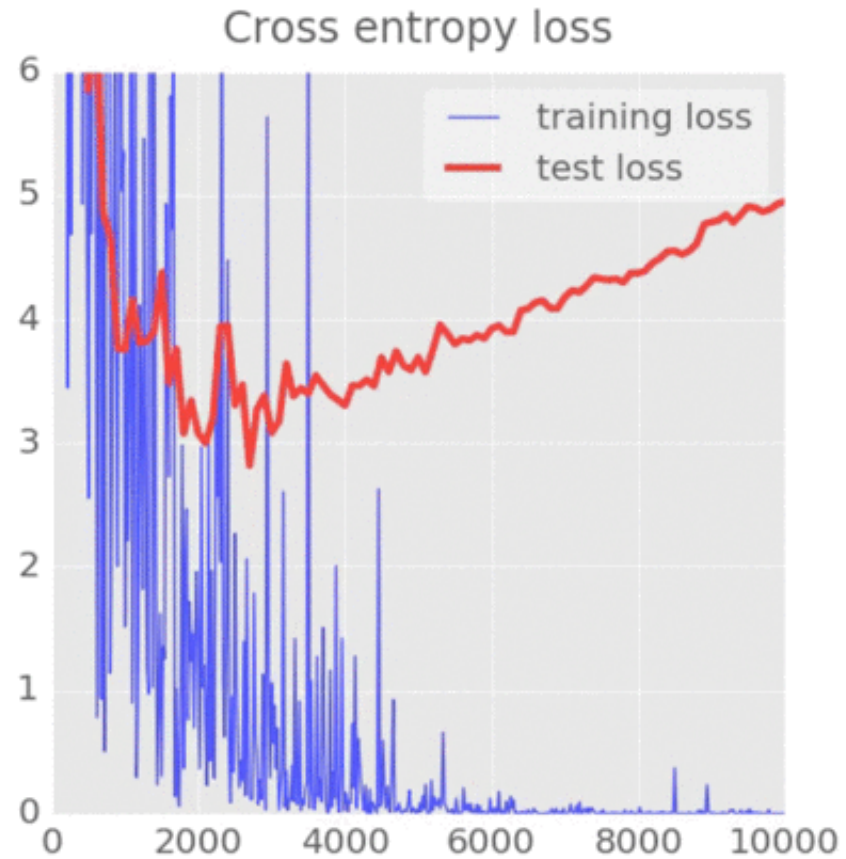
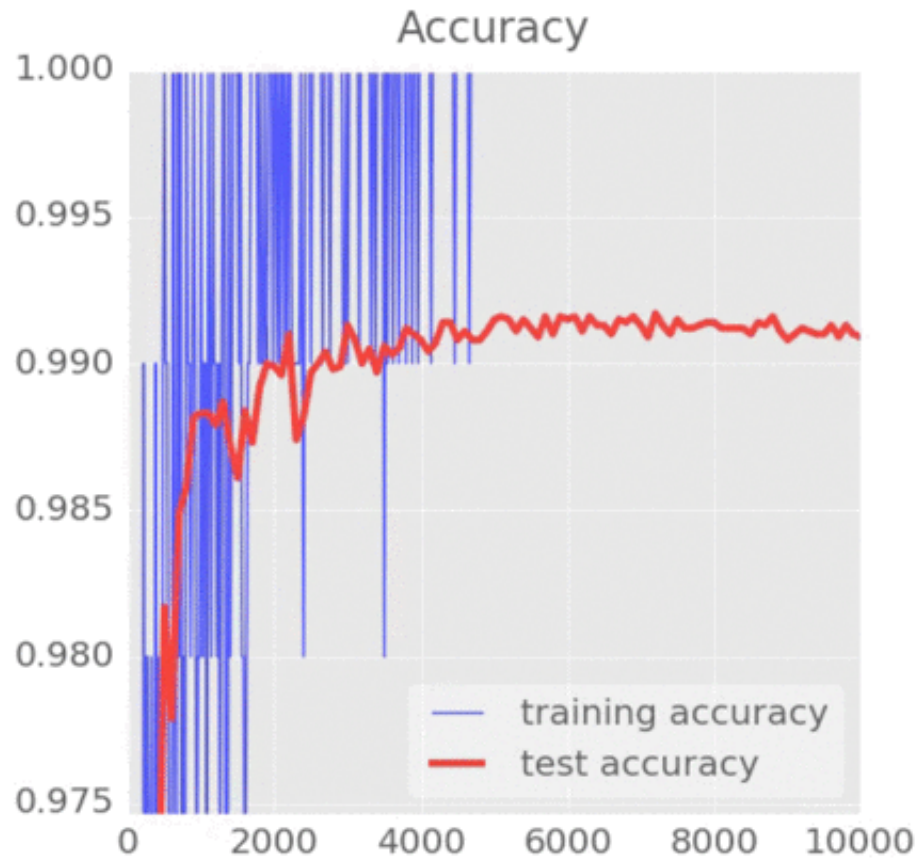
Test digits

100% 4 7 5 2 2 1 5 6 4 1 1 1 3 0 7 4 6 5 3 0 0 0 4 1 5 1 6 1 1 0 1 0
98% 6 2 1 5 1 7 1
96% 5 9 0 6 9 0 1 5 9 7 5 1 9 6 5 4 0 7 4 0 1 3 1 3 4 7 2 9 1 2 1 1 7 4 4 1 9 4
94% 4 6 3 5 5 6 0 4 1 9 5 7 2 5 3 7 1 0 4 3 0 7 0 2 1 7 1 2 1 7 1 6 2 7 8 4 7 3
92% 6 1 3 6 7 3 1 4 1 7 6 4 0 5 4 9 2 1 4 4 2 1 3 1 4 4 2 5 6 7 4 0 5 8
90% 5 6 6 5 2 8 1 0 1 0 4 7 3 1 1 1 2 0 4 9 8 5 1 5 4 0 3 4 6 5 4 5 8 1 1
4 1 2 0 5 2 1 1 8 1 8 5 0 8 4 2 5 0 1 1 0 9 0 1 1 6 1 2 3 6 1 1 1 3 1 5 2
4 1 5 4 1 9 0 3 6 5 9 8 2 7 1 2 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4 1 4 5 7 2 1 3 6 2 7 7 1 1 1 1 8 0 3 0 1 9 9 4 1 1 2 2 4 1 5 9 2 6 1 5
4 2 9 2 0 4 0 0 2 8 4 1 2 4 0 2 7 3 3 0 0 3 1 8 4 5 0 5 1 1 3 6 4 0 1 1
2 1 5 3 3 9 7 8 6 5 1 2 5 1
2 0 8 8 5 4 1 1 4 0 3 7 6 1 5 2 1 1 2 0 6 1 4 6 2 5 4 0 8 8 2 4 6 0 3 1 7
3 3 7 1 1 1 1 1 1 1 1 1 2 9 2 0 4 1 1 4 8 1 8 4 5 1 1 1 1 1 1 1 1 1 1 1 1 1
3 0 1 1 6 8 0 4 6 6 7 3 2 2 7 5 1
3 7 4 4 1 6 7 3 9 1 2 1 3 1 3 6 5 7 1 2 2 4 1 2 4 5 1 1 1 1 1 1 1 1 1 1 1 1
4 6 4 2 1 1 2 5 4 1 1 4 0 0 2 3 2 1 1 6 5 7 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3 3 1 3 3 7 7 0 1 7 0 6 5 1 3 8 0 1 6 5 5 0 9 4 2 6 5 5 7 8 4 0 2 4 0 2 1
7 4 7 5 1 0 1 4 6 2 6 7
2 0 3 4 6 0 1 1 2 6 1 1 4 1 9 1 1 7 3 9 7 4 9 1 3 7 3 3 6 1 2 4 1 1 1 1
4 4 3 1 0 7 7 0 7 9 4 2 8 5 4 0 5 2 1 0 1 4 6 0 4 6 1 1 3 2 6 7 2 6 9 1 4
6 2 5 1 2 0 6 2 7 1 3 4 1 0 5 4 3 1 1 7 9 9 5 1 1 4 0 2 4 5 1 1 6 4 7 1 1 4 2 4
1 5 3 5 3 1 4 5 6 8 9 4 1 5 3 8 0 3 1 1 2 5 3 4 0 8 8 3 3 1 2 3 5 6 3 2
6 1 3 6 0 2 1 7 1 2 2 1
5 2 1 6 6 9 2 8 3 8 2 5 6 0 9 2 7 0 8 0 8 7 9 3 0 6 6 3 2 1 5 2 3 5 8 0
5 7 8 1 4 1 4 0 2 1 1 1 7 4 7 3 9 3 3 4 7 1 2 1 2 3 2 3 2 3 9 1 7 4 0 3 5 5 5

99.3%



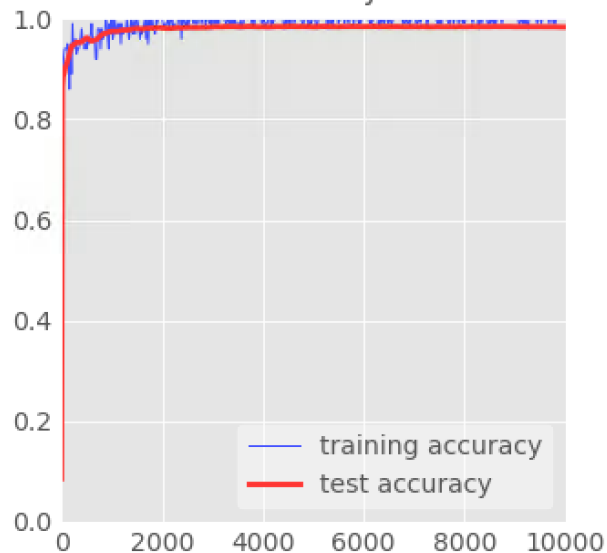
TensorFlow MNIST Tutorial



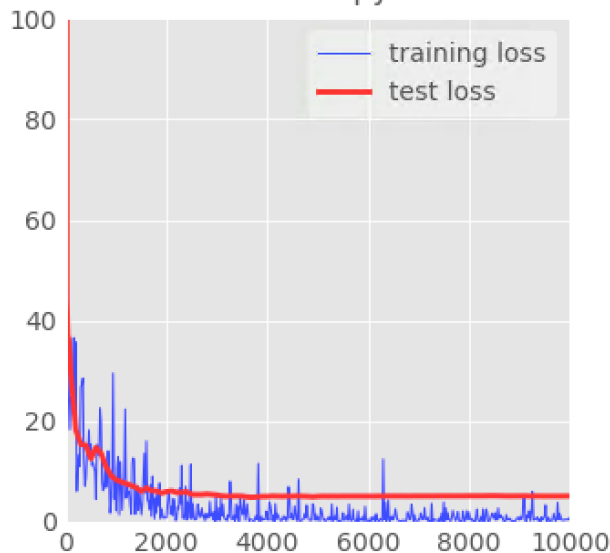
larger convolutional network

TensorFlow MNIST Tutorial

Accuracy



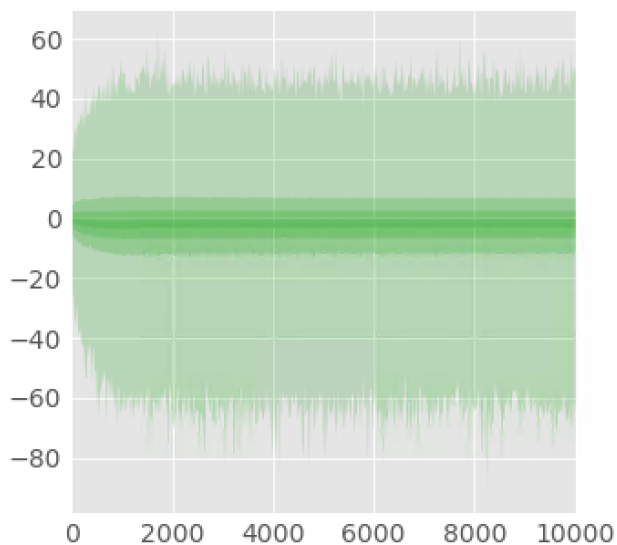
Cross entropy loss



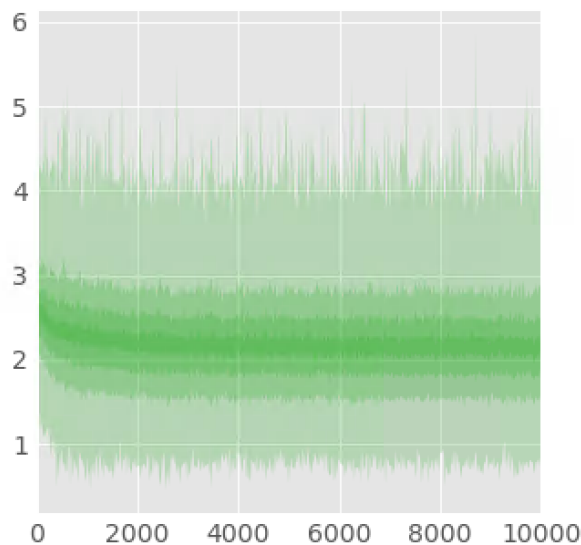
Training digits

6 6 4 3 5 5 0 7 1 0
4 5 4 4 1 2 7 4 3 4
7 4 2 2 4 0 9 2 3 0
9 5 8 0 4 1 4 0 1 0
2 7 5 4 5 7 1 7 5 9
0 1 1 6 4 7 5 5 1 3
7 5 2 9 5 9 2 9 9 4
6 1 8 0 0 7 8 0 2 3
7 1 7 6 0 2 7 1 1 0
5 4 0 6 3 4 4 9 9 3

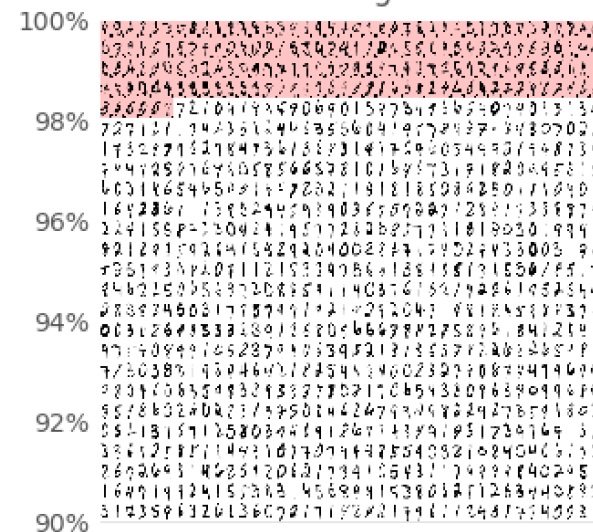
Logits



Max activations across batch



Test digits



TensorFlow MNIST Tutorial

```
python3 mnist_1.0_softmax.py
```

```
python mnist_1.0_softmax.py
```

```
pythonw mnist_1.0_softmax.py
```

```
python3 mnist_2.0_five_layers_sigmoid.py
```

```
python3 mnist_2.2_five_layers_relu_lrdecay_dropout.py
```

```
python3 mnist_3.0_convolutional.py
```

```
python3 mnist_3.1_convolutional_bigger_dropout.py
```

```
python3 mnist_4.0_batchnorm_five_layers_sigmoid.py
```

```
python3 mnist_4.1_batchnorm_five_layers_relu.py
```

```
python3 mnist_4.2_batchnorm_convolutional.py
```

References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), <https://www.youtube.com/watch?v=u4aIGiomYP4>
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), <https://www.youtube.com/watch?v=fTUwdXUffl8>
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, <https://goo.gl/pHeXe7>, <https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist>
- Deep Learning Basics: Neural Networks Demystified, <https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU>
- Deep Learning SIMPLIFIED, <https://www.youtube.com/playlist?list=PLjJh1vISEYgvGod9wWiydumYl8hOXixNu>
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, <https://www.youtube.com/watch?v=aircAruvnKk>
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, <https://www.youtube.com/watch?v=IHZwWFHWa-w>
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=llg3gGewQ5U>
- TensorFlow: <https://www.tensorflow.org/>
- Keras: <http://keras.io/>
- Deep Learning Studio: Cloud platform for designing Deep Learning AI without programming, <http://deepcognition.ai/>
- Natural Language Processing with Deep Learning (Winter 2017), https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFt9ulXLuQrImzHfOV
- <http://p.migdal.pl/2017/04/30/teaching-deep-learning.html>
- <https://github.com/leriomaggio/deep-learning-keras-tensorflow>



Q & A

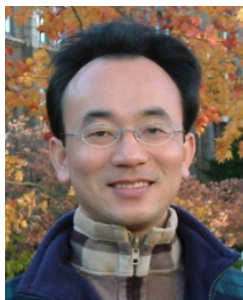


深度學習視覺辨識 (Deep Learning for Visual Recognition)

Time: 2018/01/25 (Thu) (9:00 -12:00, 13:00-16:00)

Place: 國立臺北護理健康大學 (台北市明德路365號) 親仁樓 B112

Host: 祝國忠 院長 (健康科技學院院長)



Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2018-01-25

