國立臺北護理健康大學
National Taipei University of Nursing and Health Sciences

Tamkang University
淡江大學

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

Source: https://en.wikipedia.org/wiki/MNIST_database

CIFAR-10 dataset

airplane

automobile

bird

cat

deer

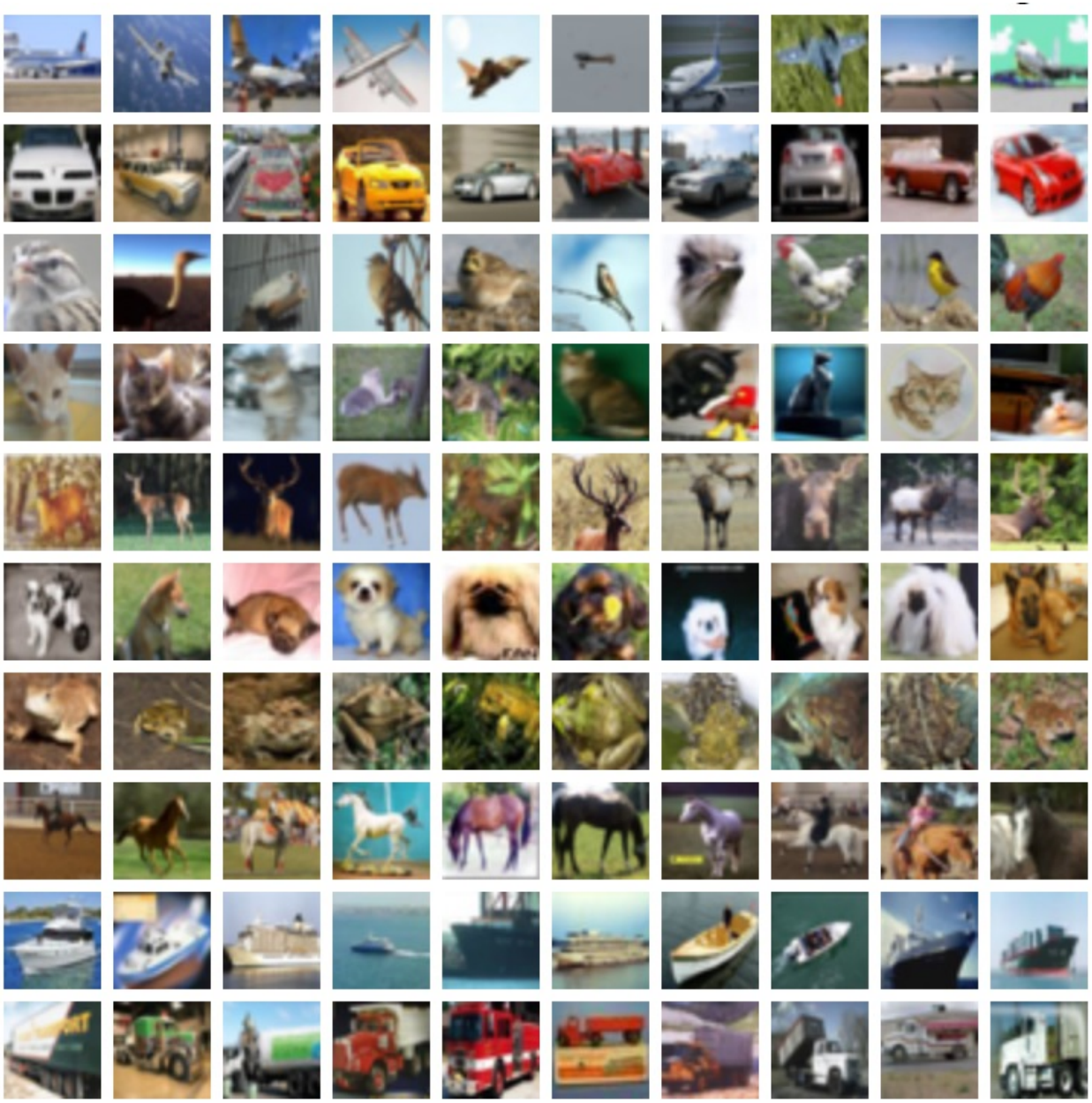
dog

frog

horse

ship

truck



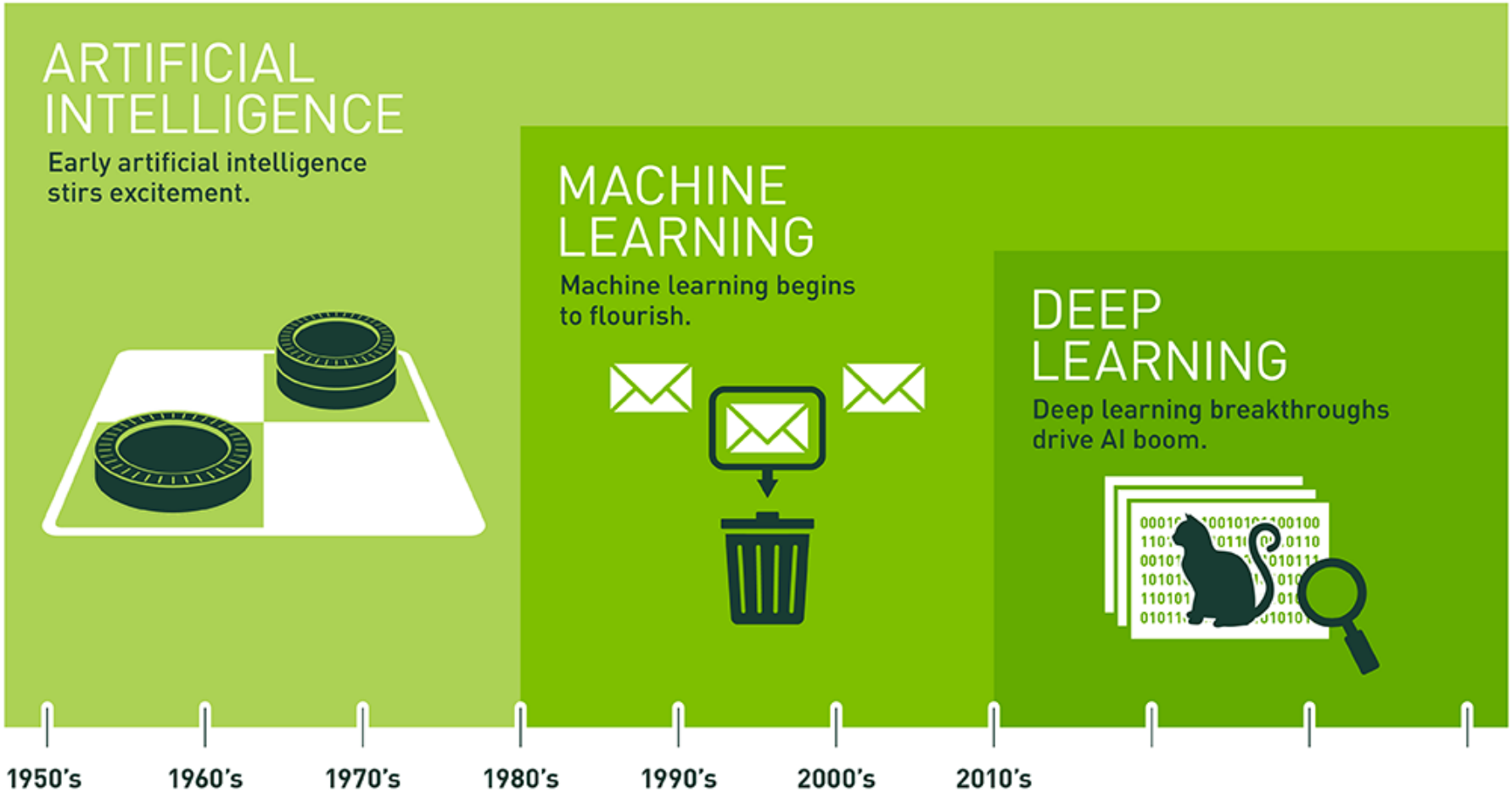
Source: https://www.cs.toronto.edu/~kriz/cifar.html

Deep Learning Foundations:

Neural Networks

Keras:
High-level API for TensorFlow

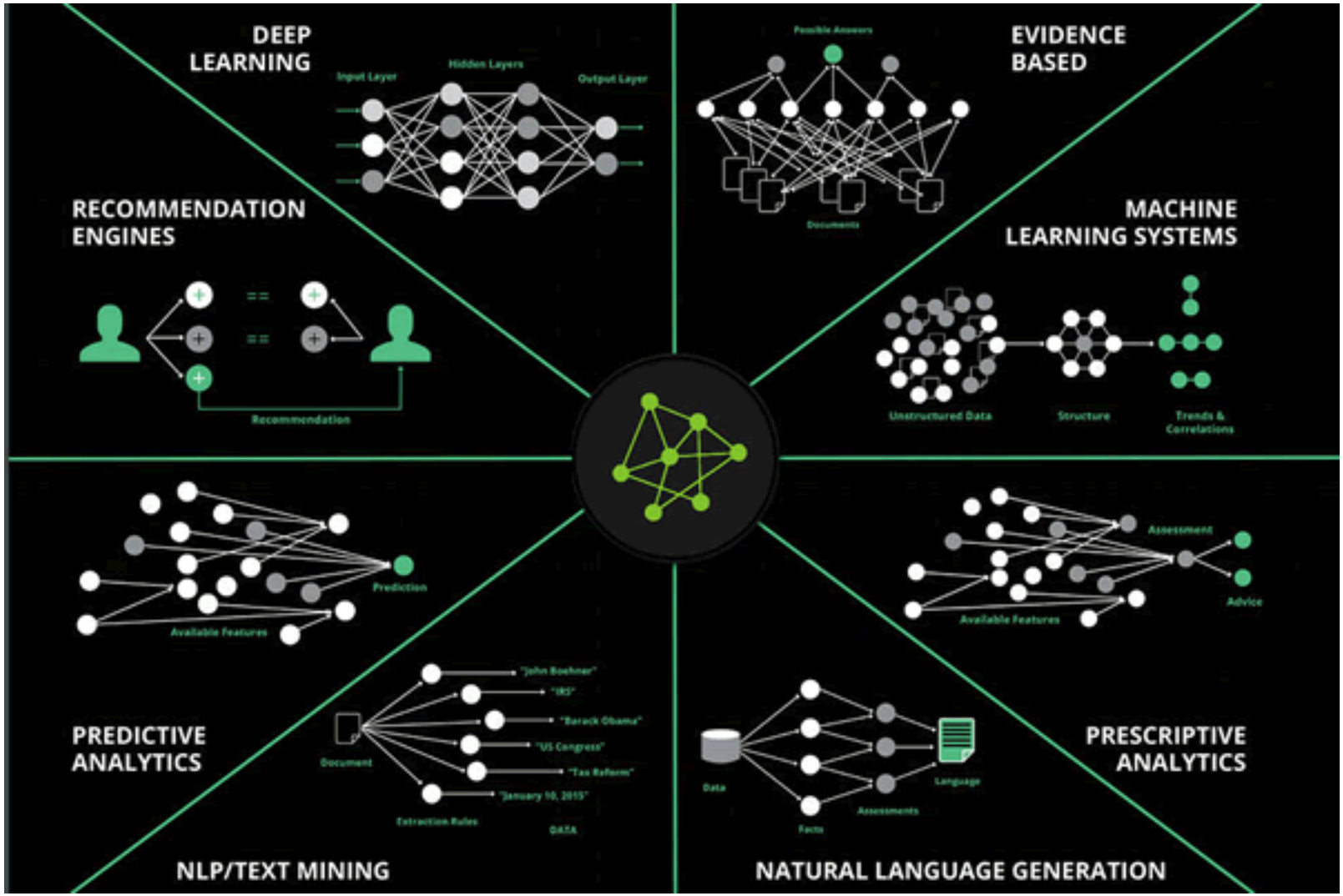
Artificial Intelligence
Machine Learning & Deep Learning



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.

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Artificial Intelligence (AI) is many things

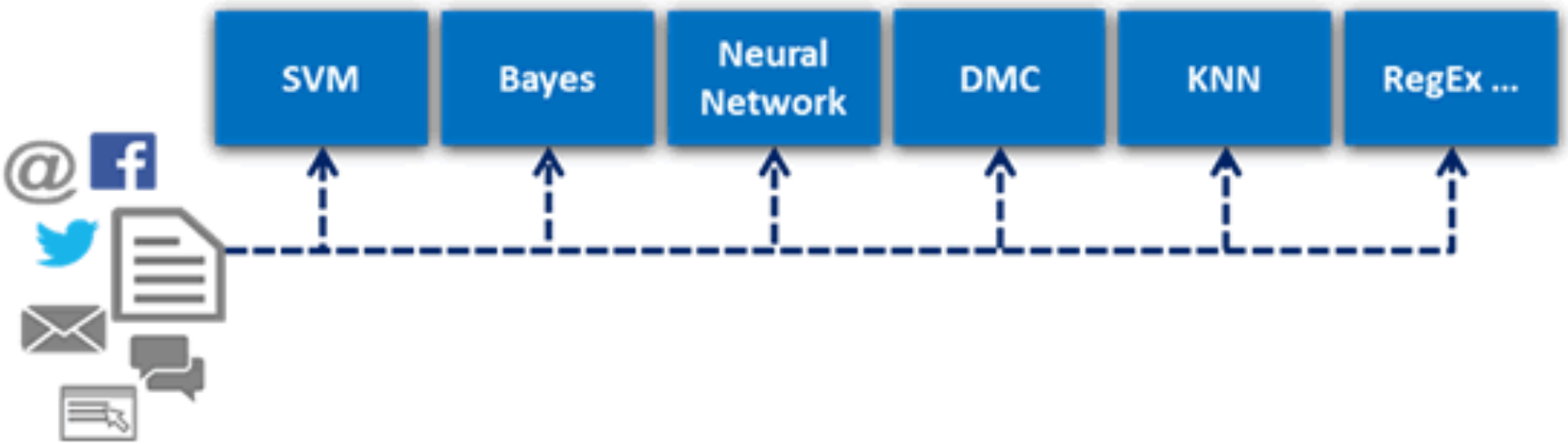


Ecosystem of AI

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

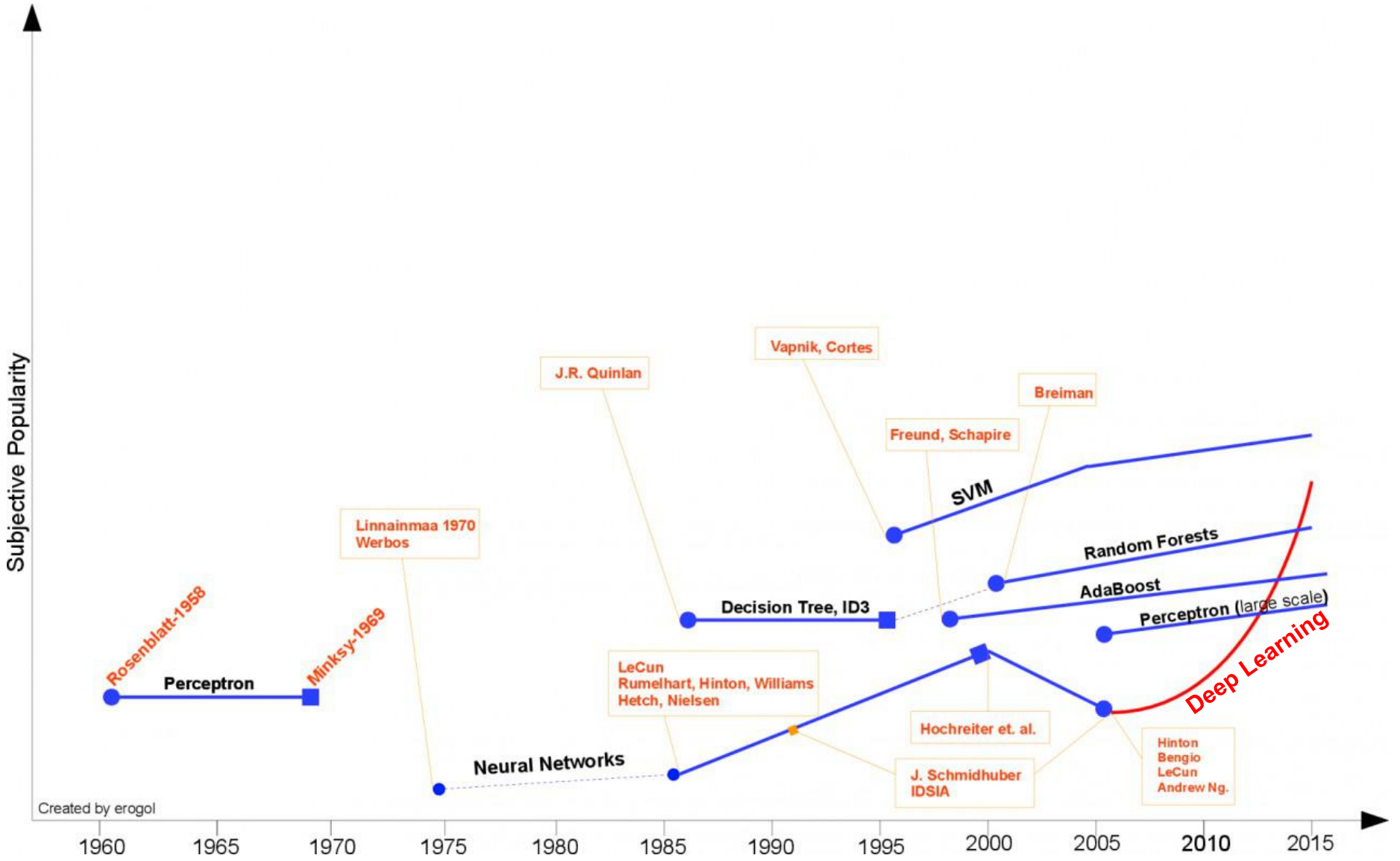
Artificial Intelligence (AI)

Intelligent Document Recognition algorithms



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

Deep Learning Evolution



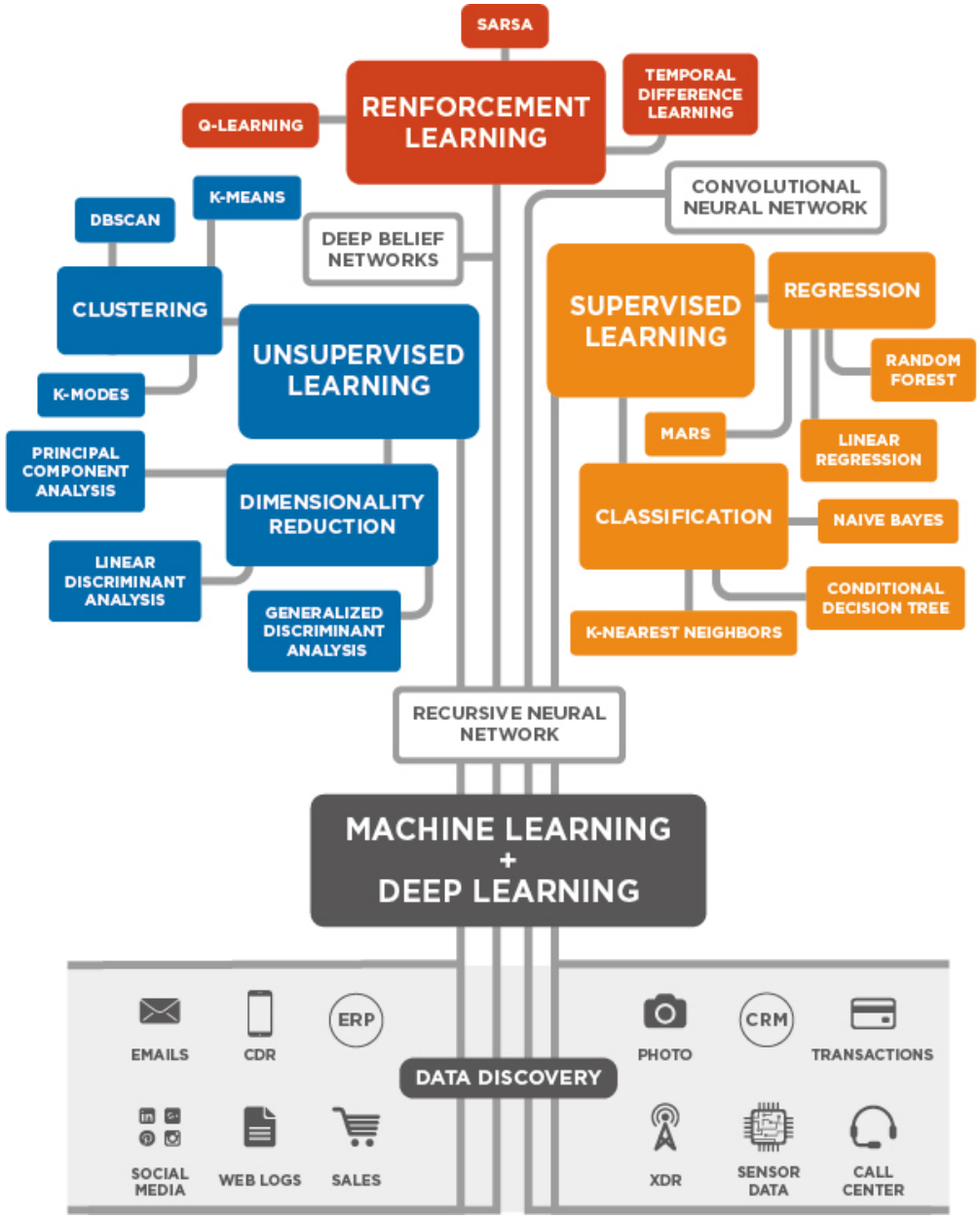
Source: http://www.erogol.com/brief-history-machine-learning/

Machine Learning Models

- Deep Learning
- Association rules
- Decision tree
- Clustering
- Bayesian
- Kernel
- Ensemble
- Dimensionality reduction
- Regression Analysis
- Instance based

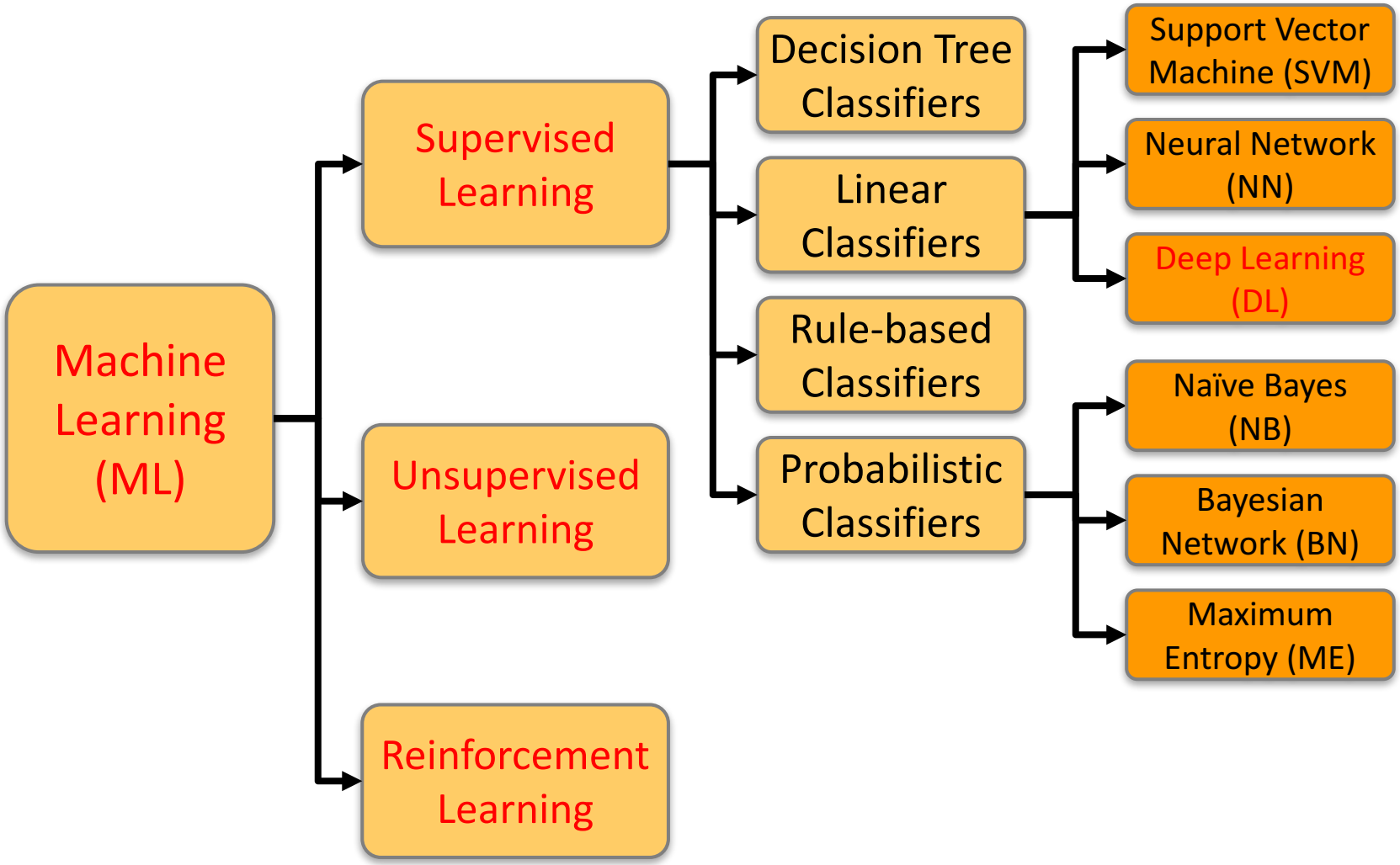
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

3 Machine Learning Algorithms



Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning (ML) / Deep Learning (DL)



Source: Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015), "Sentiment analysis: A review and comparative analysis of web services," Information Sciences, 311, pp. 18-38.

Deep Learning
and
Deep Neural Networks

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

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

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & 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, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

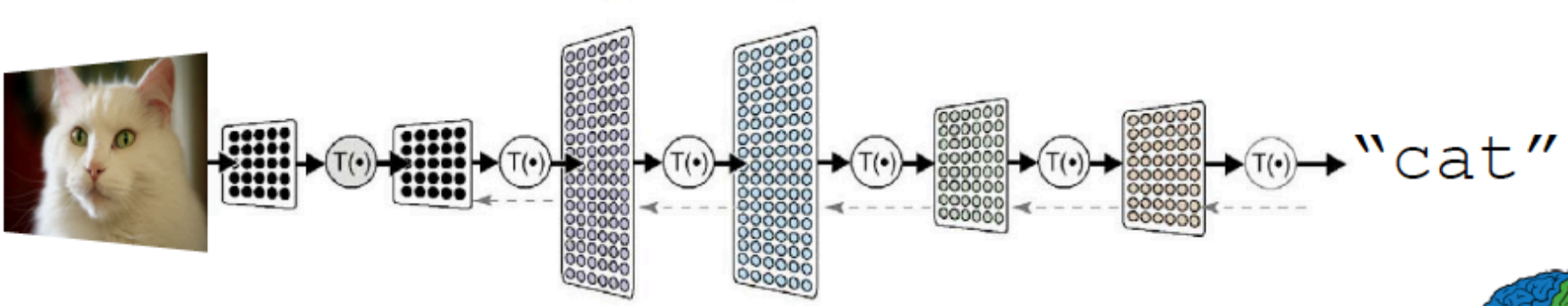
Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering 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 molecules[8], analysing particle accelerator data[9,10], reconstructing brain circuits[11], 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 understanding[14], particularly topic classification, sentiment analysis, question answering[15] and language translation[16,17].

Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

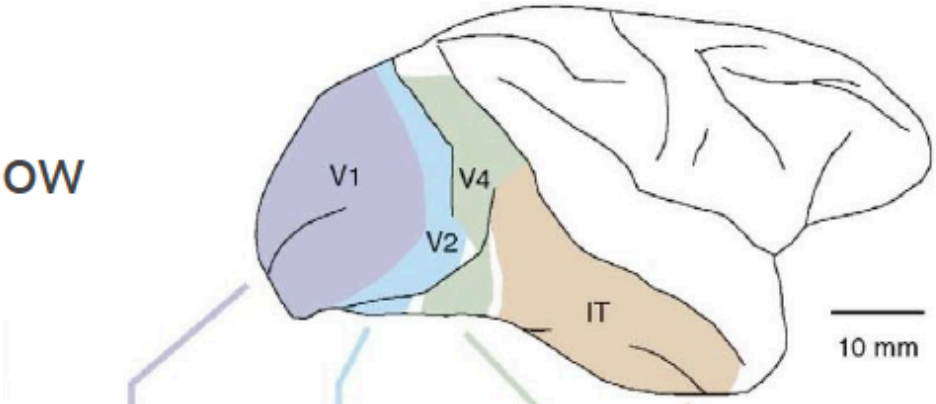
Deep Learning

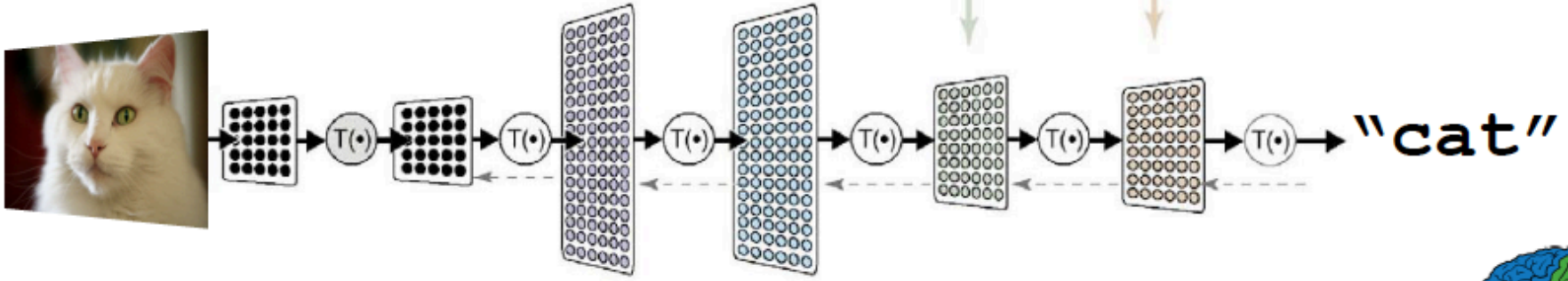
- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple,
  trainable mathematical functions
- Compatible with many variants of machine learning



What is Deep Learning?

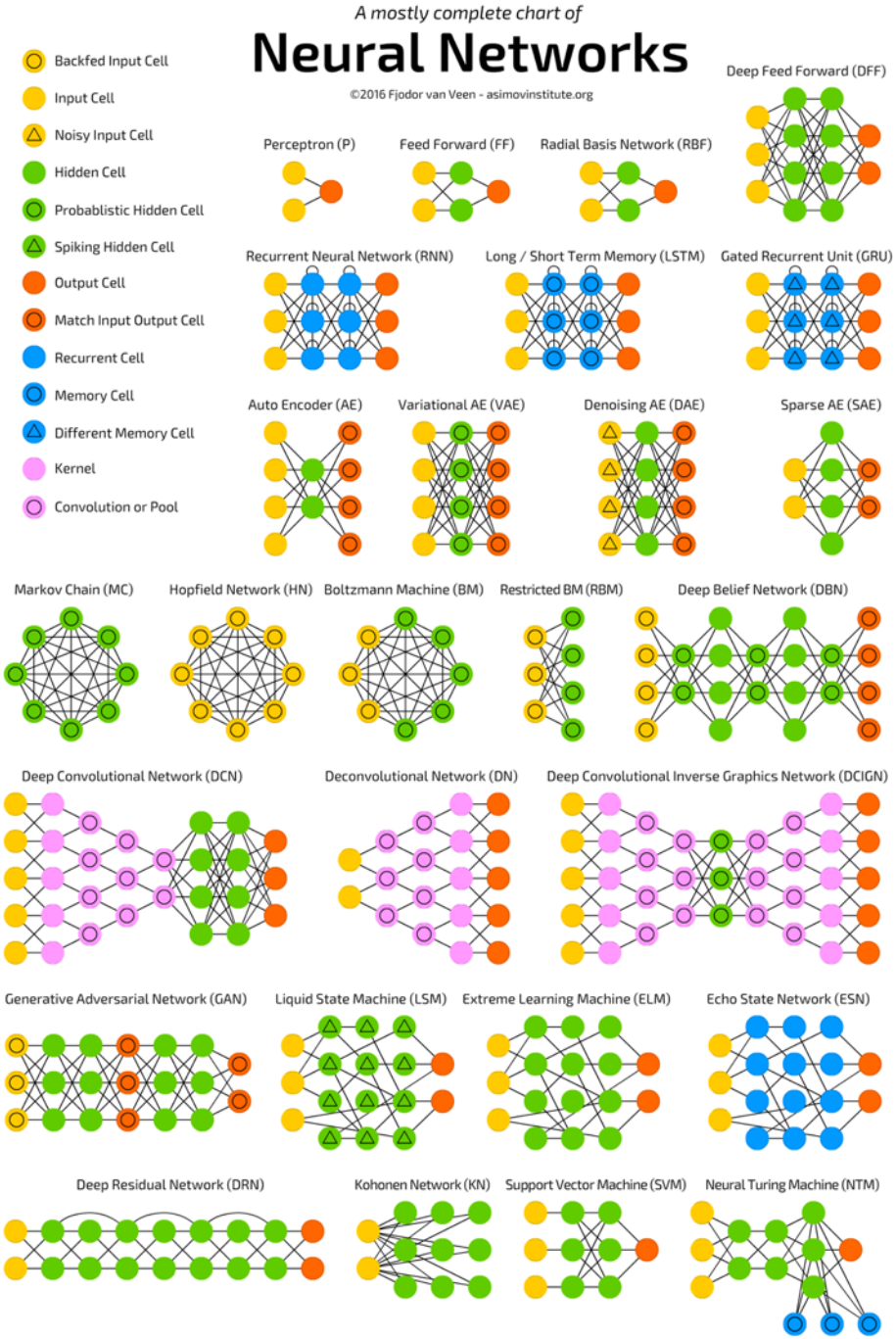
- Loosely based on (what little) we know about the brain





Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

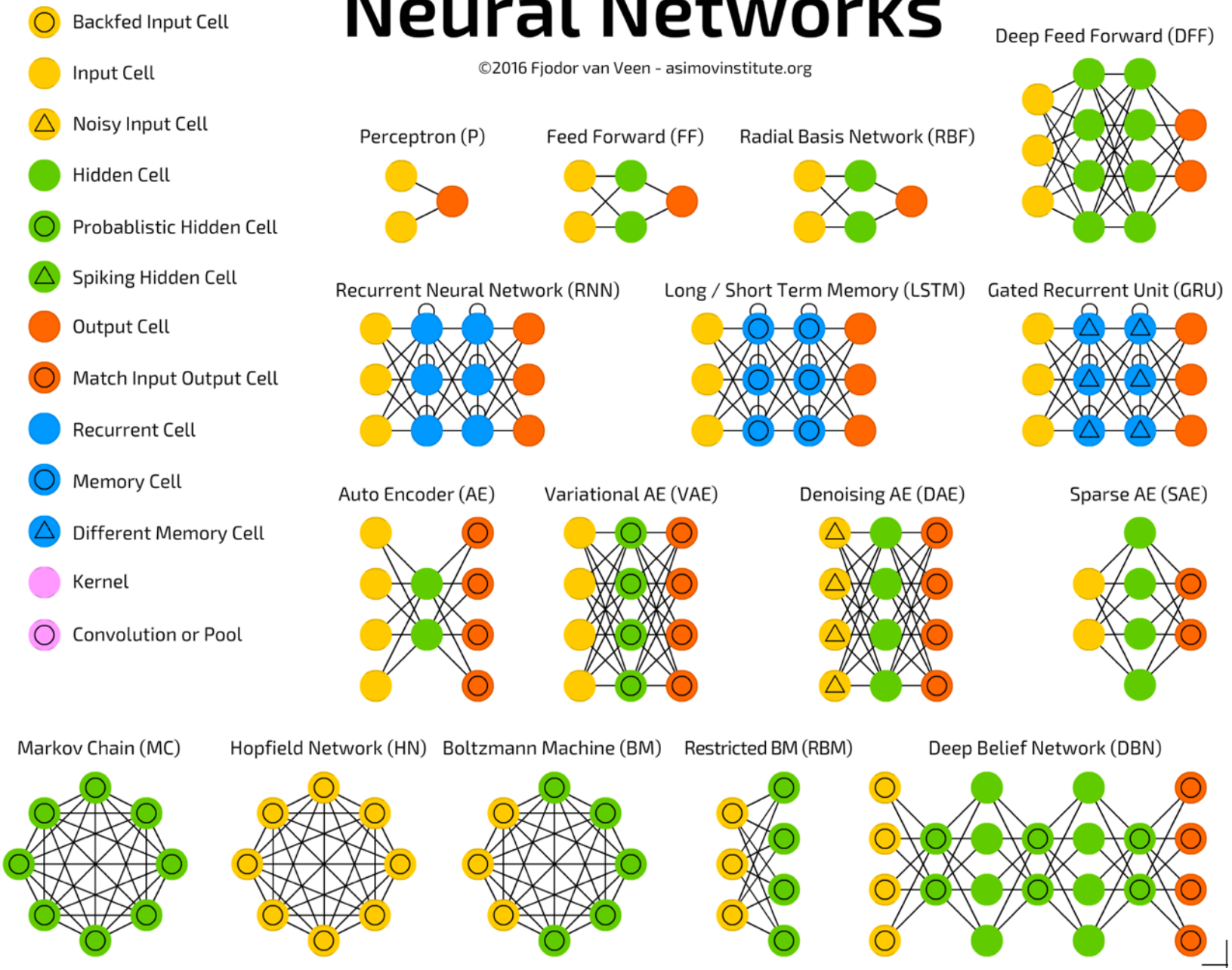
Neural Networks (NN)



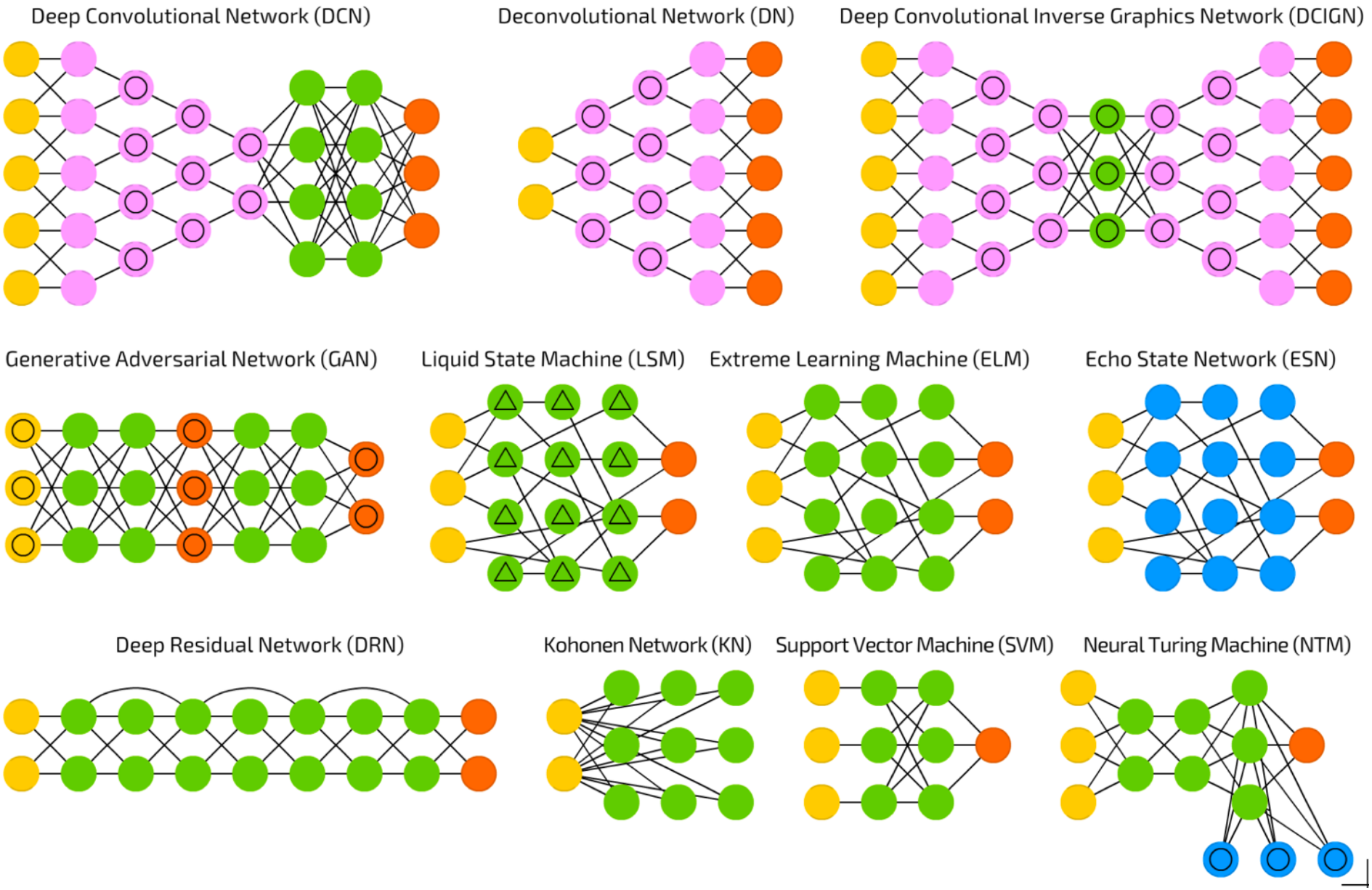
Source: http://www.asimovinstitute.org/neural-network-zoo/

A mostly complete chart of

Neural Networks



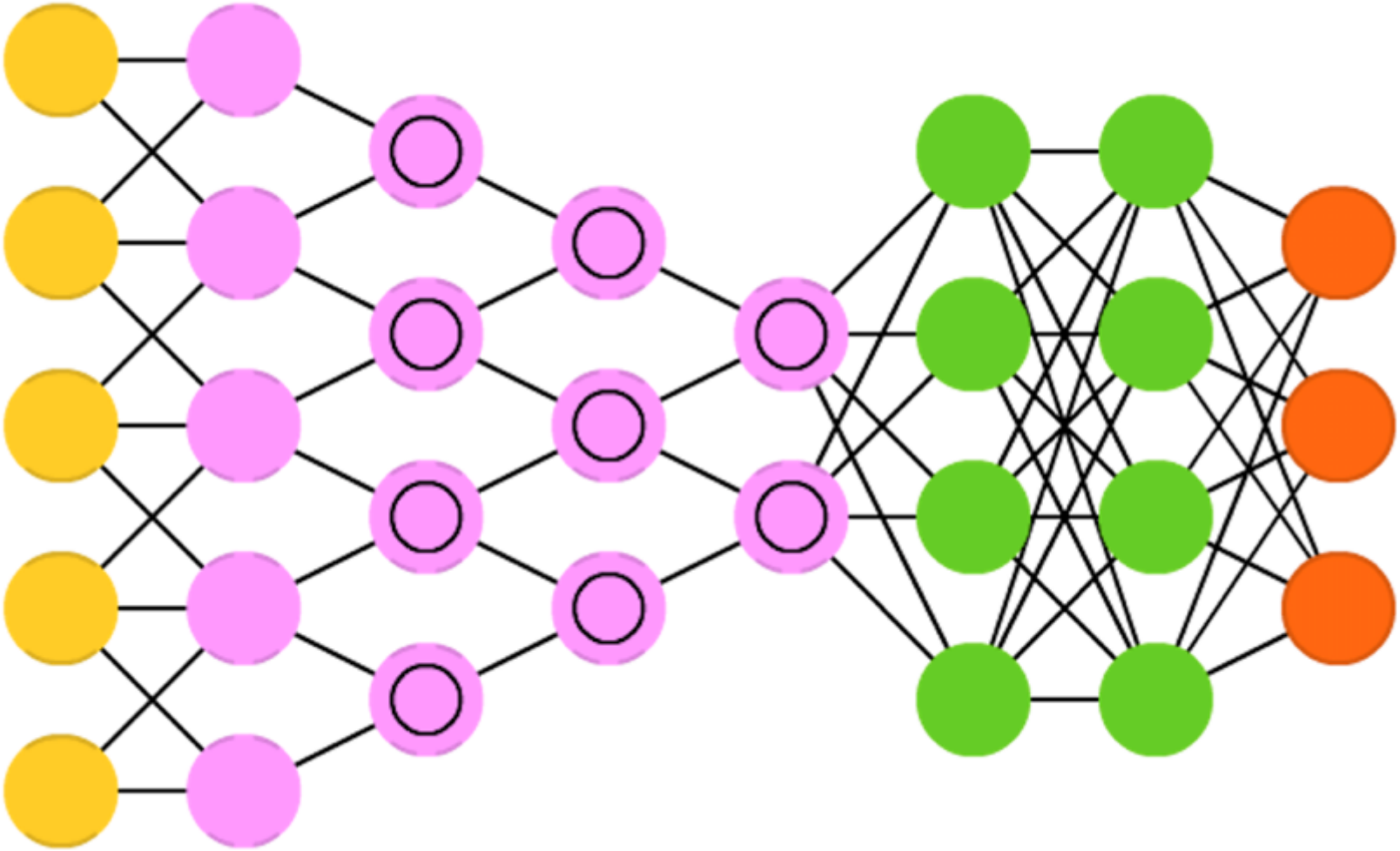
Source: http://www.asimovinstitute.org/neural-network-zoo/



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

Convolutional Neural Networks

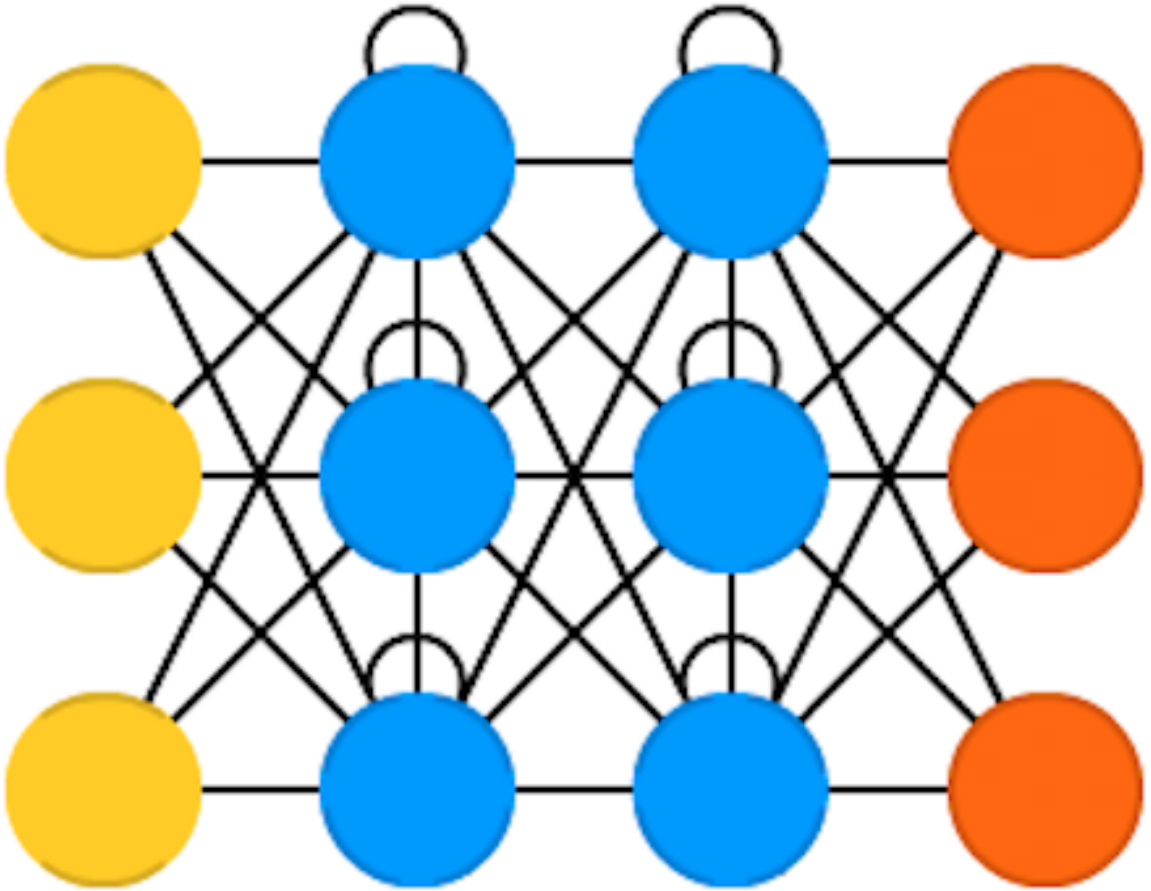
(CNN or Deep Convolutional Neural Networks, DCNN)



LeCun, Yann, et al. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86.11 (1998): 2278-2324.

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

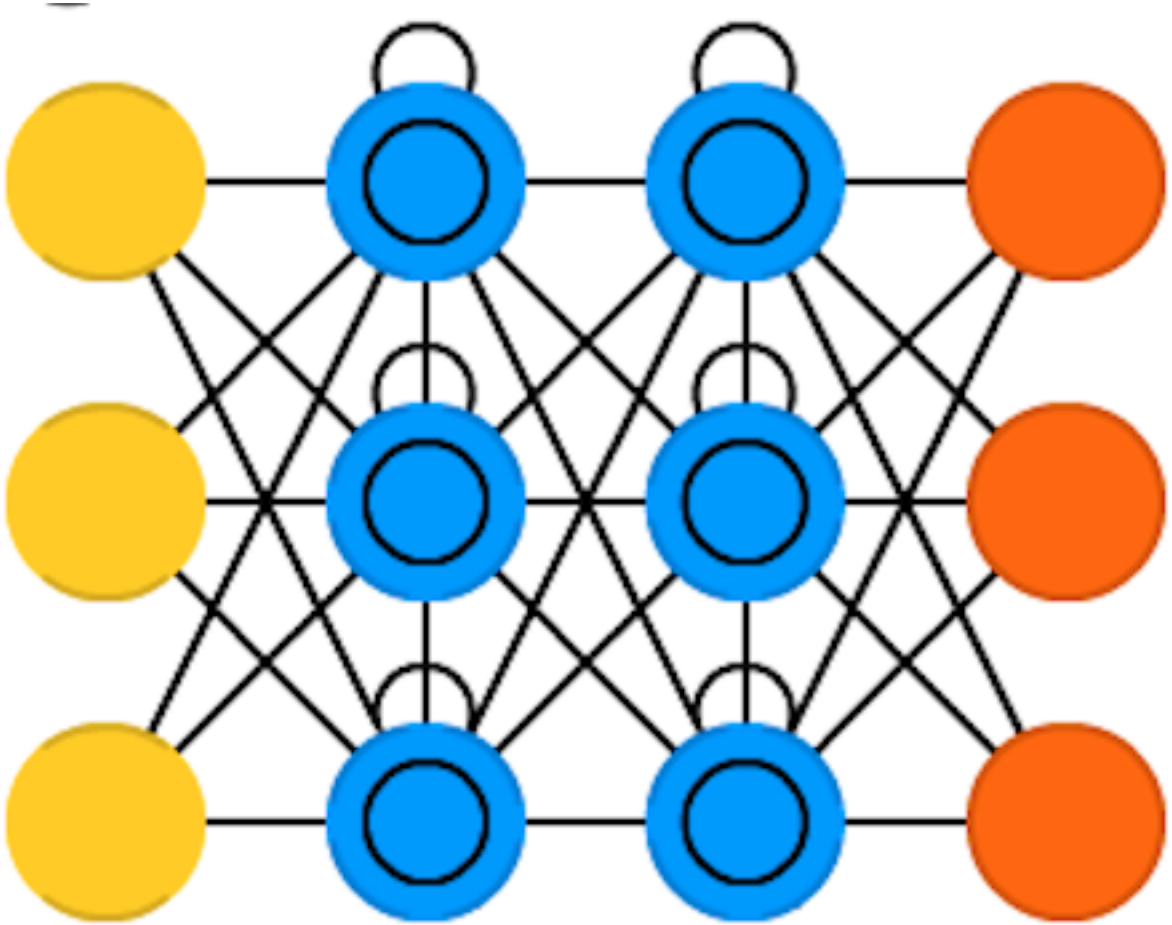
Recurrent Neural Networks (RNN)



Elman, Jeffrey L. “Finding structure in time.” Cognitive science 14.2 (1990): 179-211

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

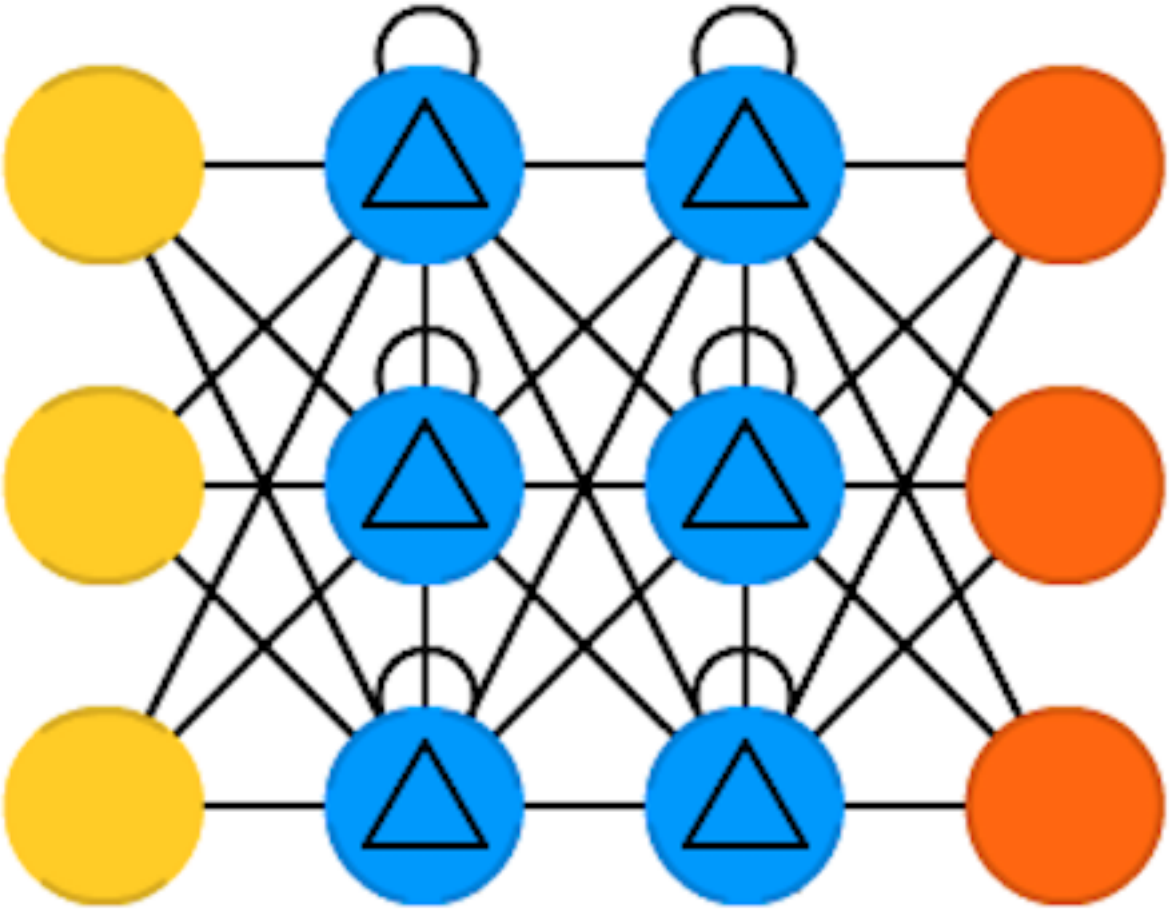
Long / Short Term Memory (LSTM)



Hochreiter, Sepp, and Jürgen Schmidhuber. “Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

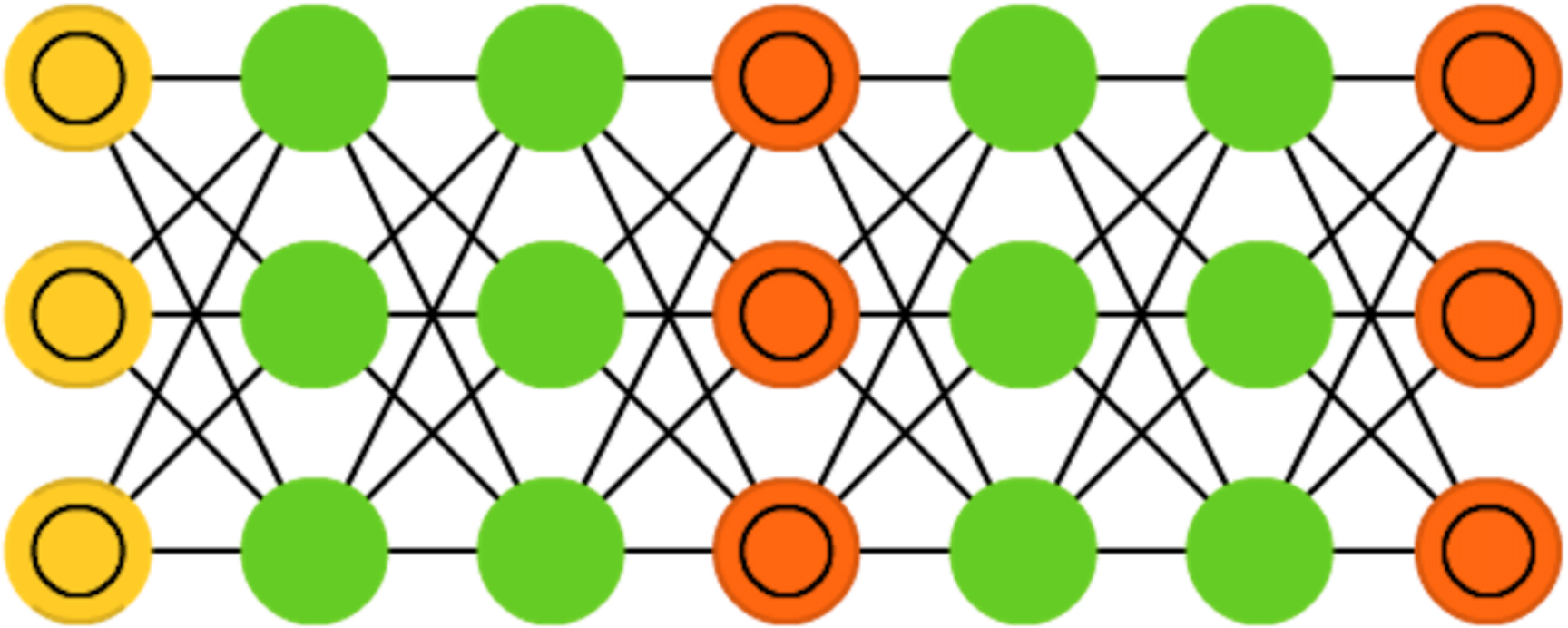
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



Chung, Junyoung, et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling.” arXiv preprint arXiv:1412.3555 (2014).
Source: http://www.asimovinstitute.org/neural-network-zoo/

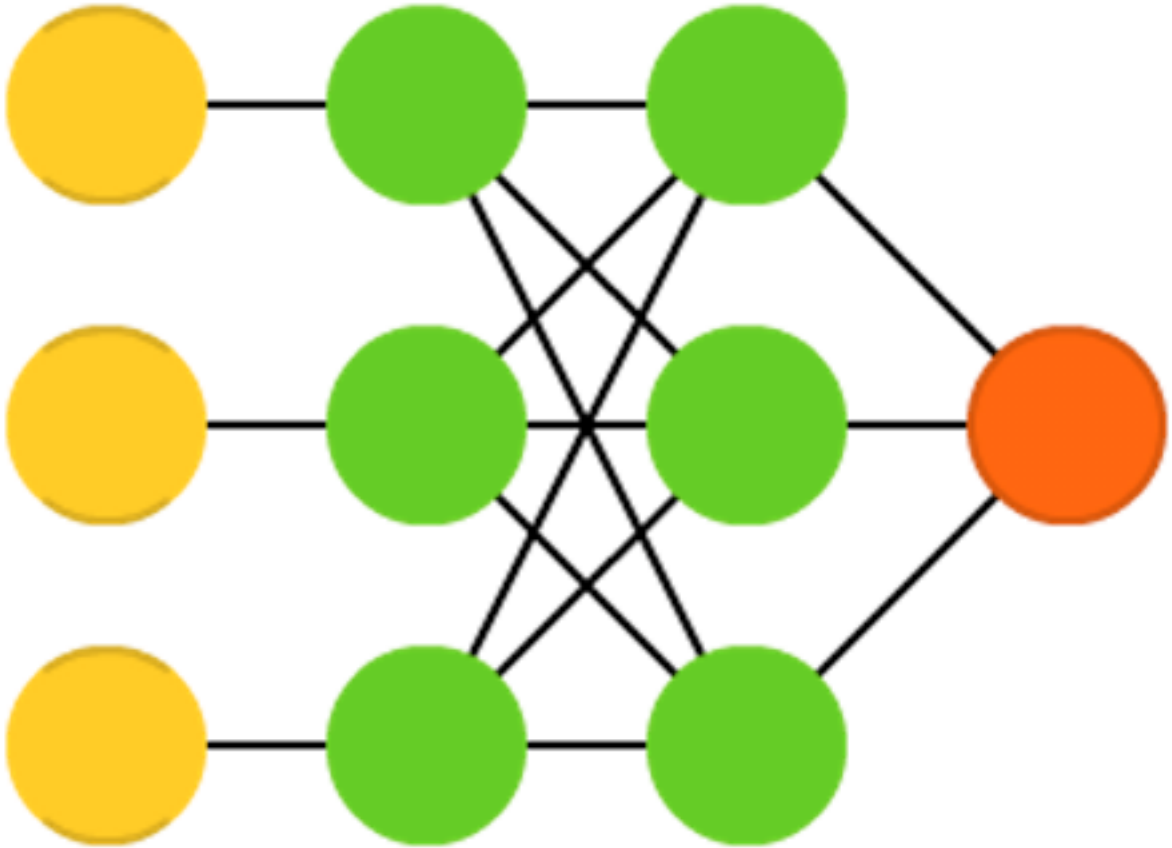
Generative Adversarial Networks (GAN)



Goodfellow, Ian, et al. “Generative adversarial nets.” Advances in Neural Information Processing Systems. 2014.

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

Support Vector Machines (SVM)



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

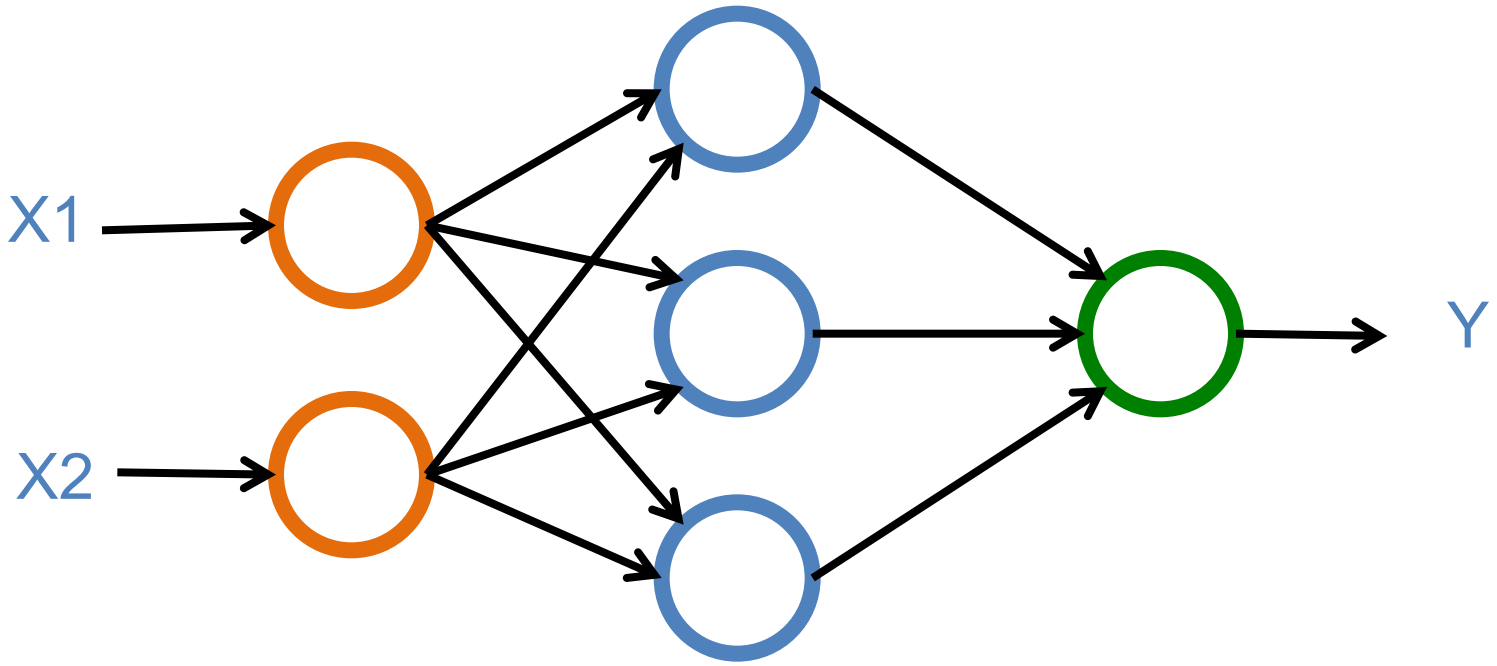
Source: http://www.asimovinstitute.org/neural-network-zoo/

Neural networks (NN) 1960

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Multilayer Perceptrons (MLP) 1985

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Support Vector Machine (SVM) 1995

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Hinton presents the

Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST

2005

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Deep Recurrent Neural Network (RNN) 2009

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Convolutional DBN 2010

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Max-Pooling CDBN 2011

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Deep Learning

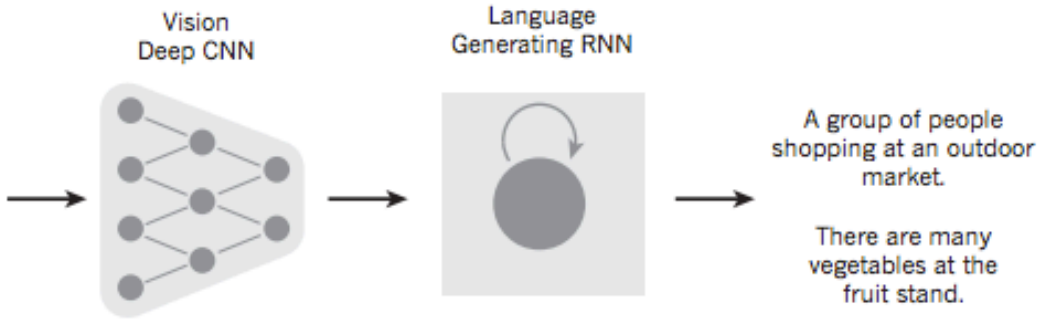
Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng

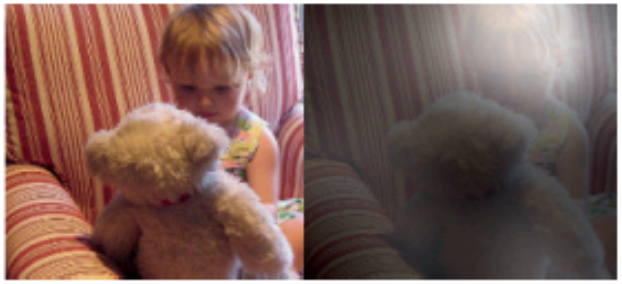
From image to text



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.

Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

From image to text

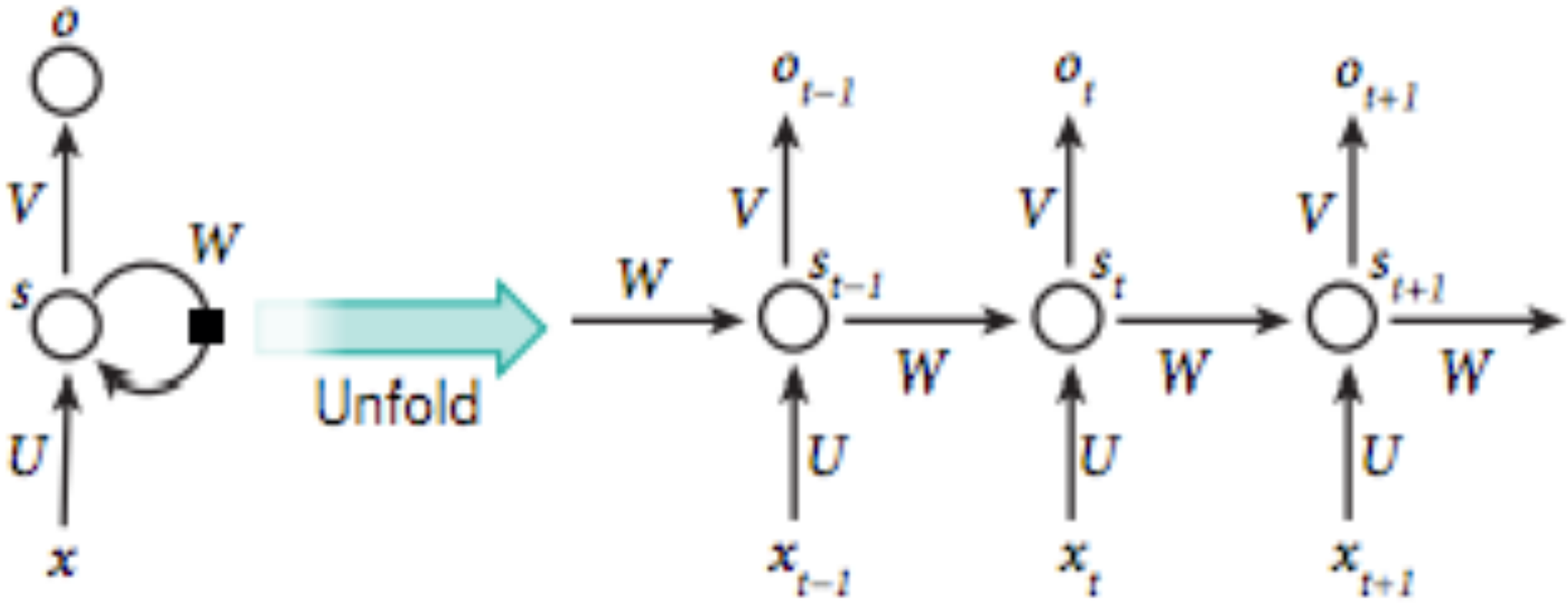
Image: deep convolution neural network (CNN)

Text: recurrent neural network (RNN)

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

Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

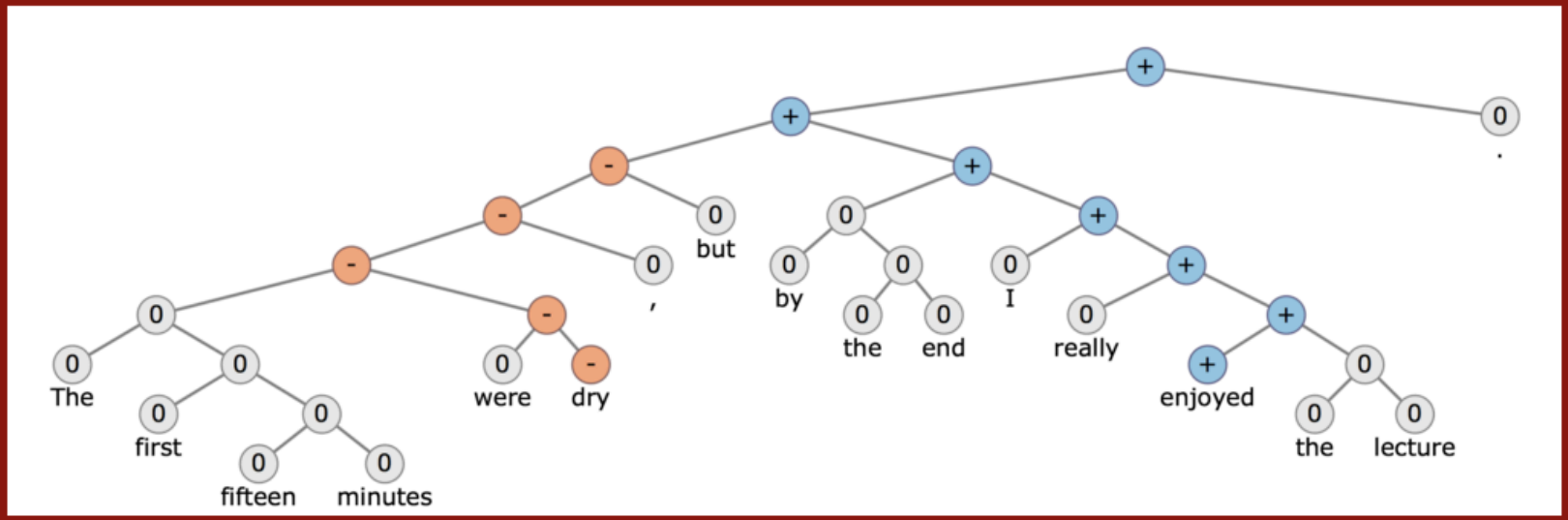
Recurrent Neural Network (RNN)



Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

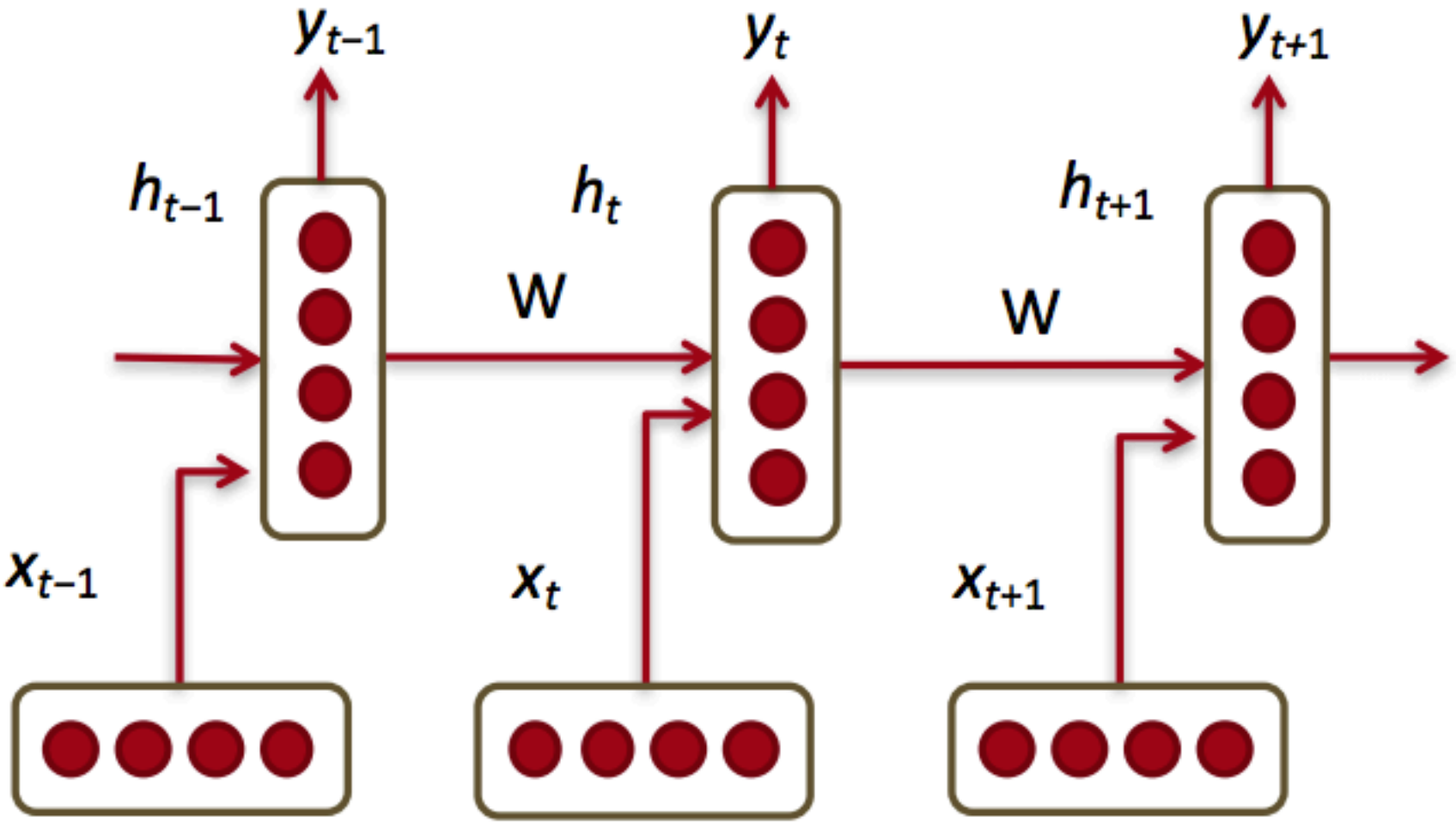


Course Description

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,

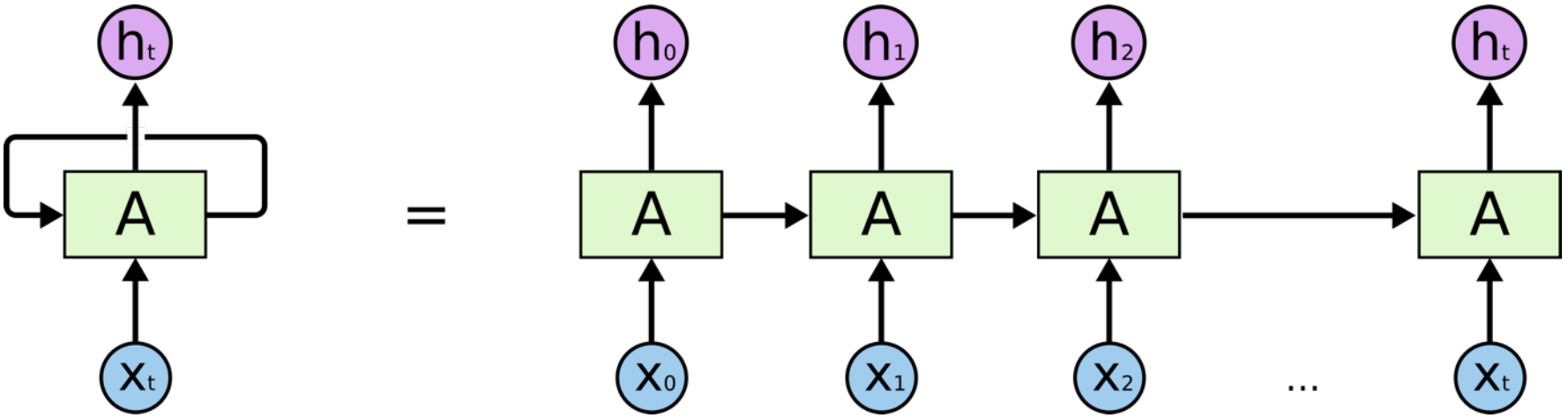
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



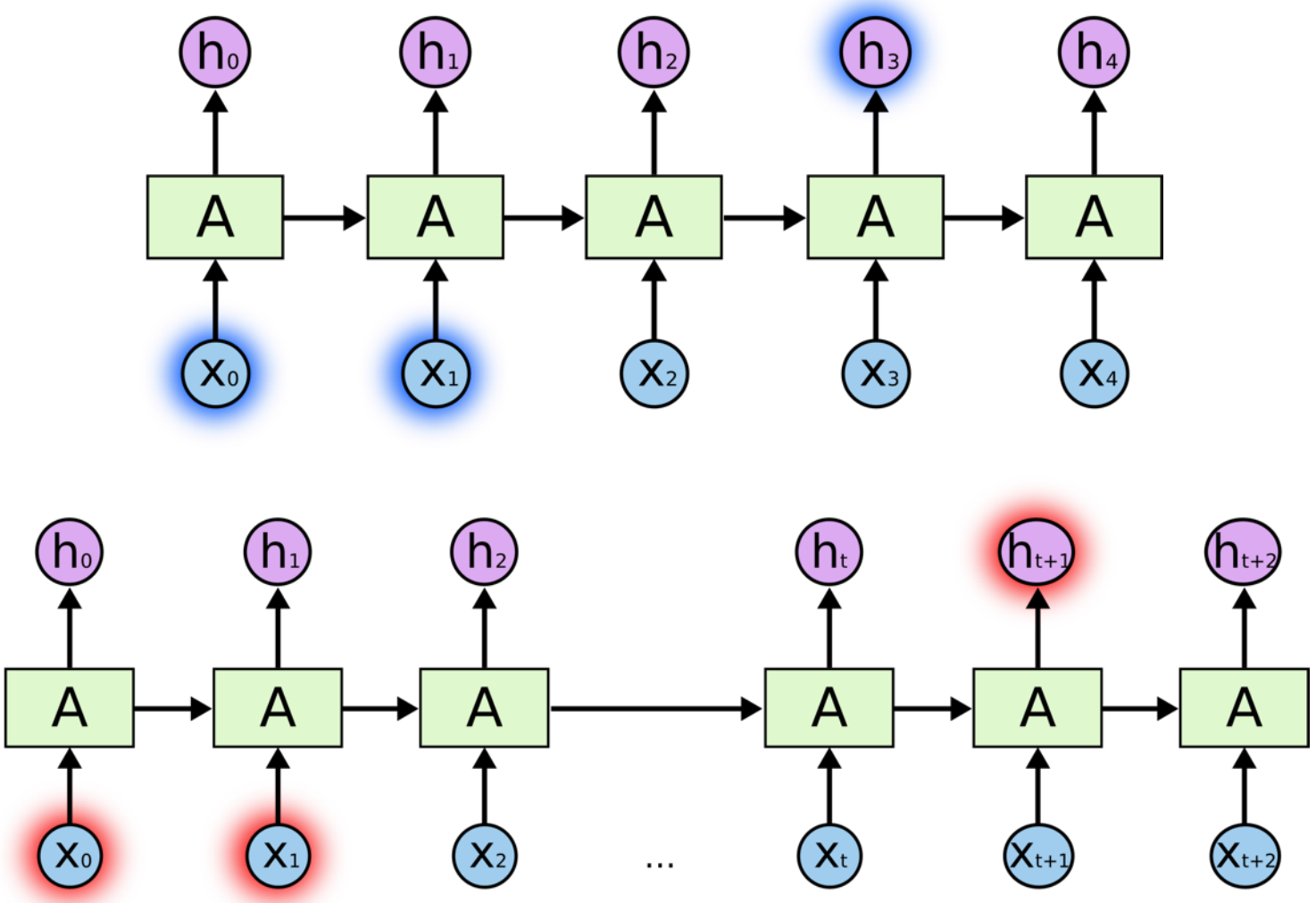
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

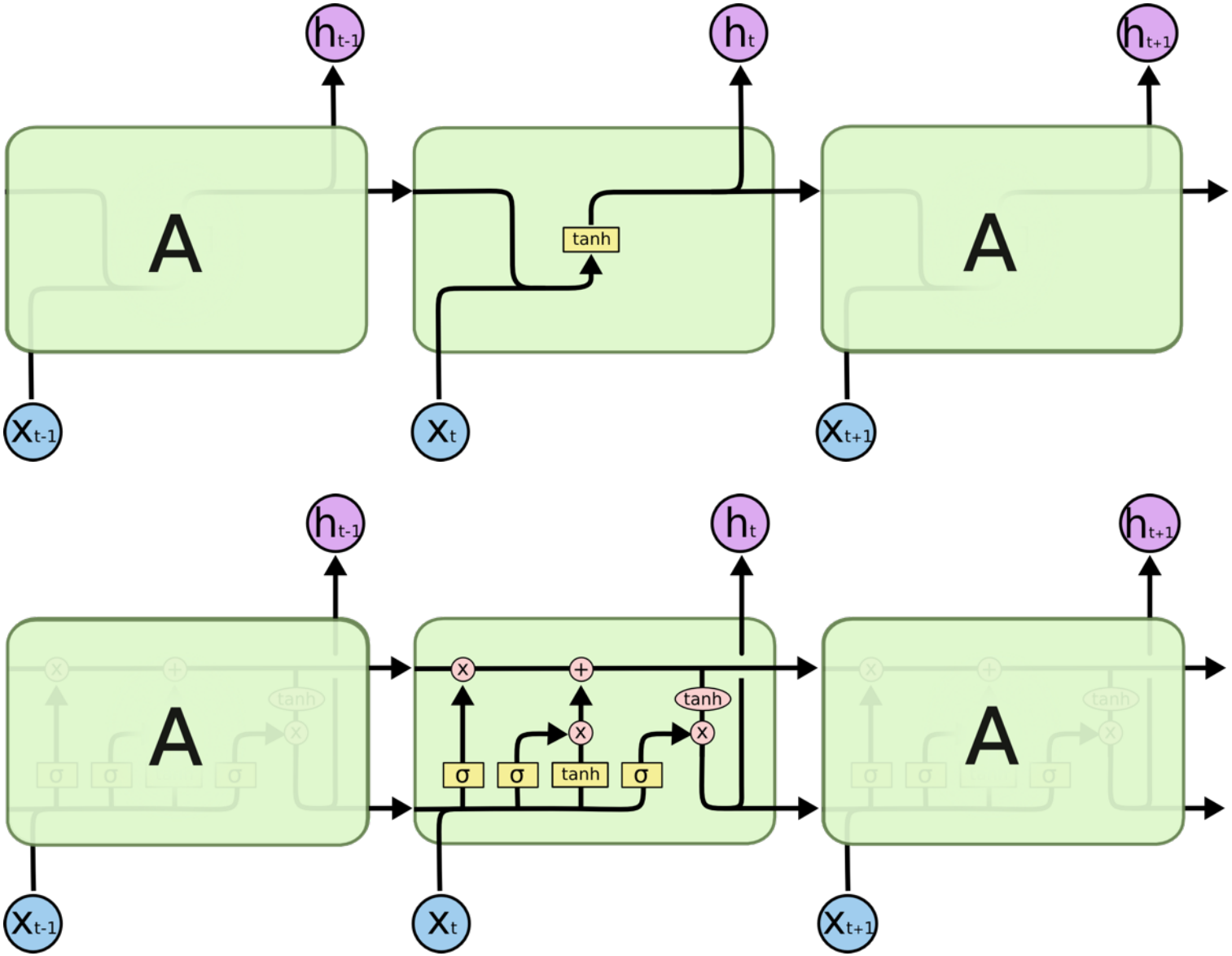
RNN long-term dependencies



I grew up in France… I speak fluent French.

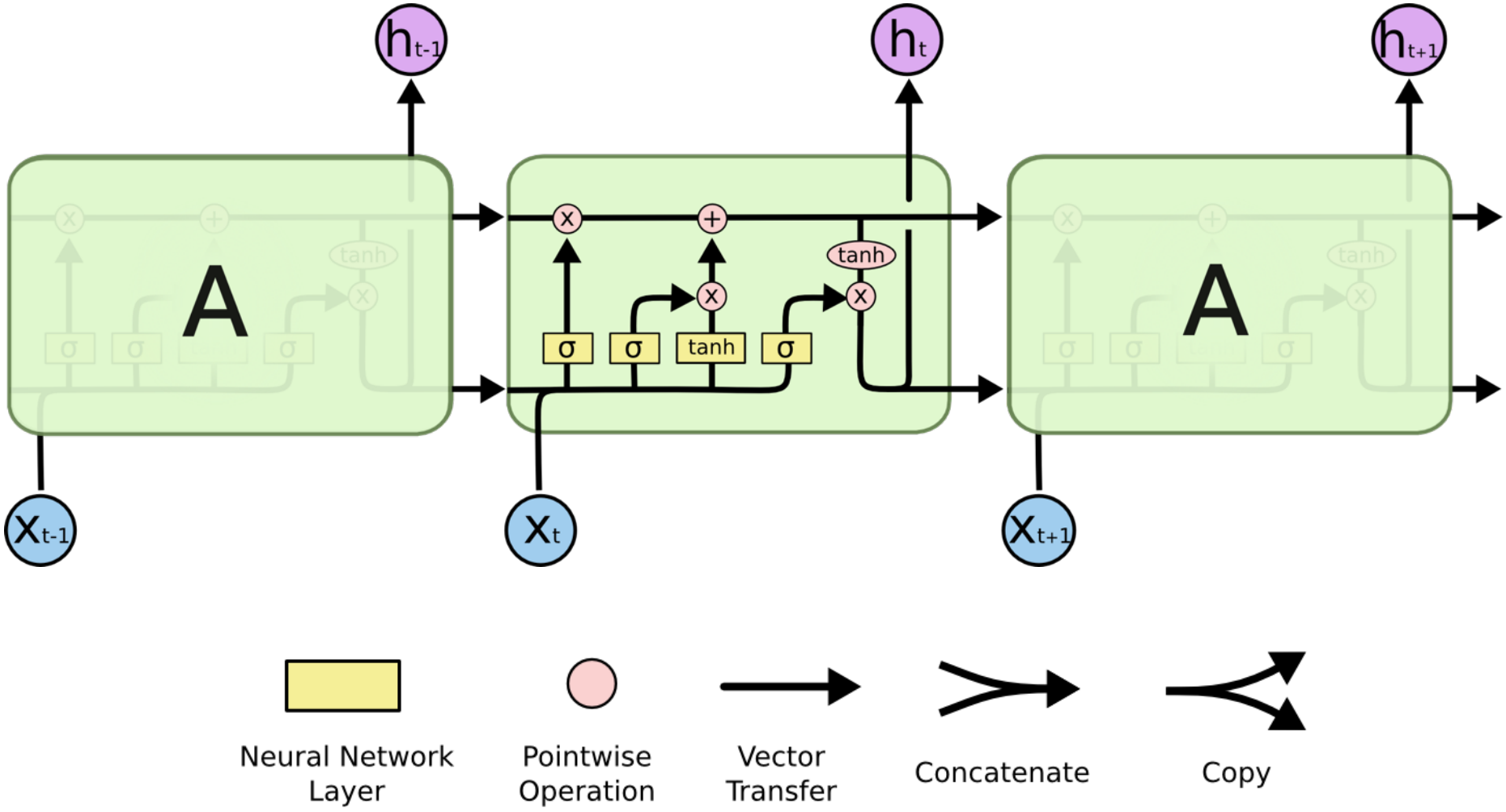
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN LSTM



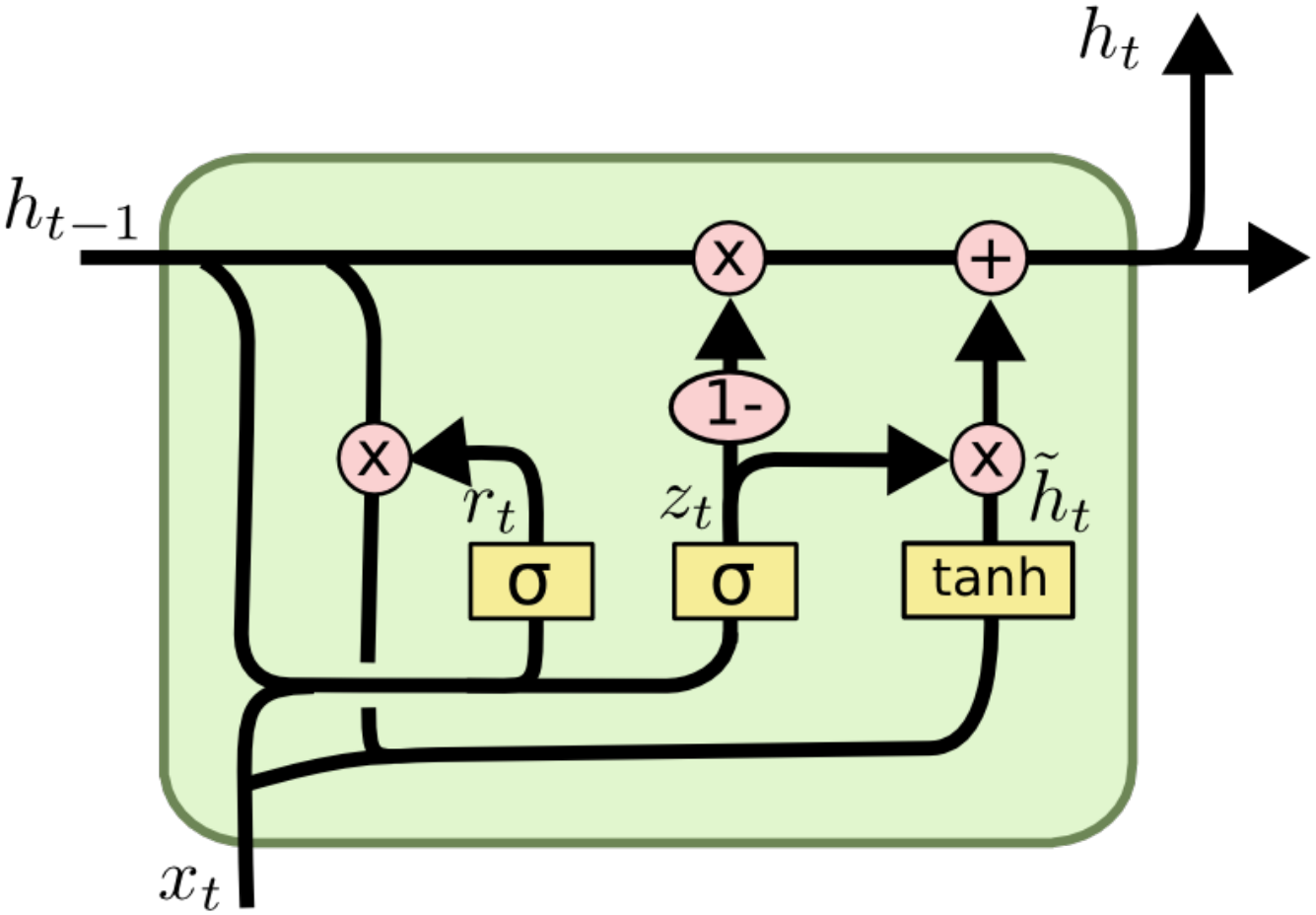
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



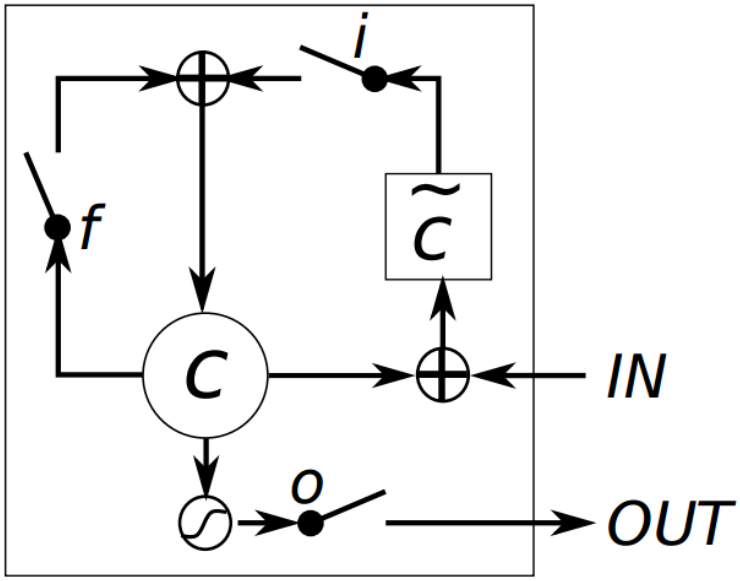
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

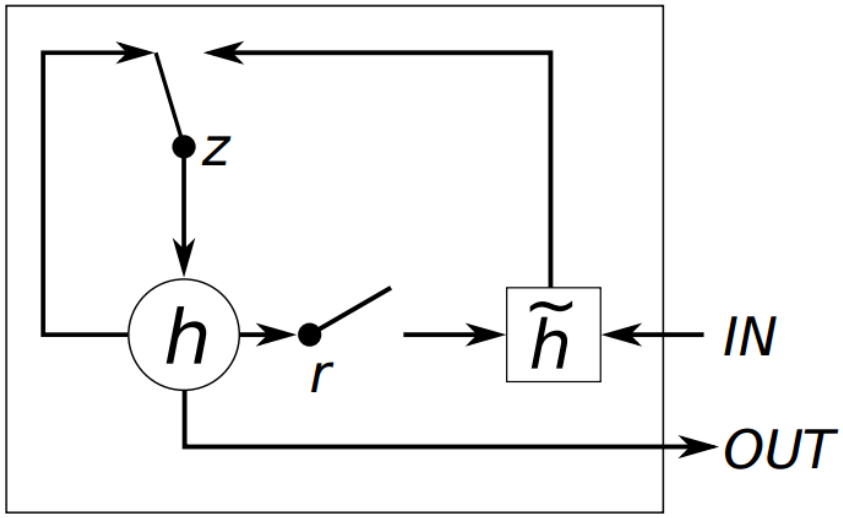
Gated Recurrent Unit (GRU)



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM vs GRU





LSTM

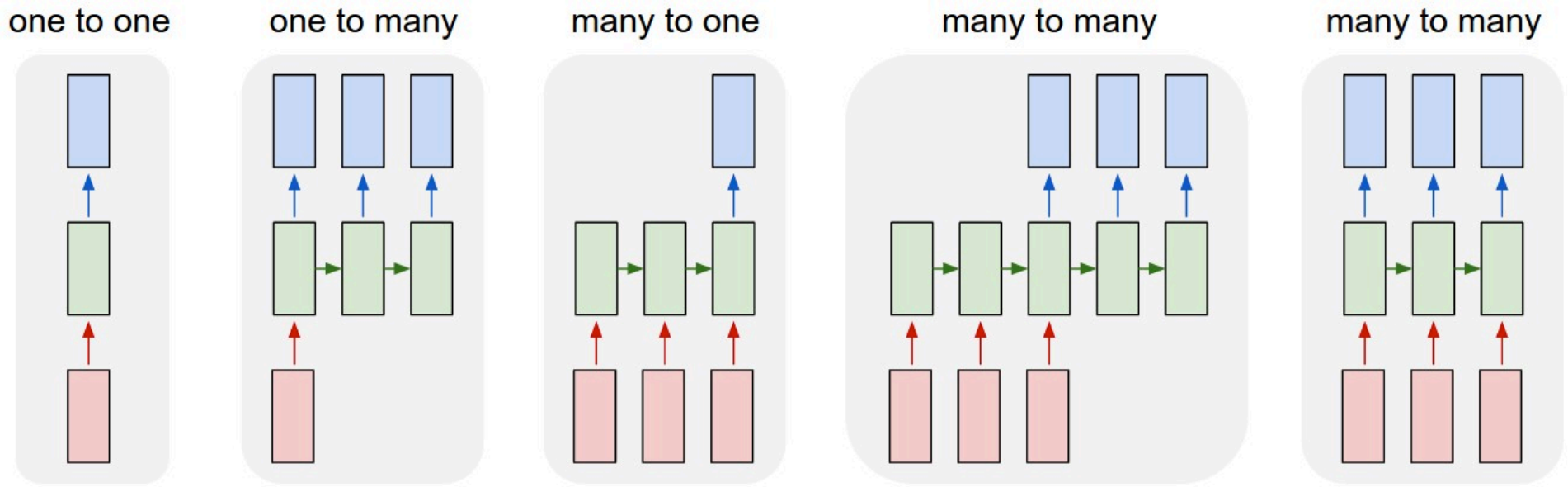
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.

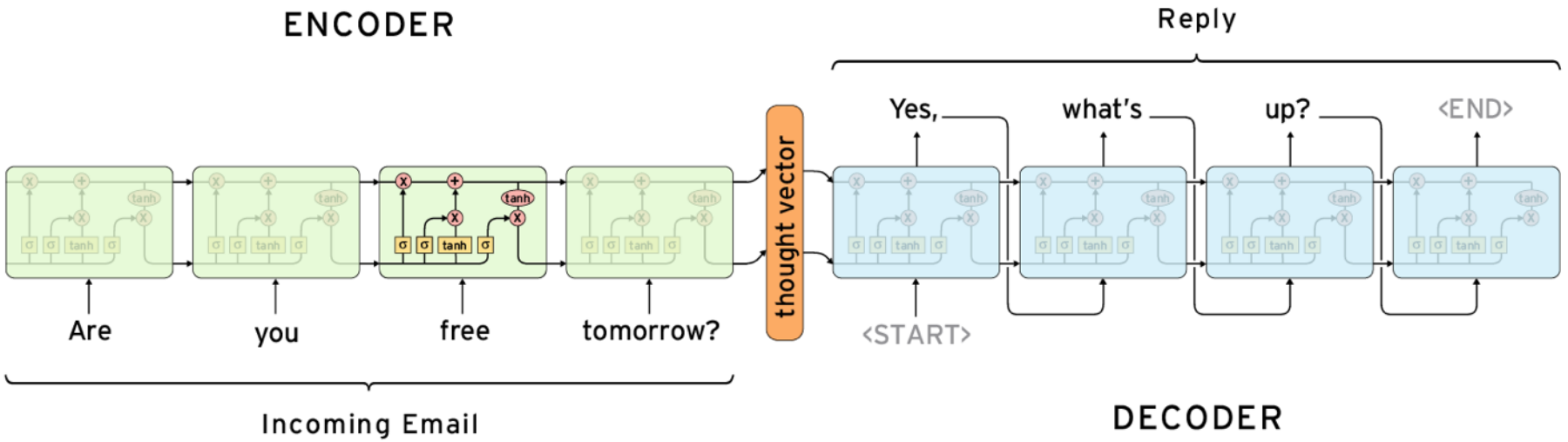
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

LSTM Recurrent Neural Network



Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

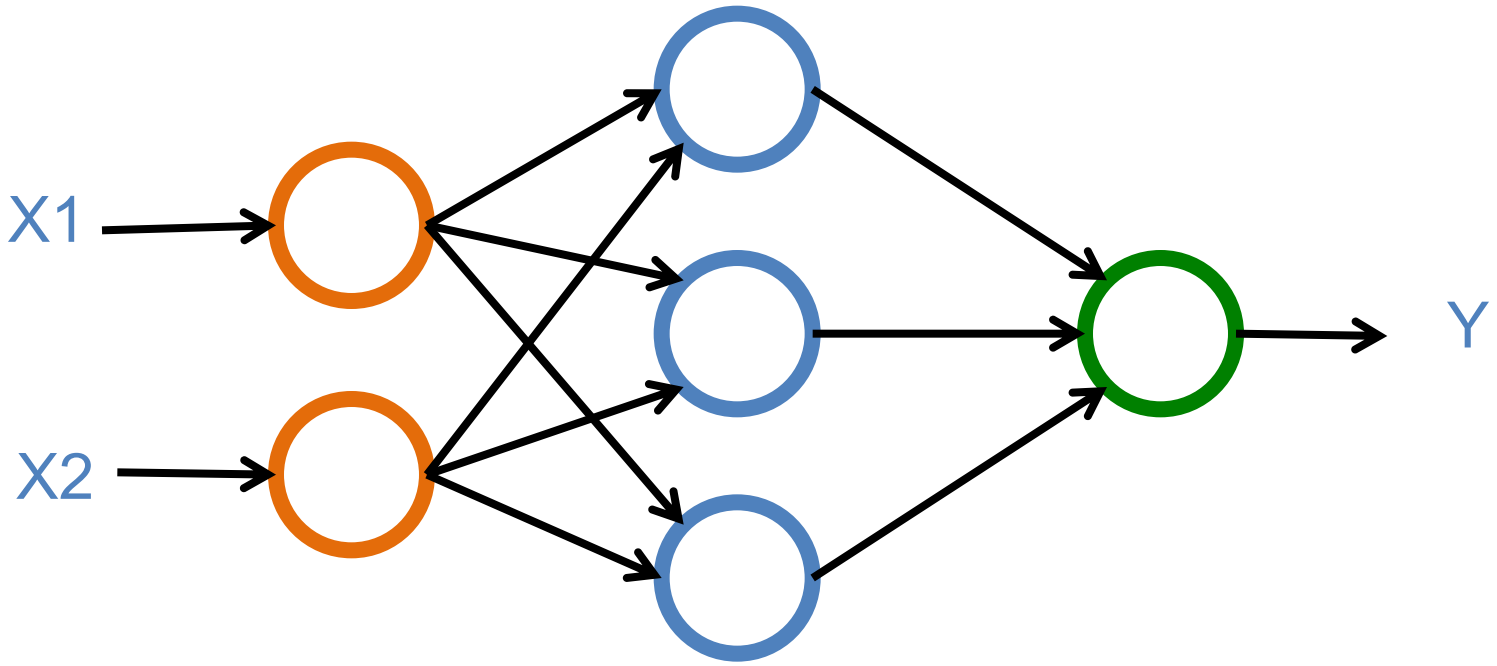
The Sequence to Sequence model (seq2seq)



Source: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

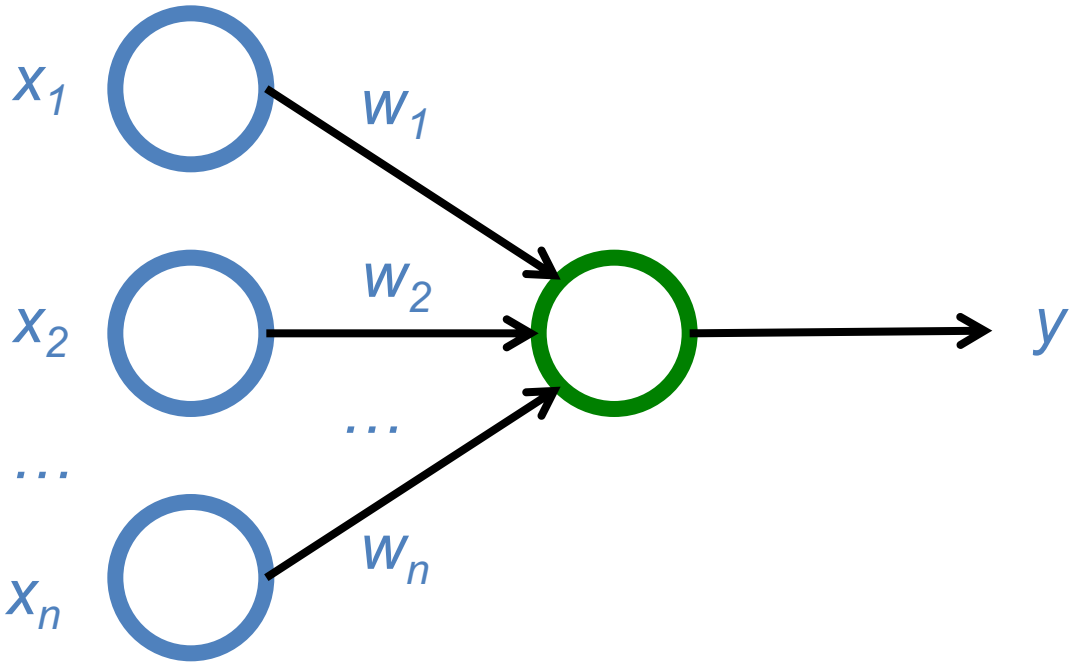
Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



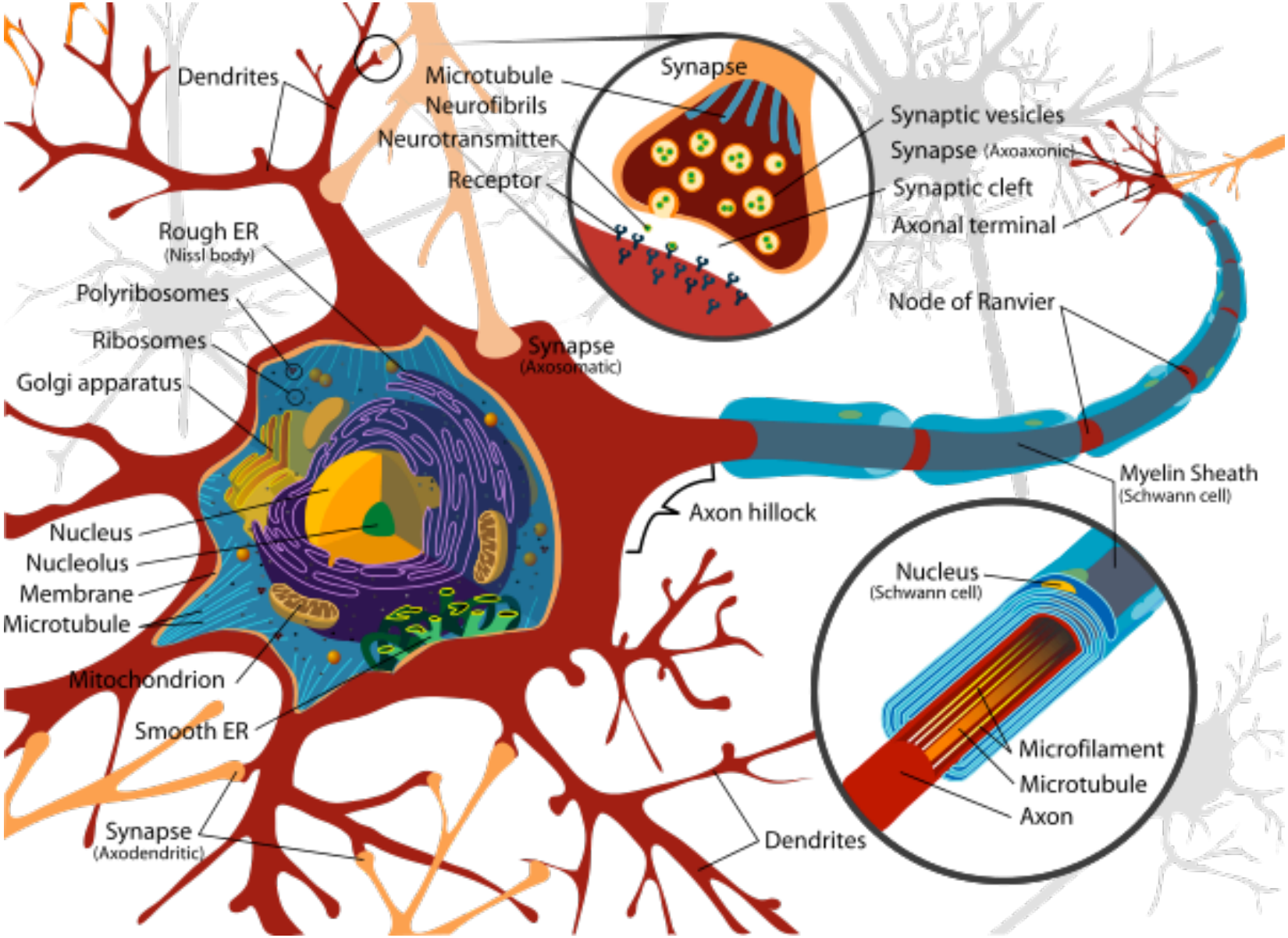
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

The Neuron



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

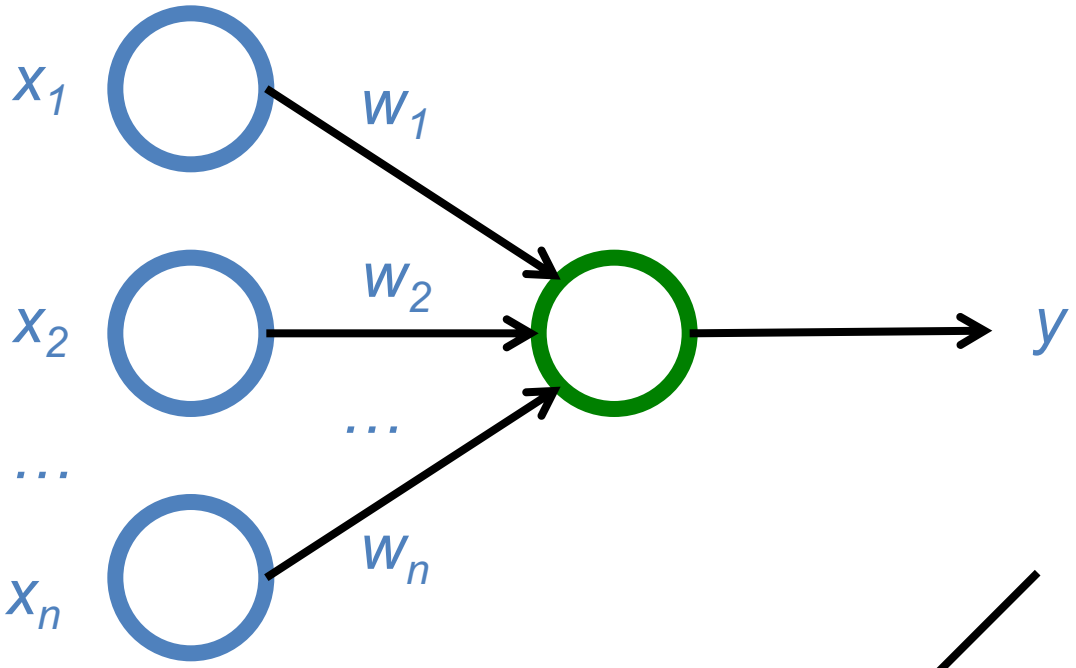
Neuron and Synapse

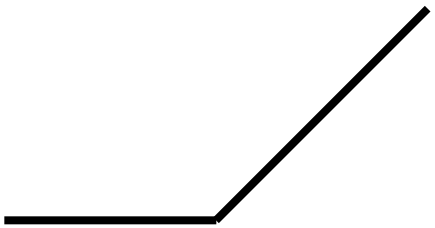


Source: https://en.wikipedia.org/wiki/Neuron

The Neuron

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

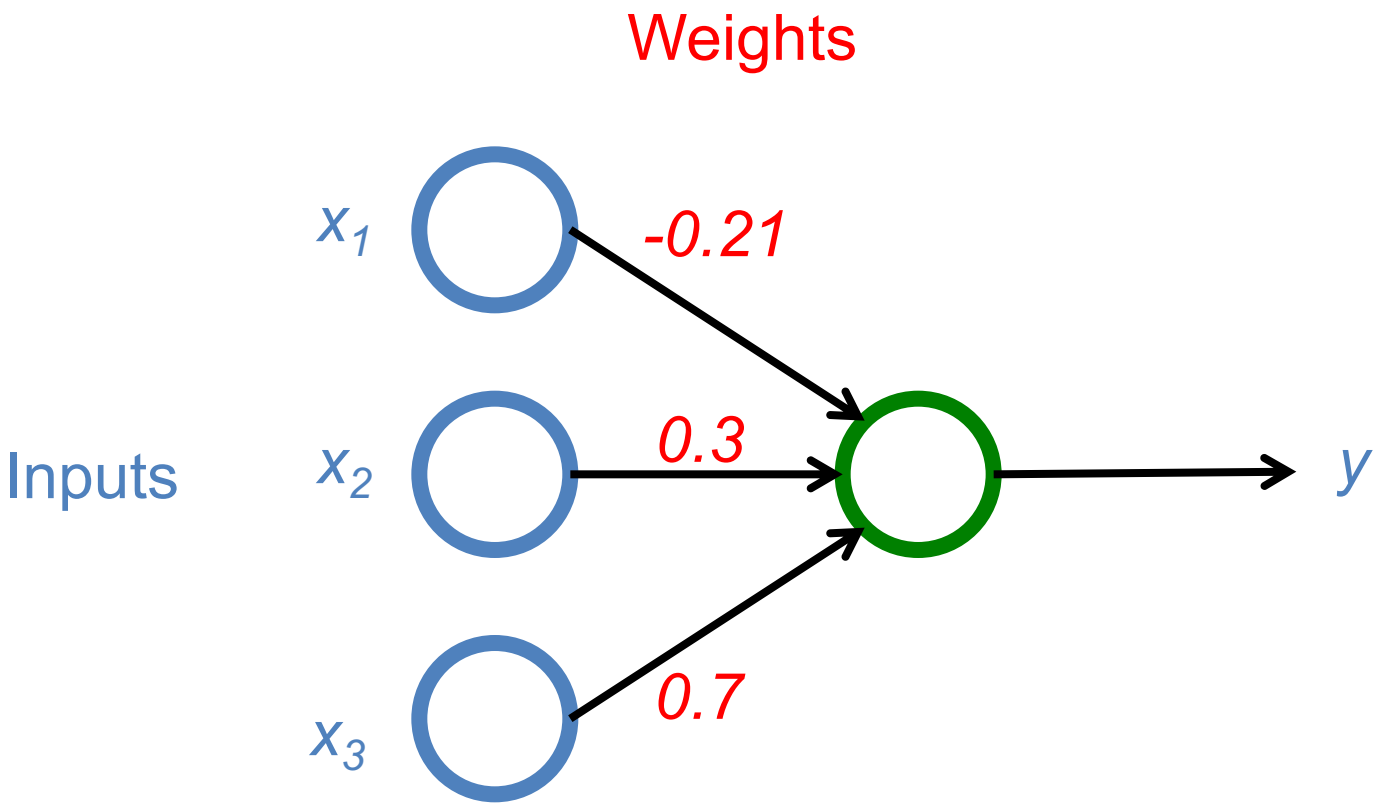




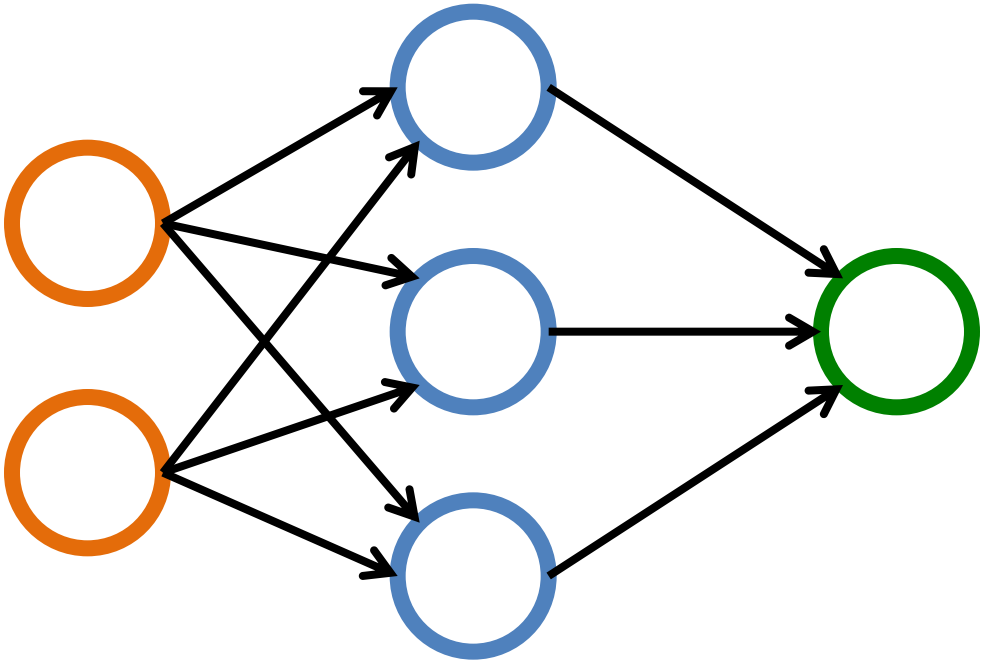
$$F(x) = \max(0, x)$$

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

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



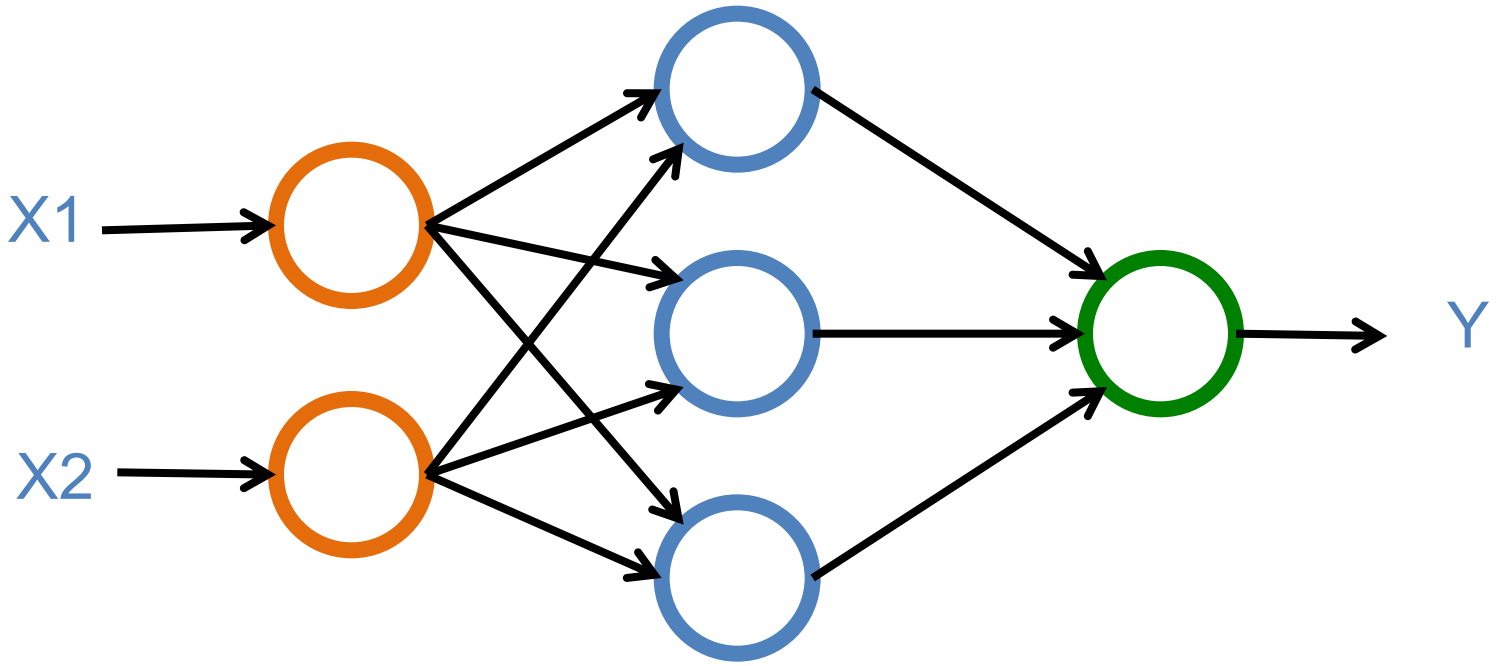
Neural Networks



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

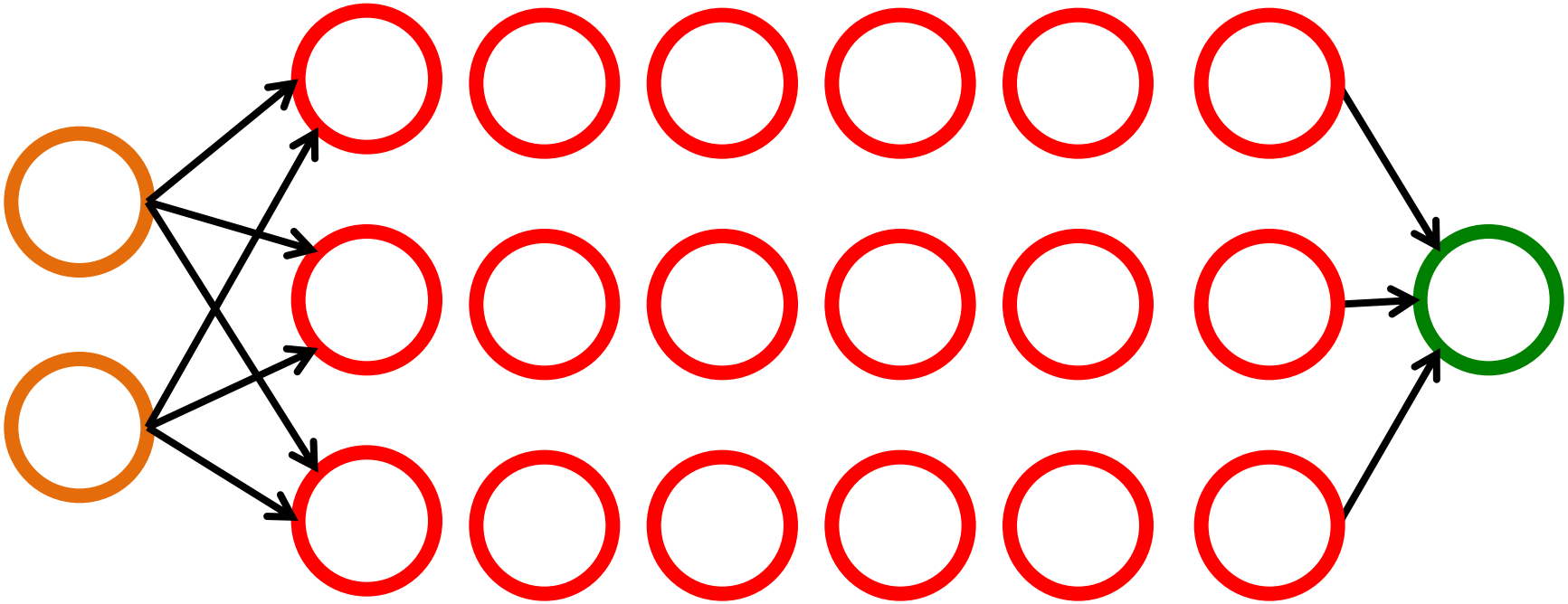
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

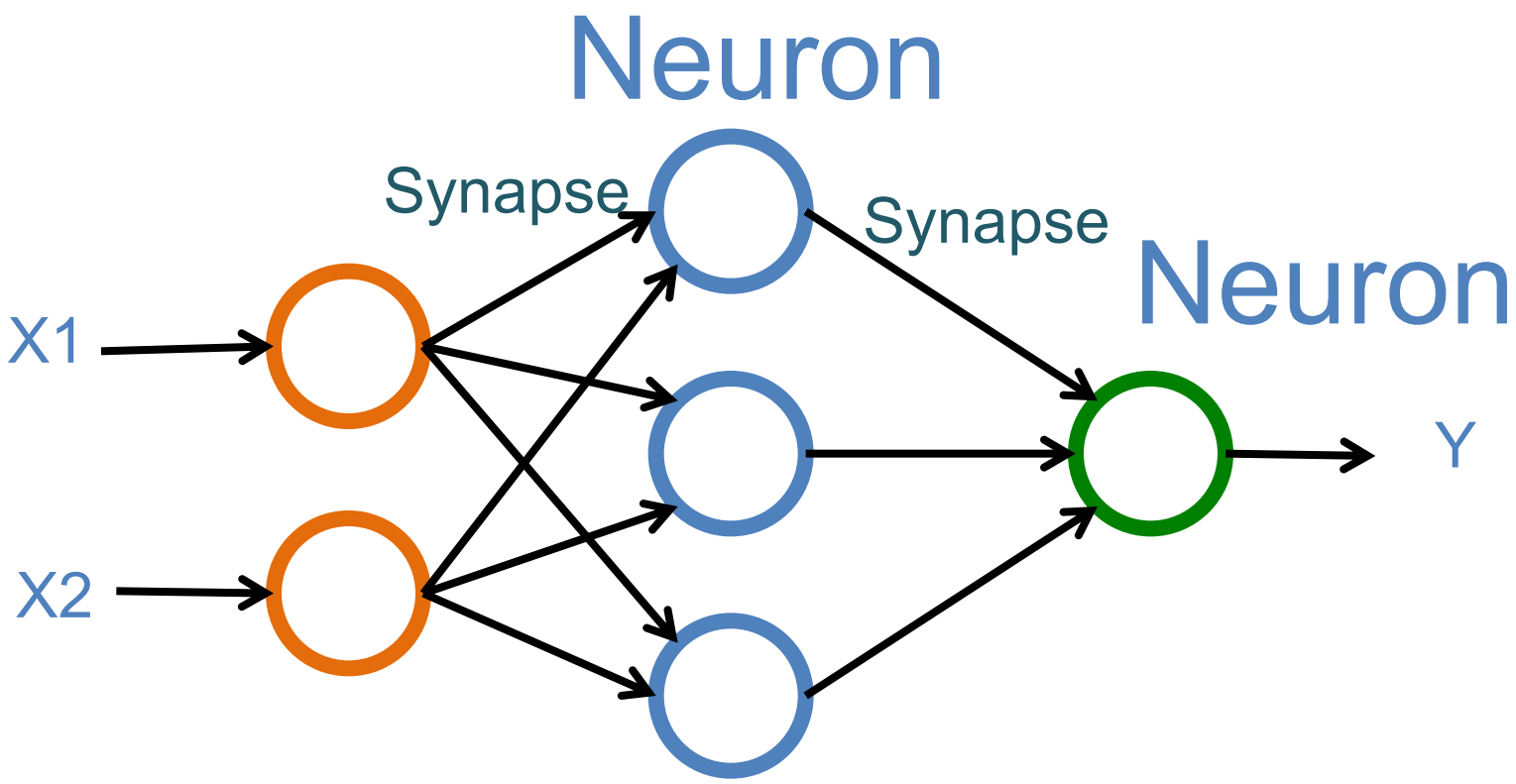
Deep Neural Networks
Deep Learning



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

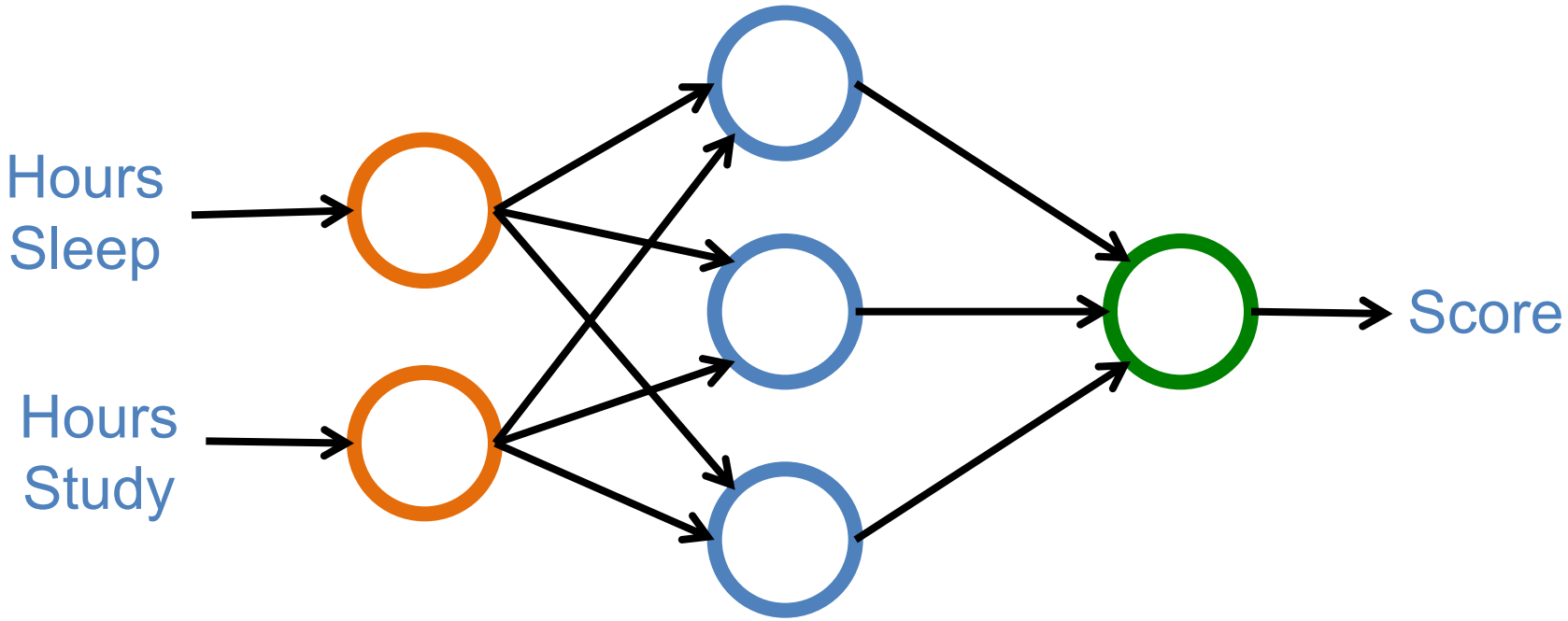
Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



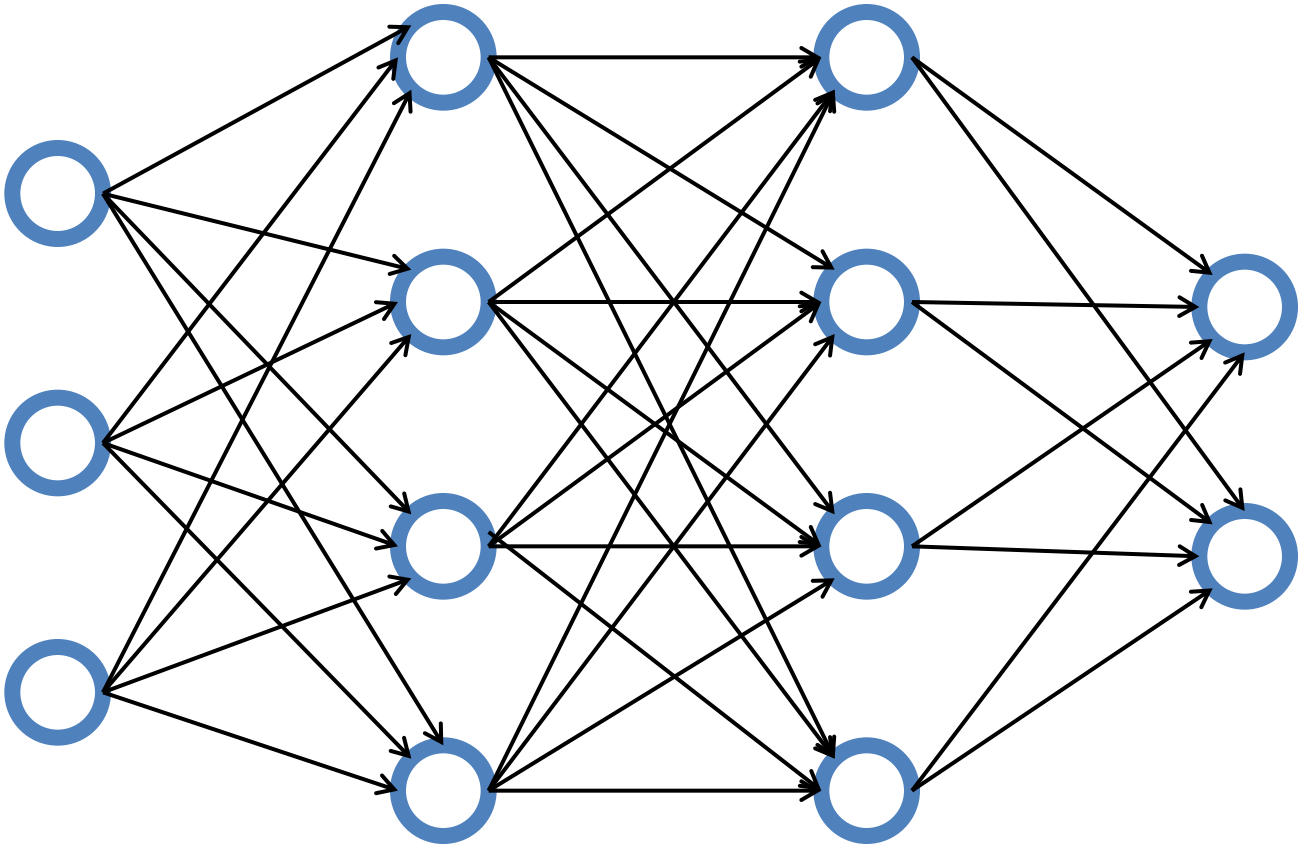
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X)

Hidden Layer (H)

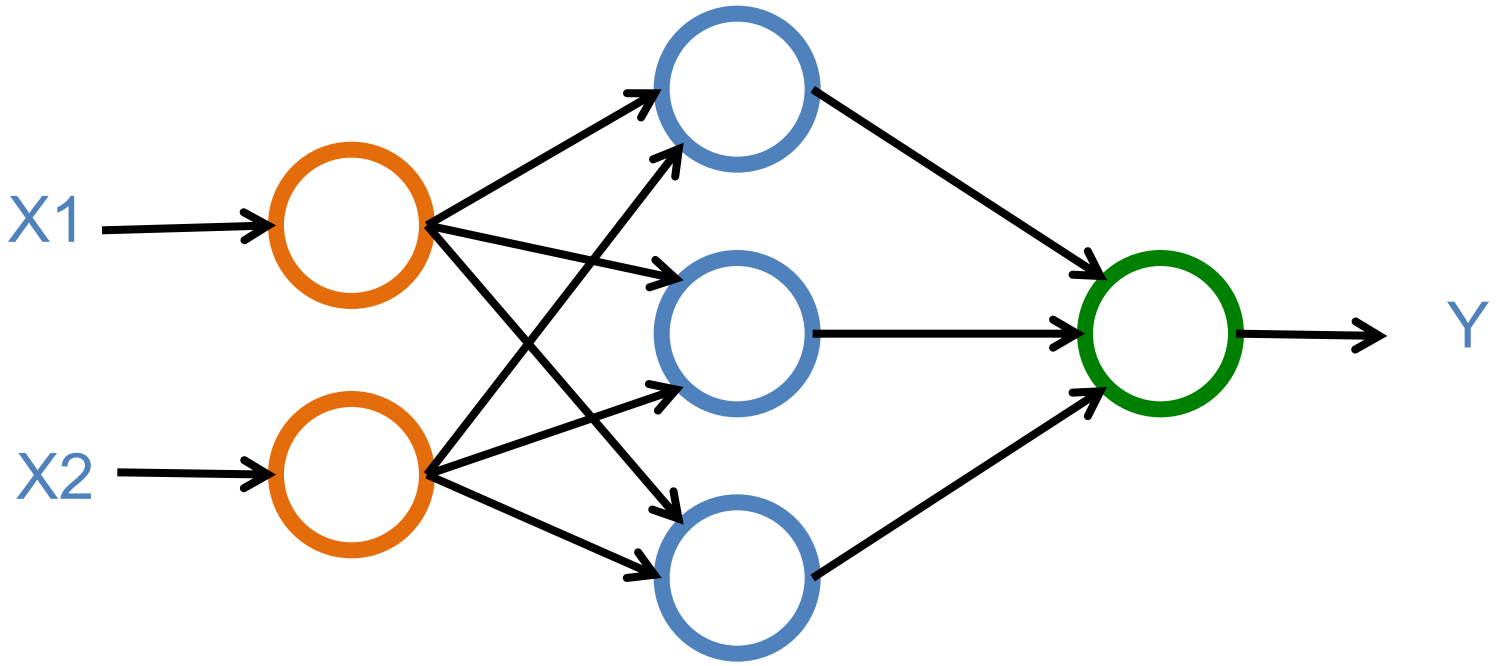
Output Layer (Y)



Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2

Neural Networks

Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

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

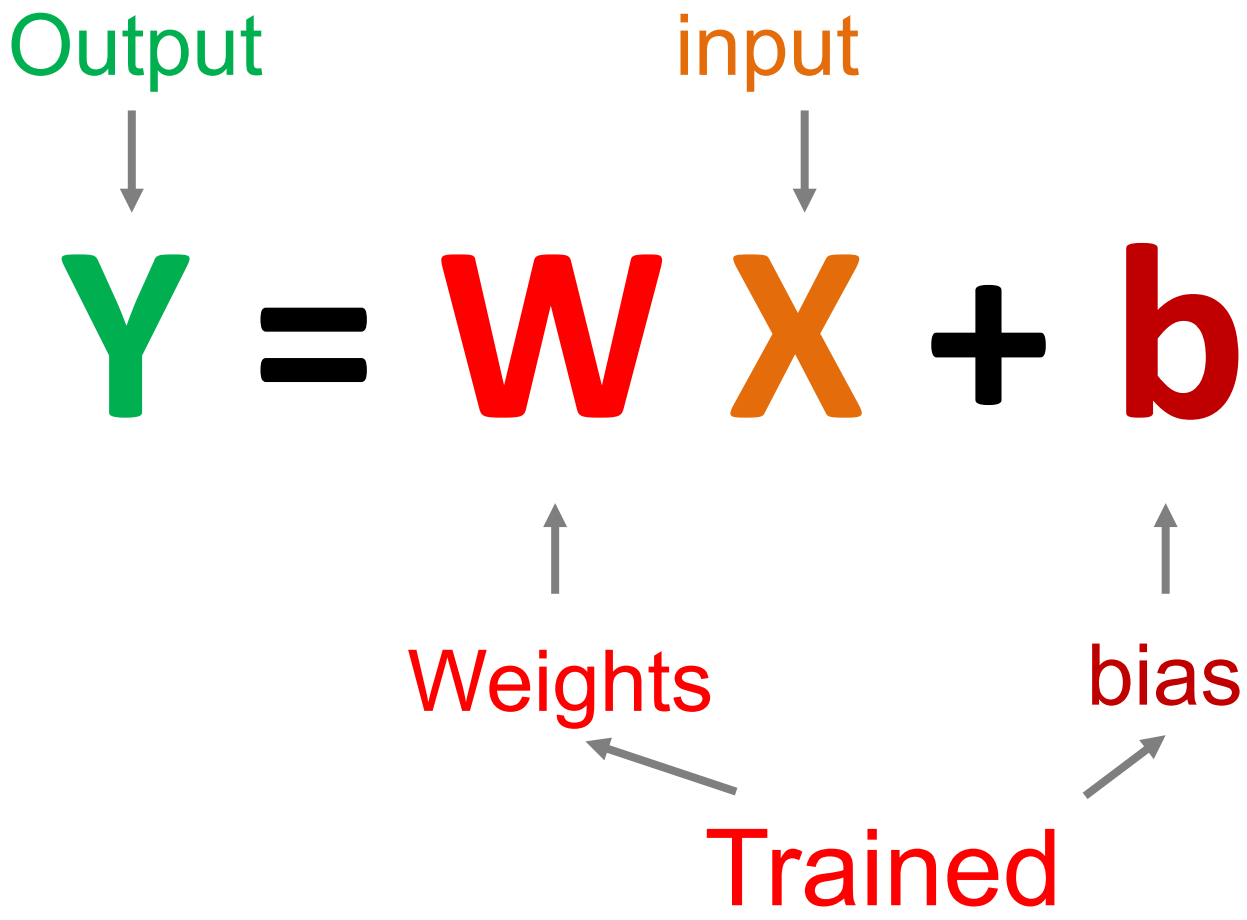
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

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

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

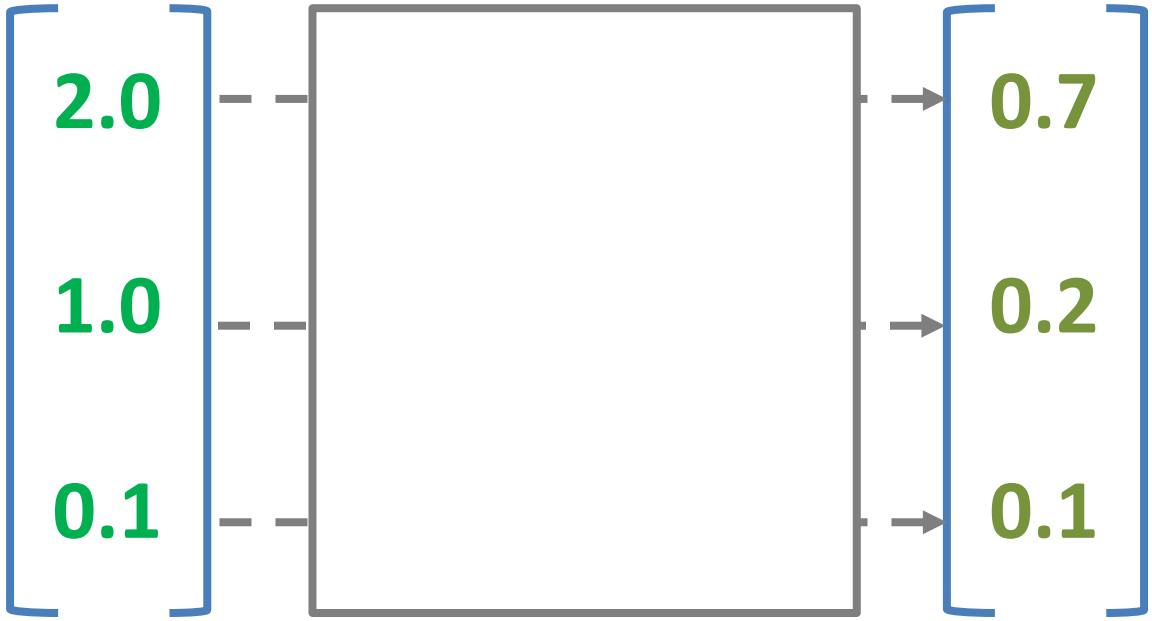
$$Y = W X + b$$

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE



Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

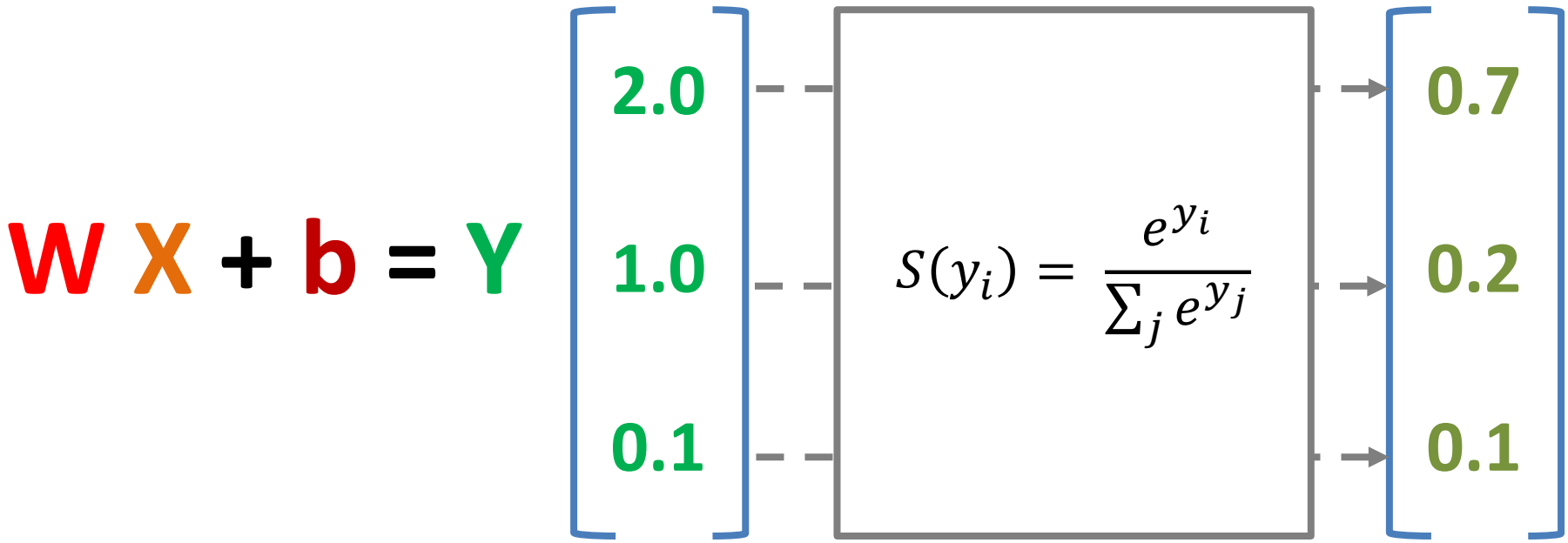
$$W\,X + b = Y$$



Scores → Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

SoftMAX



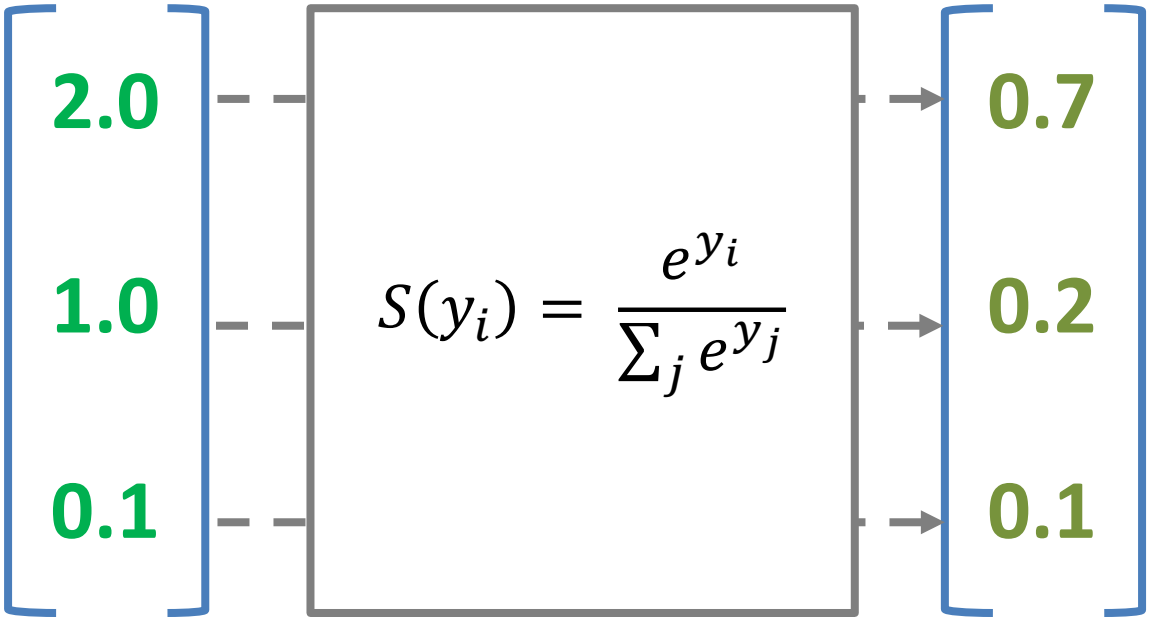
Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

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

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

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

W X + b = Y



Logits Scores → Probabilities

Training a Network
=
Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Training a Network

=

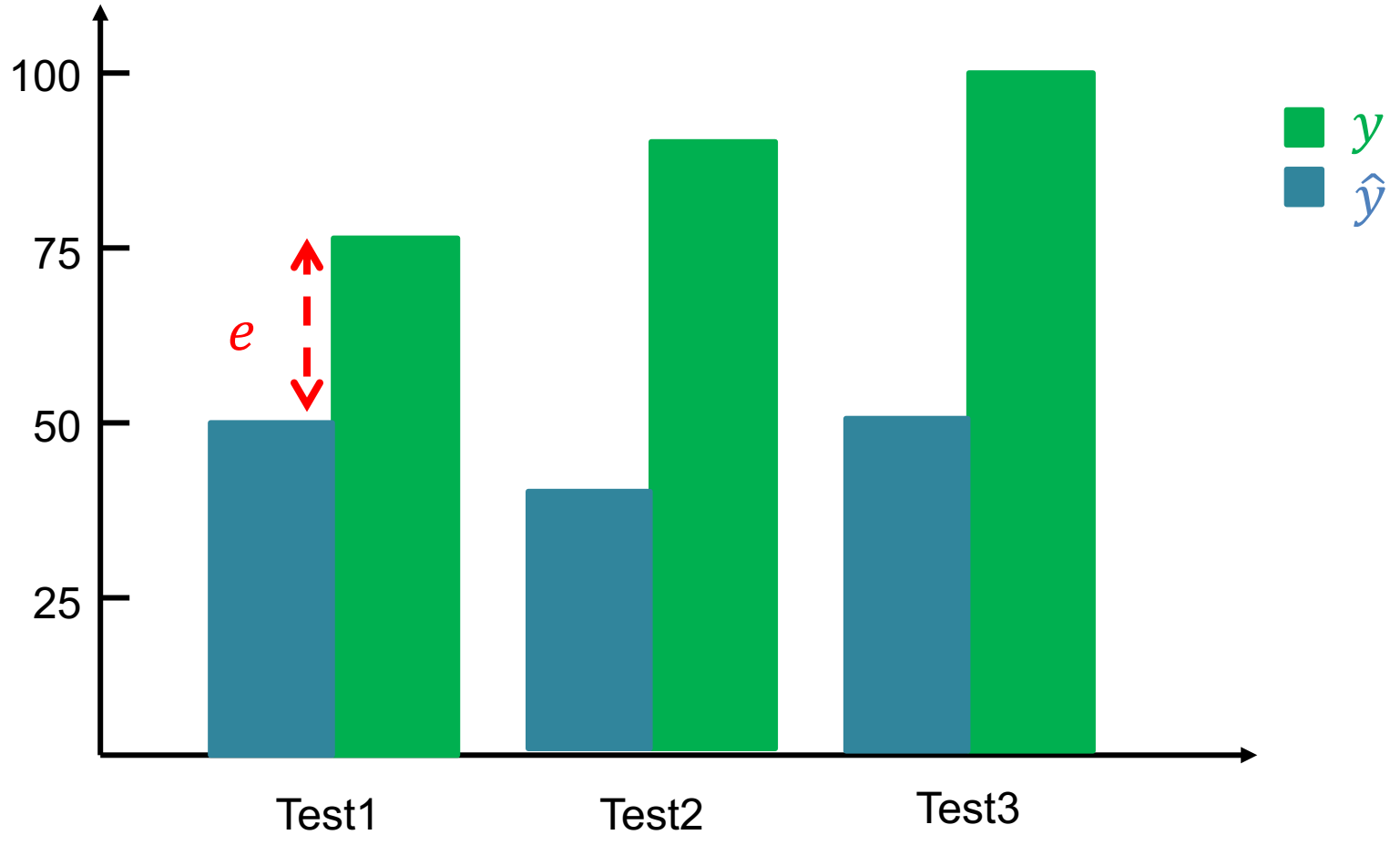
Minimize the Cost Function

Minimize the Loss Function

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Error = Predict Y - Actual Y

Error : Cost : Loss



Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

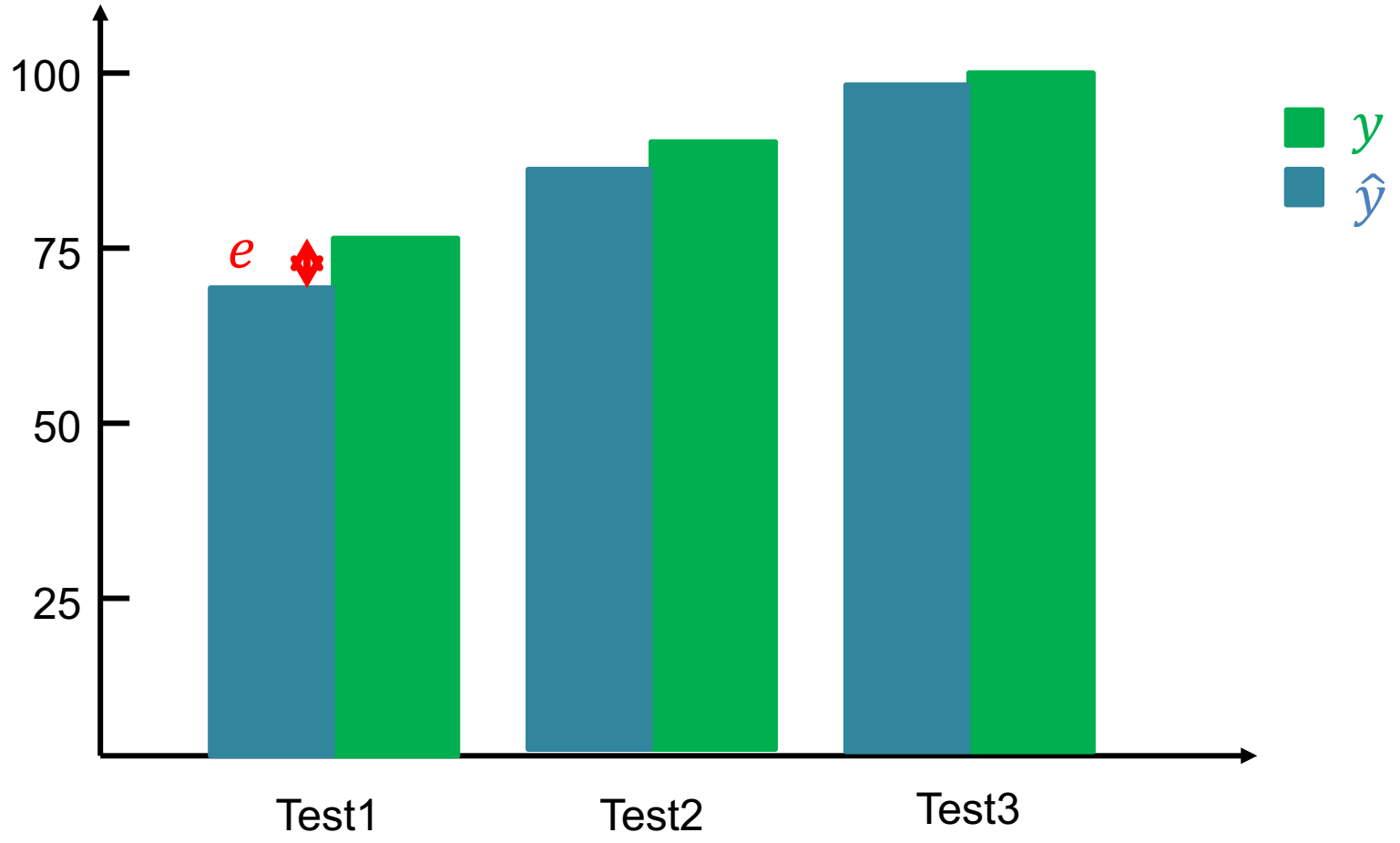
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Error = Predict Y - Actual Y

Error : Cost : Loss

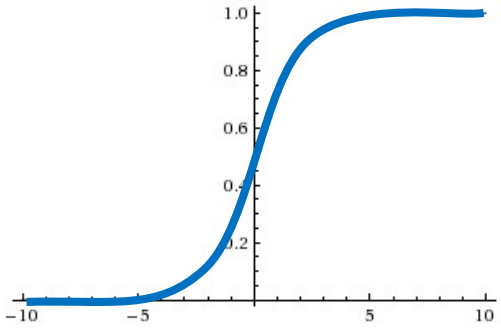


Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Activation Functions

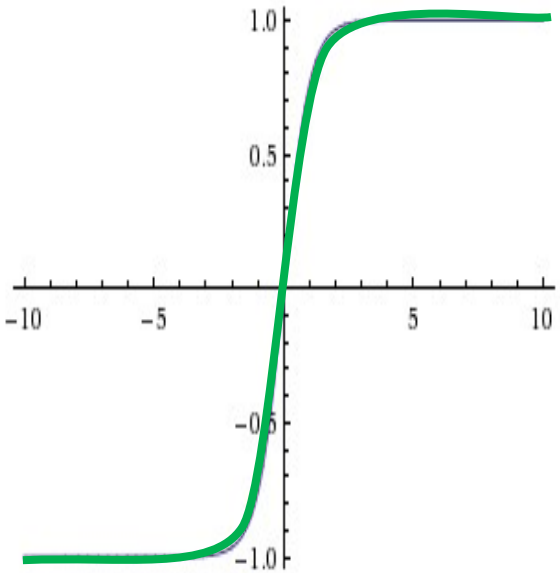
Activation Functions

Sigmoid



[0, 1]

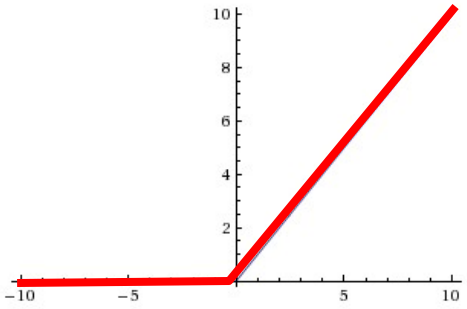
TanH



[-1, 1]

ReLU

(Rectified Linear Unit)

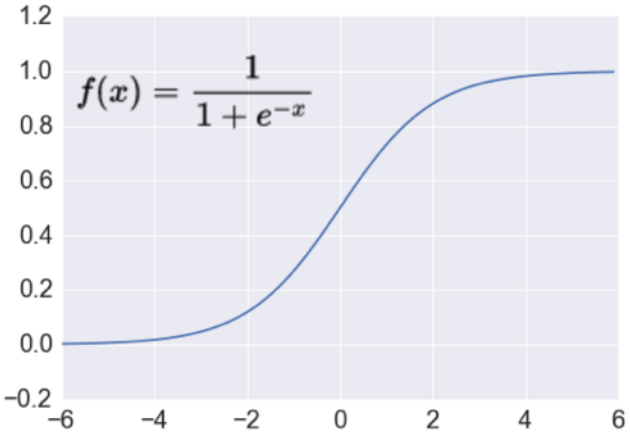


$f(x) = max(0, x)$

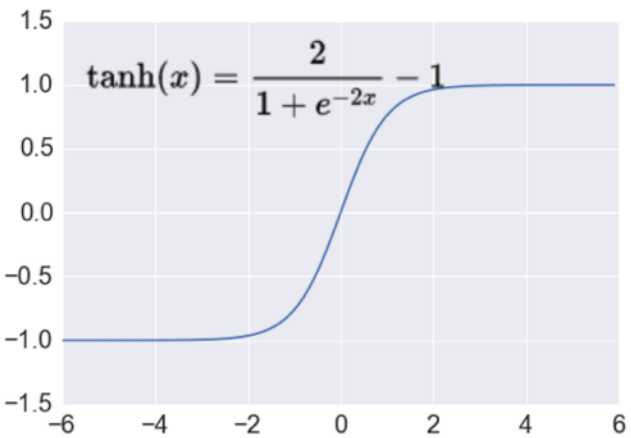
Source: http://cs231n.github.io/neural-networks-1/

Activation Functions

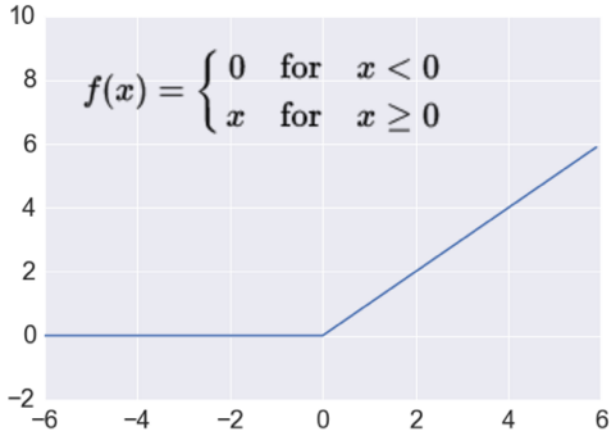
Sigmoid



TanH



ReLU



Source: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Loss Function

Binary Classification: 2 Class

Activation Function: Sigmoid

Loss Function: Binary Cross-Entropy

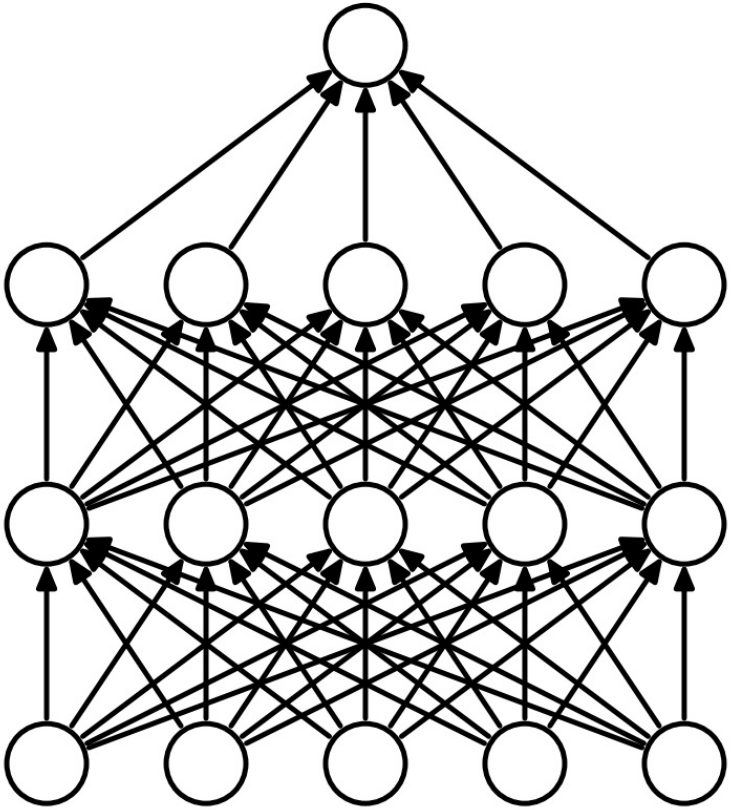
Multiple Classification: 10 Class

Activation Function:
SoftMAX

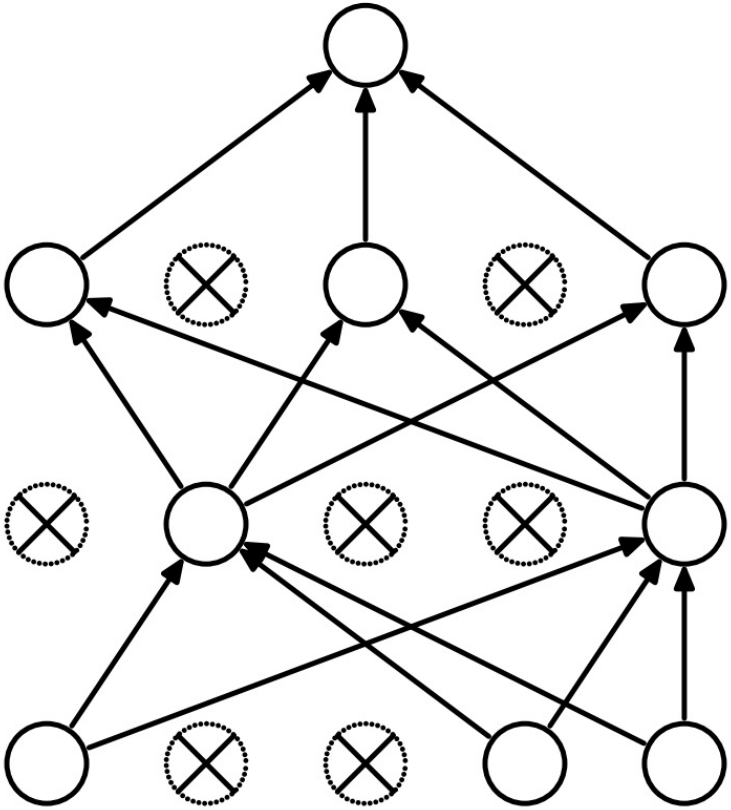
Loss Function:
Categorical Cross-Entropy

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.

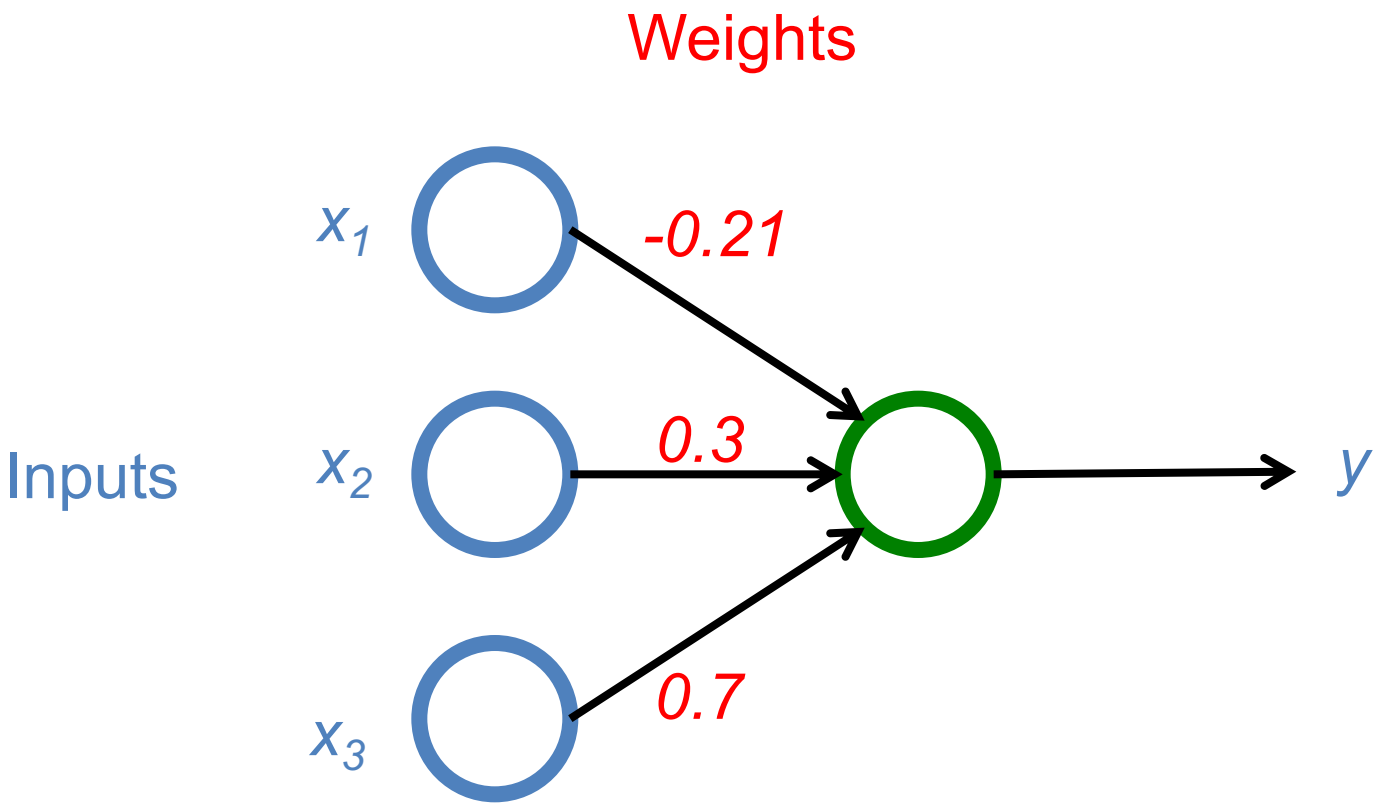
Learning Algorithm

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”

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

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

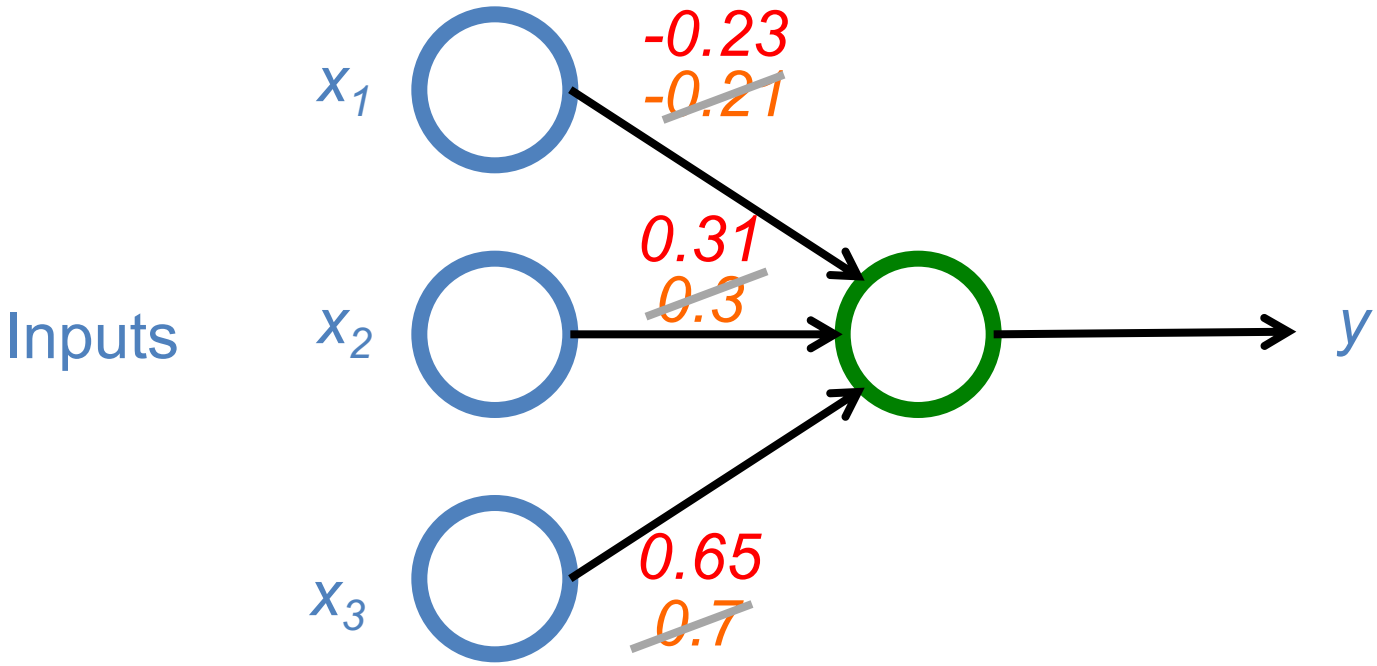


Next time:

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

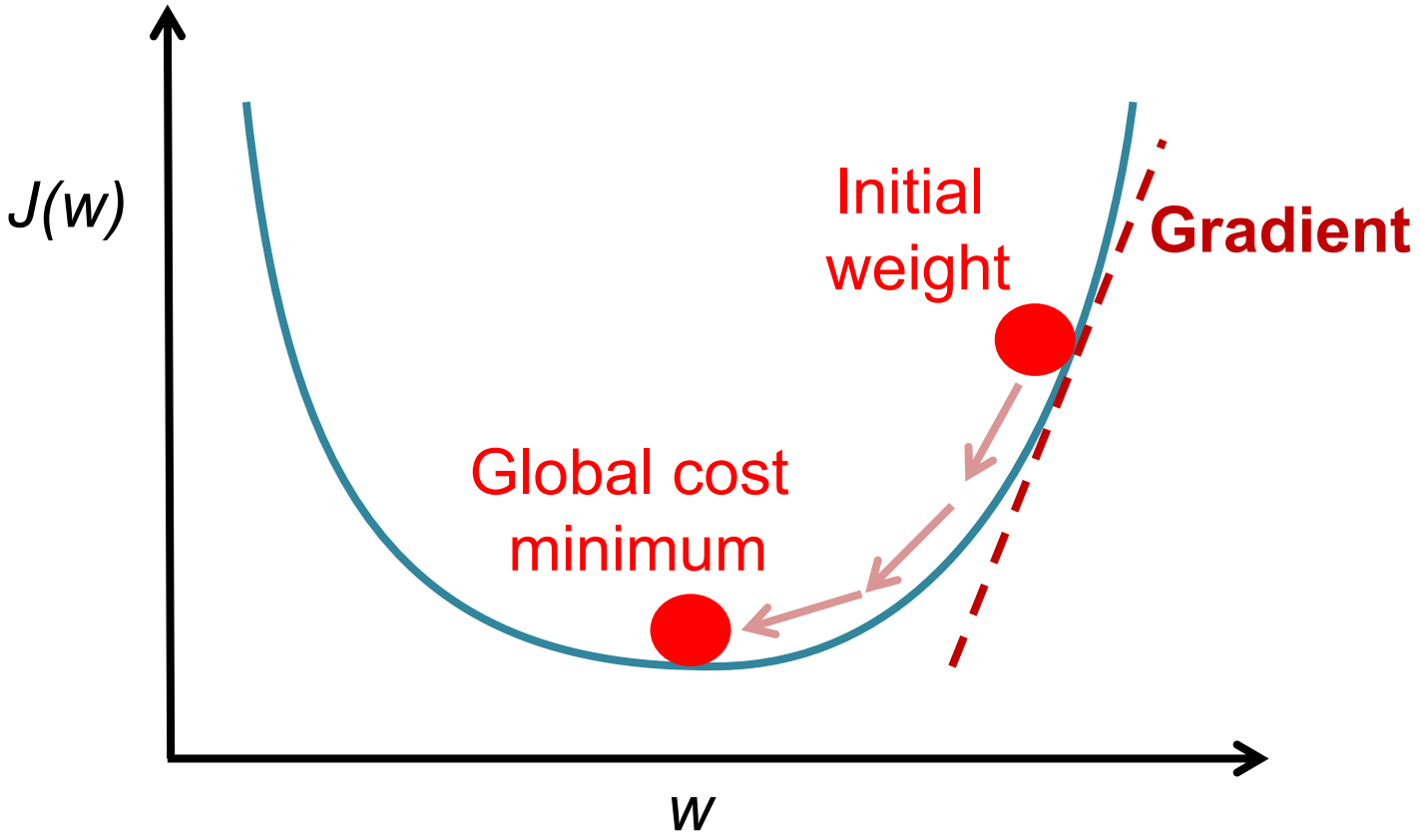
~~$$y = max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$~~

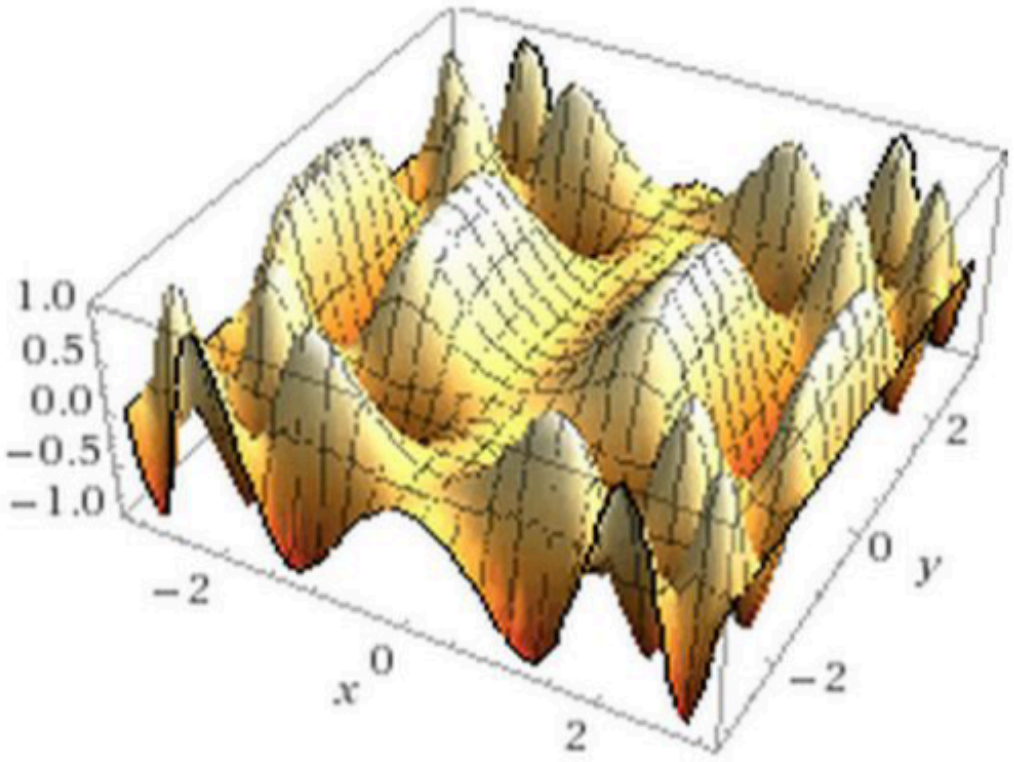
Weights



Optimizer:

Stochastic Gradient Descent (SGD)





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

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Visual Recognition

Image Classification

IS THIS A
CAT or DOG?

CAT DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

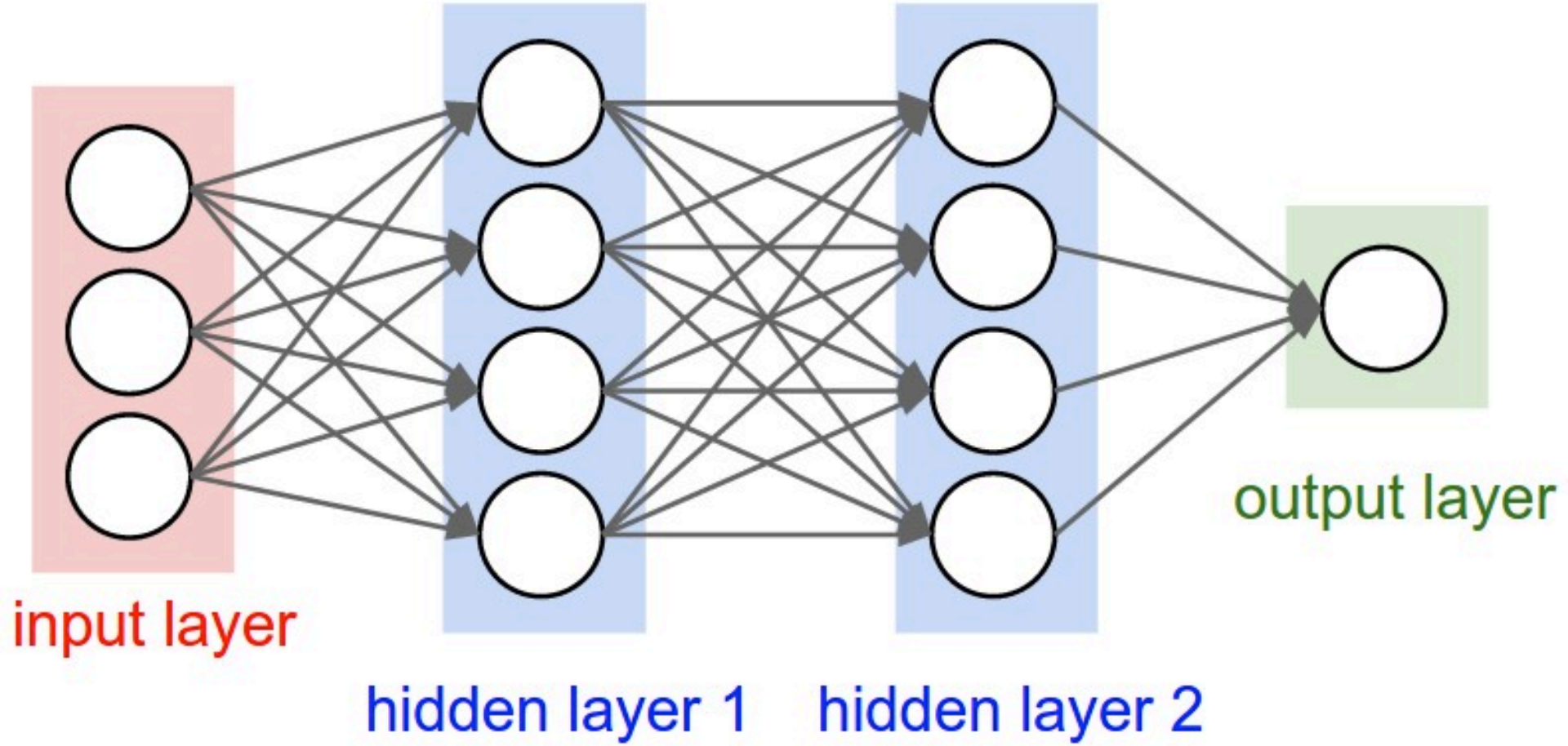
DEEP *neural* NETWORK

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Convolutional Neural Networks (CNNs / ConvNets)

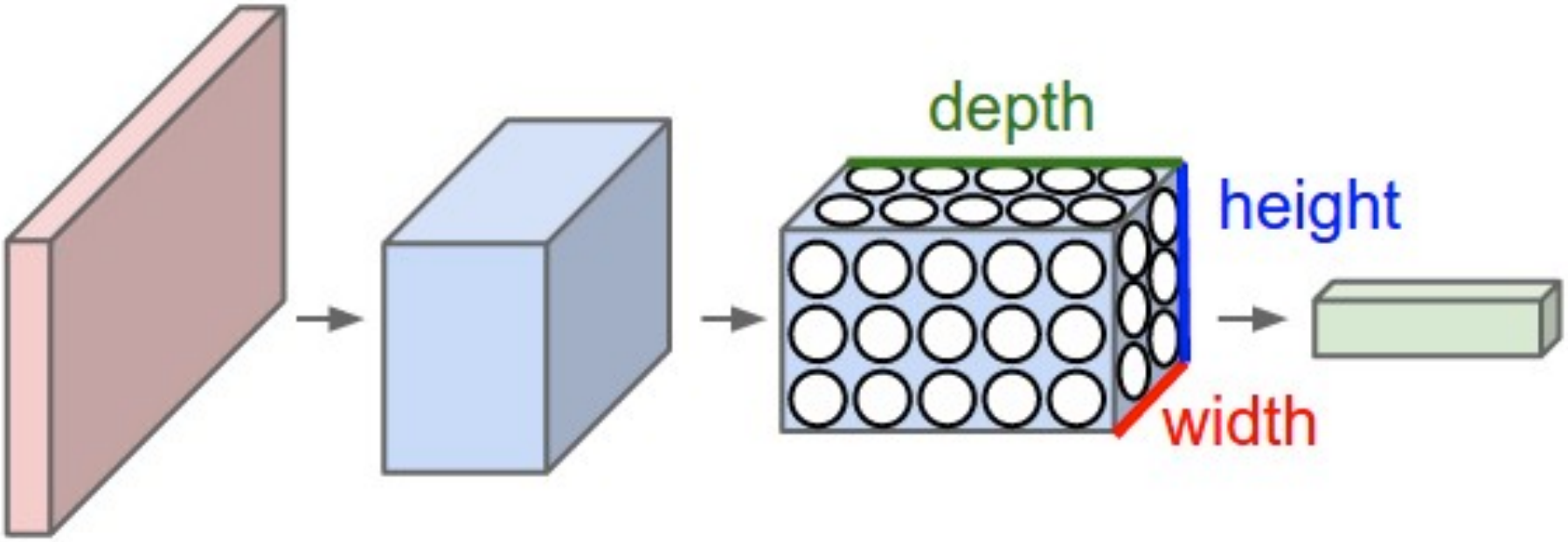
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



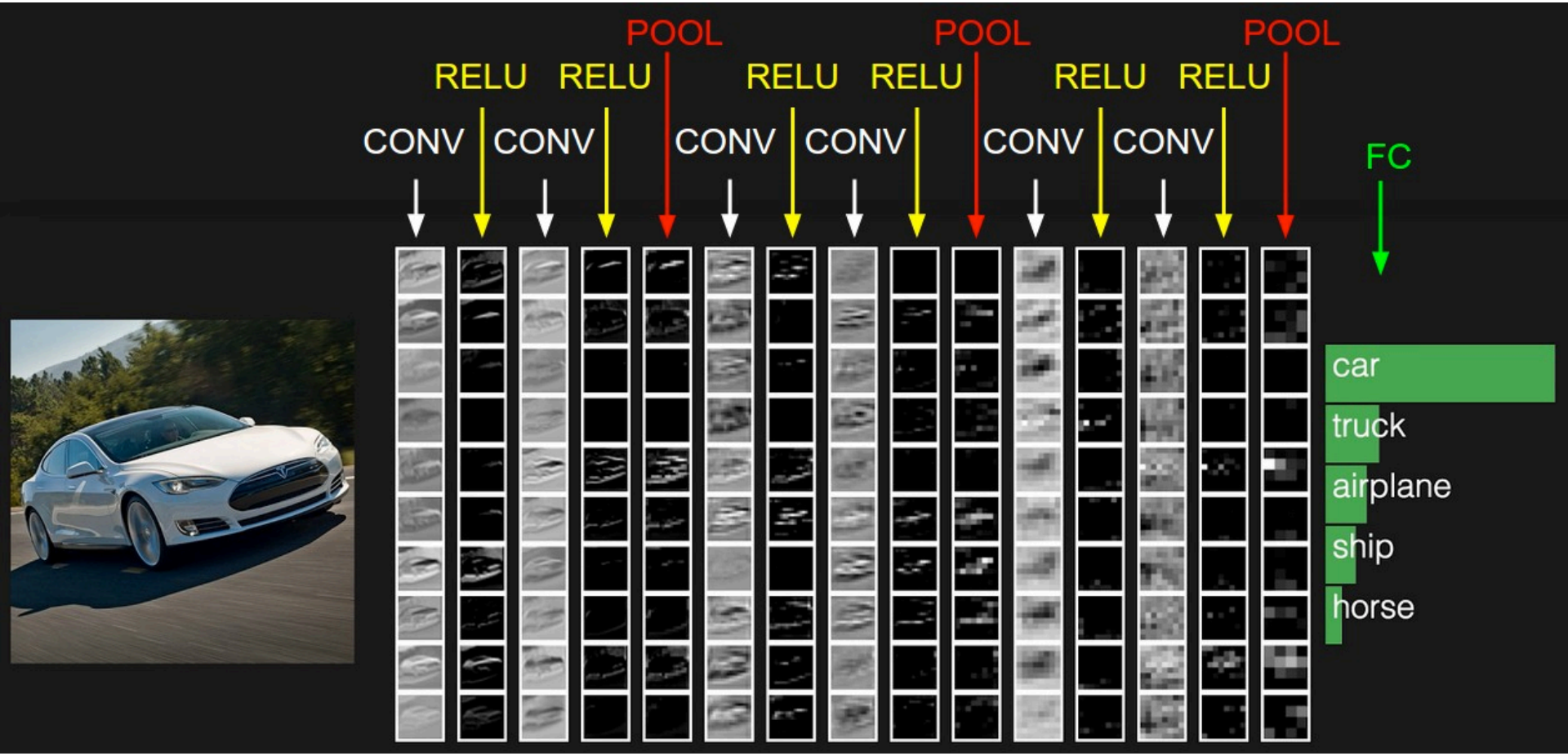
http://cs231n.github.io/convolutional-networks/

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



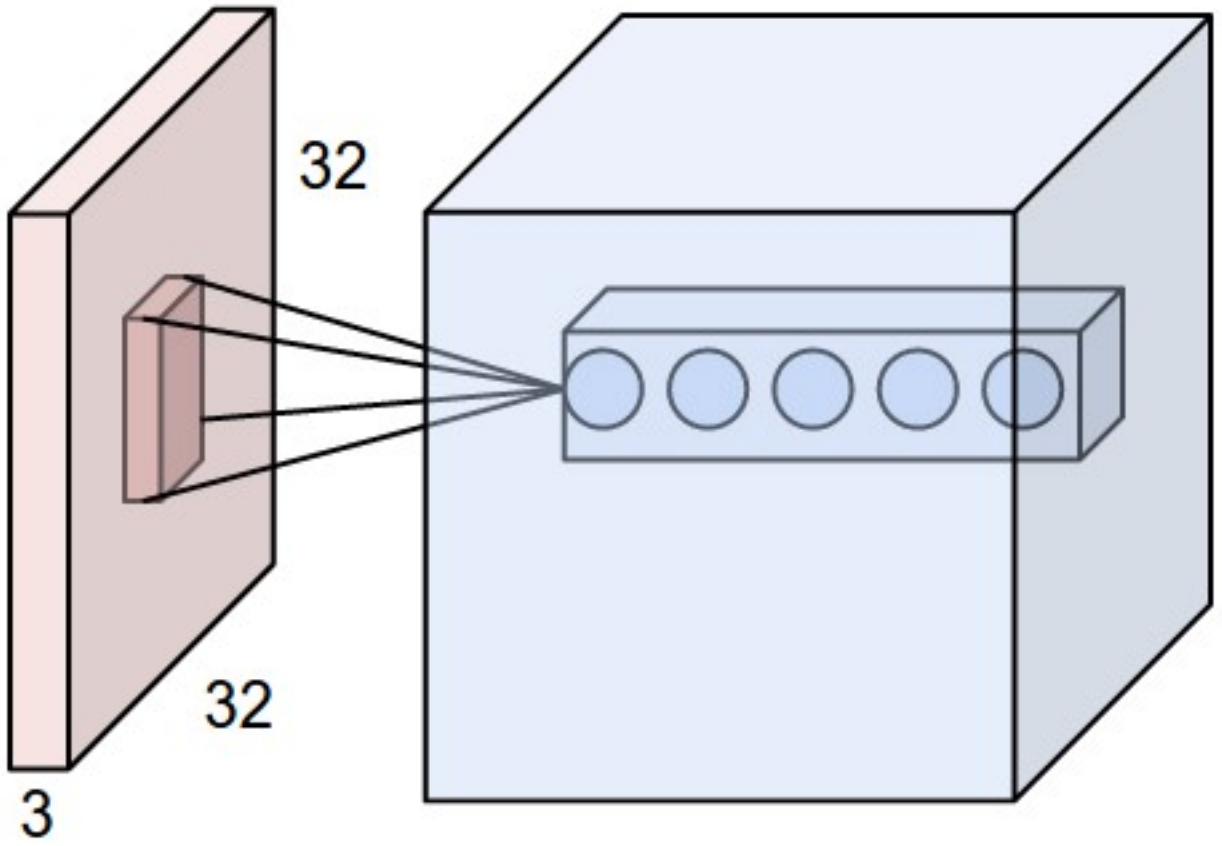
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



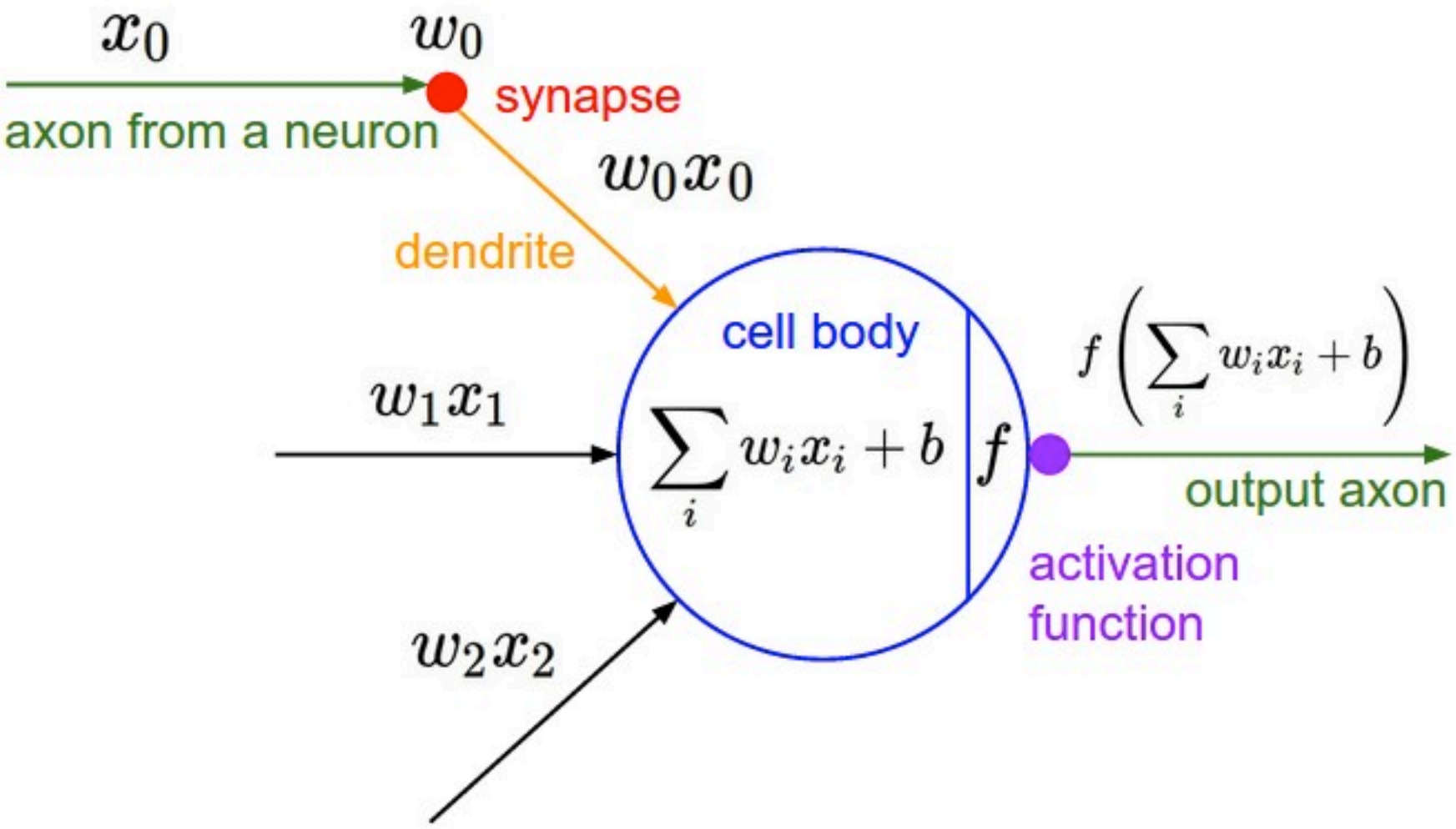
http://cs231n.github.io/convolutional-networks/

ConvNets



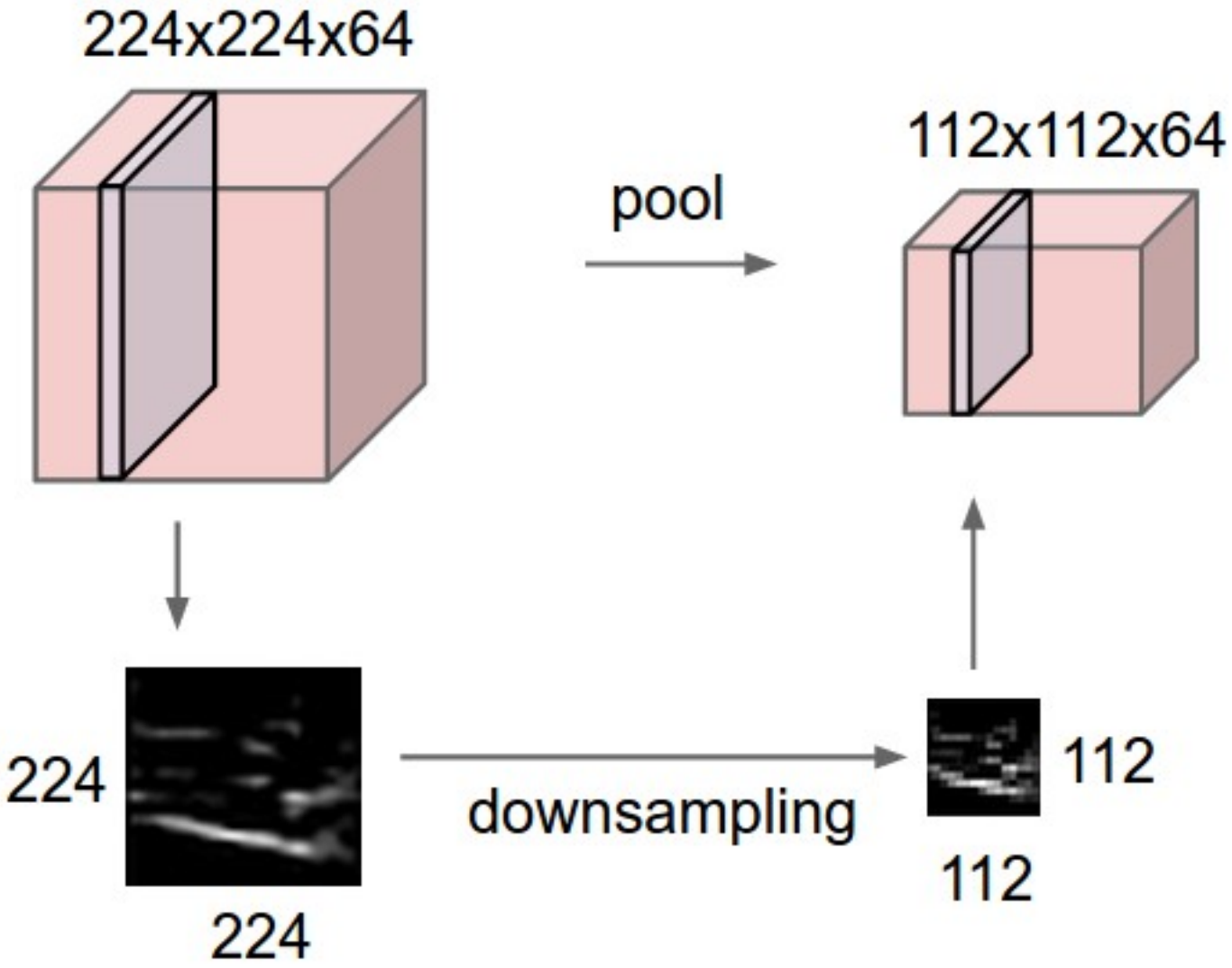
http://cs231n.github.io/convolutional-networks/

ConvNets



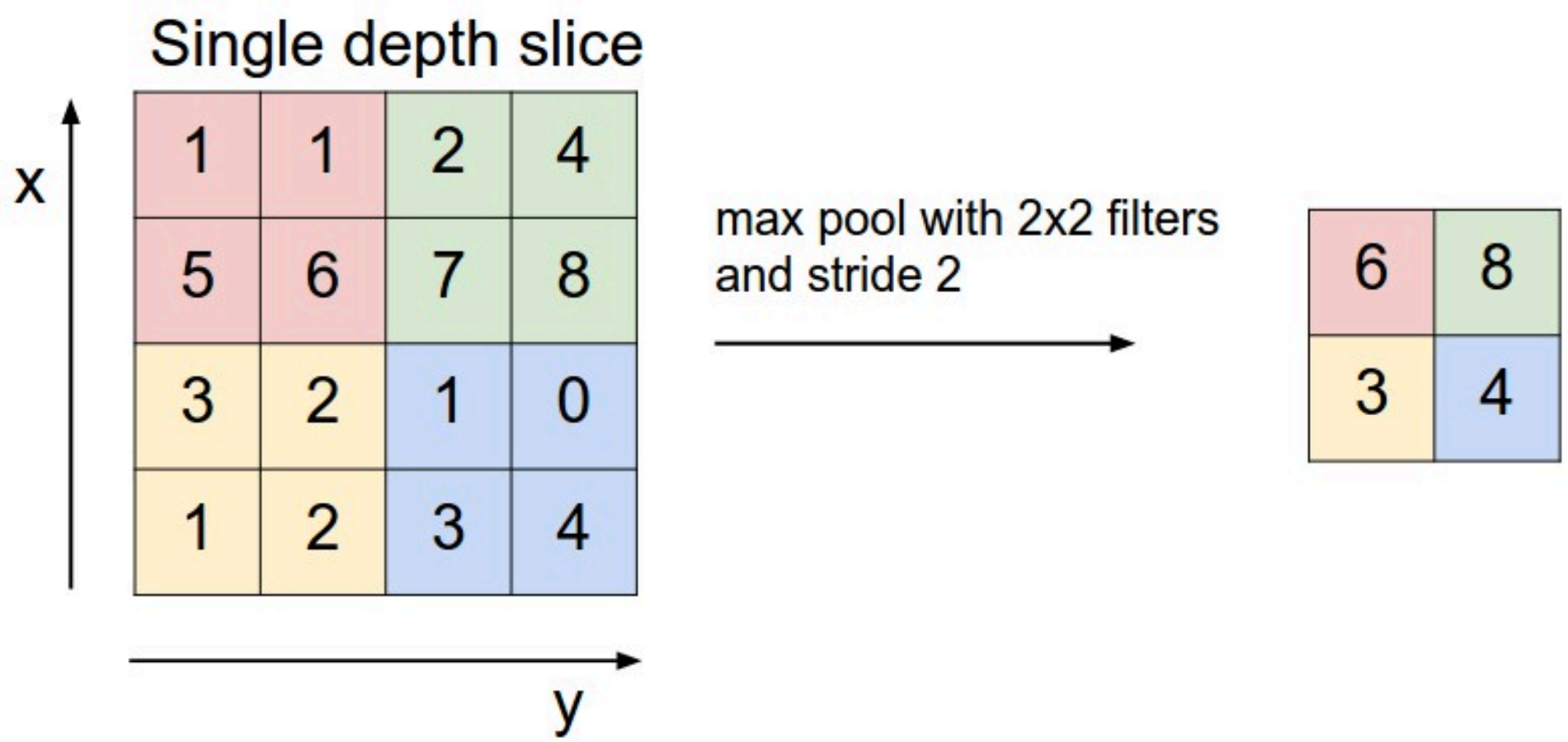
http://cs231n.github.io/convolutional-networks/

ConvNets



http://cs231n.github.io/convolutional-networks/

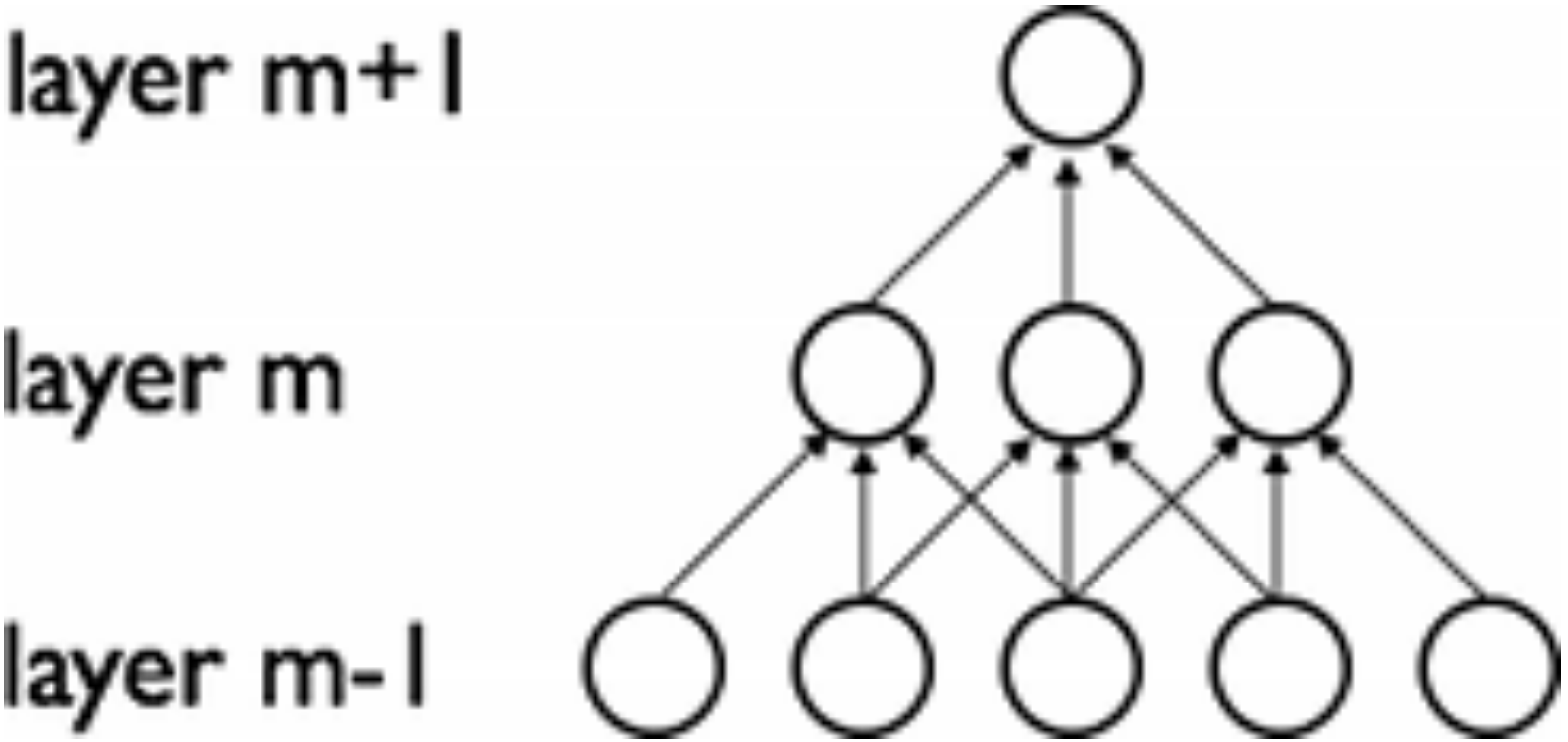
ConvNets
max pooling



http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks (CNN) (LeNet)

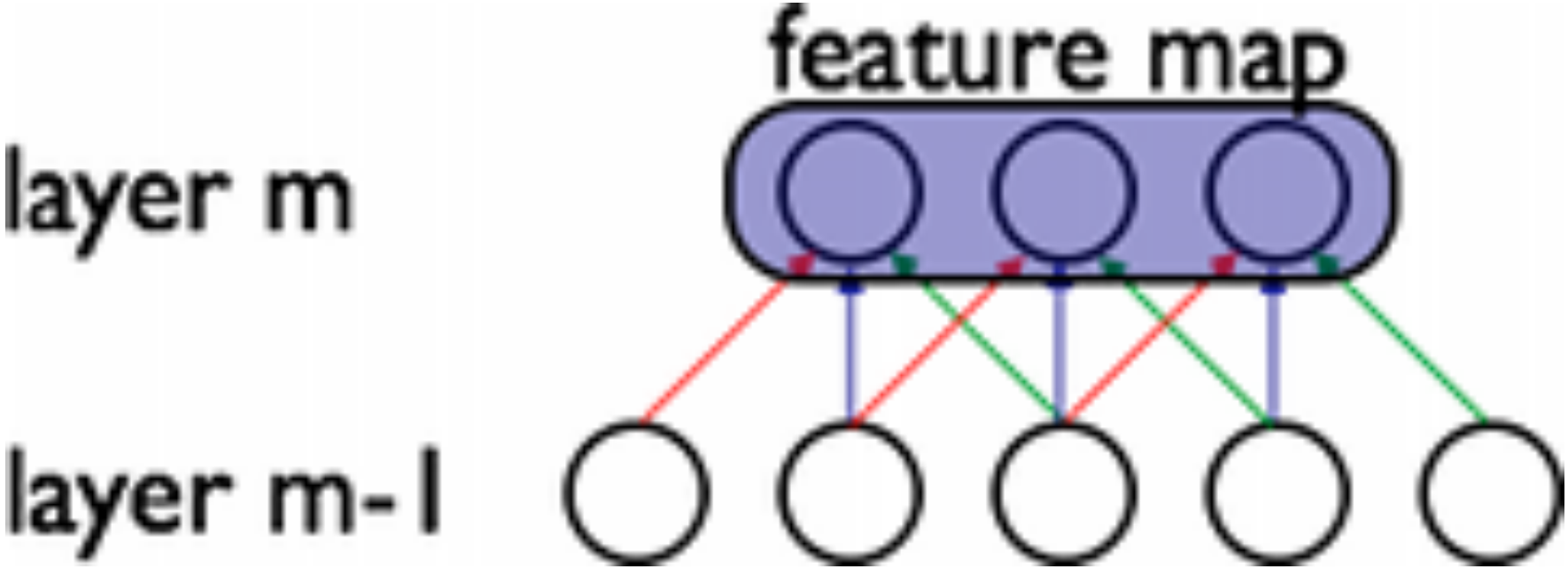
Sparse Connectivity



Source: http://deeplearning.net/tutorial/lenet.html

Convolutional Neural Networks (CNN) (LeNet)

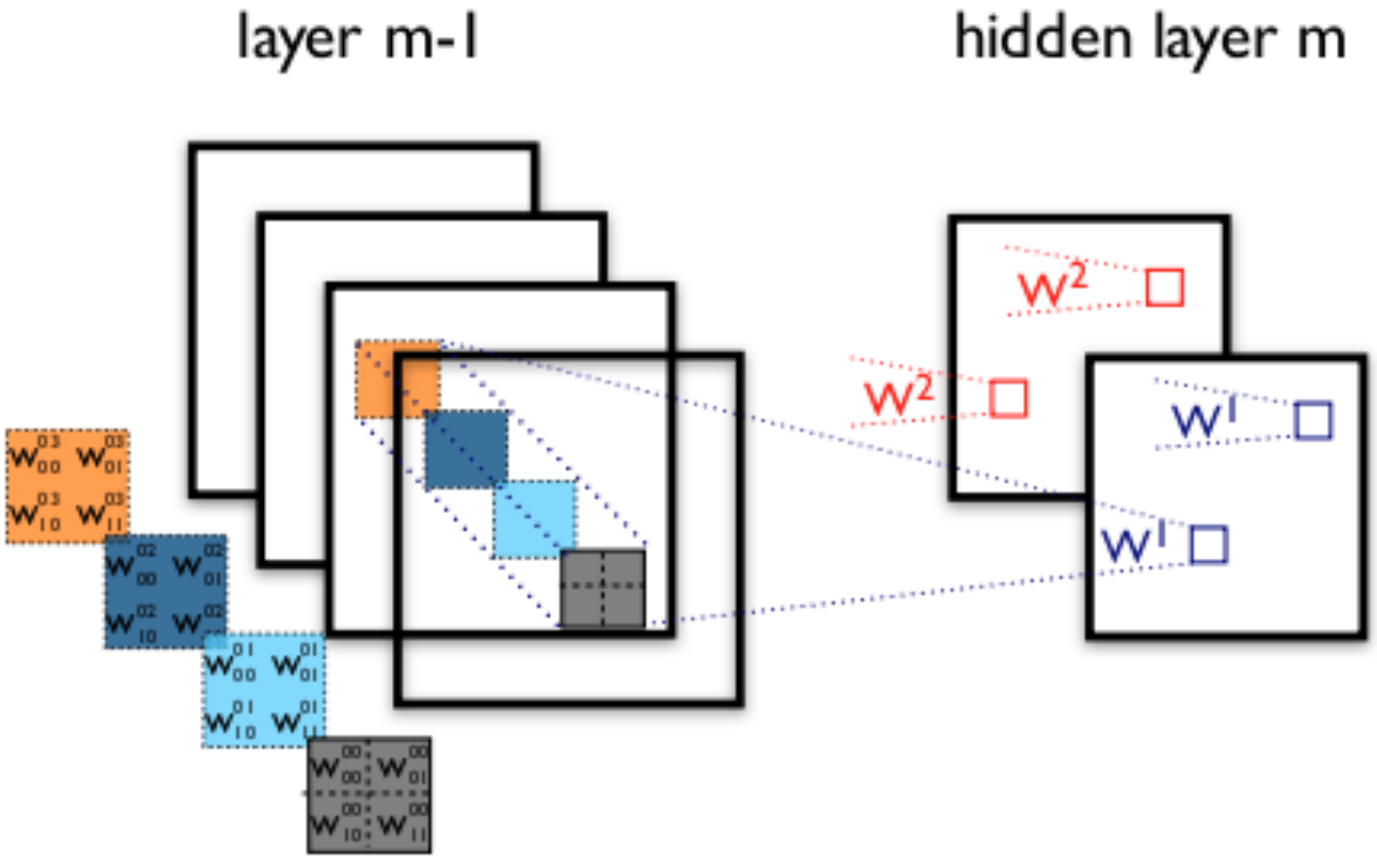
Shared Weights



Source: http://deeplearning.net/tutorial/lenet.html

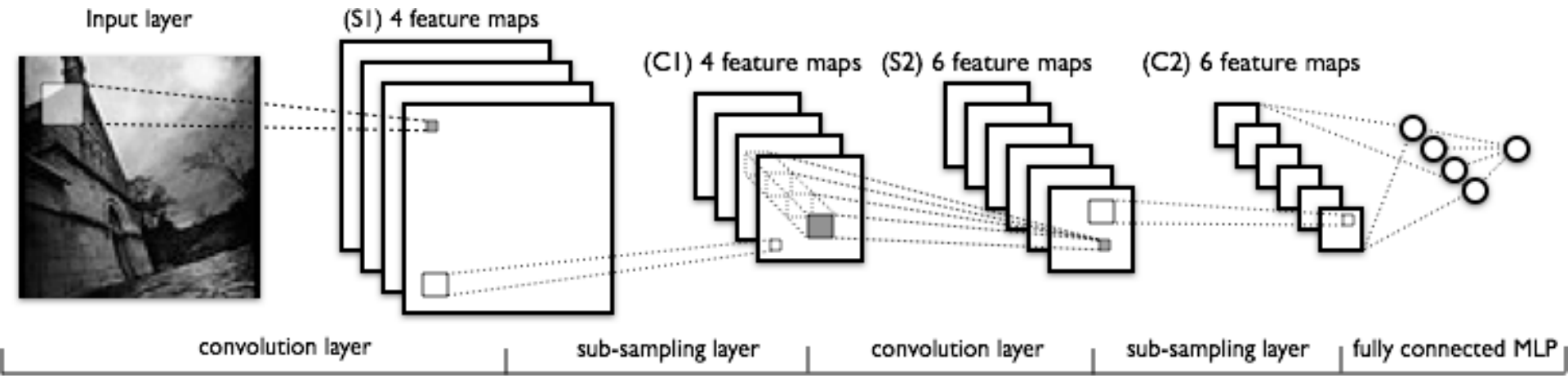
Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer



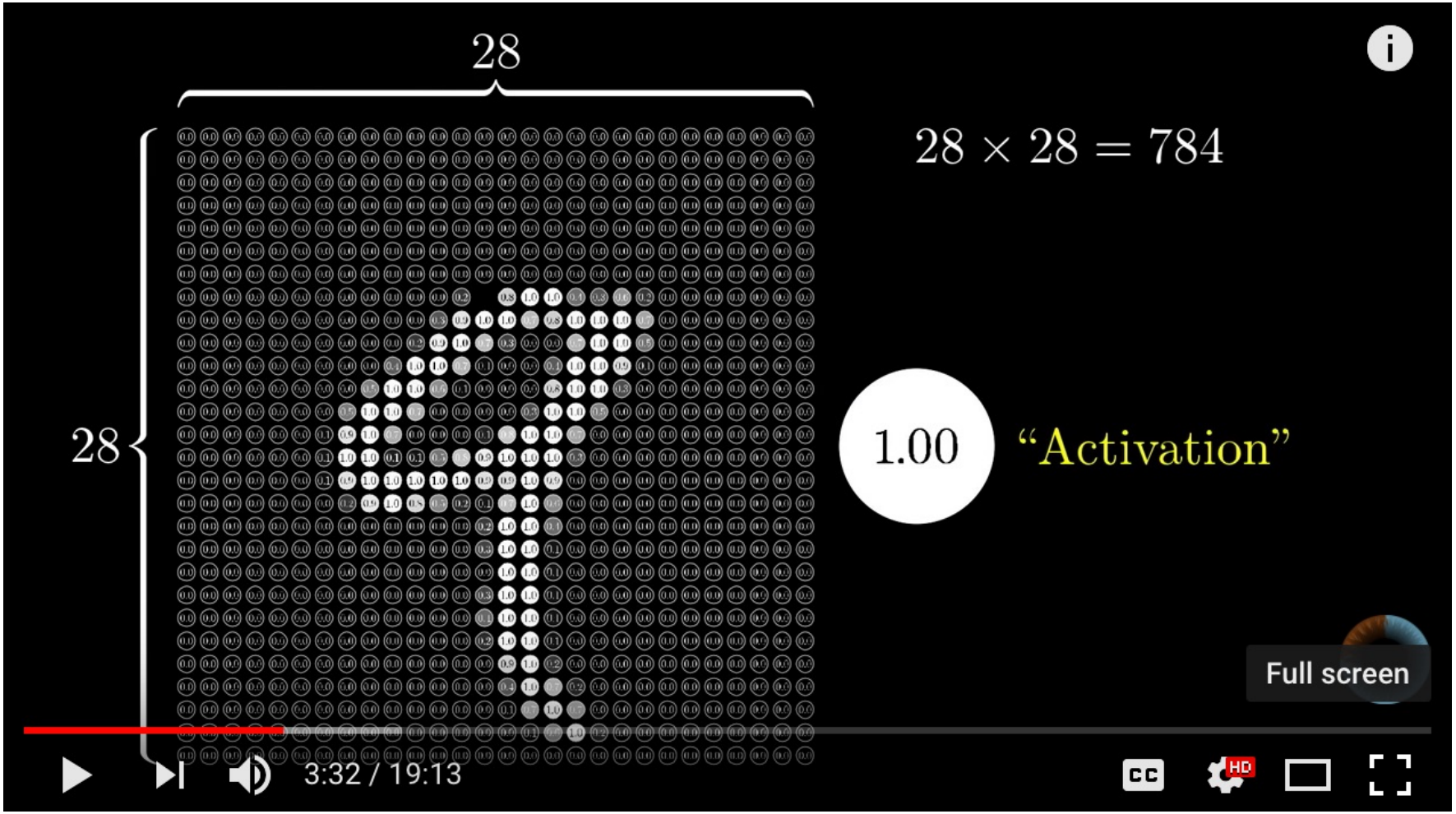
Source: http://deeplearning.net/tutorial/lenet.html

Convolutional Neural Networks (CNN) (LeNet)



Source: http://deeplearning.net/tutorial/lenet.html

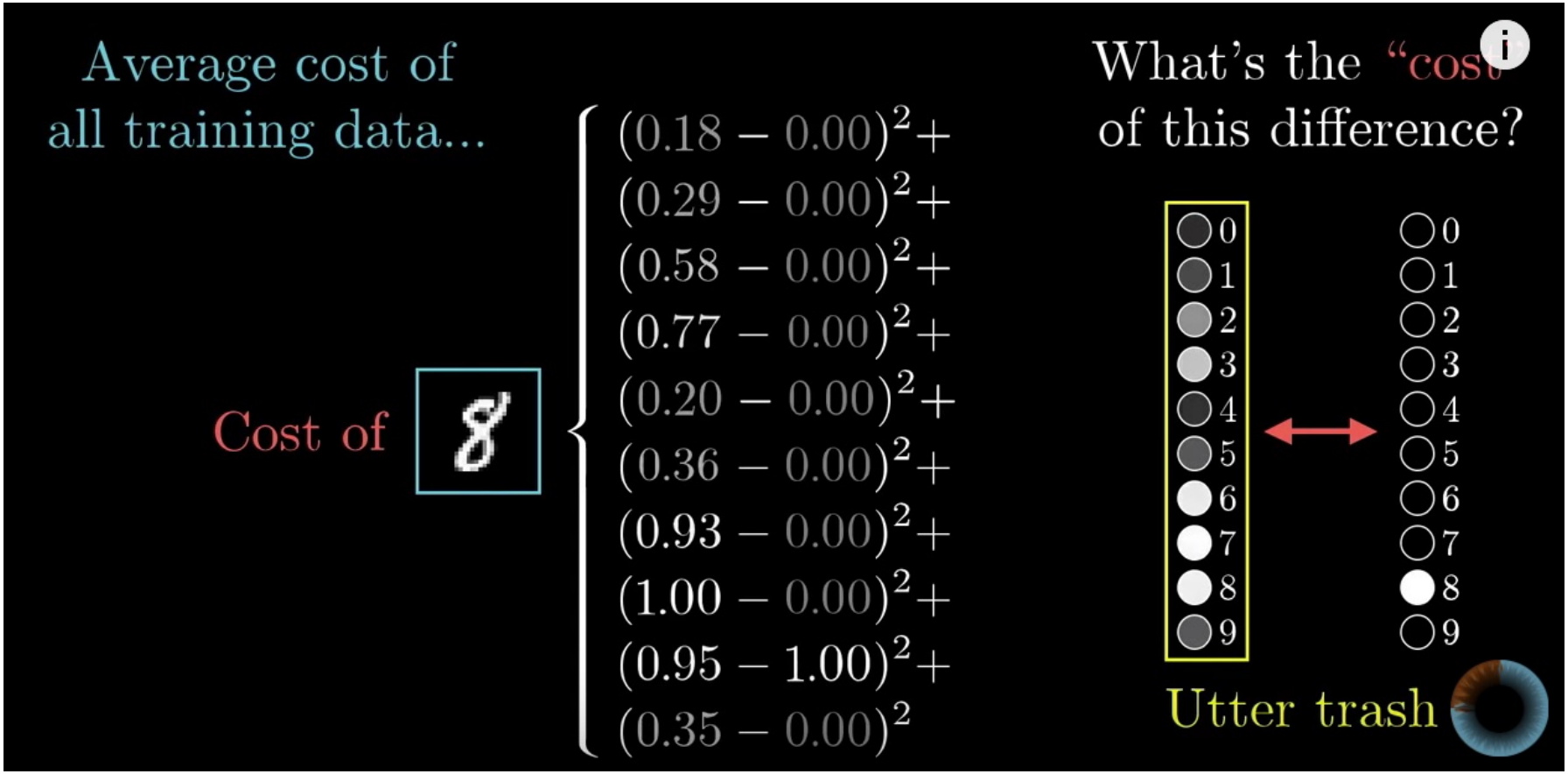
Neural Network and Deep Learning



Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

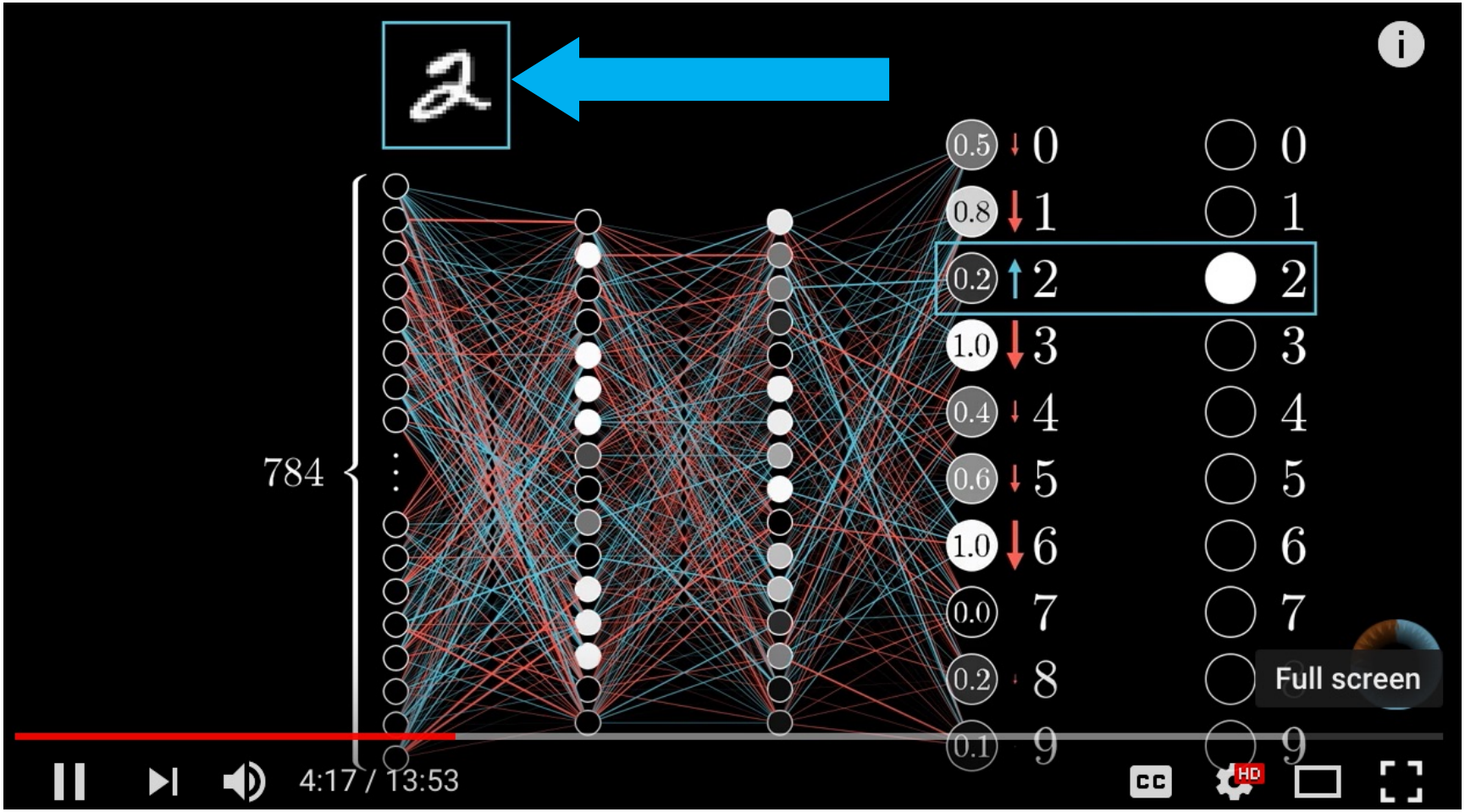
Gradient Descent

how neural networks learn



Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w

Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U

Learning Algorithm

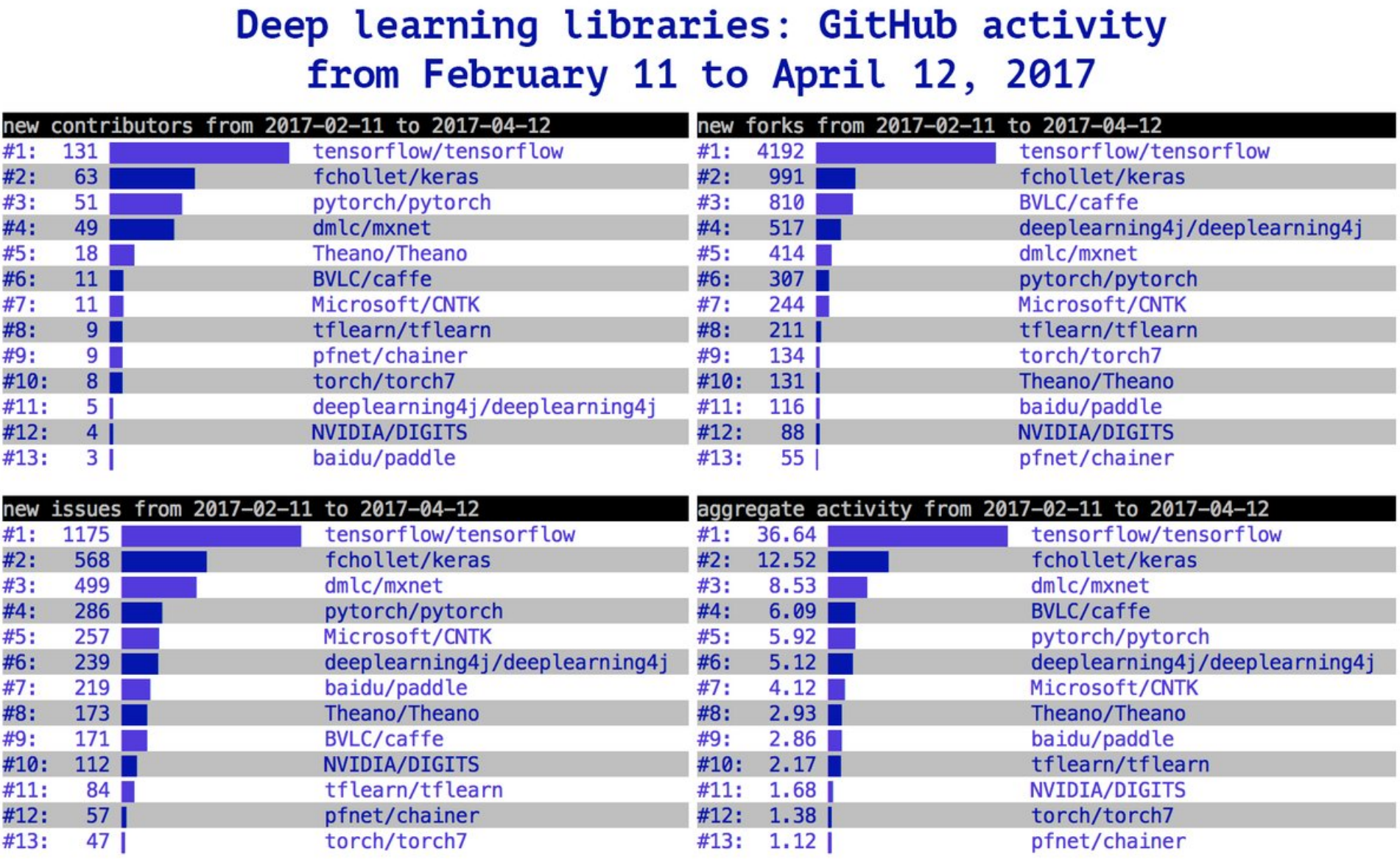
While not done:

- Pick a random training example “(input, label)”
- Run neural network on “input”
- Adjust weights on edges to make output closer to “label”

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Deep Learning with TensorFlow

Deep Learning Libraries: Tensorflow and Keras



Source: https://twitter.com/fchollet/status/852194634470223873

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

Source: http://deeplearning.net/software_links/

Keras

Secure | https://keras.io

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:

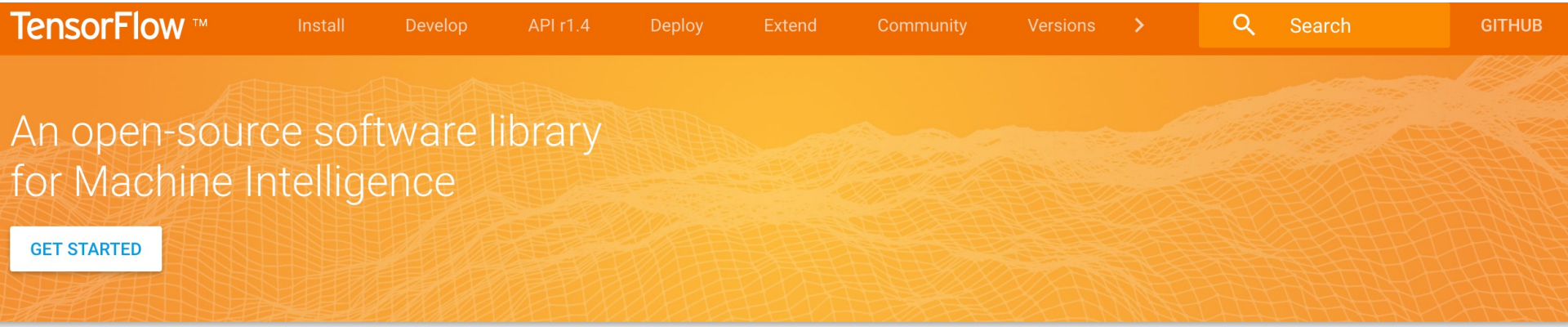
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

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

http://keras.io/

Tensorflow



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!

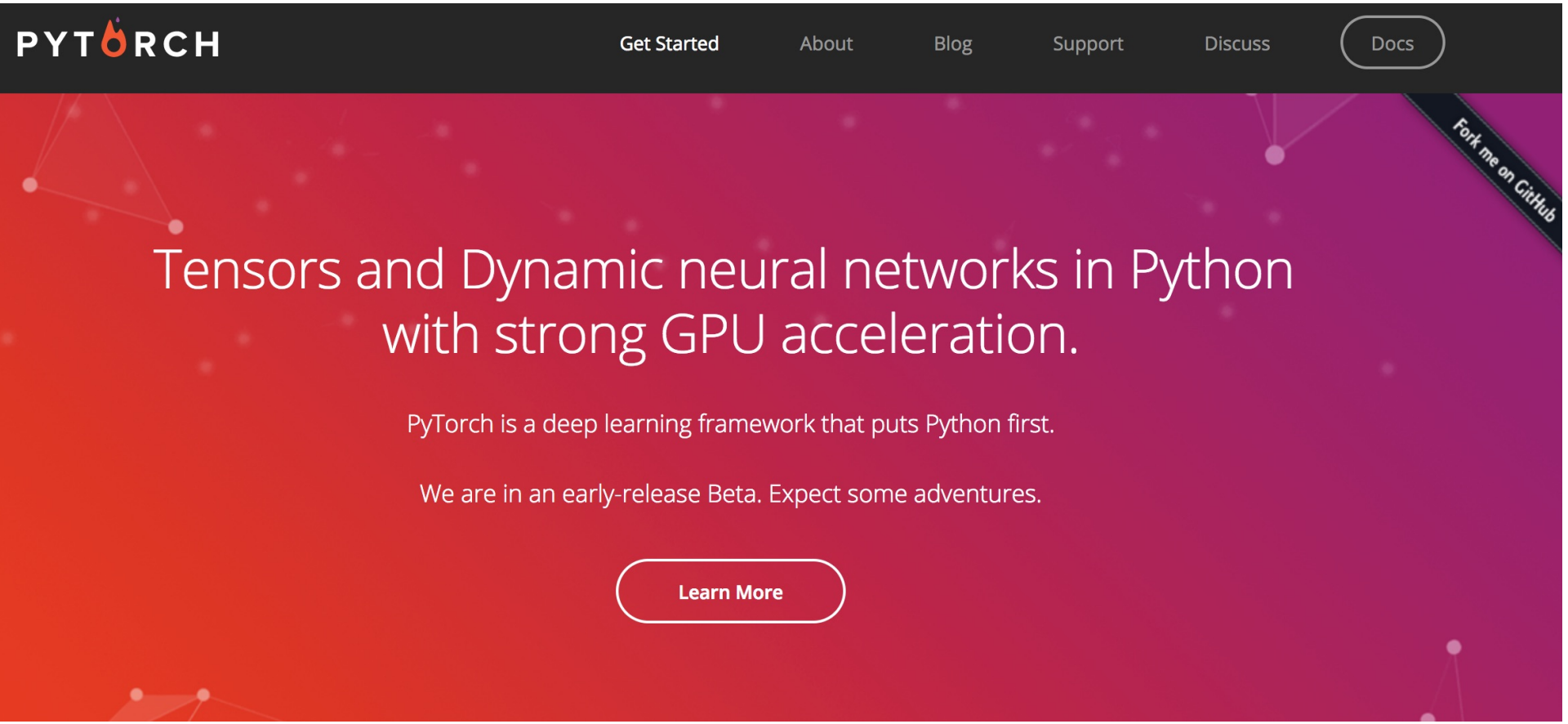
We're excited to announce the release of TensorFlow 1.4! Check out the release notes for all the latest.

Announcing TensorFlow Lite

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

https://www.tensorflow.org/

PyTorch



http://pytorch.org/

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.

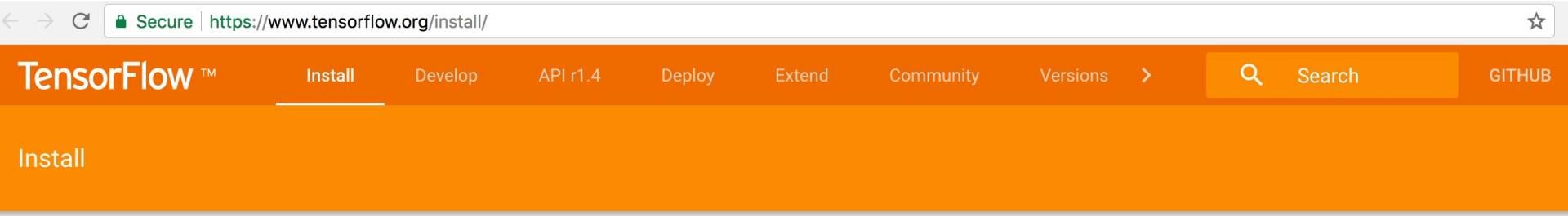
Source: https://keras.io/

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

https://keras.io/#installation

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.

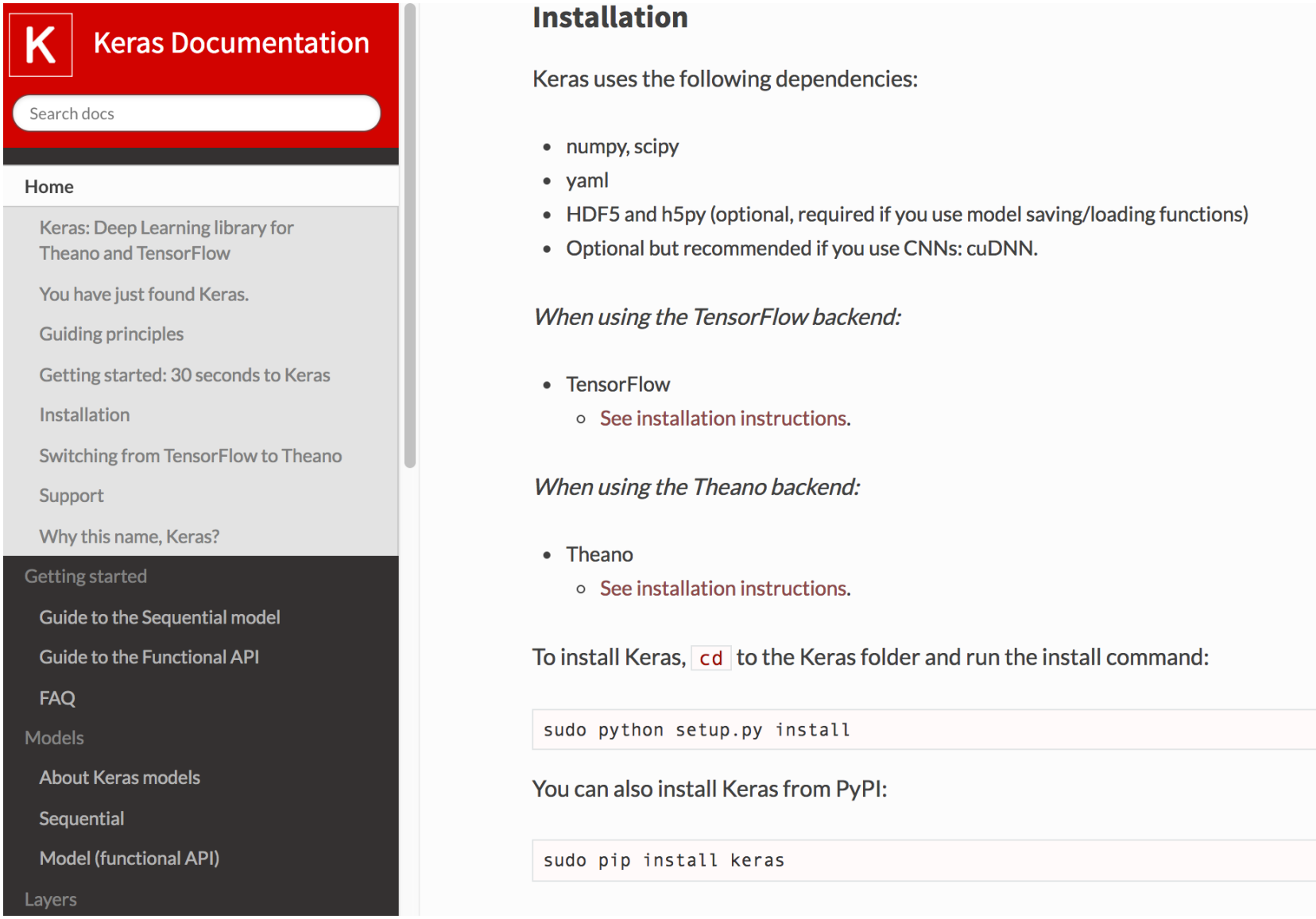
The following guides explain how to install a version of TensorFlow that enables you to write applications in Python:

- 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



Keras Documentation

Search docs

Home

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Switching from TensorFlow to Theano

Support

Why this name, Keras?

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

Layers

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:

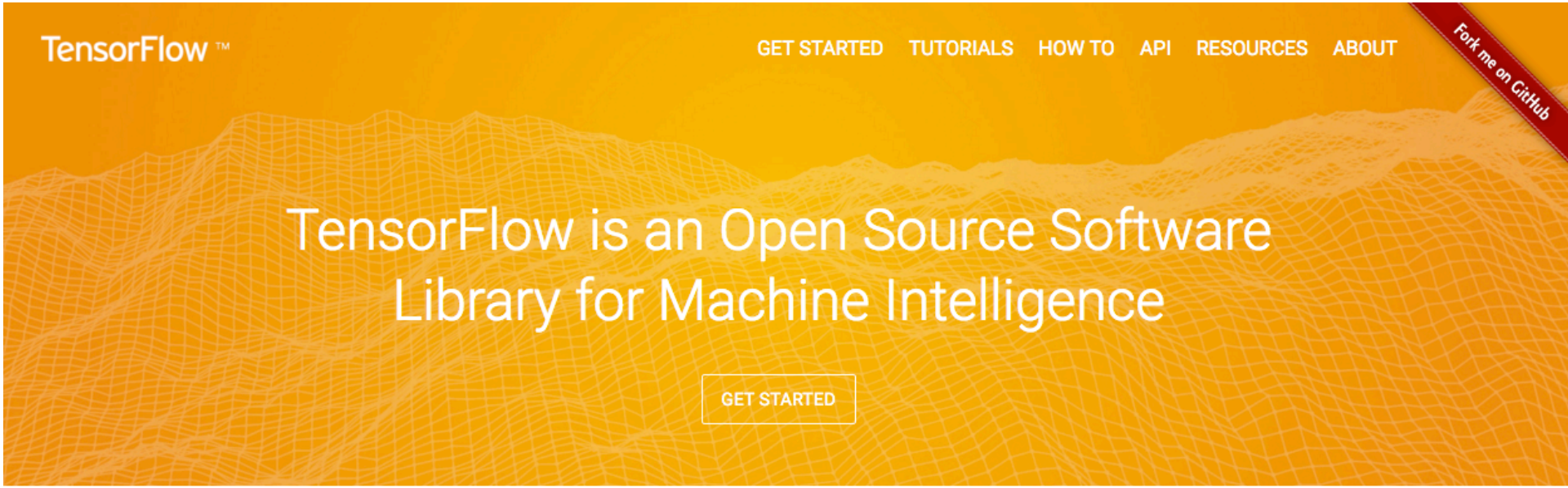
```
sudo pip install keras
```

https://keras.io/#installation

TensorFlow

Source: https://github.com/tensorflow/tensorflow

Google TensorFlow

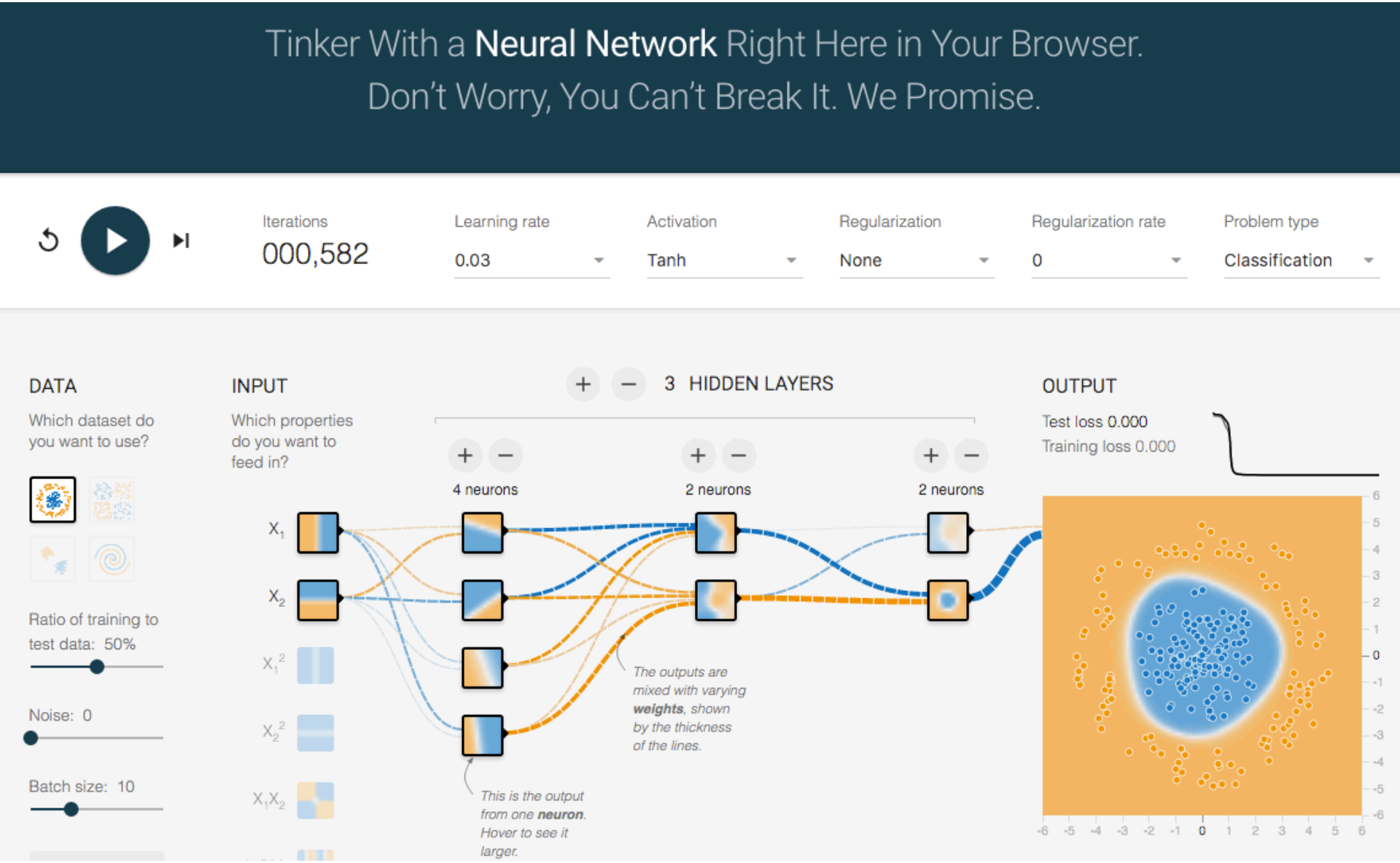


About TensorFlow

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



http://playground.tensorflow.org/

TensorFlow

TensorFlow

is an

Open Source Software Library

for

Machine Intelligence

https://www.tensorflow.org/

TensorFlow

numerical computation using data flow graphs

https://www.tensorflow.org/

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]

https://www.tensorflow.org/

TensorFlow

Nodes:

mathematical operations

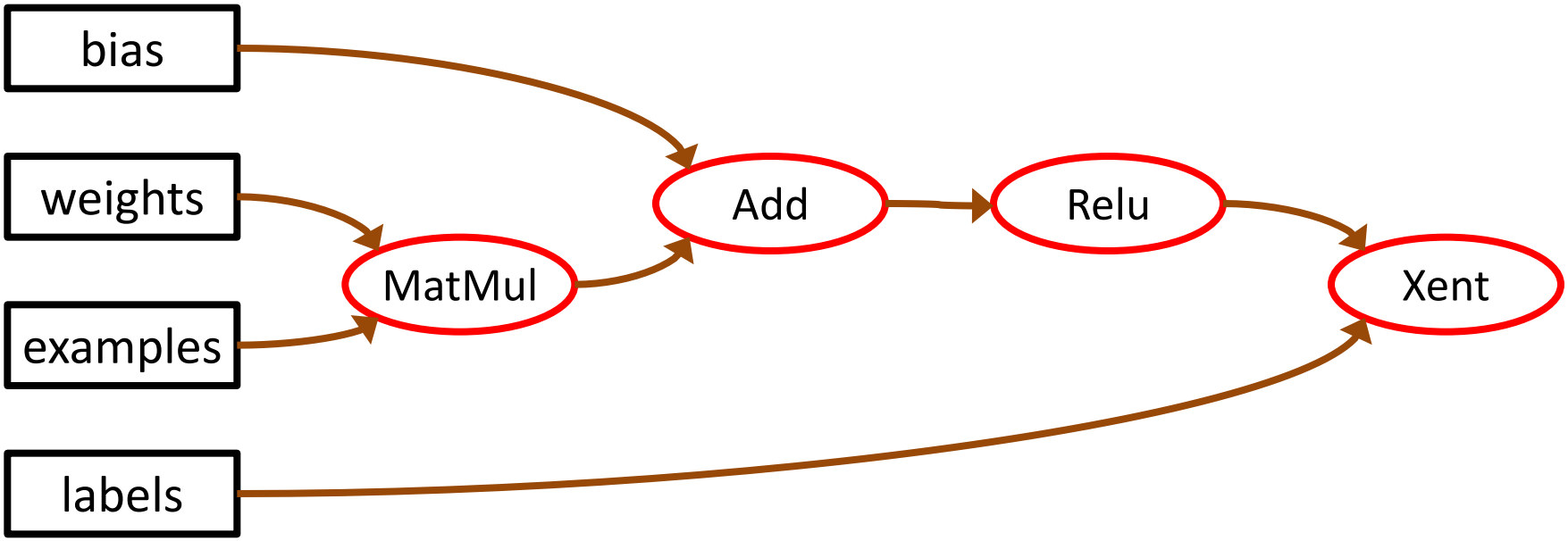
edges:

multidimensional data arrays (tensors)

communicated between nodes

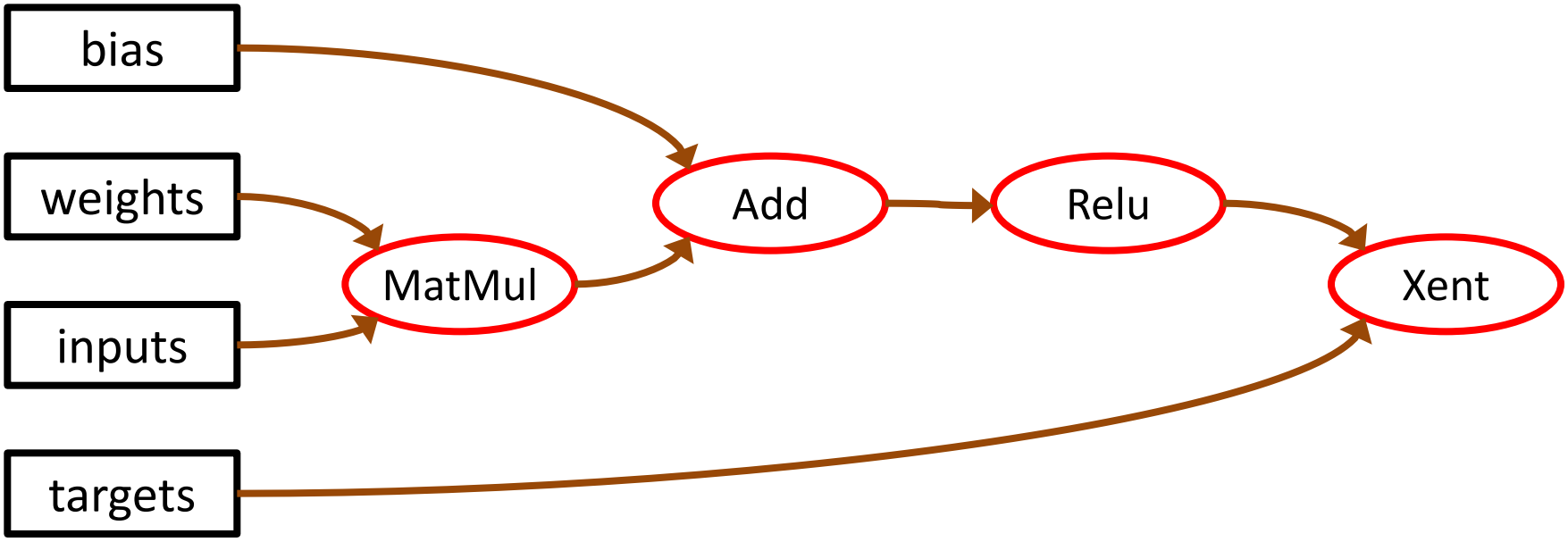
Computation is a Dataflow Graph

Graph of Nodes,
also called Operations or ops.



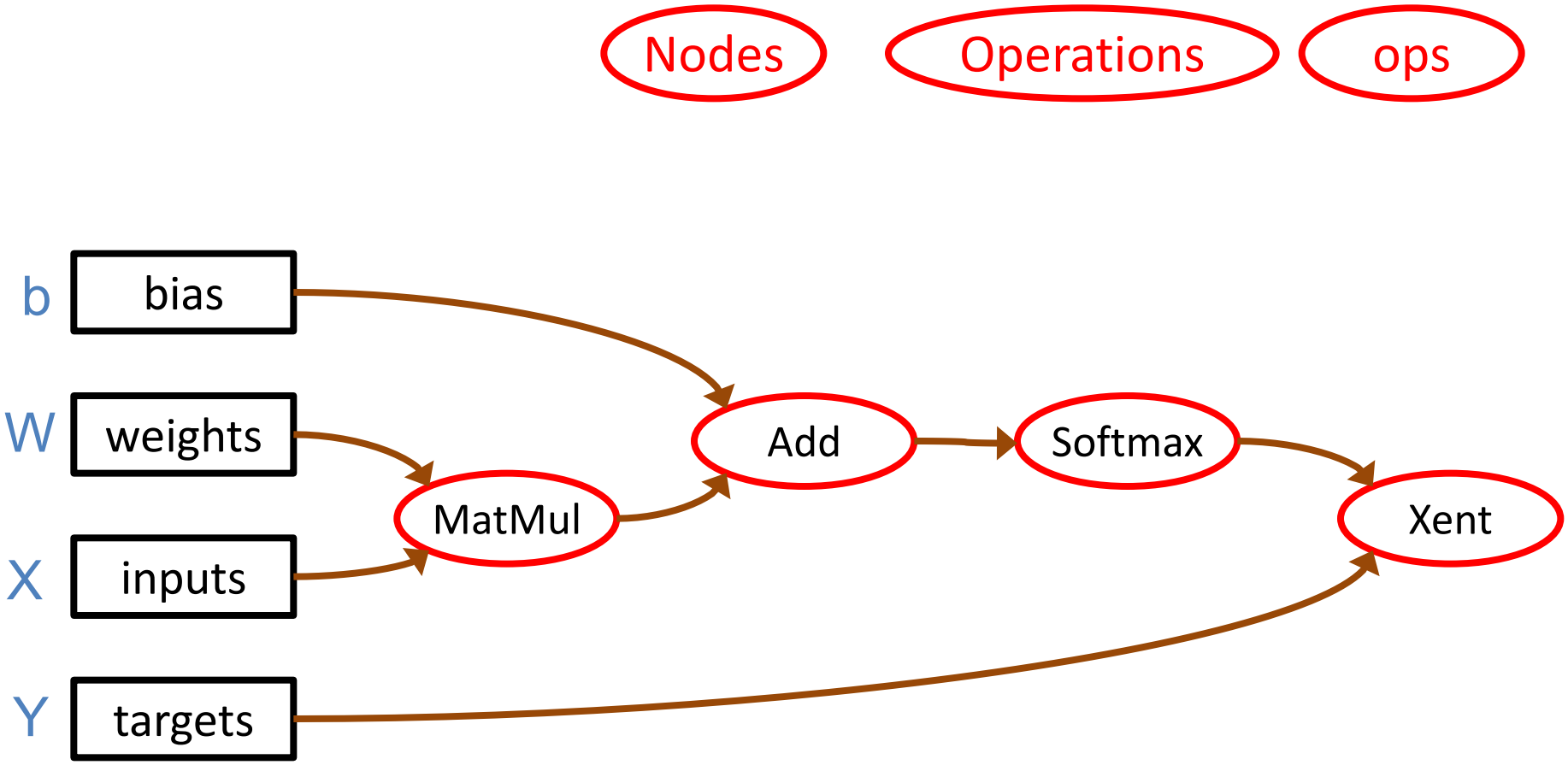
Computation is a Dataflow Graph

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



Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Logistic Regression as Dataflow Graph

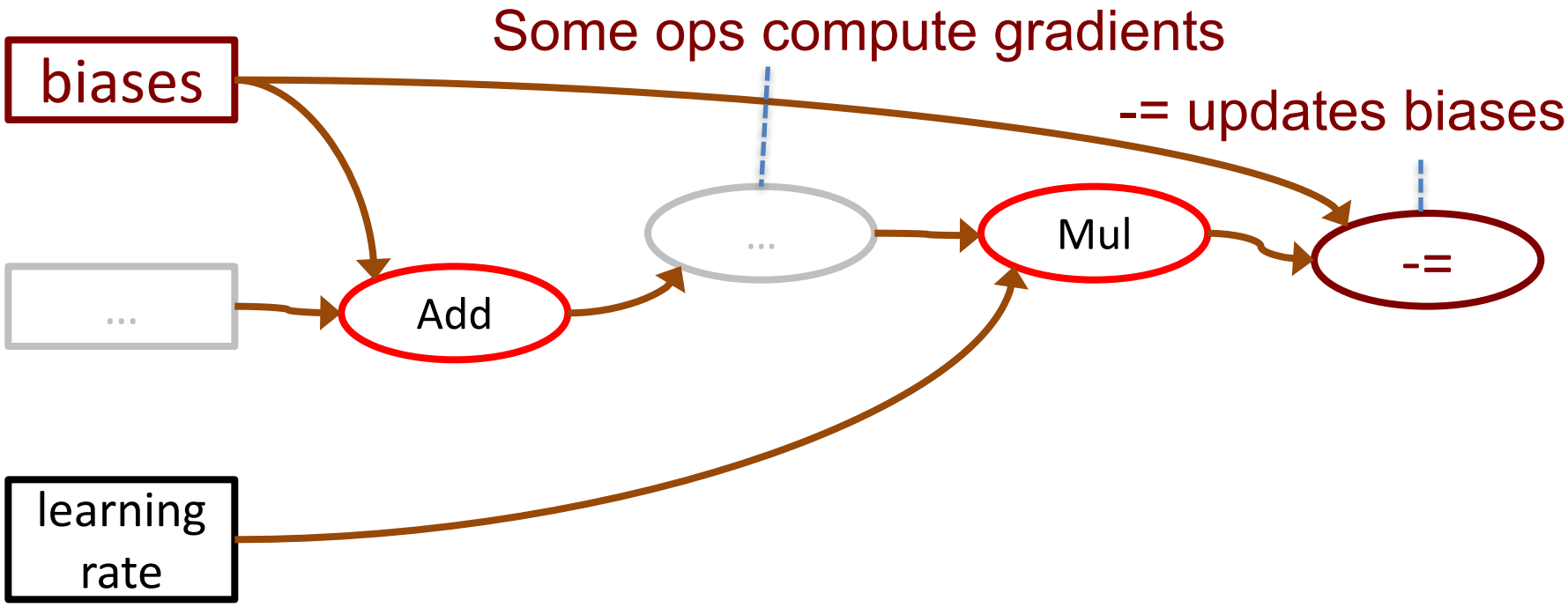


Edges are N-dimensional arrays: **Tensors**

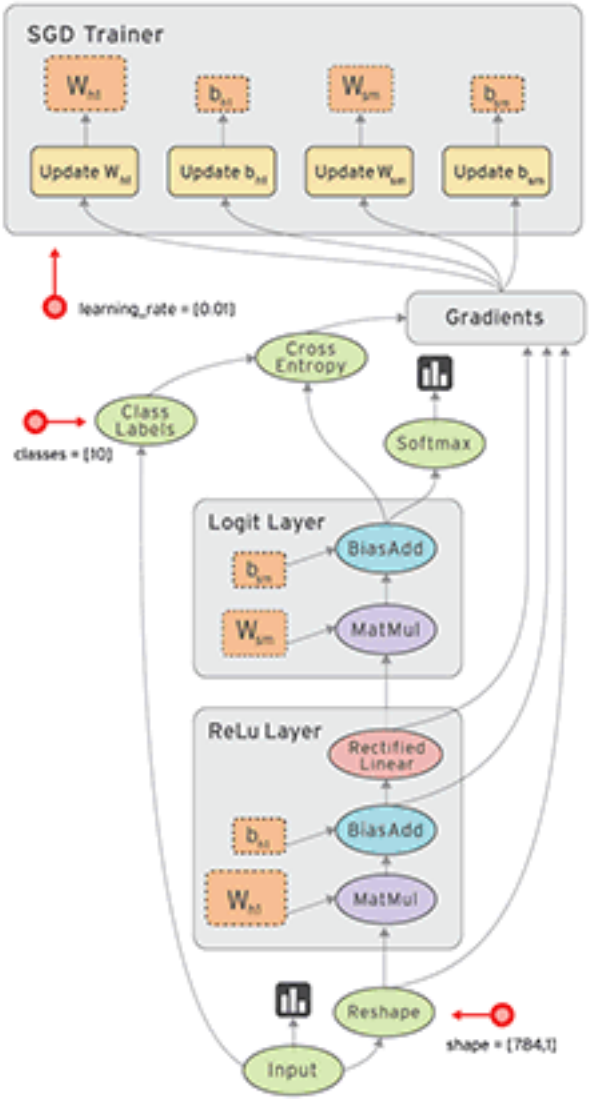
Computation is a Dataflow Graph

with state

‘Biases’ is a variable



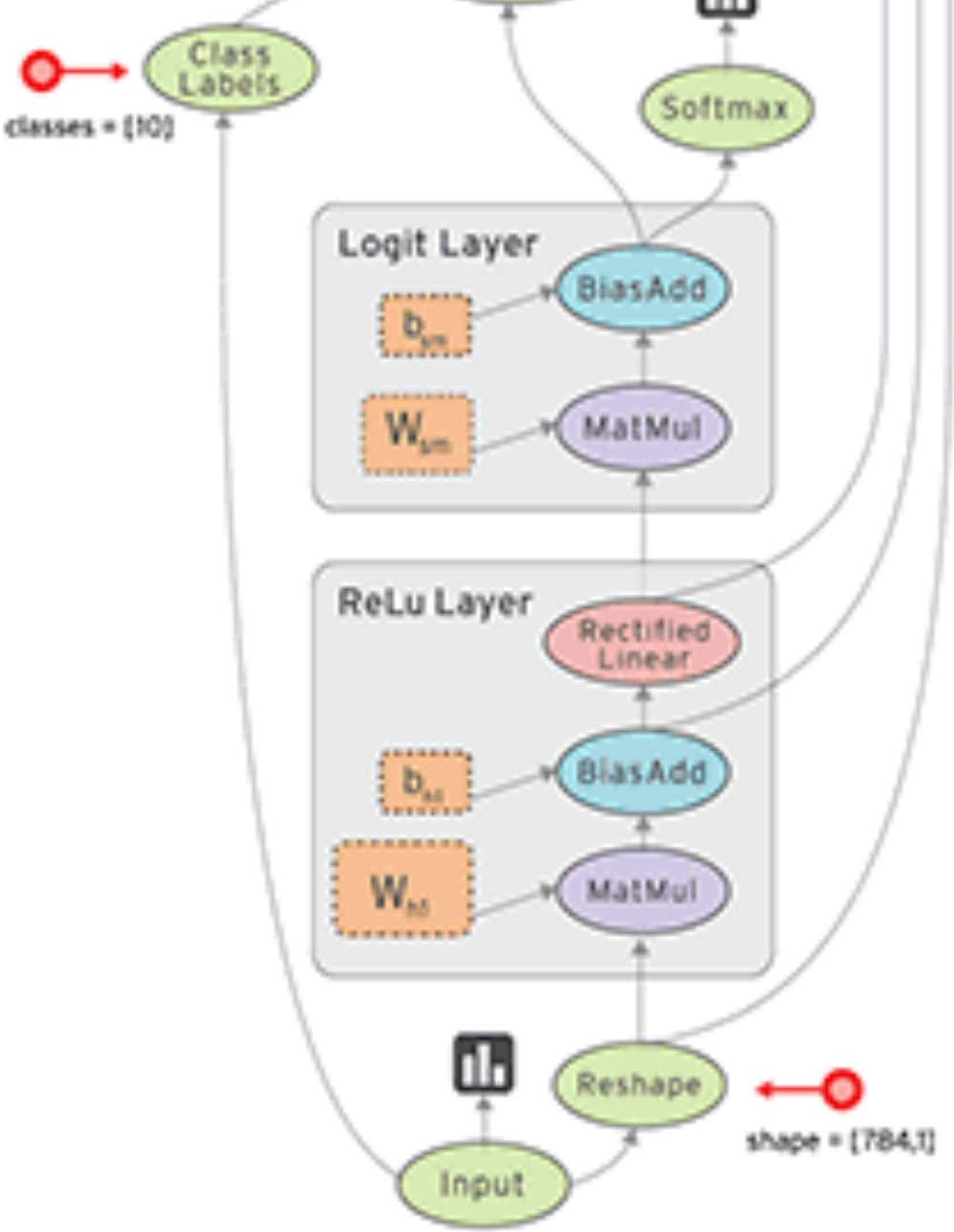
Data Flow Graph



Source: https://www.tensorflow.org/

TensorFlow

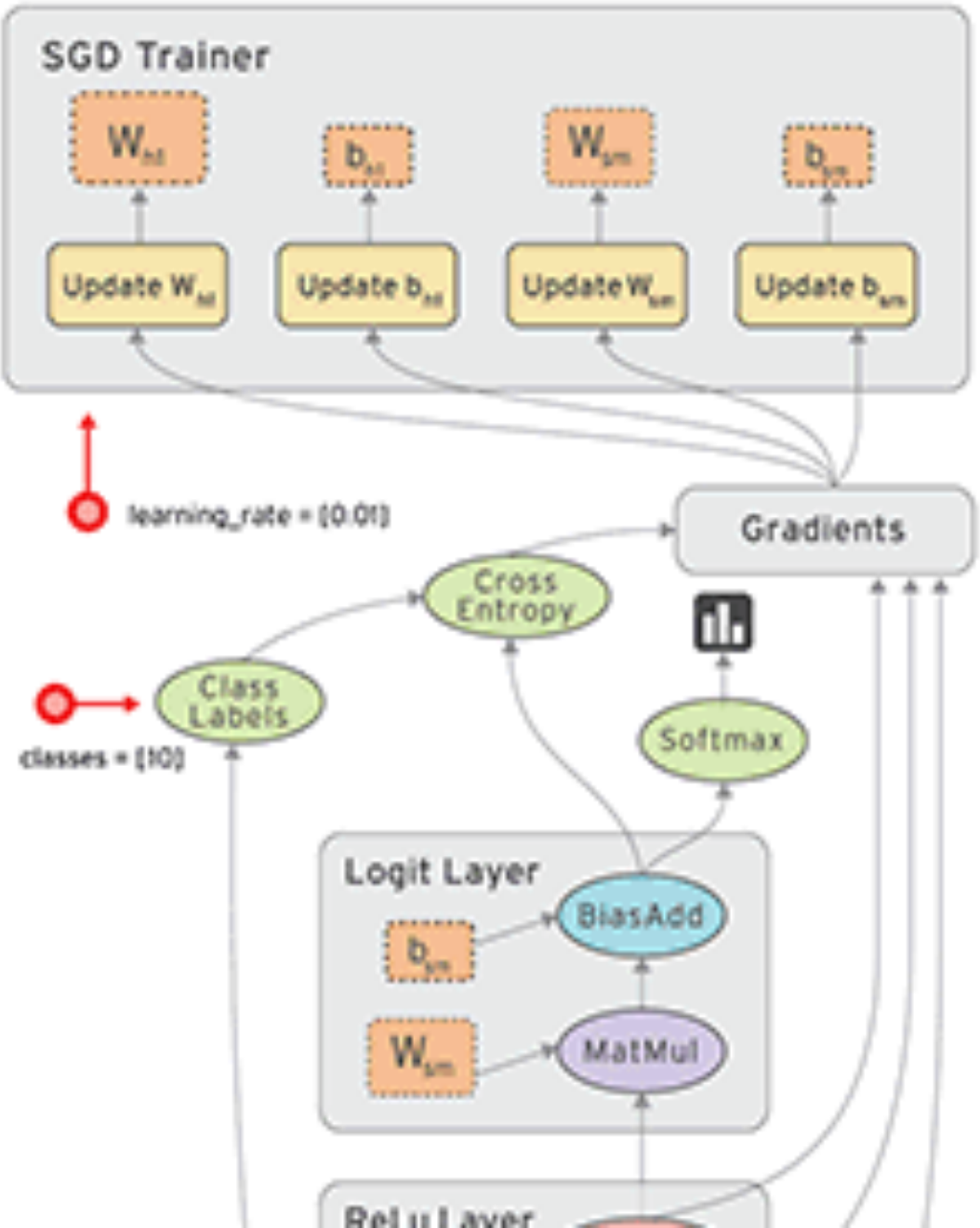
Data Flow Graph



Source: https://www.tensorflow.org/

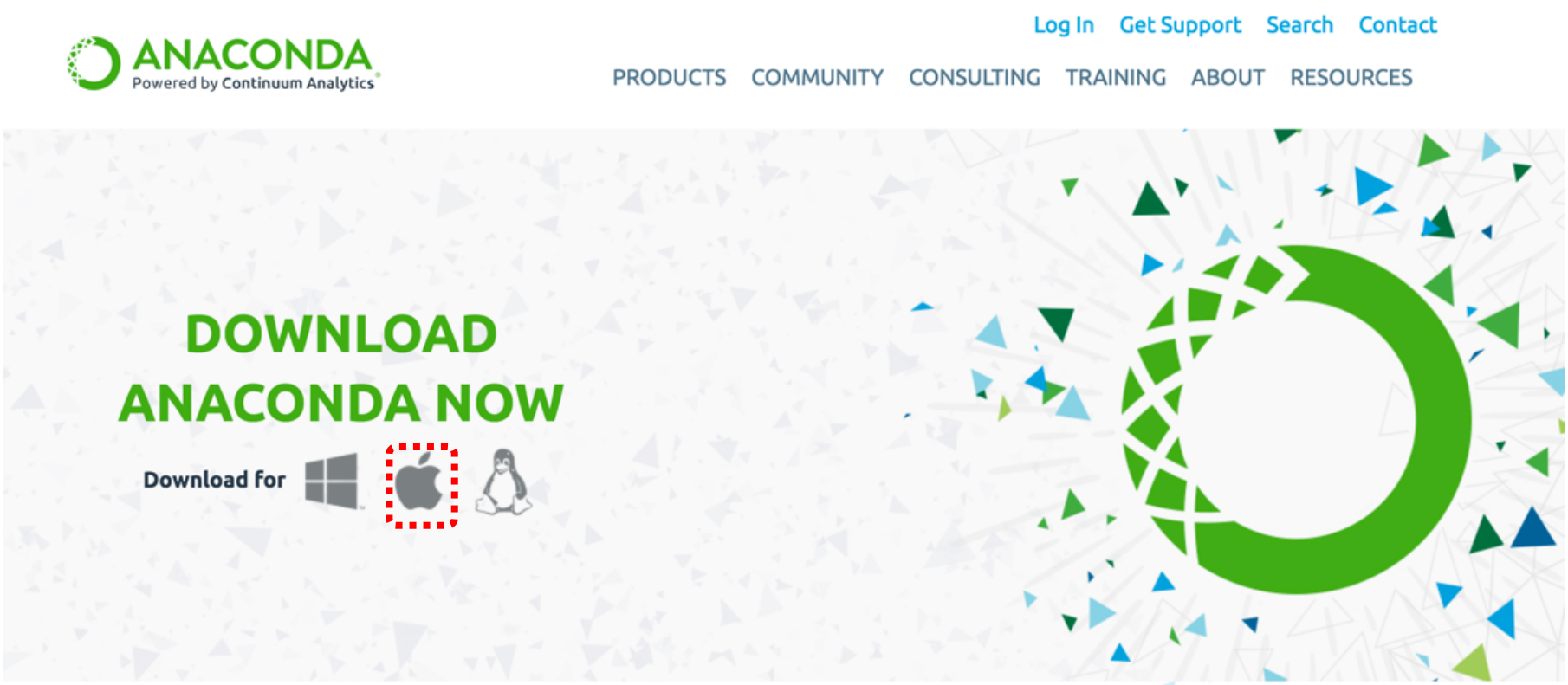
TensorFlow

Data Flow Graph



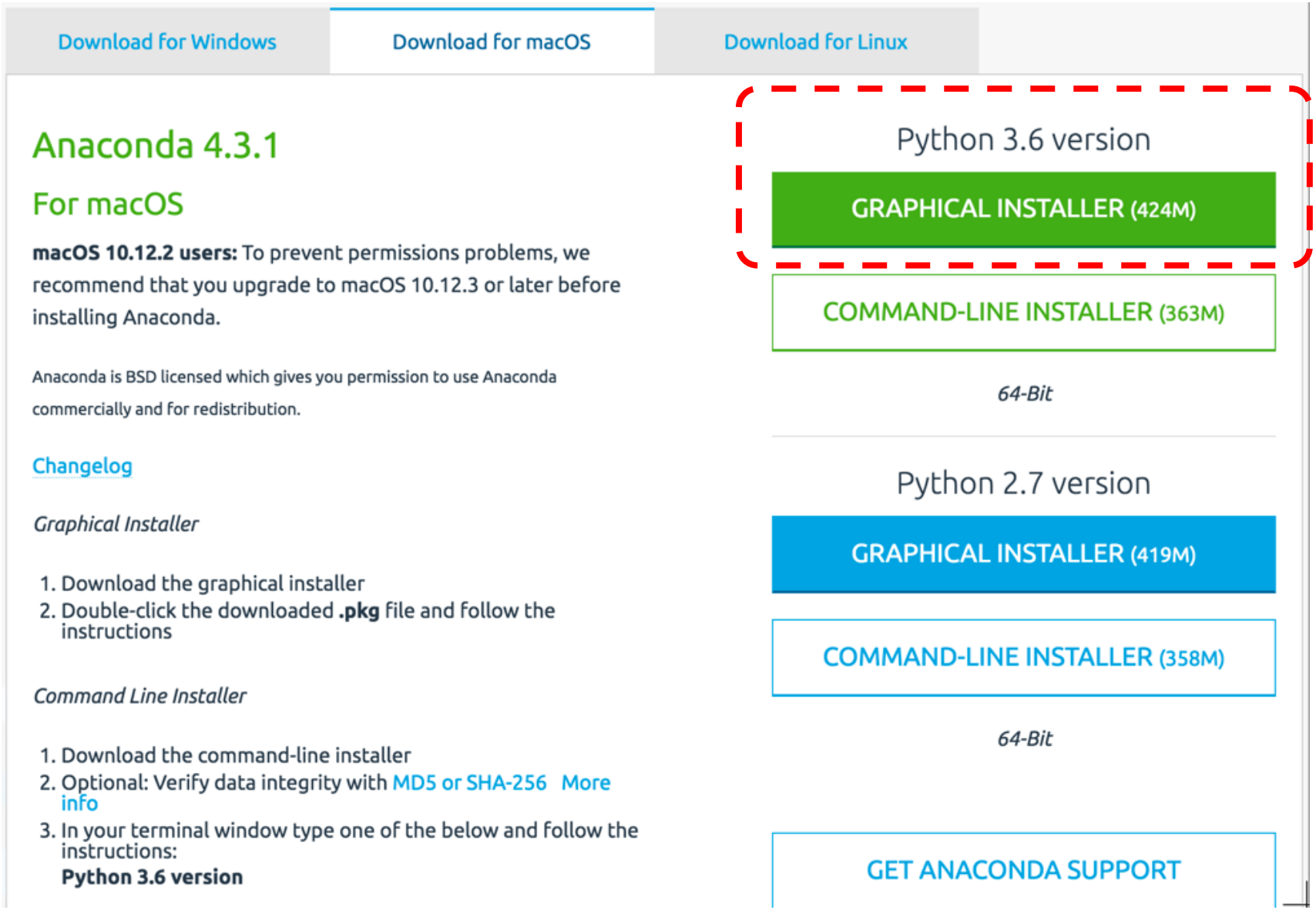
Python

Download Anaconda



https://www.continuum.io/downloads

Download Anaconda Python 3.6



https://www.continuum.io/downloads

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

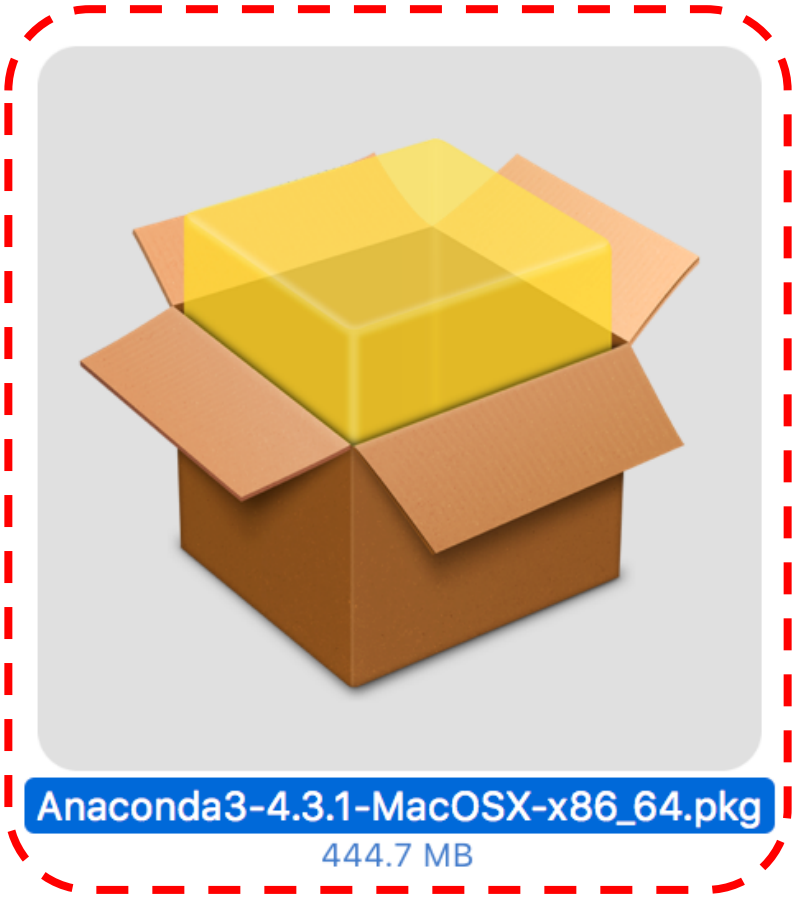
```
bash Anaconda2-4.3.1-MacOSX-x86_64.sh
```

https://www.continuum.io/downloads

OS X Anaconda 3 - 4.3.1

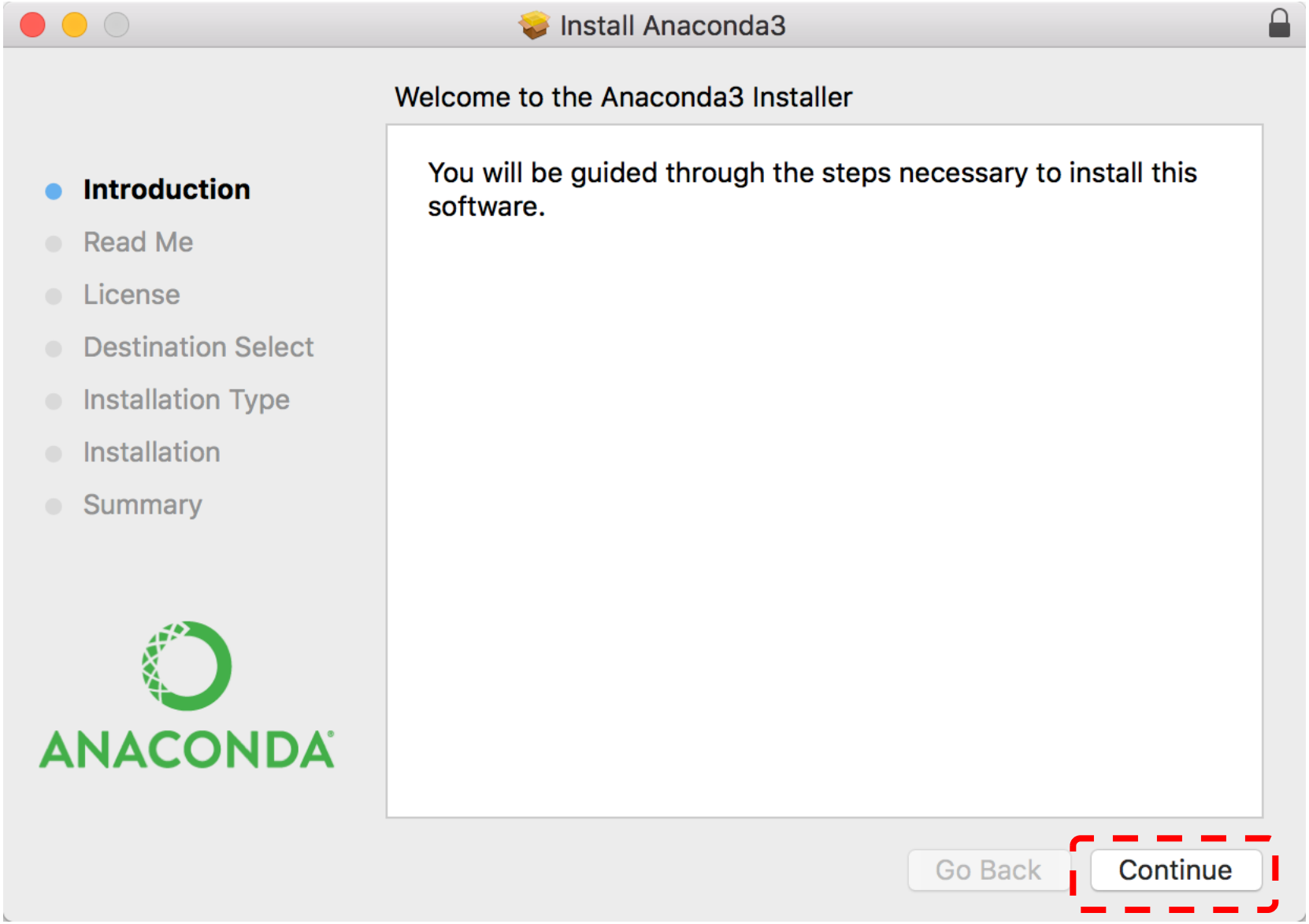
Python 3.6 Installation

Anaconda3-4.3.1-MacOSX-x86_64.pkg

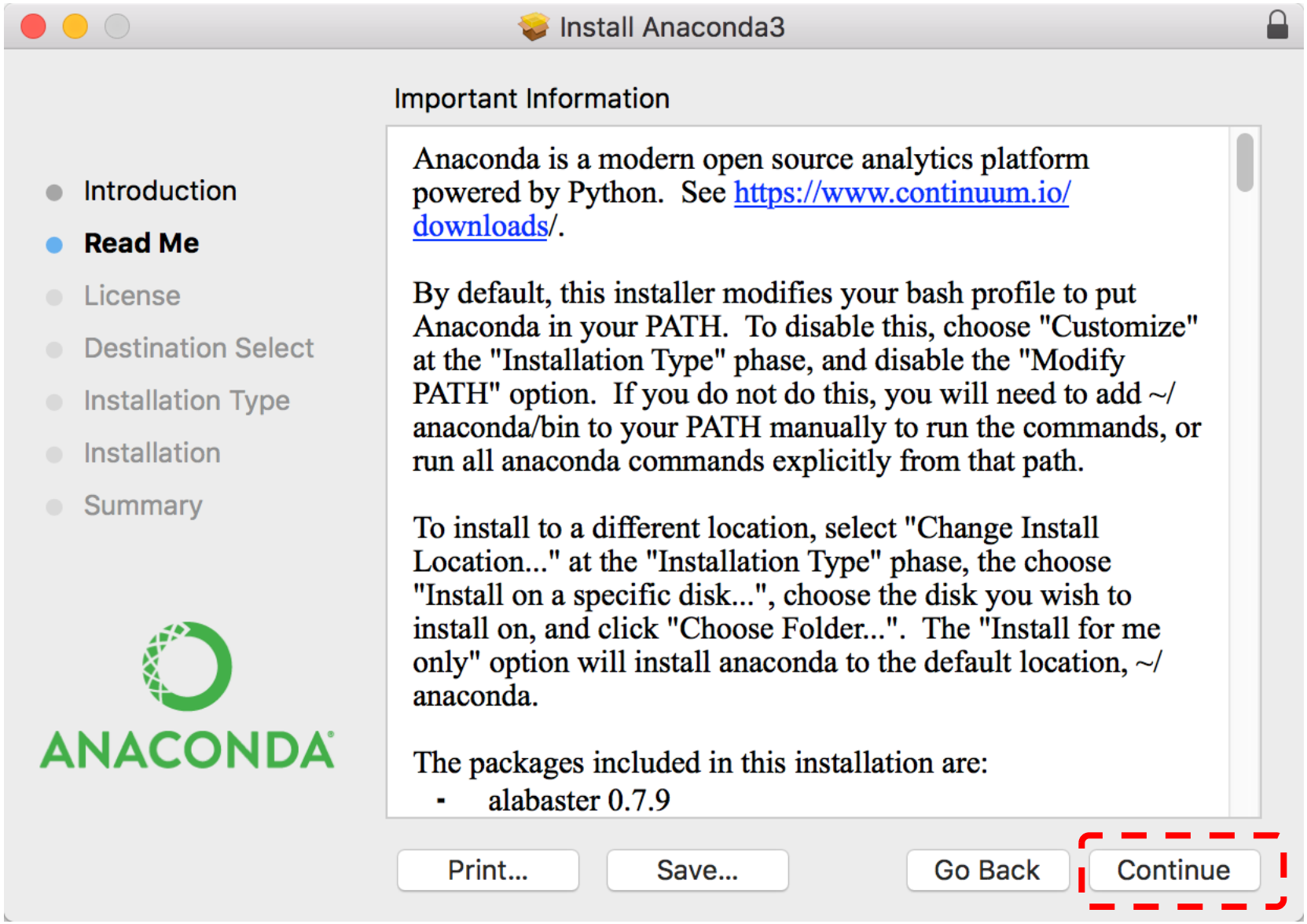


Installer package

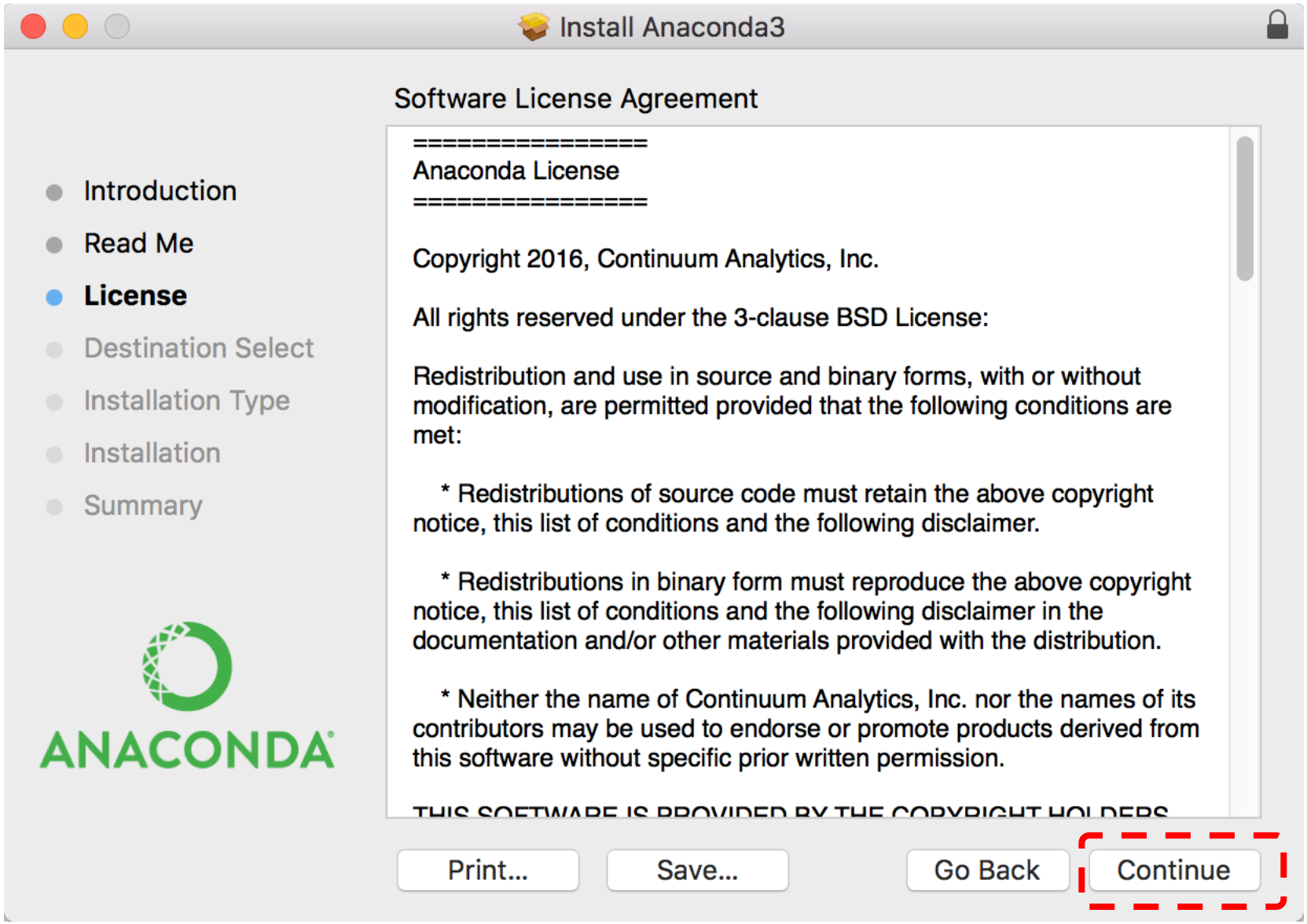
Install Anaconda 3



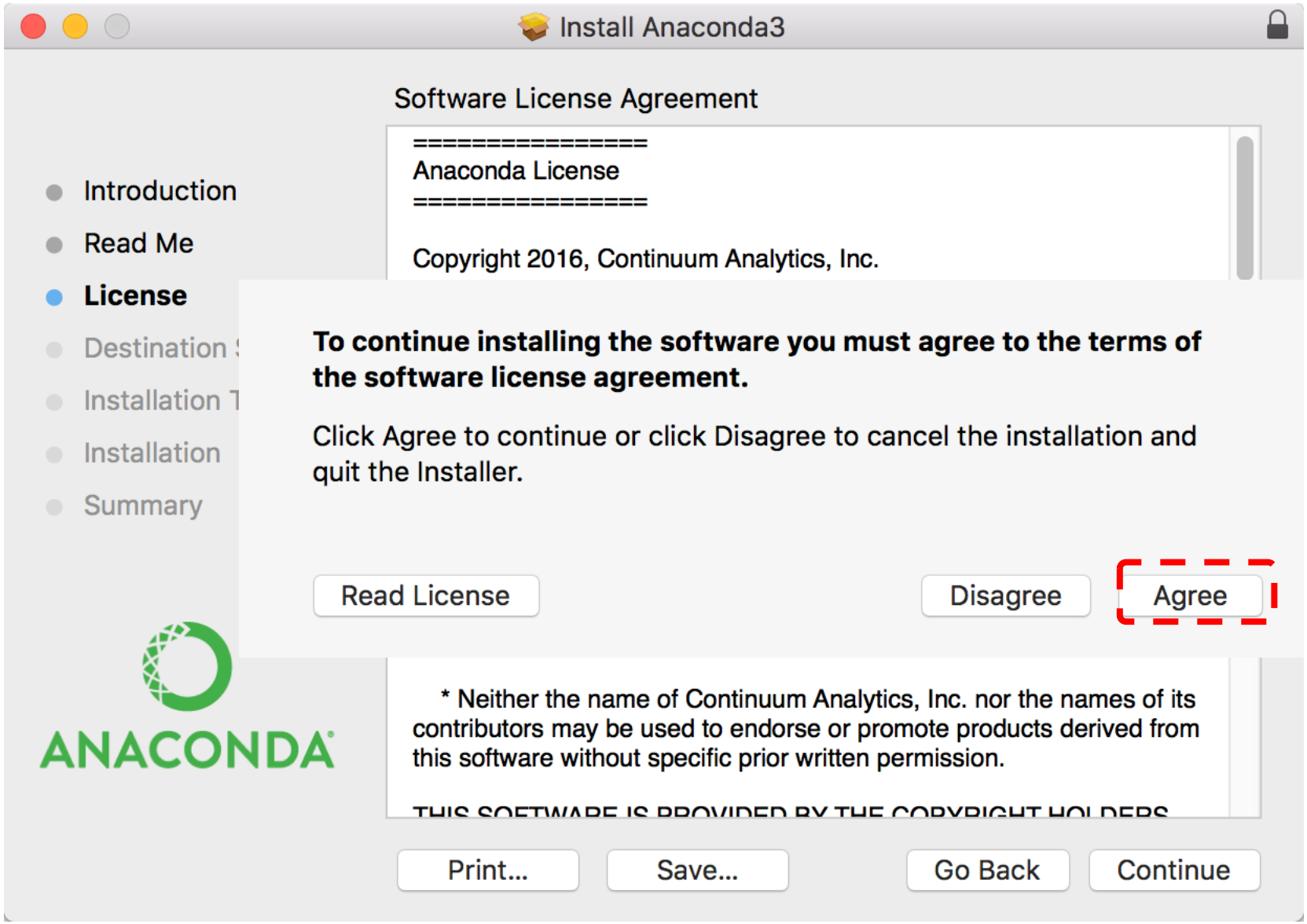
Install Anaconda 3



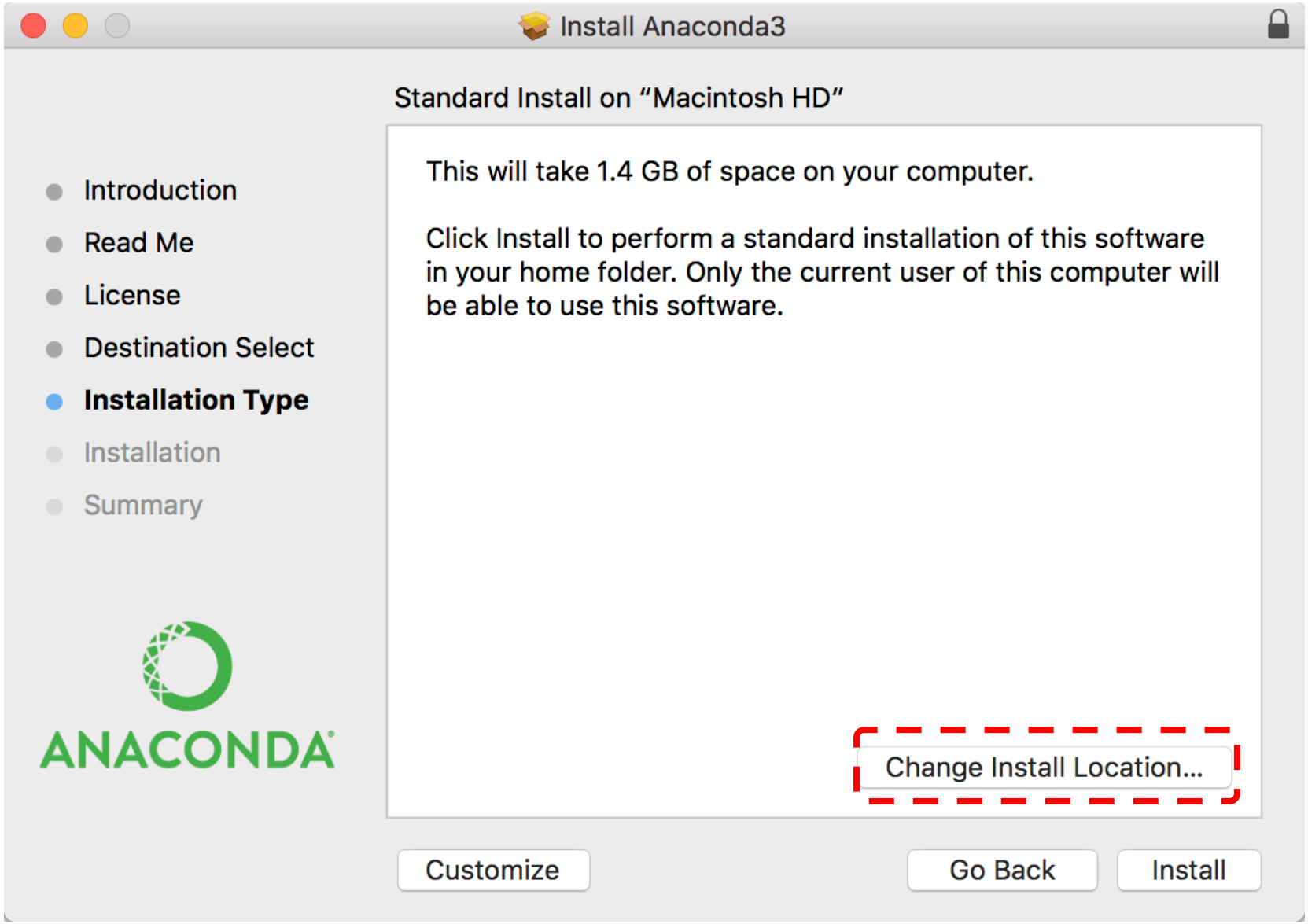
Install Anaconda 3



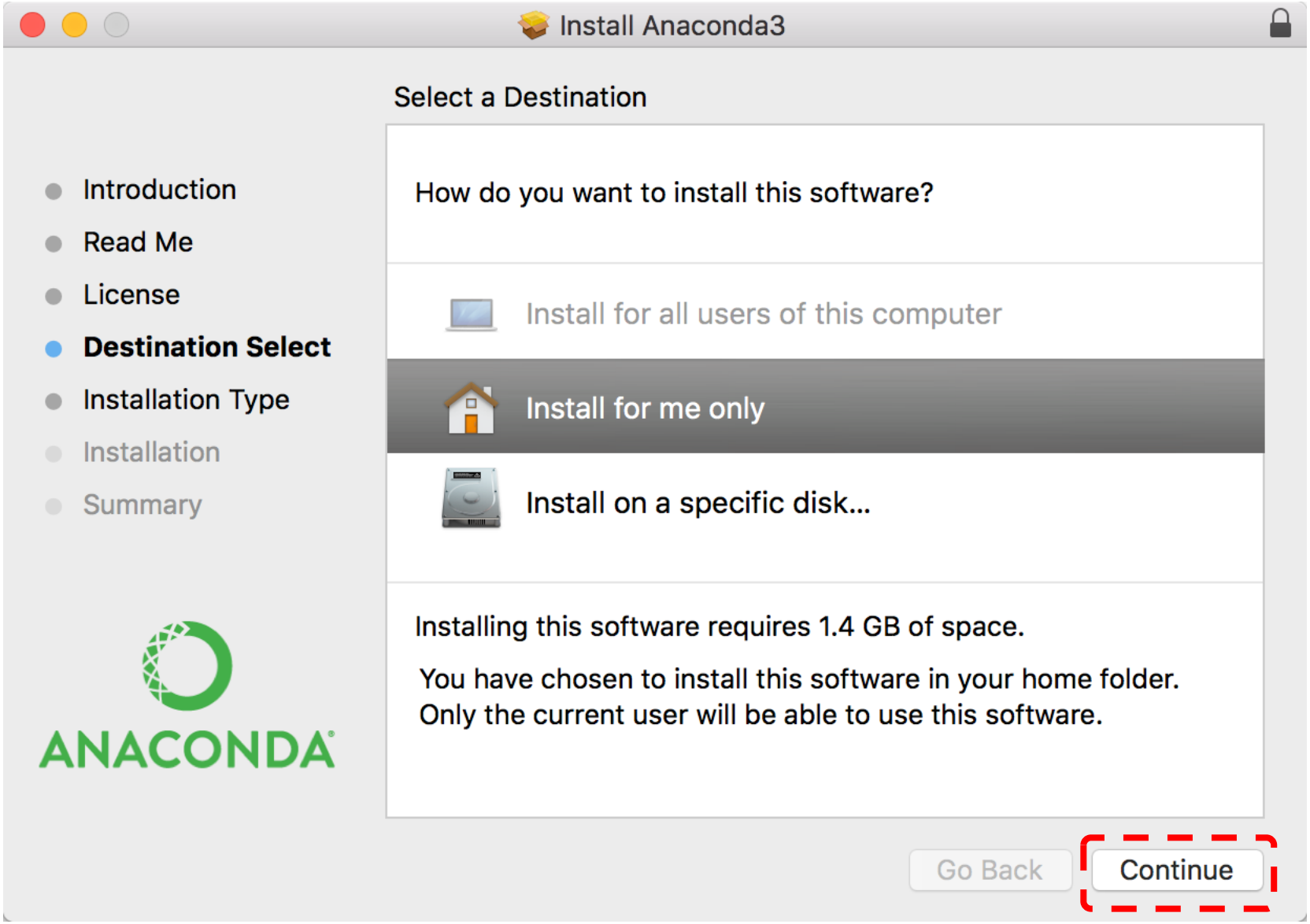
Install Anaconda 3



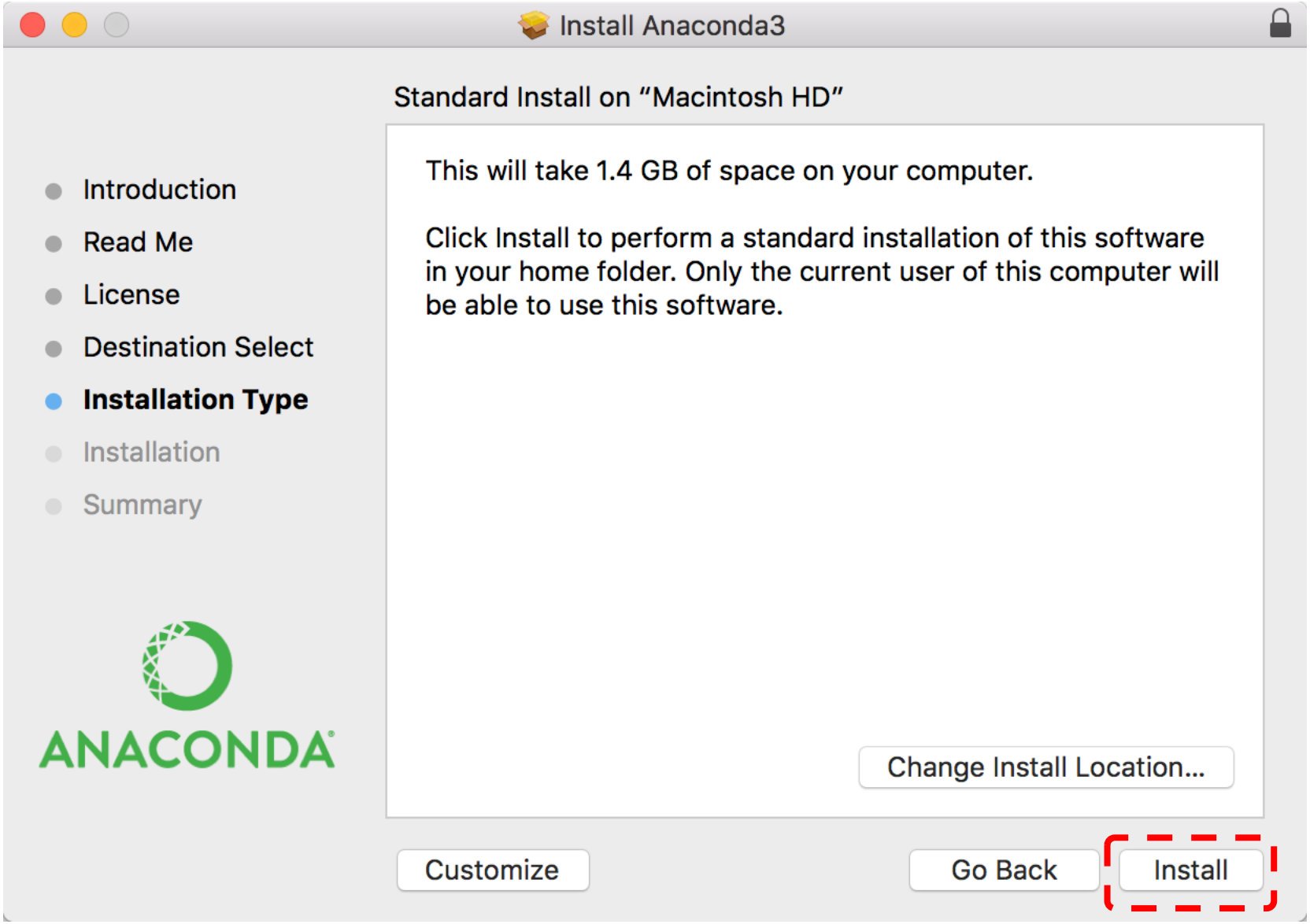
Install Anaconda 3



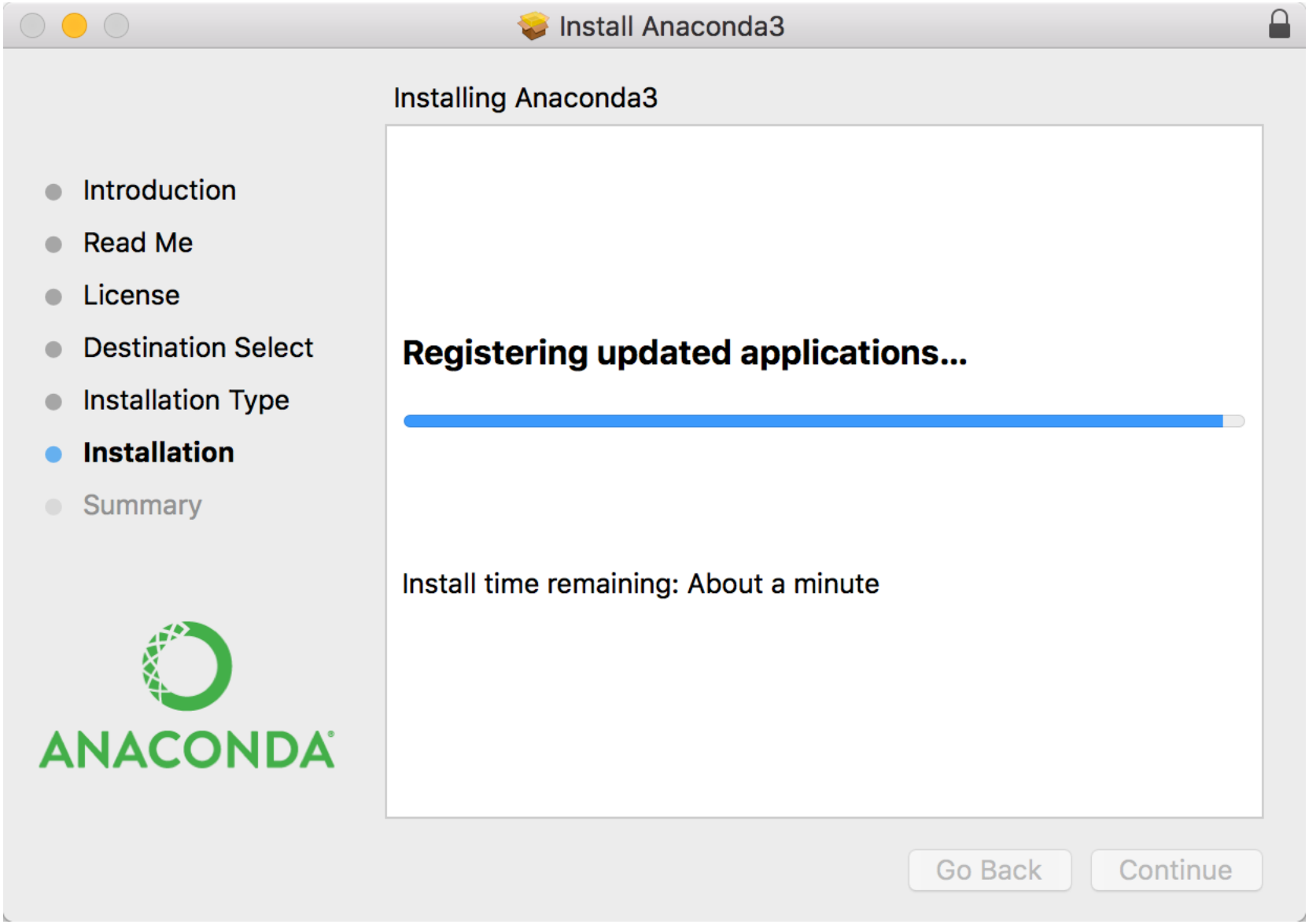
Install Anaconda 3



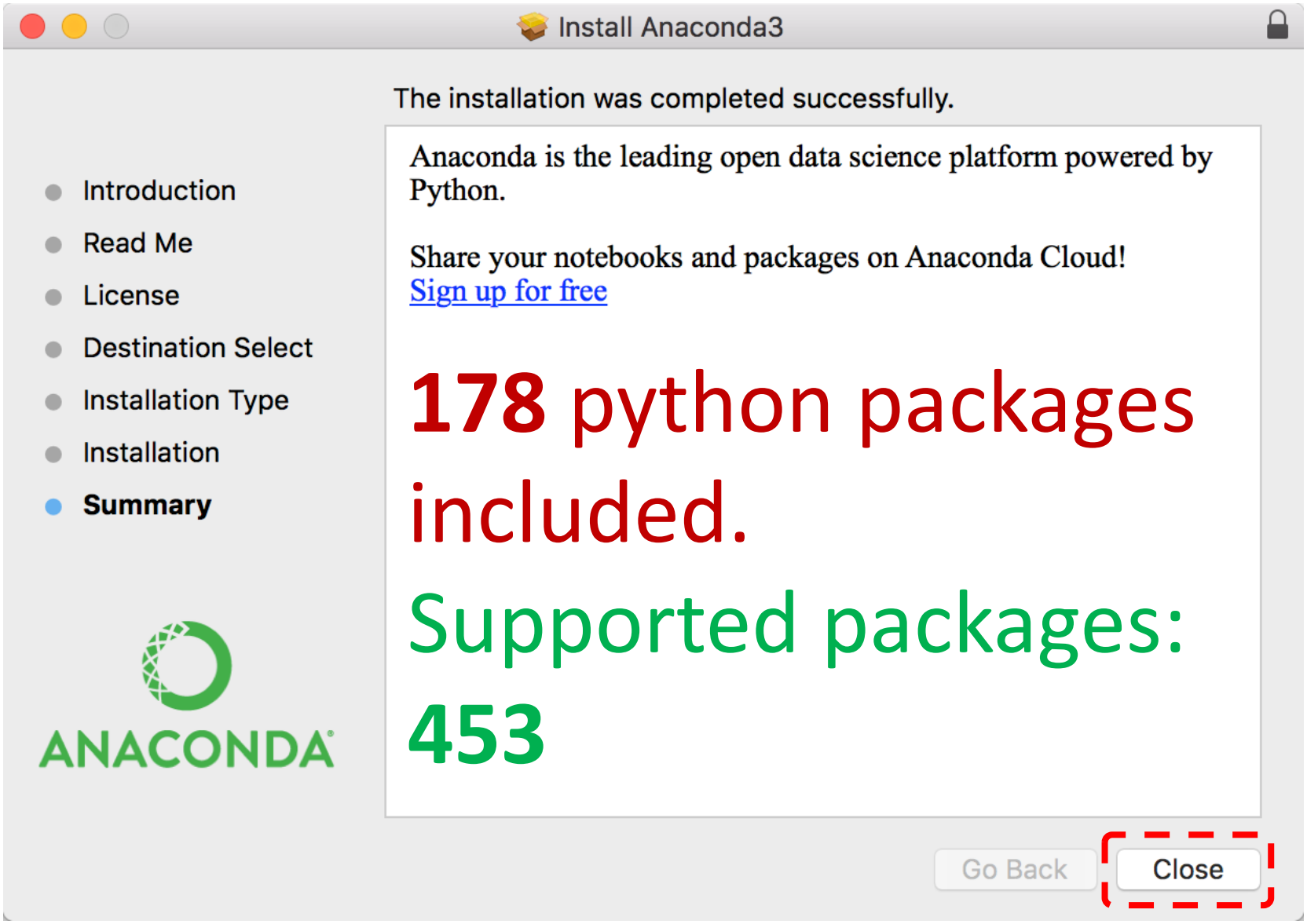
Install Anaconda 3



Install Anaconda 3



Install Anaconda 3



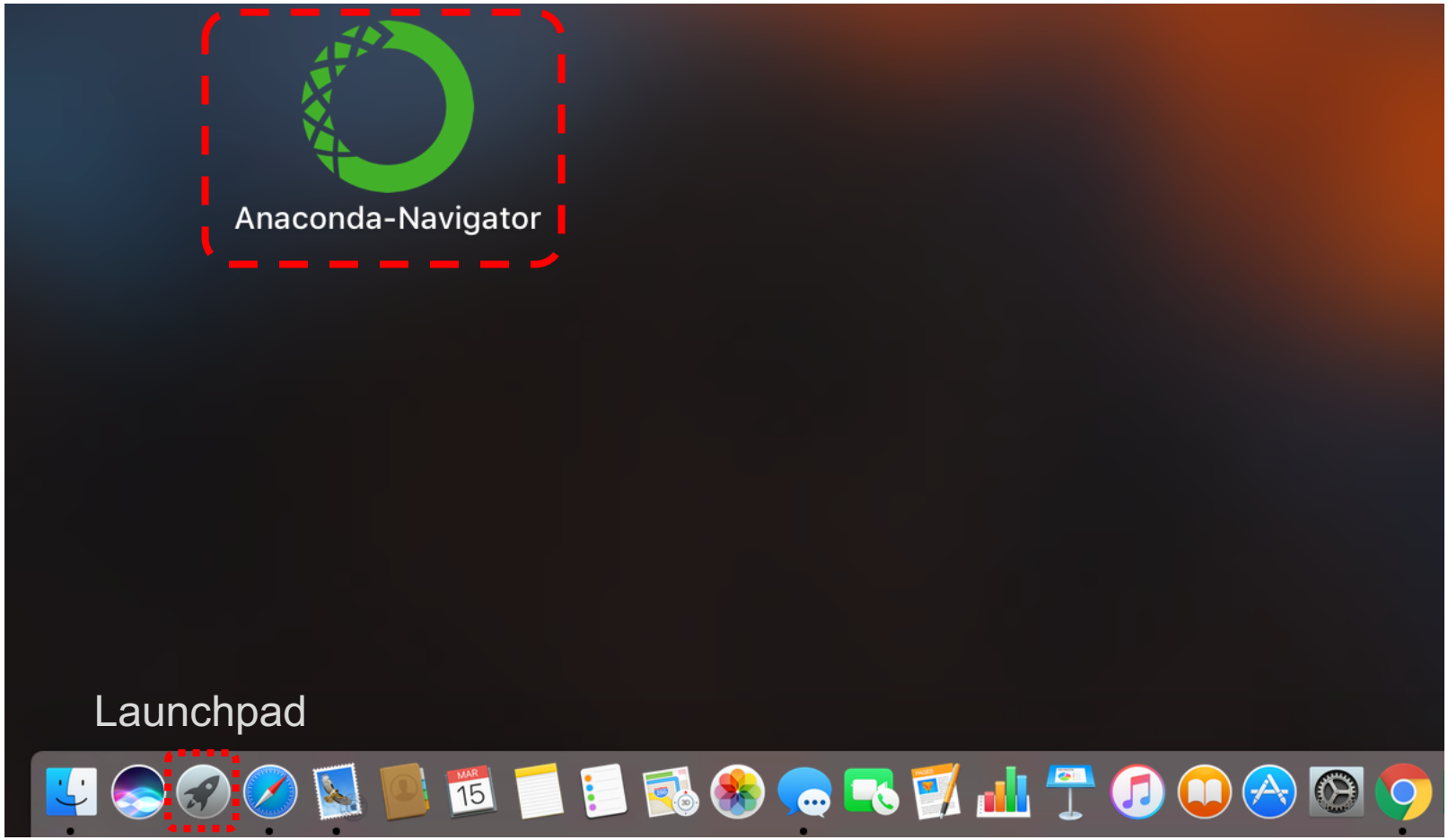
Source: https://docs.continuum.io/anaconda/pkg-docs

Install Anaconda 3

No.	Package	Version	Build
1	_license	1.1	
2	alabaster	0.7.9	
3	anaconda	4.3.1	
4	anaconda-client	1.6.0	
5	anaconda-navigator	1.5.0	
6	anaconda-project	0.4.1	
7	appnope	0.1.0	
8	appscript	1.0.1	
9	astroid	1.4.9	
10	astropy	1.3	
11	babel	2.3.4	
12	backports	1.0	
13	beautifulsoup4	4.5.3	
14	bitarray	0.8.1	
15	blaze	0.10.1	
16	bokeh	0.12.4	
17	boto	2.45.0	
18	bottleneck	1.2.0	
19	cffi	1.9.1	
20	chardet	2.3.0	
21	chest	0.2.3	
22	click	6.7	
23	cloudpickle	0.2.2	
24	clyent	1.2.2	
25	colorama	0.3.7	
26	conda	4.3.14	
27	conda-env	2.6.0	
28	configobj	5.0.6	
29	contextlib2	0.5.4	
30	cryptography	1.7.1	
31	curl	7.52.1	
32	cycler	0.10.0	
33	cython	0.25.2	
34	cytoolz	0.8.2	
35	dask	0.13.0	
36	datashape	0.5.4	
37	decorator	4.0.11	
38	dill	0.2.5	
39	docutils	0.13.1	
40	entrypoints	0.2.2	
41	et_xmlfile	1.0.1	
42	fastcache	1.0.2	
43	flask	0.12	
44	flask-cors	3.0.2	
45	freetype	2.5.5	
46	get_terminal_size	1.0.0	
47	gevent	1.2.1	
48	greenlet	0.4.11	
49	h5py	2.6.0	
50	hdf5	1.8.17	
51	heapdict	1.0.0	
52	icu	54.1	
53	idna	2.2	
54	imagesize	0.7.1	
55	ipykernel	4.5.2	
56	ipython	5.1.0	
57	ipython_genutils	0.1.0	
58	ipywidgets	5.2.2	
59	isort	4.2.5	
60	itsdangerous	0.24	
61	jbig	2.1	
62	jdcal	1.3	
63	jedi	0.9.0	
64	jinja2	2.9.4	
65	jpeg	9b	
66	jsonschema	2.5.1	
67	jupyter	1.0.0	
68	jupyter_client	4.4.0	
69	jupyter_console	5.0.0	
70	jupyter_core	4.2.1	
71	lazy-object-proxy	1.2.2	
72	libiconv	1.14	
73	libpng	1.6.27	
74	libtiff	4.0.6	
75	libxml2	2.9.4	
76	libxslt	1.1.29	
77	llvmlite	0.15.0	
78	locket	0.2.0	
79	lxml	3.7.2	
80	markupsafe	0.23	
81	matplotlib	2.0.0	
82	mistune	0.7.3	
83	mkl	2017.0.1	
84	mkl-service	1.1.2	
85	mpmath	0.19	
86	multipledispatch	0.4.9	
87	nbconvert	4.2.0	
88	nbformat	4.2.0	
89	networkx	1.11	
90	nltk	3.2.2	
91	nose	1.3.7	
92	notebook	4.3.1	
93	numba	0.30.1	
94	numexpr	2.6.1	
95	numpy	1.11.3	
96	numpydoc	0.6.0	
97	odo	0.5.0	
98	openpyxl	2.4.1	
99	openssl	1.0.2k	
100	pandas	0.19.2	
101	partd	0.3.7	
102	path.py	10.0	
103	pathlib2	2.2.0	
104	patsy	0.4.1	
105	pep8	1.7.0	
106	pexpect	4.2.1	
107	pickleshare	0.7.4	
108	pillow	4.0.0	
109	pip	9.0.1	
110	ply	3.9	
111	prompt_toolkit	1.0.9	
112	psutil	5.0.1	
113	ptyprocess	0.5.1	
114	py	1.4.32	
115	pyasn1	0.1.9	
116	pycosat	0.6.1	
117	pycparser	2.17	
118	pycrypto	2.6.1	
119	pycurl	7.43.0	
120	pyflakes	1.5.0	
121	pygments	2.1.3	
122	pylint	1.6.4	
123	pyopenssl	16.2.0	
124	pyparsing	2.1.4	
125	pyqt	5.6.0	
126	pytables	3.3.0	
127	pytest	3.0.5	
128	python	3.6.0	
129	python-dateutil	2.6.0	py36_0
130	python.app	1.2	py36_4
131	pytz	2016.10	py36_0
132	pyyaml	3.12	py36_0
133	pyzmq	16.0.2	py36_0
134	qt	5.6.2	0
135	qtawesome	0.4.3	py36_0
136	qtconsole	4.2.1	py36_1
137	qtpy	1.2.1	py36_0
138	readline	6.2	2
139	redis	3.2.0	0
140	redis-py	2.10.5	py36_0
141	requests	2.12.4	py36_0
142	rope	0.9.4	py36_1
143	ruamel_yaml	0.11.14	py36_1
144	scikit-image	0.12.3	np111py36_1
145	scikit-learn	0.18.1	np111py36_1
146	scipy	0.18.1	np111py36_1
147	seaborn	0.7.1	py36_0
148	setuptools	27.2.0	py36_0
149	simplegeneric	0.8.1	py36_1
150	singledispatch	3.4.0.3	py36_0
151	sip	4.18	py36_0
152	six	1.10.0	py36_0
153	snowballstemmer	1.2.1	py36_0
154	sockjs-tornado	1.0.3	py36_0
155	sphinx	1.5.1	py36_0
156	spyder	3.1.2	py36_0
157	sqlalchemy	1.1.5	py36_0
158	sqlite	3.13.0	0
159	statsmodels	0.6.1	np111py36_1
160	sympy	1.0	py36_0
161	terminado	0.6	py36_0
162	tk	8.5.18	0
163	toolz	0.8.2	py36_0
164	tornado	4.4.2	py36_0
165	traitlets	4.3.1	py36_0
166	unicodecsv	0.14.1	py36_0
167	wcwidth	0.1.7	py36_0
168	werkzeug	0.11.15	py36_0
169	wheel	0.29.0	py36_0
170	widgetsnbextension	1.2.6	py36_0
171	wrapt	1.10.8	py36_0
172	xlrd	1.0.0	py36_0
173	xlsxwriter	0.9.6	py36_0
174	xlwings	0.10.2	py36_0
175	xlwt	1.2.0	py36_0
176	xz	5.2.2	1
177	yaml	0.1.6	0
178	zlib	1.2.8	3

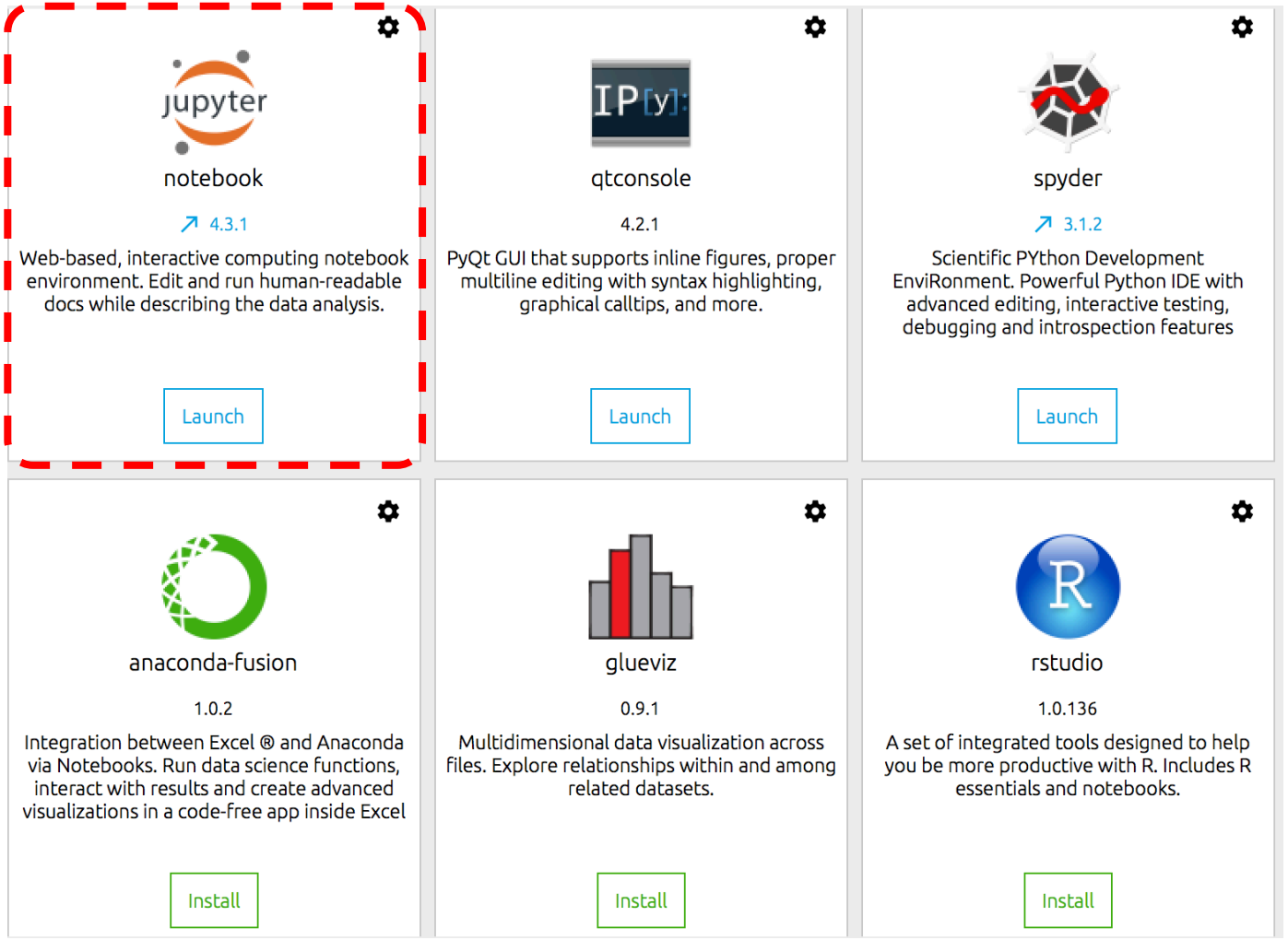
178 python packages included.

Anaconda-Navigator

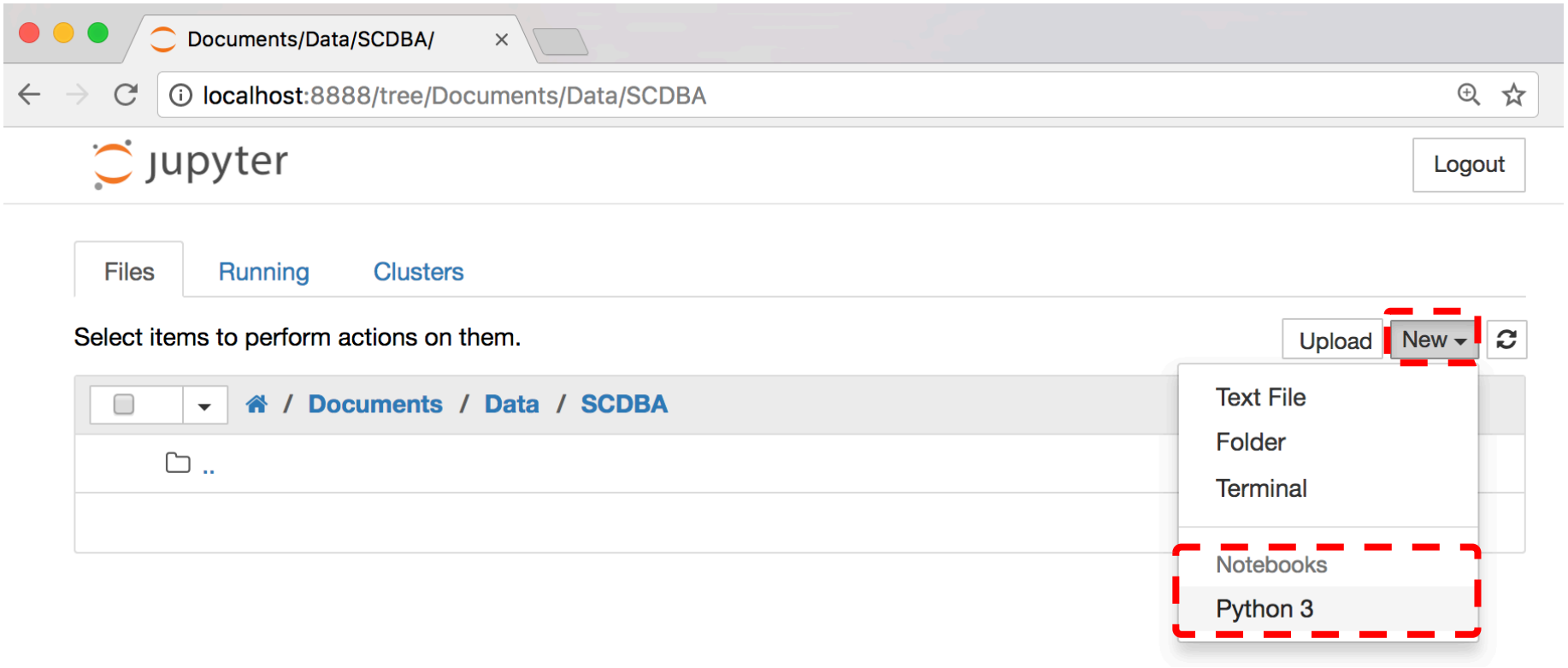


Anaconda-Navigator

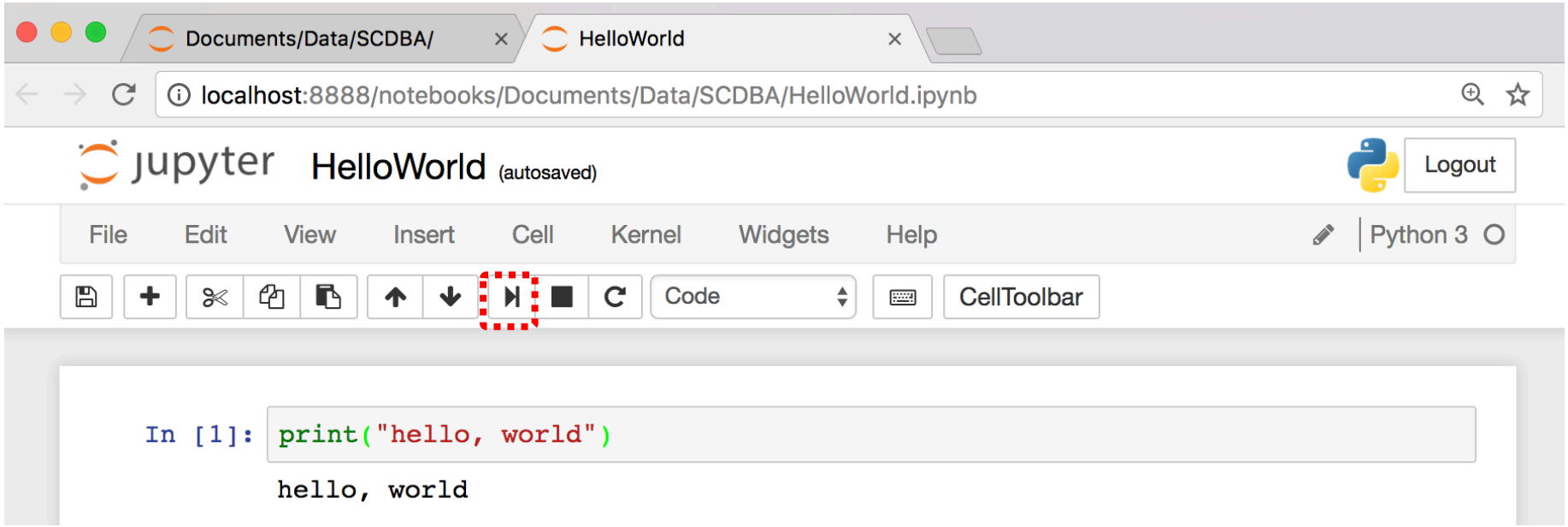
Jupyter Notebook



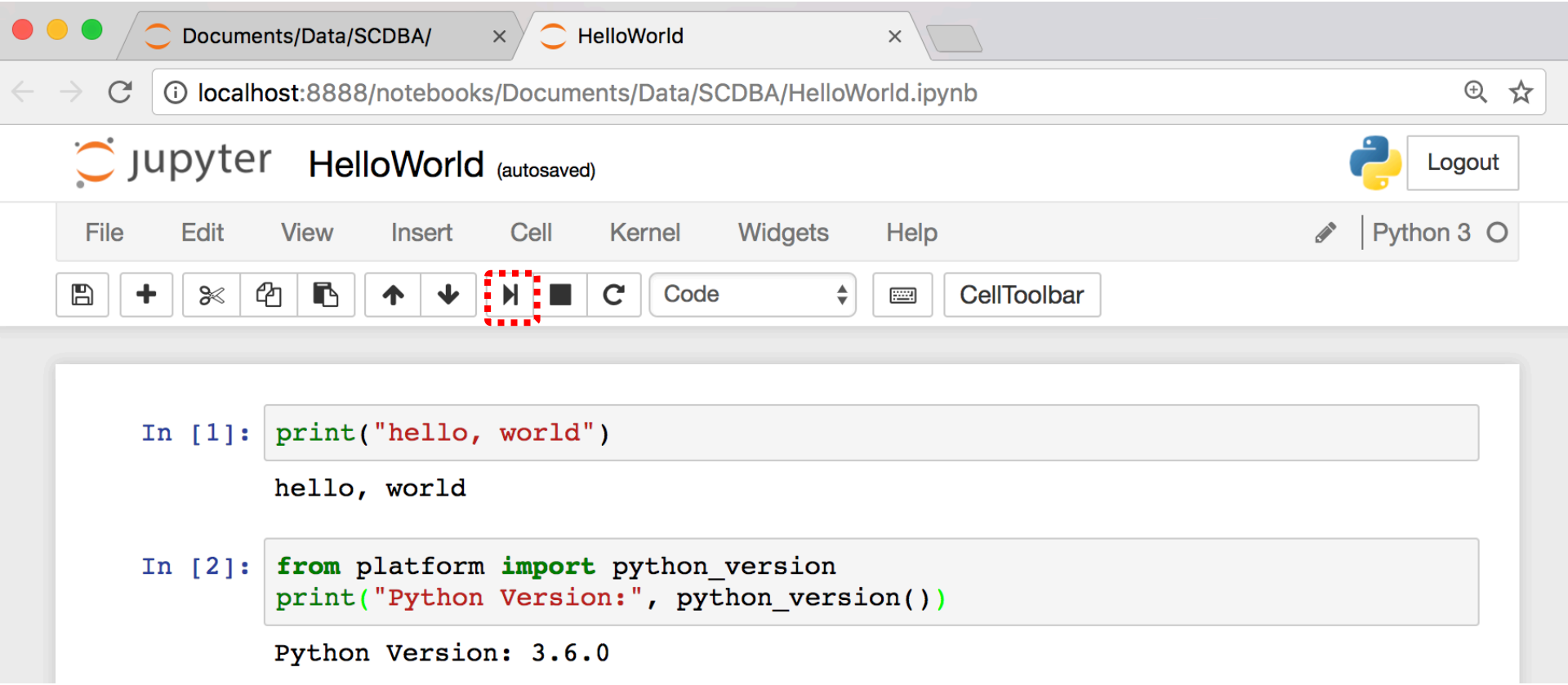
Jupyter Notebook
New Python 3



print("hello, world")



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

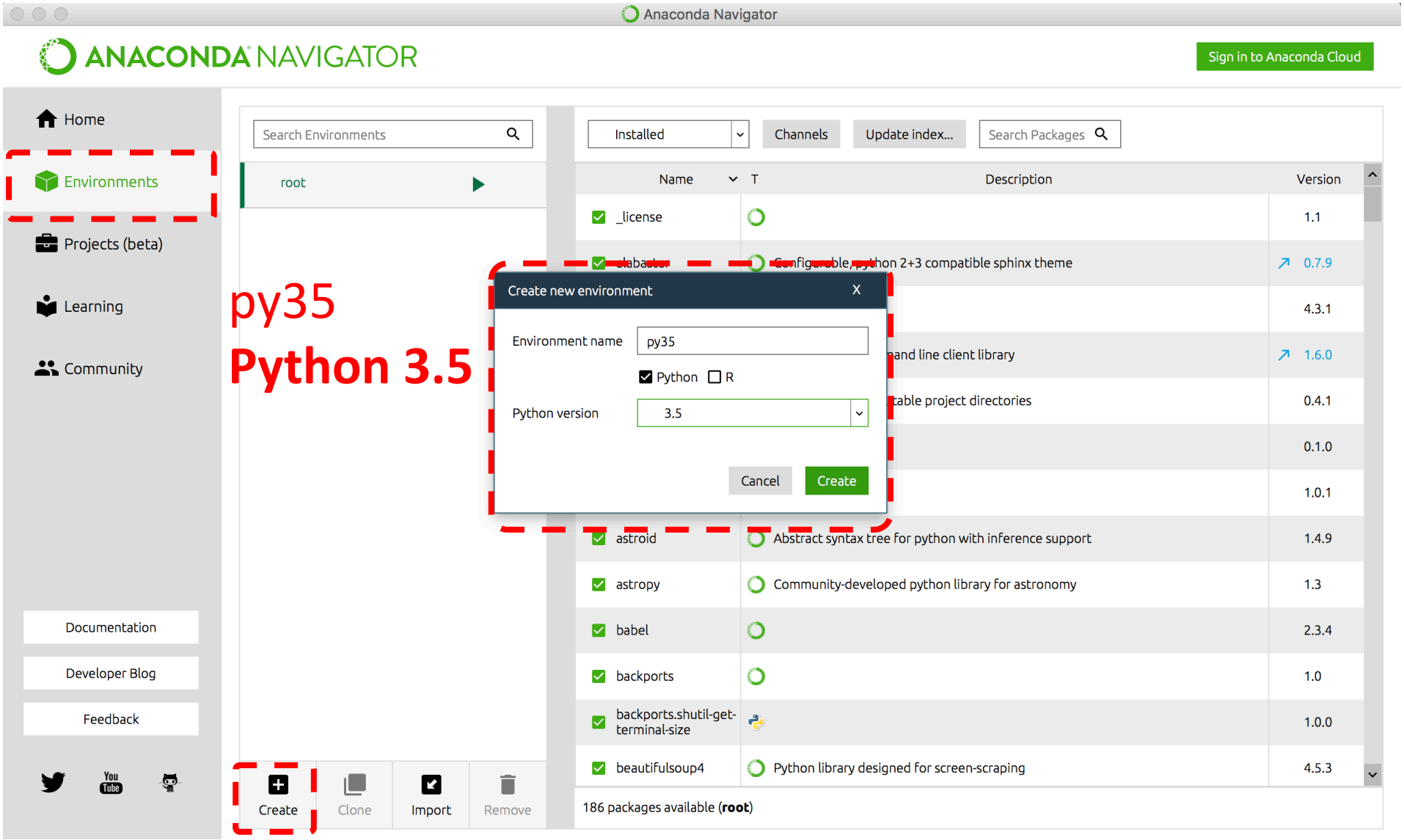


Create Python Environments with Anaconda

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

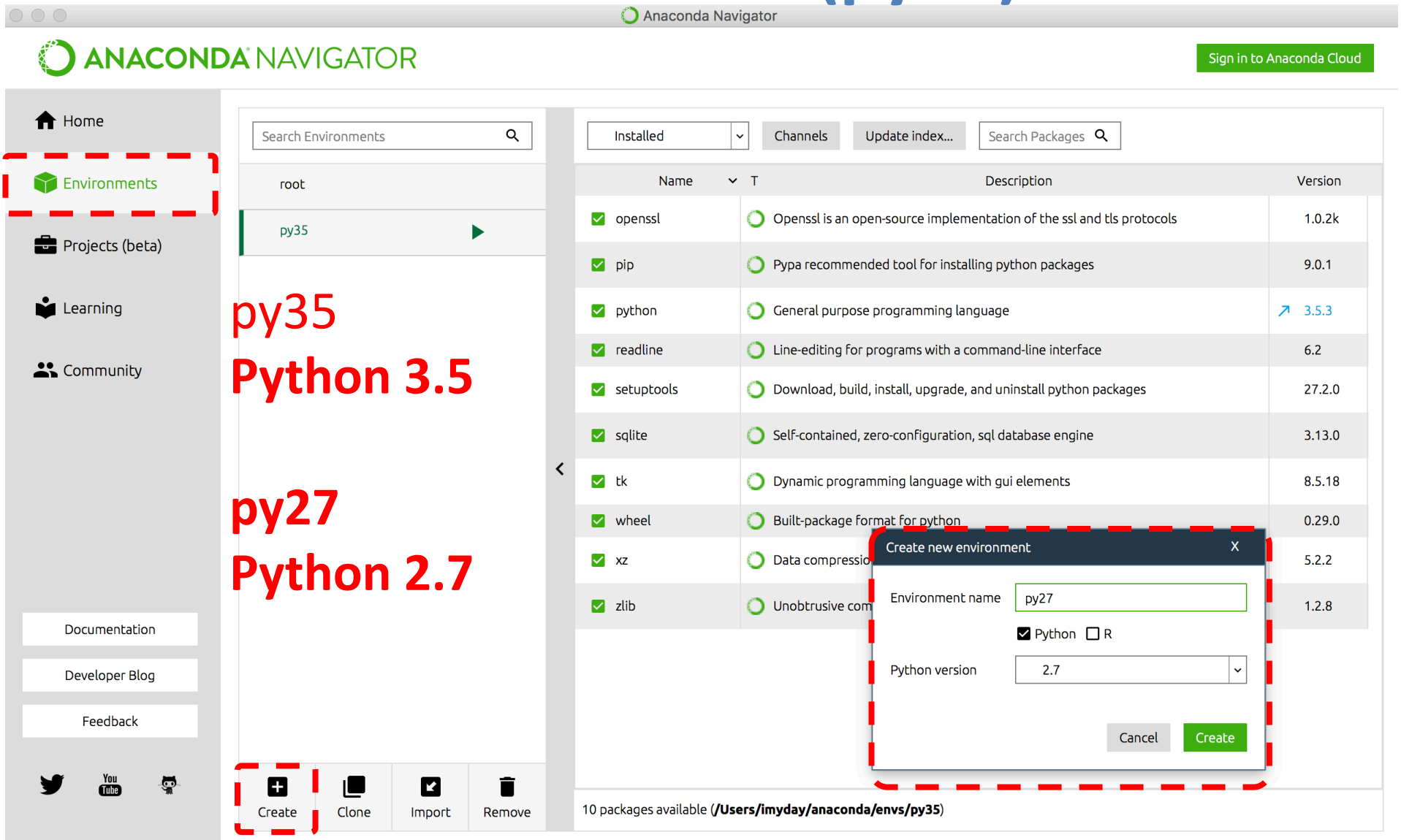
https://conda.io/docs/py2or3.html

Anaconda Create New Python 3.5 Environment (py35)



Source: http://conda.pydata.org/docs/py2or3.html

Anaconda Create New Python 2.7 Environment (py27)



Source: http://conda.pydata.org/docs/py2or3.html

Verify that conda is installed, check current conda version

- **`conda --version`**

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

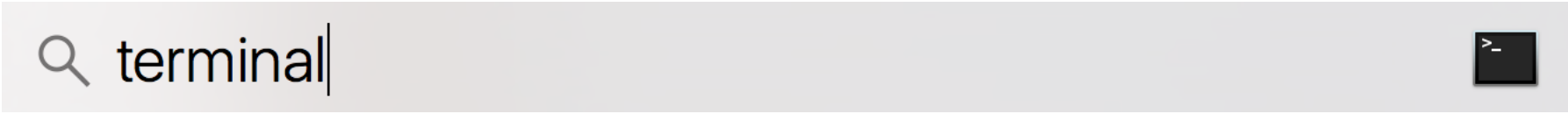
http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version

Check current conda version
Check current python version
Check conda environments

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

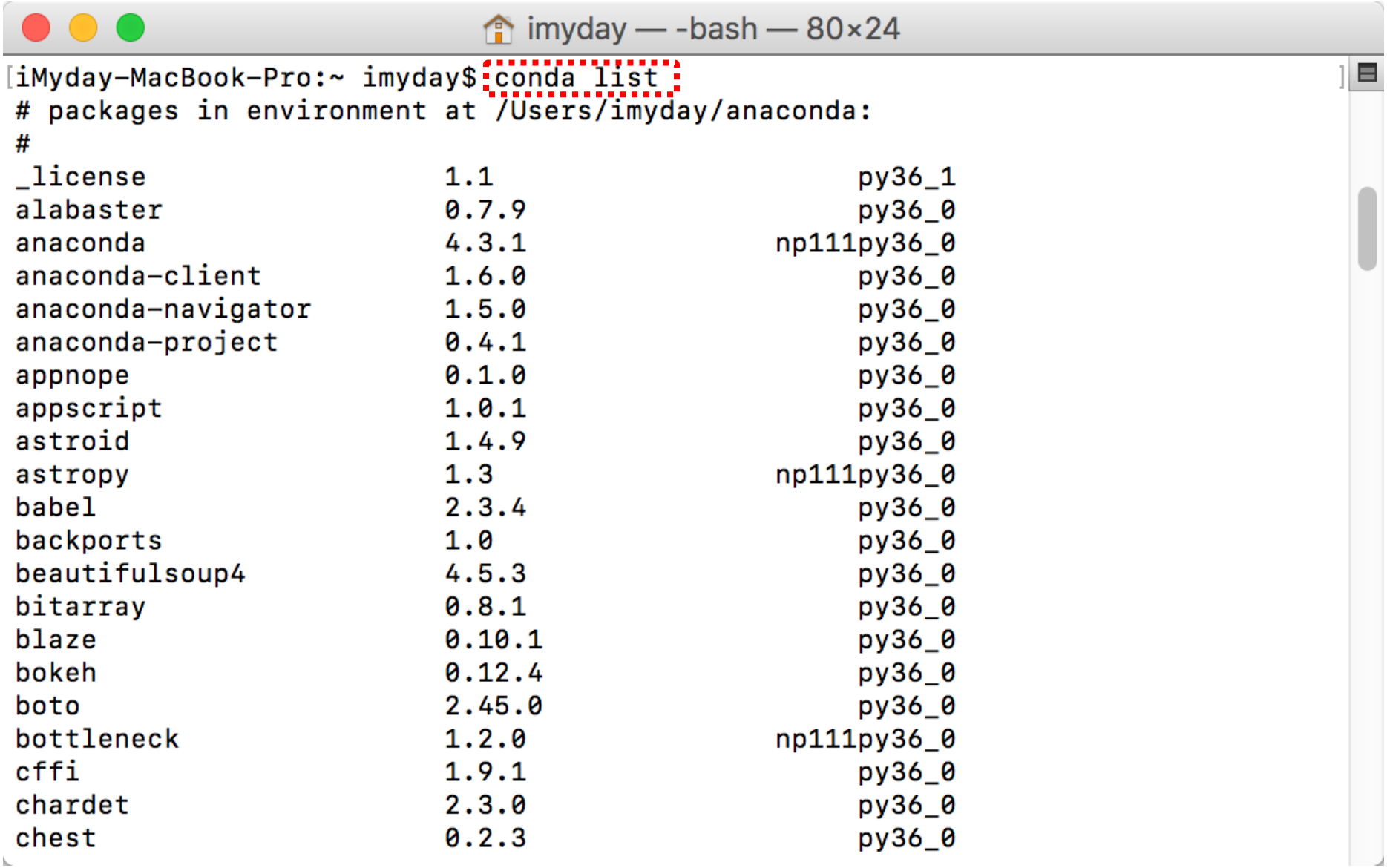
http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version

Terminal



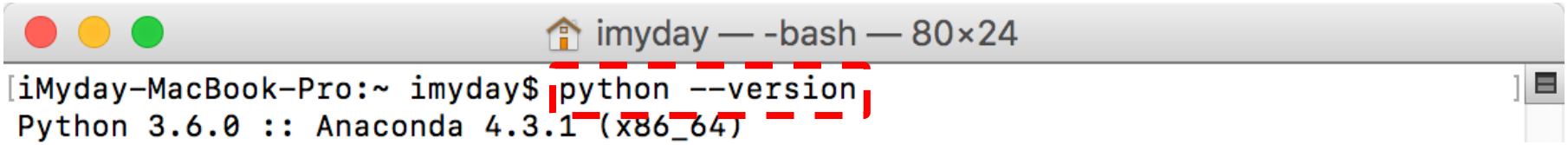

```
terminal
```

conda list

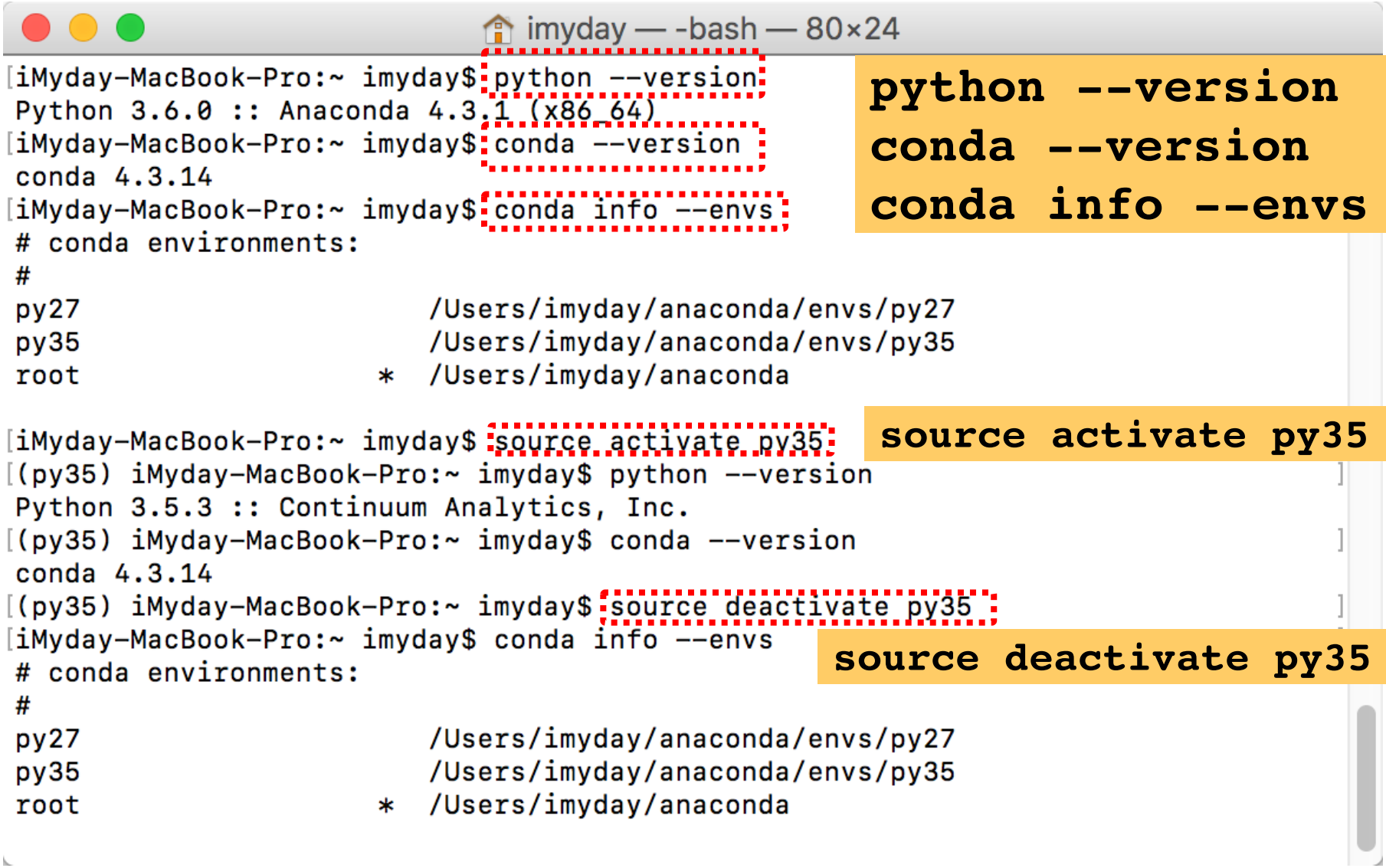


http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version

python --version



conda --version

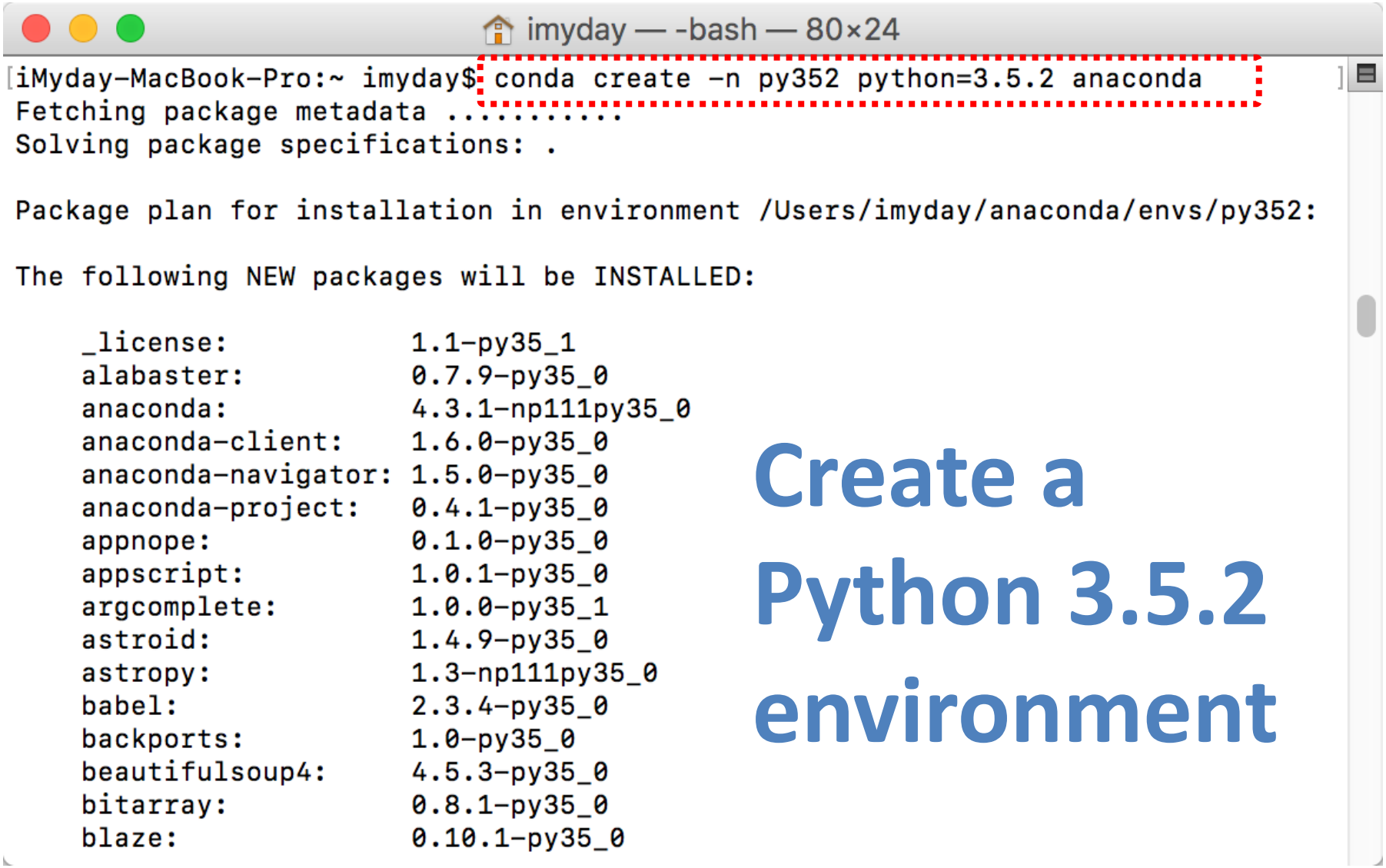


python --version
conda --version
conda info --envs

source activate py35

source deactivate py35

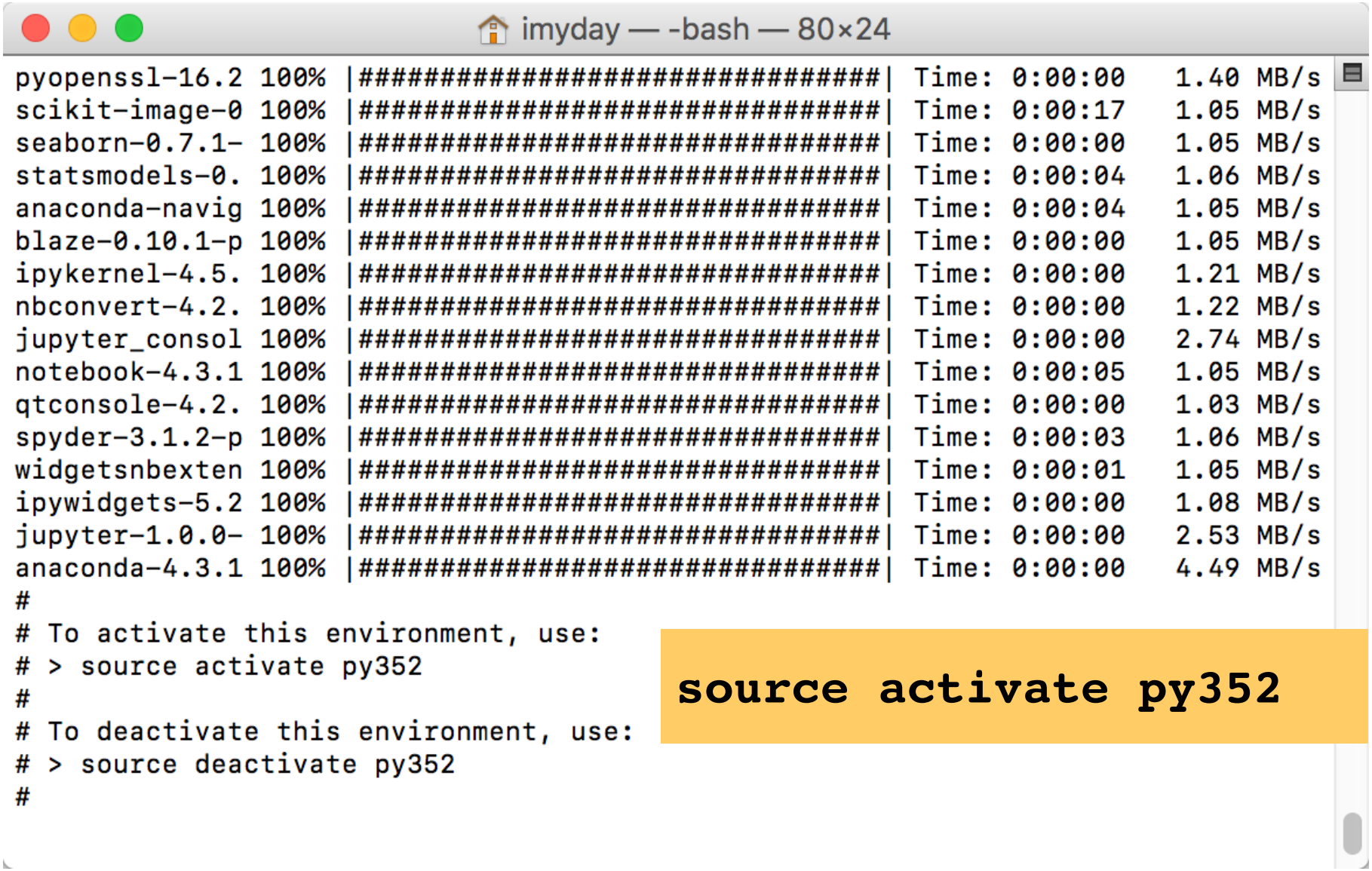
conda create -n py352 python=3.5.2 anaconda



Create a Python 3.5.2 environment

Source: http://conda.pydata.org/docs/py2or3.html

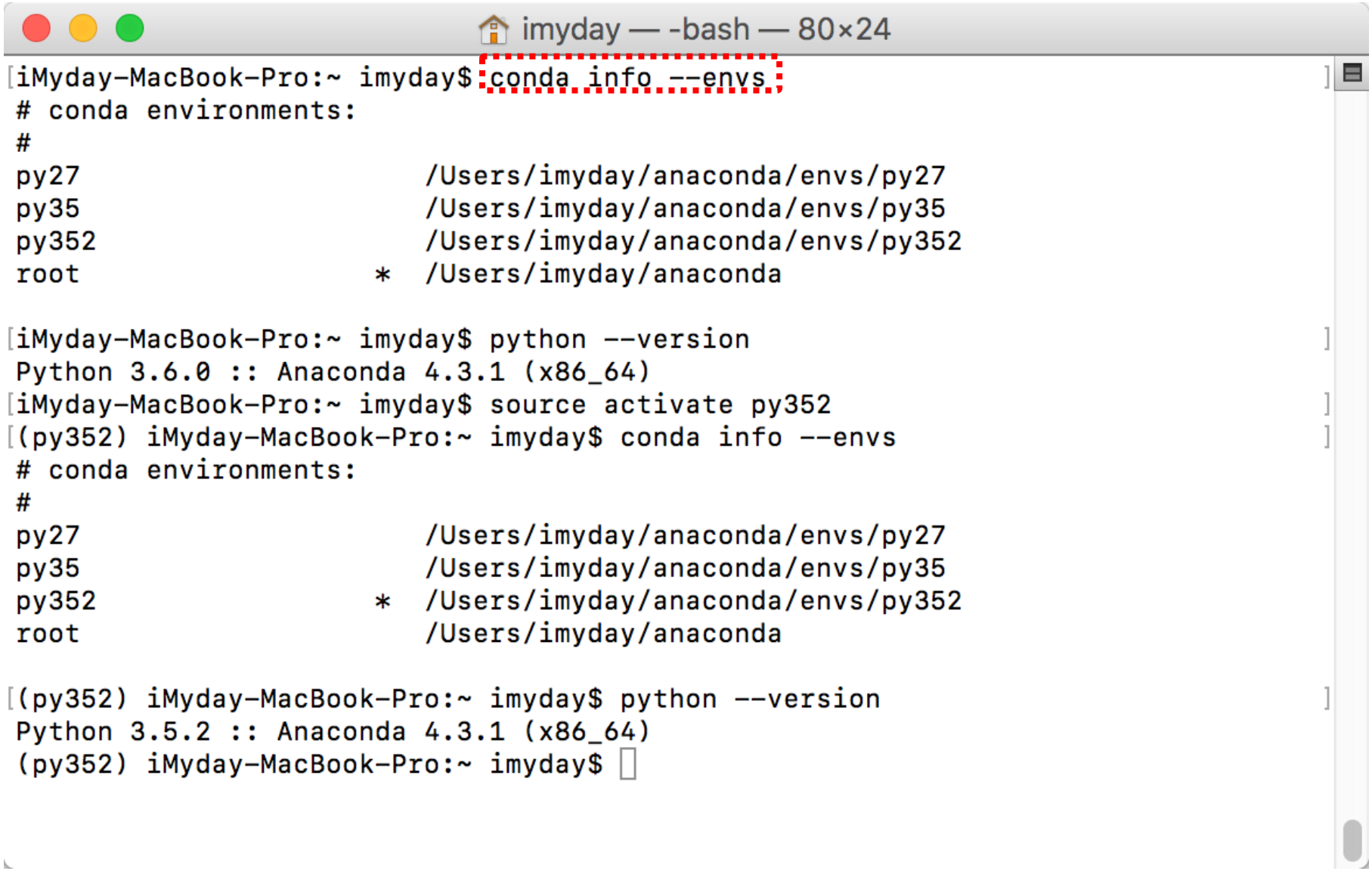
conda create -n py352 python=3.5.2 anaconda



source activate py352

Source: http://conda.pydata.org/docs/py2or3.html

conda info --envs



TensorFlow

```
conda info --envs
```

```
conda --version
python --version
```

```
conda list
```

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

```
source activate tensorflow
```

```
activate tensorflow
```

Source: http://deeplearning.net/software/theano/install.html#anaconda

```
sudo pip install tensorflow
```

```
pip install tensorflow
```

```
sudo pip install keras
```

```
pip install keras
```

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

```
jupyter notebook
```

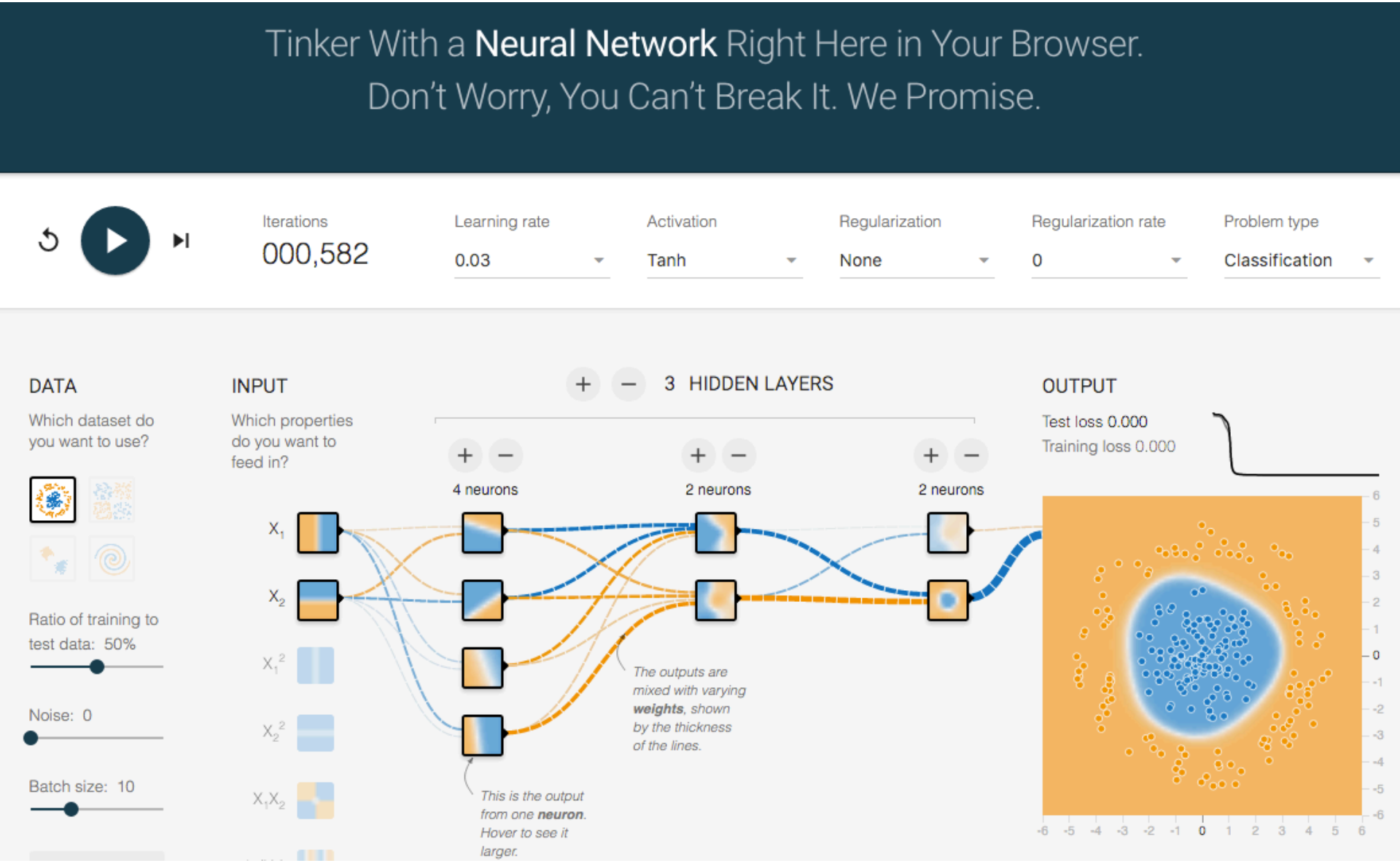
Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt

Source: http://deeplearning.net/software/theano/install.html#anaconda

pip install tensorflow

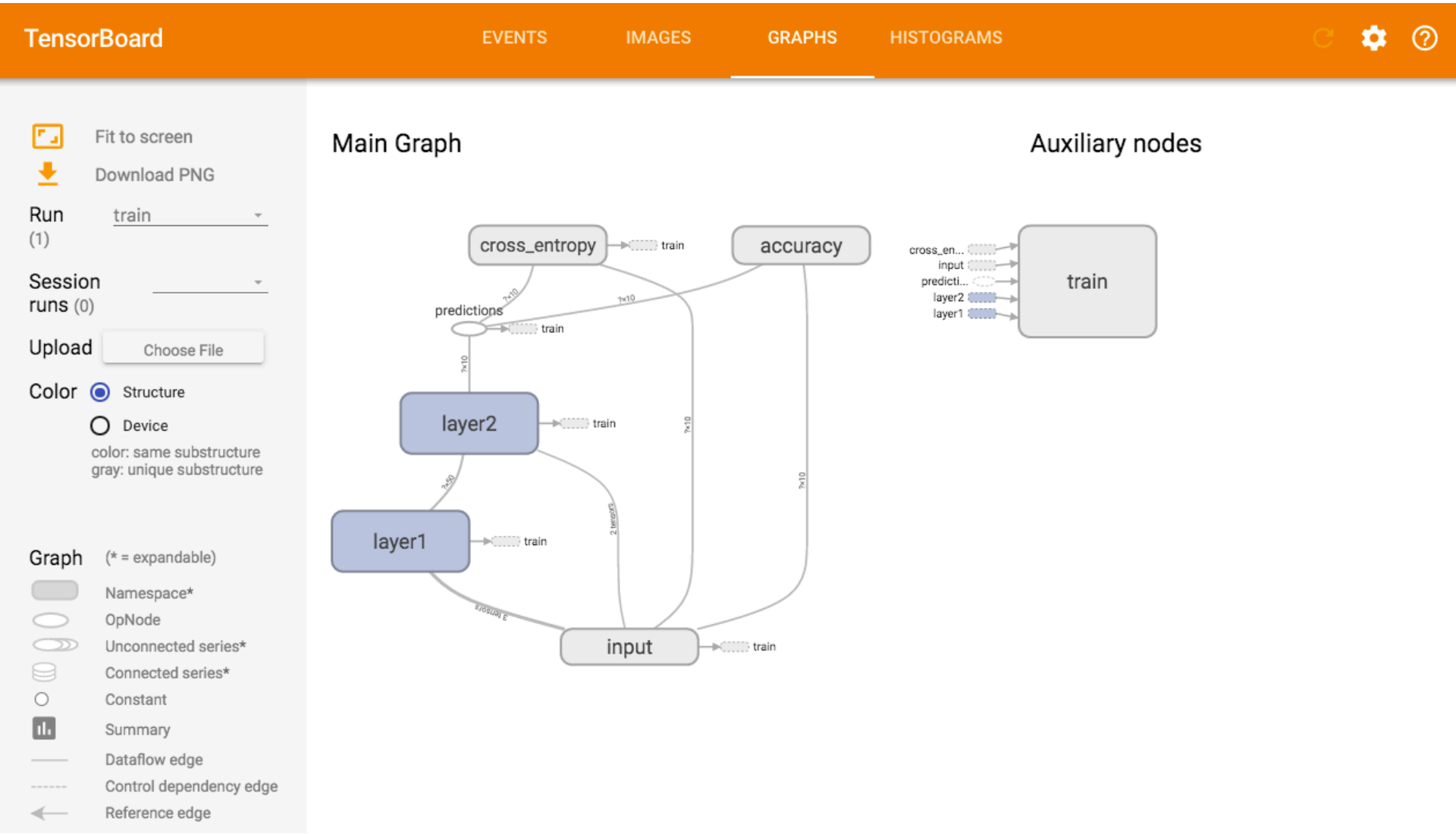
```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement 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



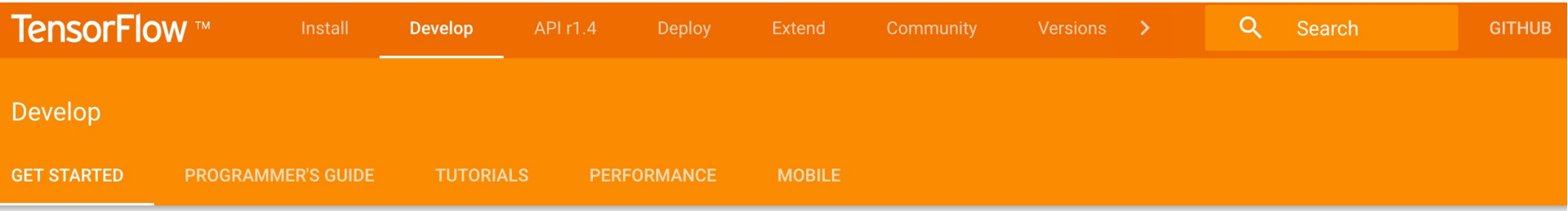
http://playground.tensorflow.org/

TensorBoard



https://www.tensorflow.org/tensorboard/index.html#graphs

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

Source: https://www.tensorflow.org/get_started/get_started

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
>>>
```

https://github.com/tensorflow/tensorflow

Hello TensorFlow

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

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

https://github.com/tensorflow/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
```

https://github.com/tensorflow/tensorflow

Linear Regression Model

```
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-.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
```

Source: https://www.tensorflow.org/get_started/get_started

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}
```

Source: https://www.tensorflow.org/get_started/get_started

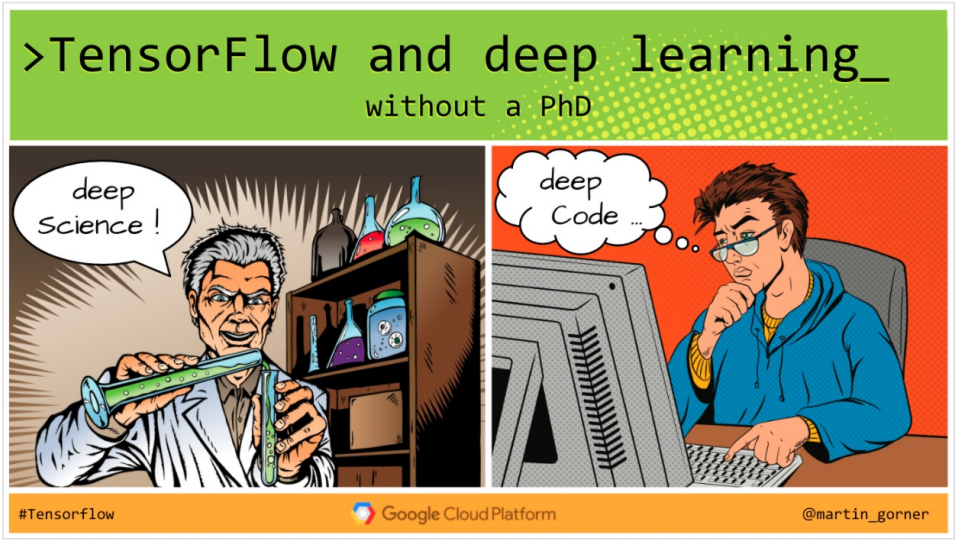
TensorFlow and Deep Learning

TensorFlow and Deep Learning

Did you find a mistake? Please file a bug.

TensorFlow and deep learning, without a PhD — 149 min remaining

1. Overview



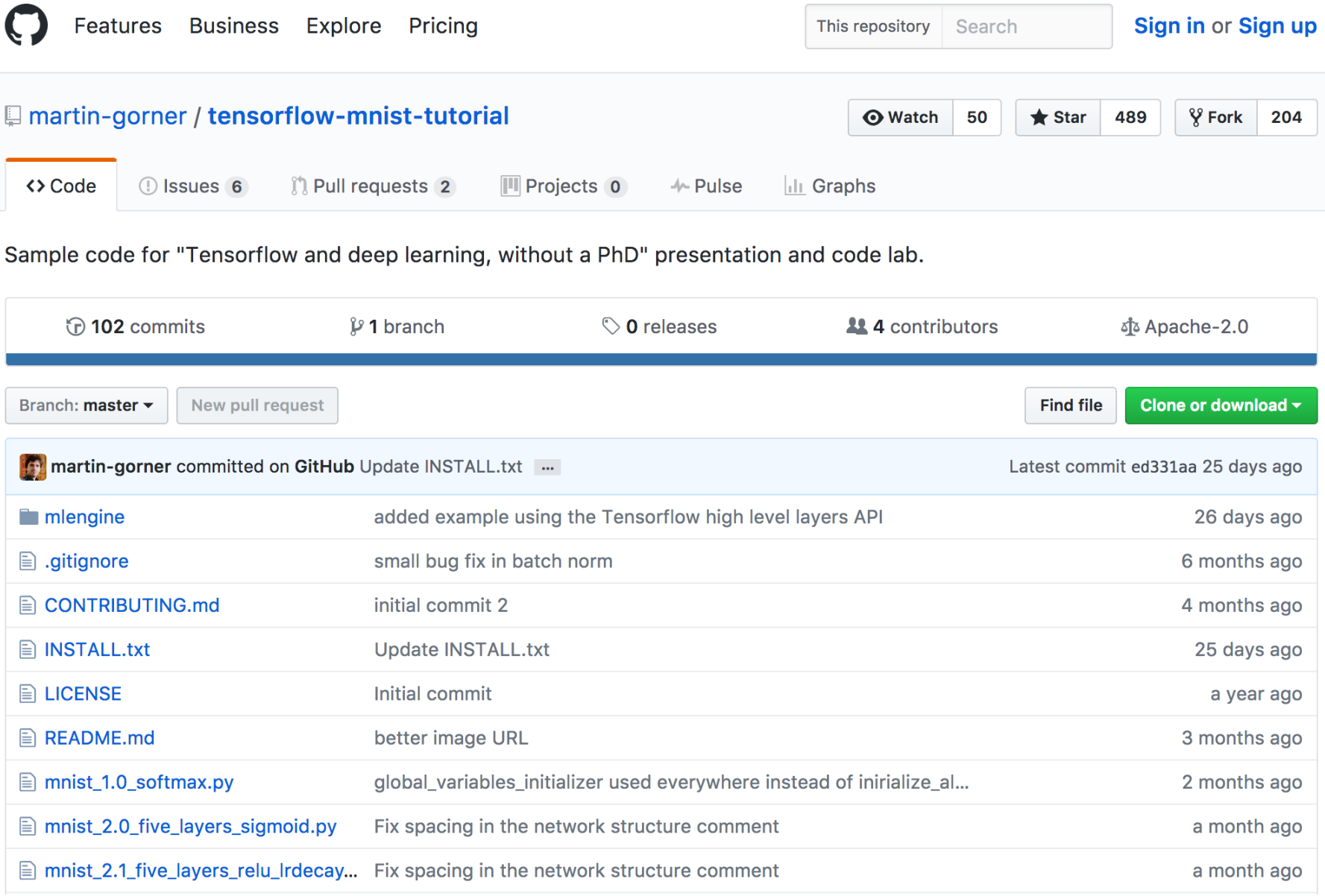
In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

TensorFlow MNIST Tutorial



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

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

TensorFlow MNIST Tutorial

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

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

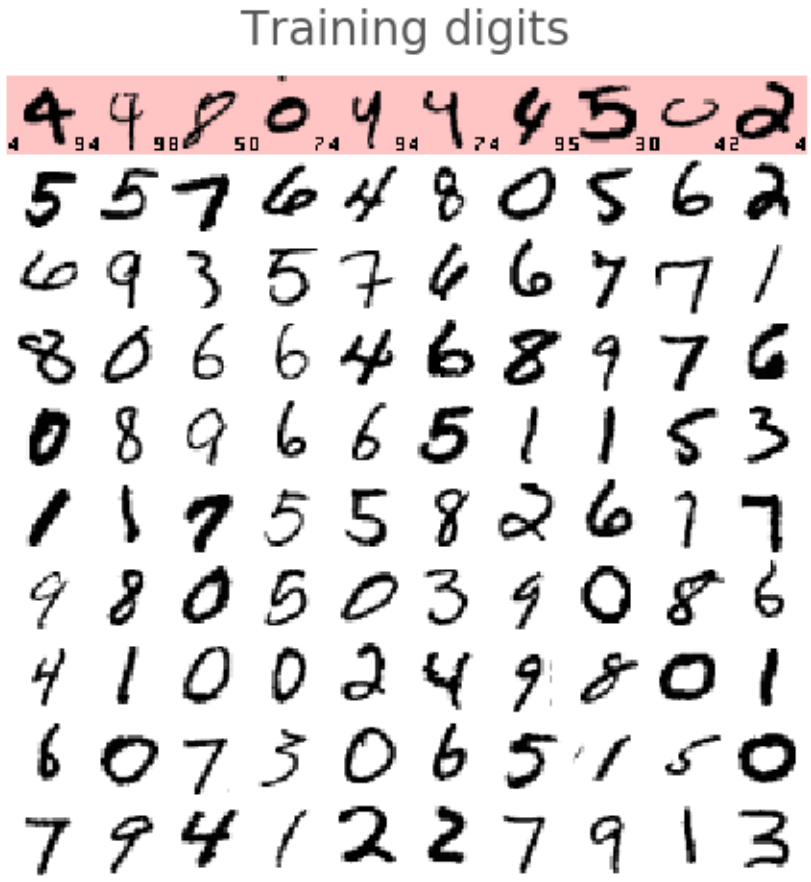
```
python3 mnist_1.0_softmax.py
```

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

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

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt

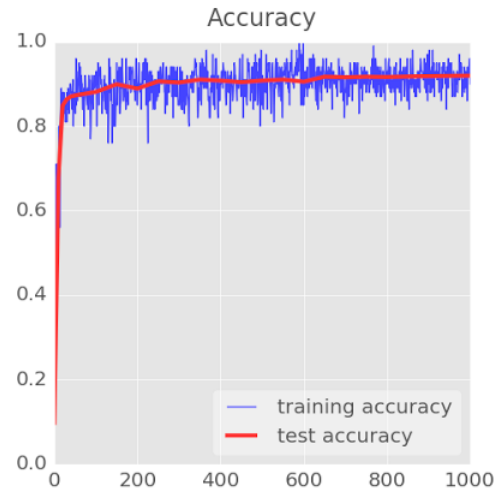
MNIST dataset: 60,000 labeled digits

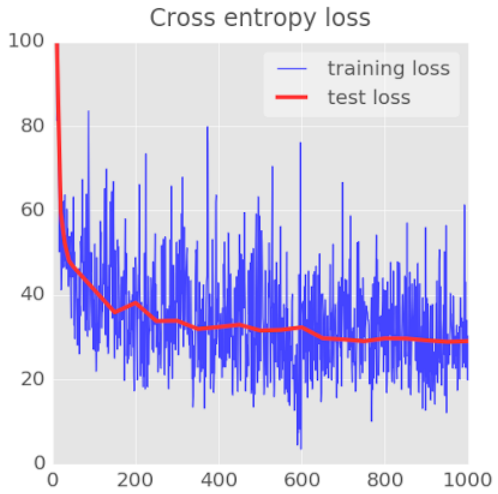


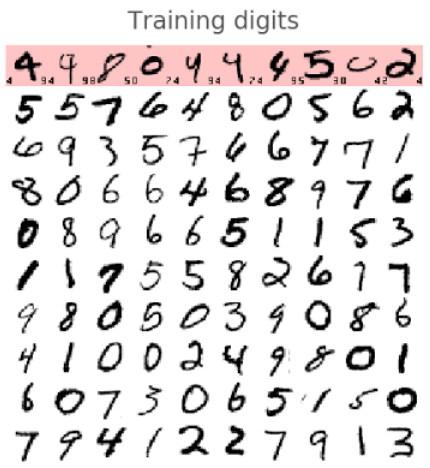
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

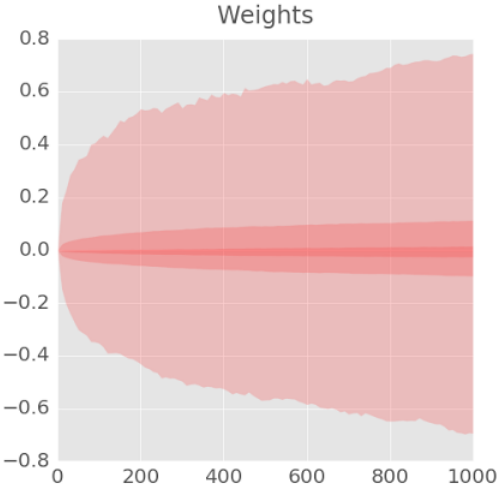
```
cd tensorflow-mnist-tutorial
```

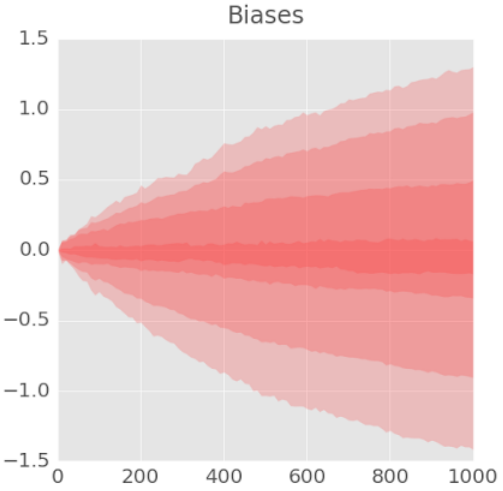
```
python3 mnist_1.0_softmax.py
```

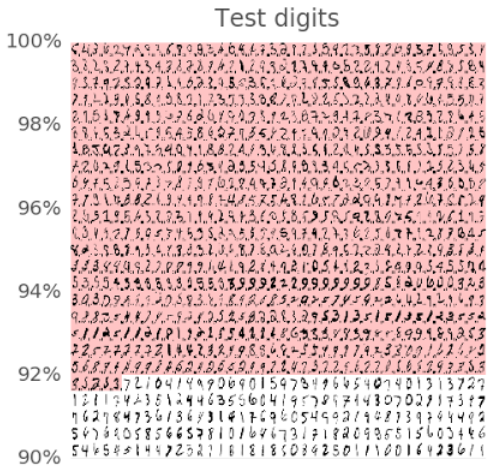












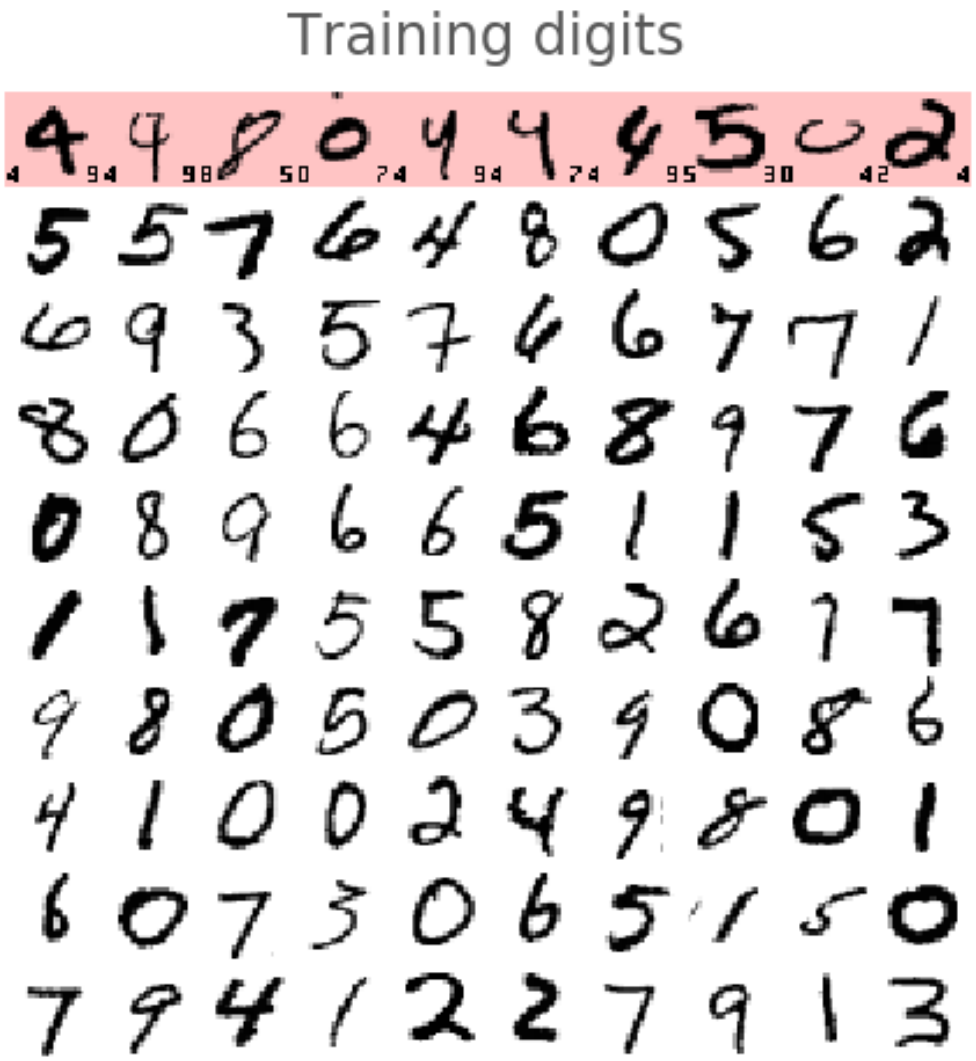
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#1

Train a Neural Network

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

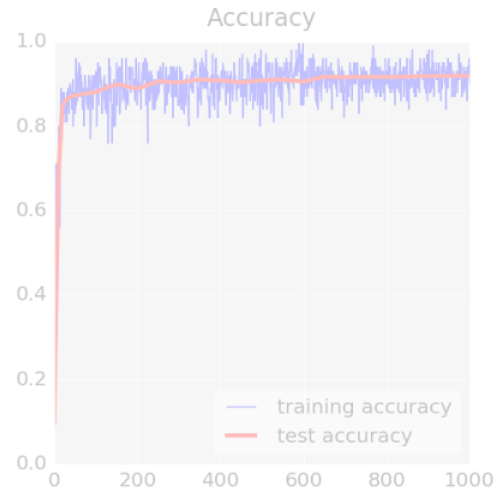
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

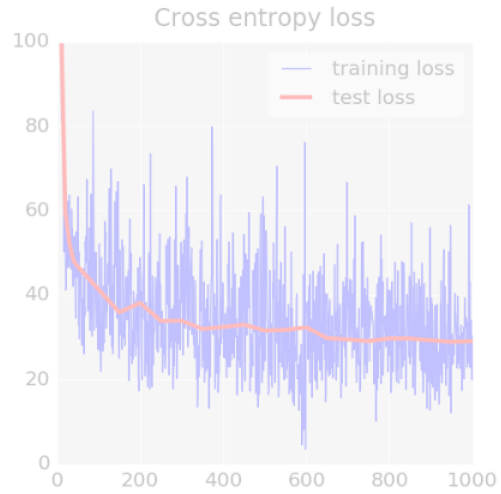
Training digits

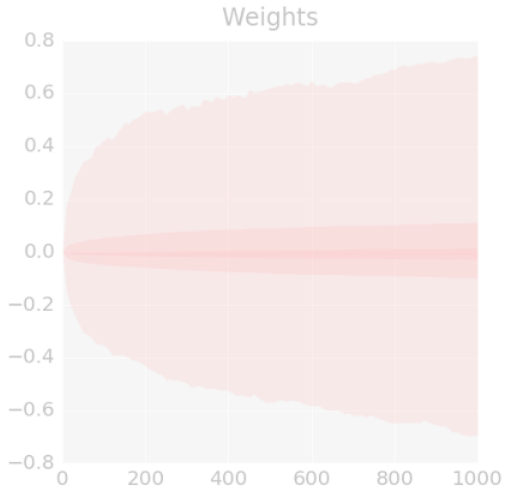


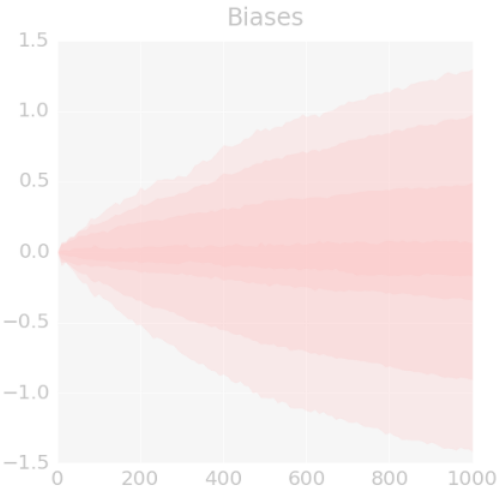
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Training digits

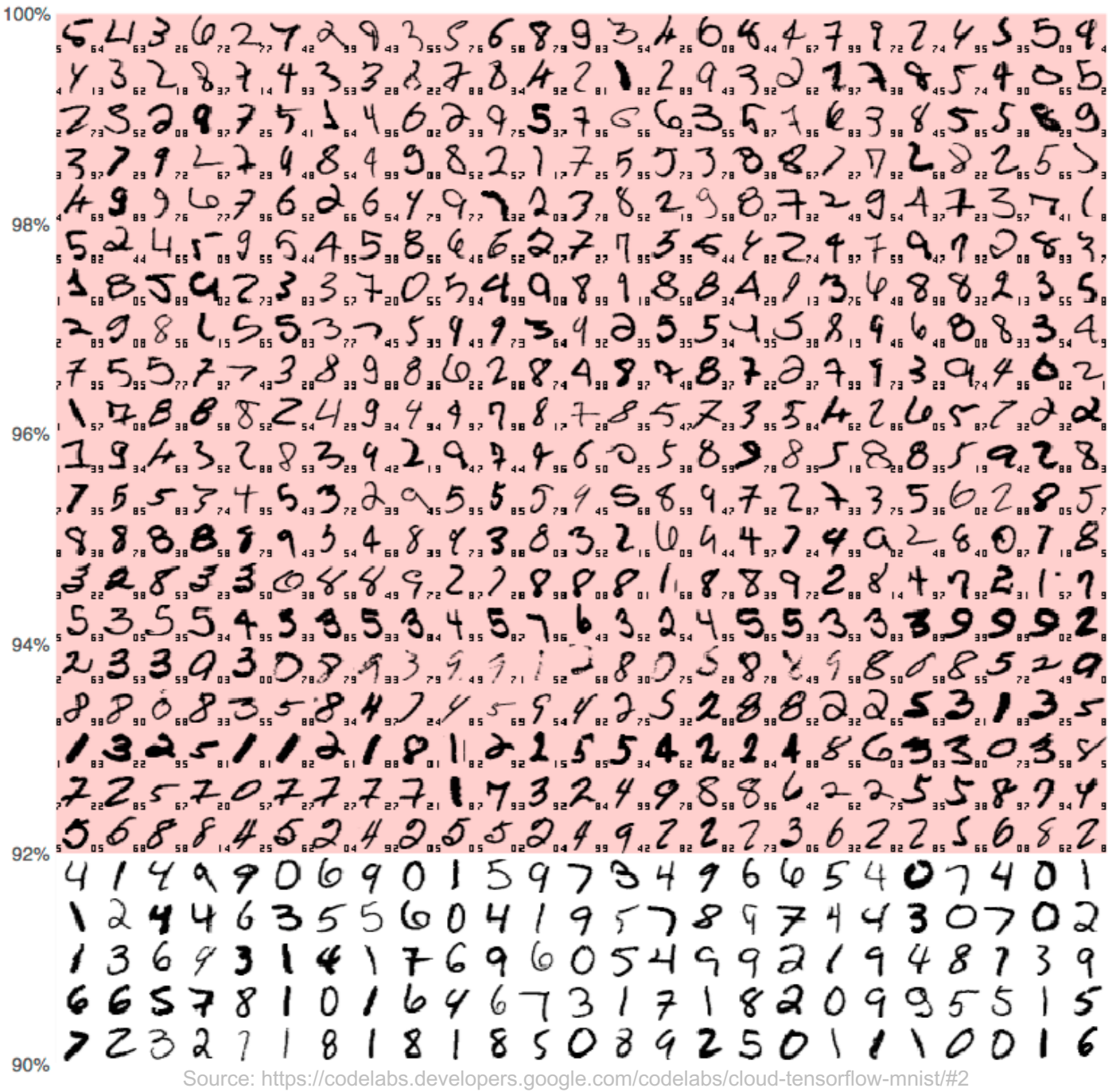






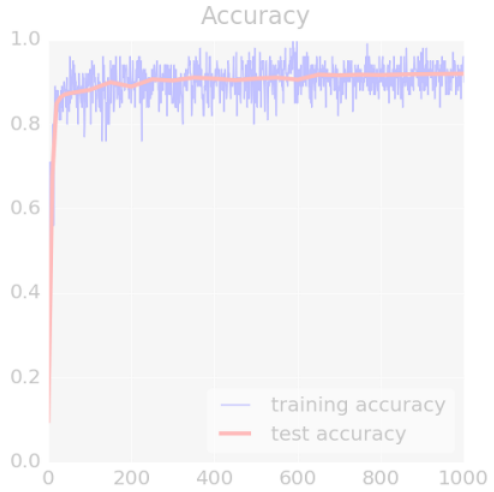


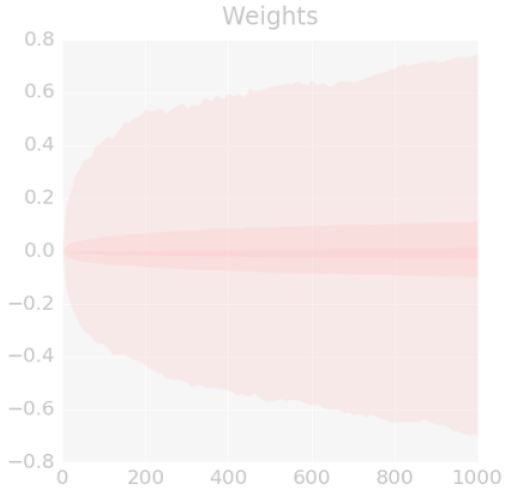
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

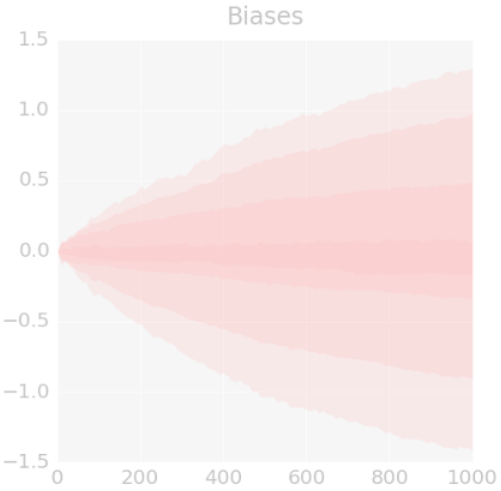


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Test digits

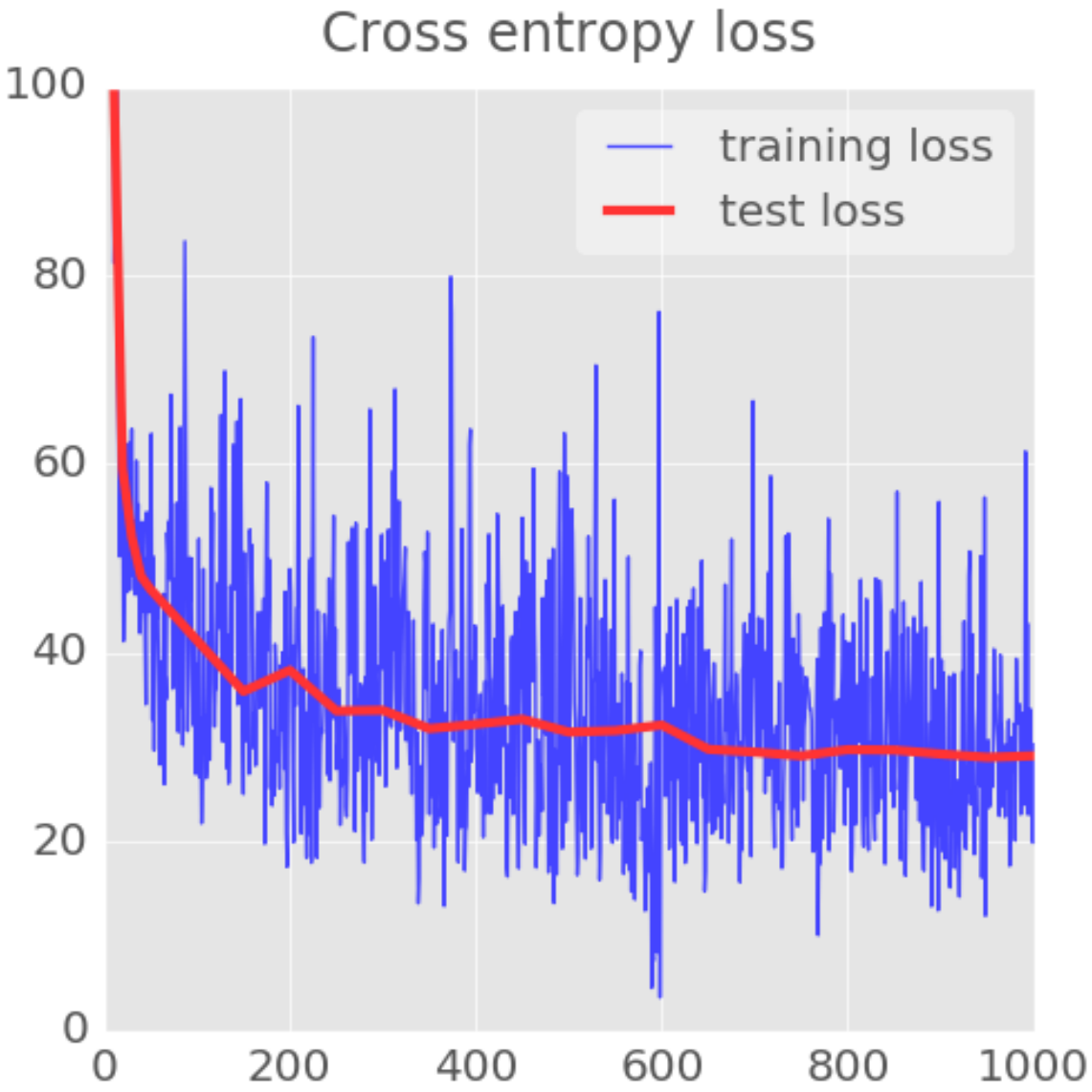






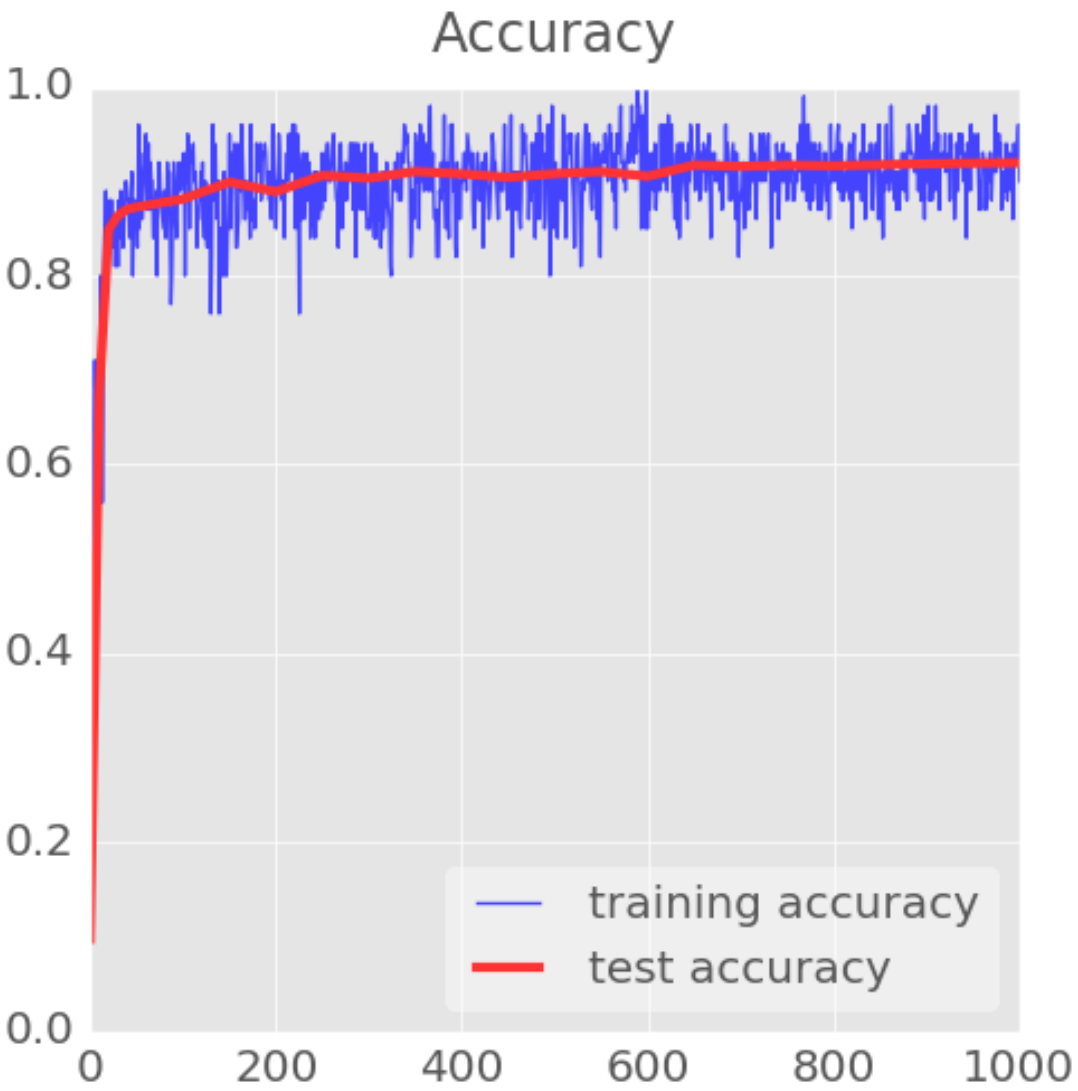
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Cross entropy loss



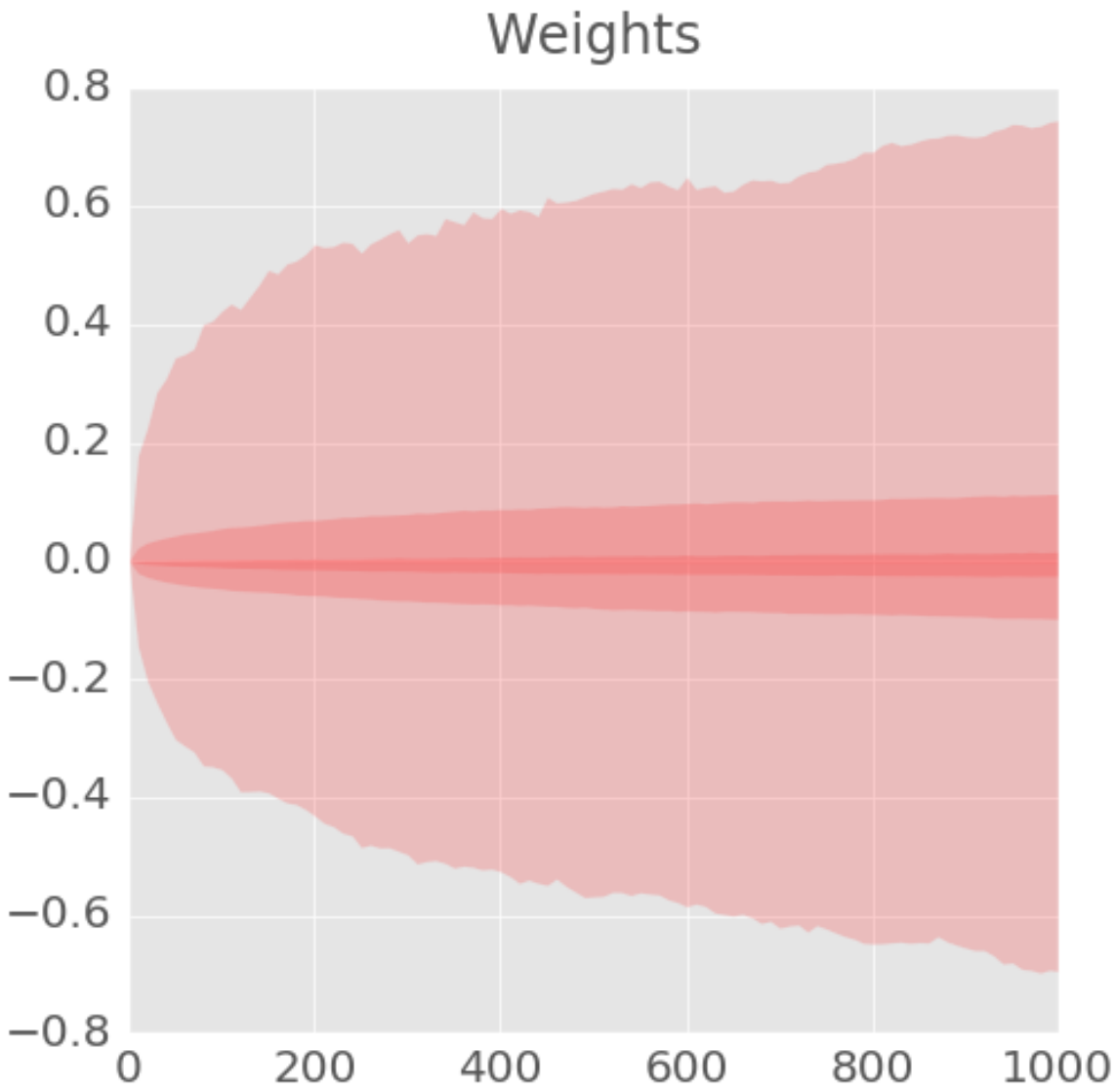
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Accuracy



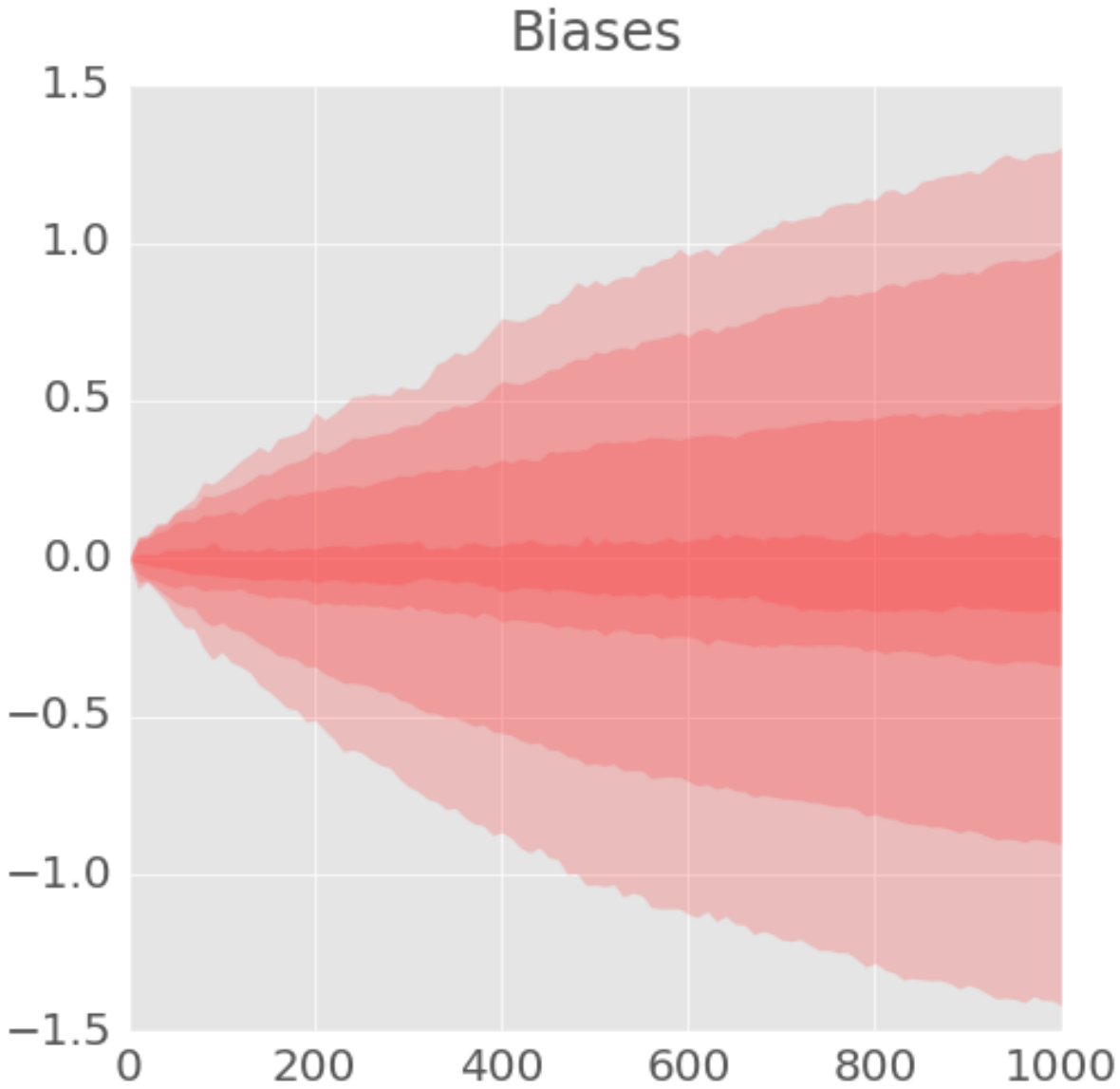
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Weights



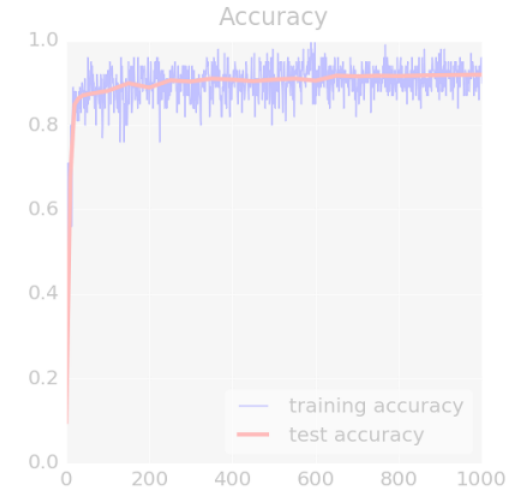
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

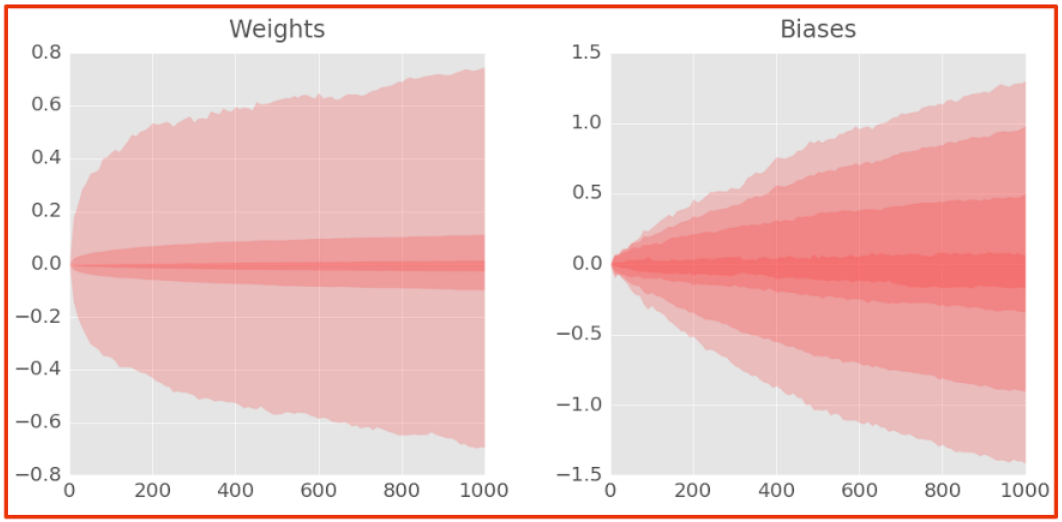
Biases

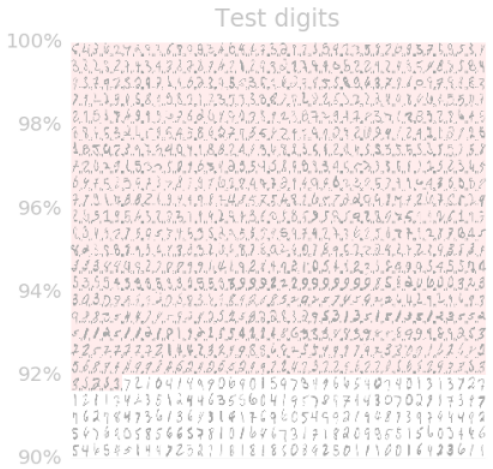


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Weights and Biases





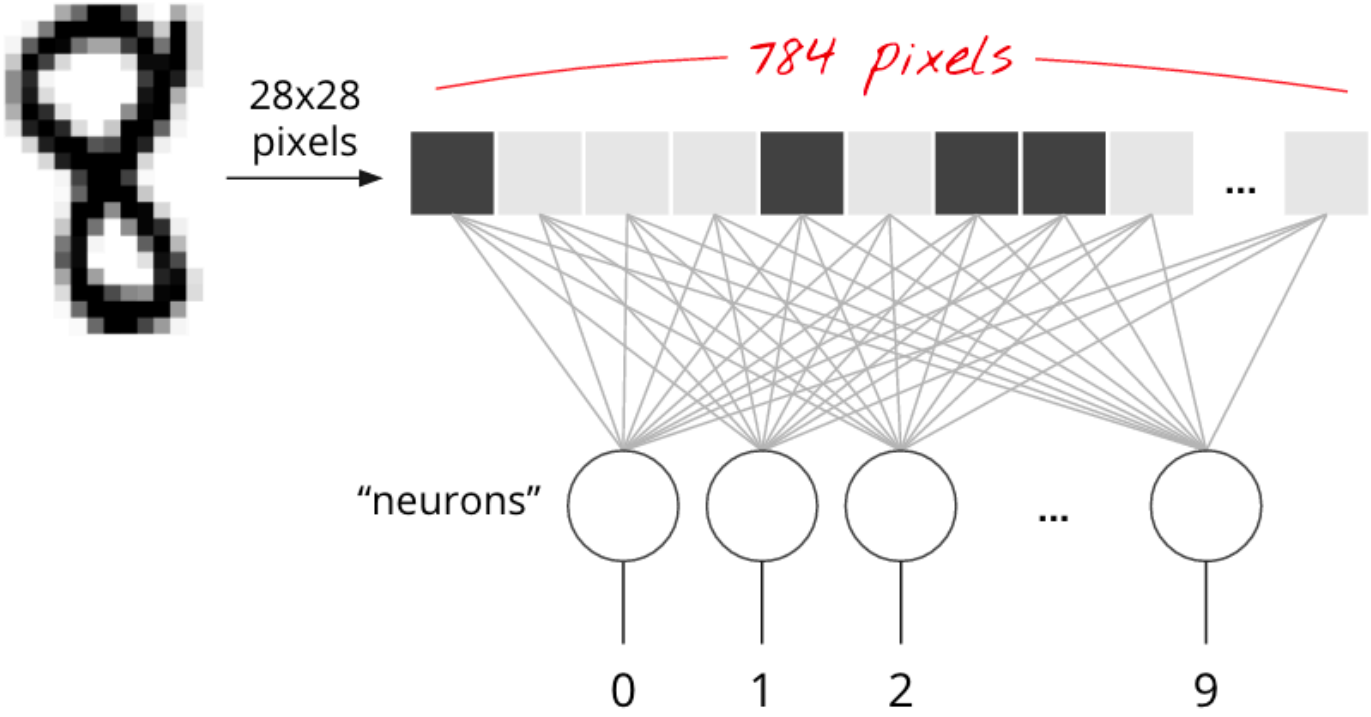


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Cookbook

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Very Simple Model: Softmax Classification

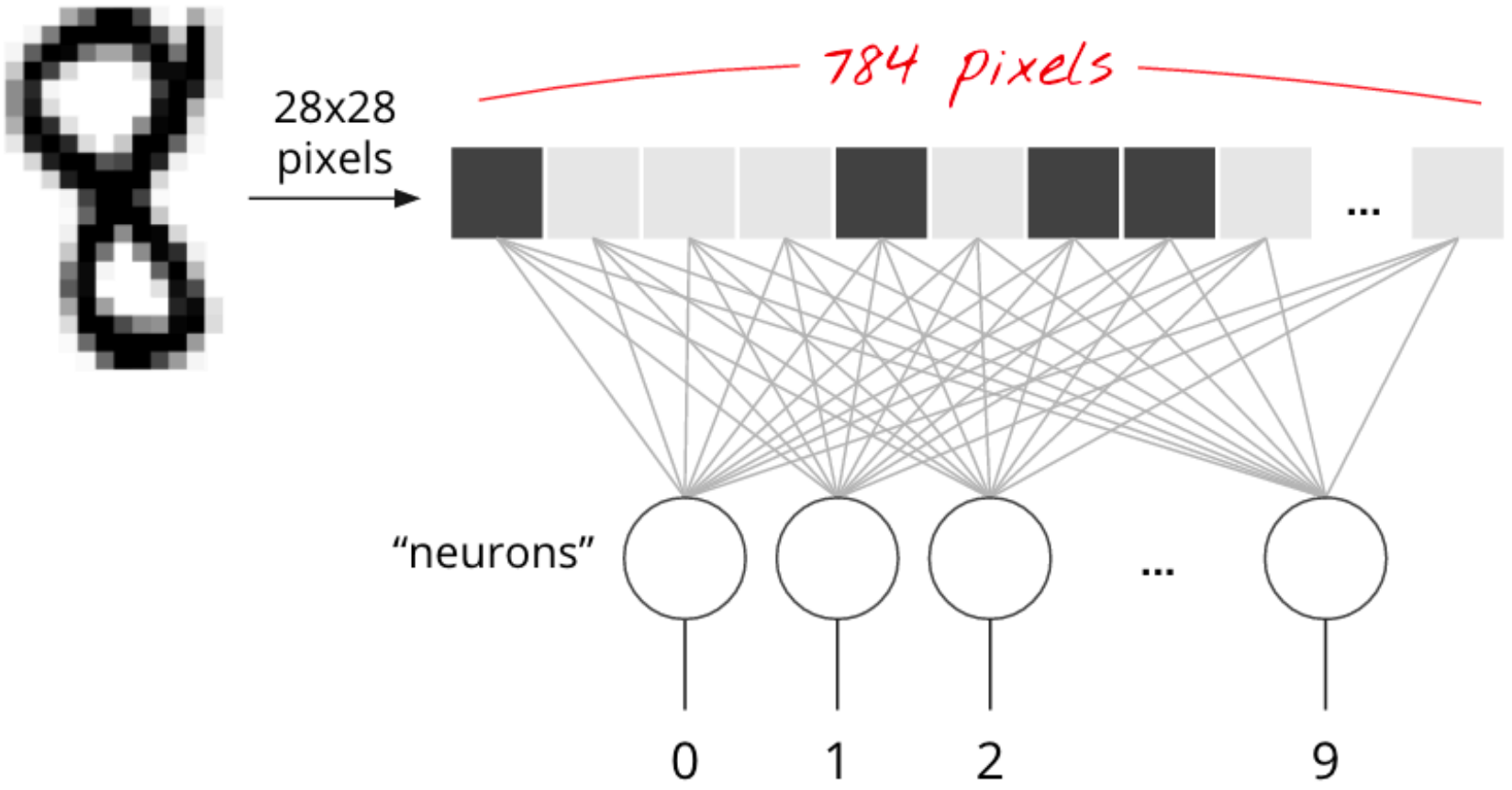


weighted sum of all pixels + bias

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

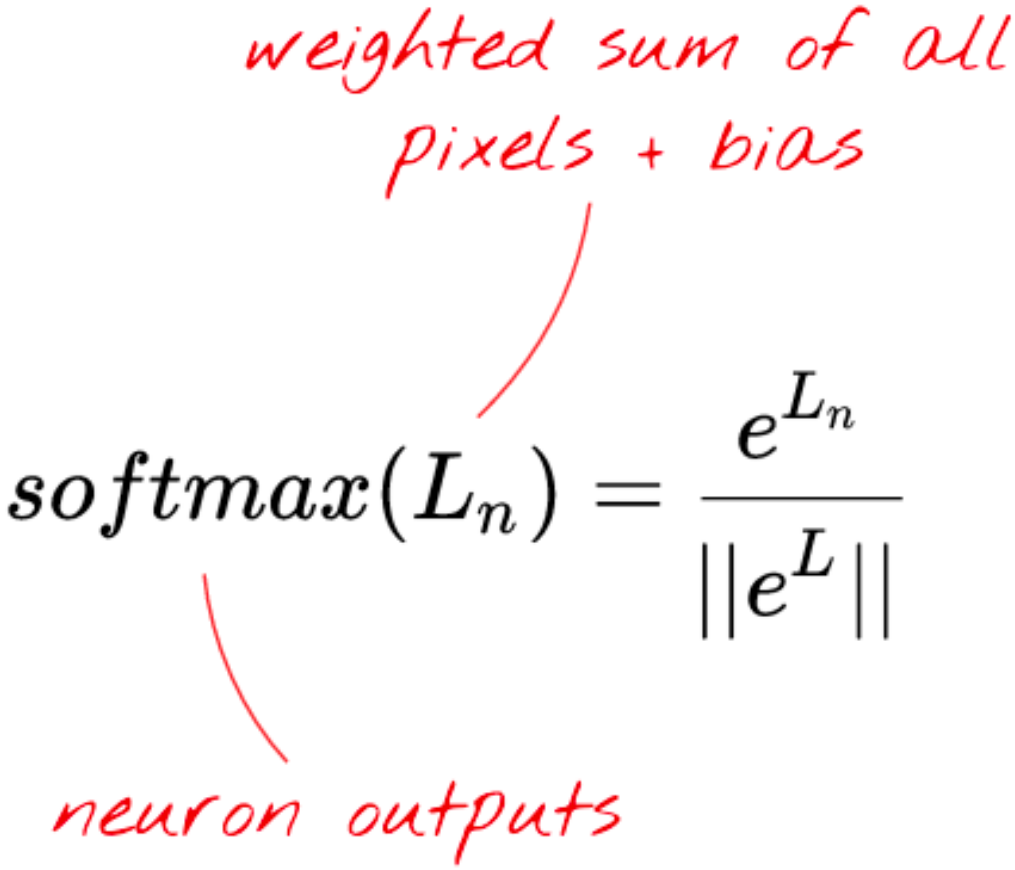
neuron outputs

Very Simple Model: Softmax Classification



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Very Simple Model: Softmax Classification



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

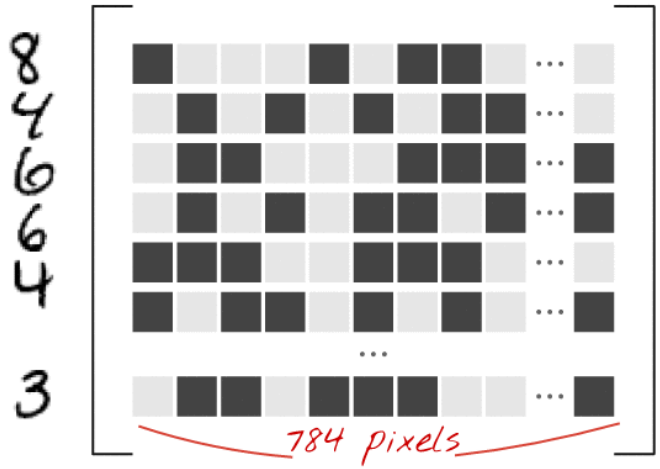
In Matrix notation, 100 images at a time

$$\begin{bmatrix} W_{0,0} & W_{0,1} & W_{0,2} & W_{0,3} & \cdots & W_{0,9} \\ W_{1,0} & W_{1,1} & W_{1,2} & W_{1,3} & \cdots & W_{1,9} \\ W_{2,0} & W_{2,1} & W_{2,2} & W_{2,3} & \cdots & W_{2,9} \\ W_{3,0} & W_{3,1} & W_{3,2} & W_{3,3} & \cdots & W_{3,9} \\ W_{4,0} & W_{4,1} & W_{4,2} & W_{4,3} & \cdots & W_{4,9} \\ W_{5,0} & W_{5,1} & W_{5,2} & W_{5,3} & \cdots & W_{5,9} \\ W_{6,0} & W_{6,1} & W_{6,2} & W_{6,3} & \cdots & W_{6,9} \\ W_{7,0} & W_{7,1} & W_{7,2} & W_{7,3} & \cdots & W_{7,9} \\ W_{8,0} & W_{8,1} & W_{8,2} & W_{8,3} & \cdots & W_{8,9} \\ & & \cdots & & & \\ W_{783,0} & W_{783,1} & W_{783,2} & & \cdots & W_{783,9} \end{bmatrix}$$

10 columns

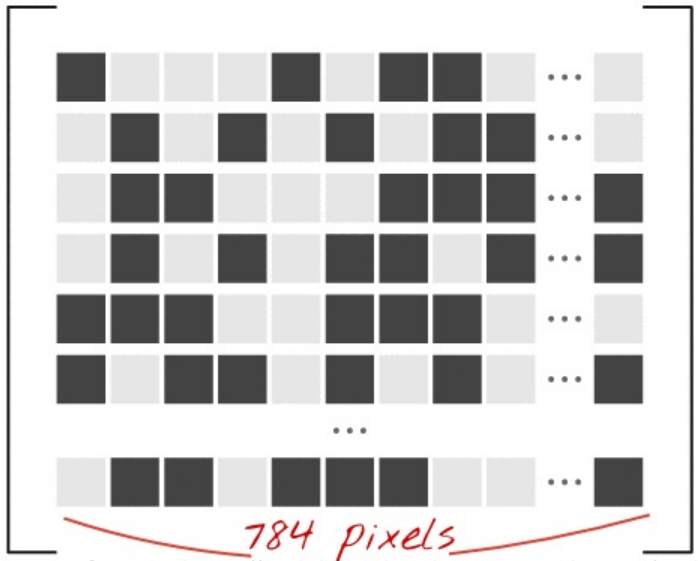
784 lines

X: 100 images, one per line, flattened



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

X: 100 images, one per line, flattened

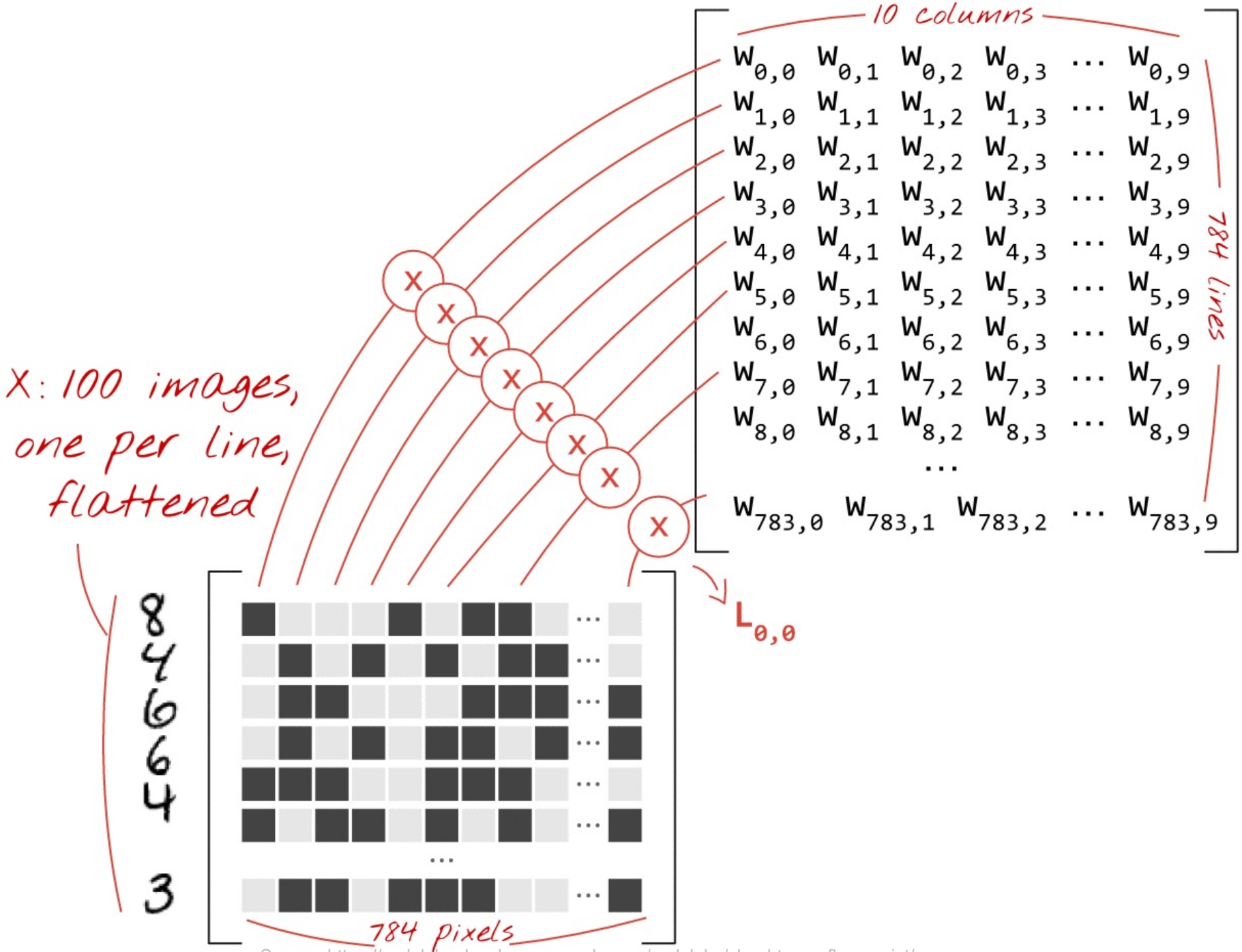


$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \dots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \dots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \dots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \dots & w_{3,9} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \dots & w_{4,9} \\ w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \dots & w_{5,9} \\ w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \dots & w_{6,9} \\ w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \dots & w_{7,9} \\ w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \dots & w_{8,9} \\ & & \dots & & & \\ w_{783,0} & w_{783,1} & w_{783,2} & & \dots & w_{783,9} \end{bmatrix}$$

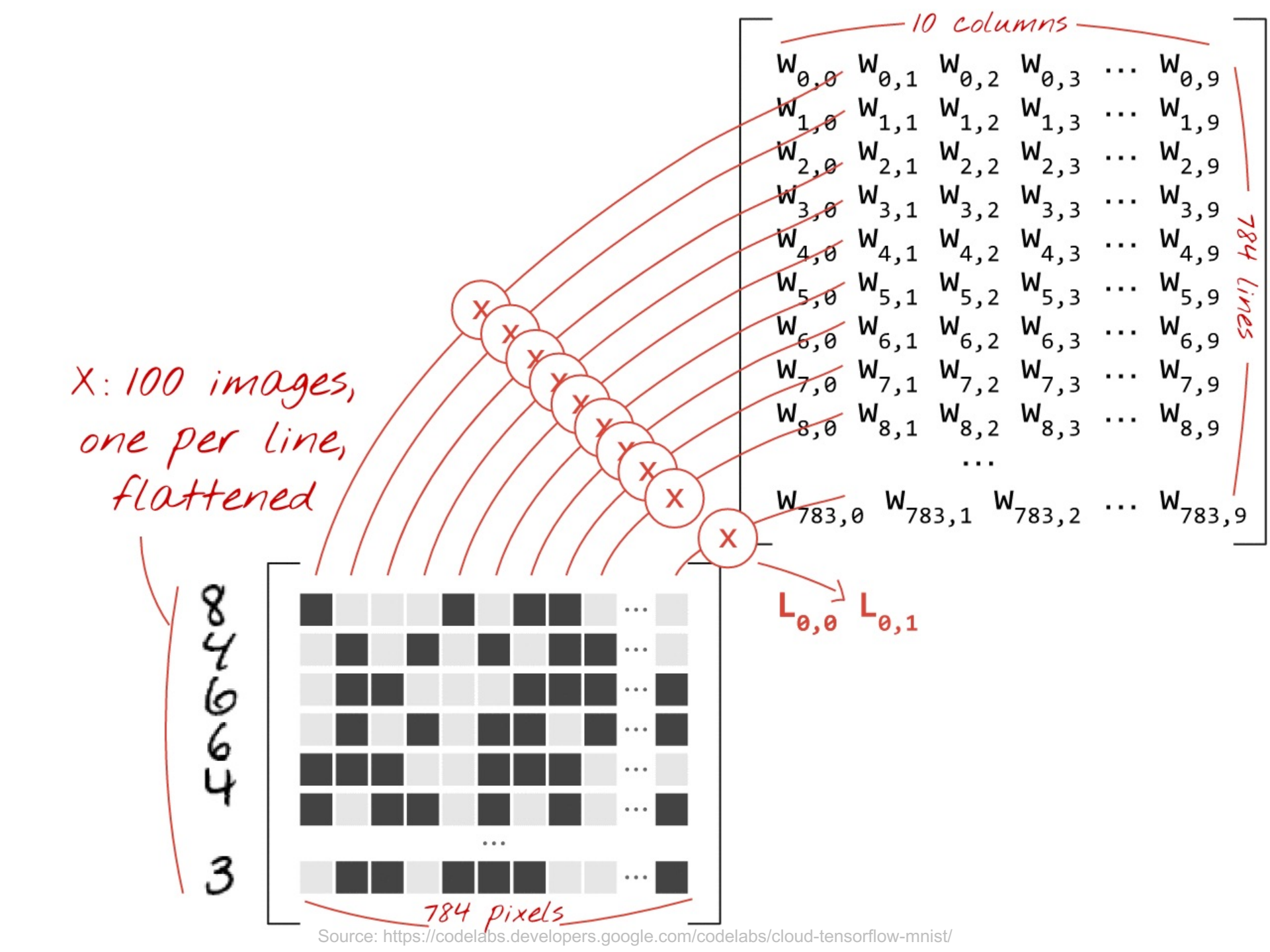
10 columns

784 lines

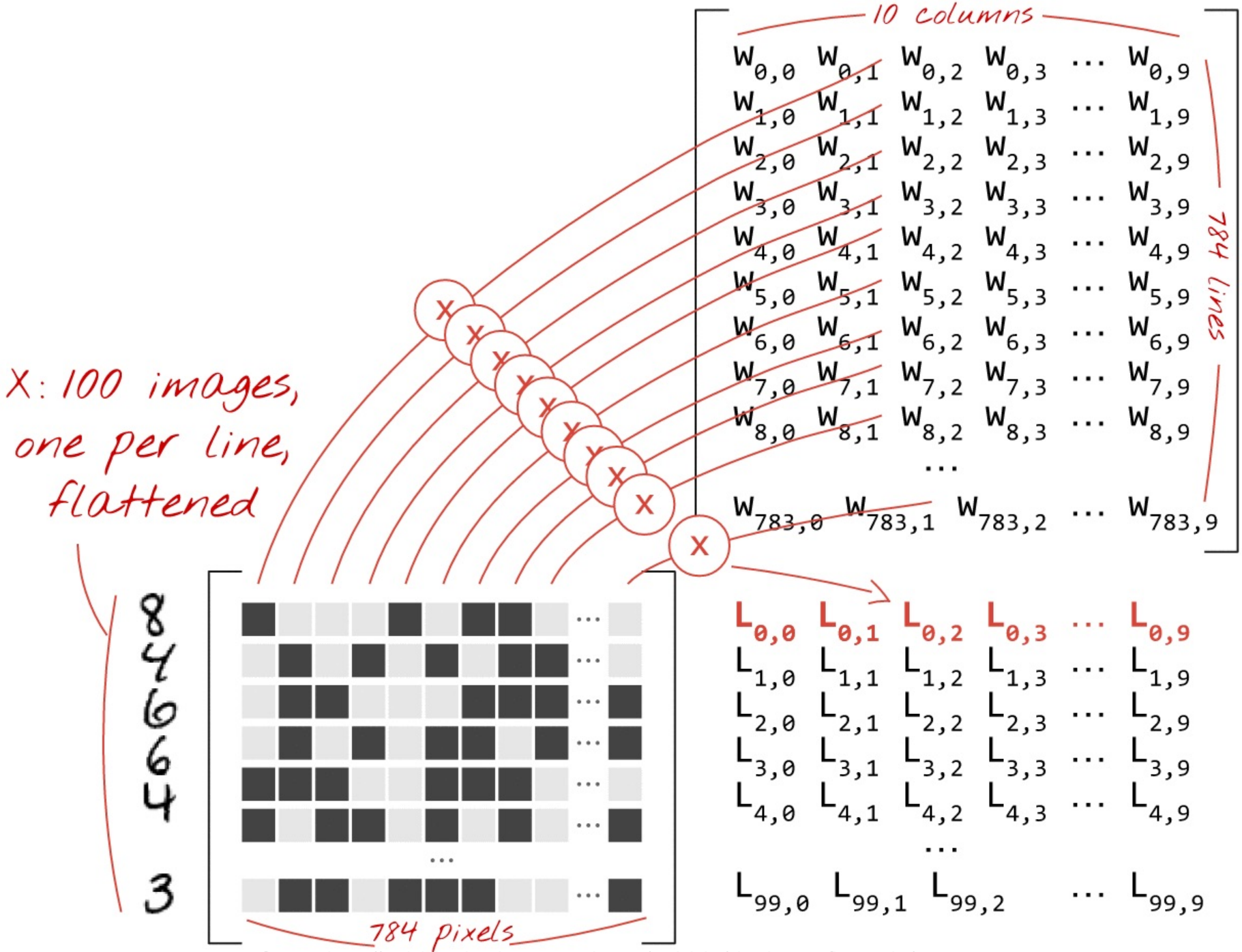
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

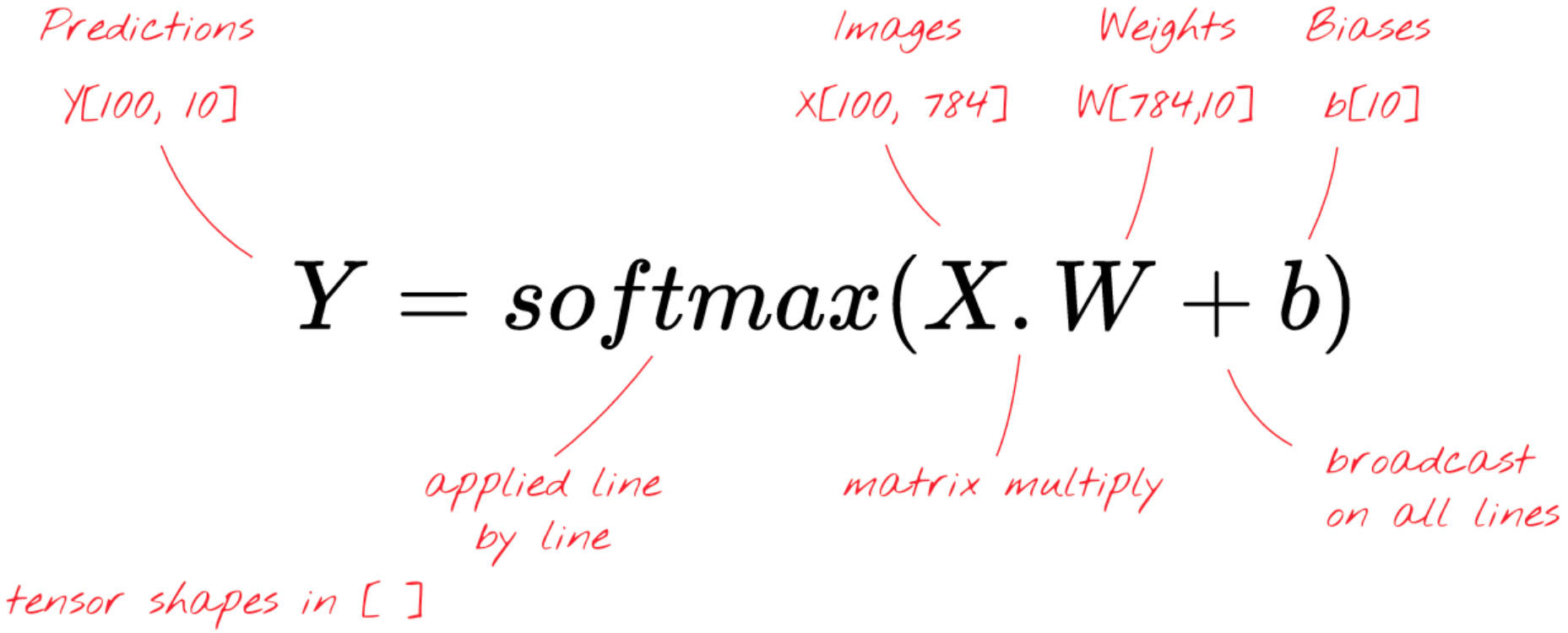
What are "weights" and "biases" ?

How is the "cross-entropy" computed ?

How exactly does the training algorithm work ?

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

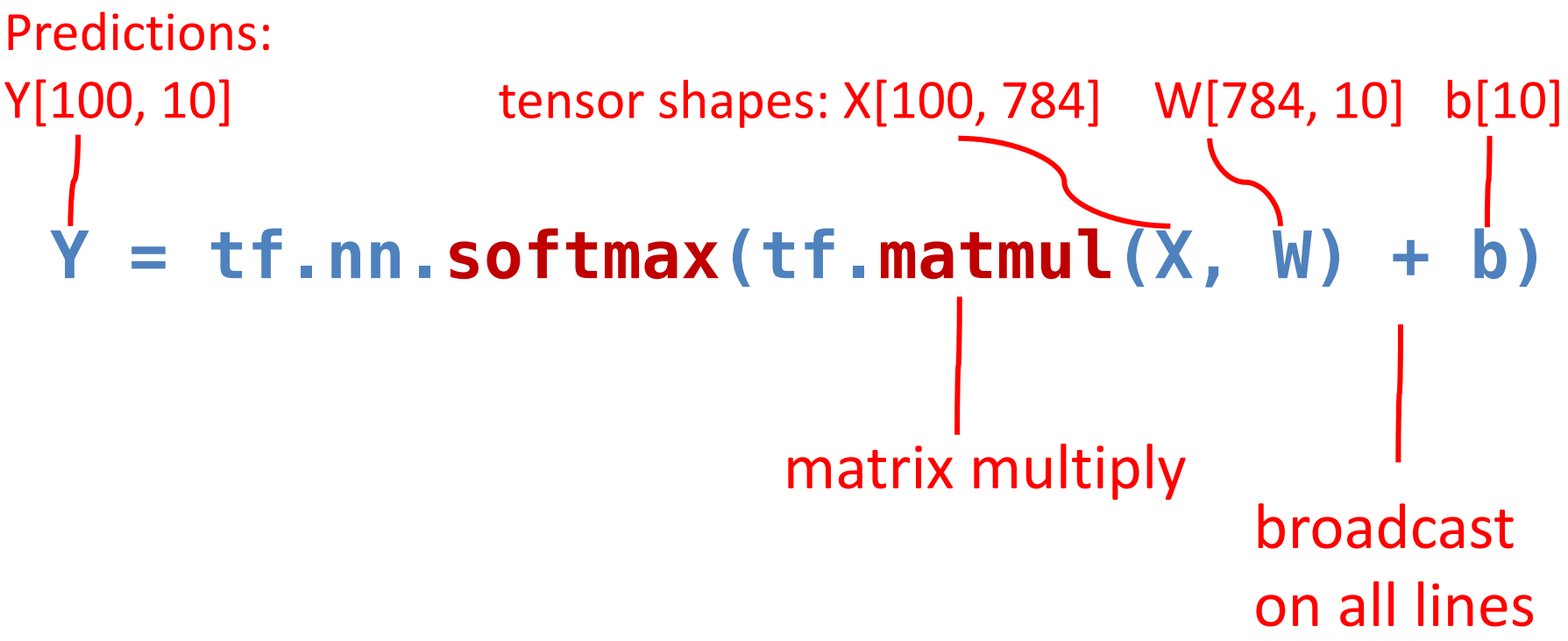
Y = f(X)




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

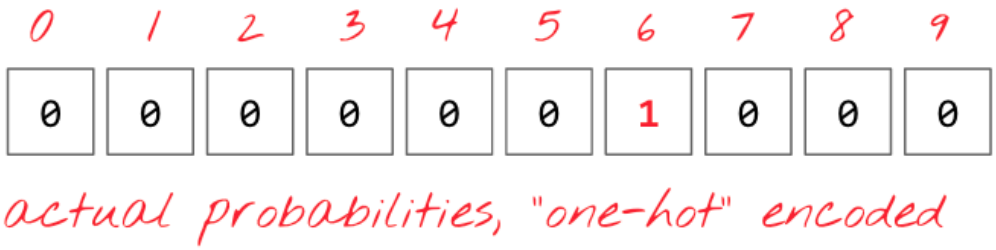
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow (Python) Softmax

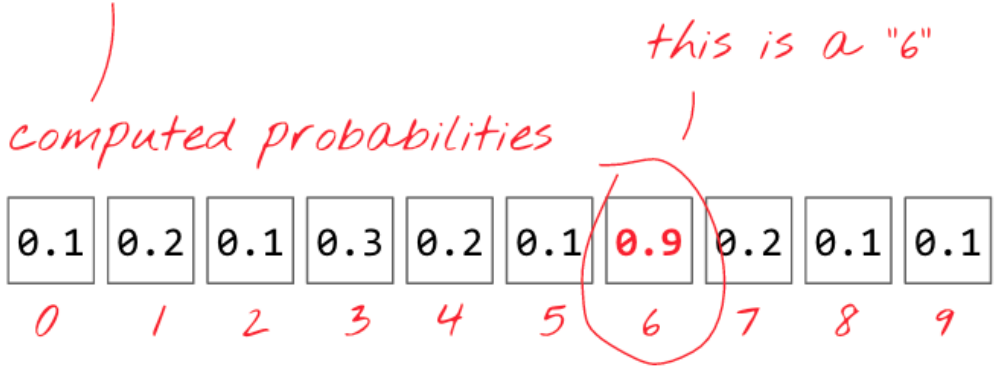


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Cross Entropy

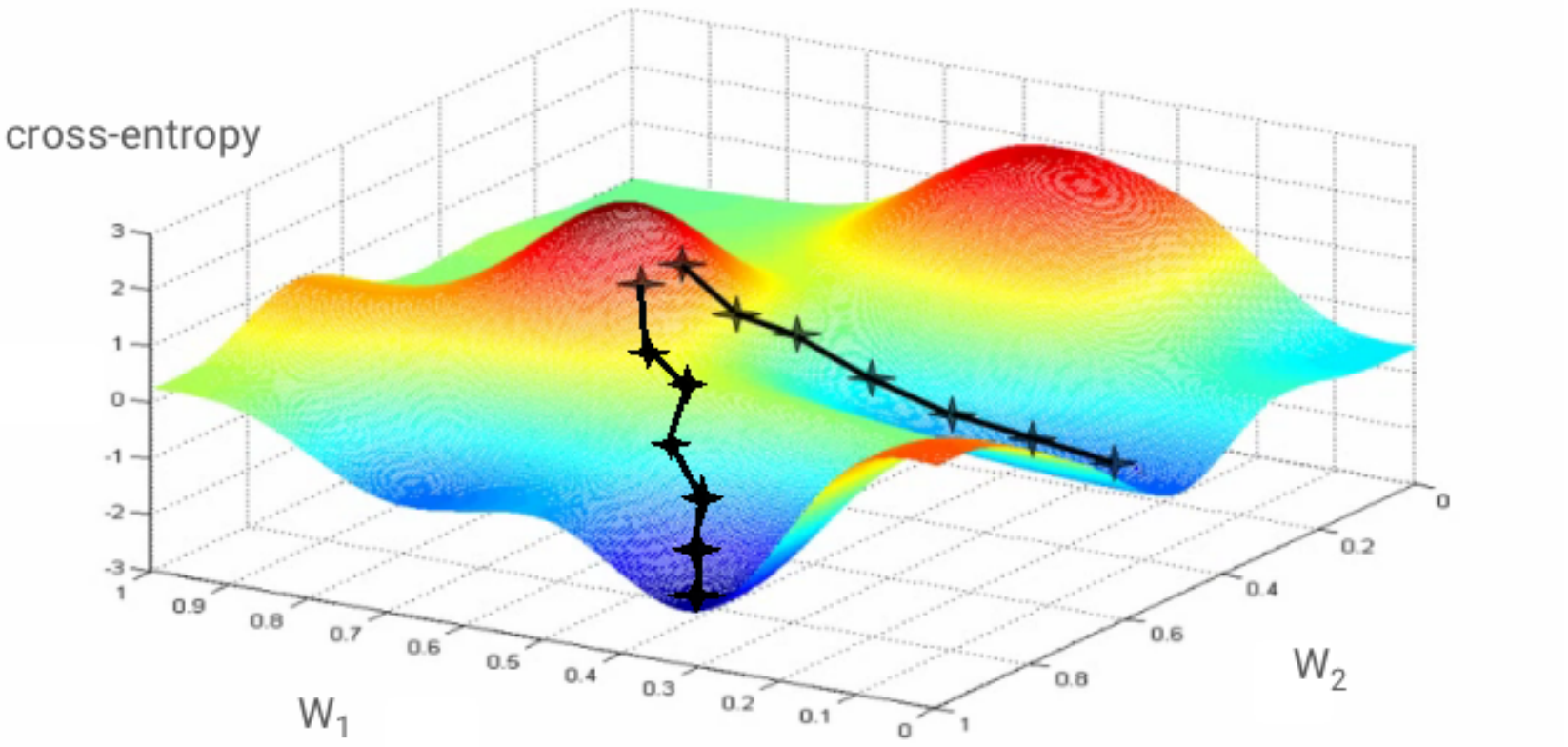


Cross entropy: $-\sum Y_i' . log(Y_i)$



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Minimizing Cross Entropy (Minimizing Loss)



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

"mini-batches":
100 images and labels

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

```
import tensorflow as tf
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

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()
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

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))
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

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)
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

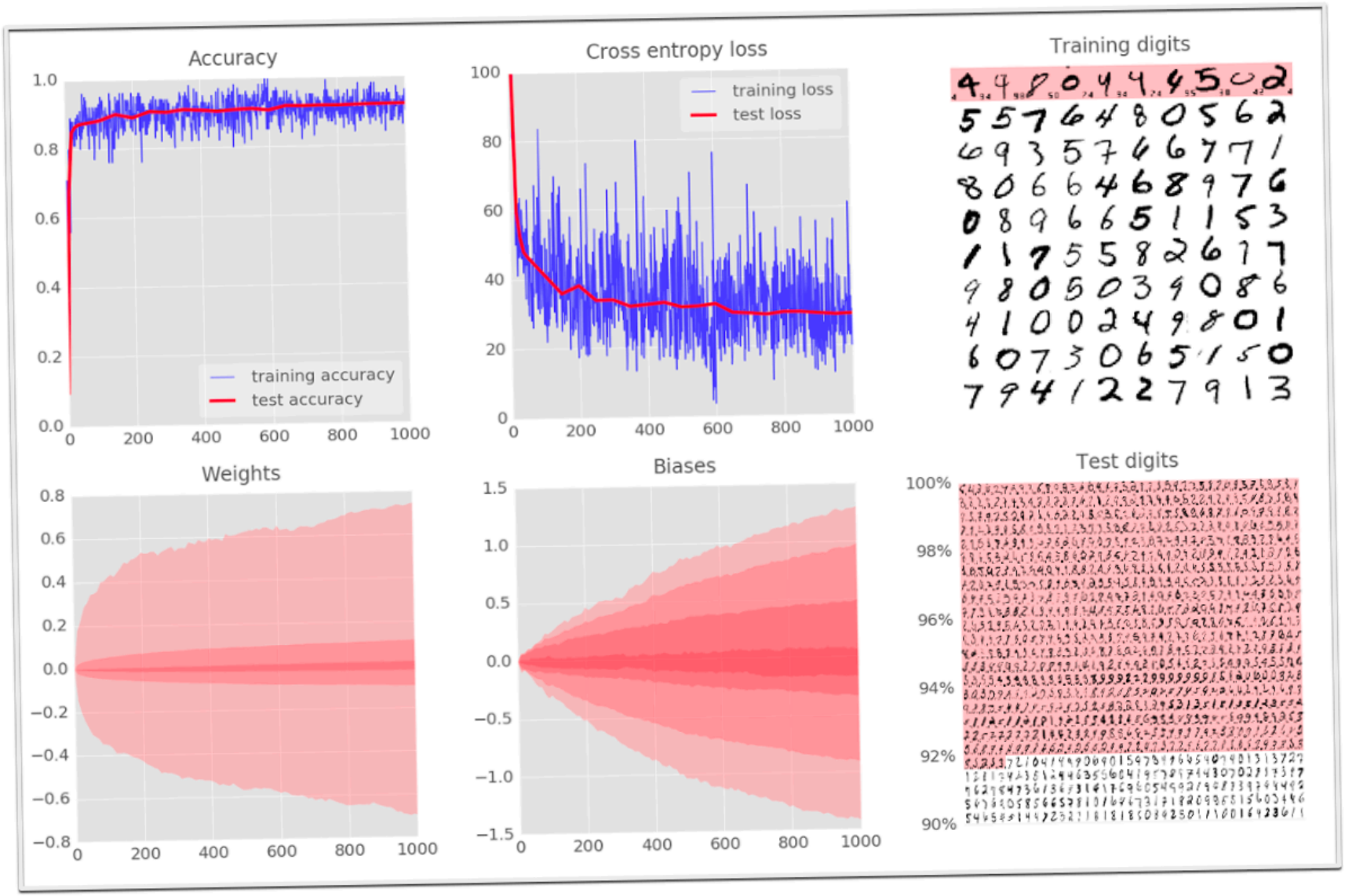
mnist_1.0_softmax.py

```
# 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)
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

mnist_1.0_softmax.py



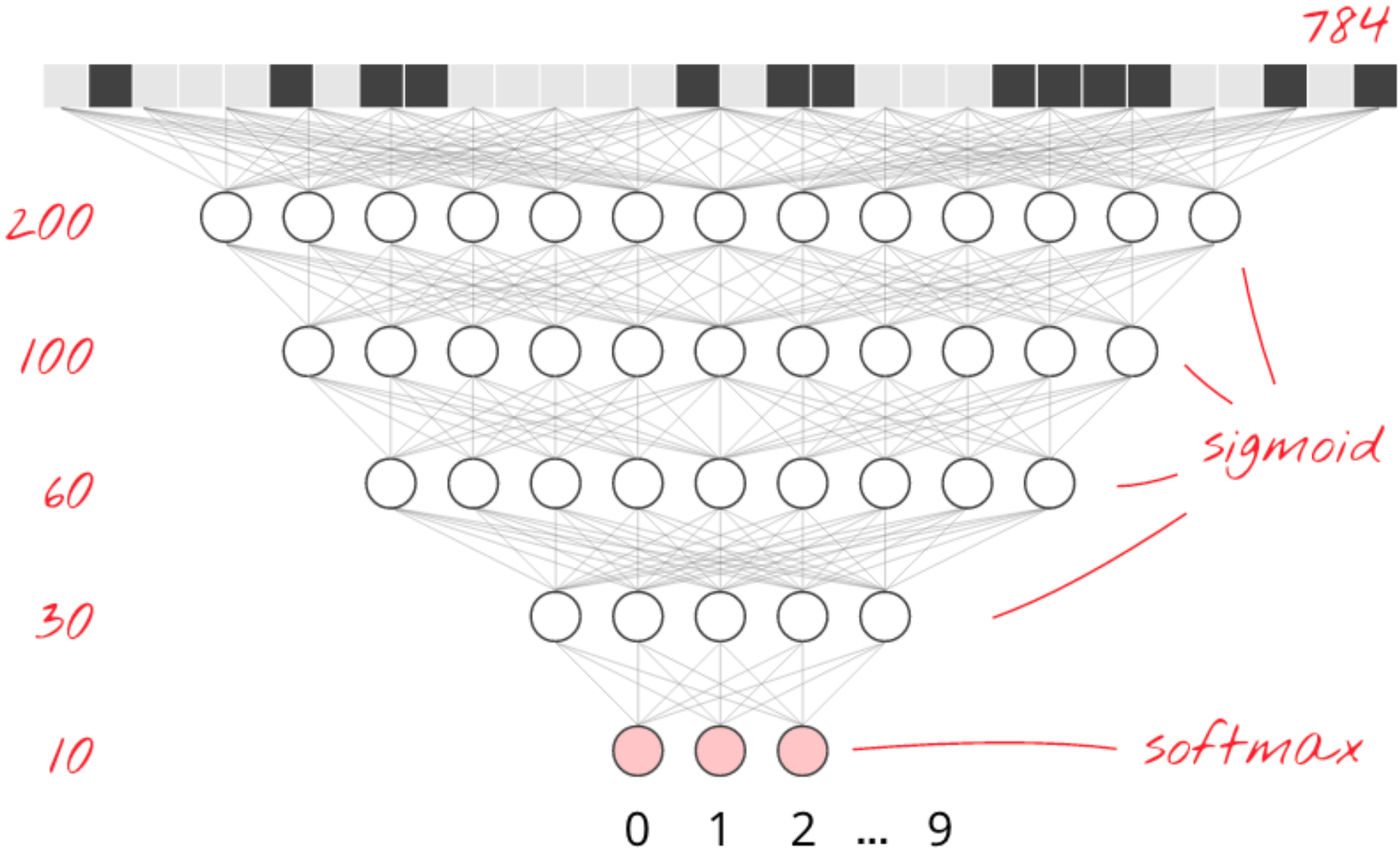
92%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

Deep Learning

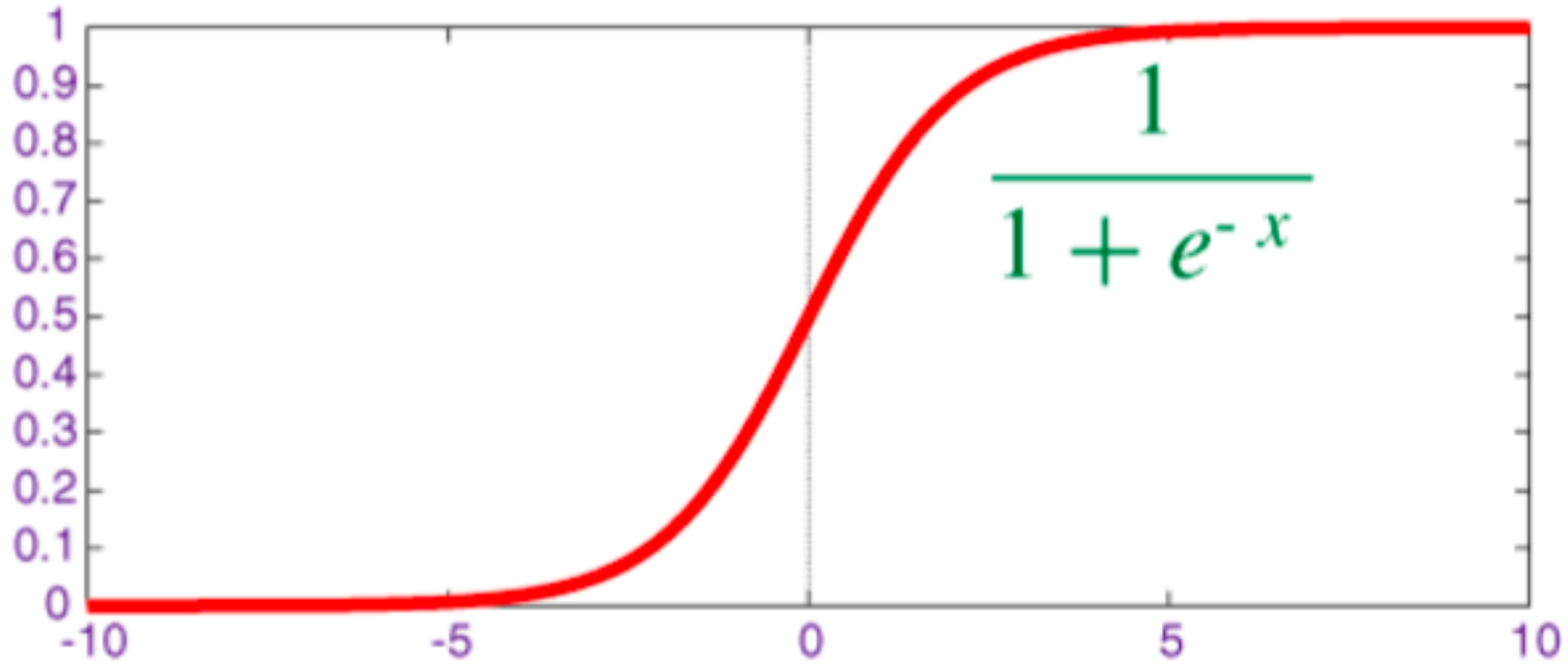
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

5 fully-connected layers



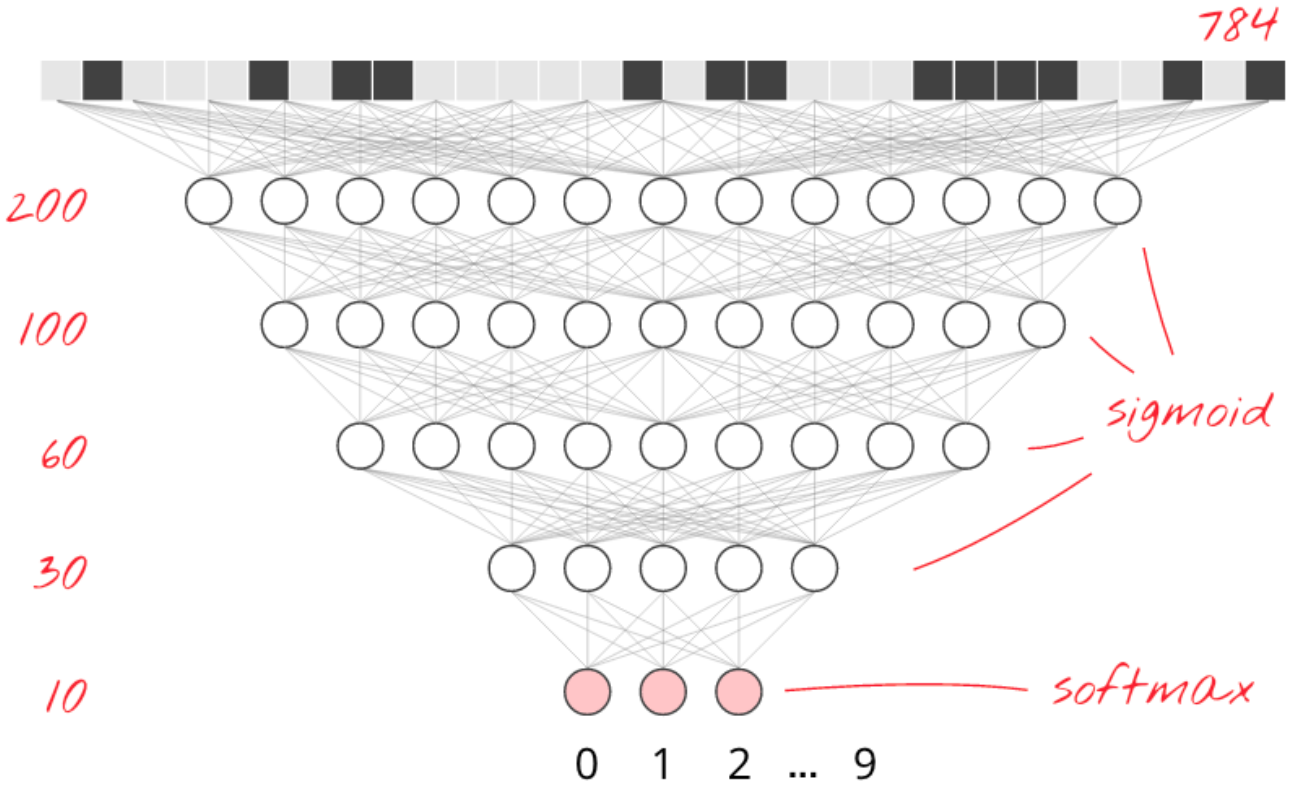
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

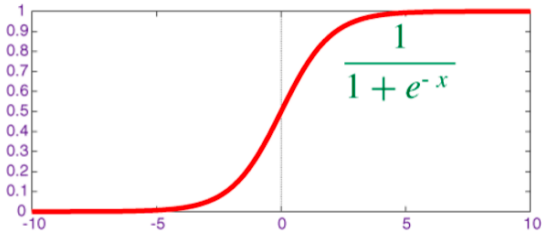
Sigmoid



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

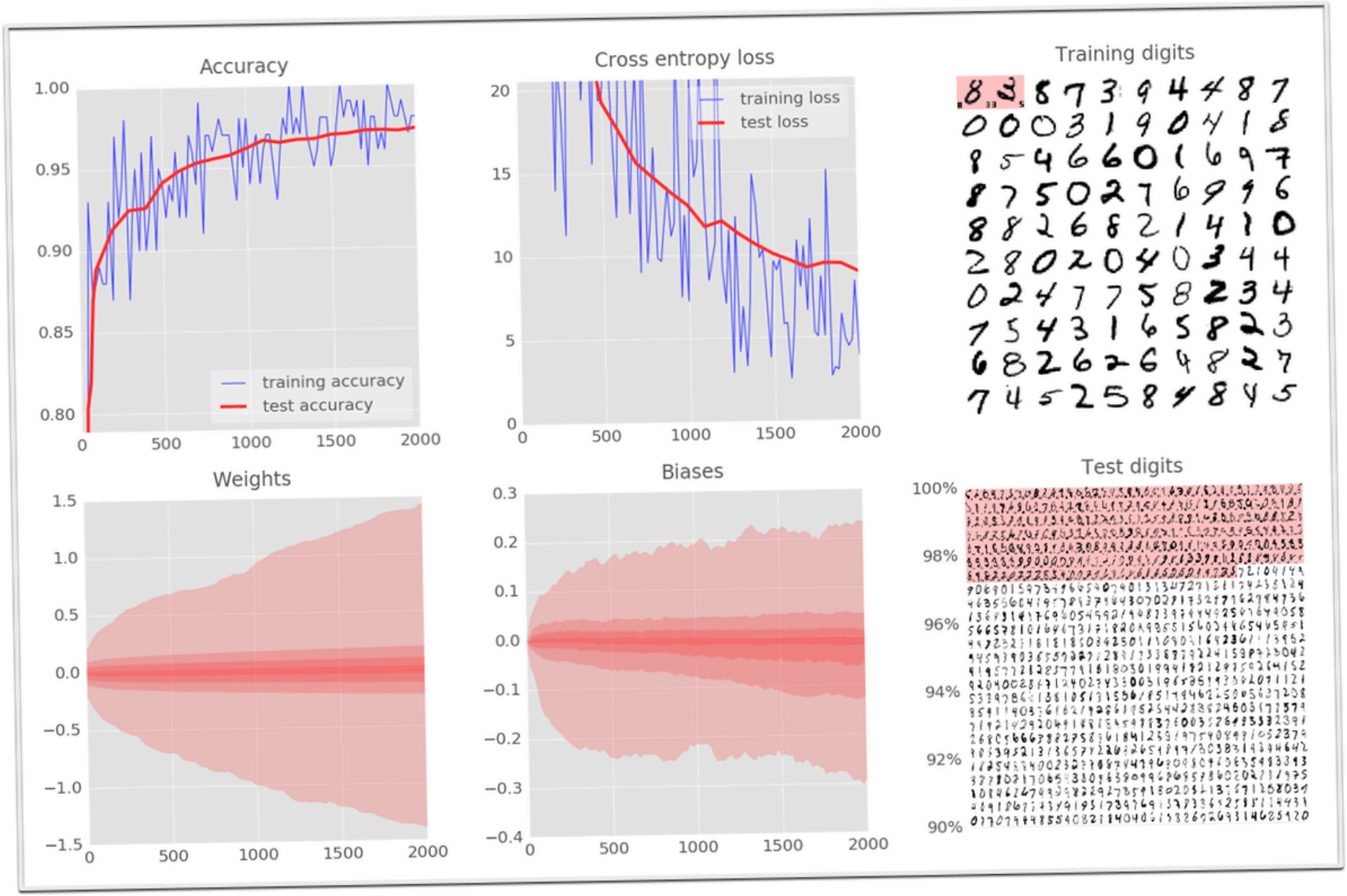
5 fully-connected layers





Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



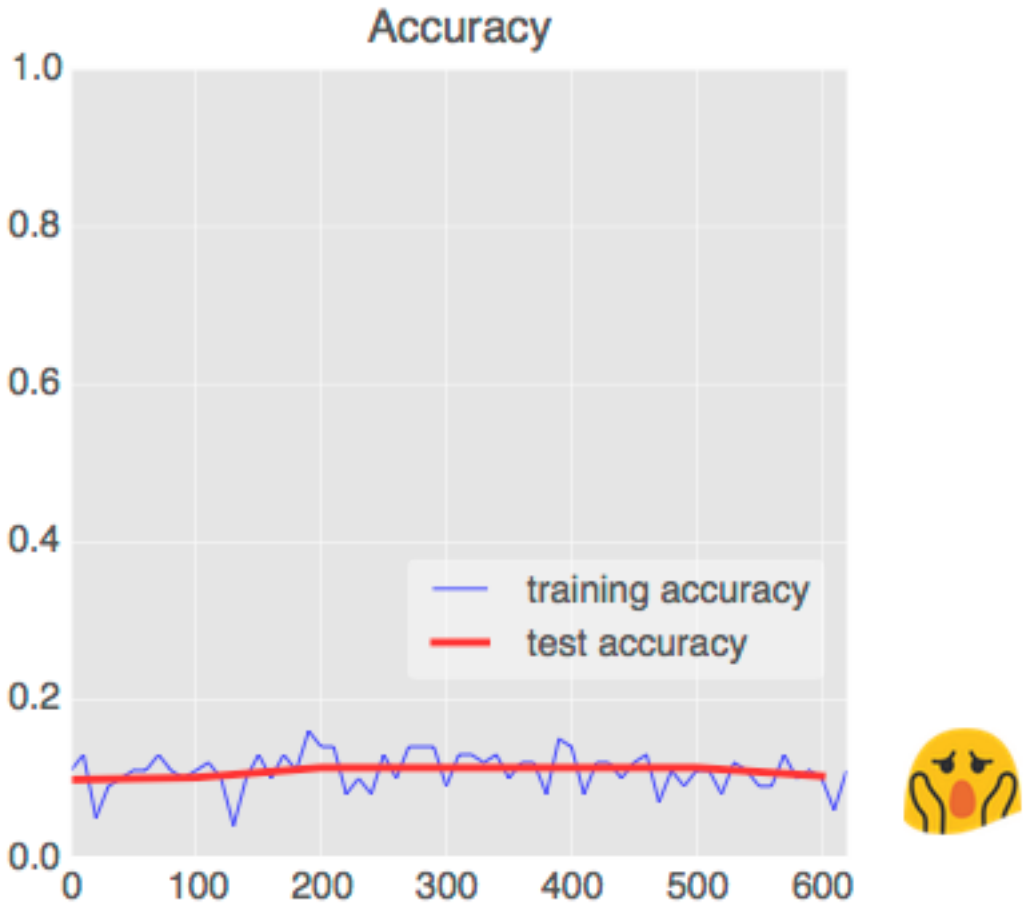
97%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

ReLU

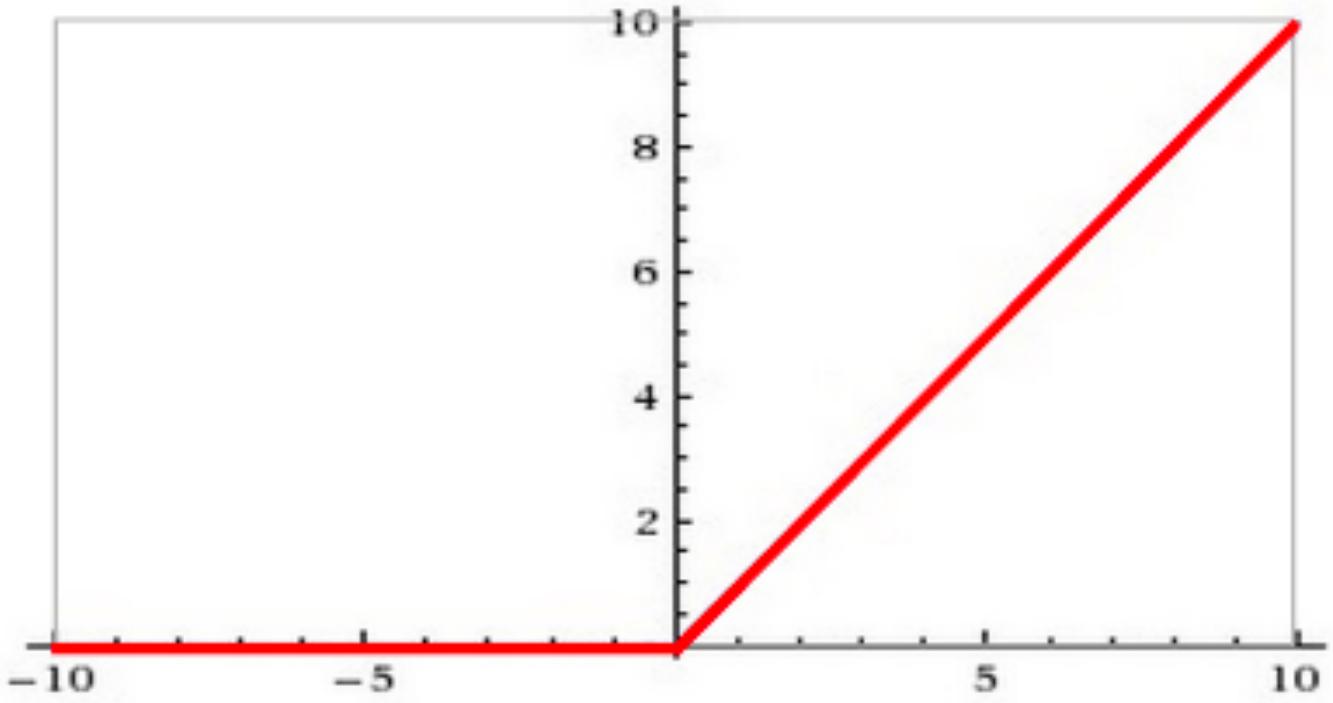
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



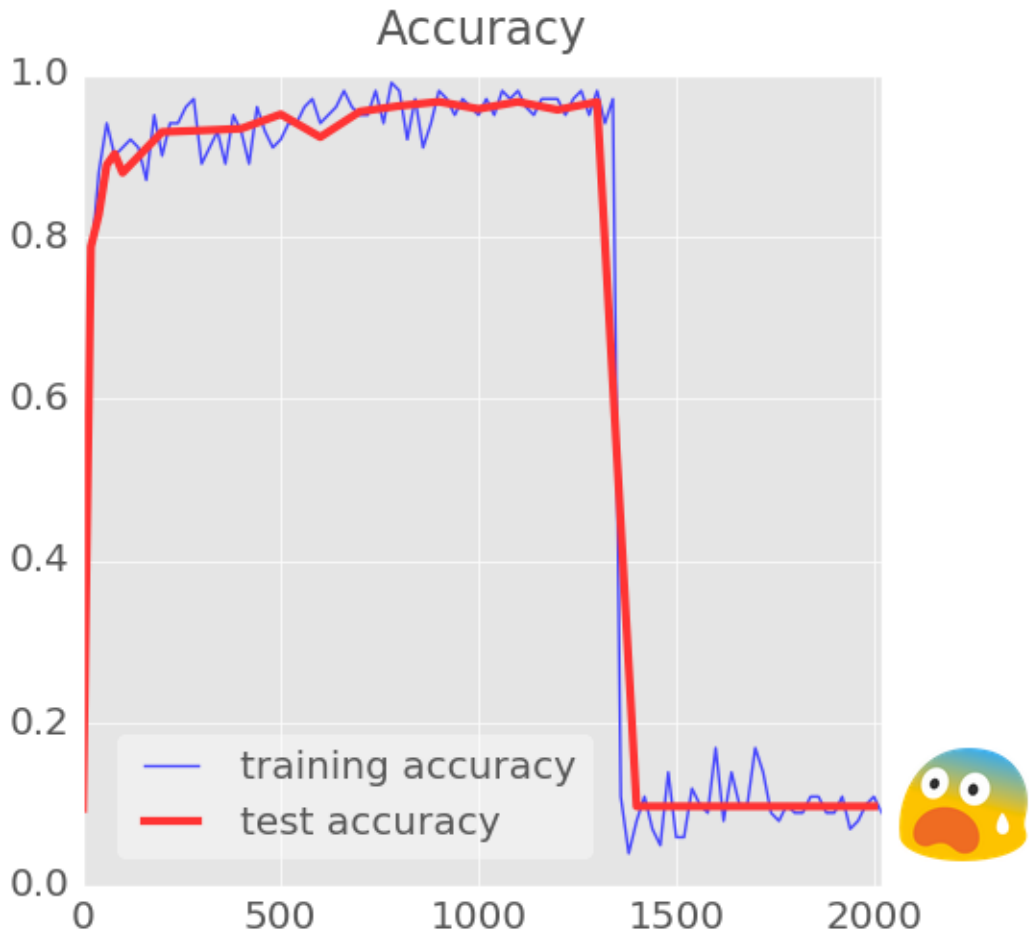
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

ReLU



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



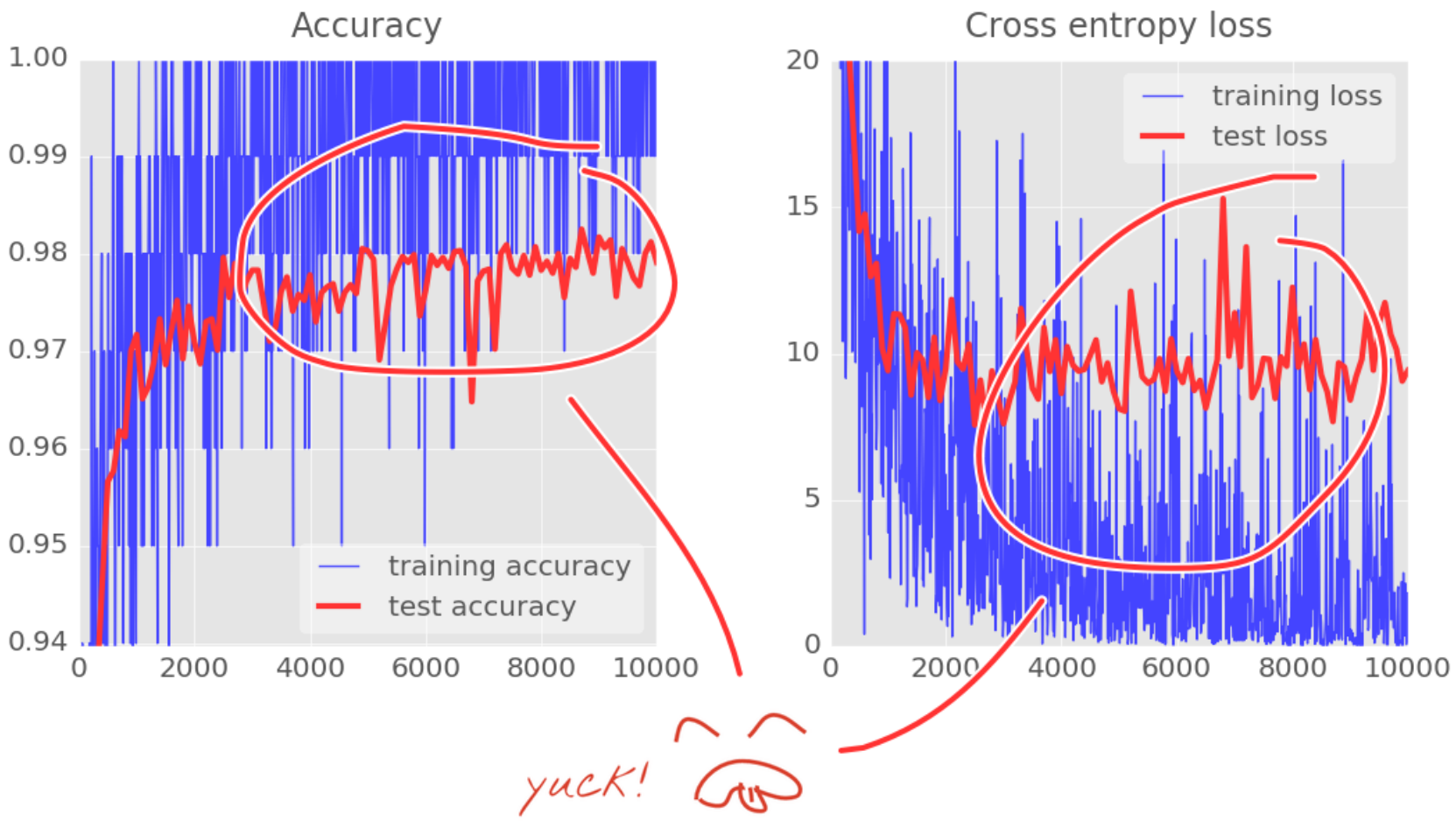
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Learning Rate

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

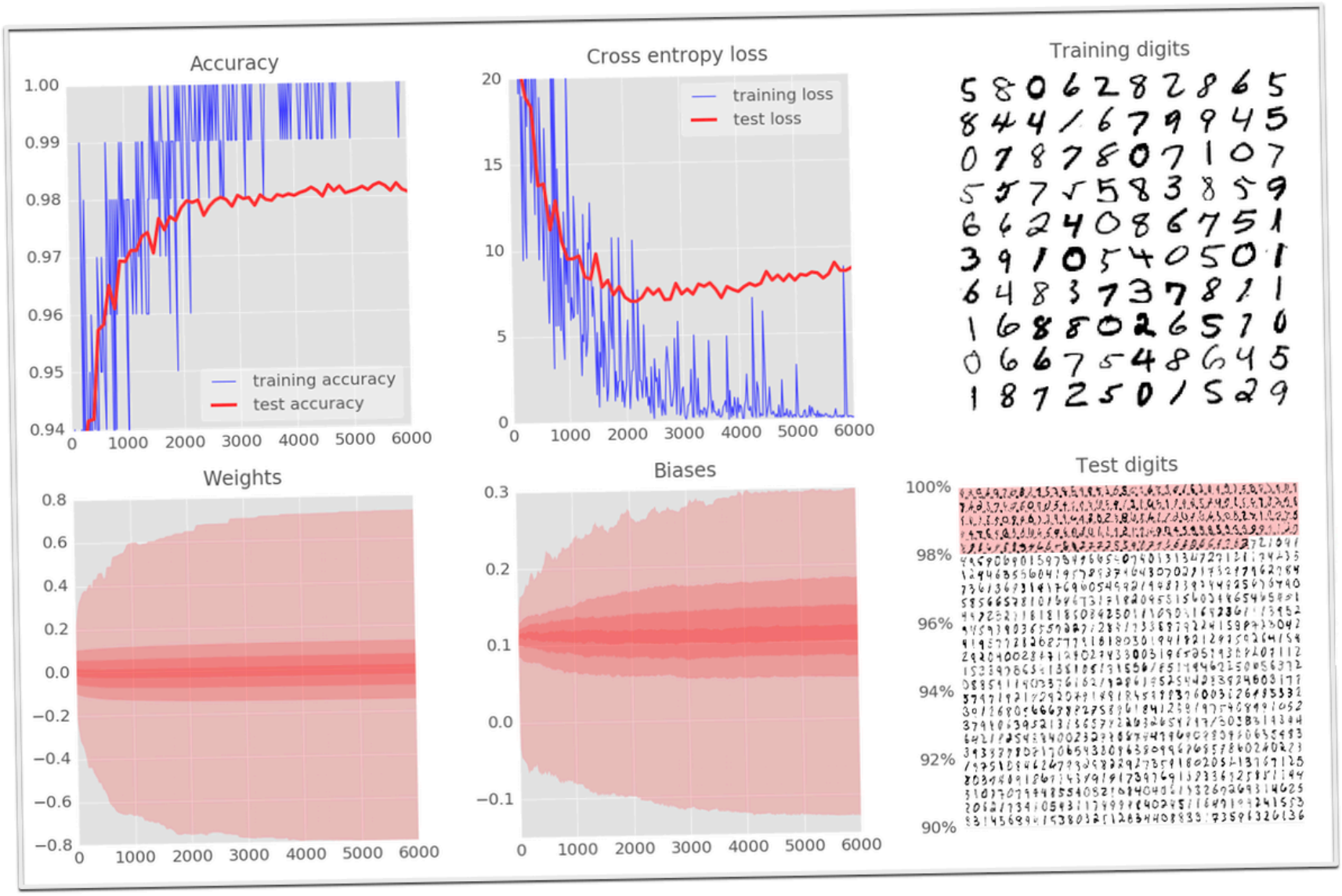
TensorFlow MNIST Tutorial

LR = 0.003



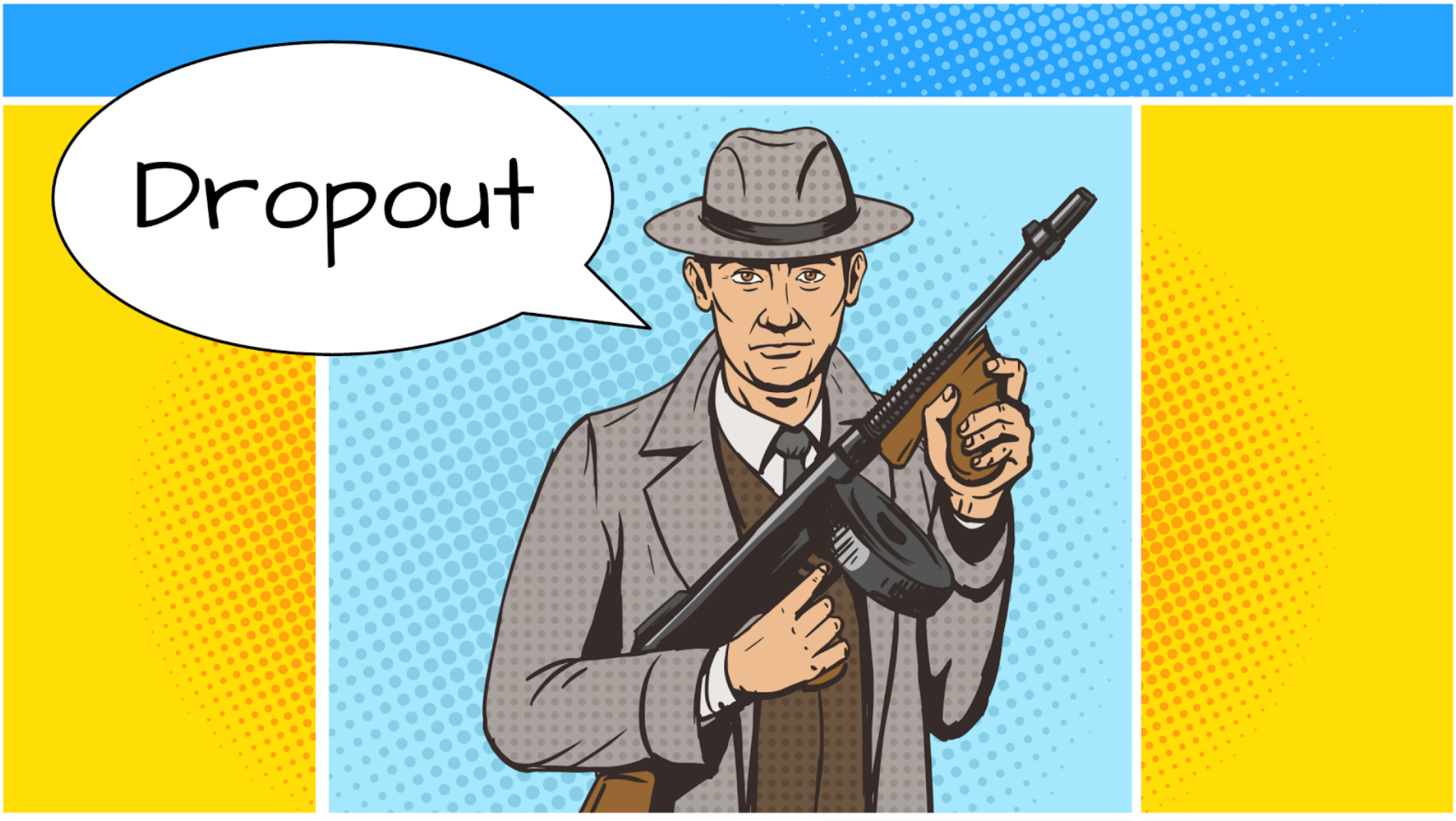
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



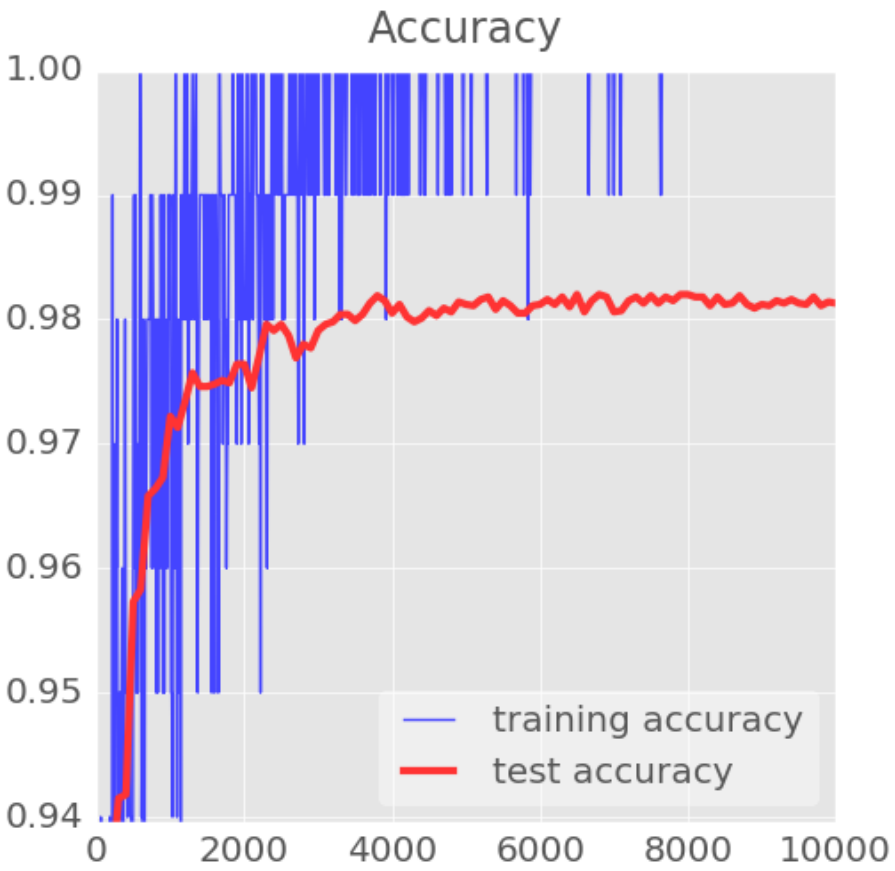
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

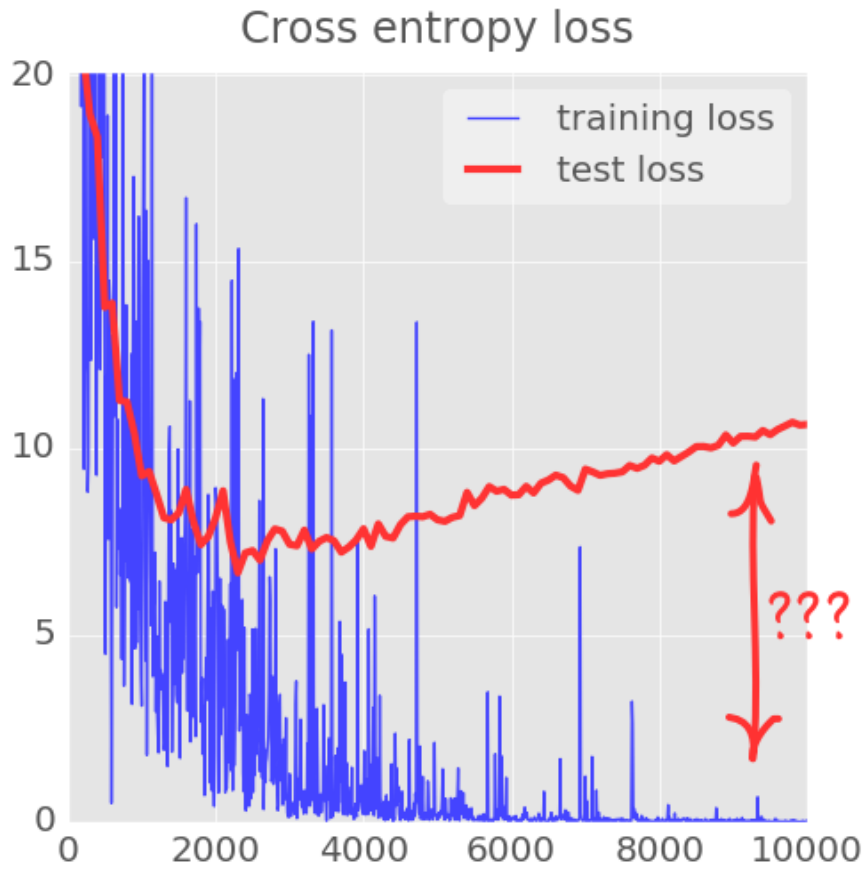
Dropout



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

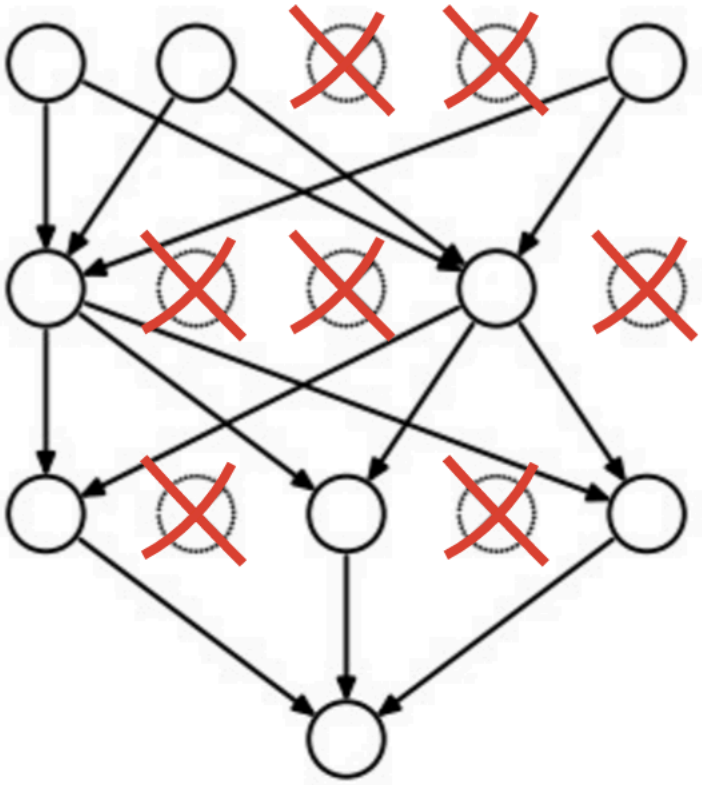
Overfitting



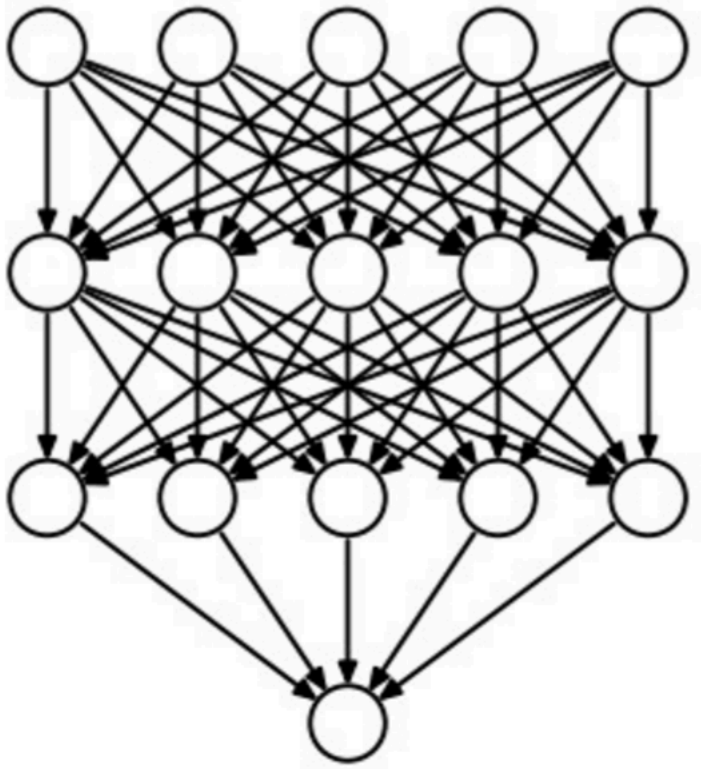


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Dropout



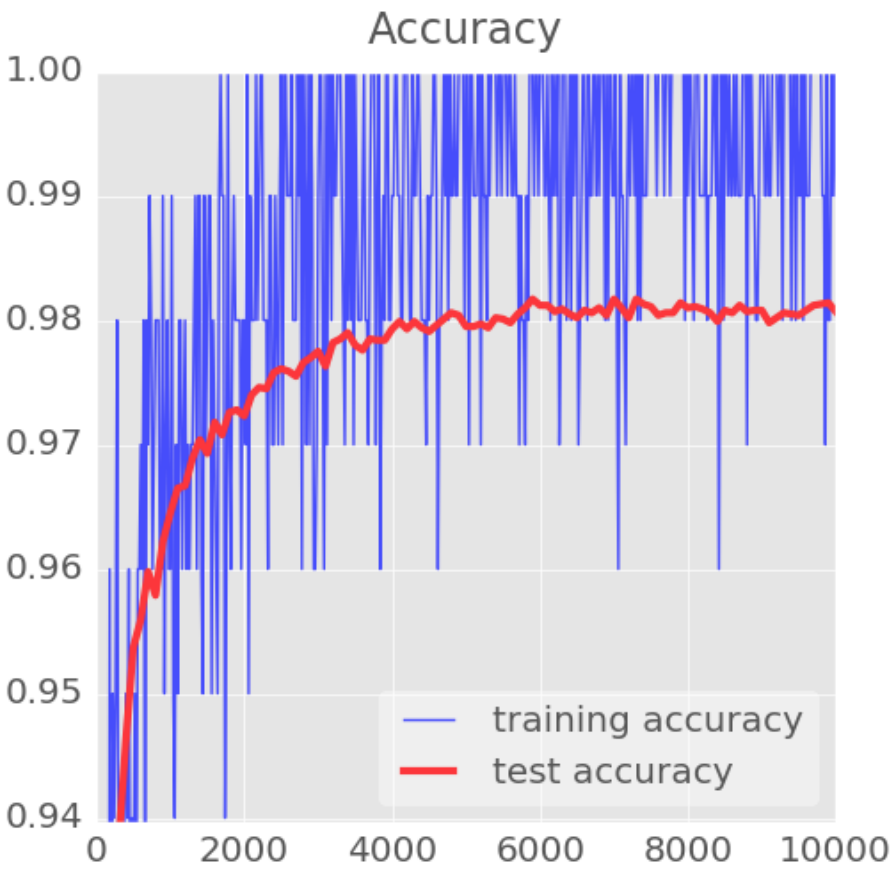
Training
pkeep = 0.75

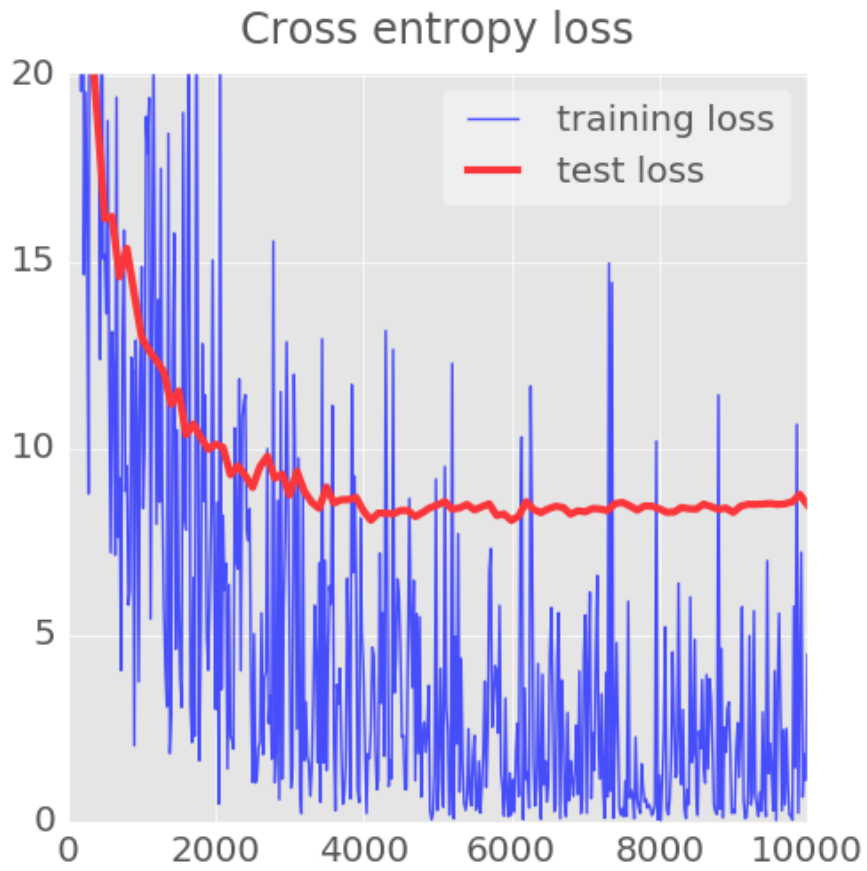


Test
pkeep = 1.0

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

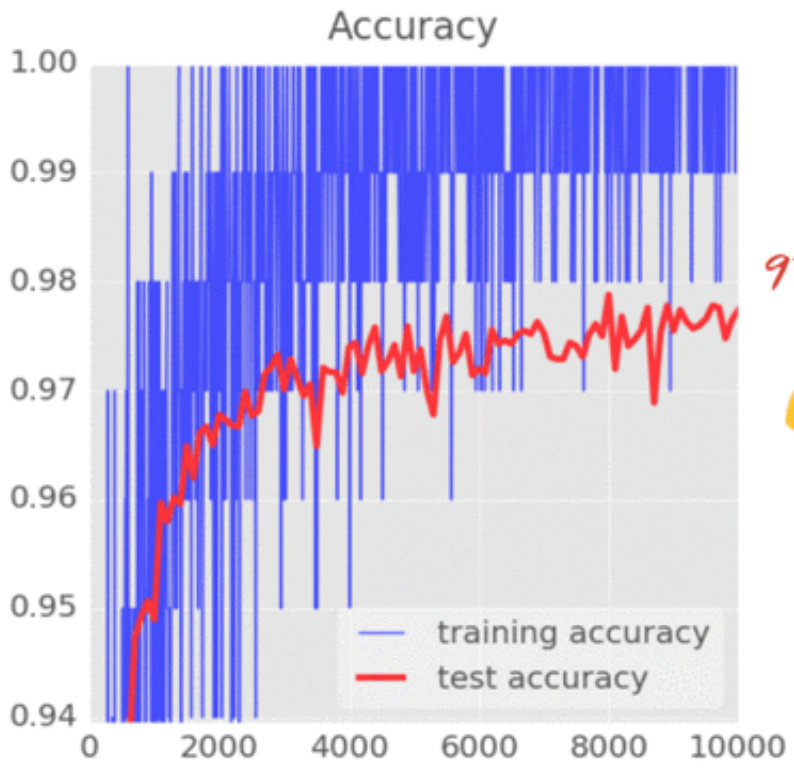




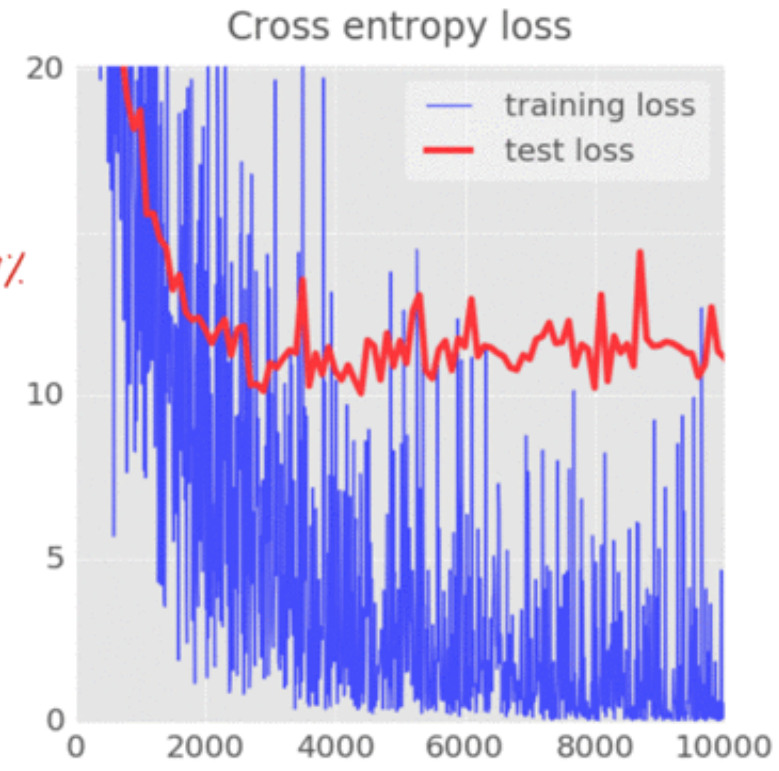
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

5 layers
sigmoid



97.9%

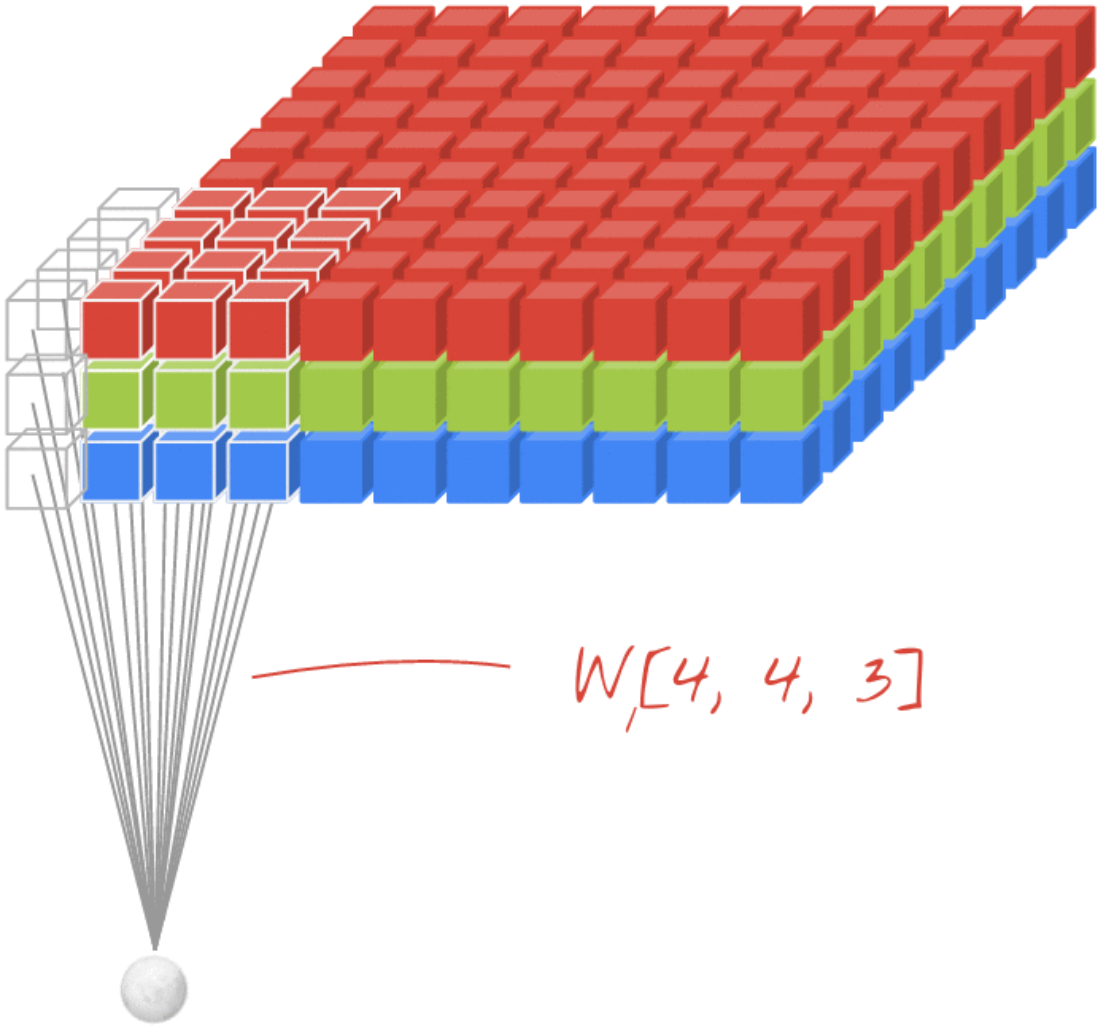


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Overfitting

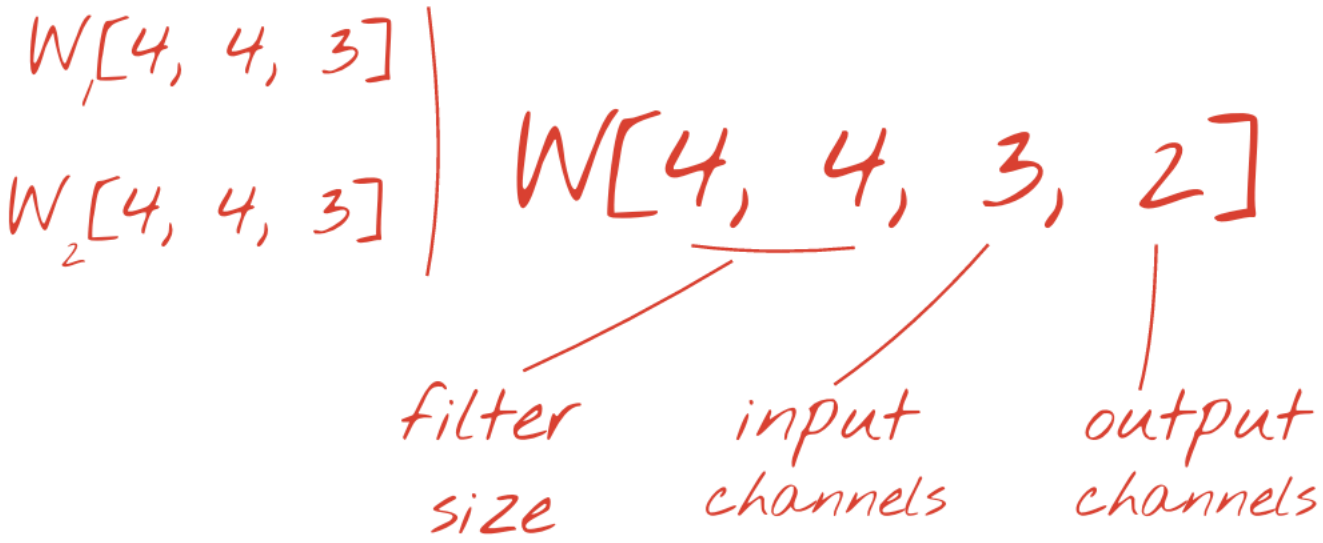
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Layer



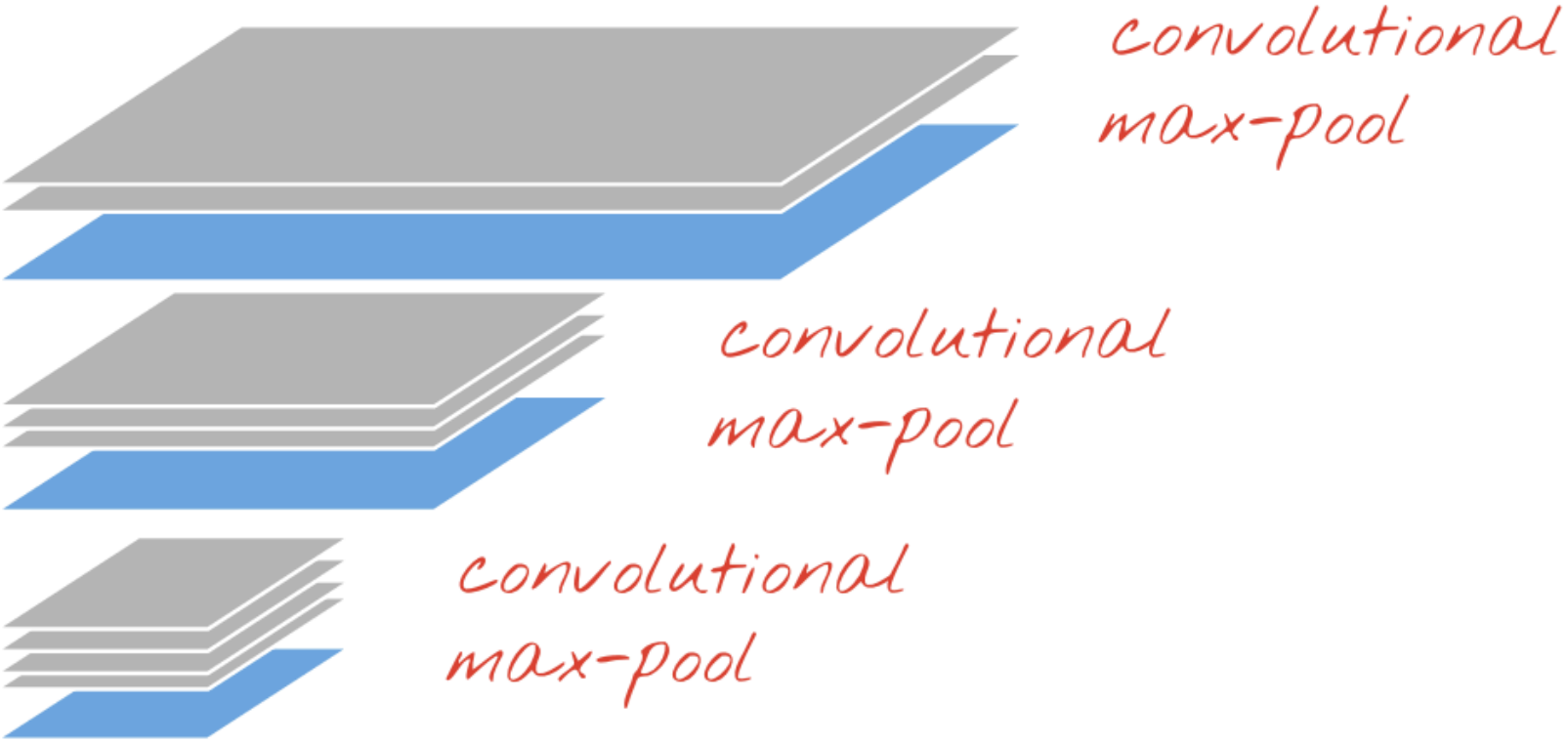
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Layer



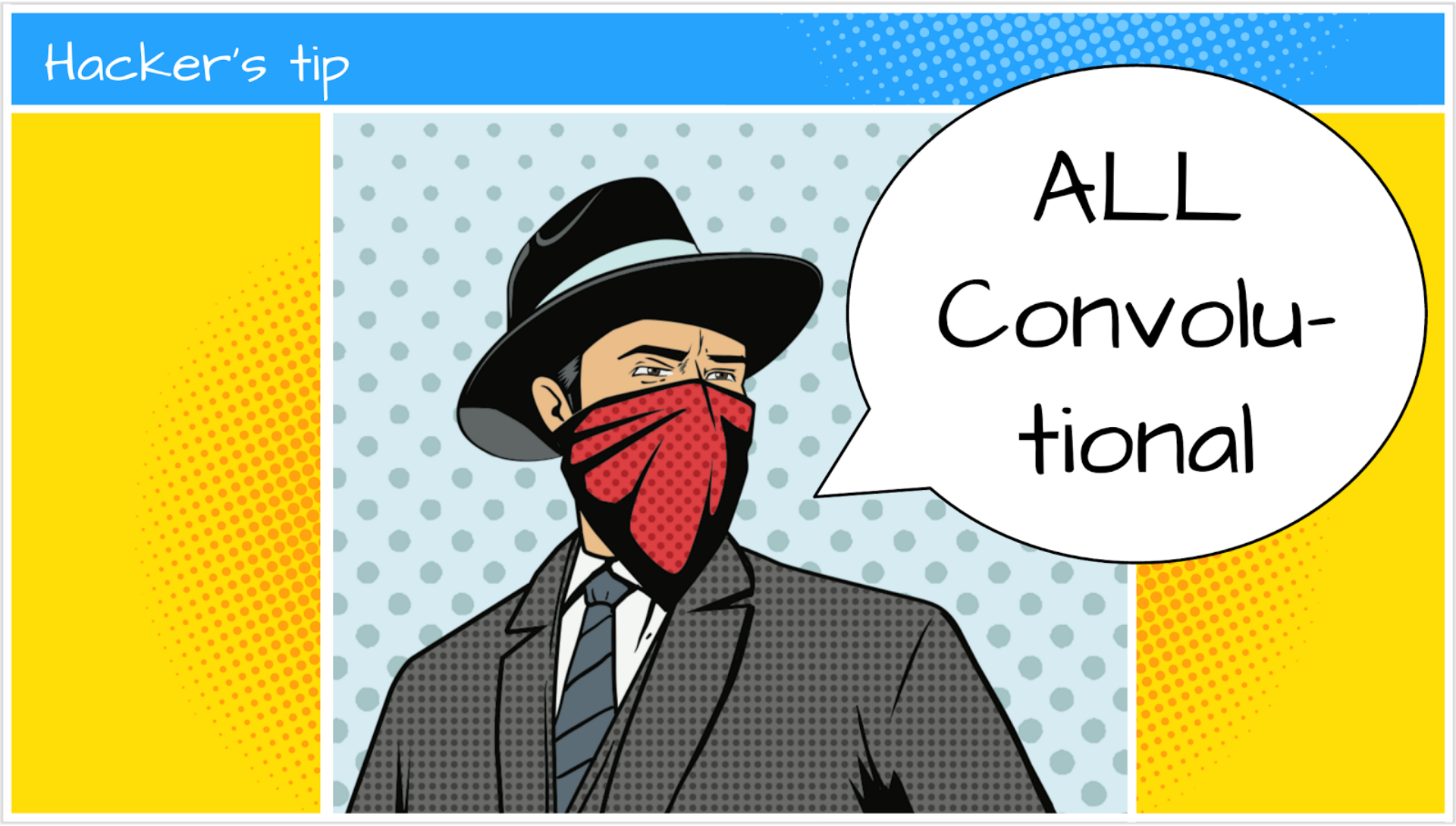
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Max-Pool



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

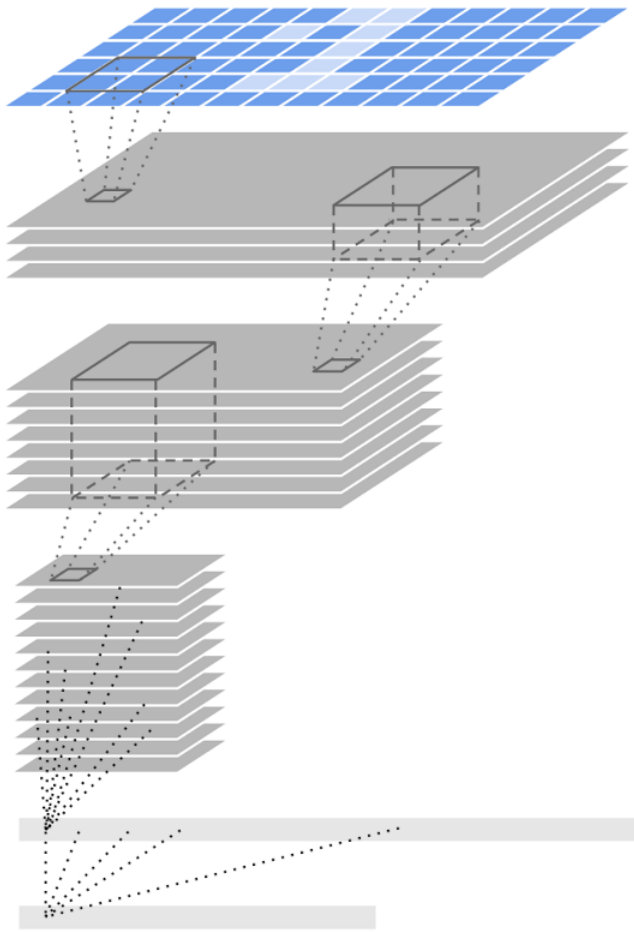
All Convolutional



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

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

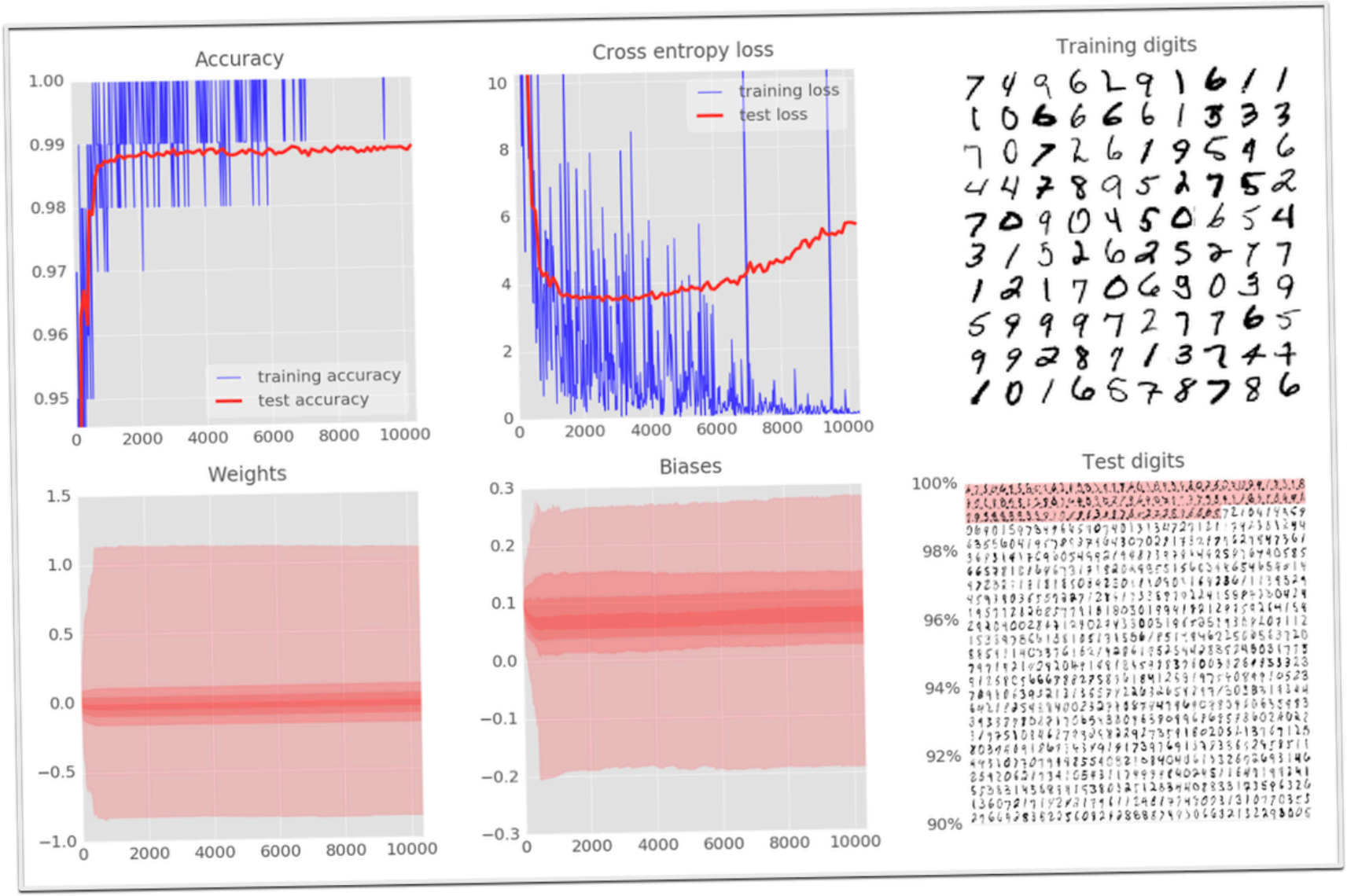
convolutional layer, 12 channels
W3[4, 4, 8, 12] stride 2

fully connected layer
W4[7x7x12, 200]

softmax readout layer
W5[200, 10]

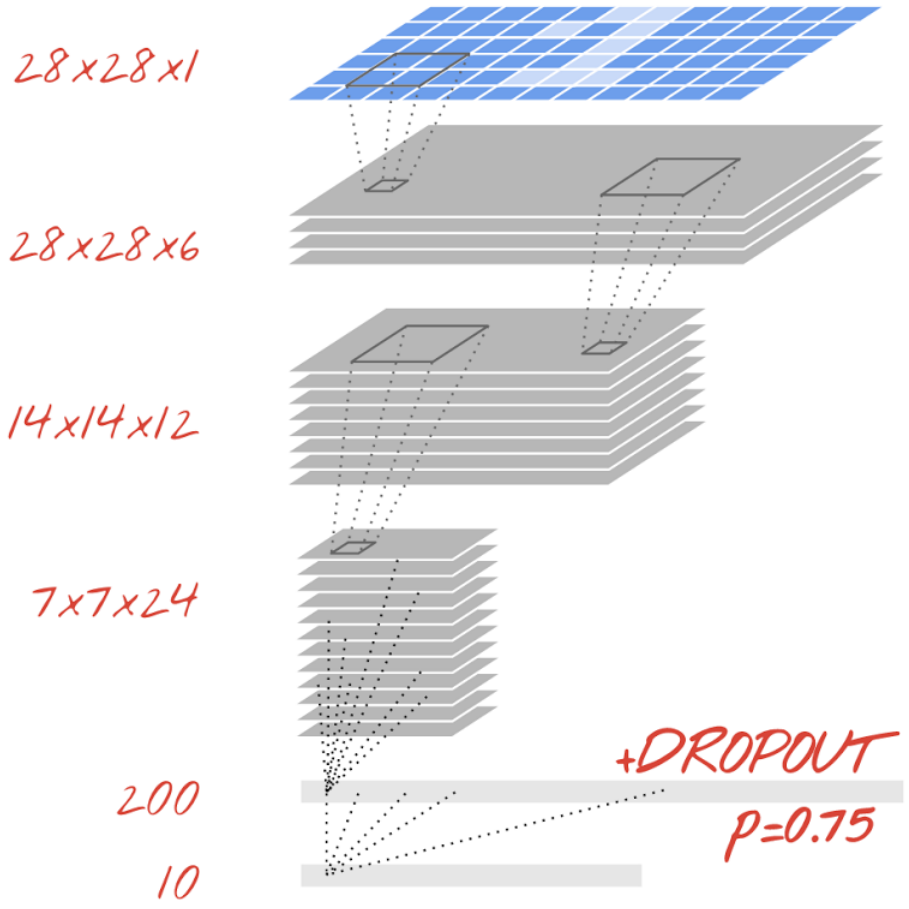
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Bigger Convolutional Neural Network



98.9%

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

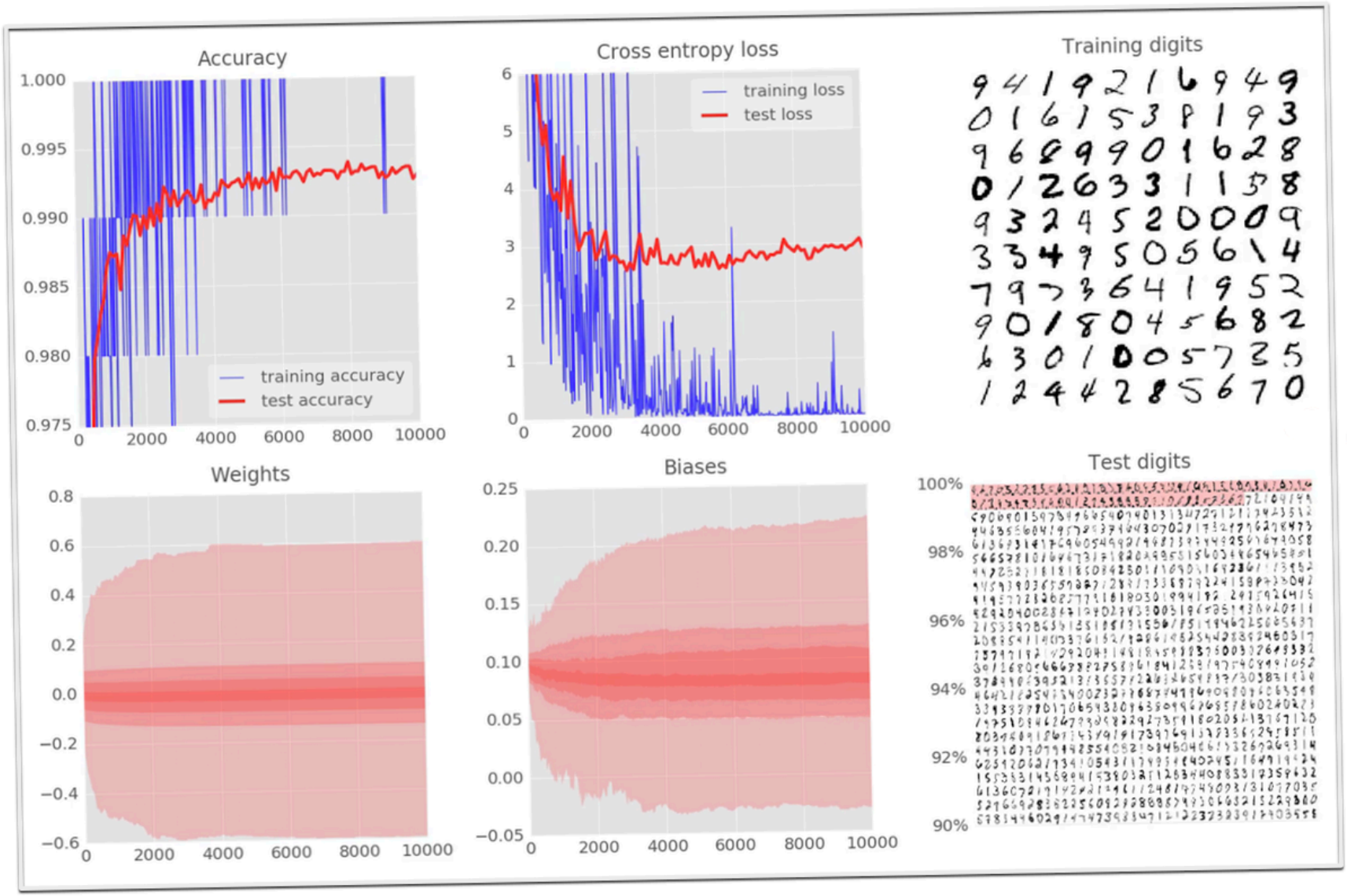
convolutional layer, **24** channels
W3[**4**, **4**, 12, 24] stride 2

fully connected layer
W4[7x7x24, 200]

softmax readout layer
W5[200, 10]

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

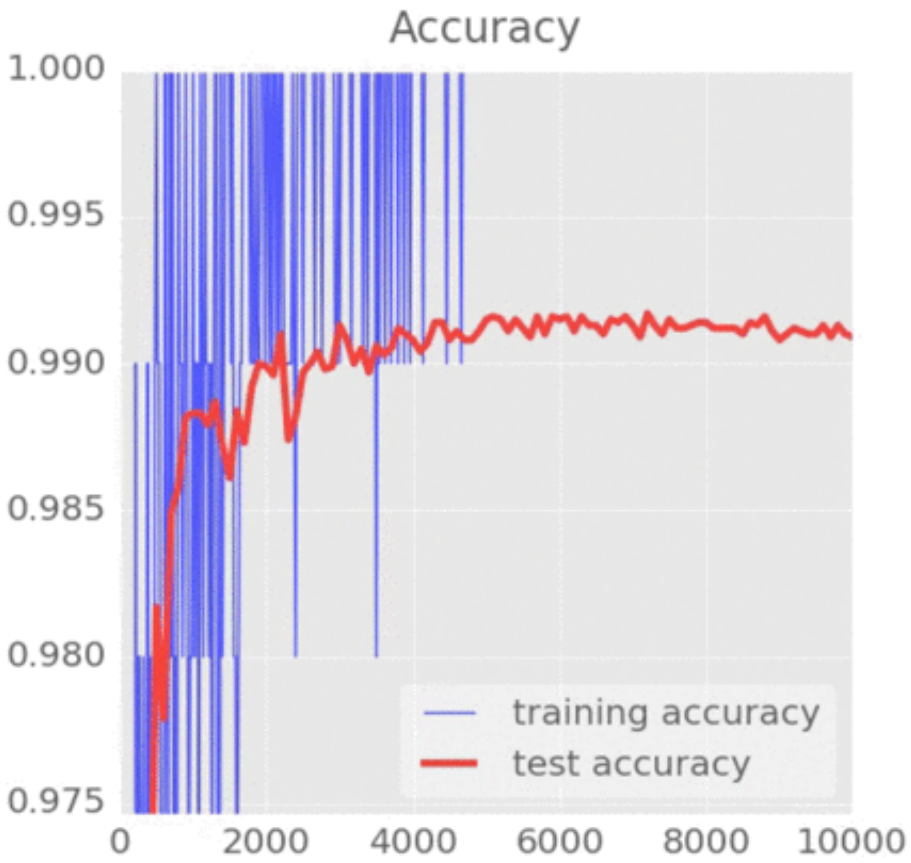
TensorFlow MNIST Tutorial

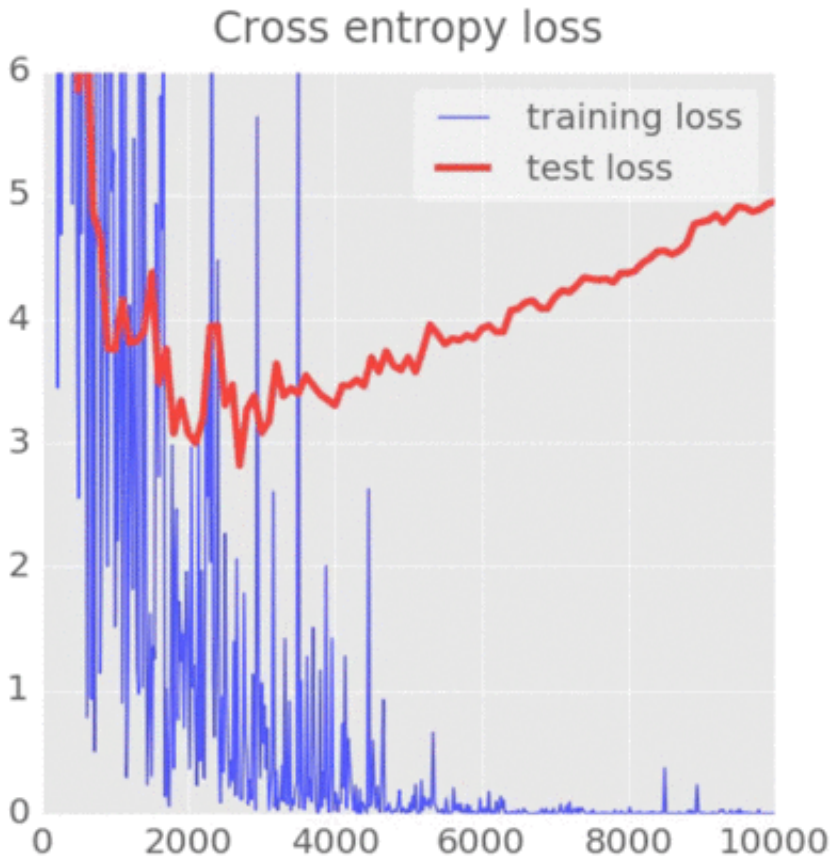


99.3%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

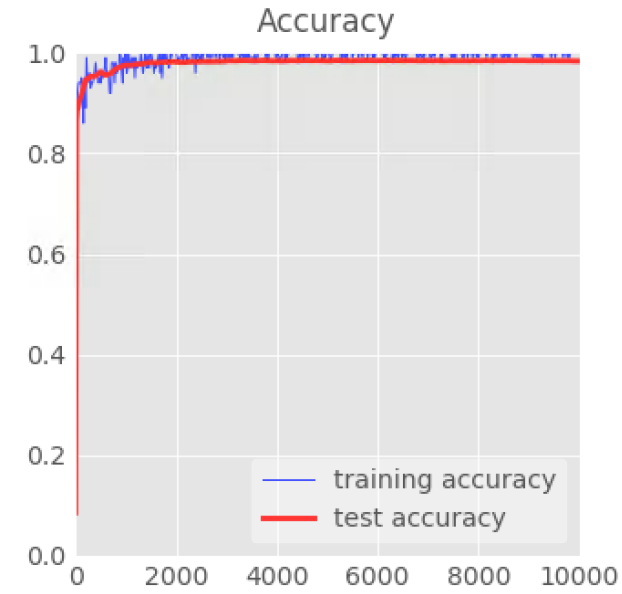


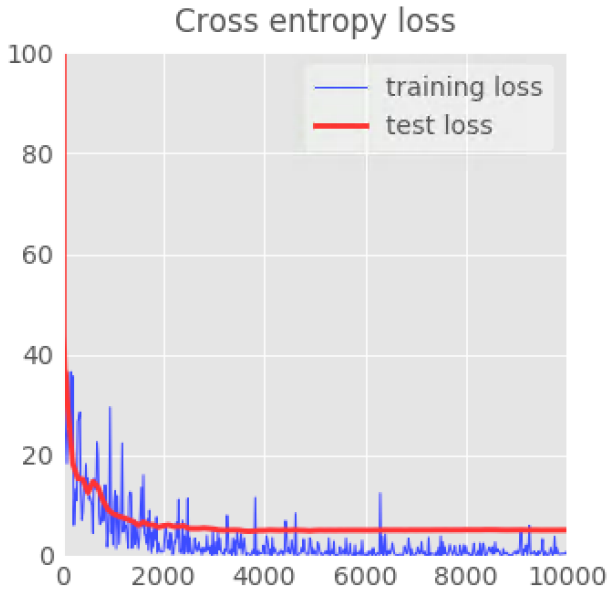


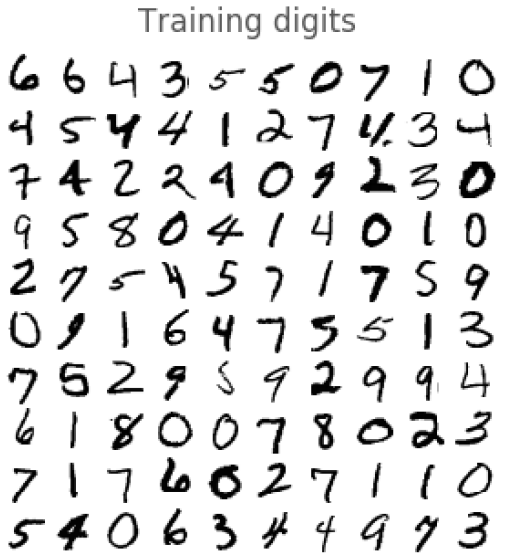
larger convolutional network

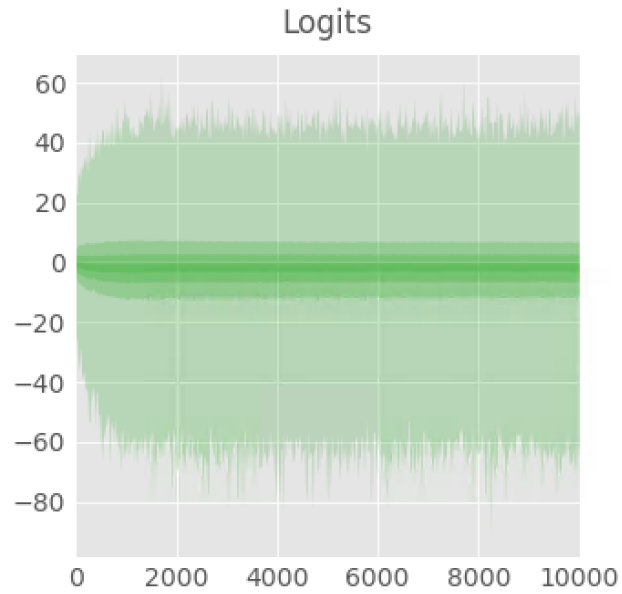
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

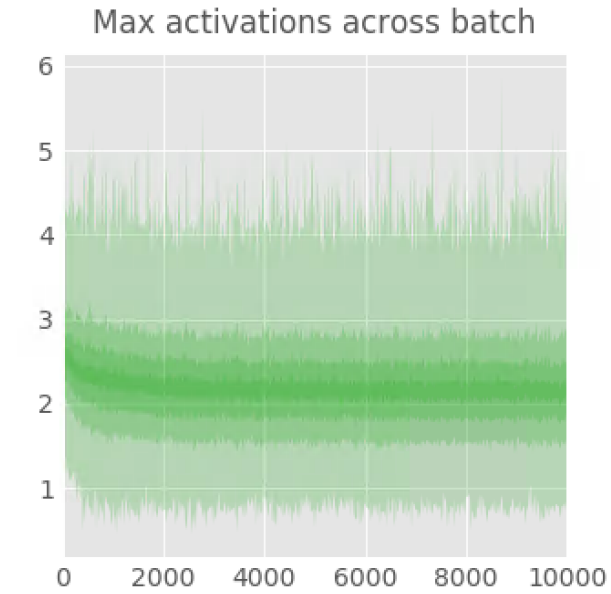
TensorFlow MNIST Tutorial

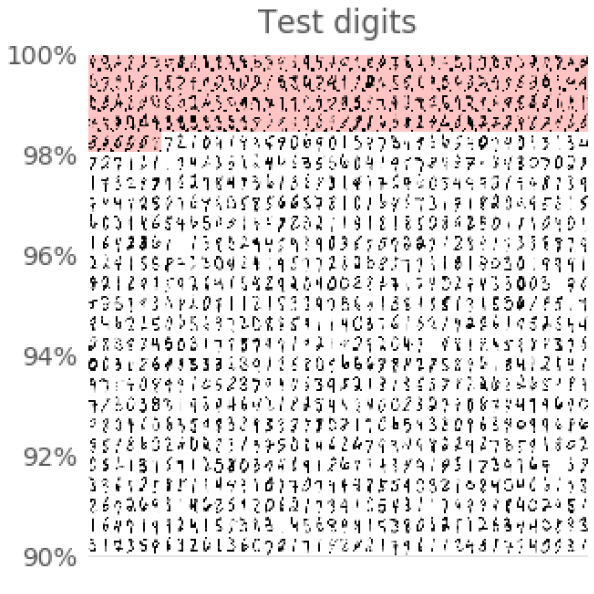












Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/

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
```

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), https://www.youtube.com/watch?v=u4alGiomYP4
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFfI8
- 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=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 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=Ilg3gGewQ5U
- 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

國立臺北護理健康大學
National Taipei University of Nursing and Health Sciences

Q & A

Tamkang University
淡江大學

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