金融科技

FinTech: Financial Technology

Tamkang University
淡江大學

人工智慧與深度學習金融科技

(Artificial Intelligence and Deep Learning for Fintech)

1052FinTech08
MIS EMBA (M2263) (8595)
Fri, 12,13,14 (19:20-22:10) (D409)

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/

2017-05-05

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/02/17	Fintech 金融科技課程介紹 (Course Orientation for Fintech: Financial Technology)
2	2017/02/24	Fintech 金融科技的演進：貨幣與金融服務 (Evolution of Fintech: Money and Financial Services)
3	2017/03/03	Fintech 金融科技：金融服務科技創新 (Fintech: Technology Innovation in Financial Services)
4	2017/03/10	Fintech 金融科技與金融服務價值鏈 (Fintech and Financial Services Value Chain)
5	2017/03/17	Fintech 金融科技商業模式創新 (Fintech Business Models Innovation)
6	2017/03/24	Fintech 金融科技個案研究 I (Case Study on Fintech I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2017/03/31	金融服務消費者心理與行為 (Consumer Psychology and Behavior on Financial Services)
8	2017/04/07	教學行政觀摩日 (Off-campus study)
9	2017/04/14	區塊鏈技術 (Blockchain Technology) [Invited Speaker: Dr. Raymund Lin, IBM (林俊叡 博士，IBM)]
10	2017/04/21	期中報告 (Midterm Project Report)
11	2017/04/28	Python Pandas財務大數據分析 (Finance Big Data Analytics with Pandas in Python)
12	2017/05/05	人工智慧與深度學習金融科技 (Artificial Intelligence and Deep Learning for Fintech)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2017/05/12	Fintech 金融科技個案研究 II (Case Study on Fintech II)
14	2017/05/19	金融科技財富管理：機器人理財顧問 (Robo-Advisors for Wealth Management in Fintech)
15	2017/05/26	投資組合最佳化與程式交易 (Portfolio Optimization and Algorithmic Trading)
16	2017/06/02	金融科技智慧問答系統 (Intelligent Question Answering System for Fintech)
17	2017/06/09	期末報告 I (Final Project Presentation I)
18	2017/06/16	期末報告 II (Final Project Presentation II)

Artificial Intelligence and Deep Learning for Fintech

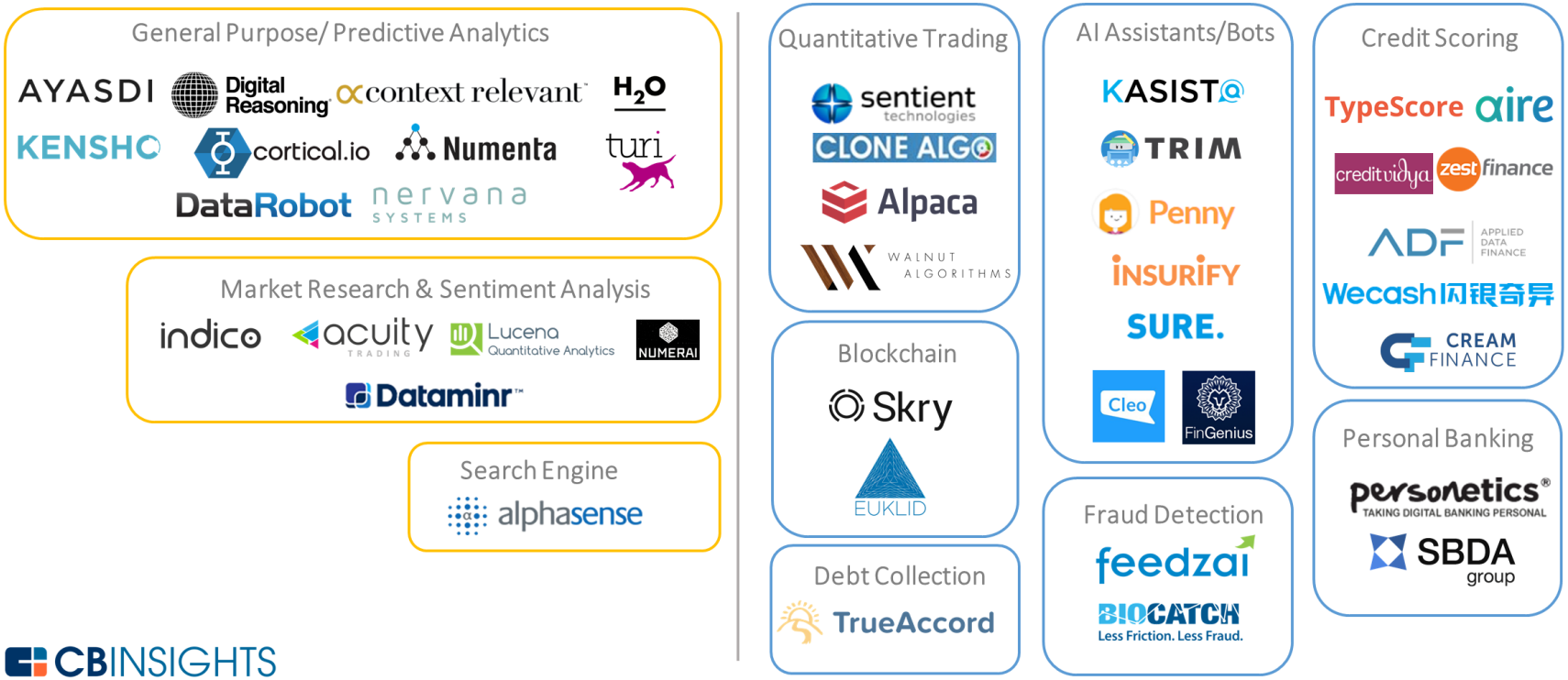
From Algorithmic Trading to Personal Finance Bots: 41 Startups Bringing AI to Fintech

Source: https://www.cbinsights.com/blog/artificial-intelligence-fintech-market-map-company-list/

From Algorithmic Trading To Personal Finance Bots: 41 Startups Bringing AI To Fintech

AI in Fintech

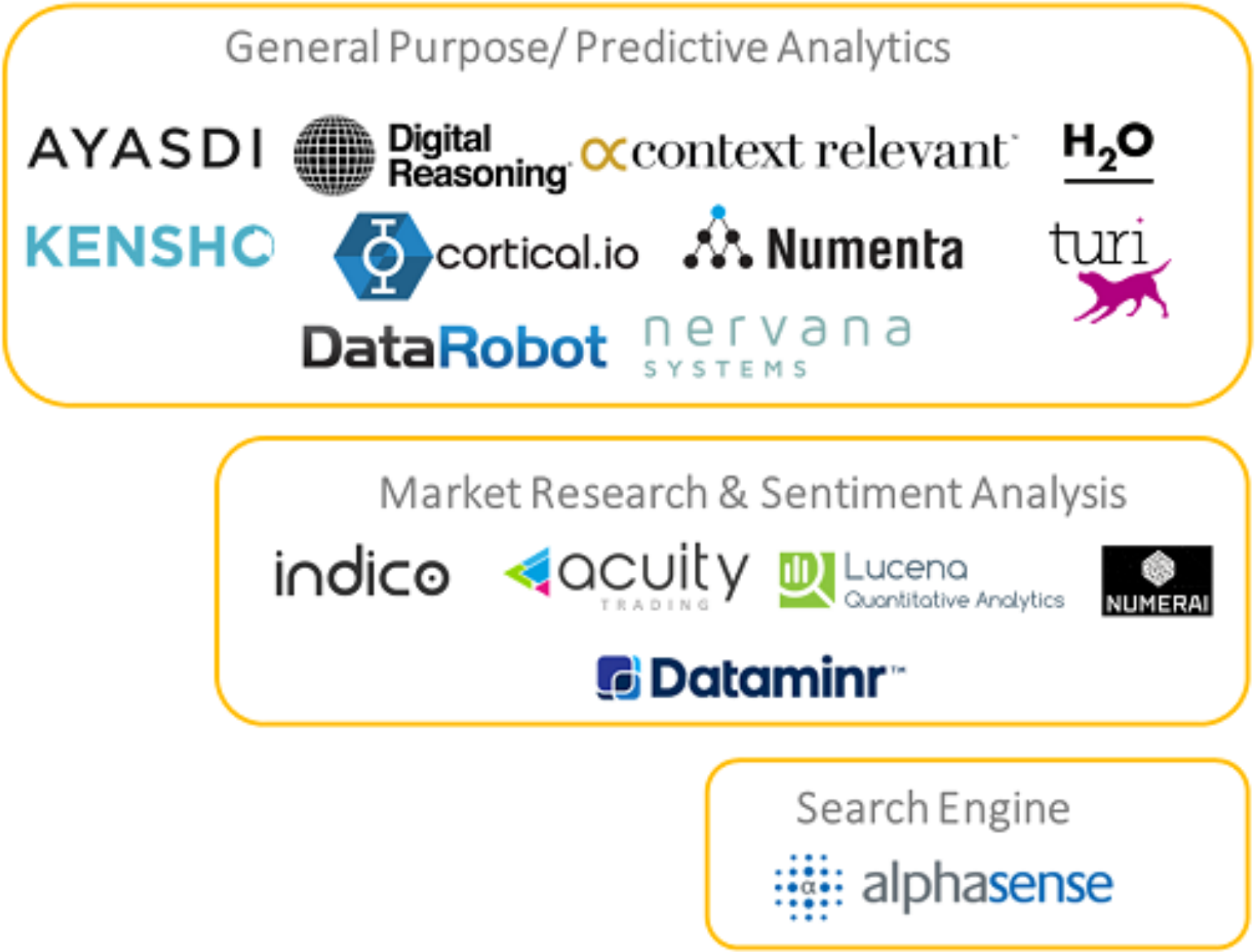
41 Startups Bringing Artificial Intelligence To Fintech



CBINSIGHTS
www.cbinsights.com

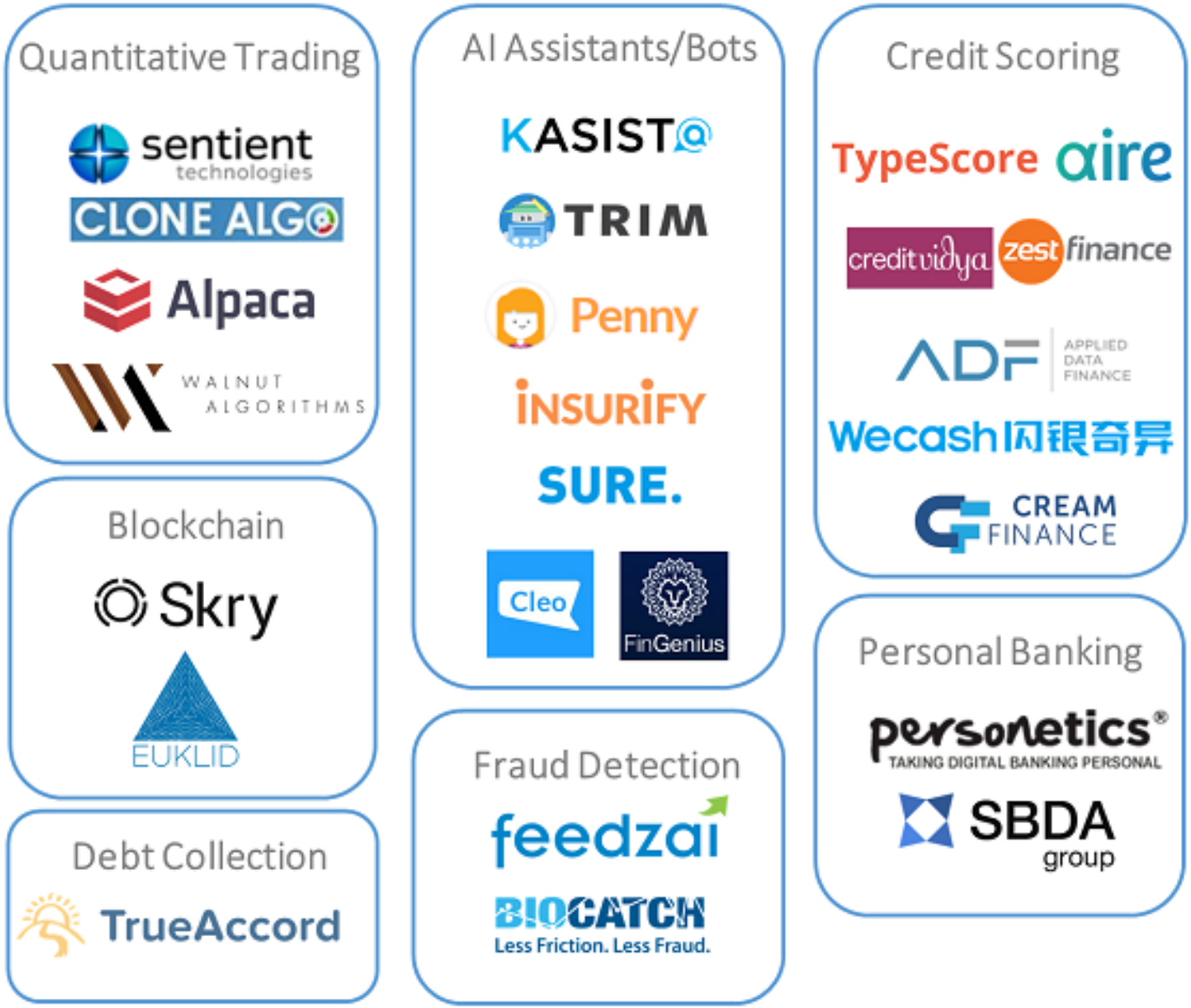
Source: https://www.cbinsights.com/blog/artificial-intelligence-fintech-market-map-company-list/

Artificial Intelligence (AI) in Fintech



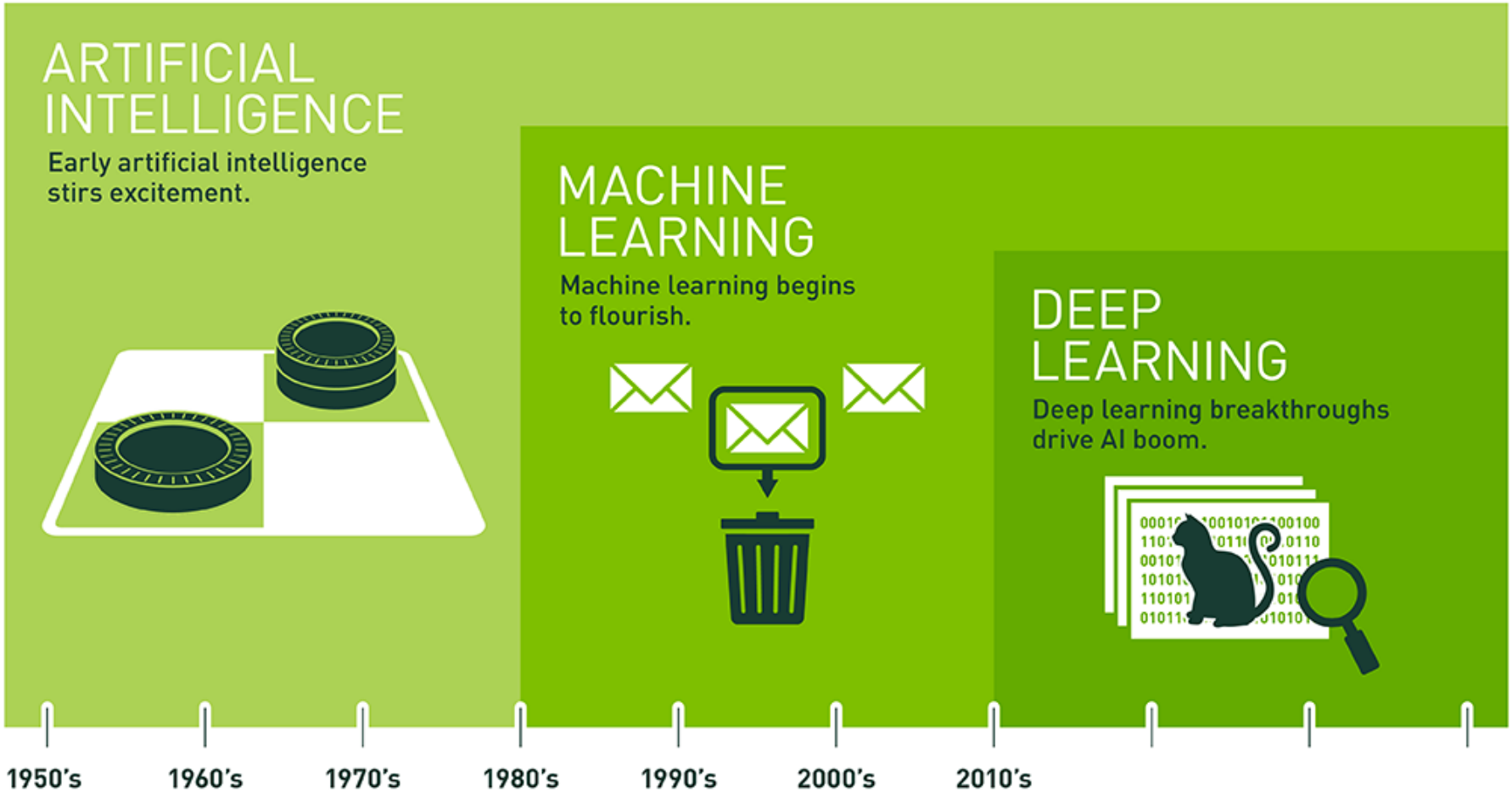
Source: https://www.cbinsights.com/blog/artificial-intelligence-fintech-market-map-company-list/

Artificial Intelligence (AI) in Fintech



Source: https://www.cbinsights.com/blog/artificial-intelligence-fintech-market-map-company-list/

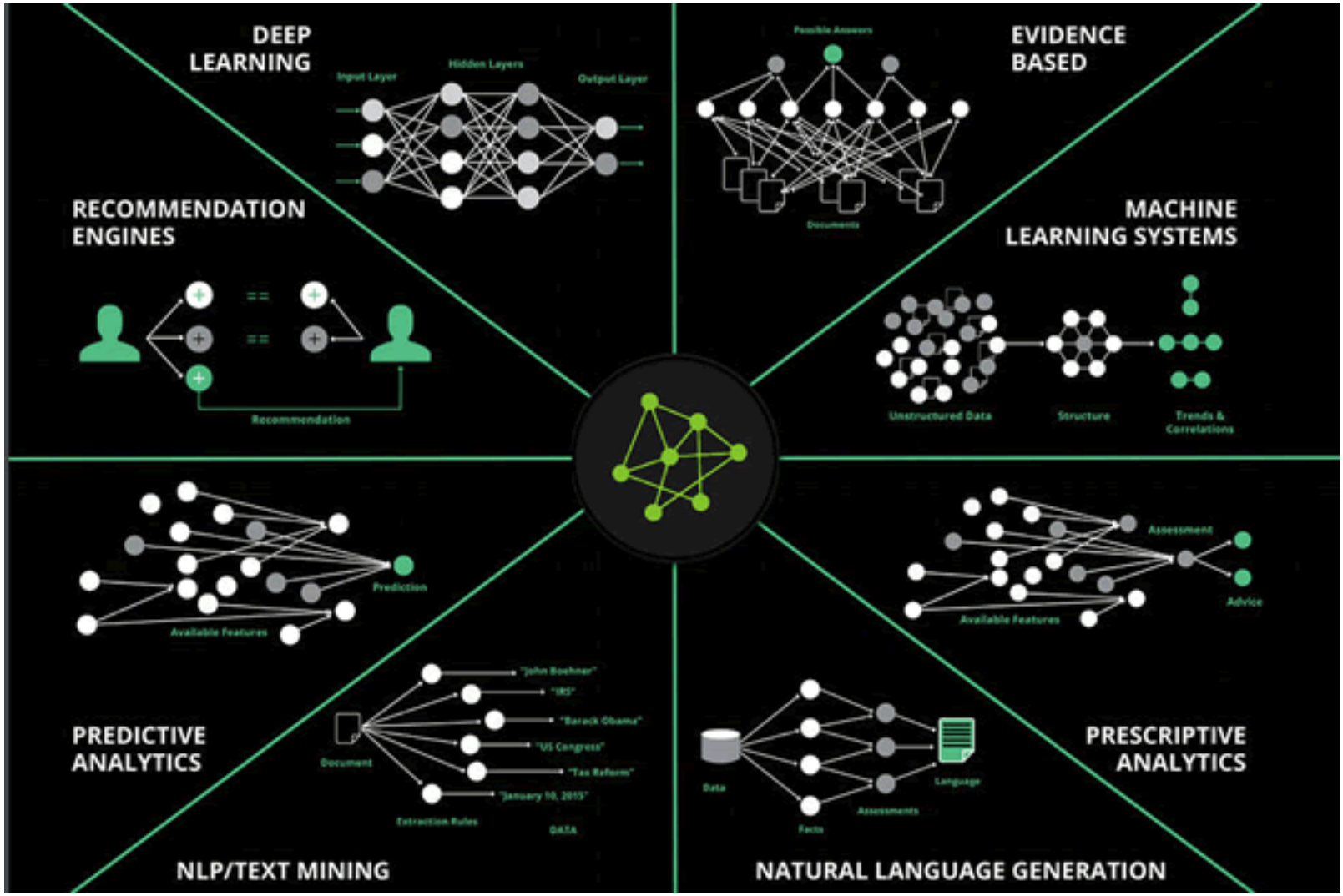
Artificial Intelligence
Machine Learning & Deep Learning



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

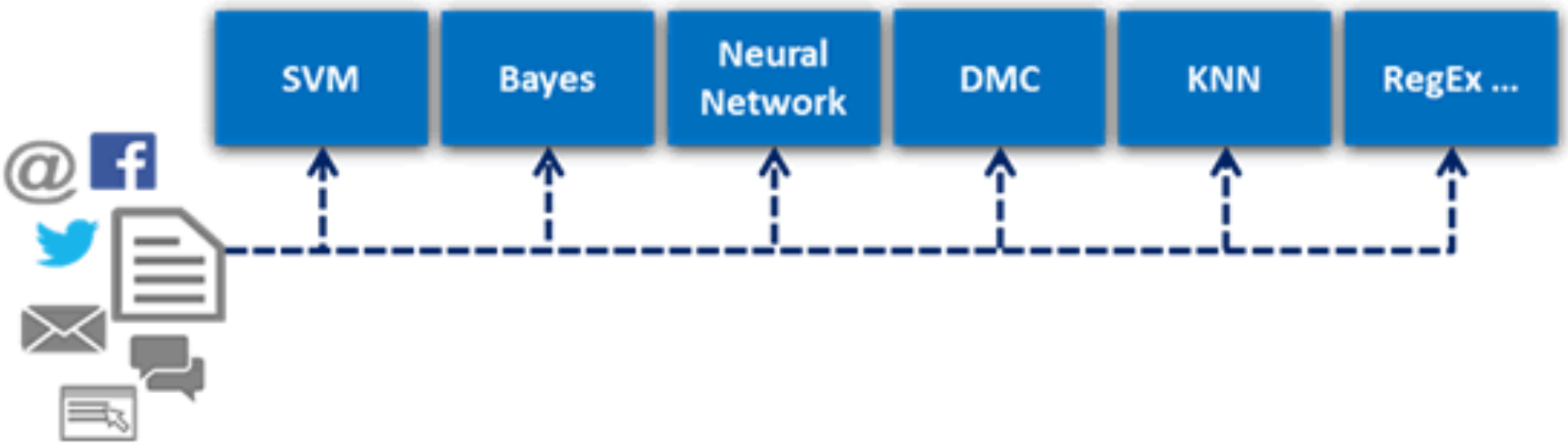


Ecosystem of AI

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

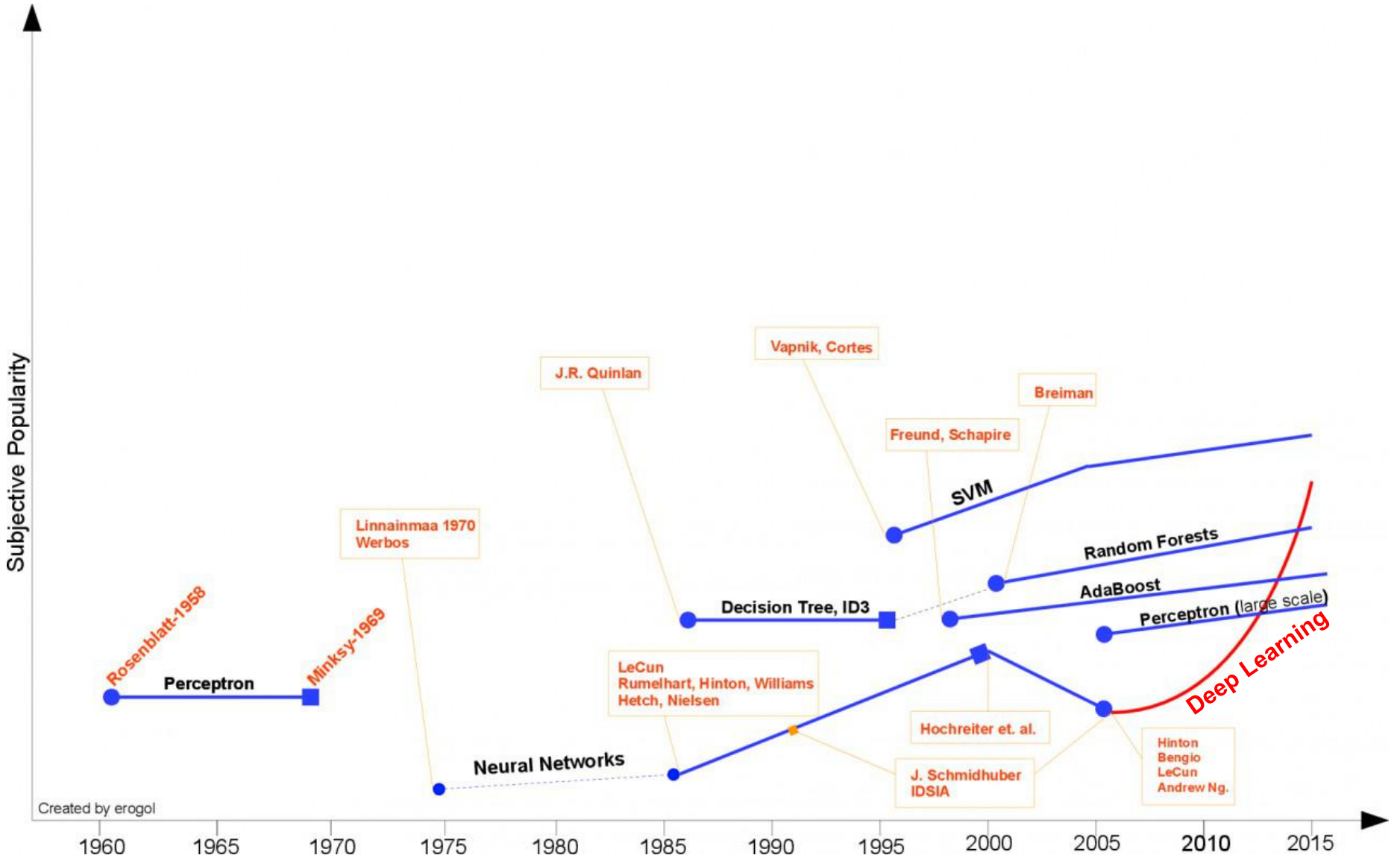
Artificial Intelligence (AI)

Intelligent Document Recognition algorithms



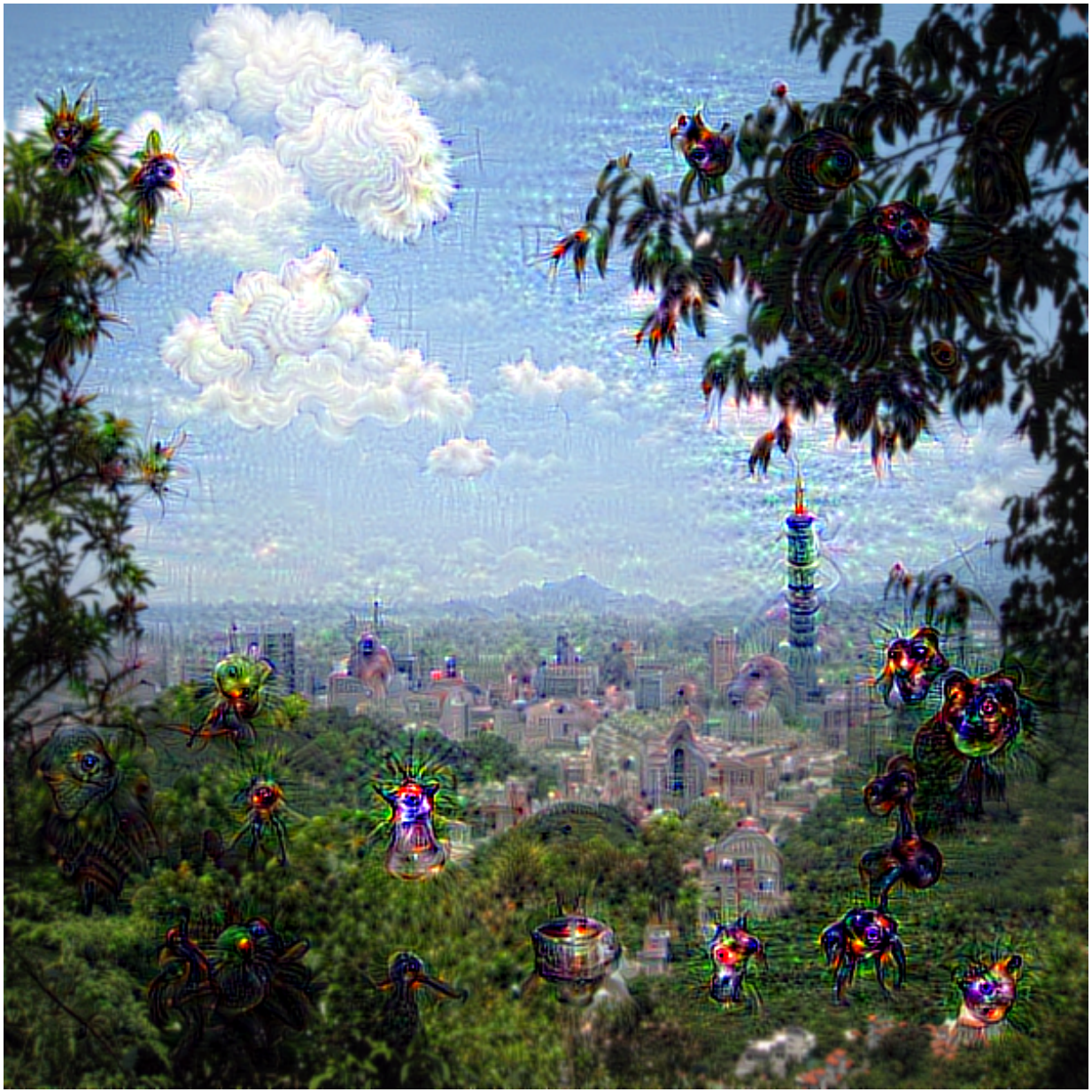
Source: https://www.i-scoop.eu/artificial-intelligence-cognitive-computing/

Deep Learning Evolution



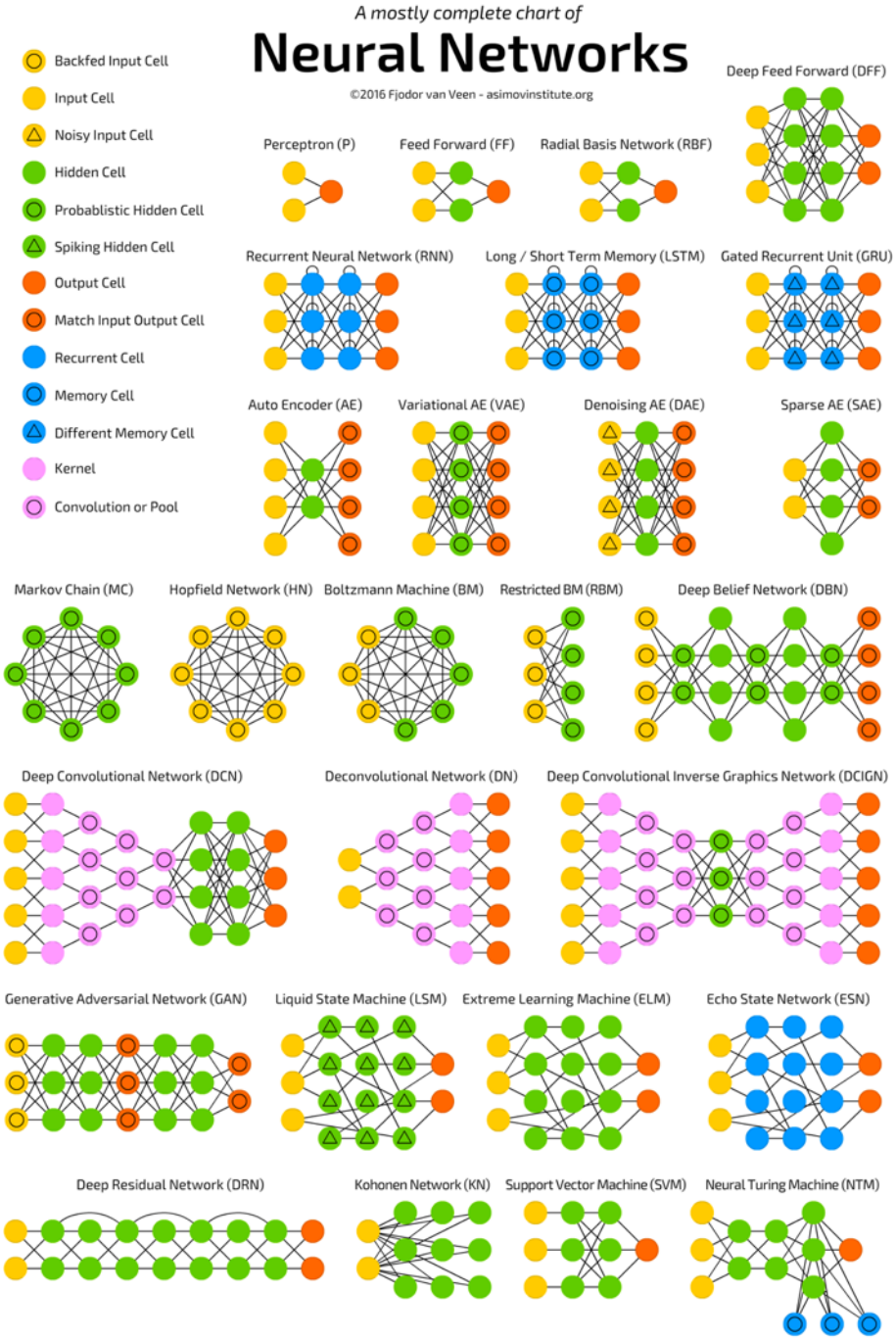
Source: http://www.erogol.com/brief-history-machine-learning/

Deep Dream



https://github.com/fchollet/keras/blob/master/examples/deep_dream.py

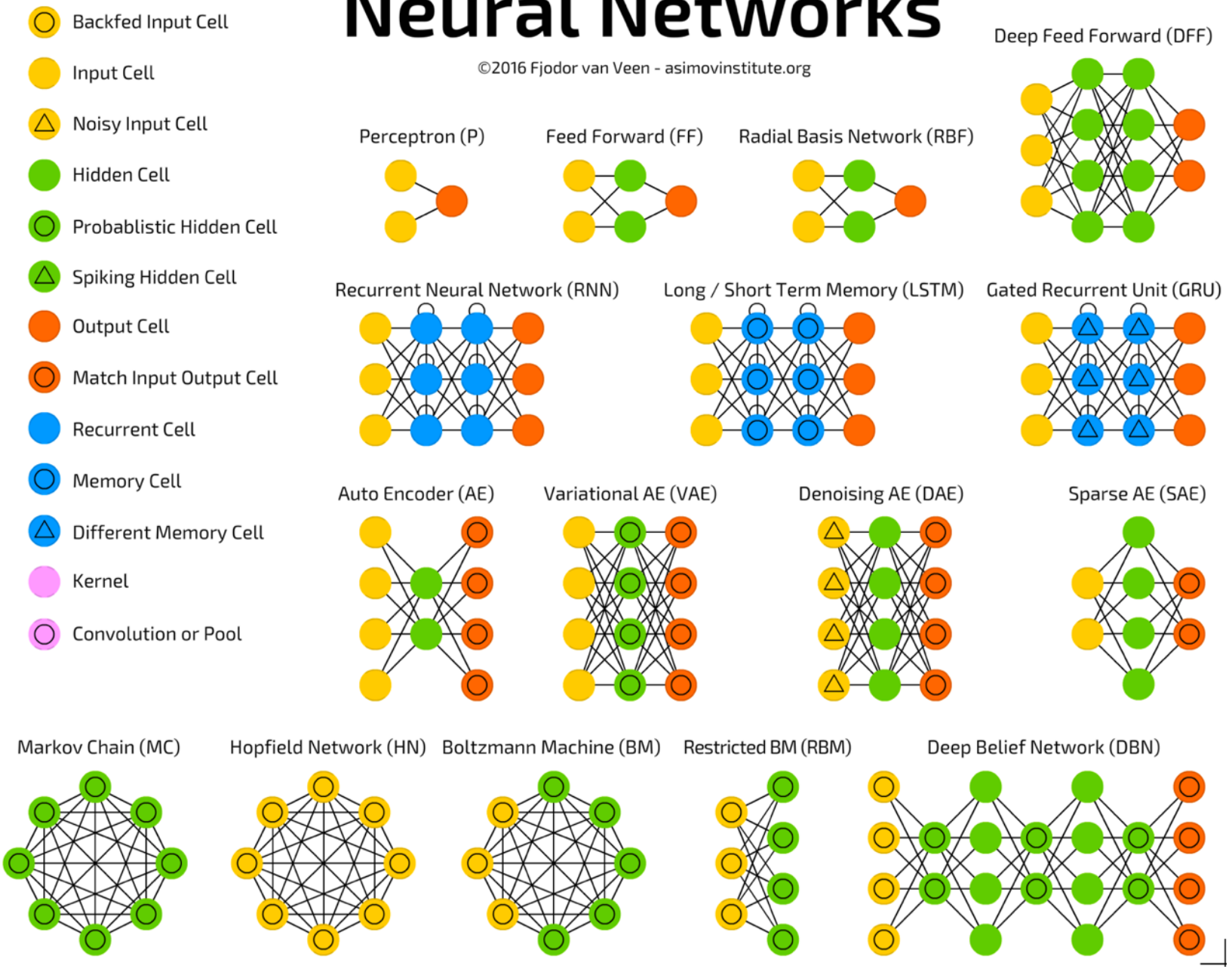
Neural Networks (NN)



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

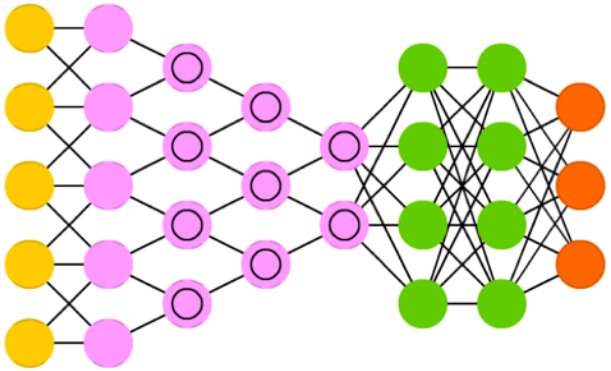
A mostly complete chart of

Neural Networks

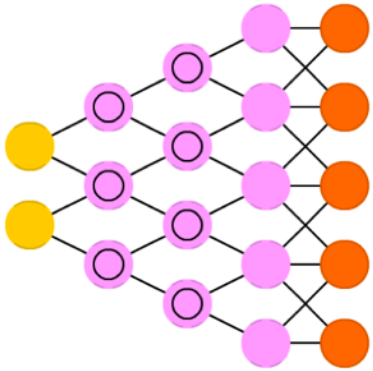


Source: http://www.asimovinstitute.org/neural-network-zoo/

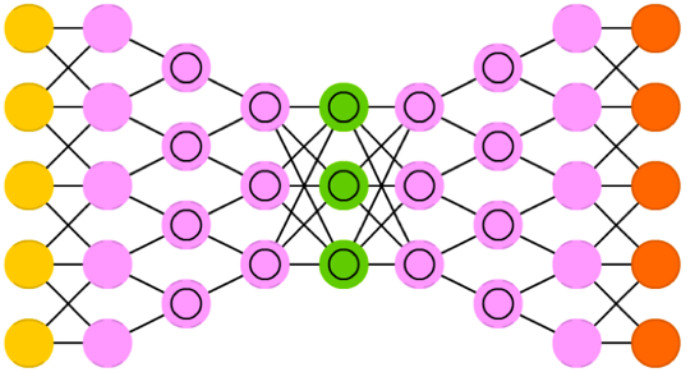
Deep Convolutional Network (DCN)



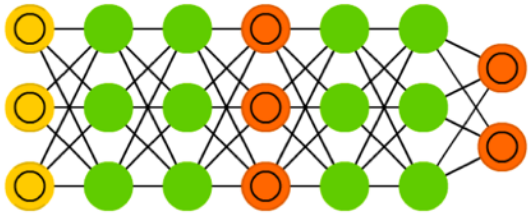
Deconvolutional Network (DN)



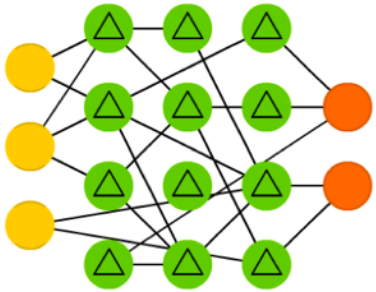
Deep Convolutional Inverse Graphics Network (DCIGN)



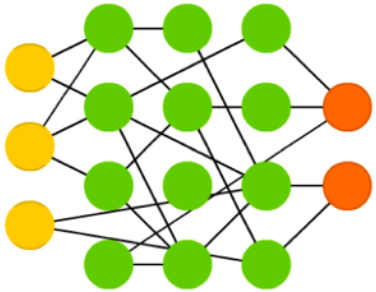
Generative Adversarial Network (GAN)



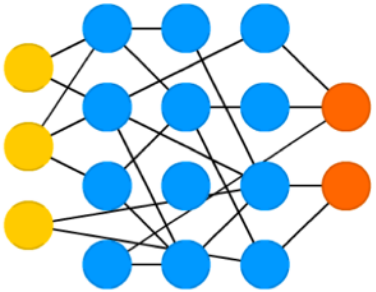
Liquid State Machine (LSM)



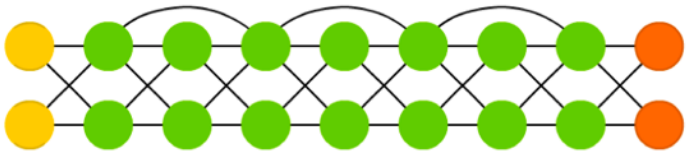
Extreme Learning Machine (ELM)



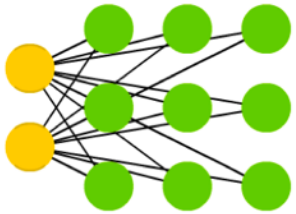
Echo State Network (ESN)



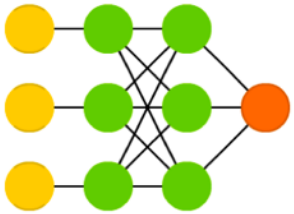
Deep Residual Network (DRN)



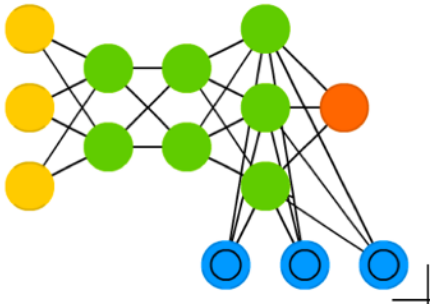
Kohonen Network (KN)



Support Vector Machine (SVM)



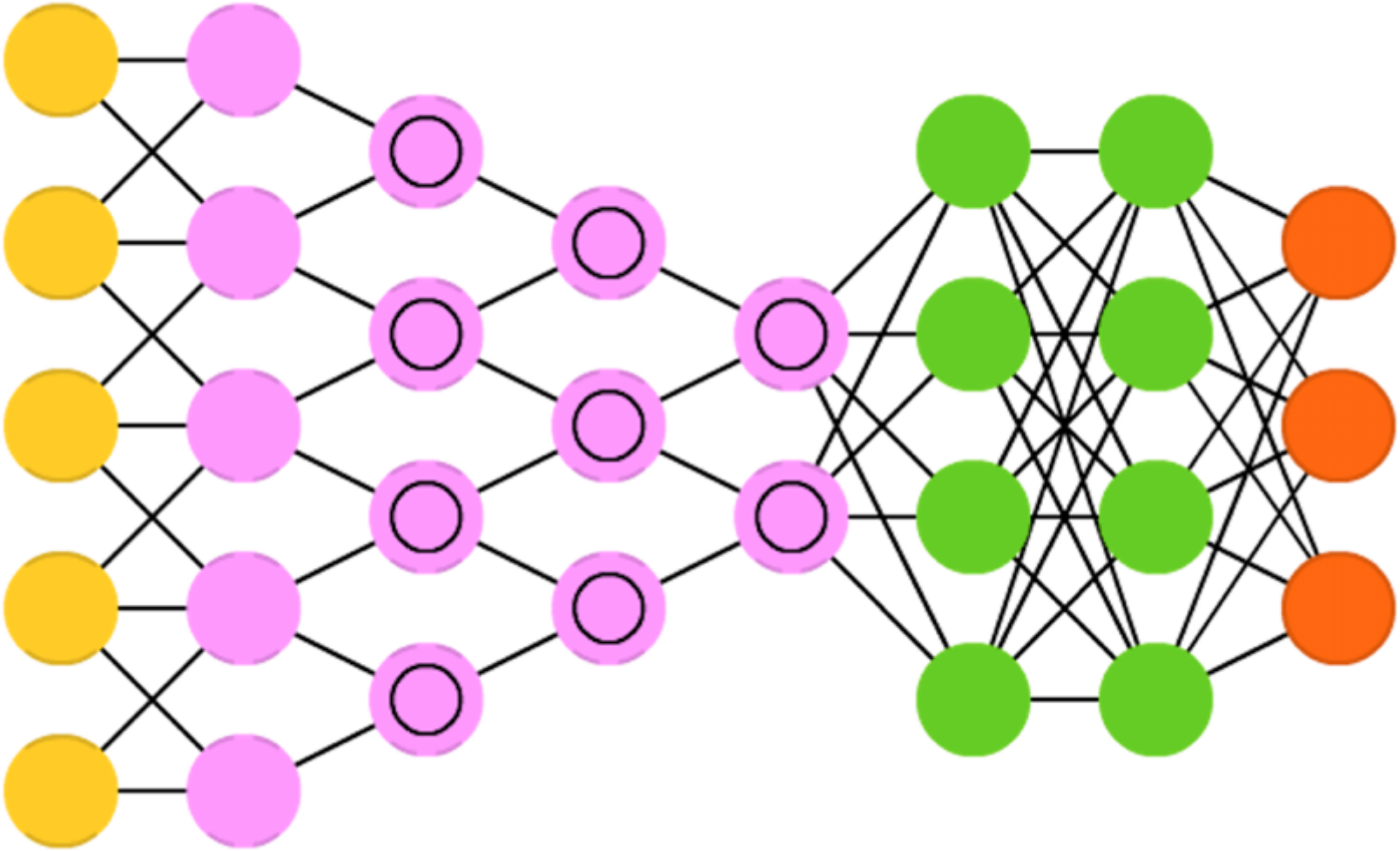
Neural Turing Machine (NTM)



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

Convolutional Neural Networks

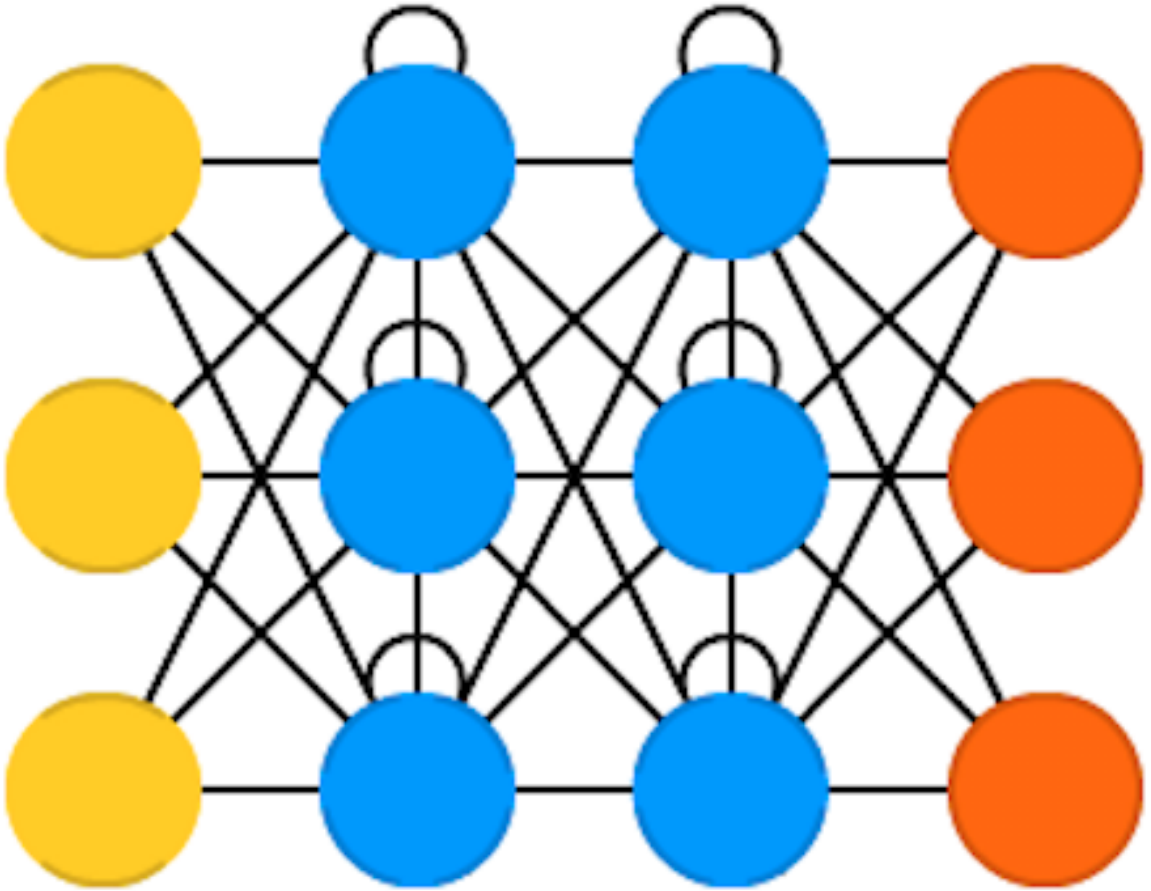
(CNN or Deep Convolutional Neural Networks, DCNN)



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/

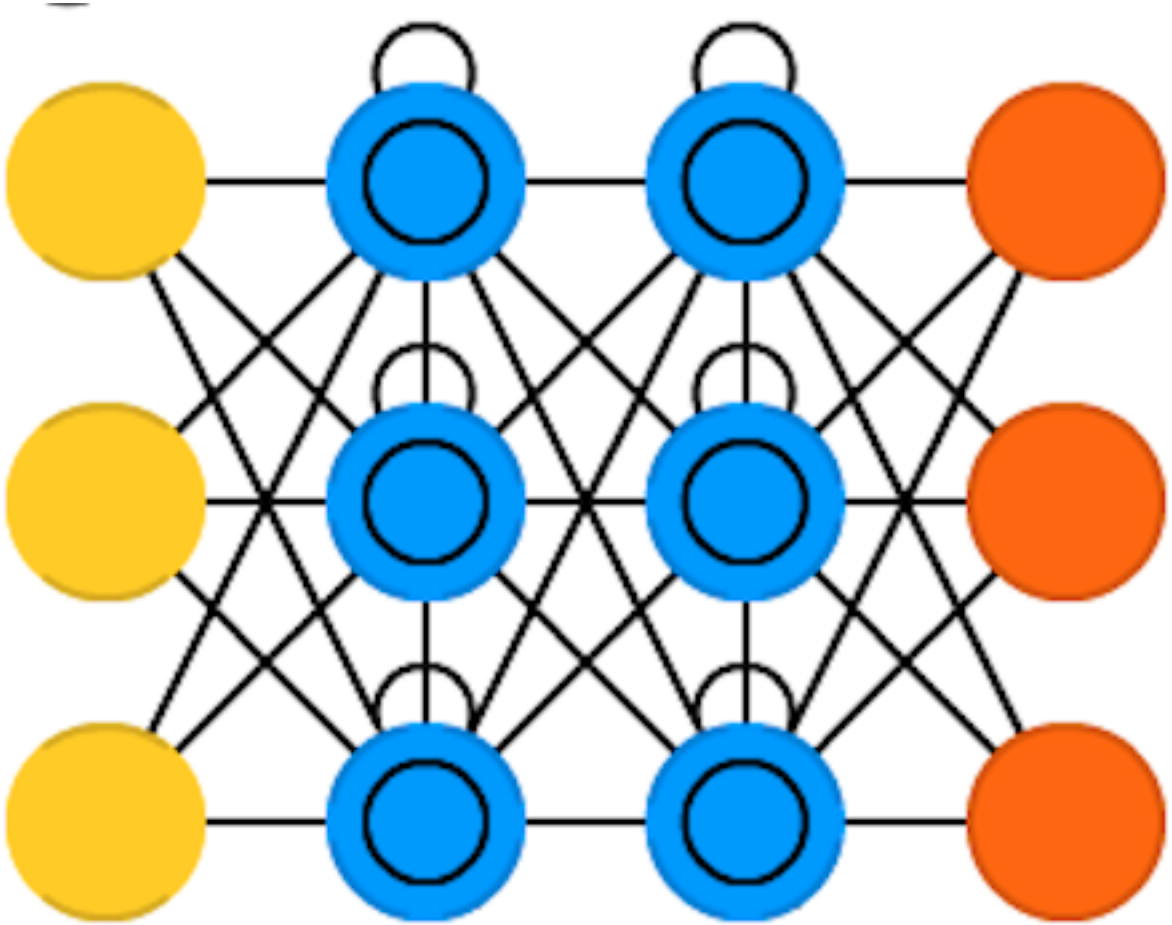
Recurrent Neural Networks (RNN)



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

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

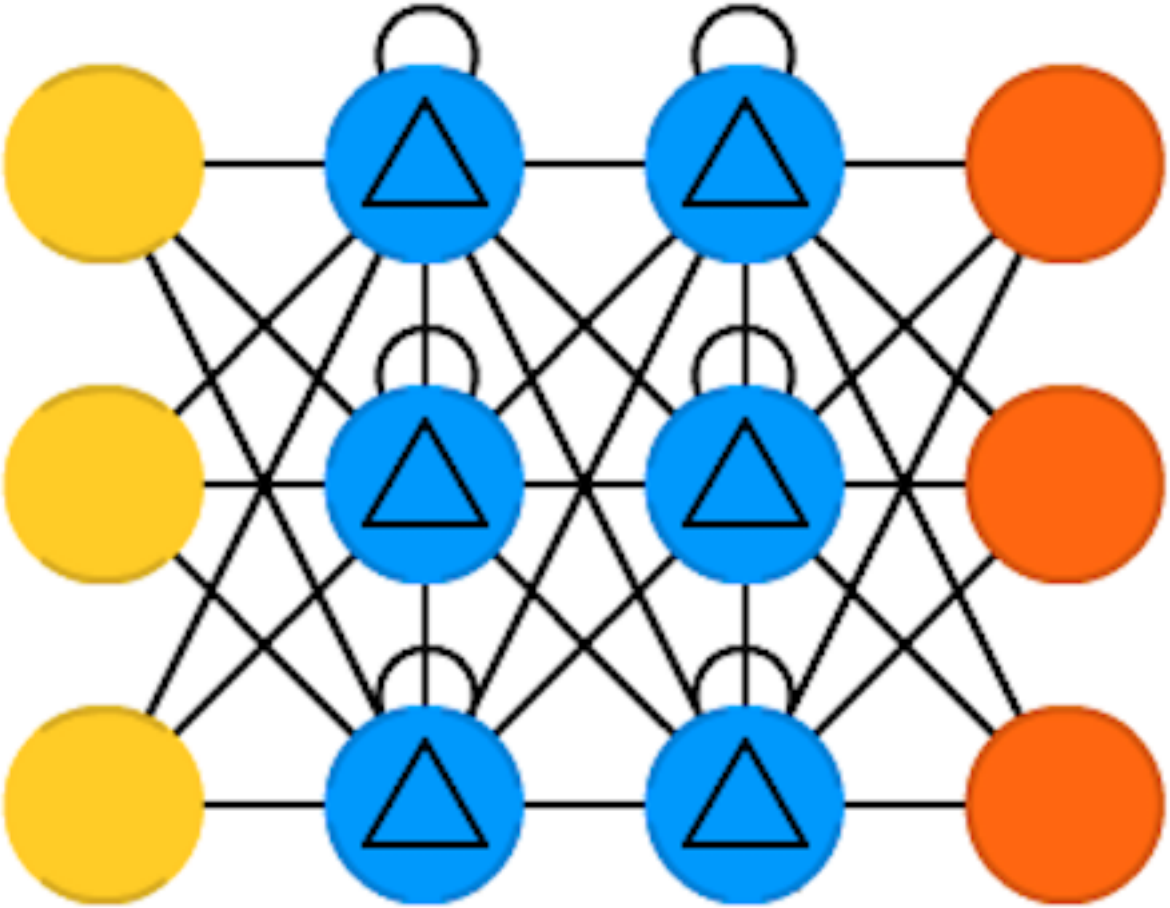
Long / Short Term Memory (LSTM)



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

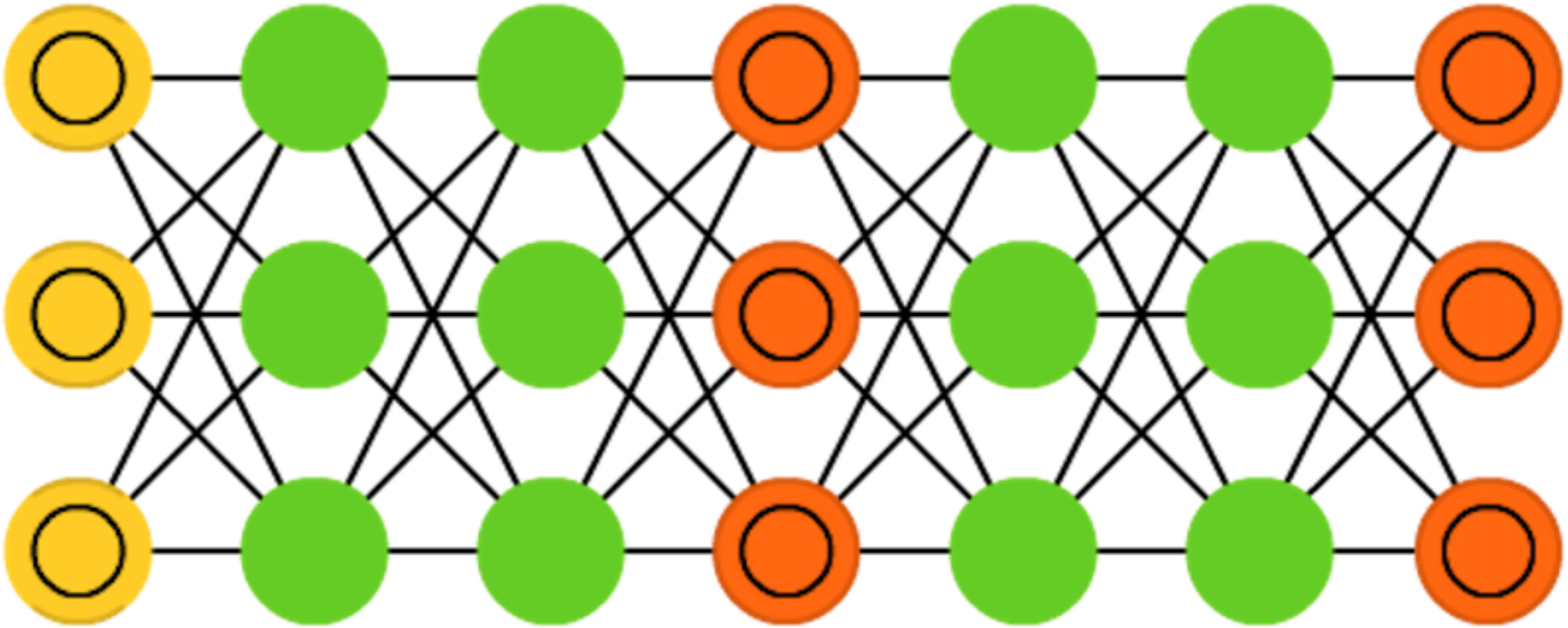
Source: http://www.asimovinstitute.org/neural-network-zoo/

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/

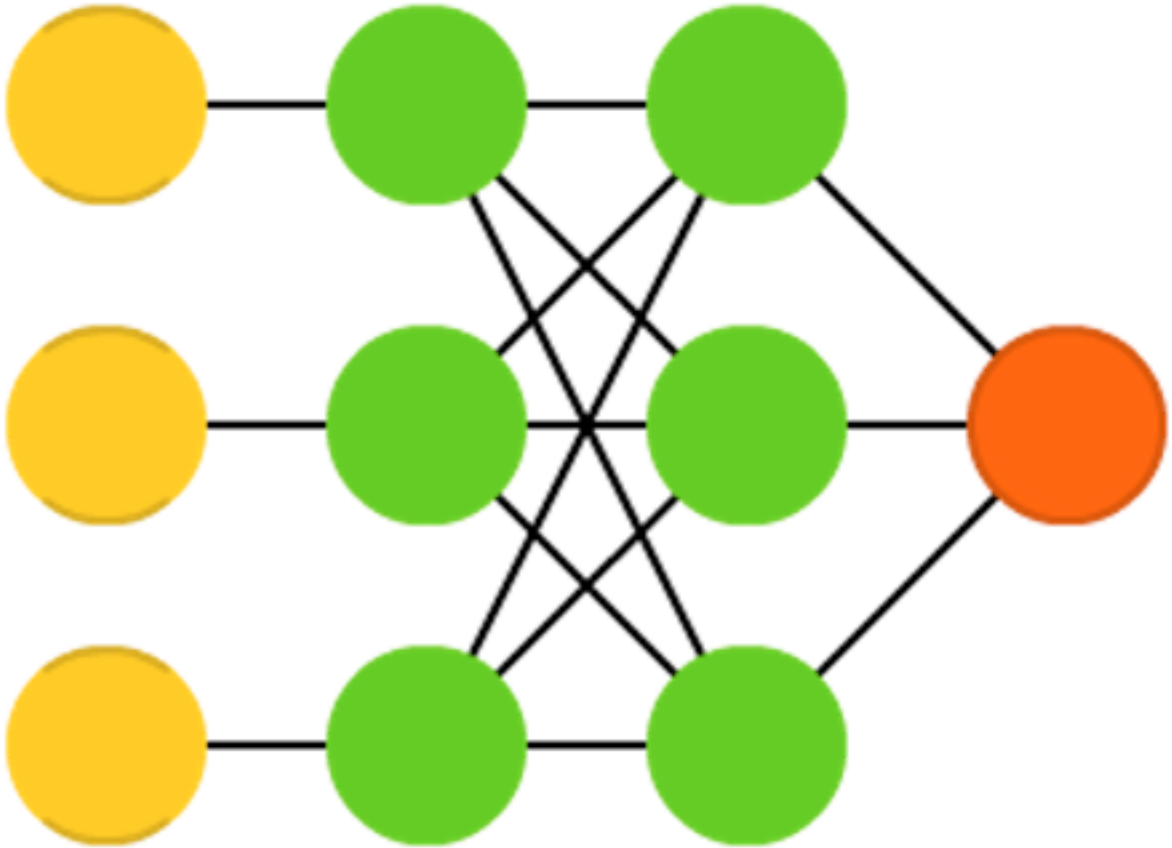
Generative Adversarial Networks (GAN)



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/

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.

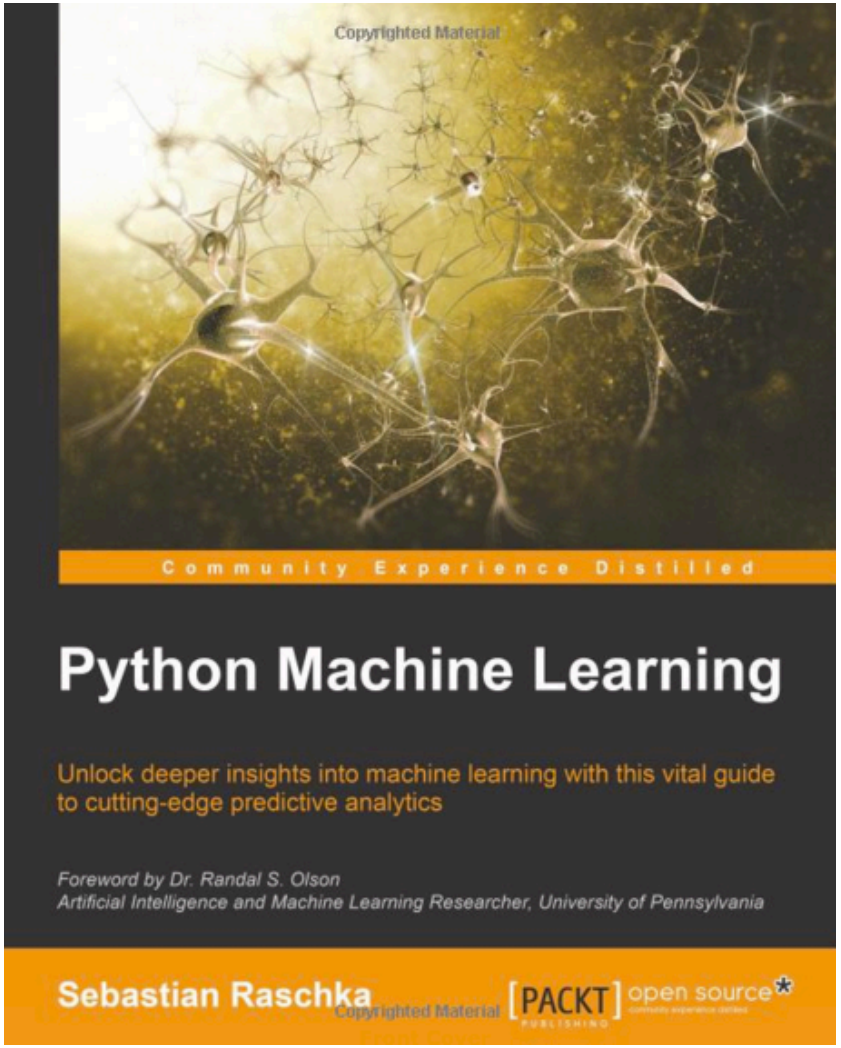
Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, 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.

Sebastian Raschka (2015),

Python Machine Learning,

Packt Publishing

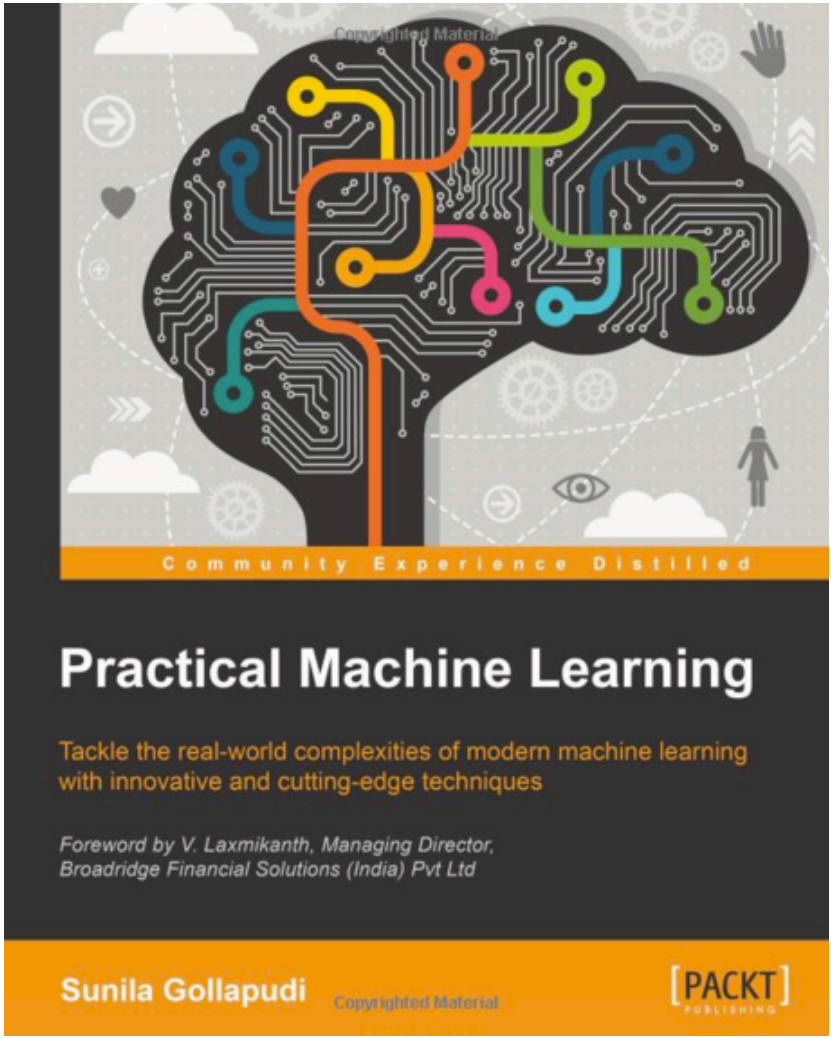


Source: http://www.amazon.com/Python-Machine-Learning-Sebastian-Raschka/dp/1783555130

Sunila Gollapudi (2016),

Practical Machine Learning,

Packt Publishing



Source: http://www.amazon.com/Practical-Machine-Learning-Sunila-Gollapudi/dp/178439968X

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

Neural networks (NN) 1960

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

Multilayer Perceptrons (MLP) 1985

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

Restricted Boltzmann Machine (RBM) 1986

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

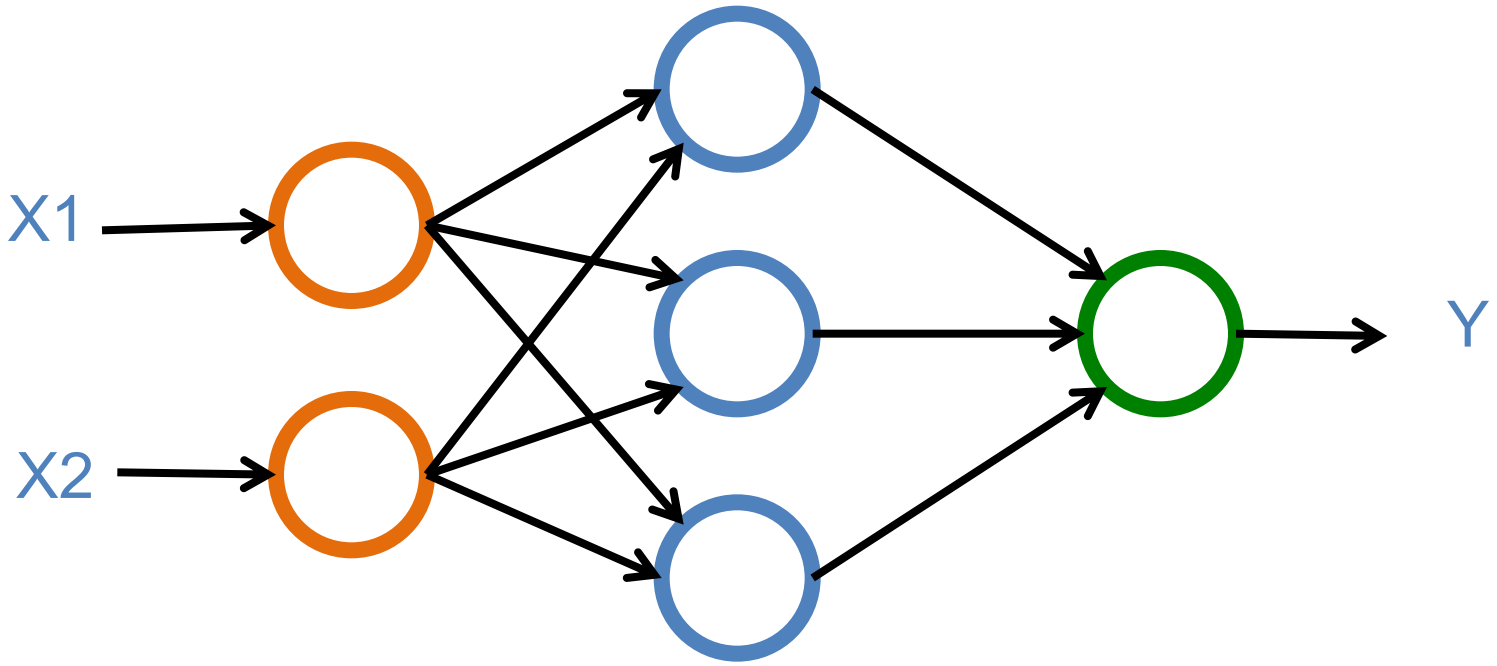
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Max-Pooling CDBN 2011

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

Neural Networks

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



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

Deep Learning

Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng

Geoffrey Hinton

Google

University of Toronto

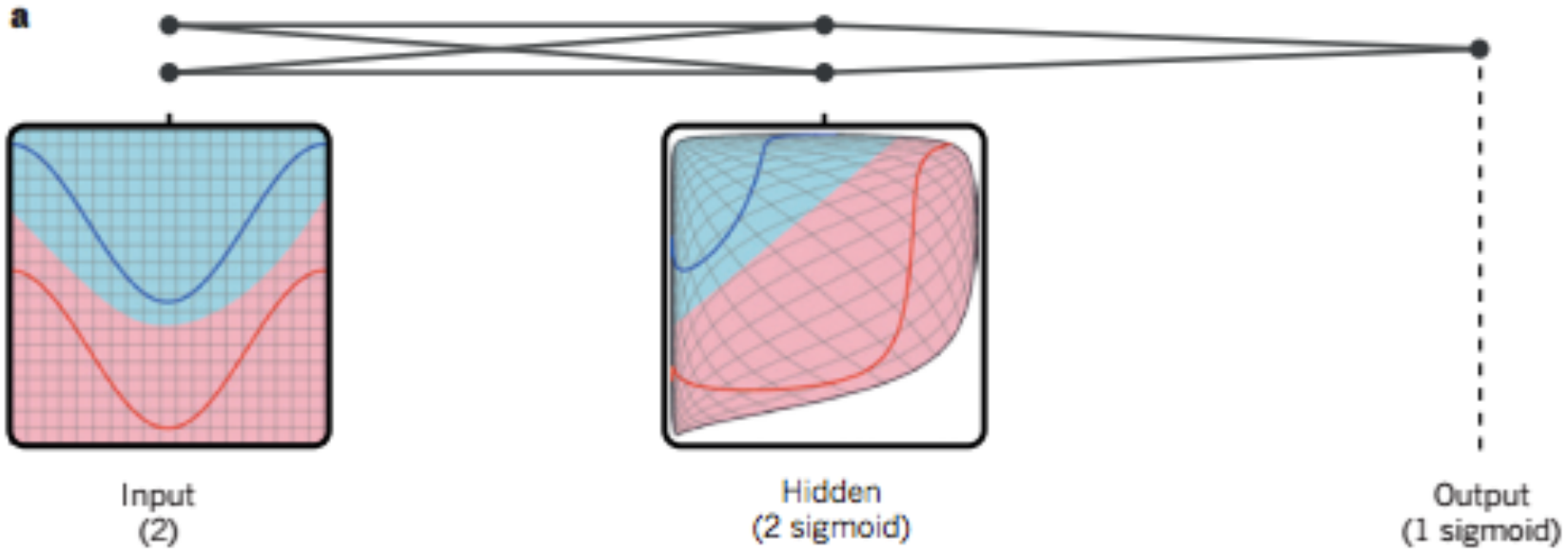
Source: https://en.wikipedia.org/wiki/Geoffrey_Hinton

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

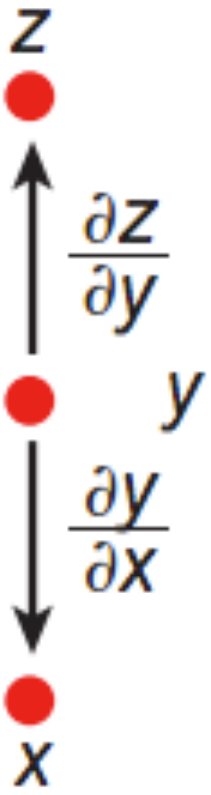
Nature 521, no. 7553 (2015): 436-444.

Deep Learning



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

Deep Learning



$$\Delta z = \frac{\partial z}{\partial y}\Delta y$$

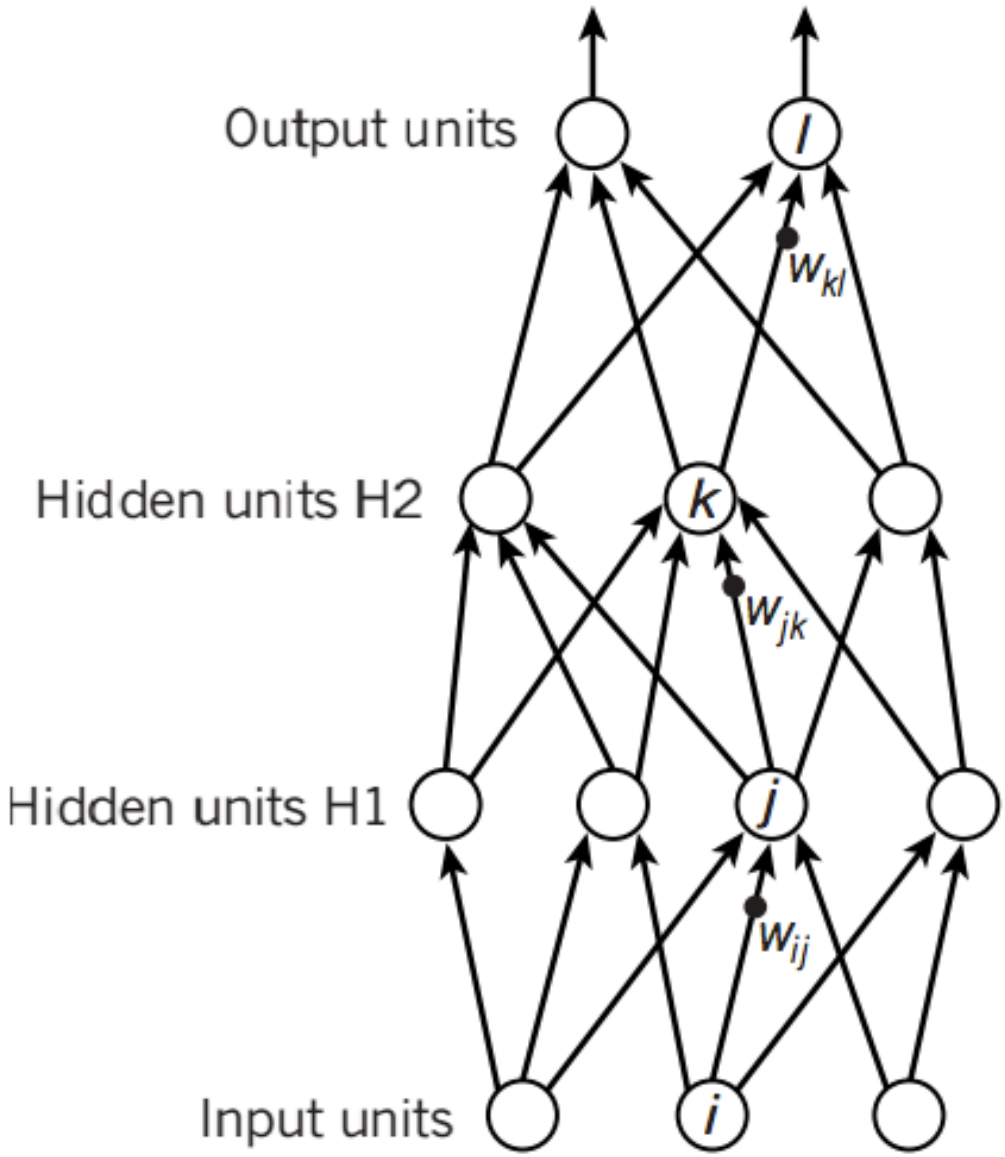
$$\Delta y = \frac{\partial y}{\partial x}\Delta x$$

$$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

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

Deep Learning



$$y_l = f(z_l)$$

$$z_l = \sum_{k \,\varepsilon\, \text{H2}} w_{kl}\, y_k$$

$$y_k = f(z_k)$$

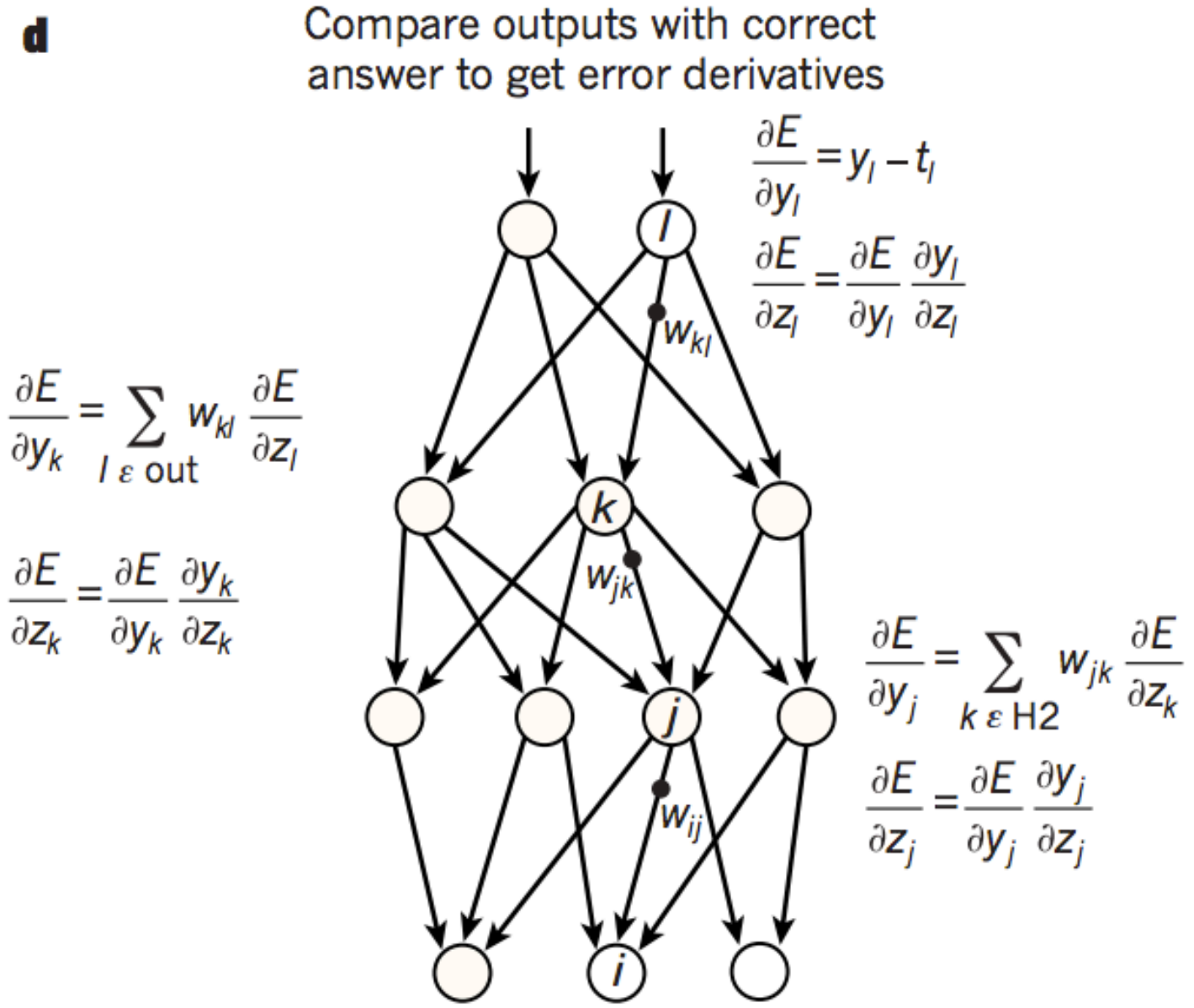
$$z_k = \sum_{j \,\varepsilon\, \text{H1}} w_{jk}\, y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \,\varepsilon\, \text{Input}} w_{ij}\, x_i$$

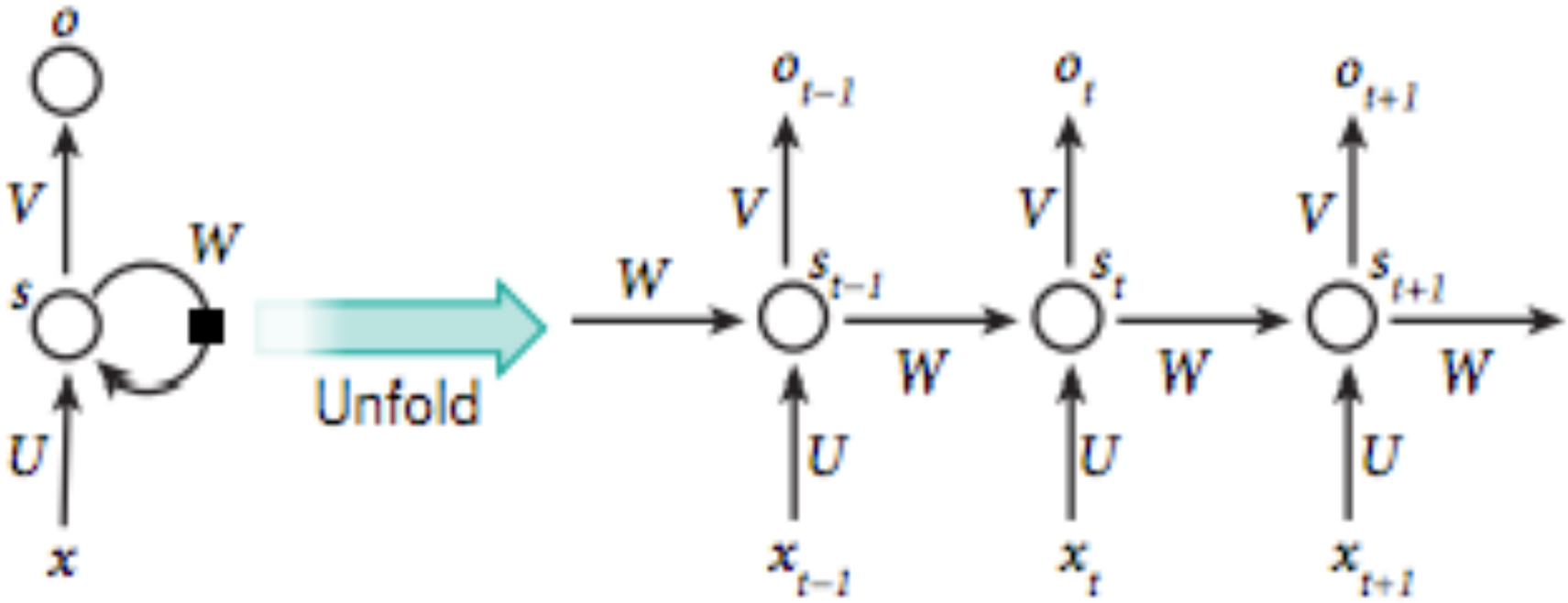
Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

Deep Learning



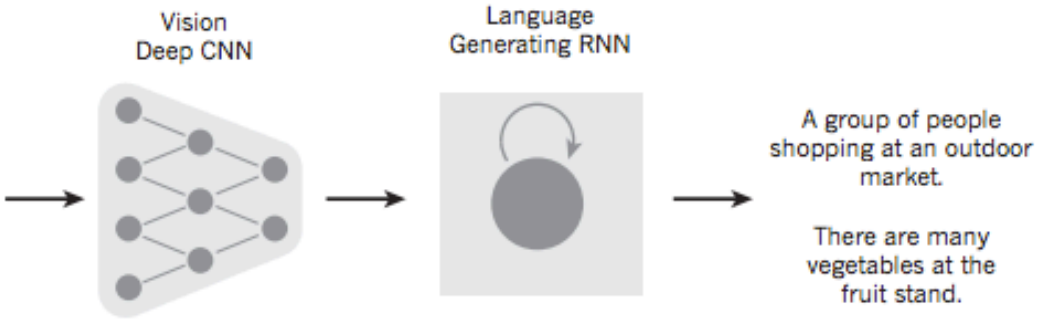
Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

Recurrent Neural Network (RNN)



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

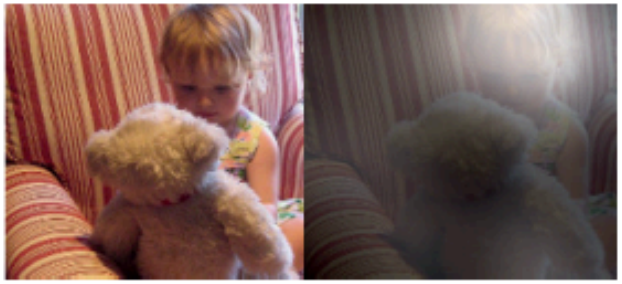
From image to text



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)

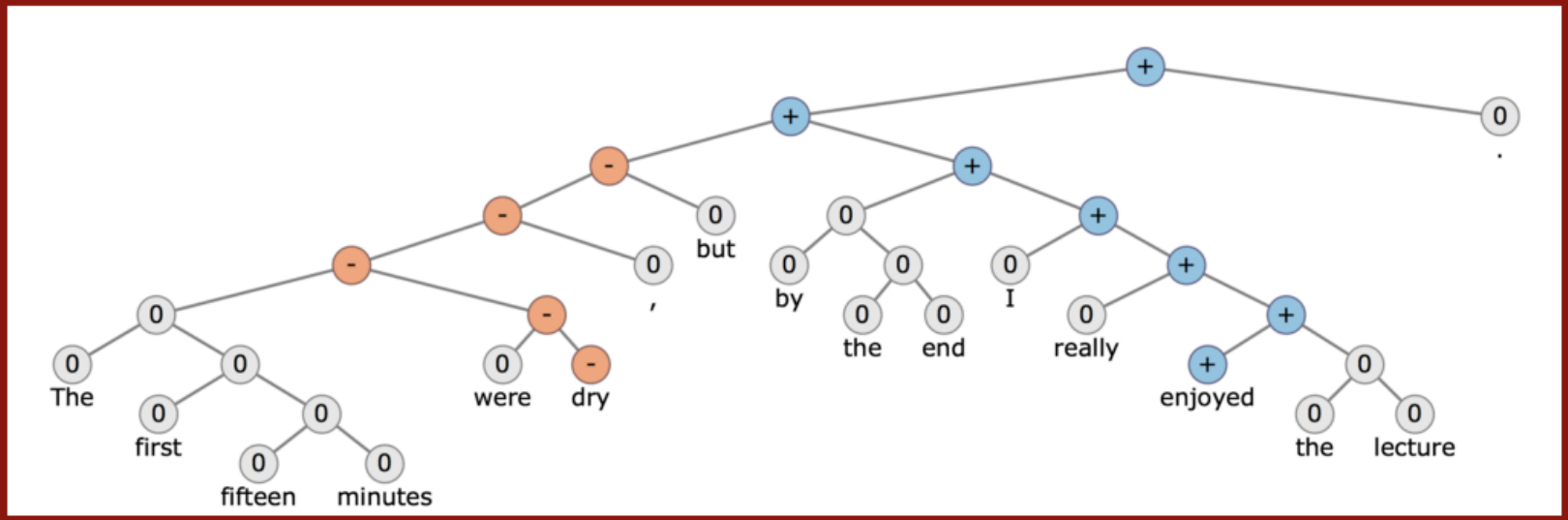
Text: recurrent neural network (RNN)

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.

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

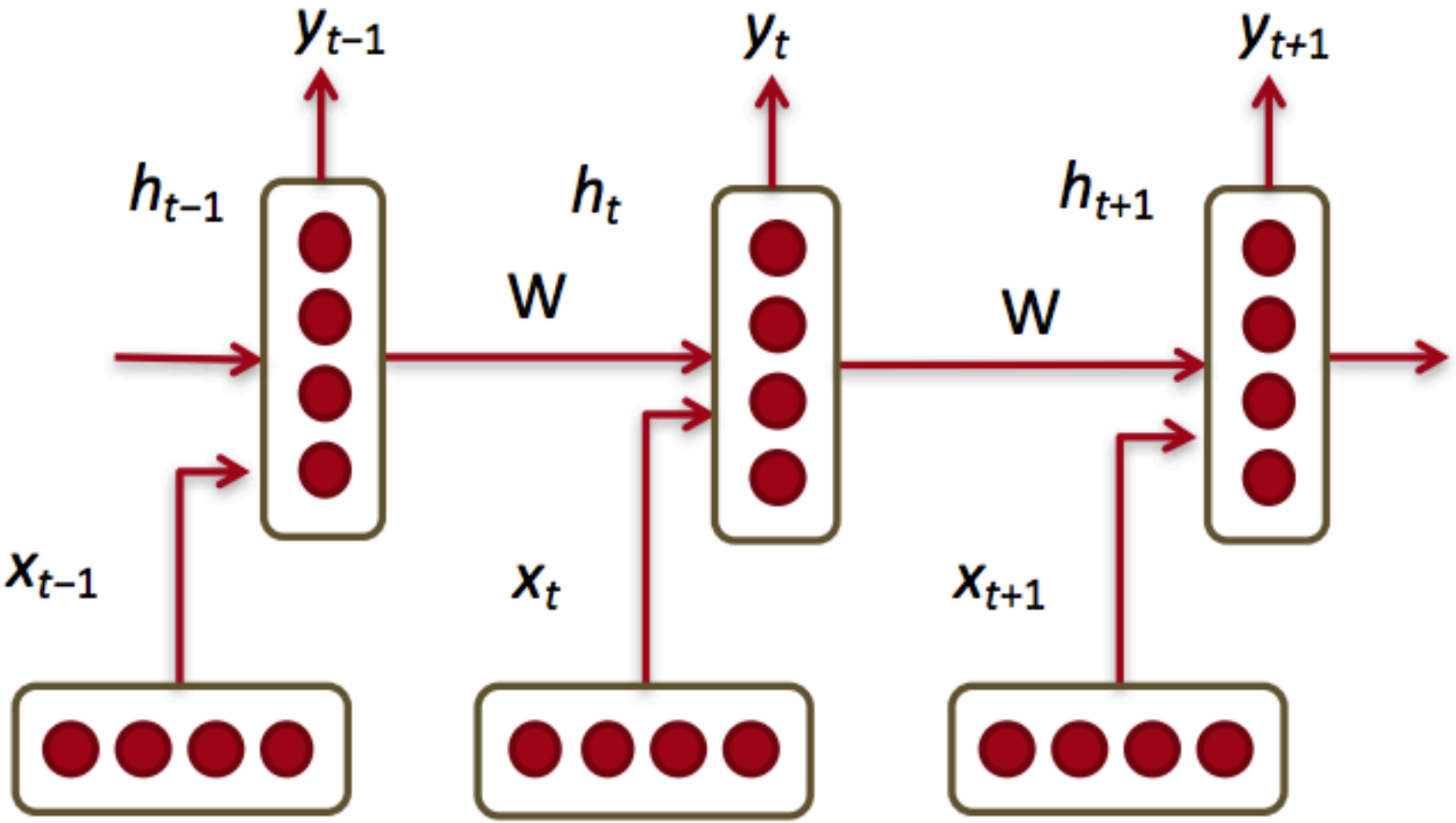


Course Description

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,

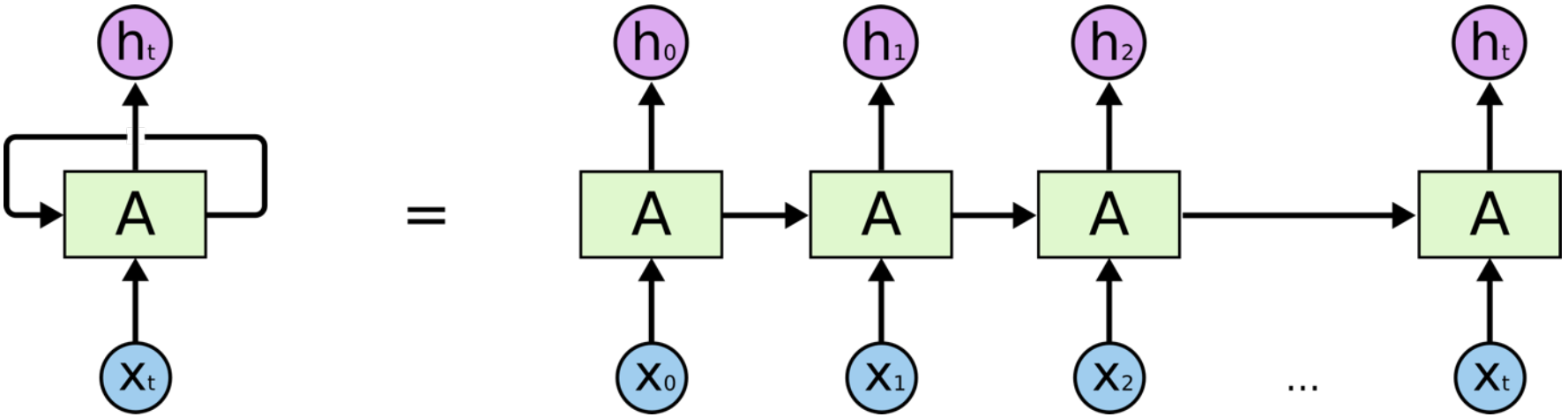
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



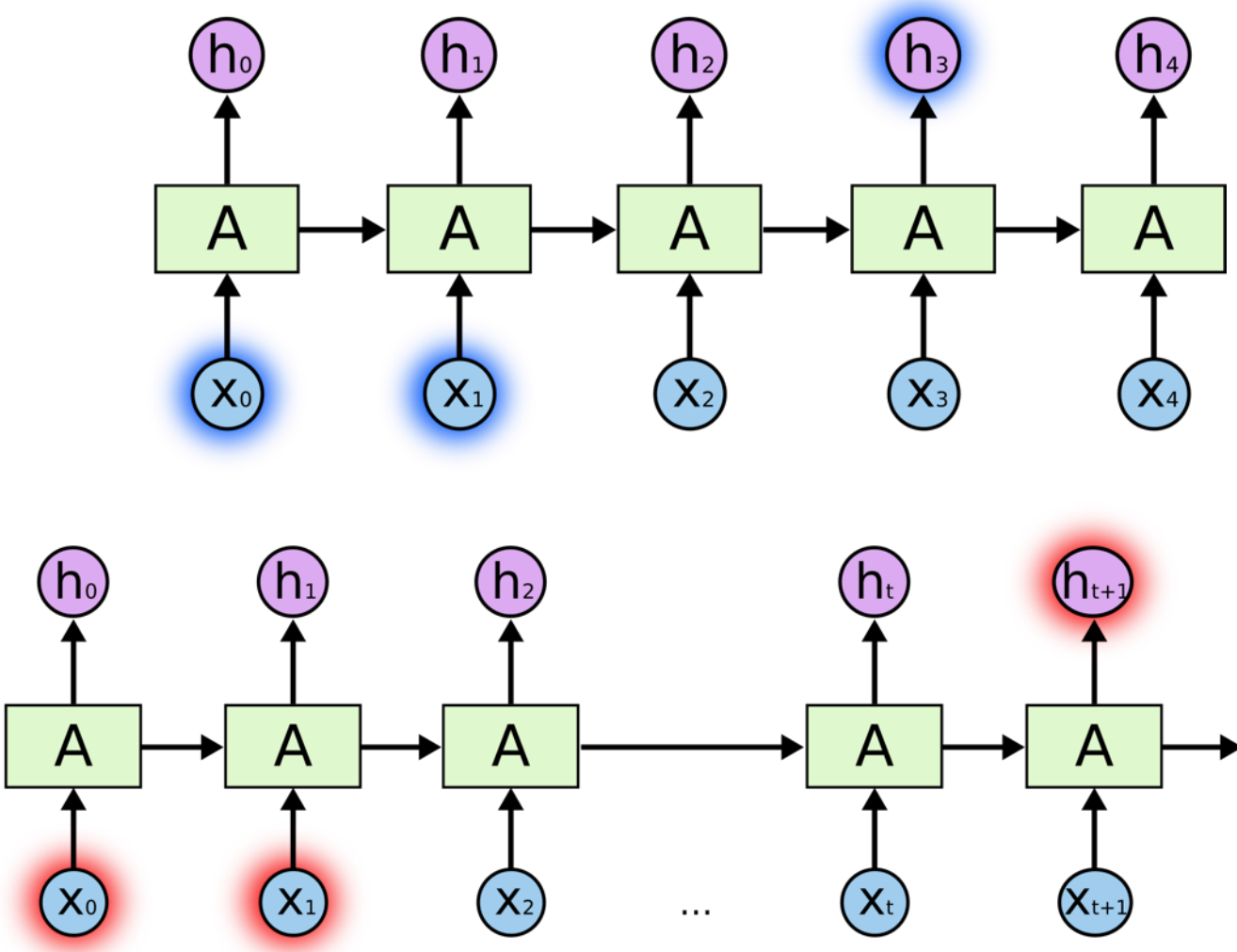
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN



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

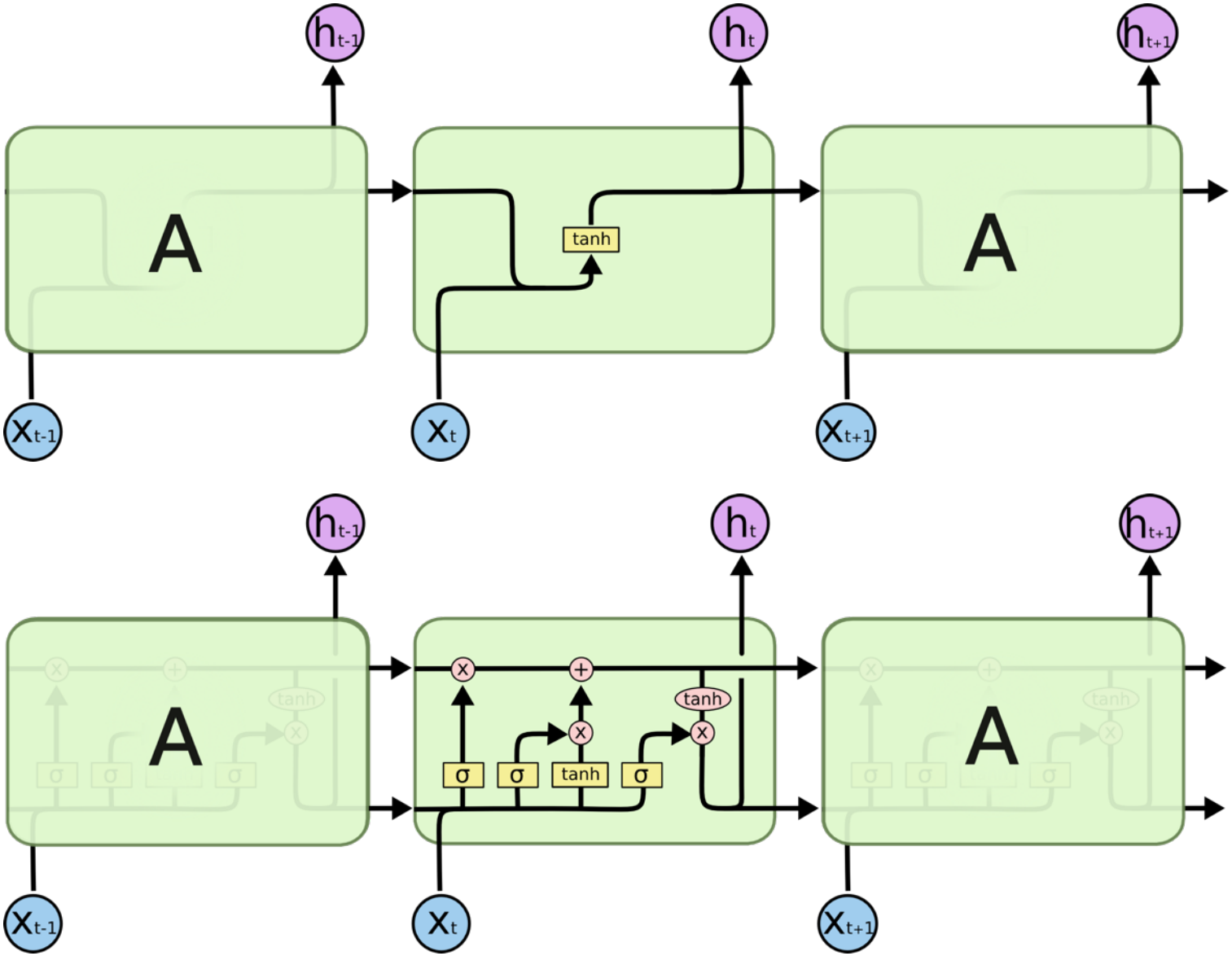
RNN long-term dependencies



I grew up in France… I speak fluent French.

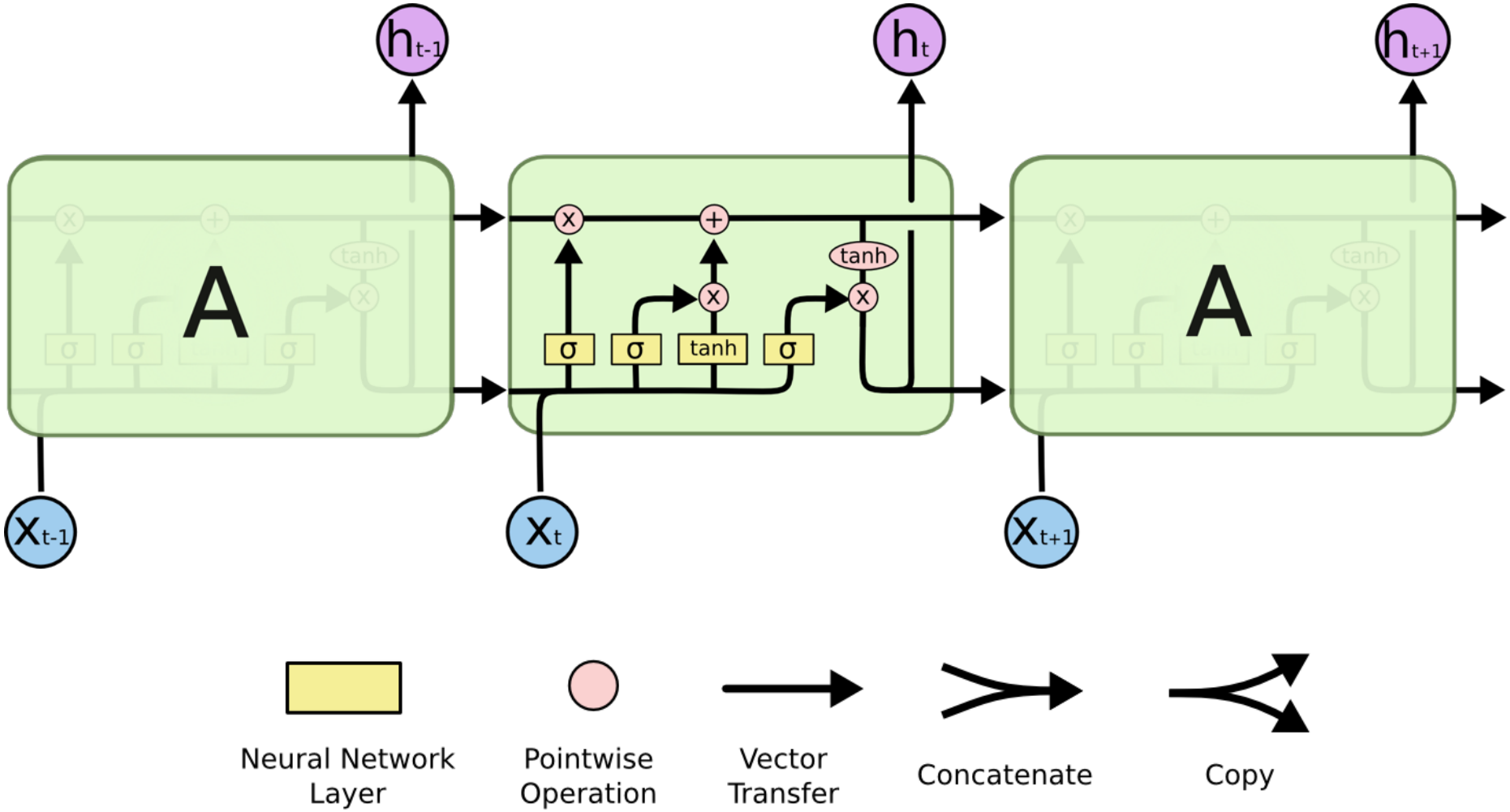
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN LSTM



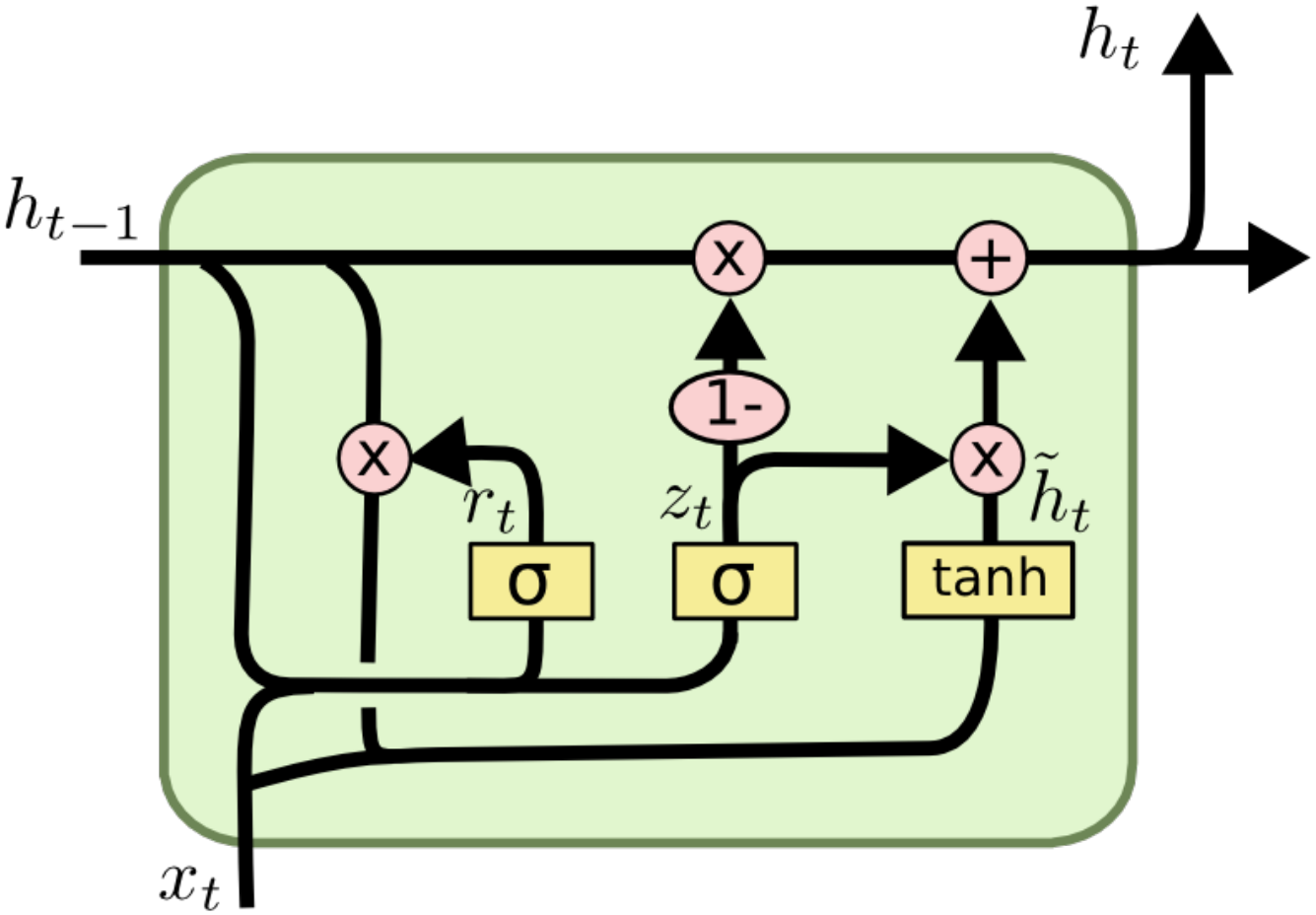
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



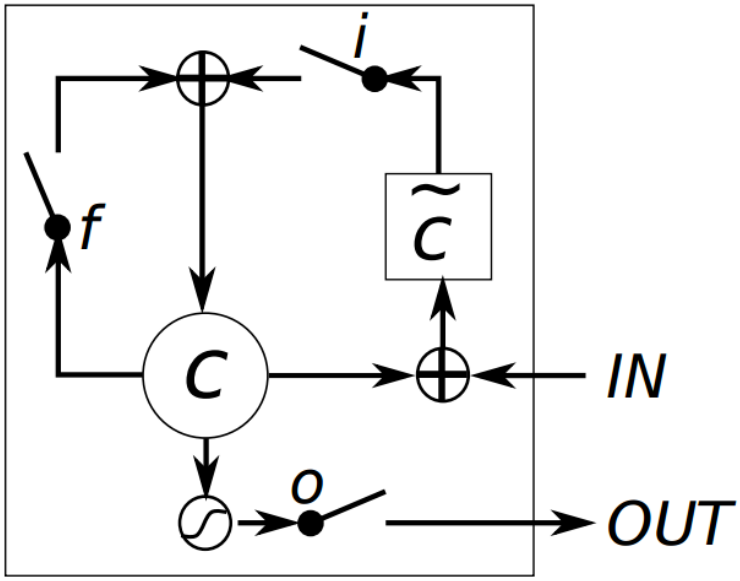
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

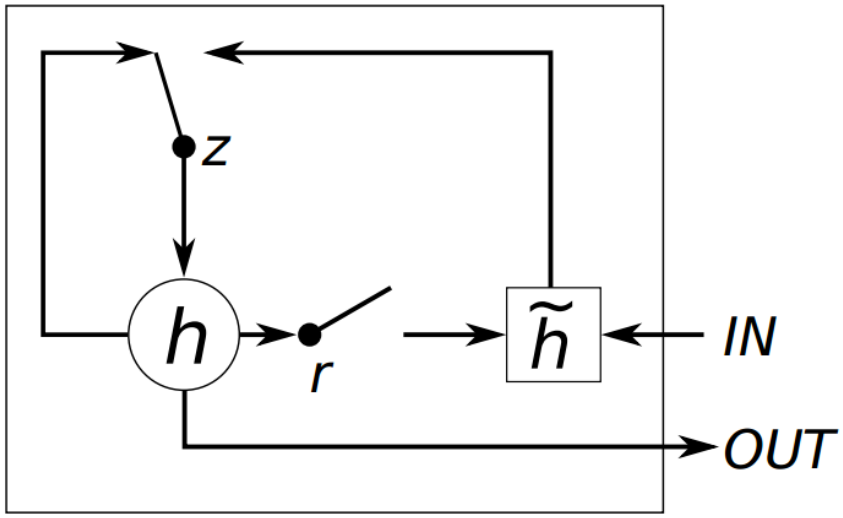
Gated Recurrent Unit (GRU)



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

LSTM vs GRU





LSTM

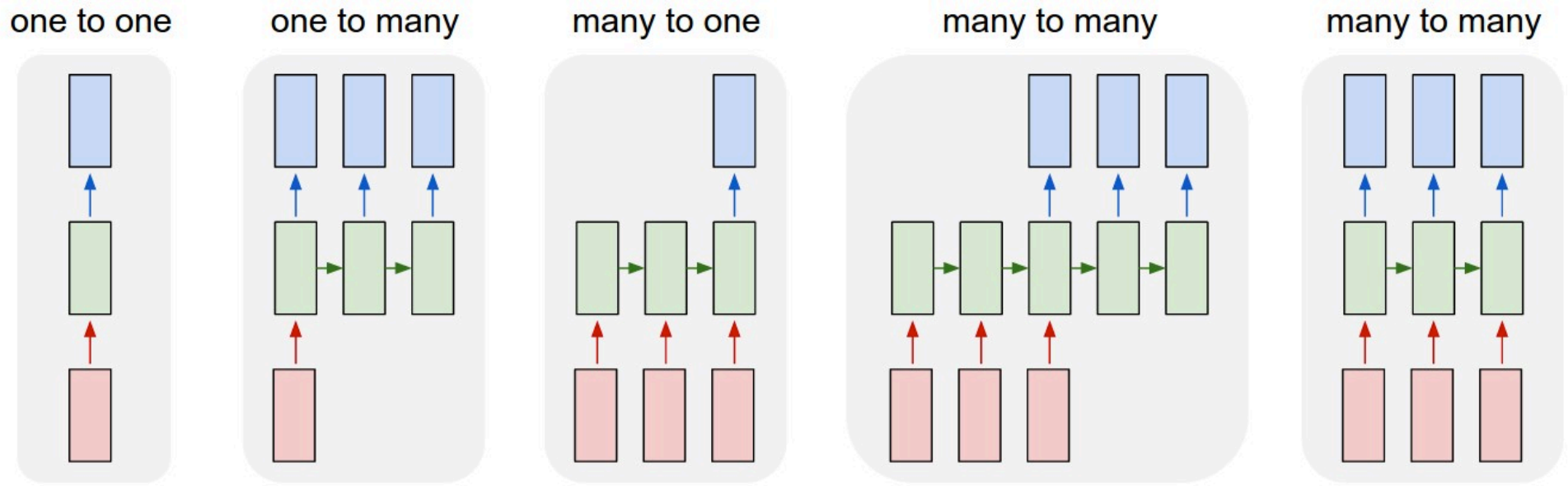
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

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

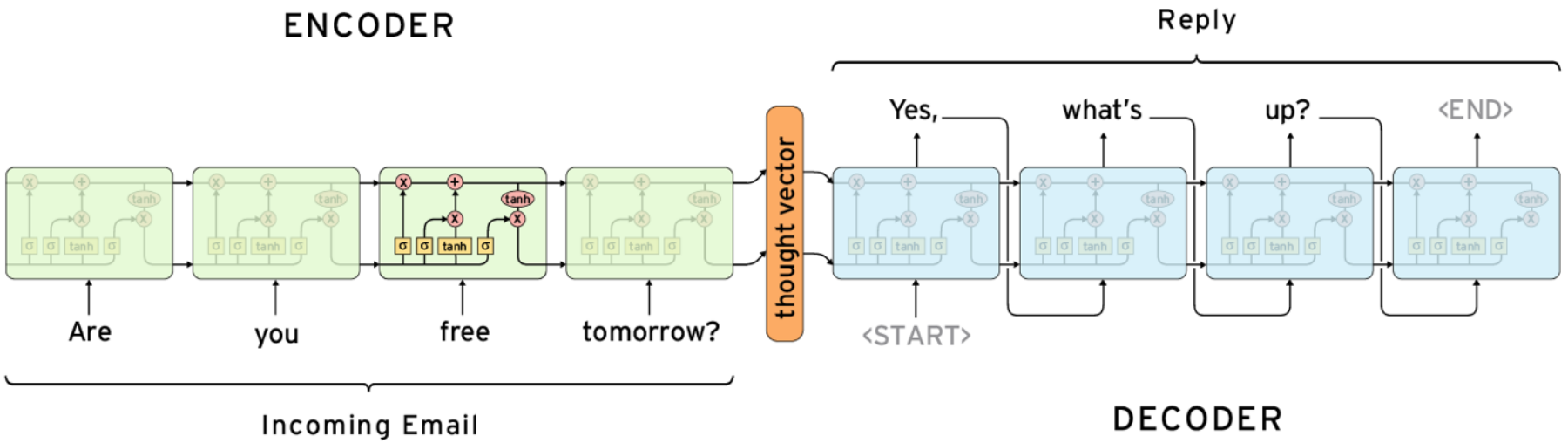
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

LSTM Recurrent Neural Network



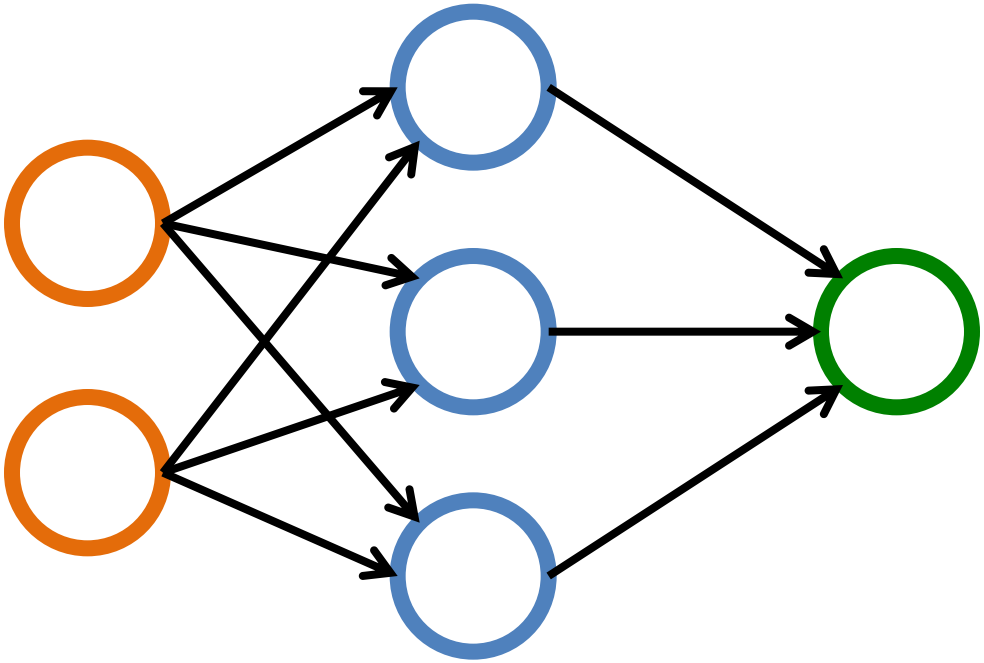
Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

The Sequence to Sequence model (seq2seq)



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

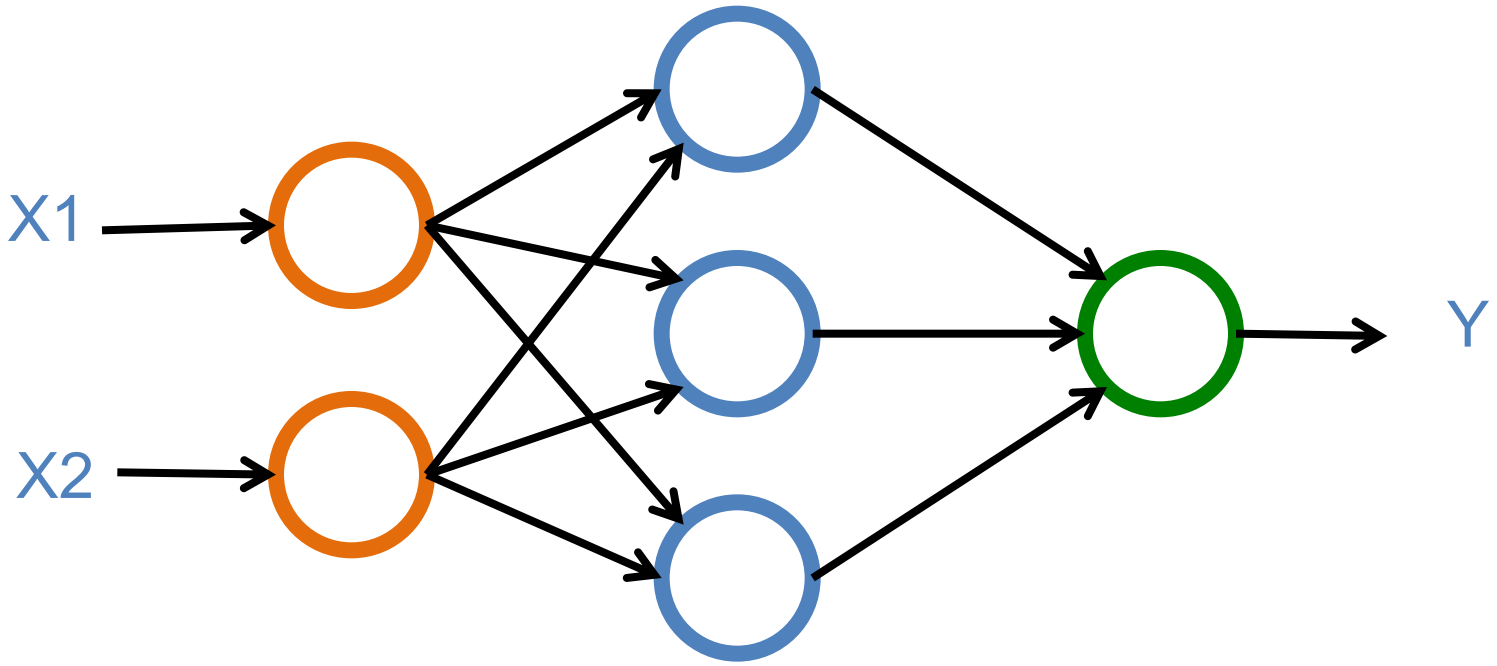
Neural Networks



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

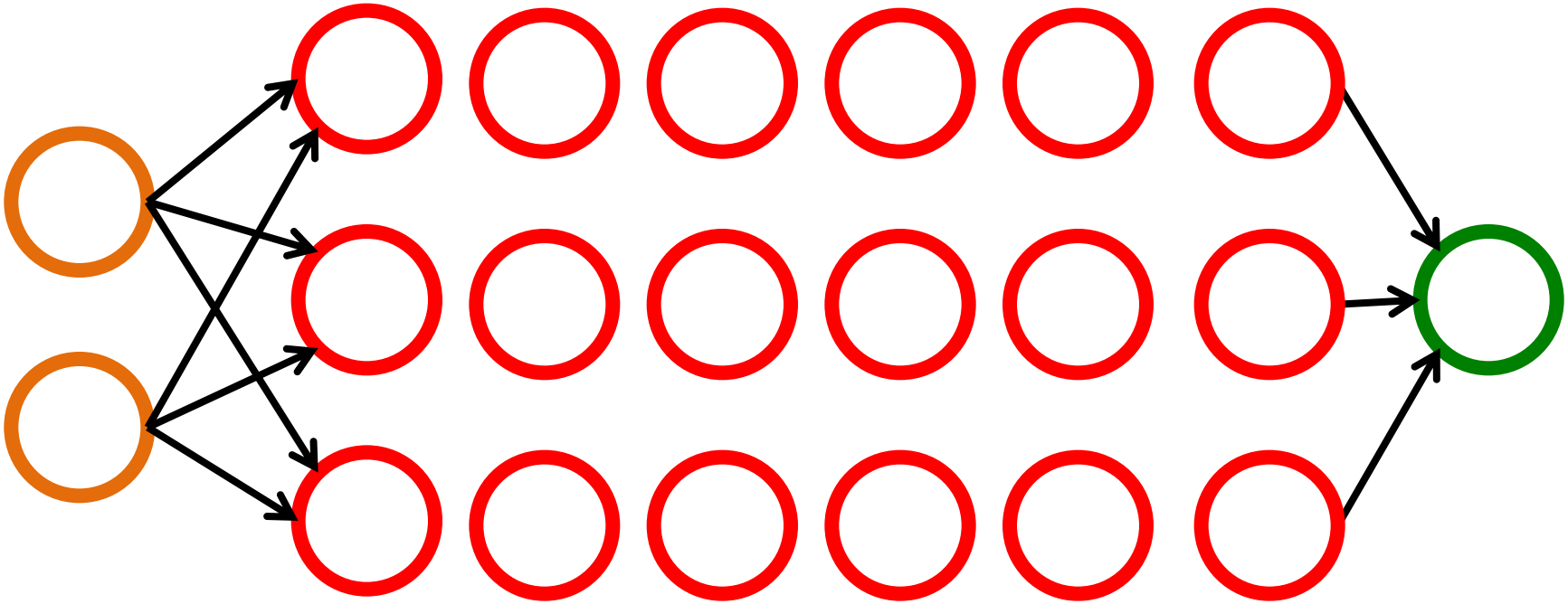
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

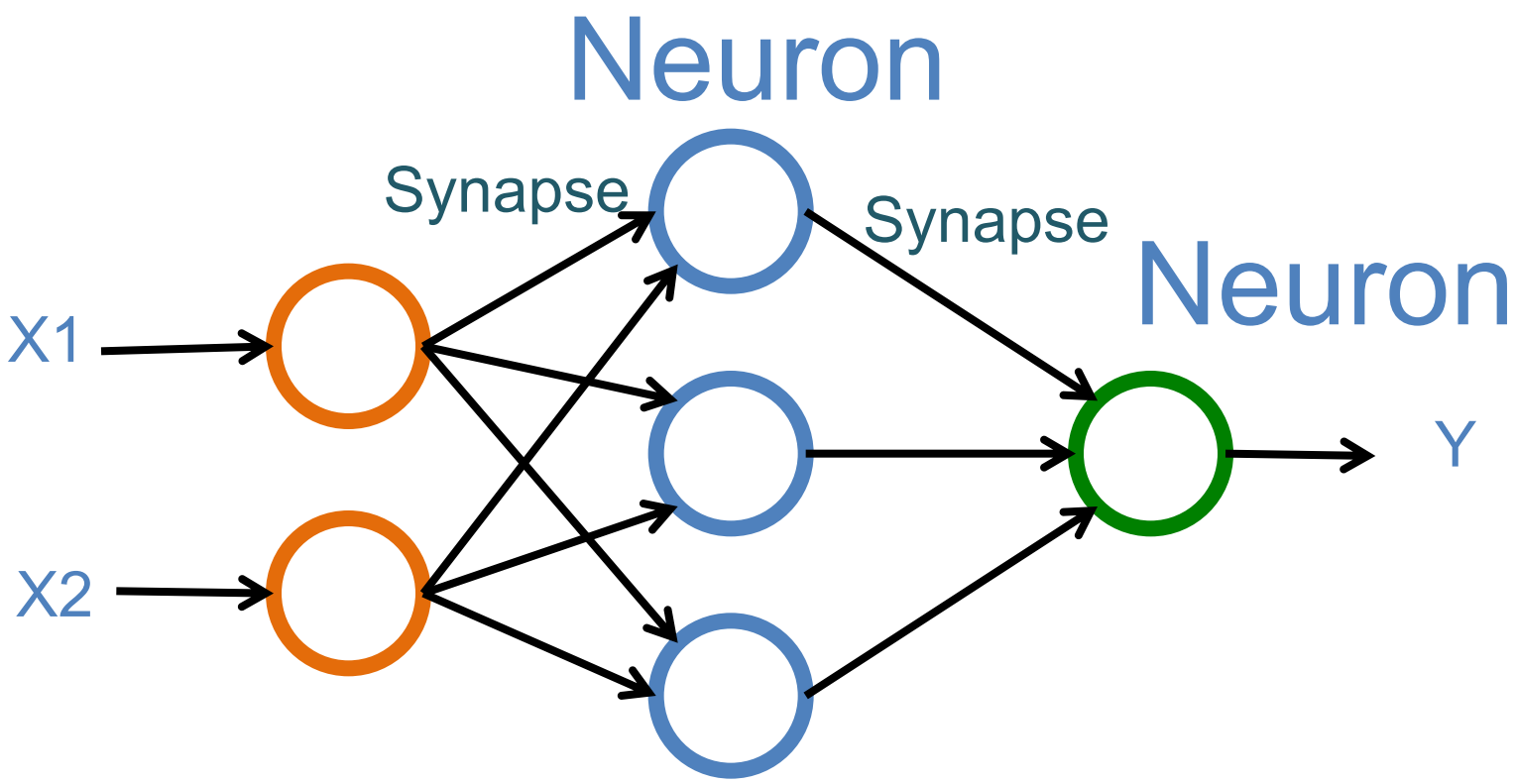
Deep Neural Networks
Deep Learning



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

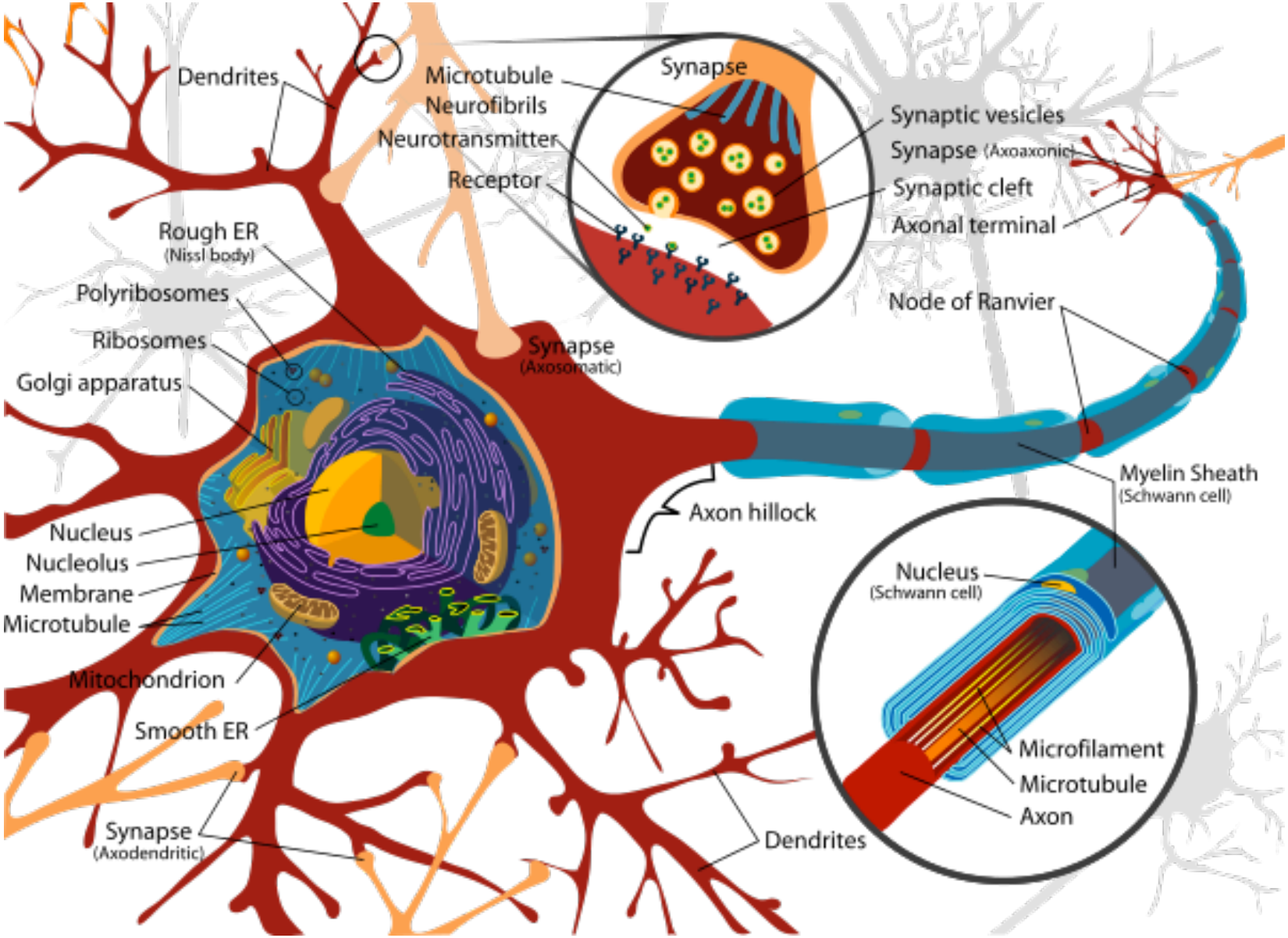
Neural Networks

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



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

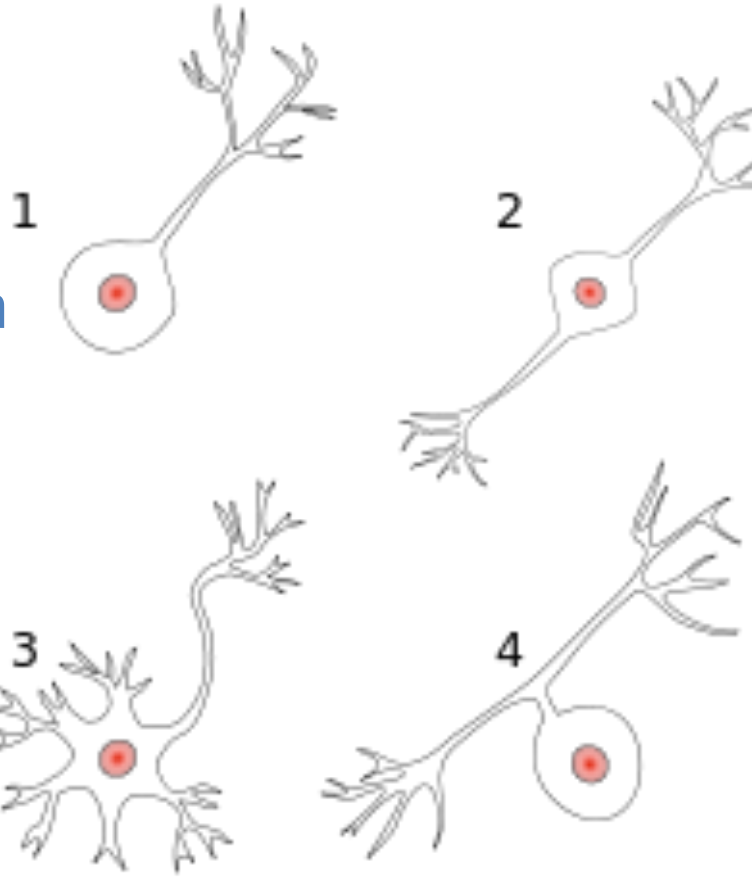
Neuron and Synapse



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

Neurons

1 Unipolar neuron



2 Bipolar neuron

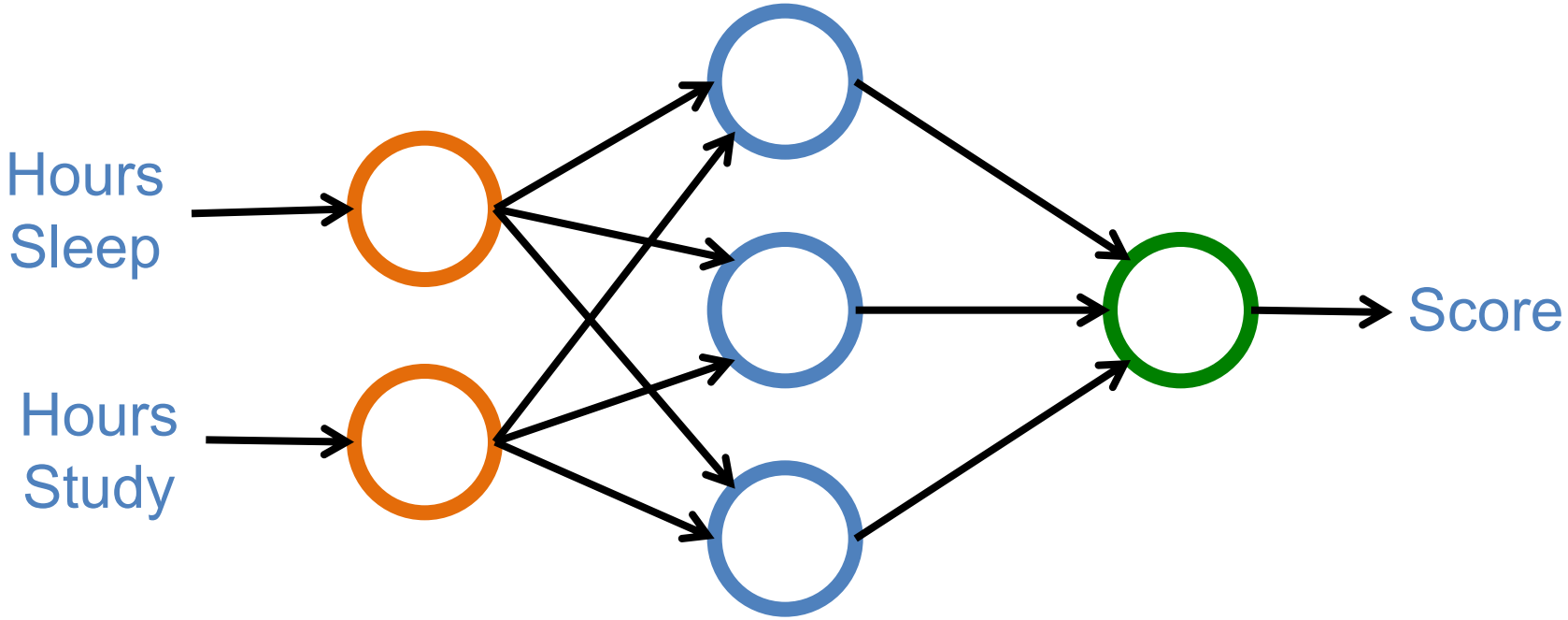
3 Multipolar neuron

4 Pseudounipolar neuron

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

Neural Networks

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



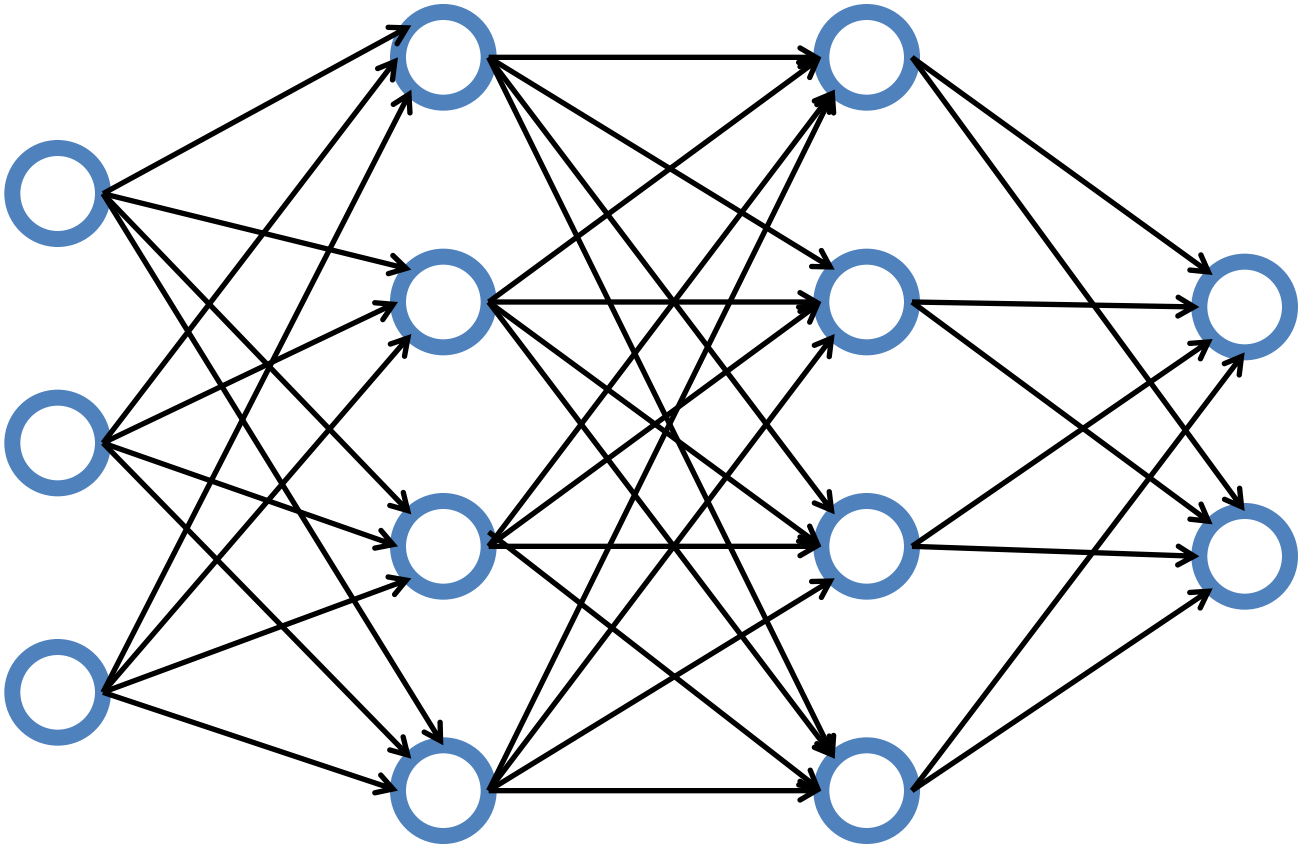
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X)

Hidden Layer (H)

Output Layer (Y)

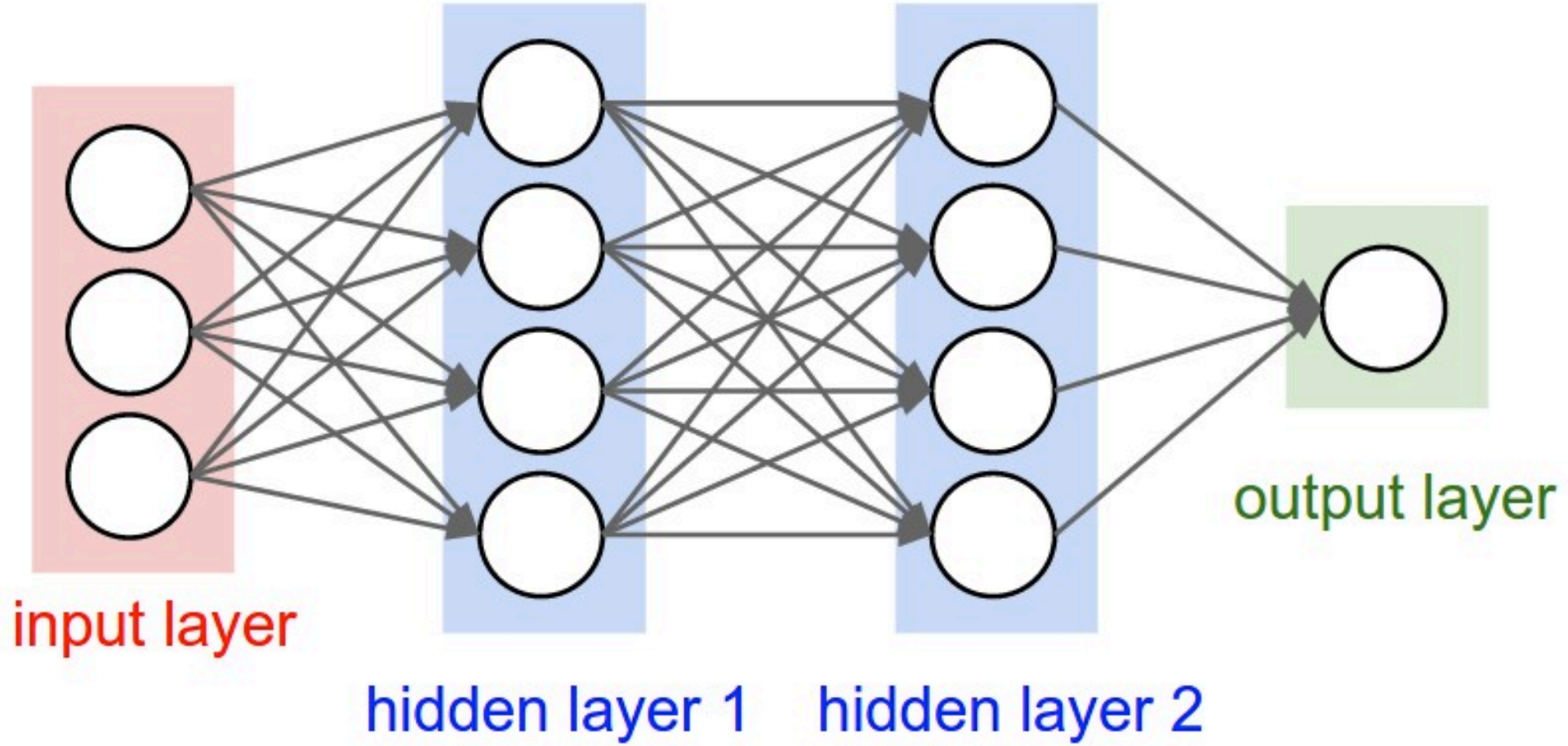


Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2

Convolutional Neural Networks (CNNs / ConvNets)

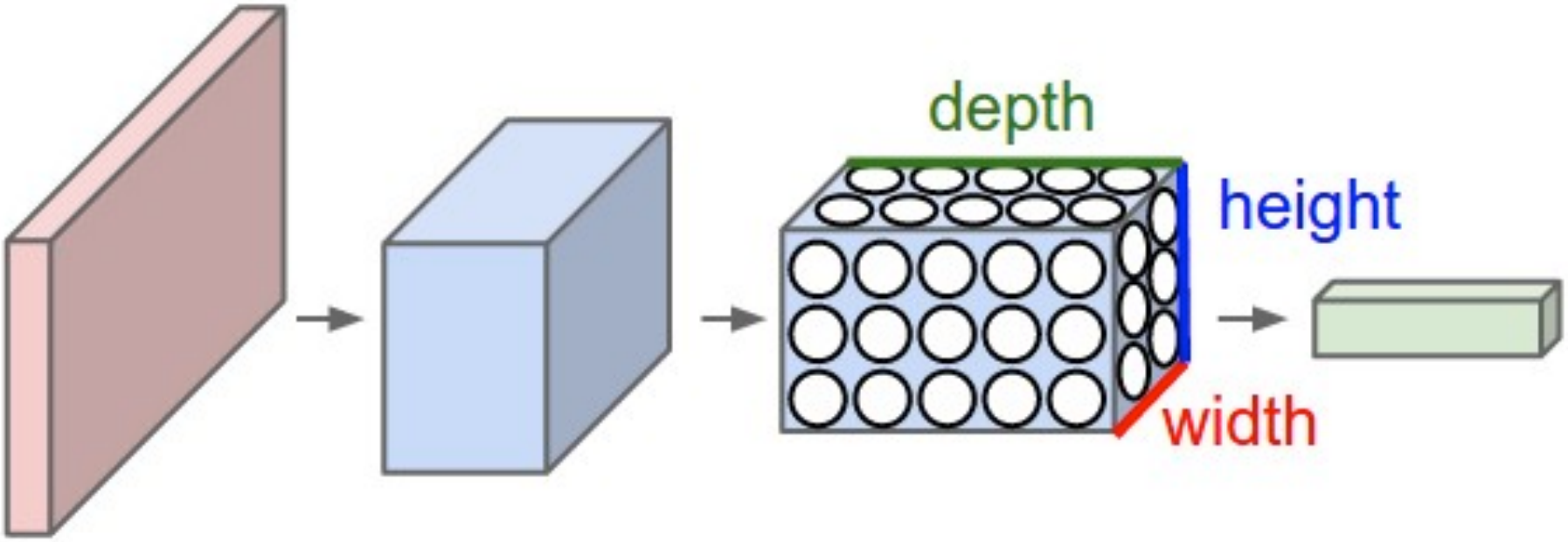
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



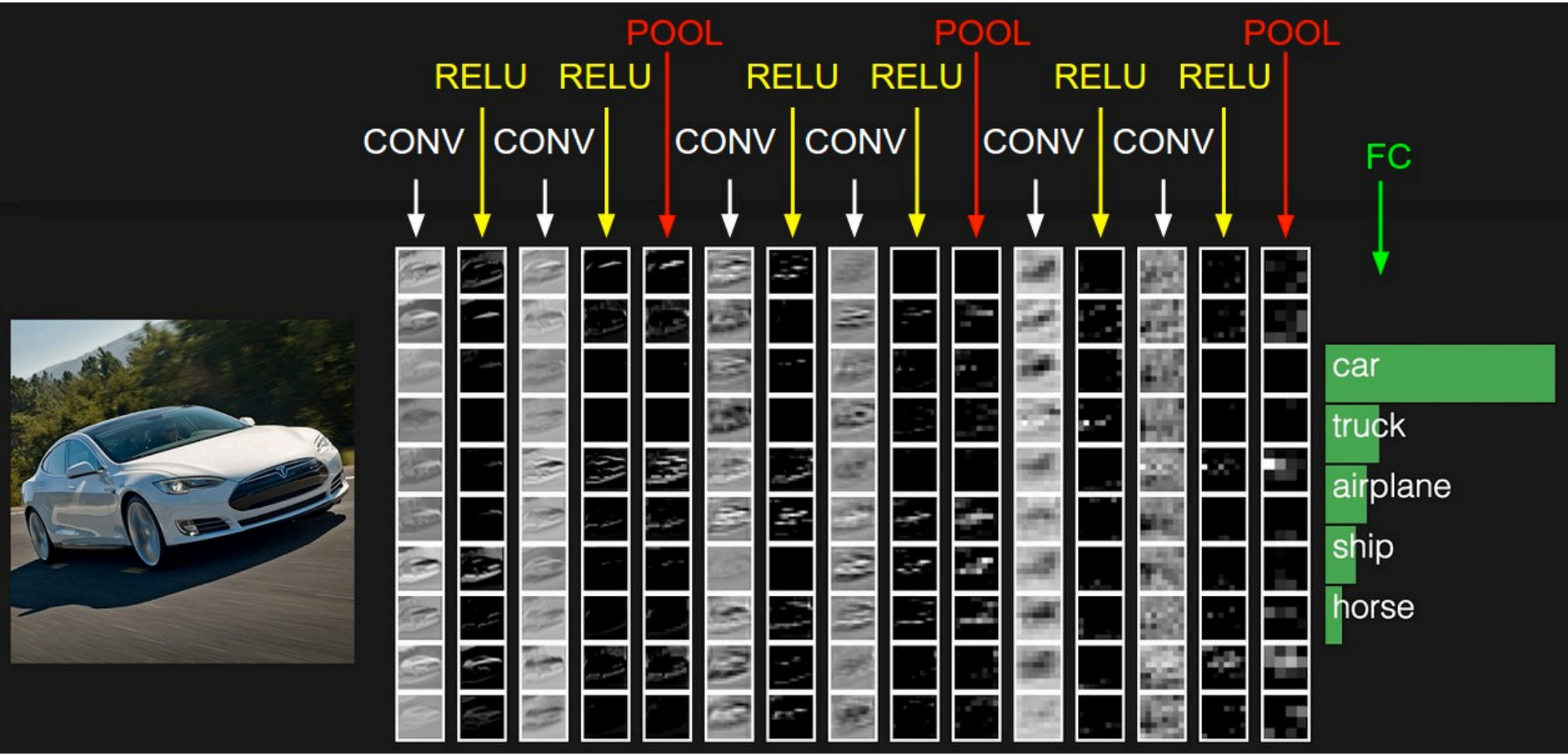
http://cs231n.github.io/convolutional-networks/

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



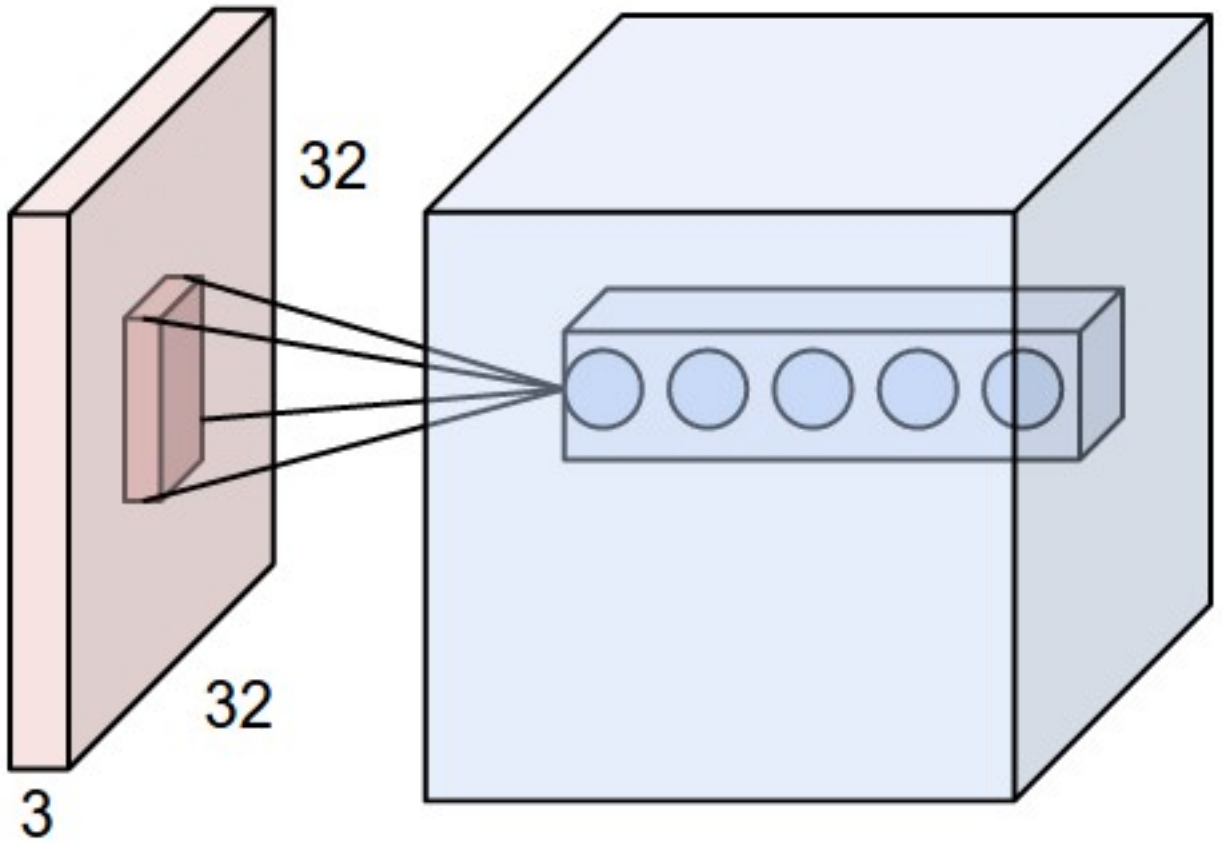
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



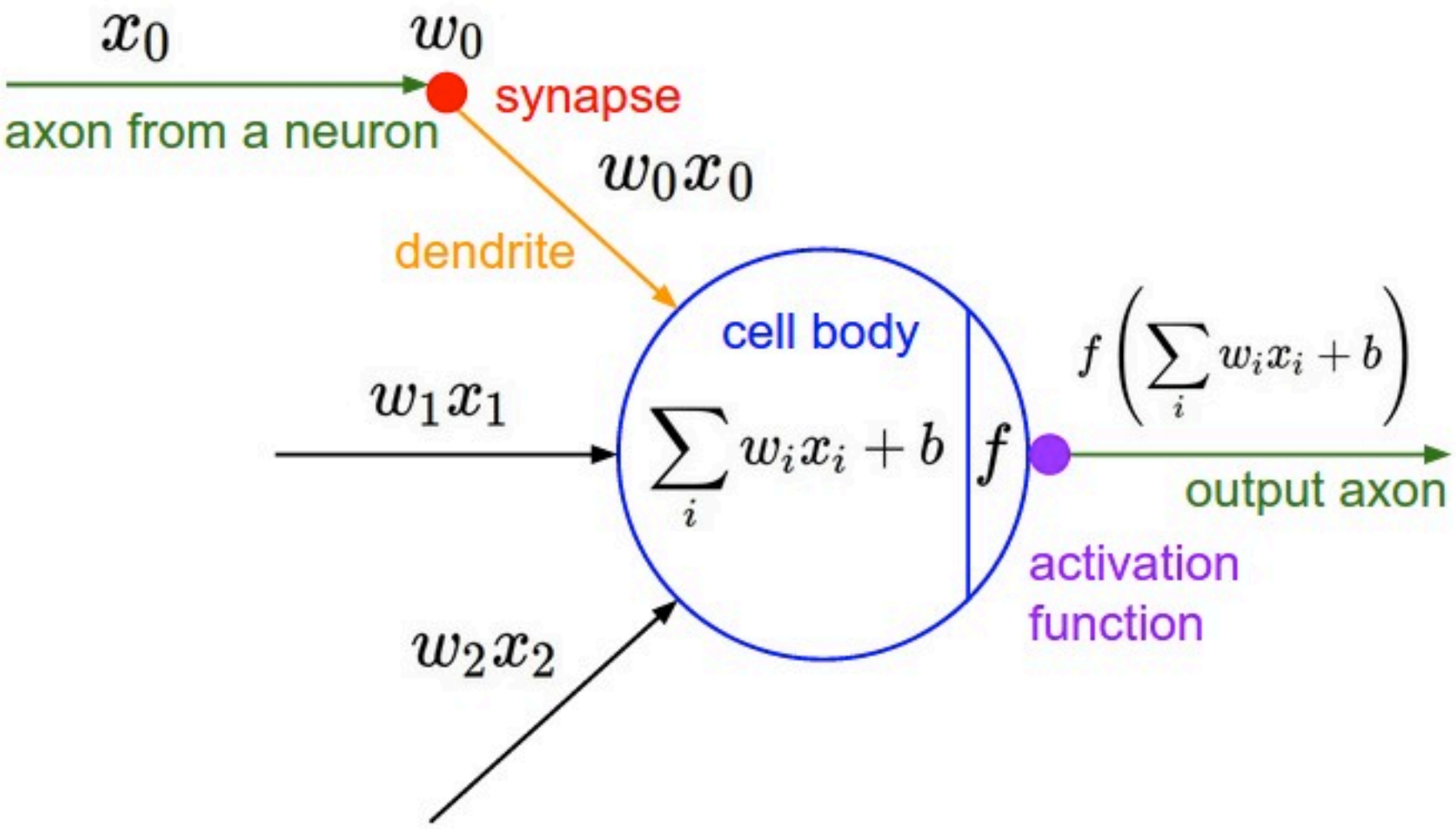
http://cs231n.github.io/convolutional-networks/

ConvNets



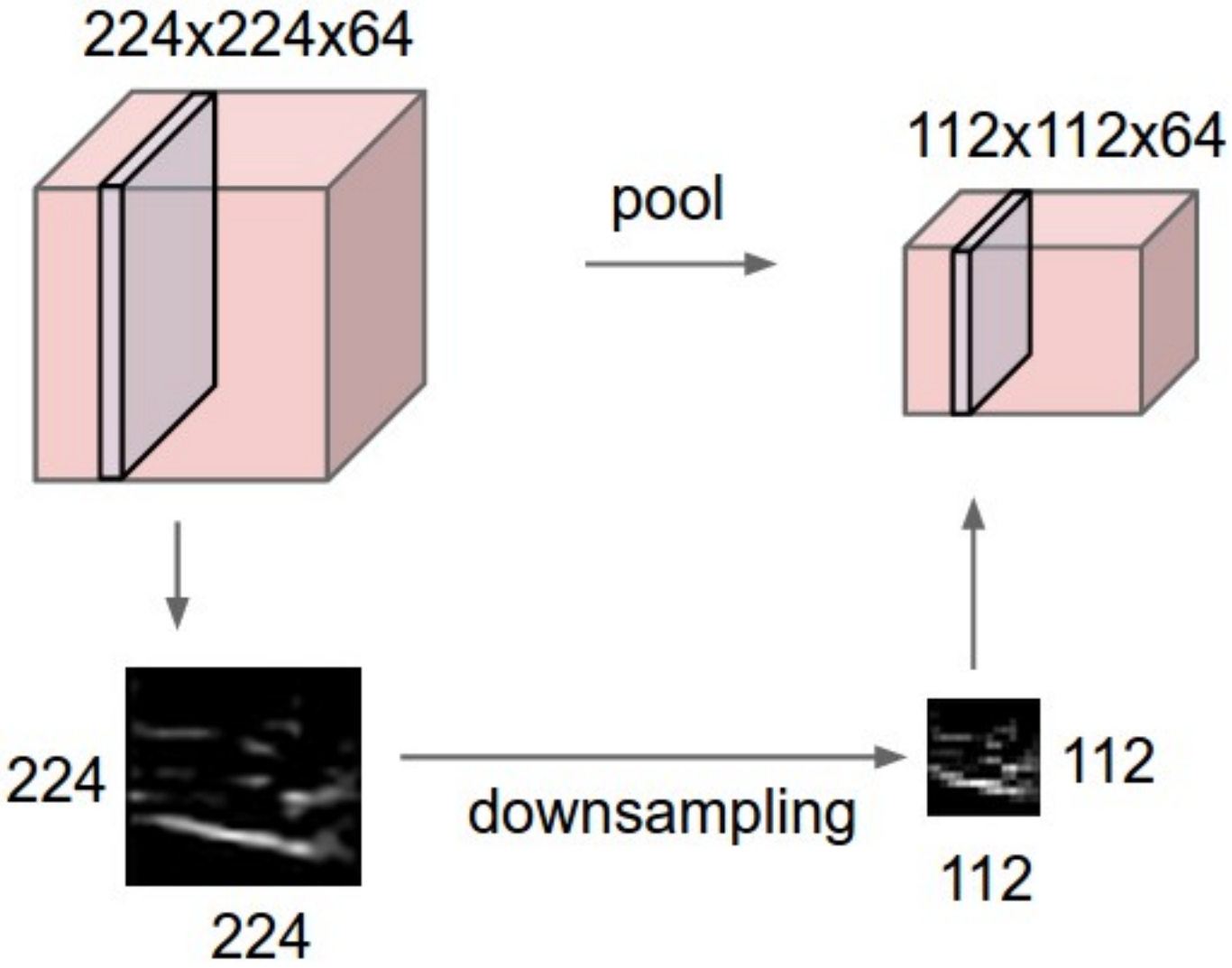
http://cs231n.github.io/convolutional-networks/

ConvNets



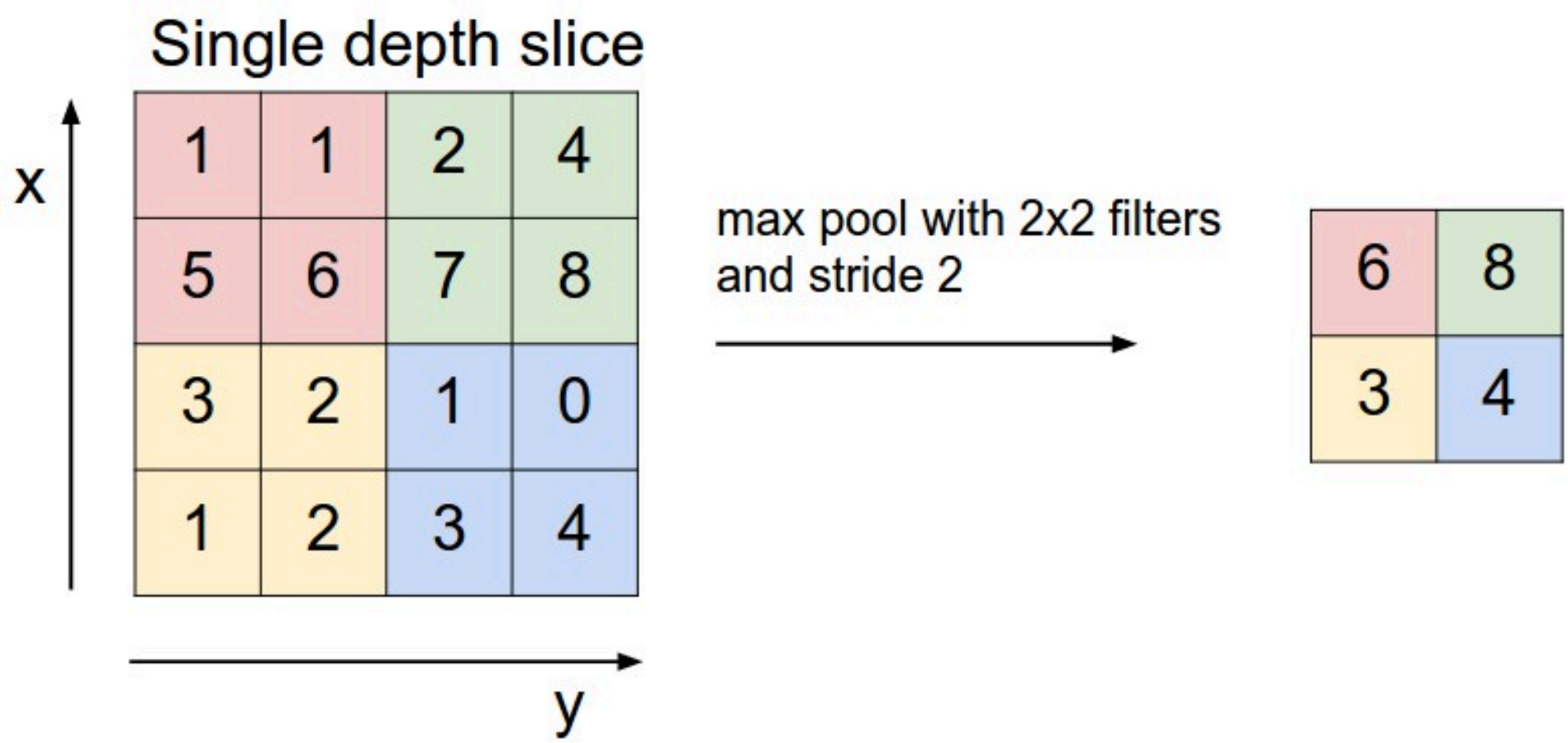
http://cs231n.github.io/convolutional-networks/

ConvNets



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

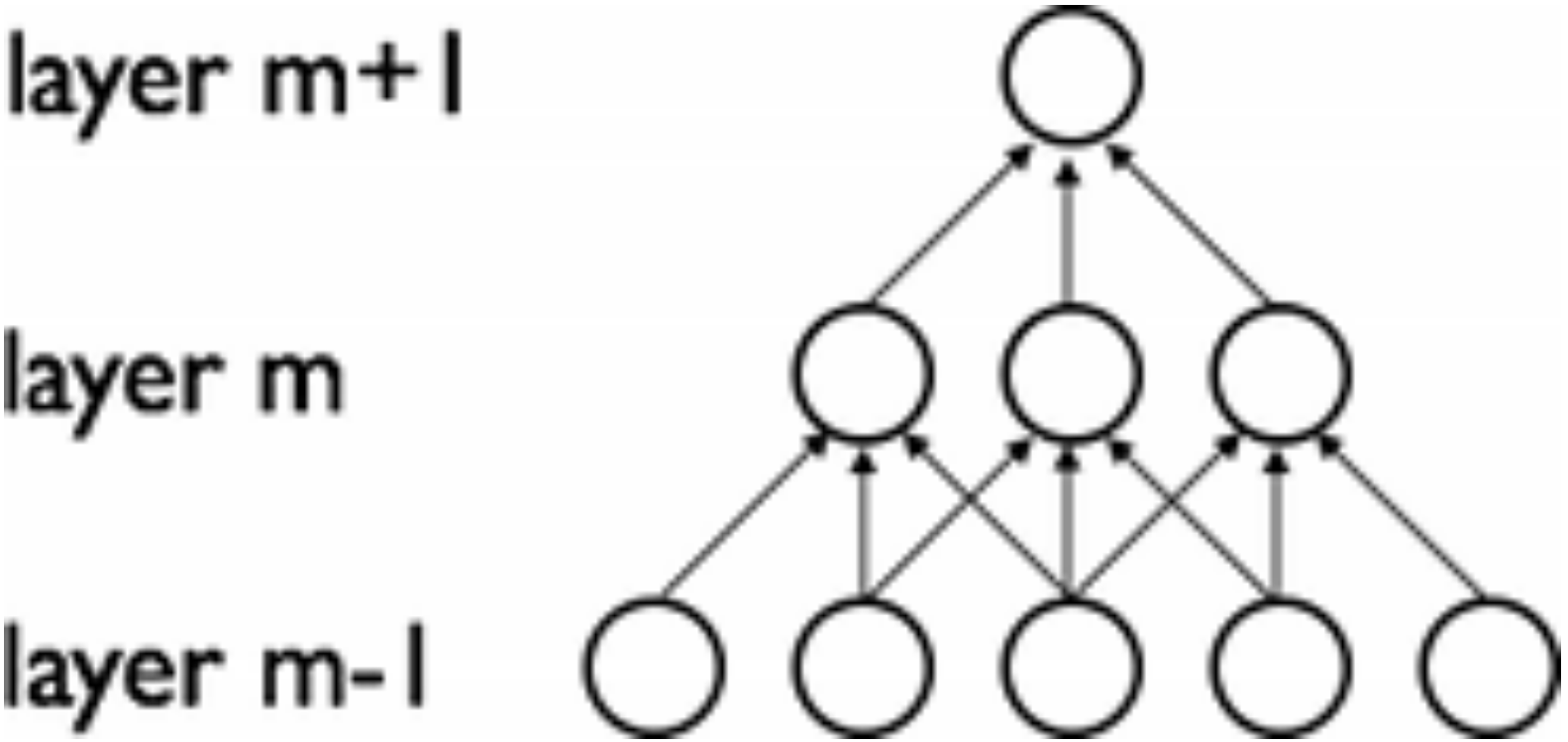
ConvNets
max pooling



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

Convolutional Neural Networks (CNN) (LeNet)

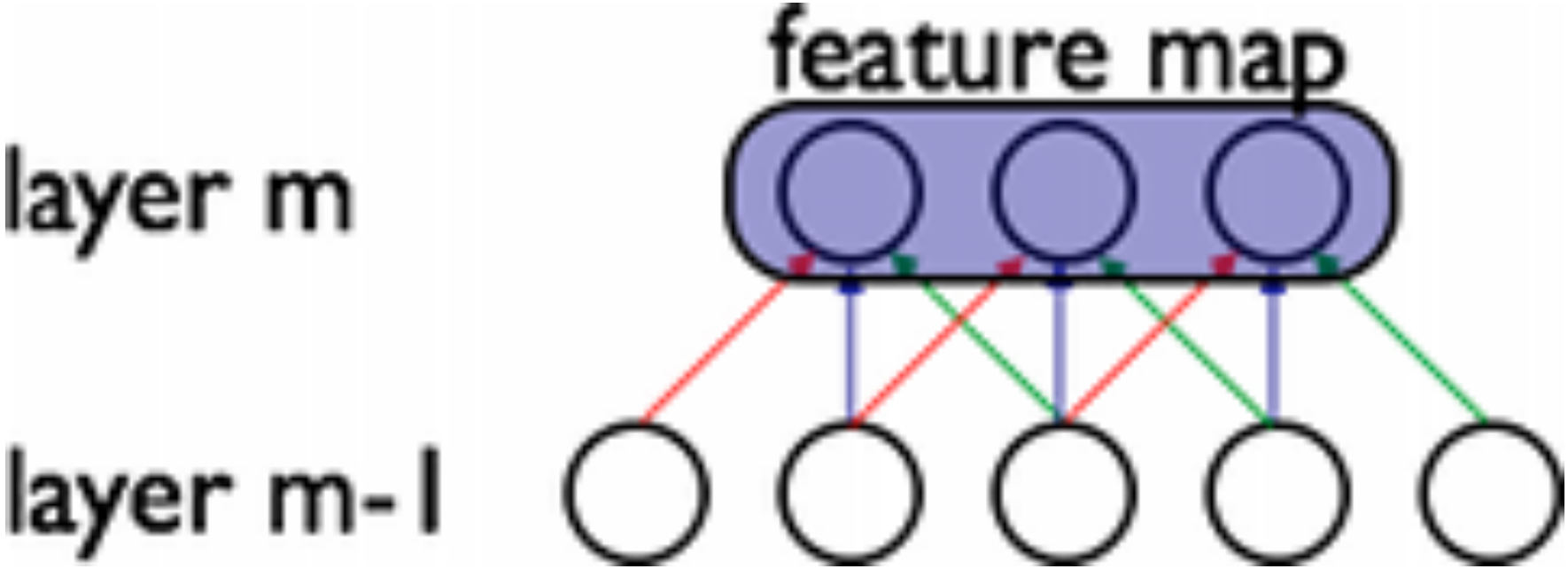
Sparse Connectivity



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

Convolutional Neural Networks (CNN) (LeNet)

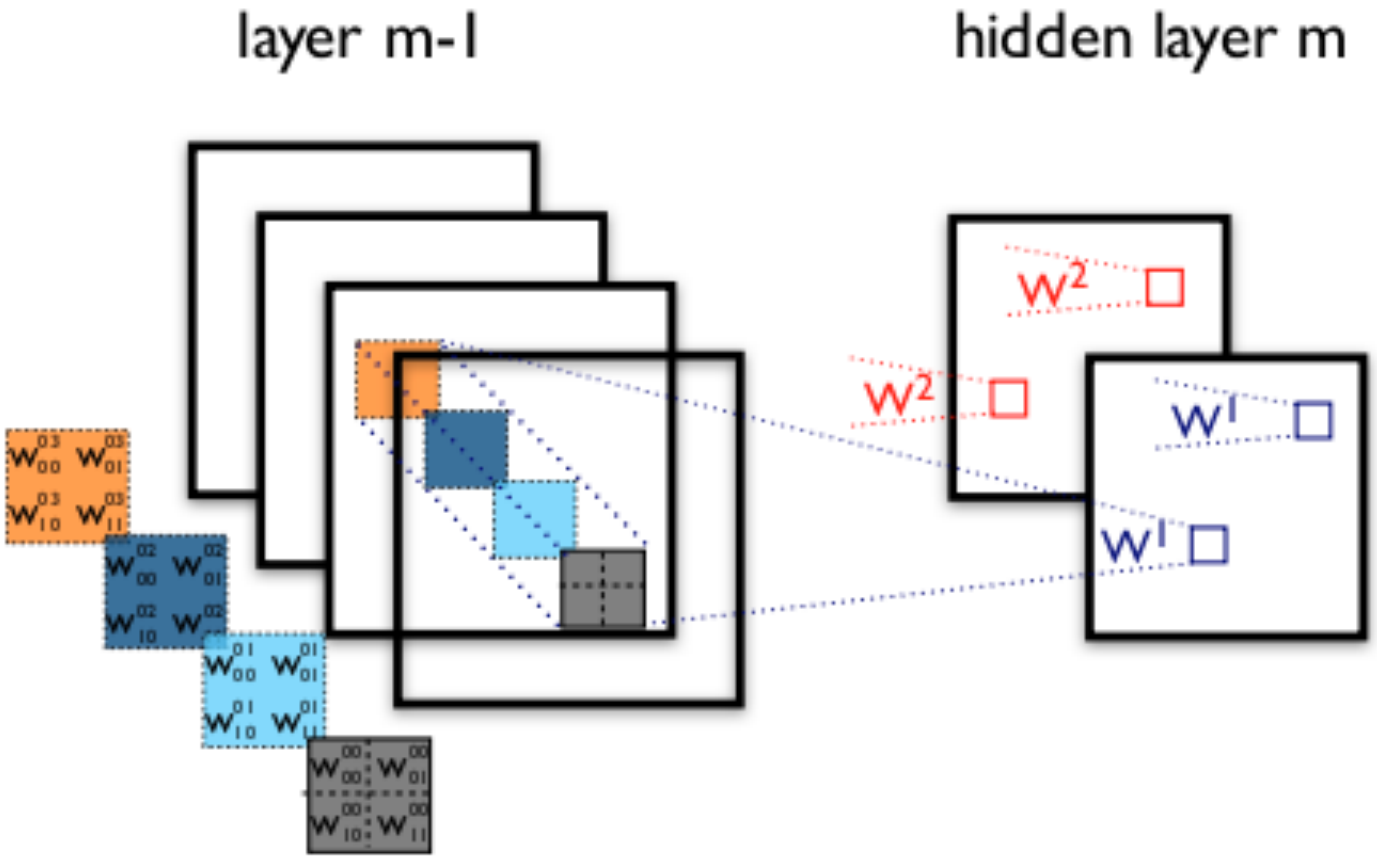
Shared Weights



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

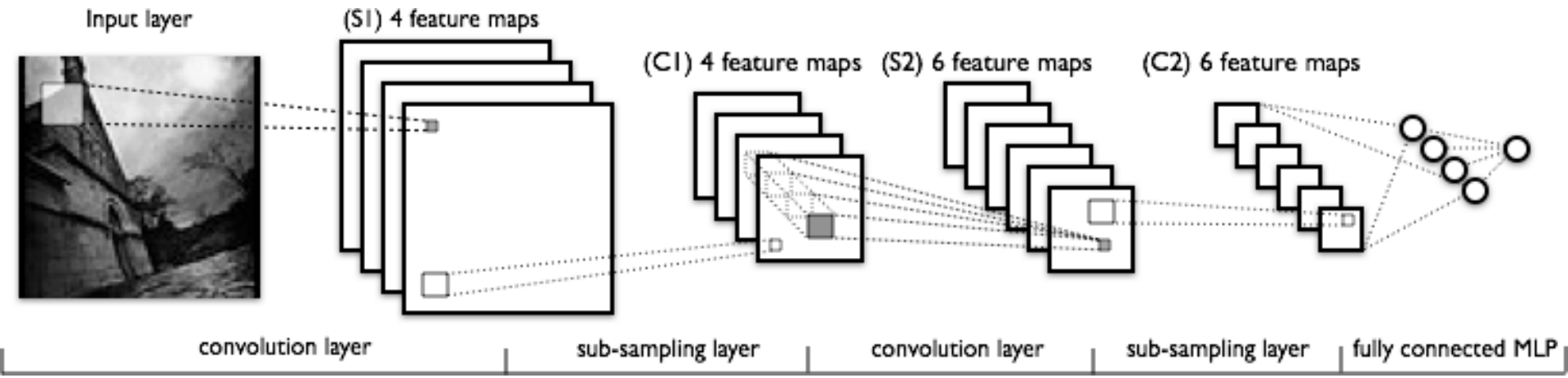
Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer



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

Convolutional Neural Networks (CNN) (LeNet)



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

show flights from Boston to New York today

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

Recurrent Neural Networks with Word Embeddings
Semantic Parsing / Slot-Filling (Spoken Language Understanding)

Input (words)	show	flights	from	Boston	to	New	York	today
Output (labels)	O	O	O	B-dept	O	B-arr	I-arr	B-date

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

show flights from Boston to New York today

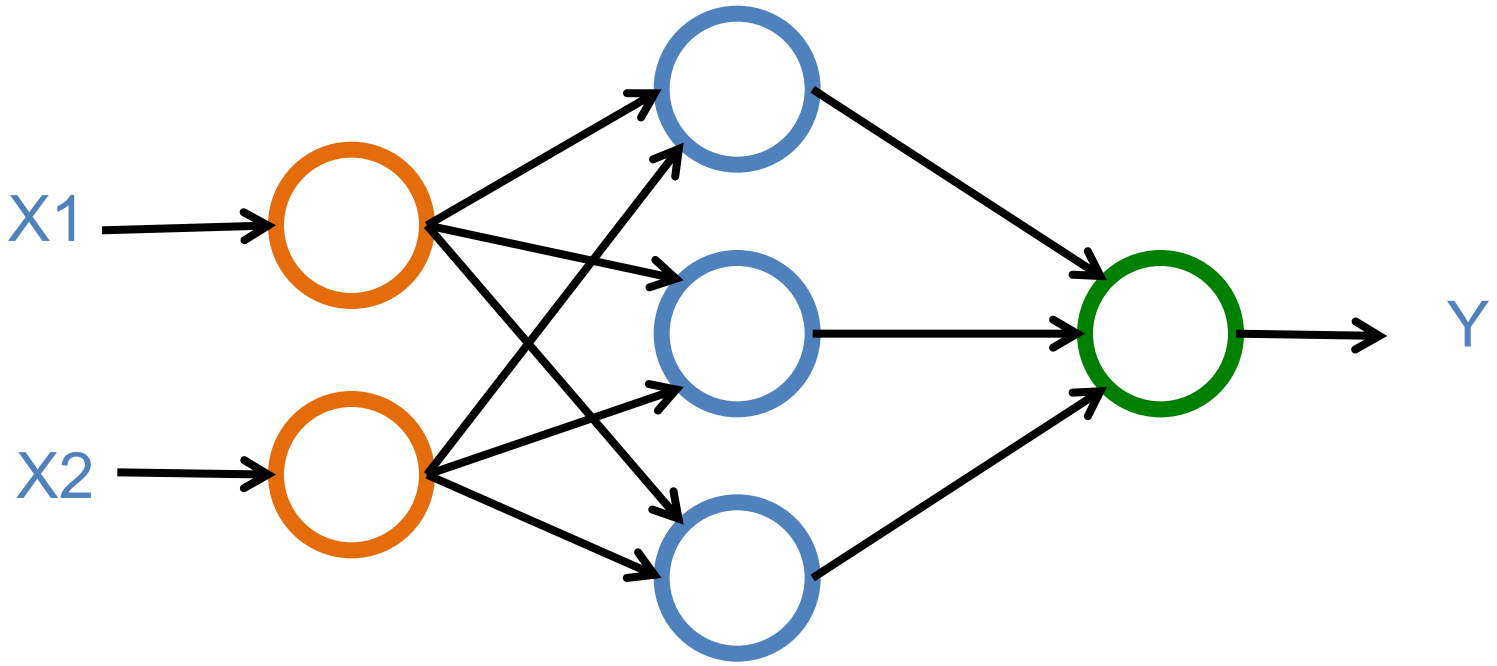
show flights from Boston to New York today

Input (words)	show	flights	from	Boston	to	New	York	today
Output (labels)	O	O	O	B-dept	O	B-arr	I-arr	B-date

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

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

Training a Network
=
Minimize the Cost Function

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

Training a Network

=

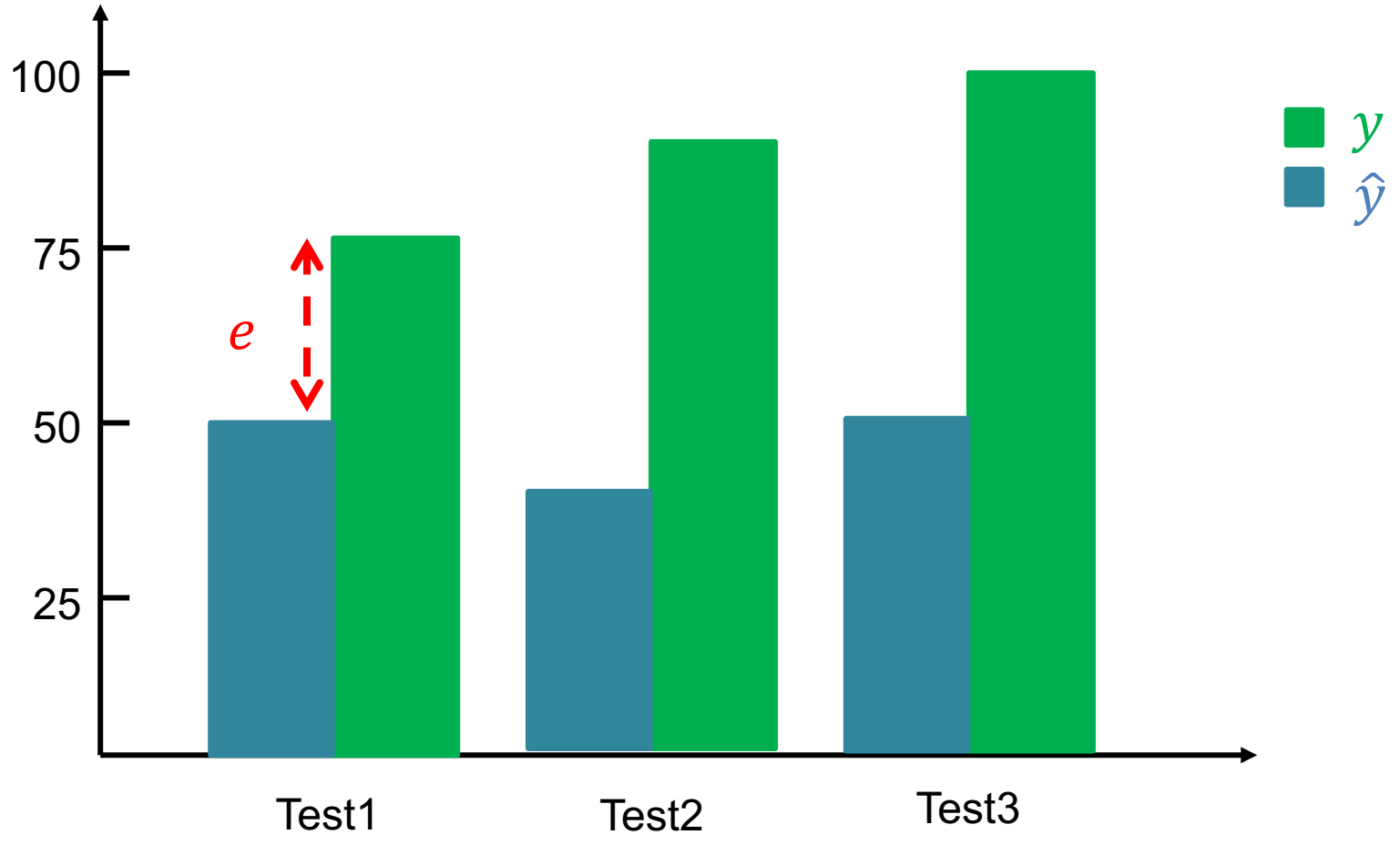
Minimize the Cost Function

Minimize the Loss Function

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

Error = Predict Y - Actual Y

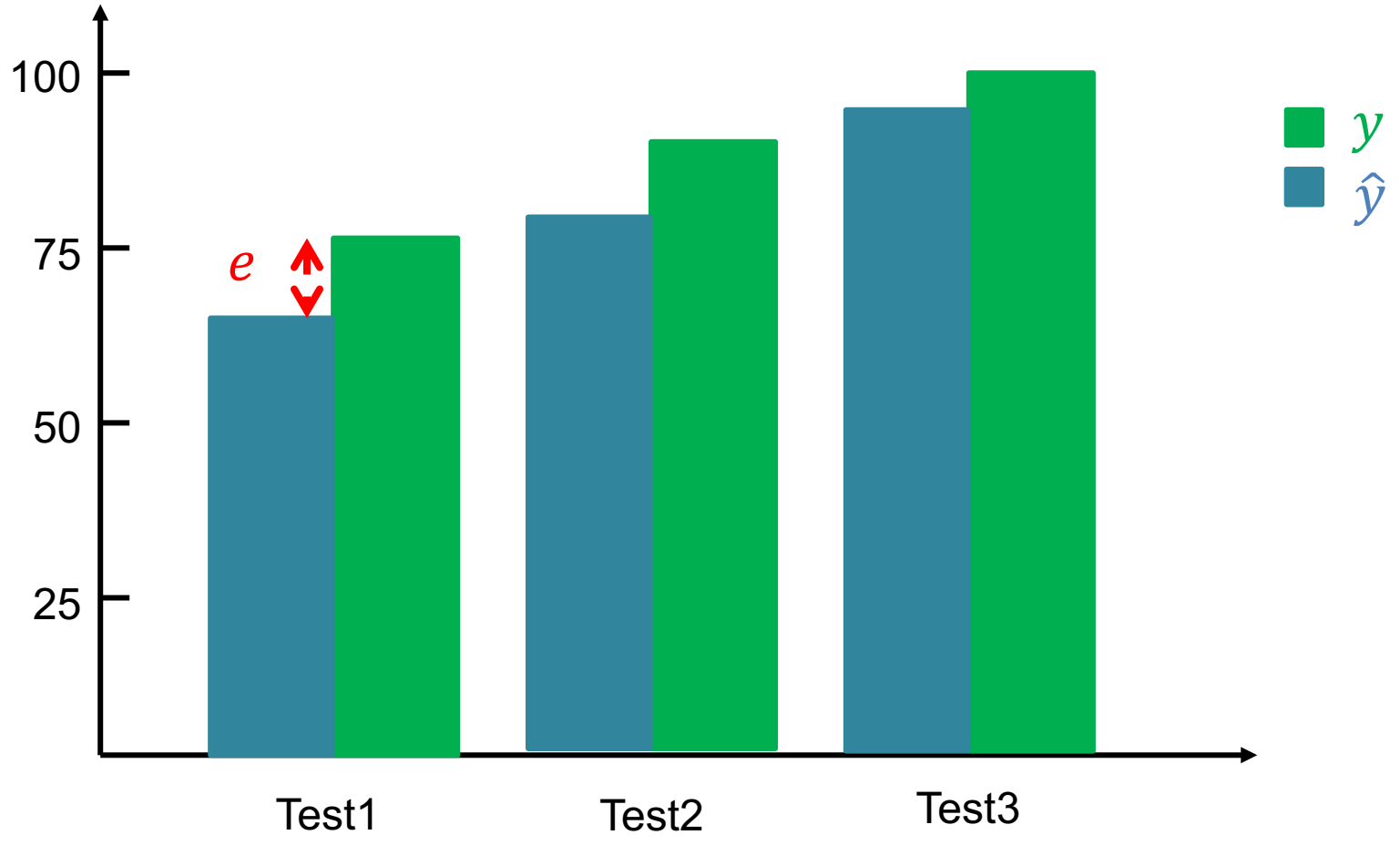
Error : Cost : Loss



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

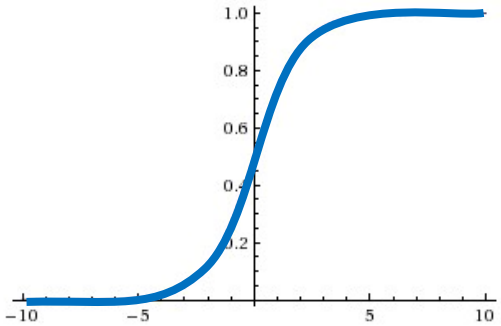
Error : Cost : Loss

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

Activation Functions

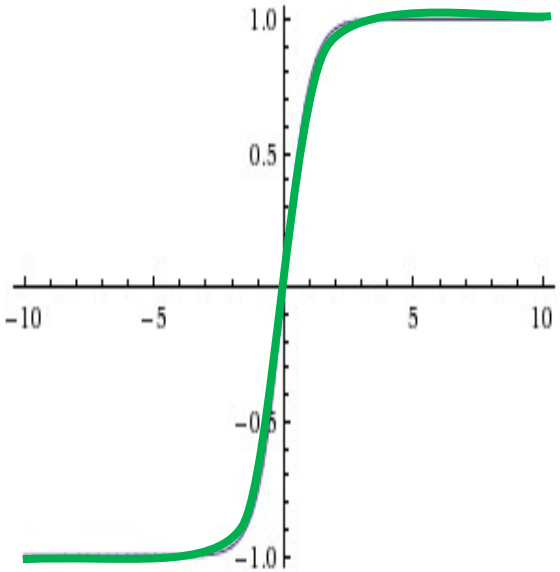
Activation Functions

Sigmoid



[0, 1]

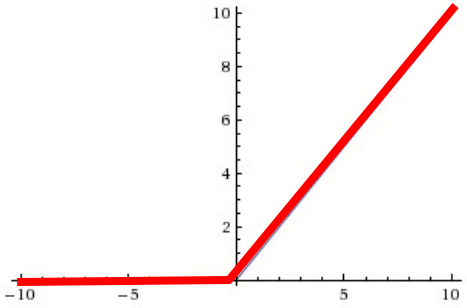
TanH



[-1, 1]

ReLU

(Rectified Linear Unit)

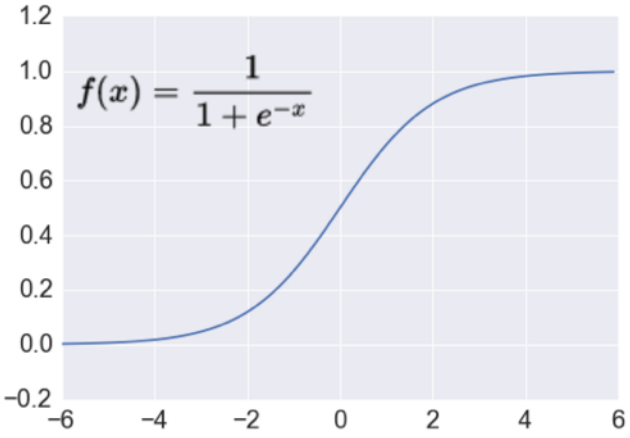


$f(x) = max(0, x)$

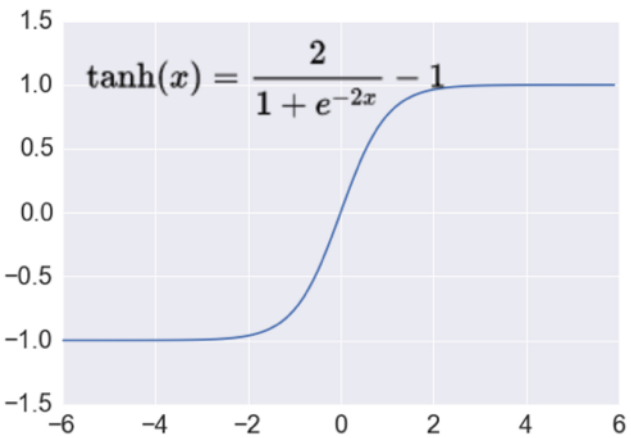
Source: http://cs231n.github.io/neural-networks-1/

Activation Functions

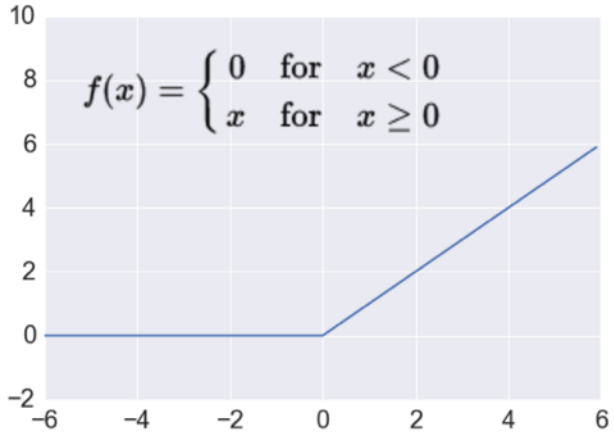
Sigmoid



TanH



ReLU



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

Loss Function

Binary Classification: 2 Class

Activation Function: Sigmoid

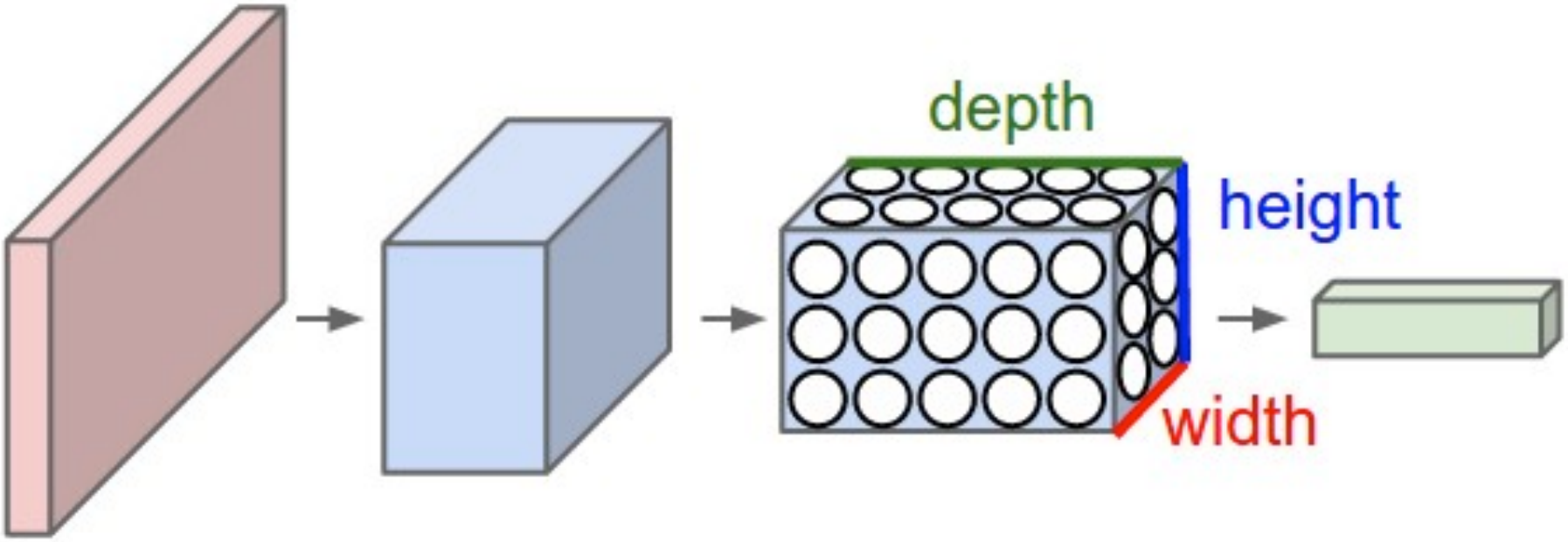
Loss Function: Binary Cross-Entropy

Multiple Classification: 10 Class

Activation Function: SoftMAX

Loss Function: Categorical Cross-Entropy

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

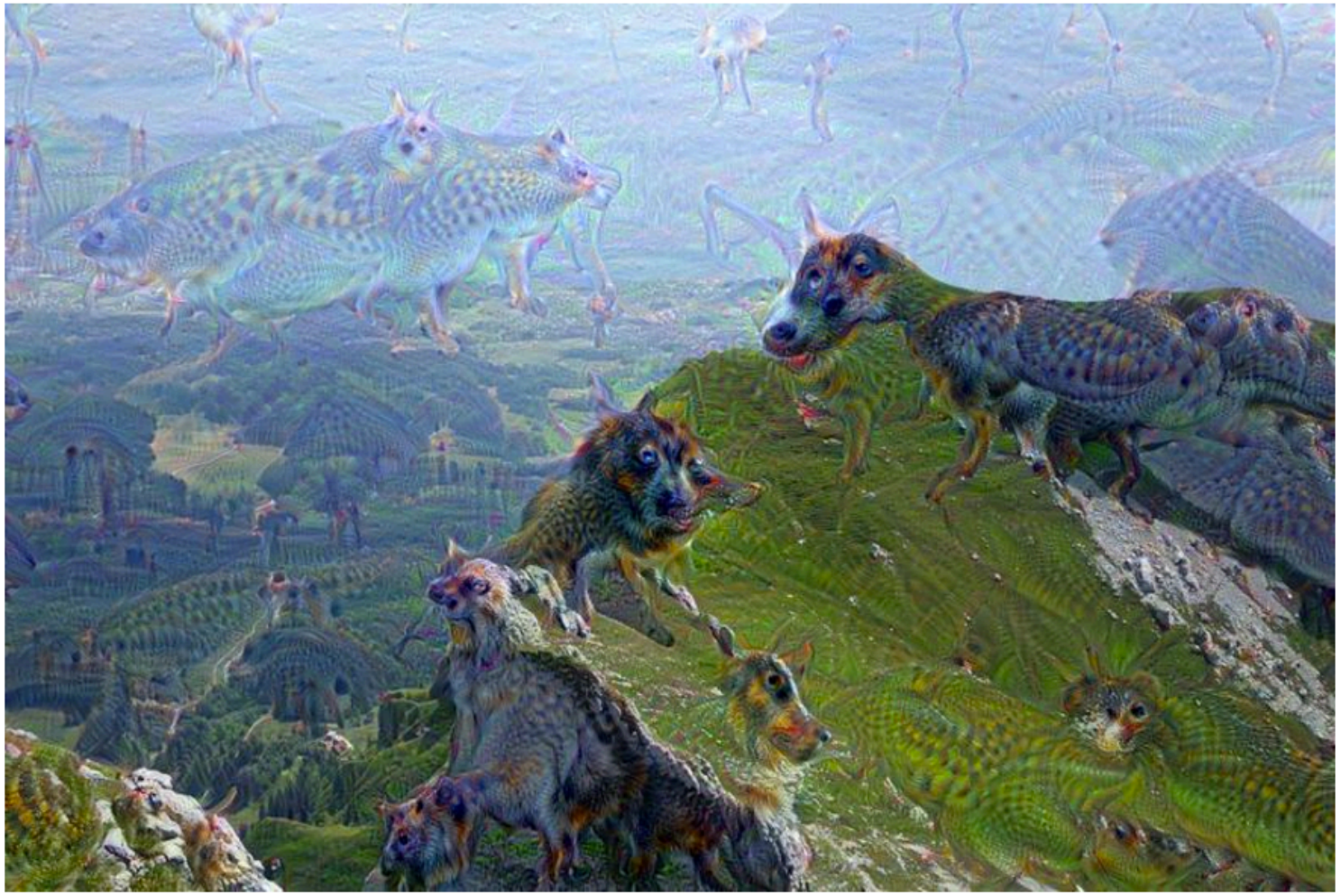


http://cs231n.github.io/convolutional-networks/

DeepDream

GitHub, Inc. [US] https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

In [15]:
```
render_deepdream(tf.square(T('mixed4c')), img0)
```

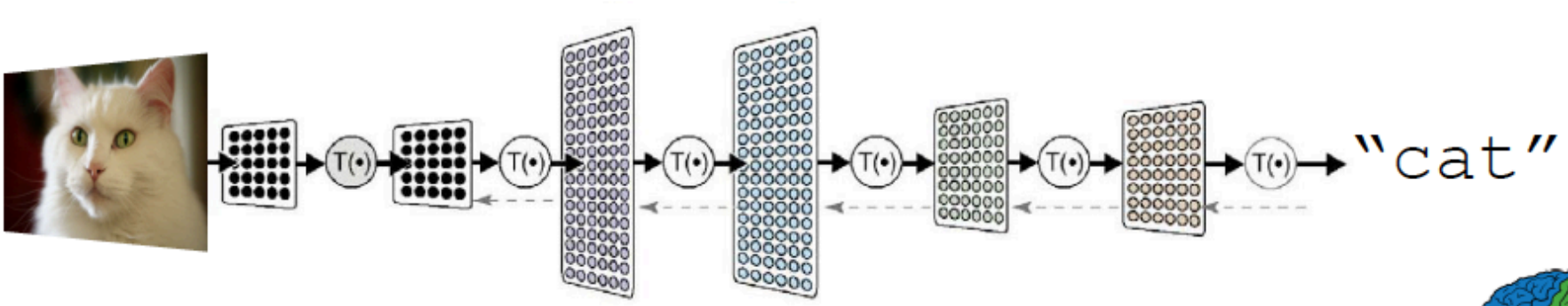


Note that results can differ from the Caffe's implementation, as we are using an independently trained network. Still, the network seems to like dogs and animal-like features due to the nature of the ImageNet dataset.

Source: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

Deep Learning

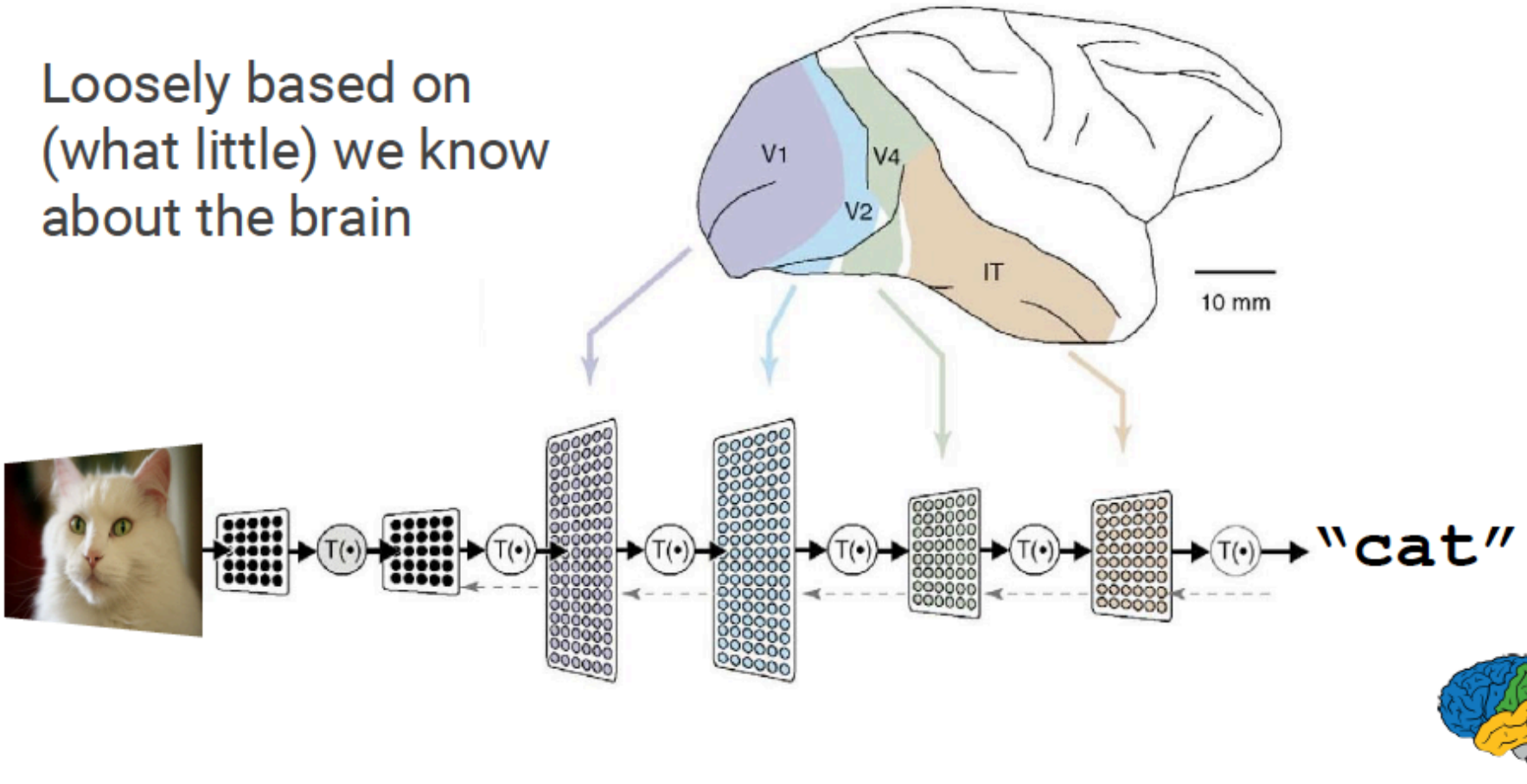
- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple,
  trainable mathematical functions
- Compatible with many variants of machine learning



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

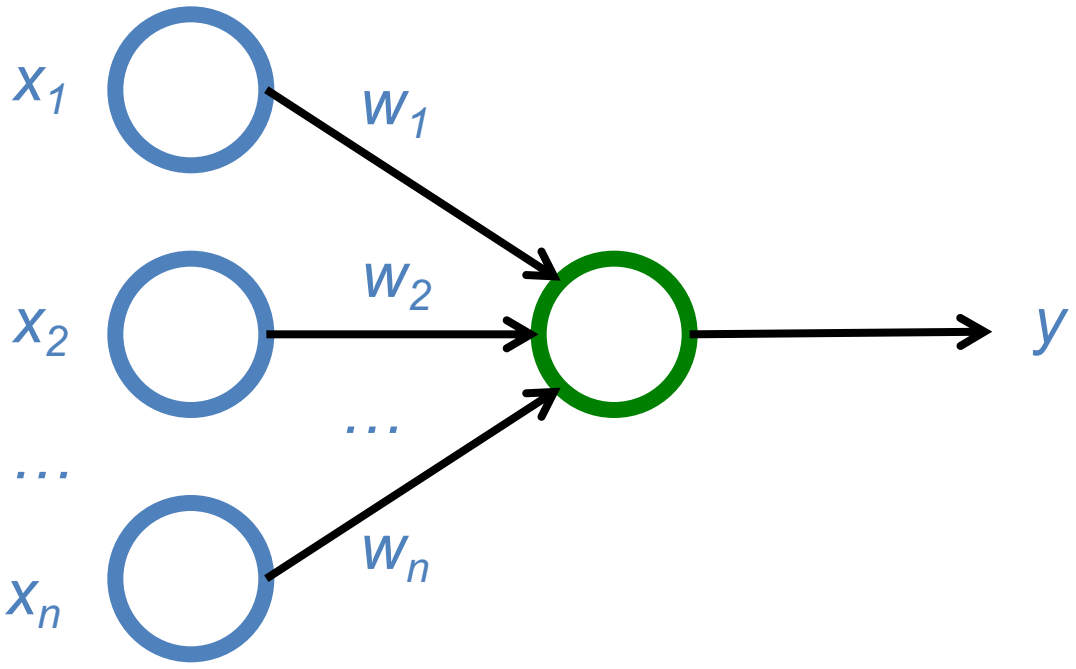
What is Deep Learning?

- 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

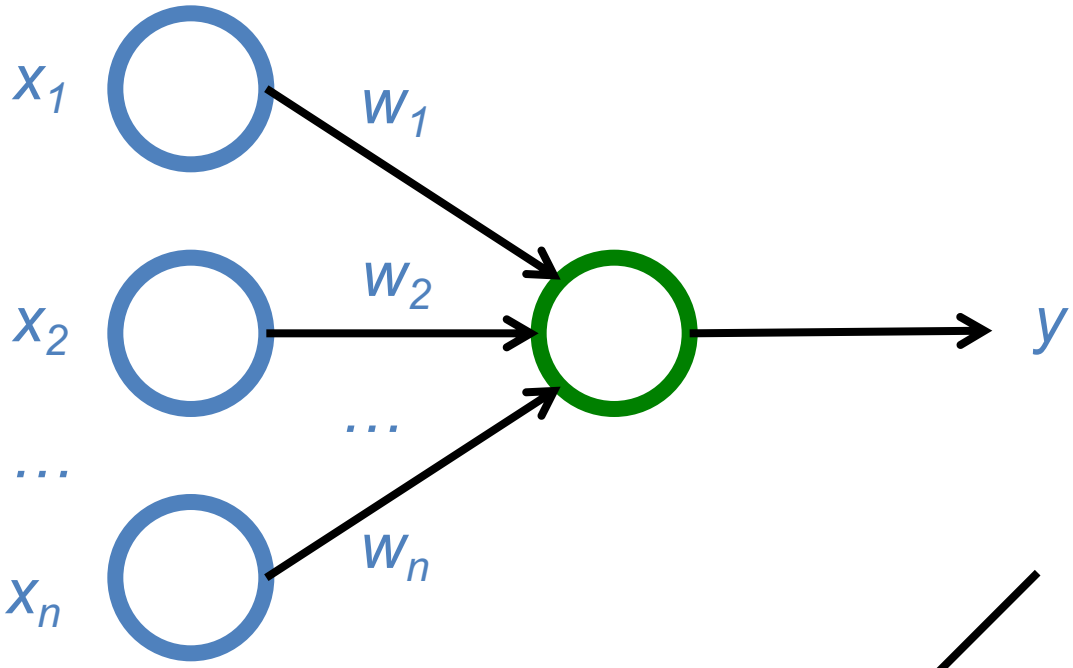
The Neuron

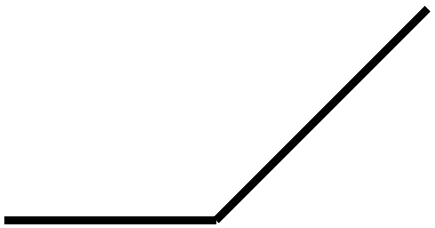


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

The Neuron

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

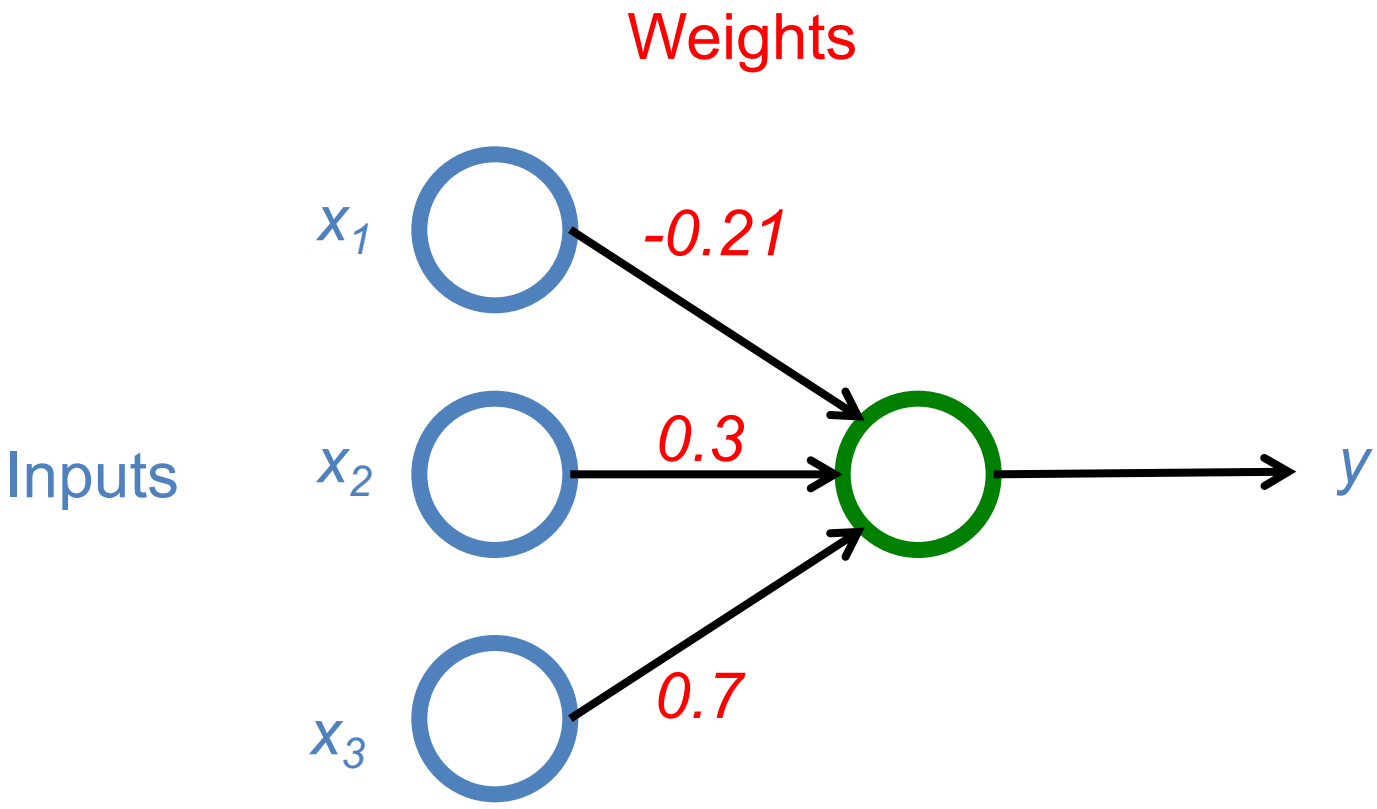




$$F(x) = \max(0, x)$$

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

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



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

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

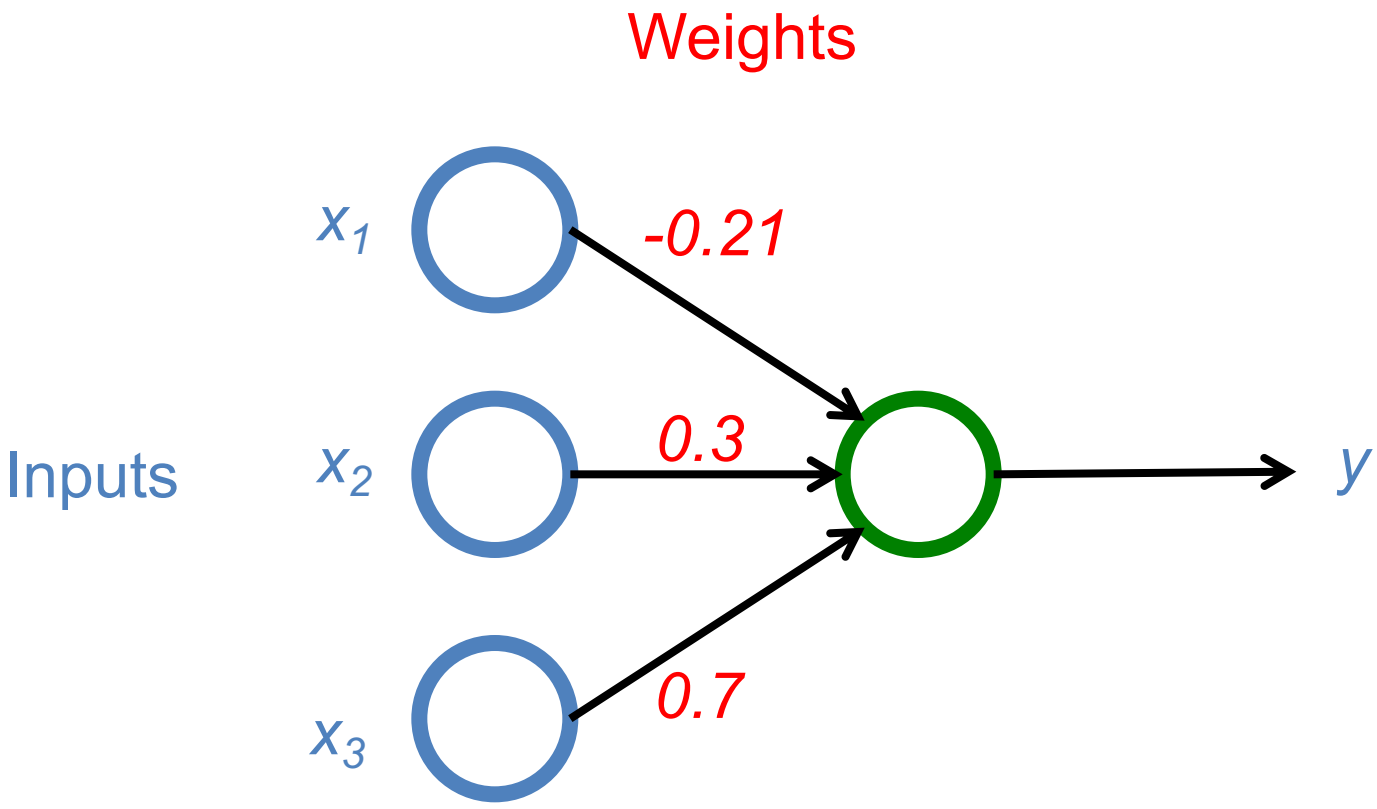
Learning Algorithm

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

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



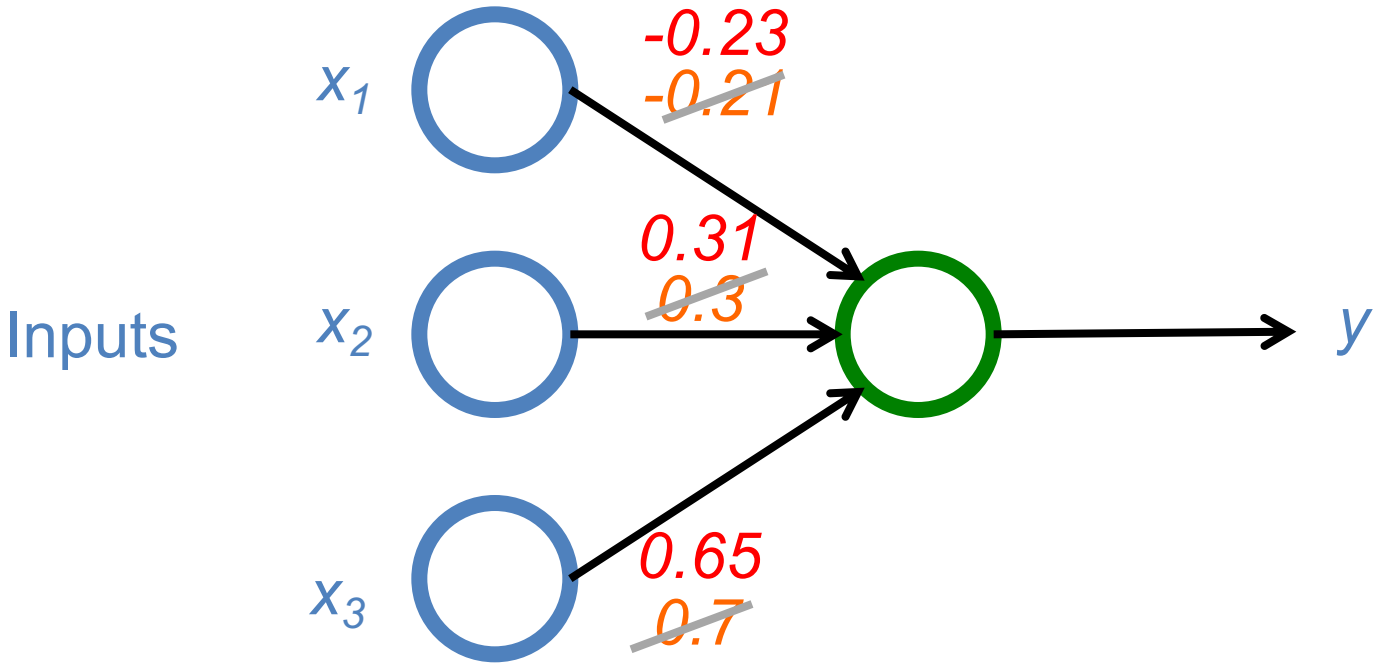
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

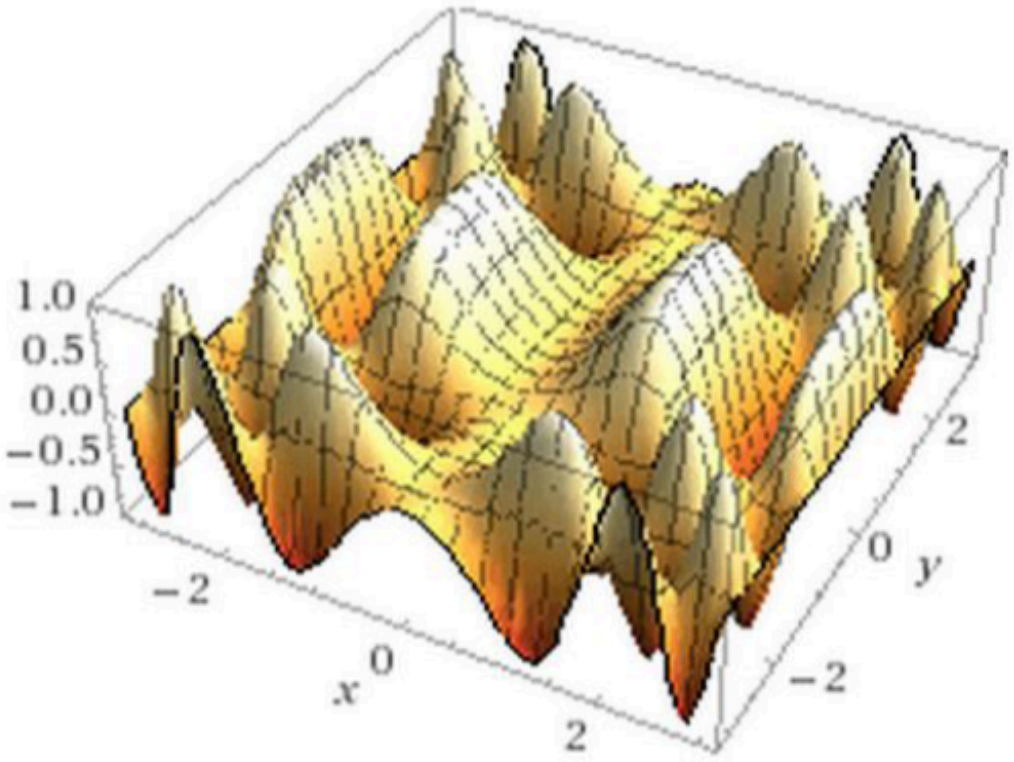
Next time:

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

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

Weights





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

Important Property of Neural Networks

Results get better with

More data +

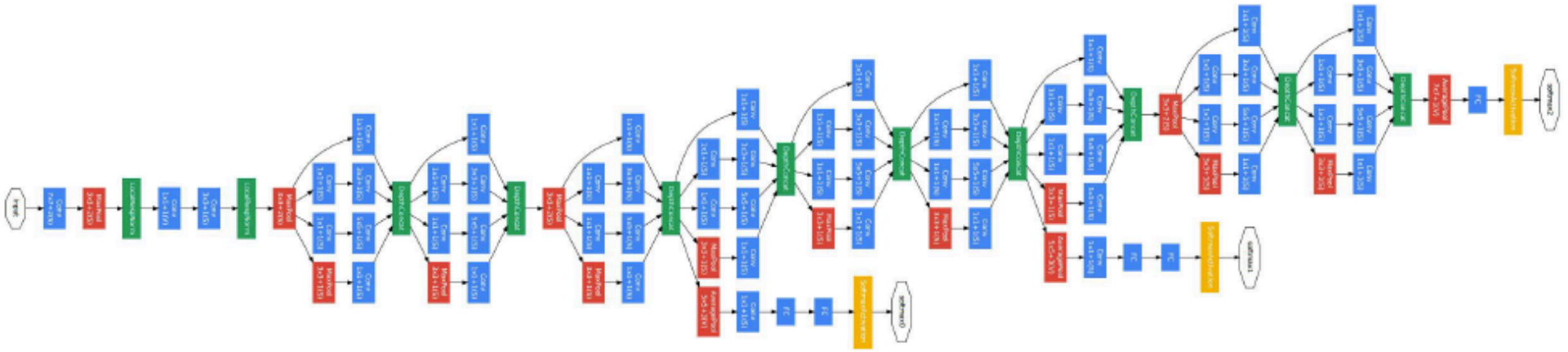
Bigger models +

More computation

(Better algorithms, new insights
and improved techniques always help, too!)

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

The Inception Architecture (GoogLeNet, 2014)



Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015

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

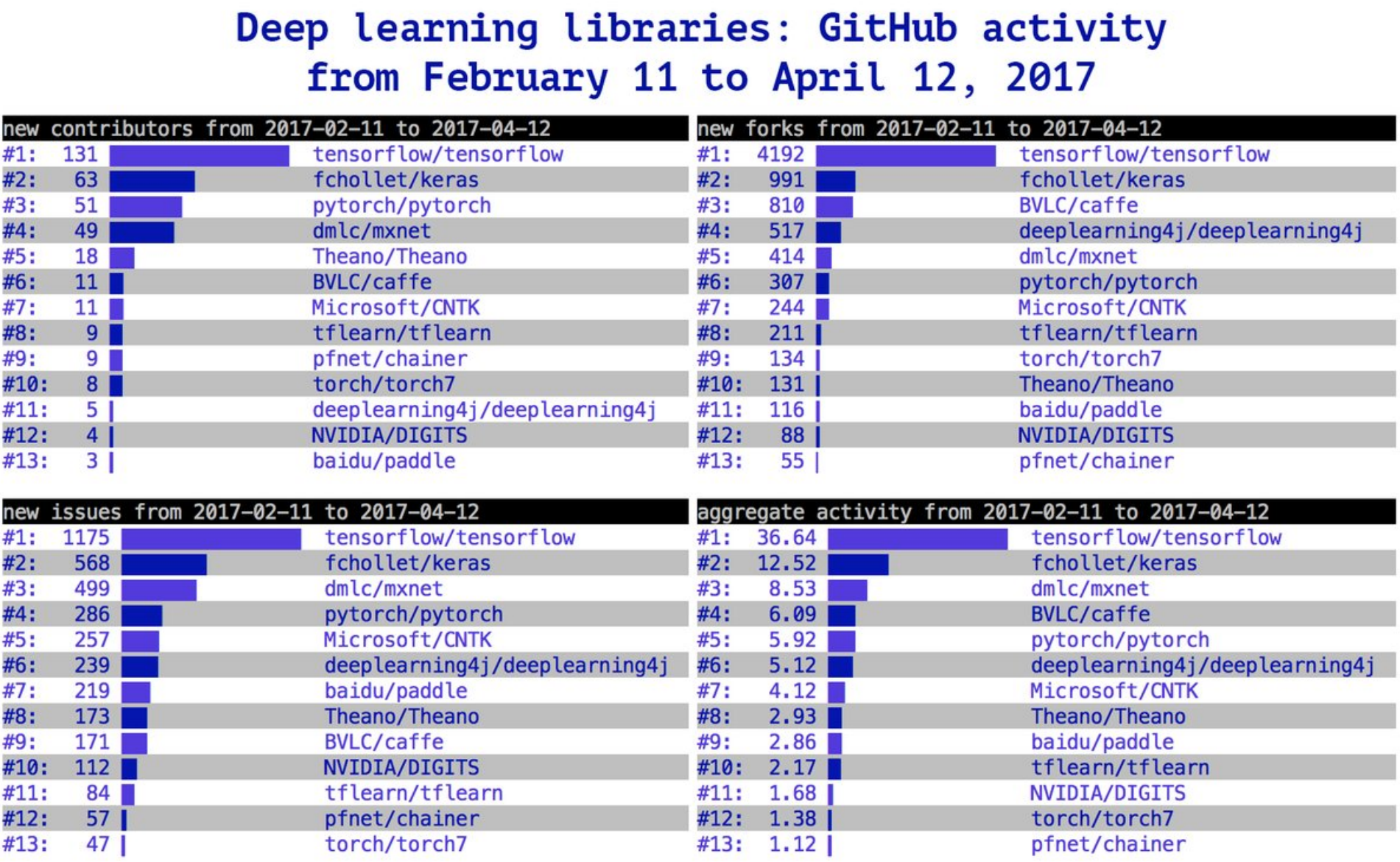
Deep Learning Software

- Keras
  - Deep Learning library for Theano and TensorFlow
- Tensorflow
  - TensorFlow™ is an open source software library for numerical computation using data flow graphs.
- Theano
  - CPU/GPU symbolic expression compiler in python (from MILA lab at University of Montreal)

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

Deep Learning with Google TensorFlow

Deep Learning Libraries: Tensorflow and Keras

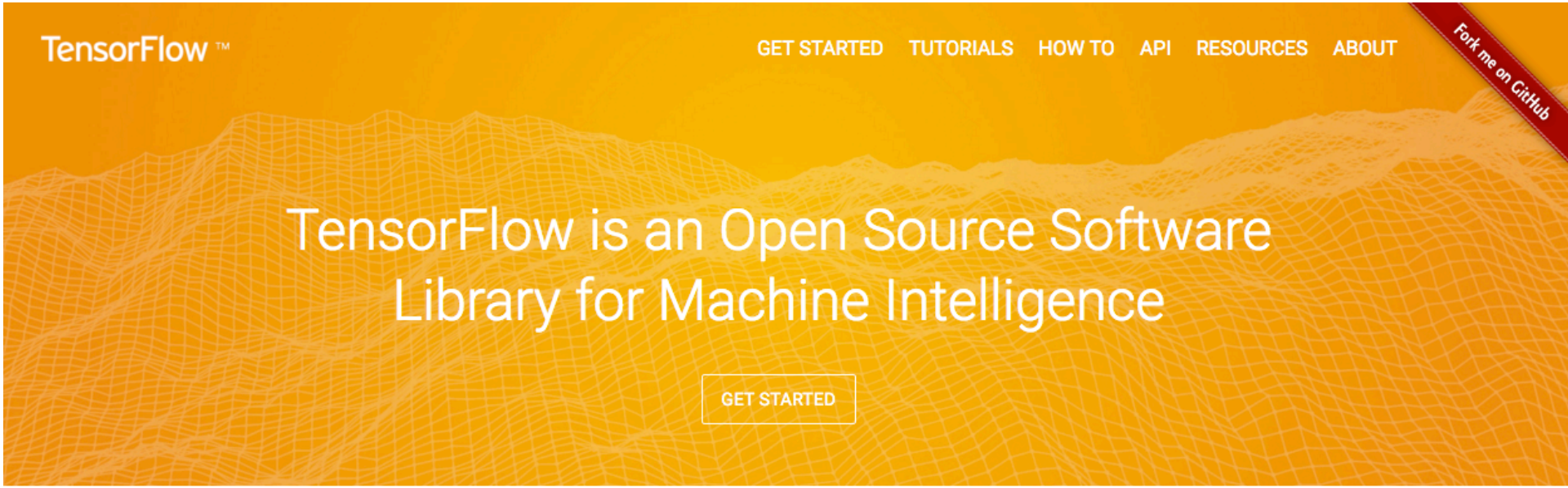


Source: https://twitter.com/fchollet/status/852194634470223873

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

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/

TensorFlow

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/

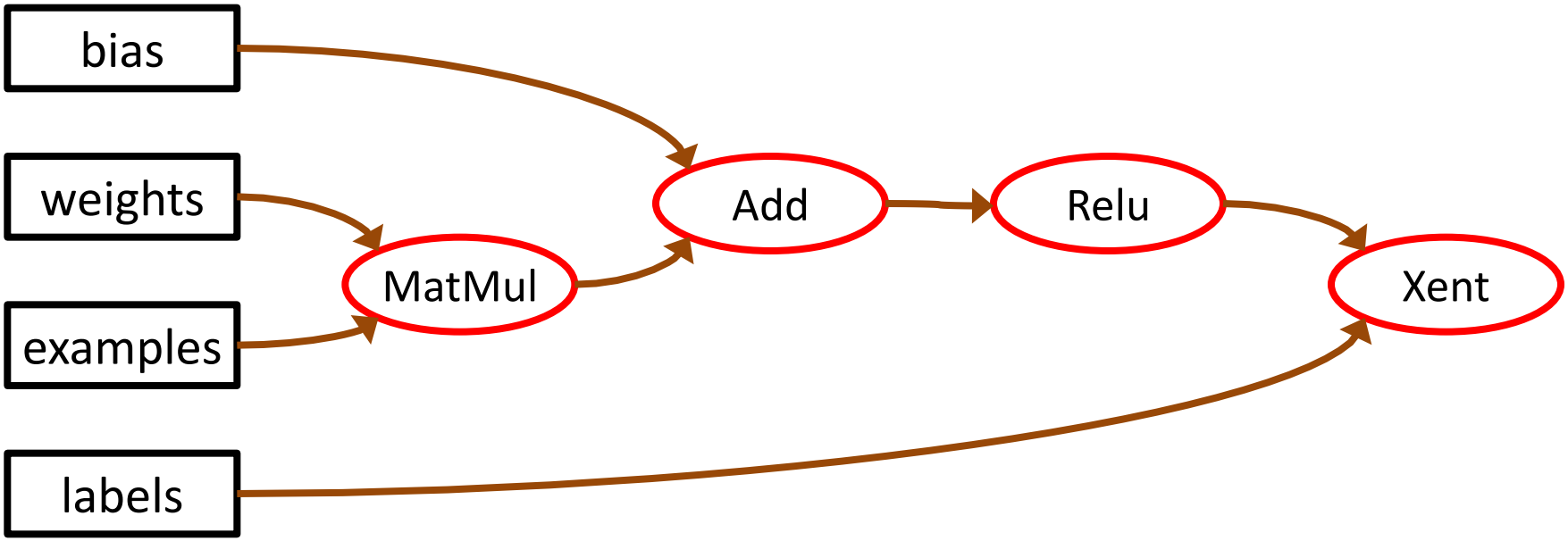
TensorFlow

Nodes:
mathematical operations

edges:
multidimensional data arrays (tensors)
communicated between nodes

Computation is a Dataflow Graph

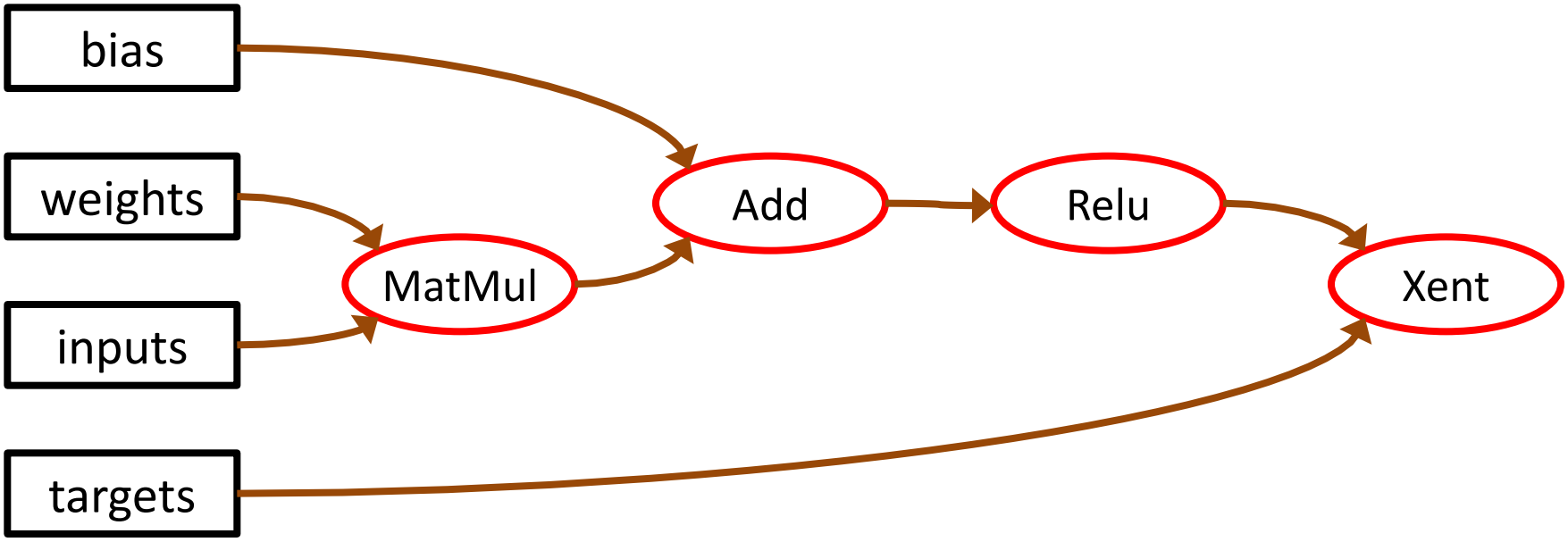
Graph of Nodes,
also called Operations or ops.



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

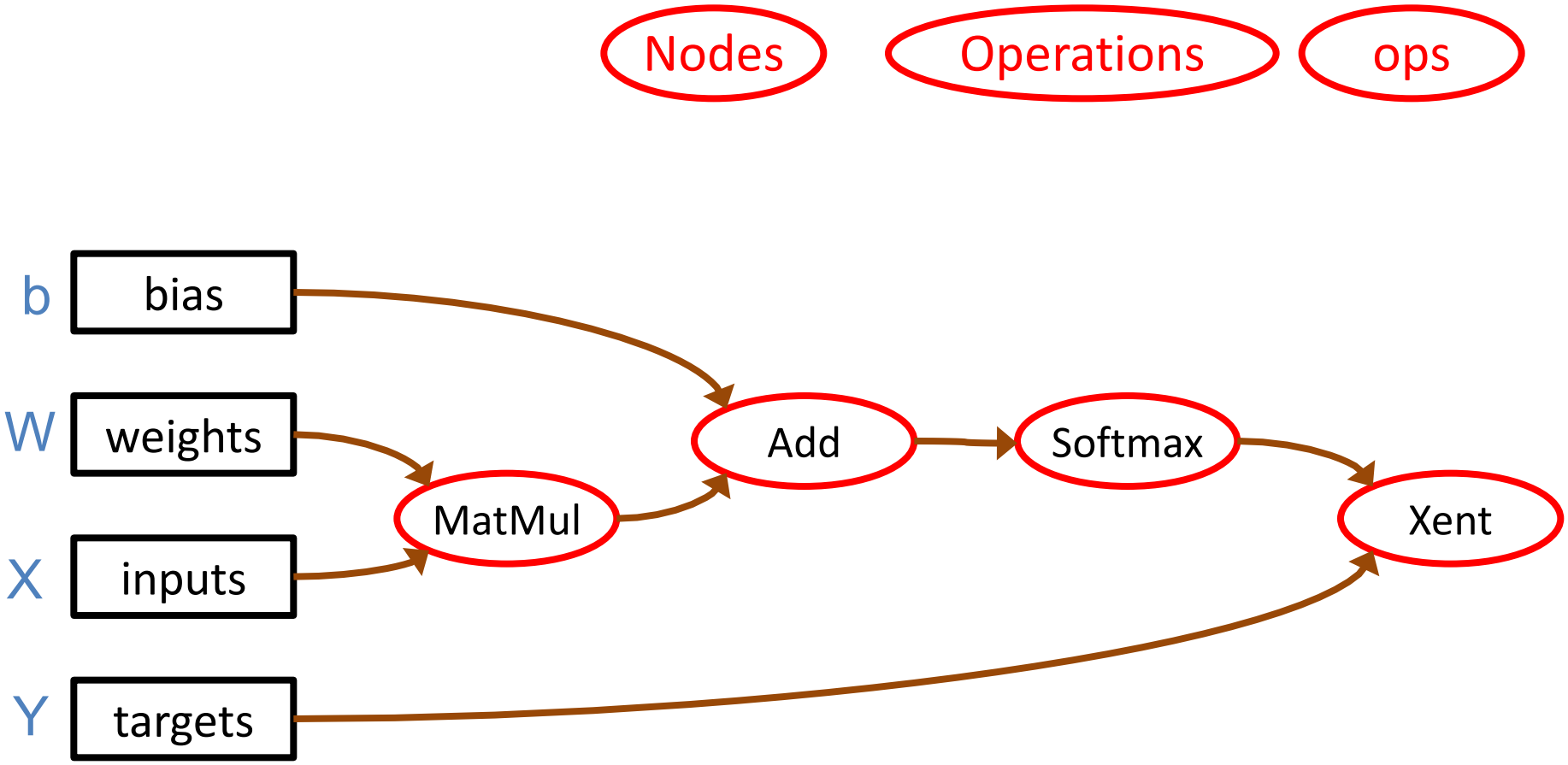
Computation is a Dataflow Graph

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

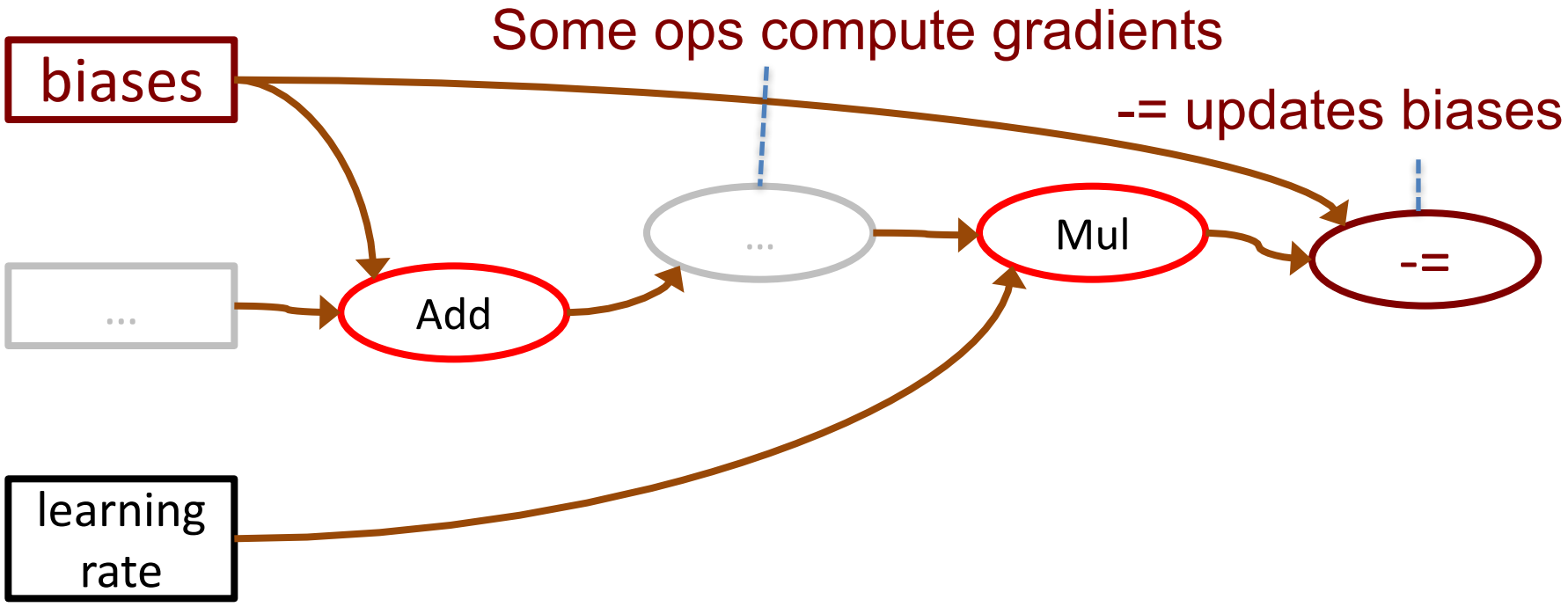


Edges are N-dimensional arrays: **Tensors**

Computation is a Dataflow Graph

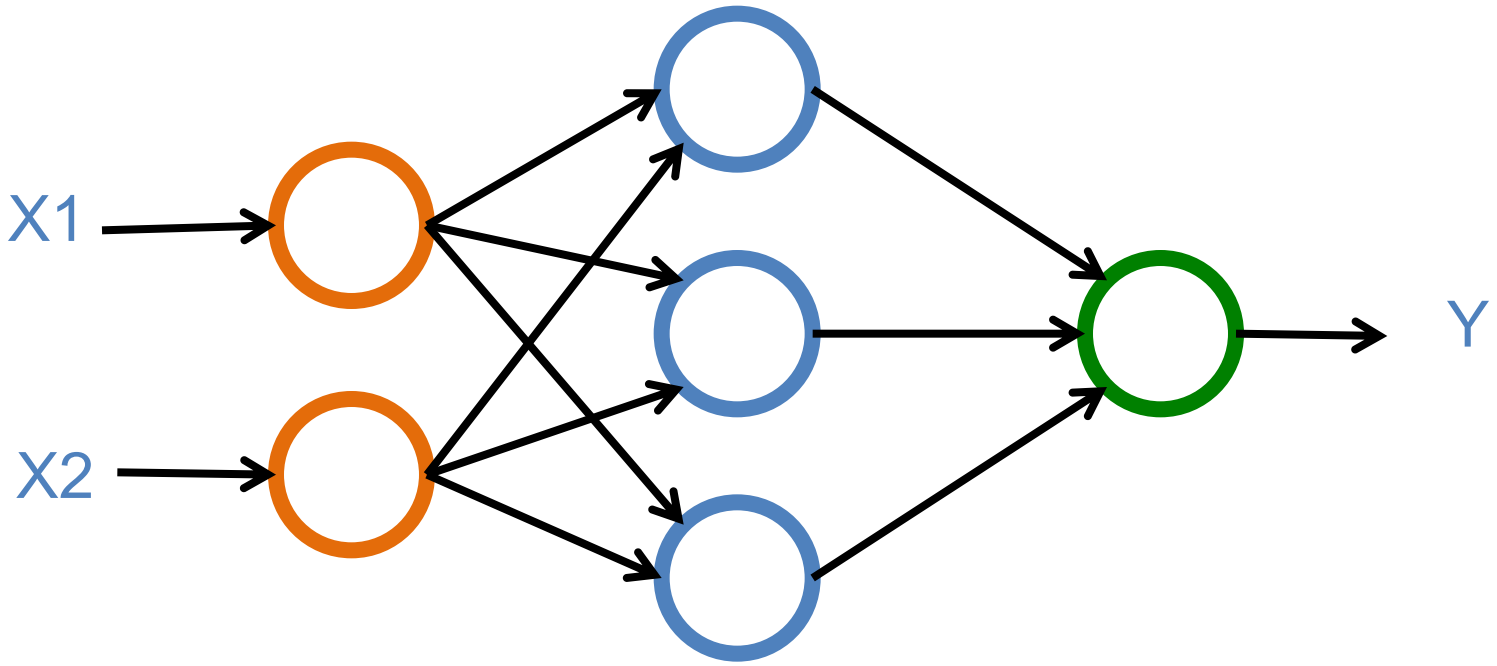
with state

‘Biases’ is a variable



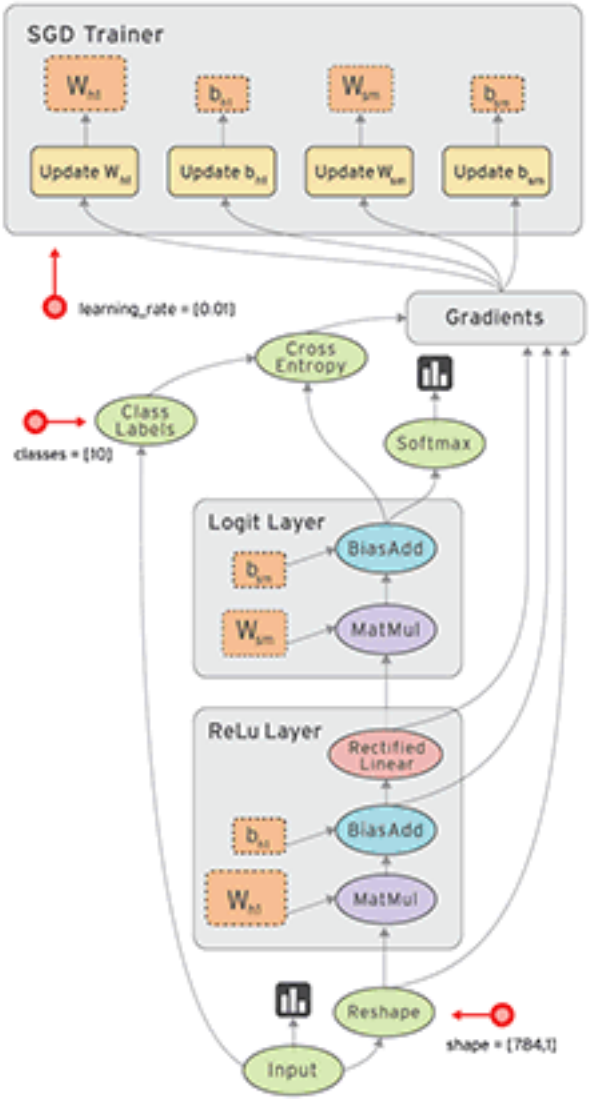
Neural Networks

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



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

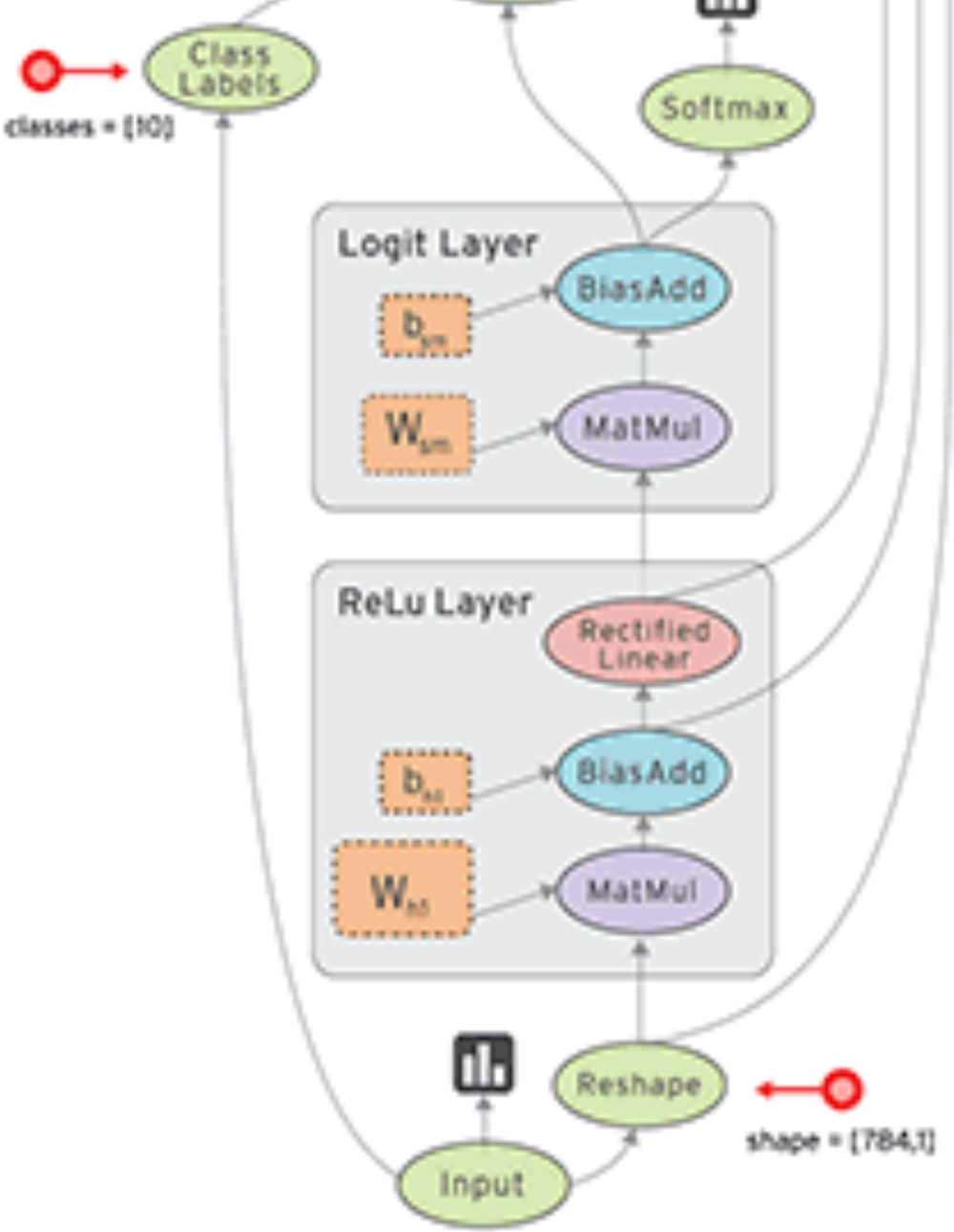
Data Flow Graph



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

TensorFlow

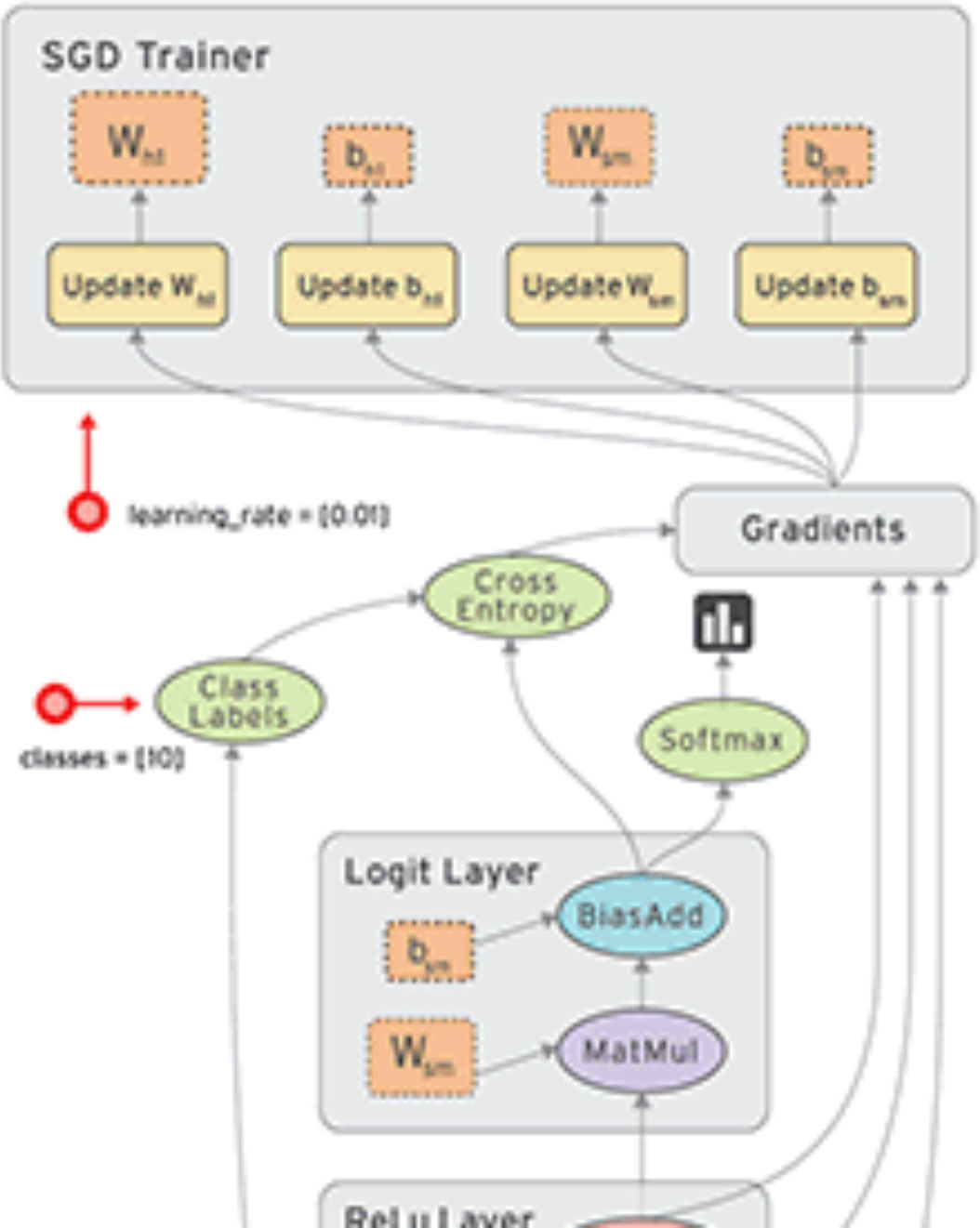
Data Flow Graph



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

TensorFlow

Data Flow Graph



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 keras
```

```
pip install keras
```

```
pip install tensorflow
```

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

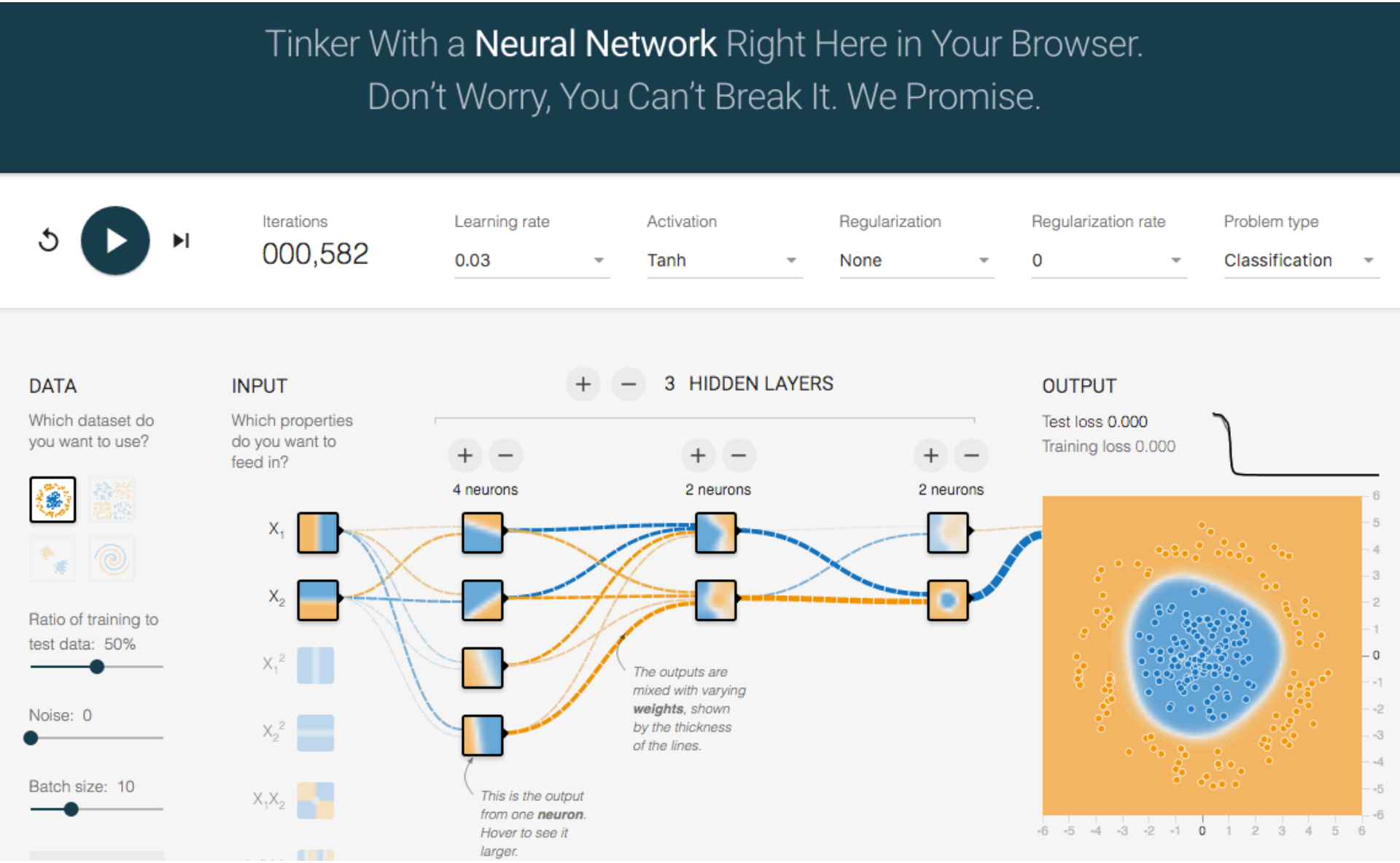
Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt

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

pip install tensorflow

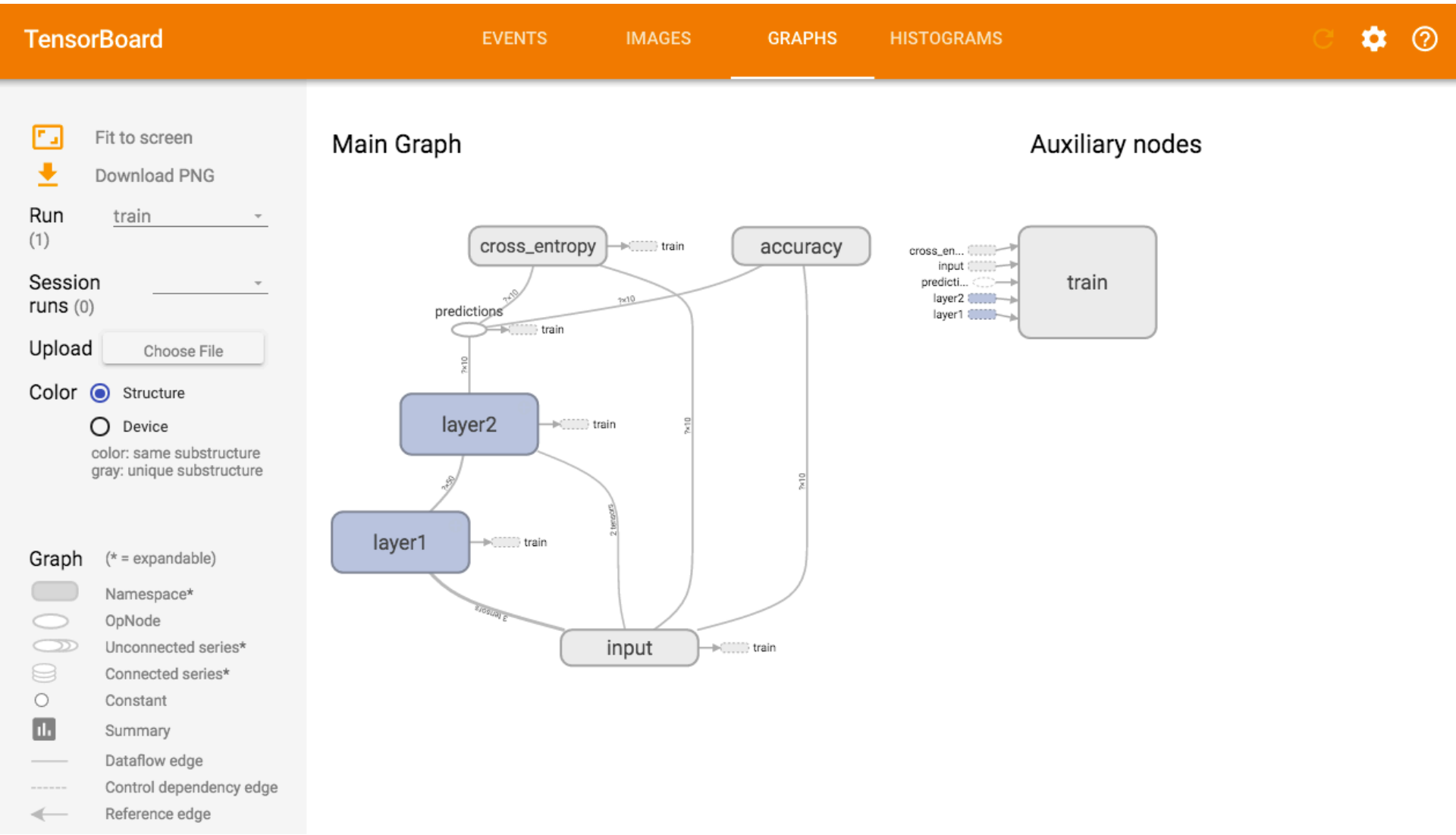
```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement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

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

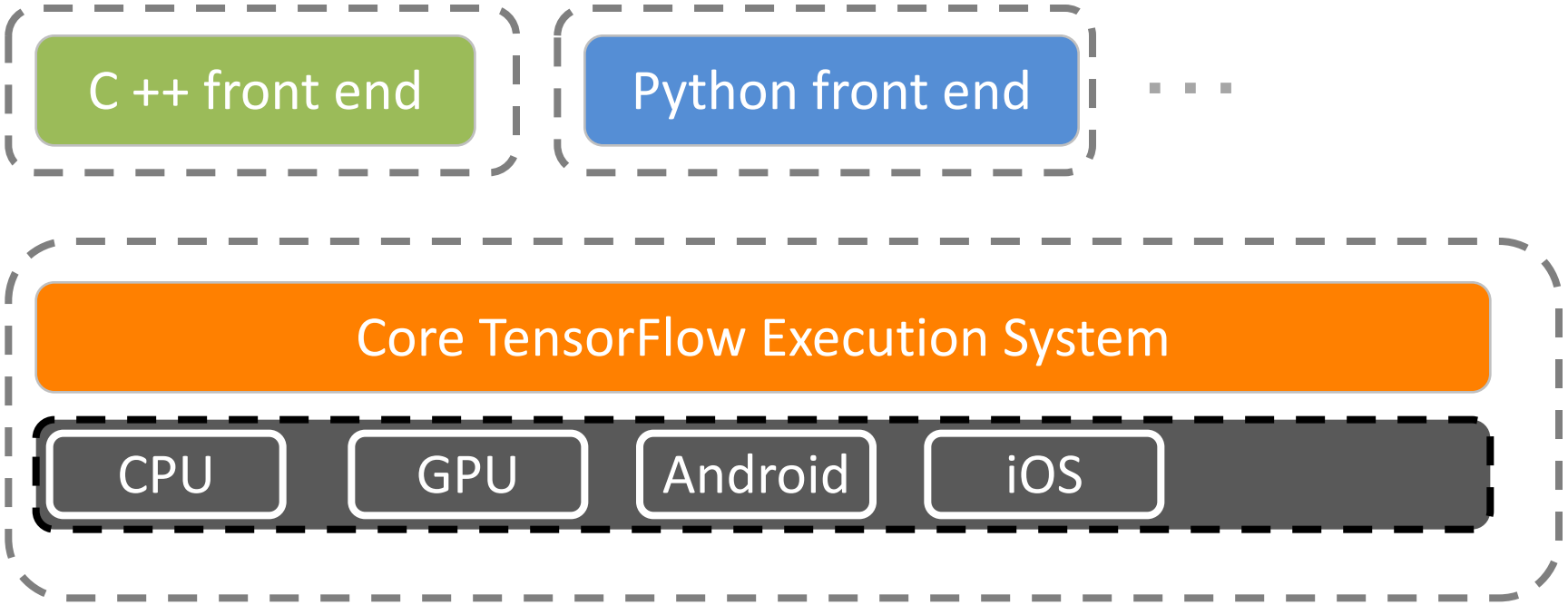
Try your first TensorFlow

```
$ 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

Architecture of TensorFlow



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

TensorFlow
and
Deep Learning

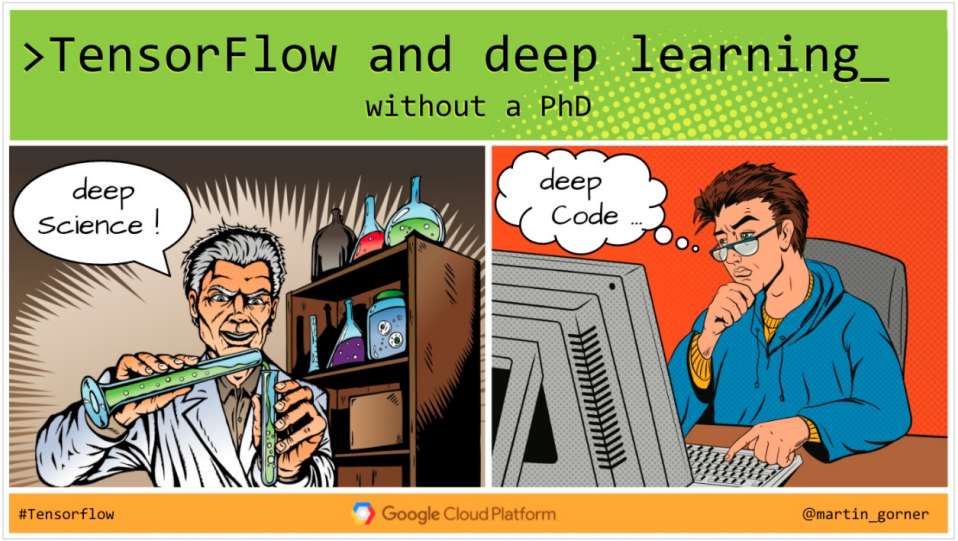
TensorFlow and Deep Learning

1 Overview
2 Preparation: Install TensorFlow, get the sample code
3 Theory: train a neural network
4 Theory: a 1-layer neural network
5 Theory: gradient descent
6 Lab: let's jump into the code
7 Lab: adding layers
8 Lab: special care for deep networks
9 Lab: learning rate decay
10 Lab: dropout, overfitting
11 Theory: convolutional networks

Did you find a mistake? Please file a bug.

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

1. Overview



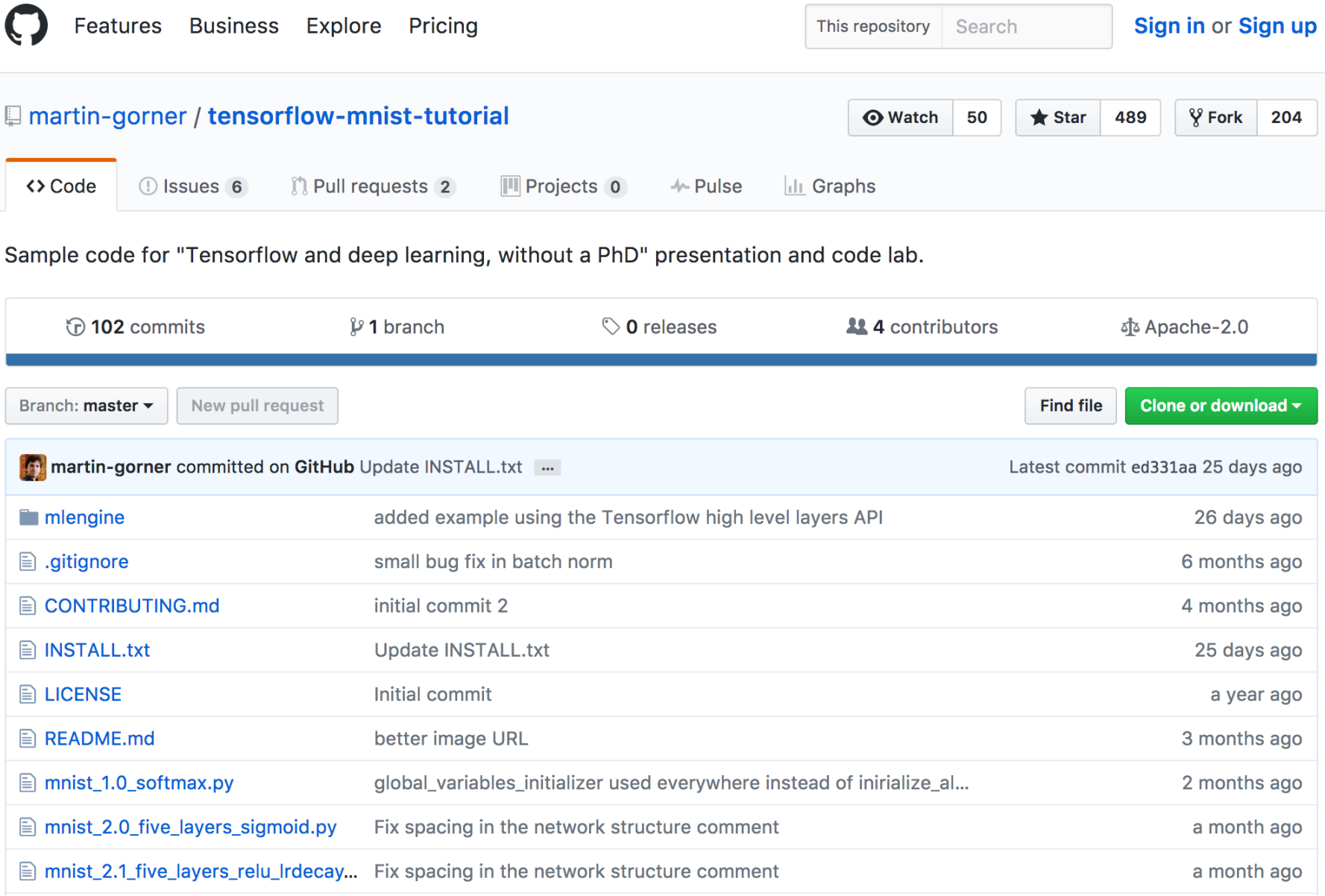
In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

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

What you'll learn

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

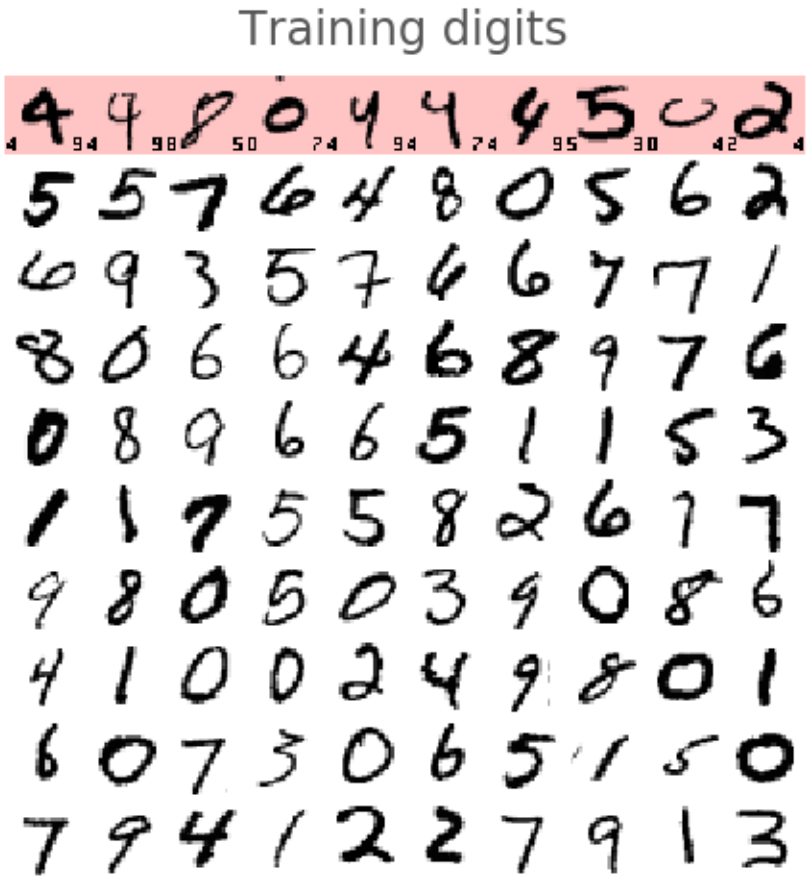
```
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git
```

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

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

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

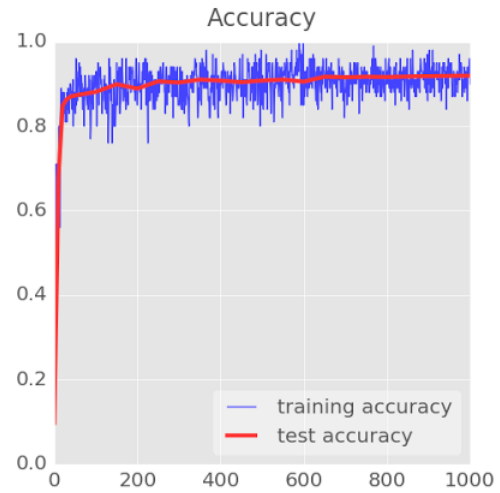
MNIST dataset: 60,000 labeled digits

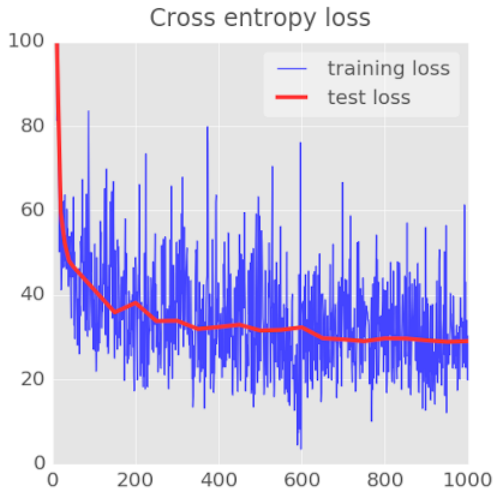


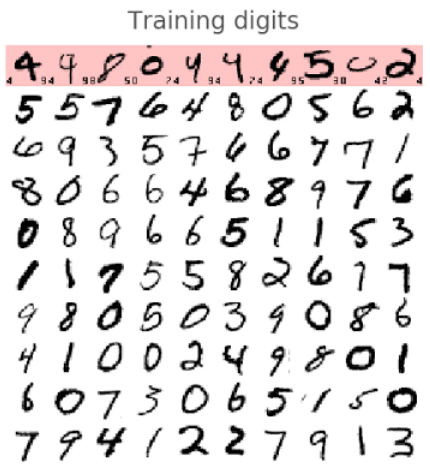
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

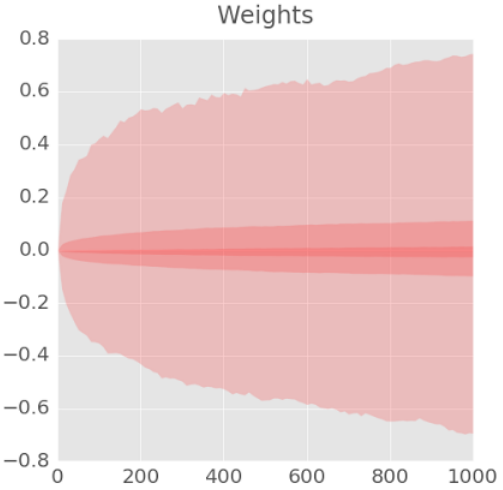
```
cd tensorflow-mnist-tutorial
```

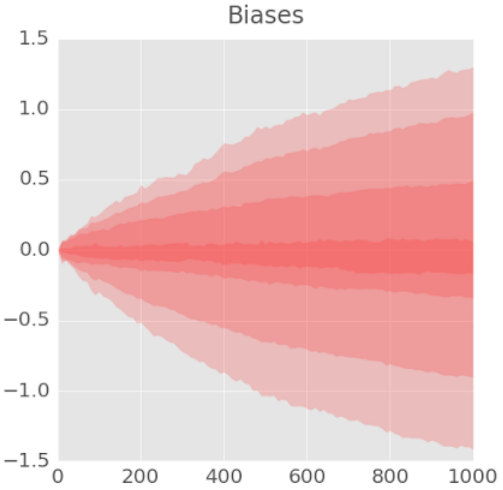
```
python3 mnist_1.0_softmax.py
```

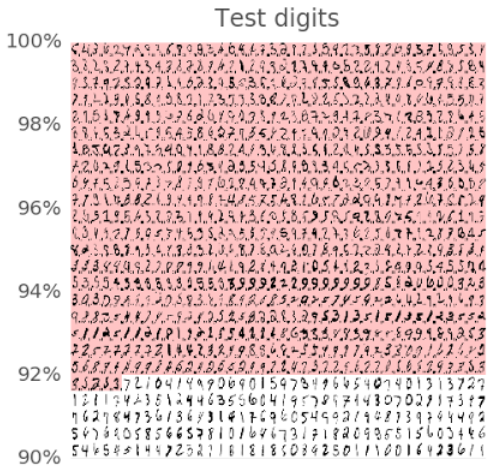












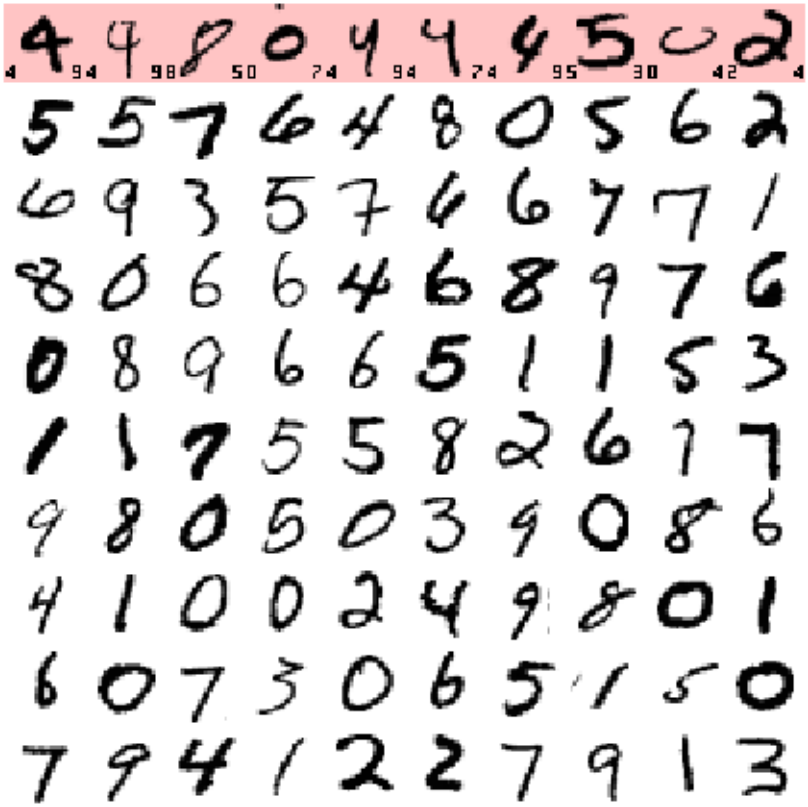
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#1

Train a Neural Network

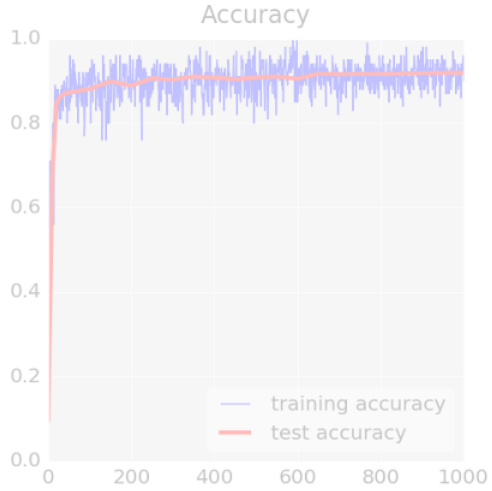
Training digits
updates to weights and biases =>
better recognition (loop)

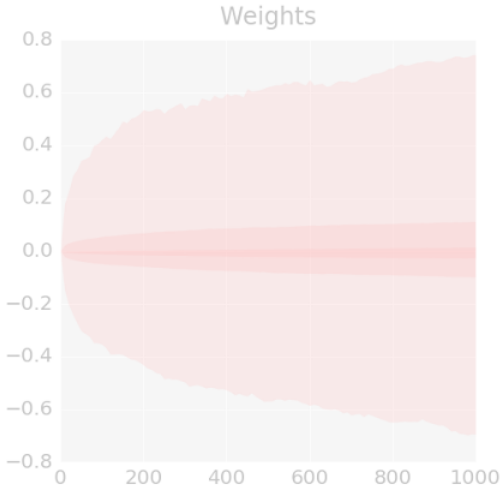
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

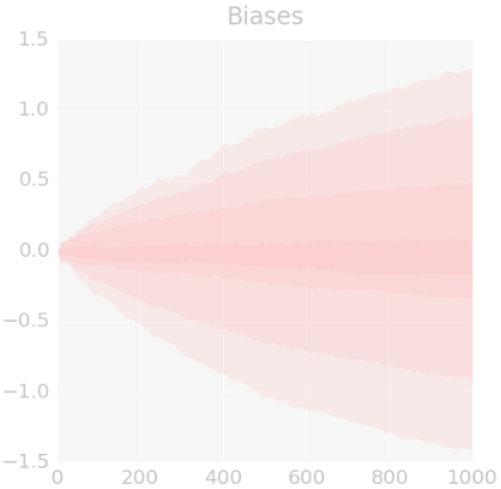
Training digits



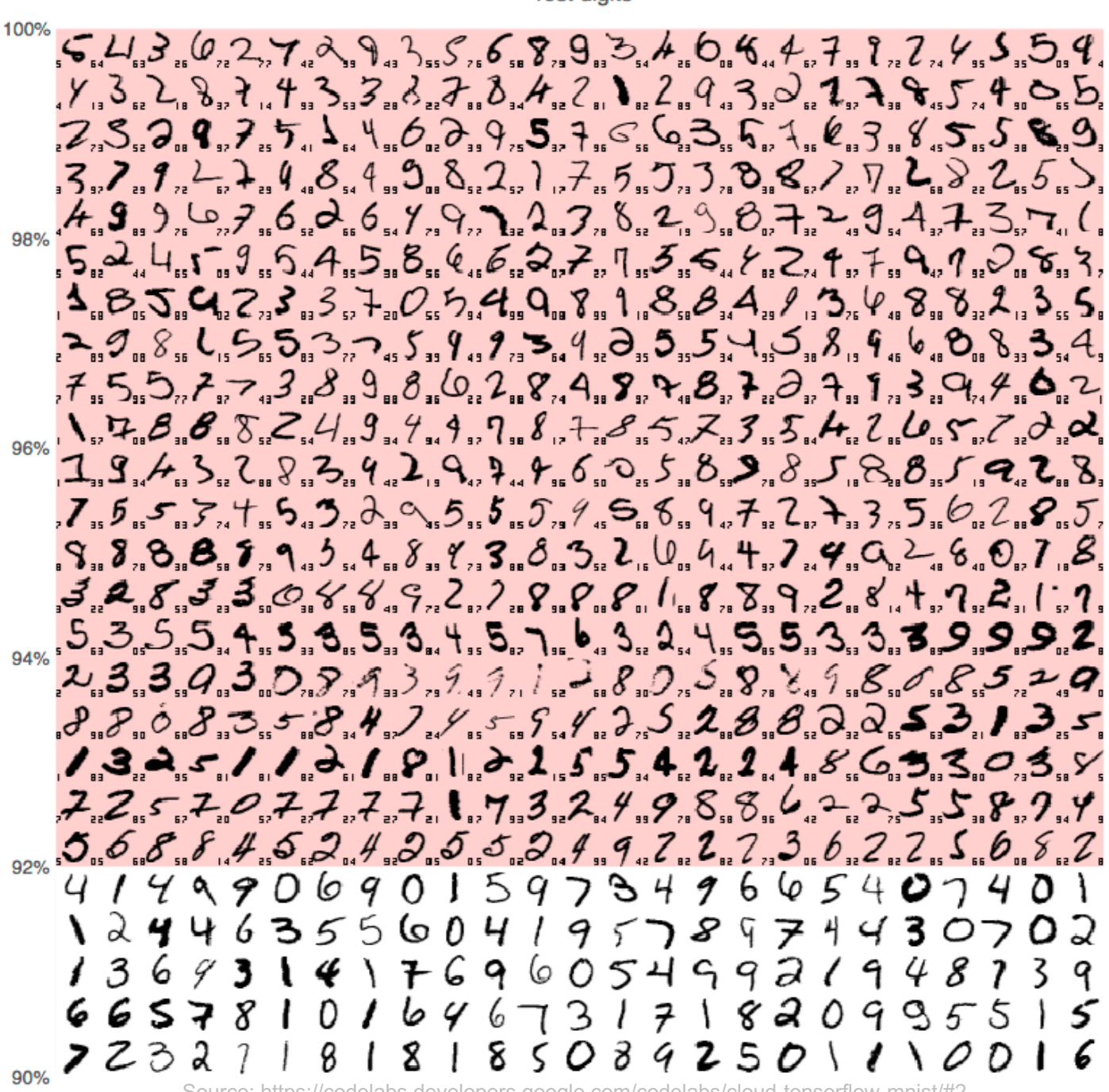
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2



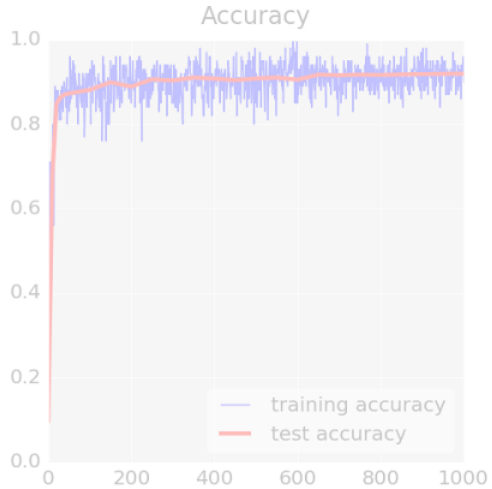


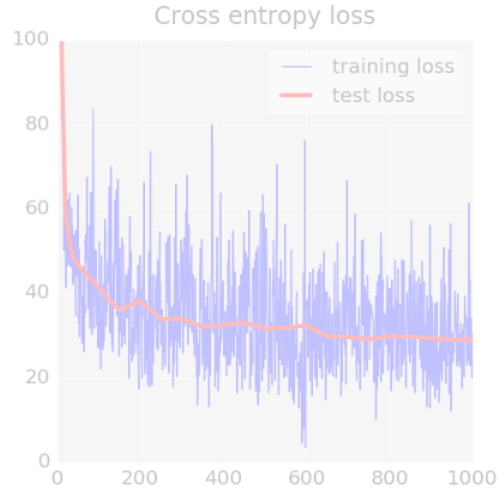


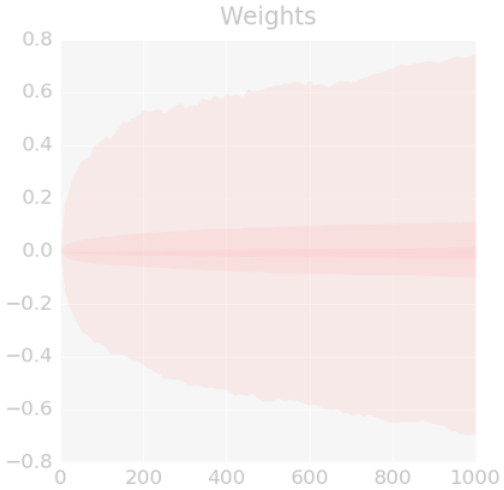
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

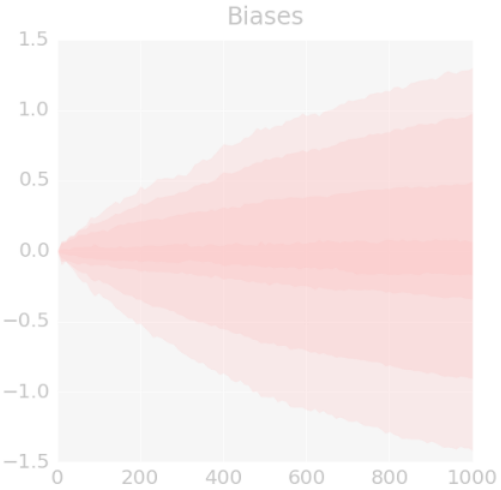


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

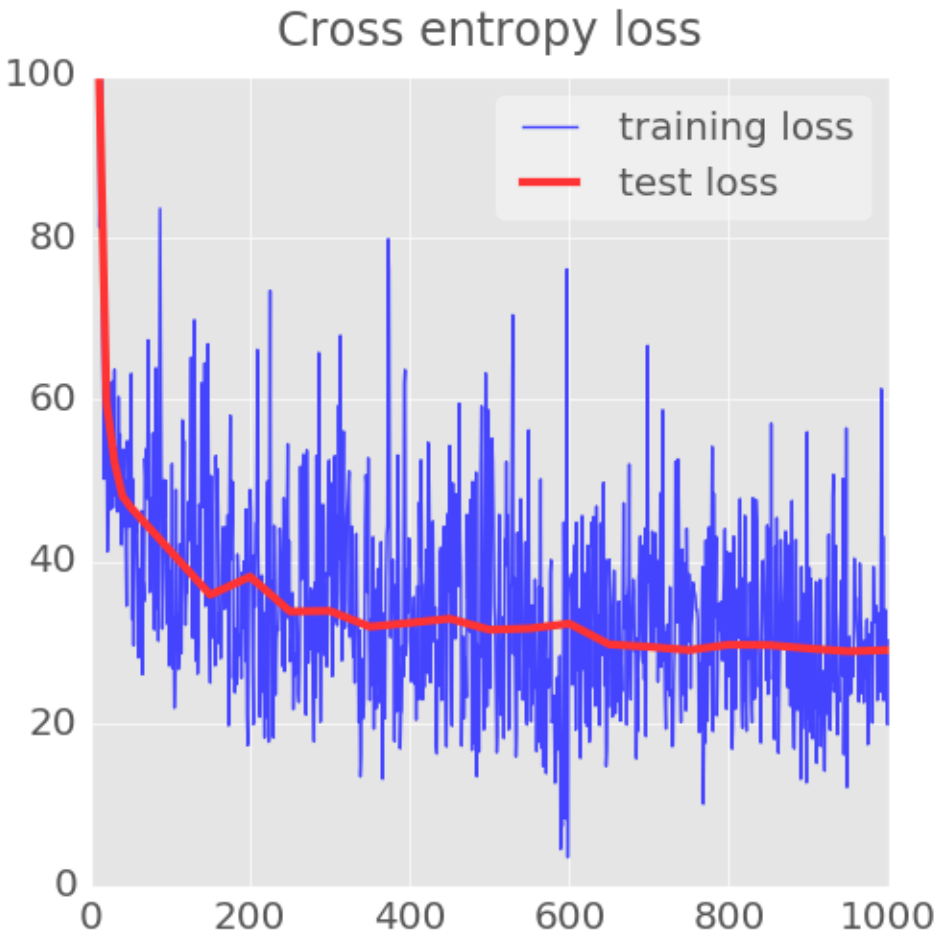




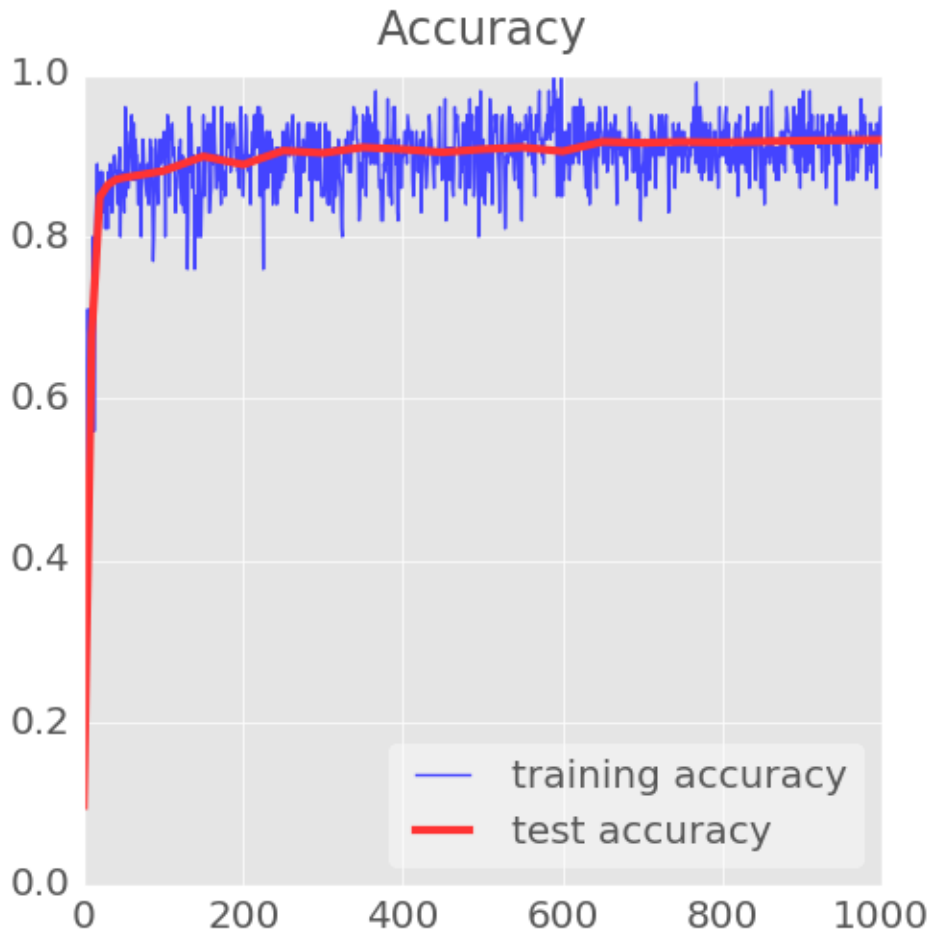




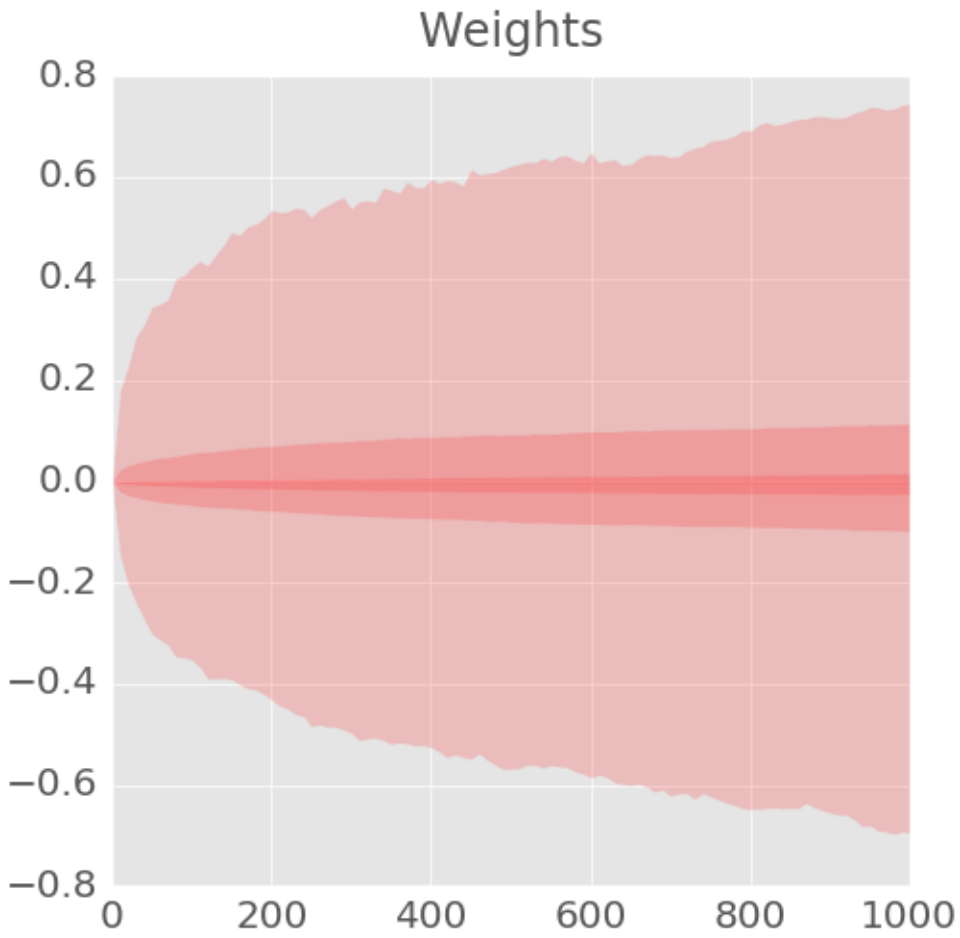
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2



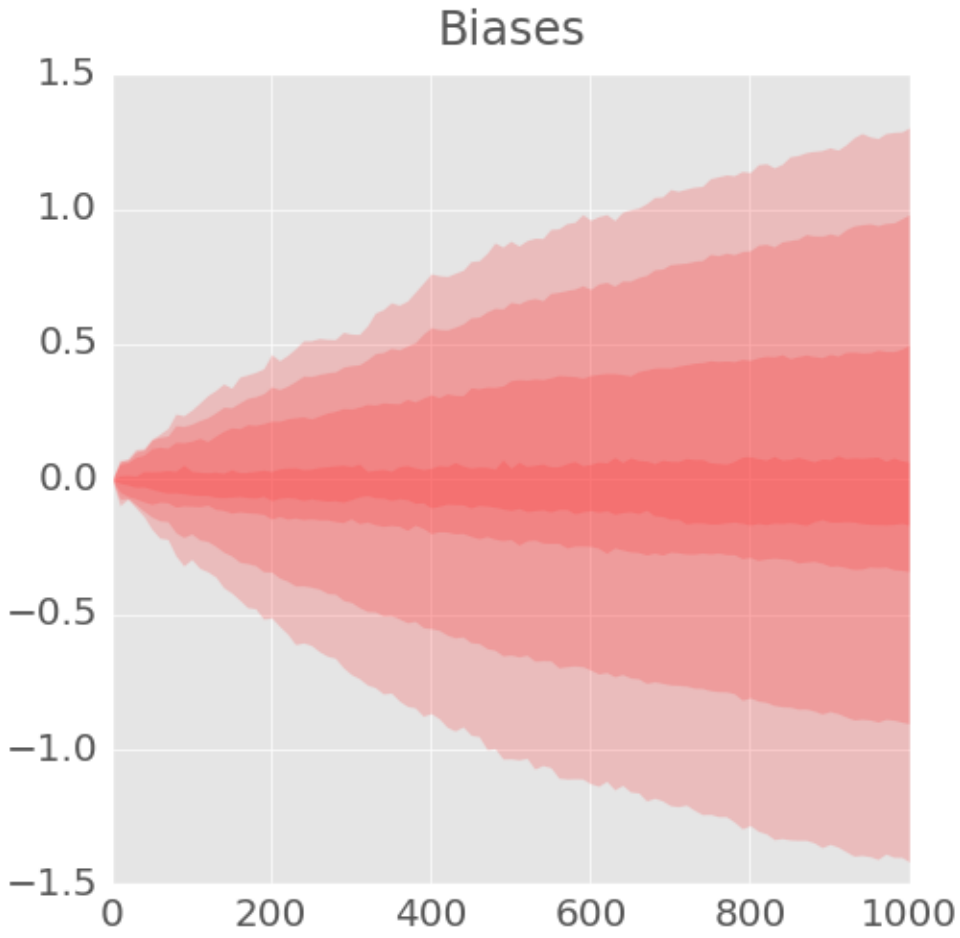
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2



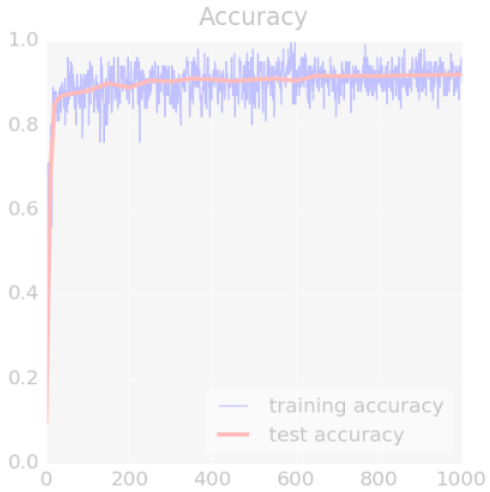
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

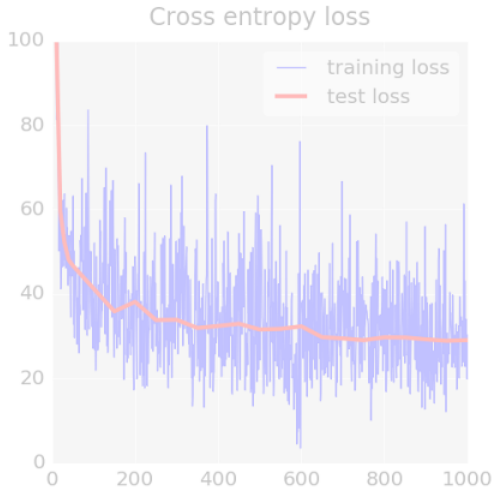


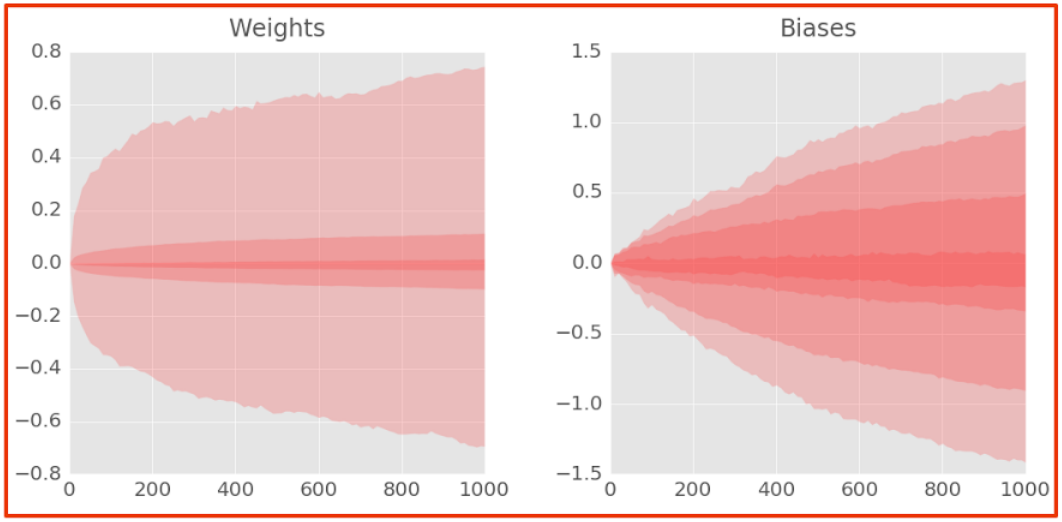
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

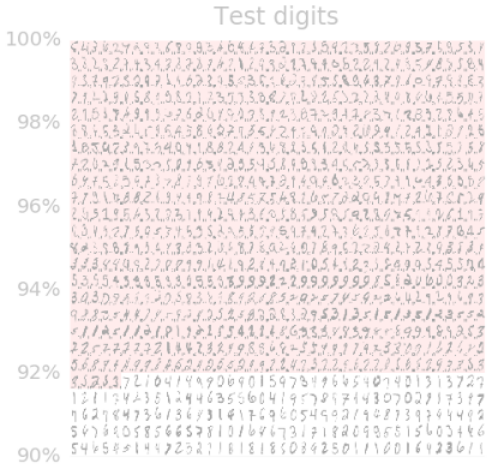


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2







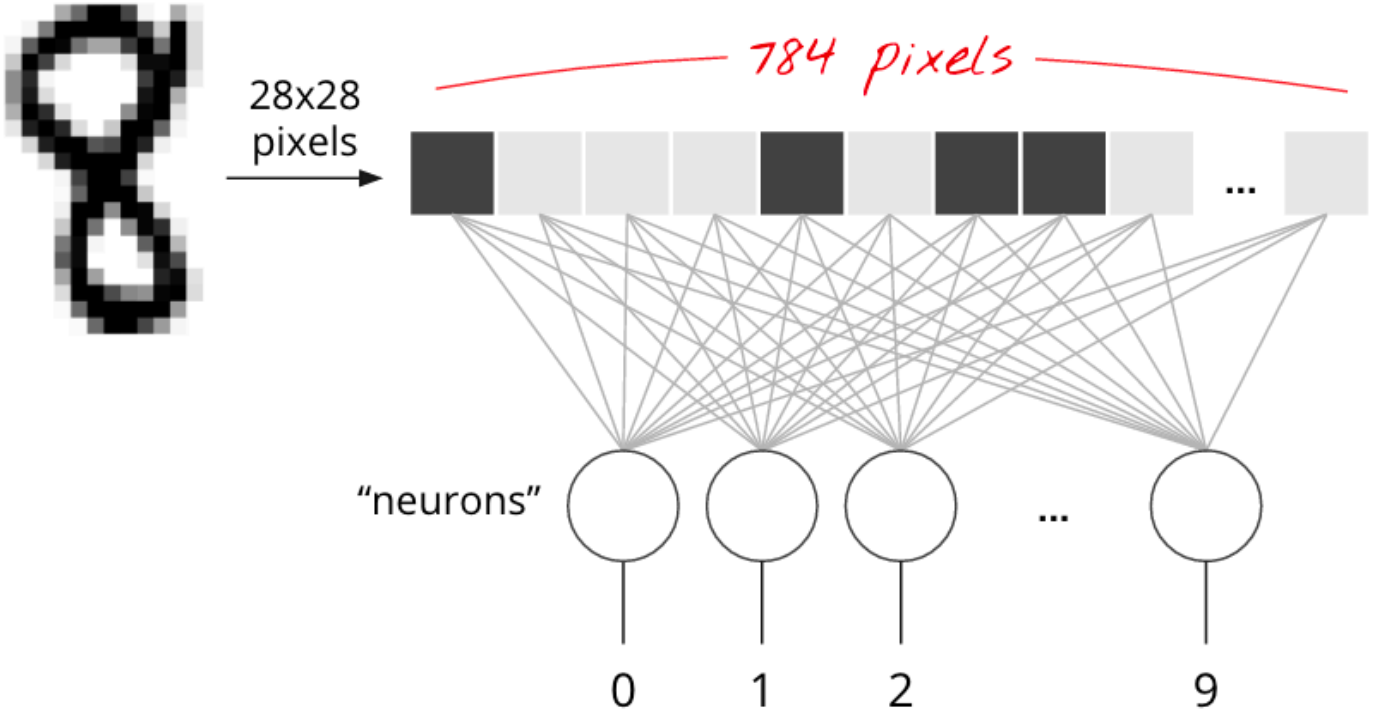


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Cookbook

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

Very Simple Model: Softmax Classification



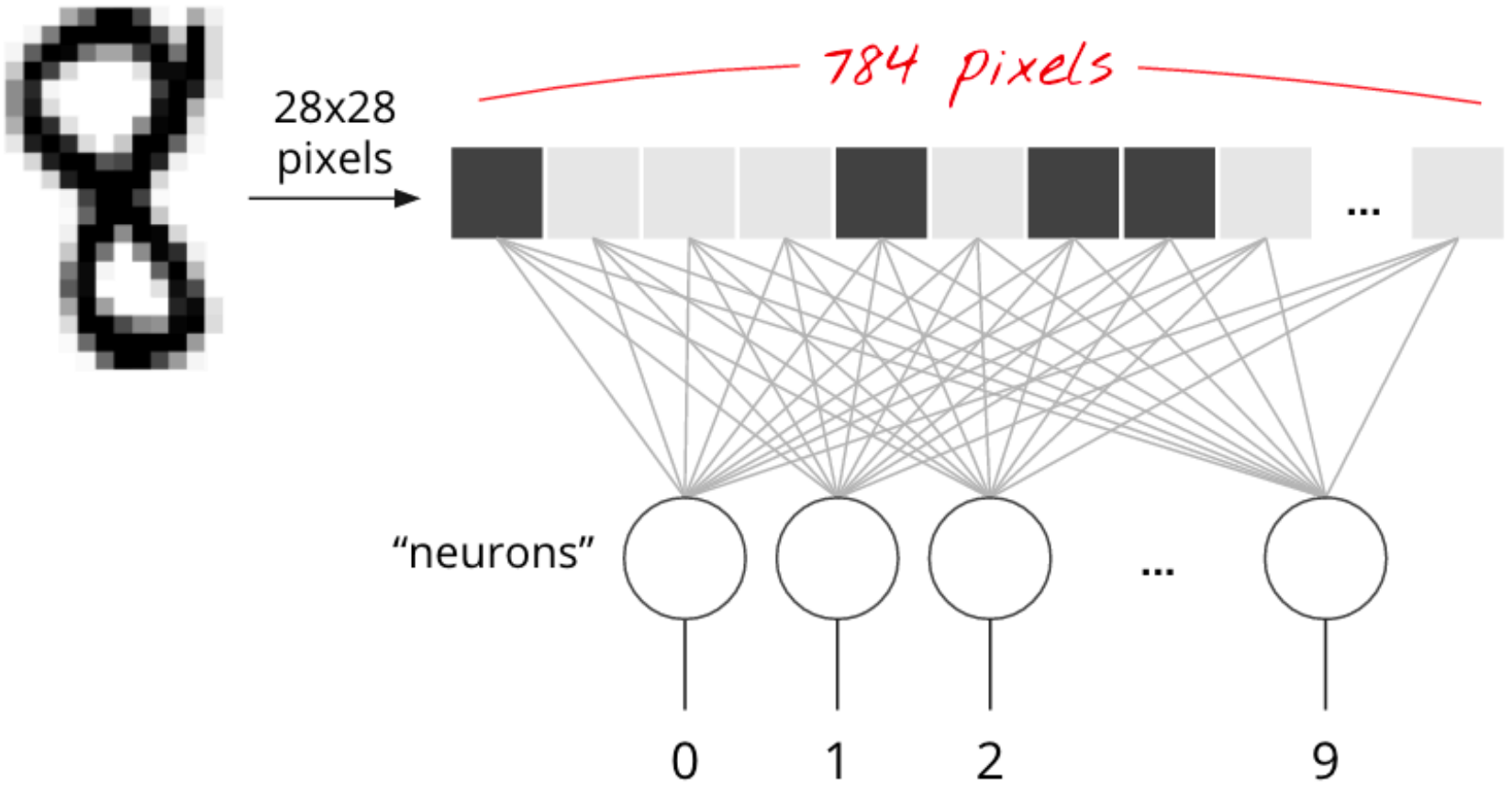
weighted sum of all pixels + bias

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

neuron outputs

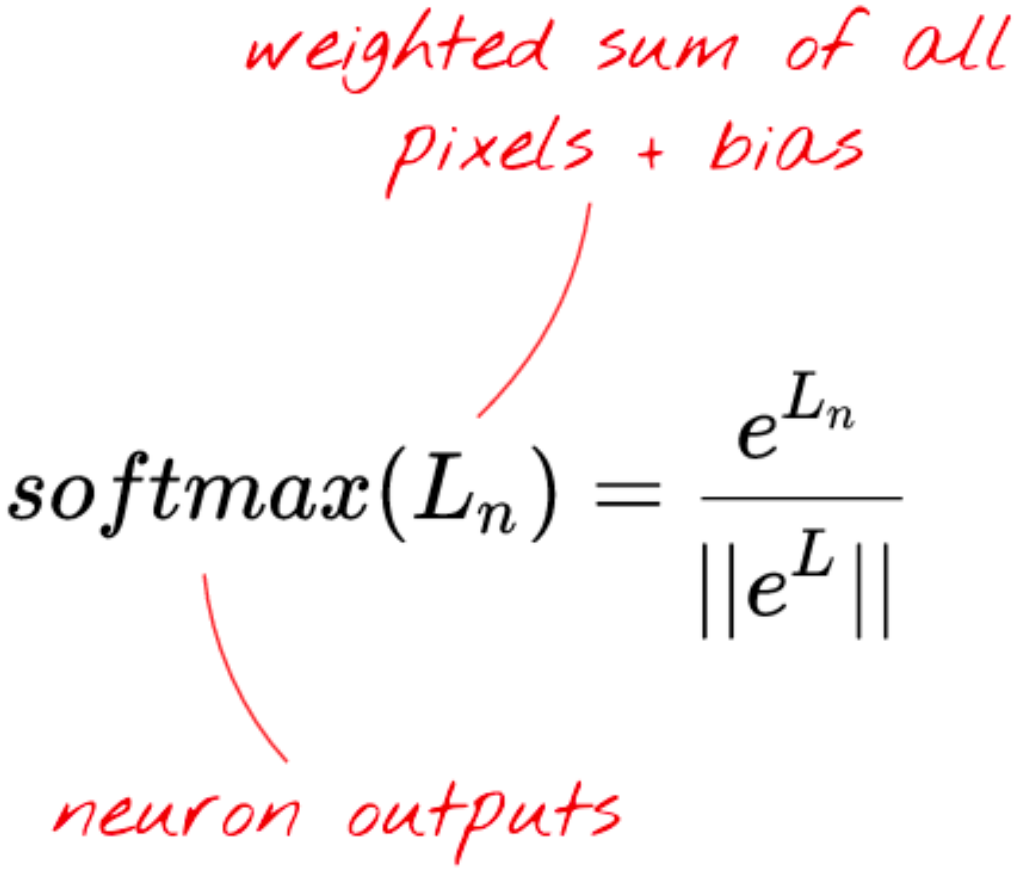
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Very Simple Model: Softmax Classification



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

Very Simple Model: Softmax Classification



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

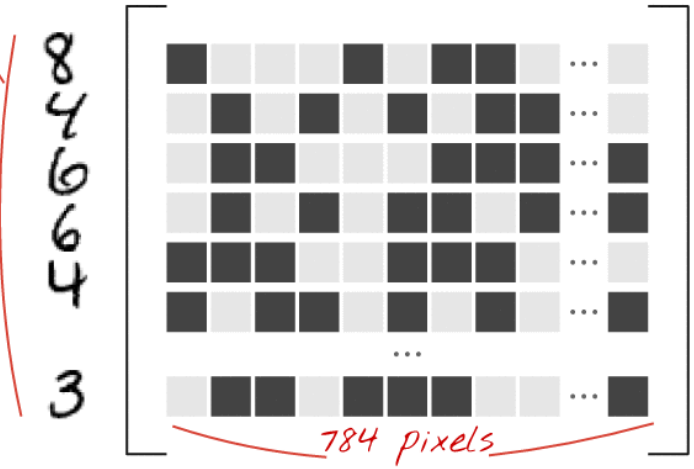
In Matrix notation, 100 images at a time

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

10 columns

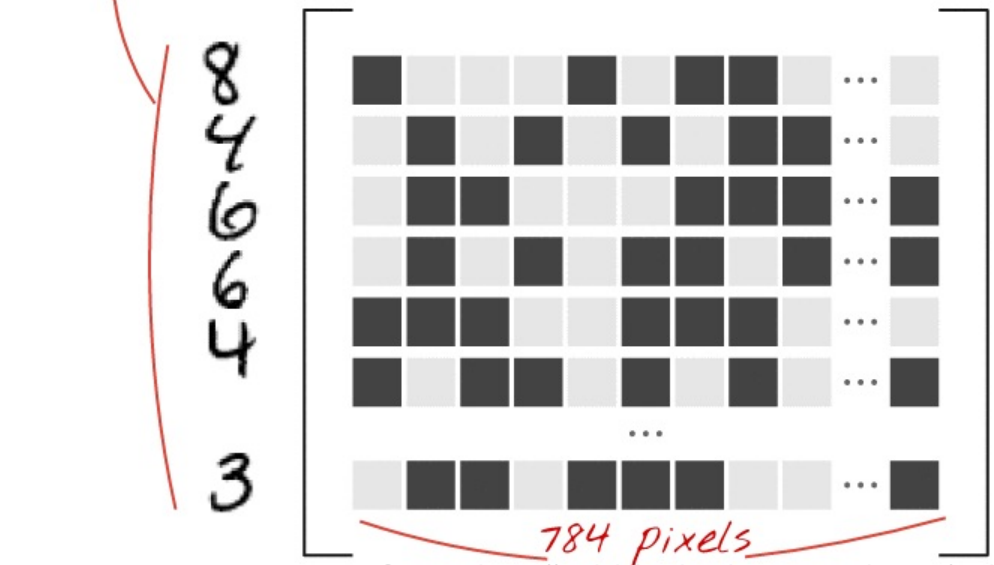
784 lines

X: 100 images, one per line, flattened



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

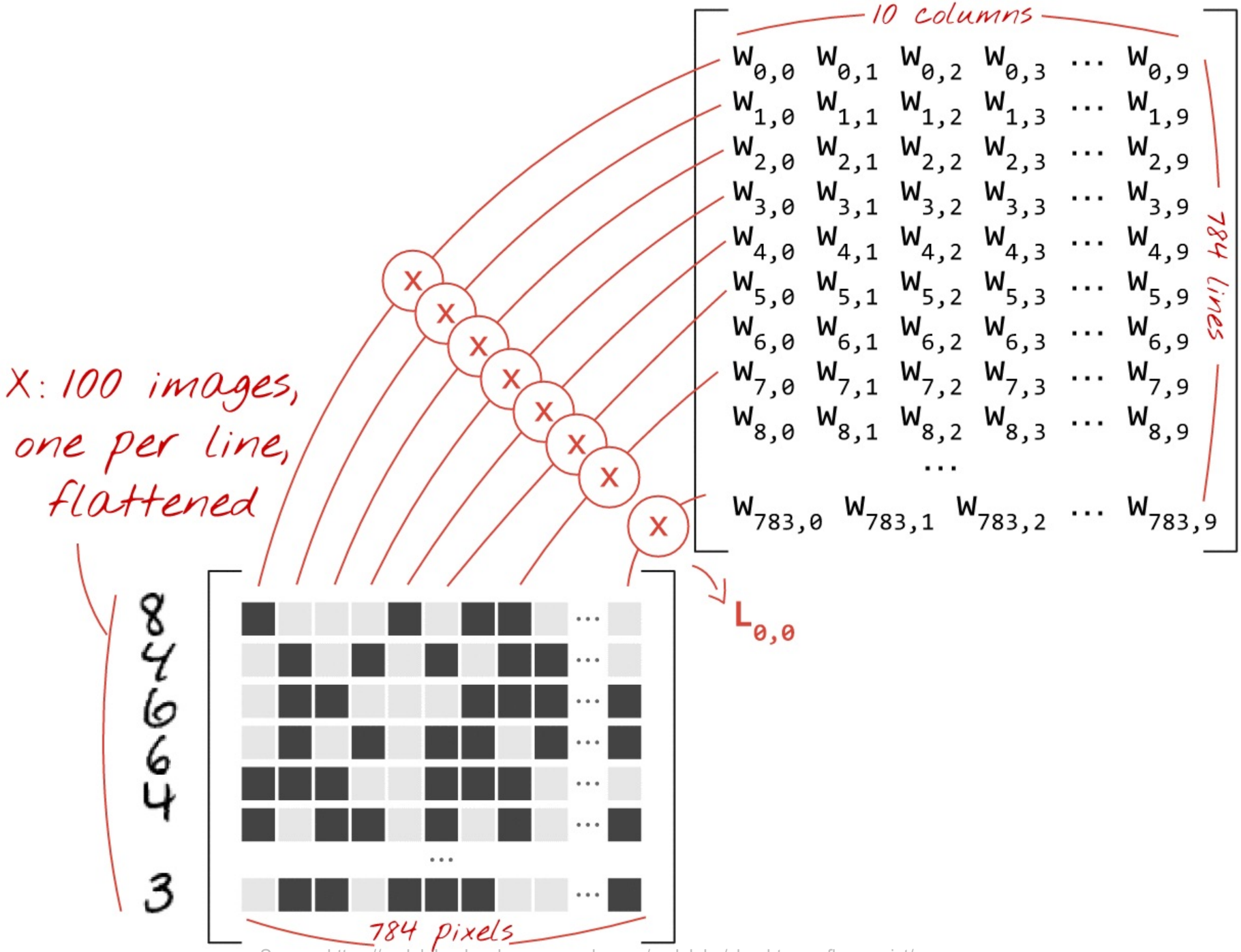
X: 100 images, one per line, flattened

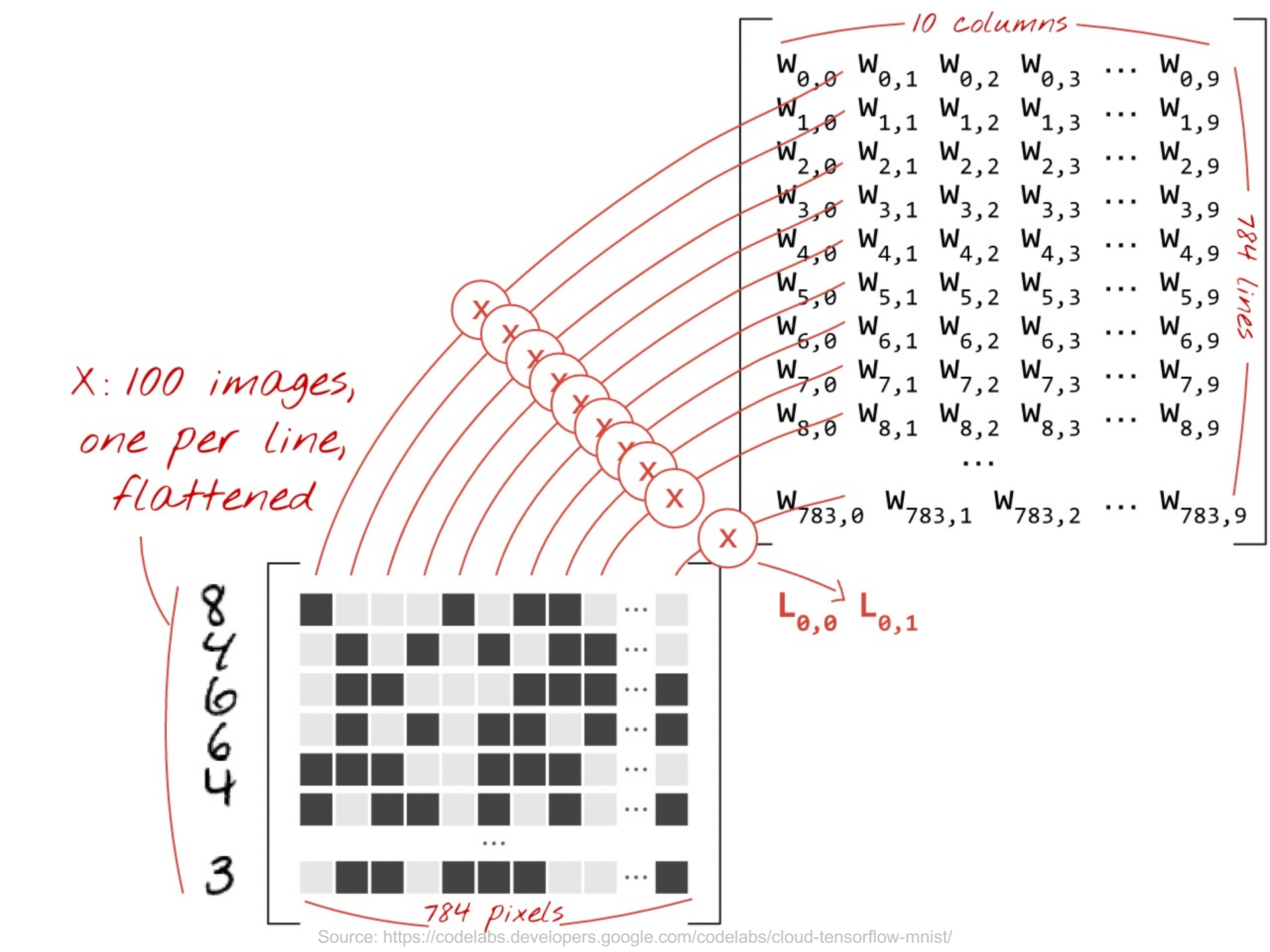


$$\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \dots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \dots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \dots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \dots & w_{3,9} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \dots &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

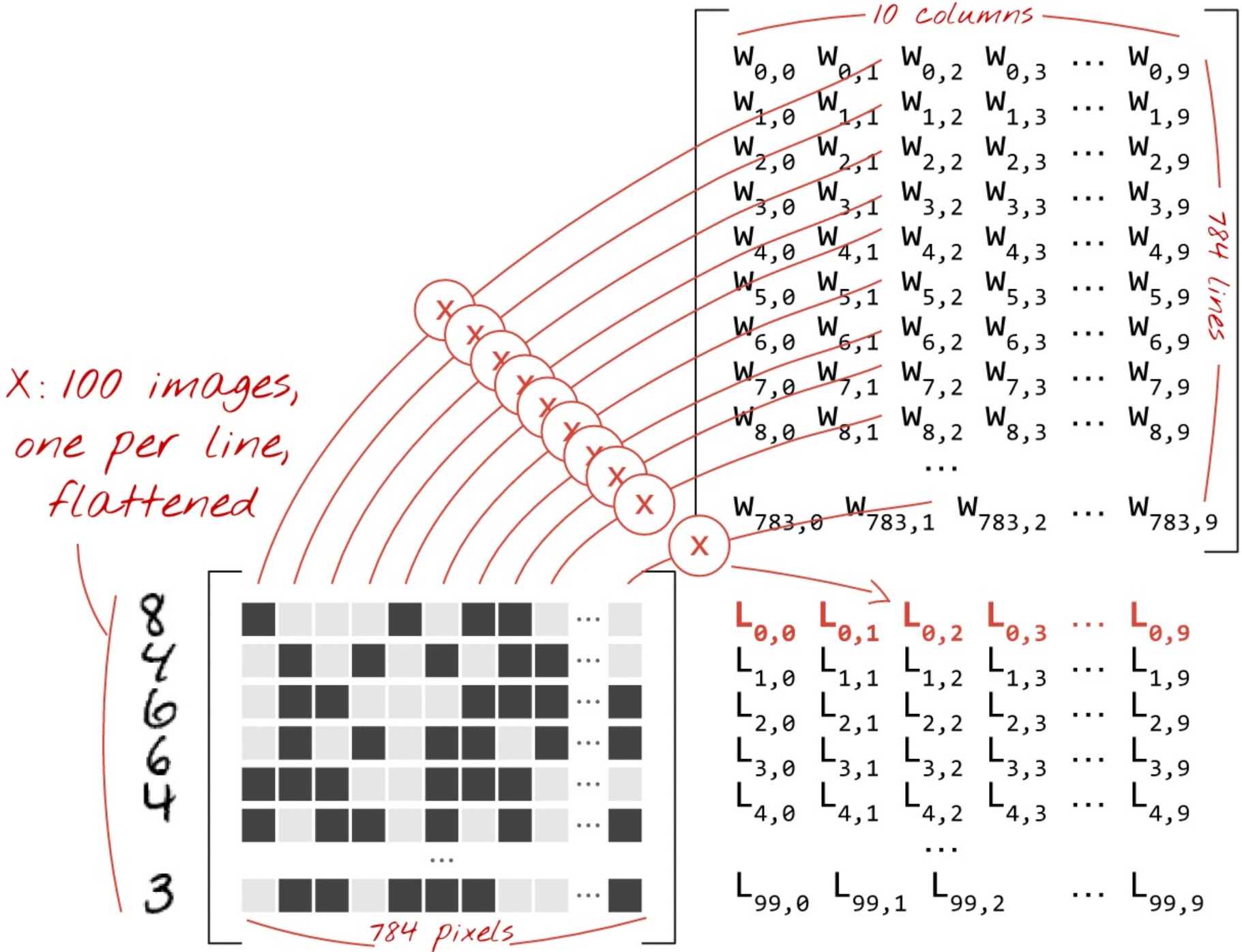
10 columns

784 lines


10 columns
784 lines
X: 100 images, one per line, flattened
784 pixels
$L_{0,0}$
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



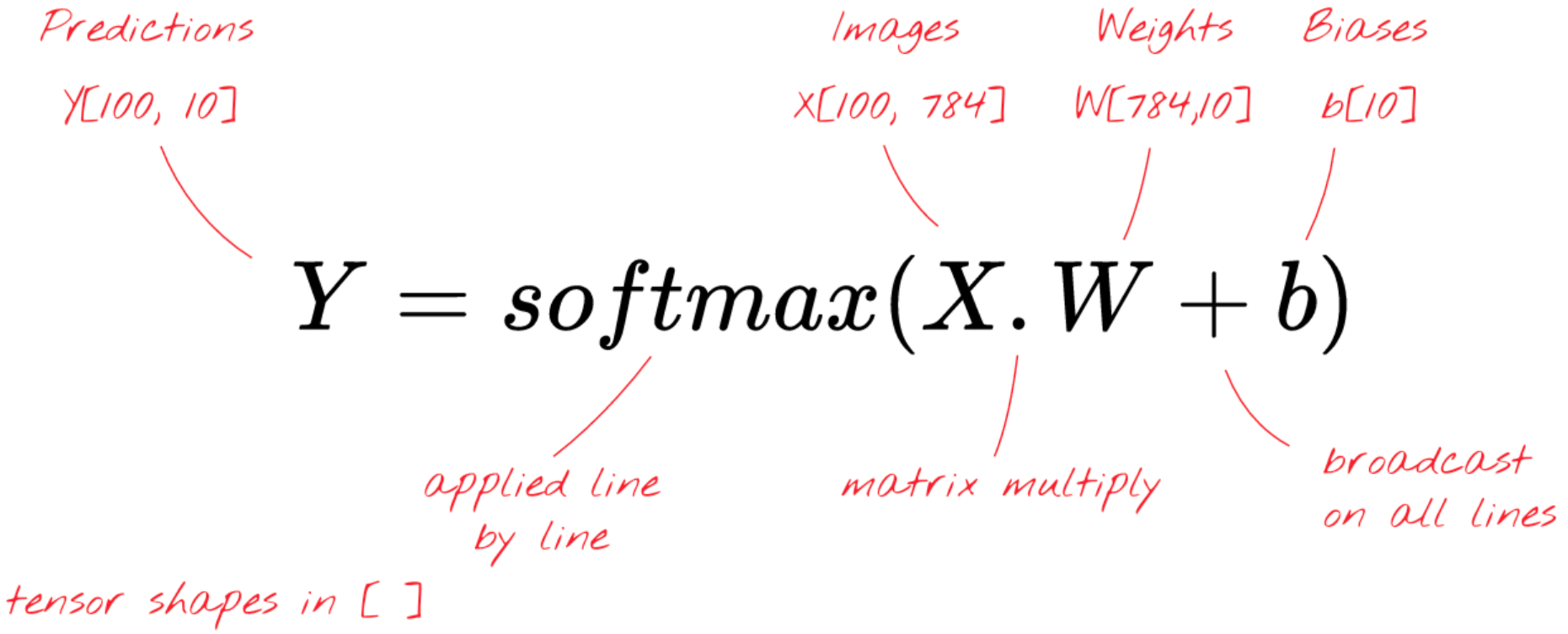
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/


10 columns
784 lines
X: 100 images, one per line, flattened
784 pixels
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 ?

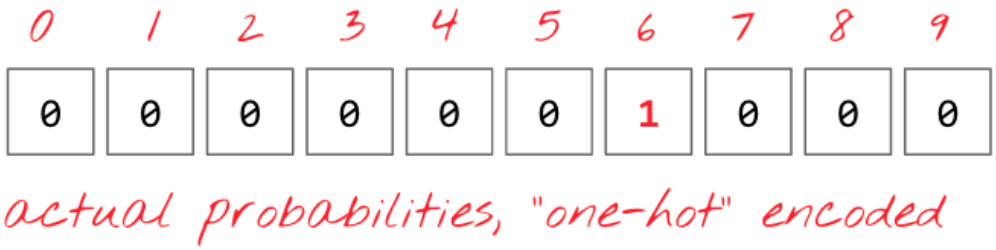
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Y = f(X)

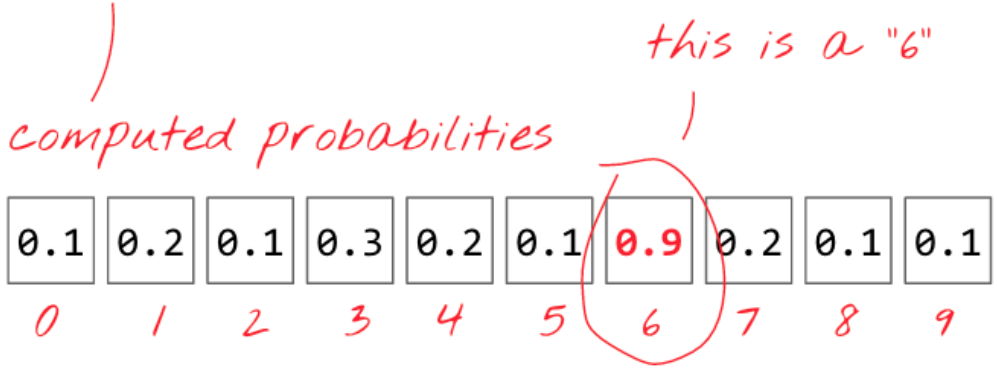


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Cross Entropy

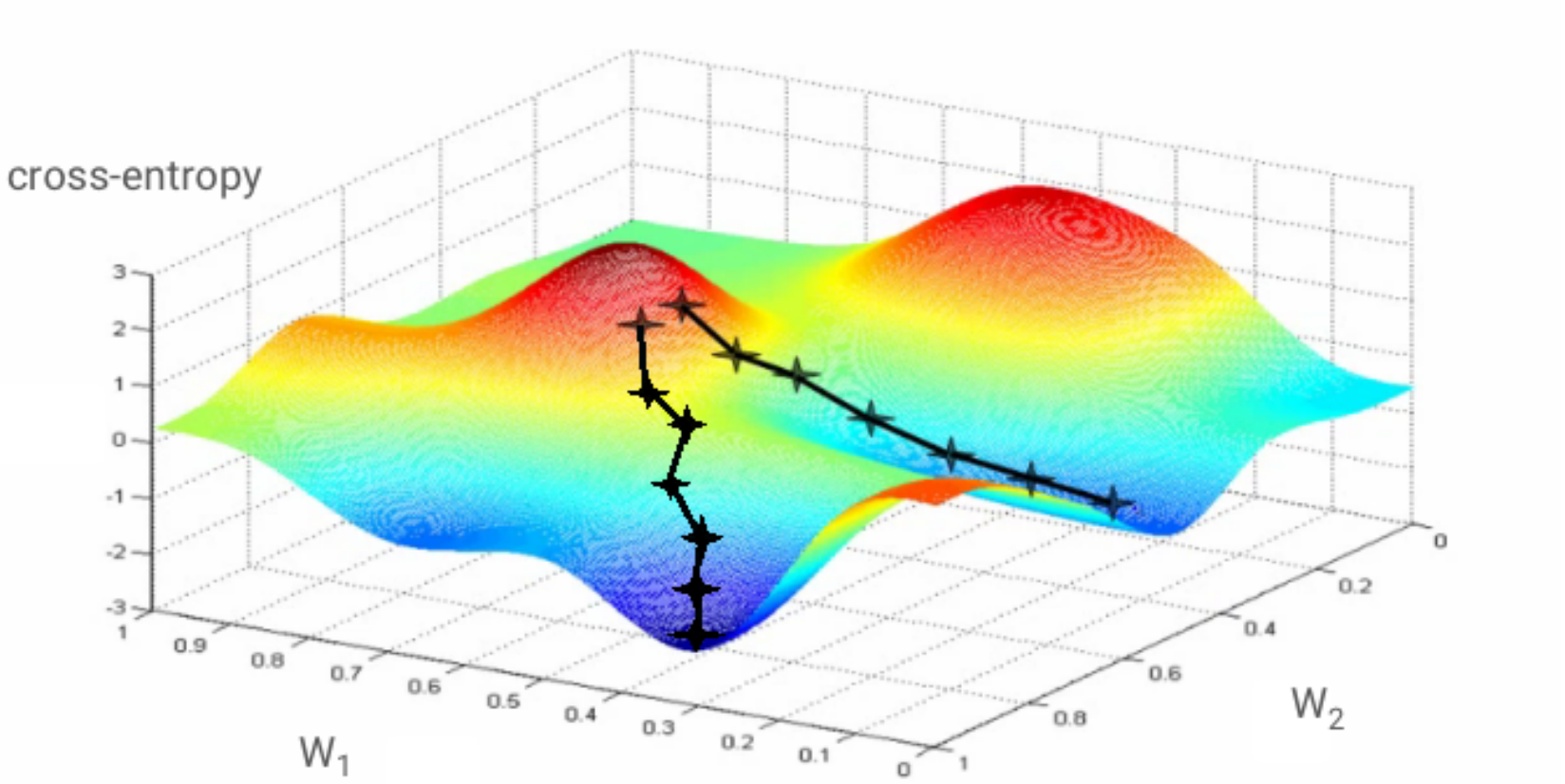


Cross entropy: $-\sum Y_i' . log(Y_i)$



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

Cross Entropy



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

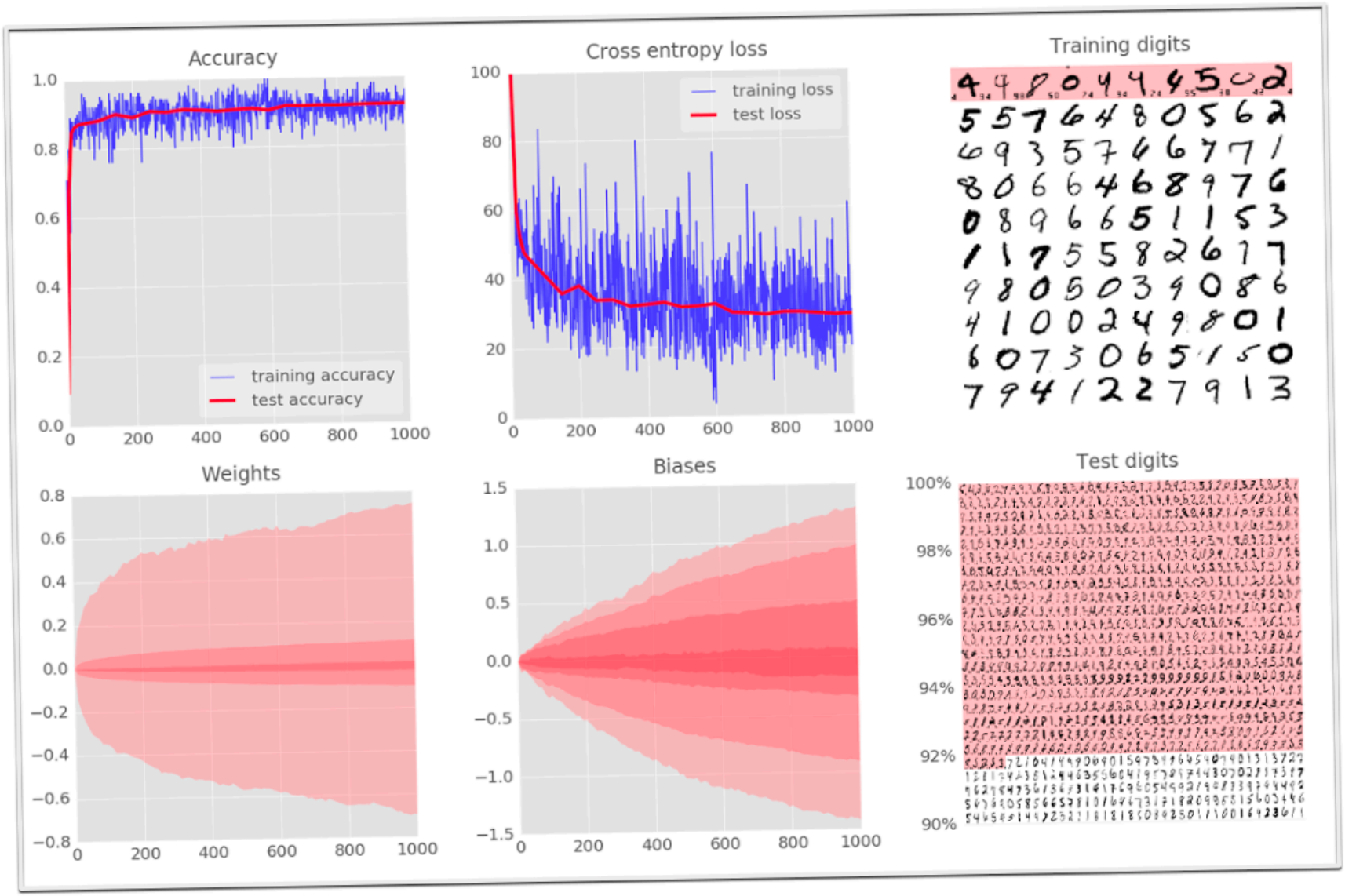
mnist_1.0_softmax.py

```
# 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

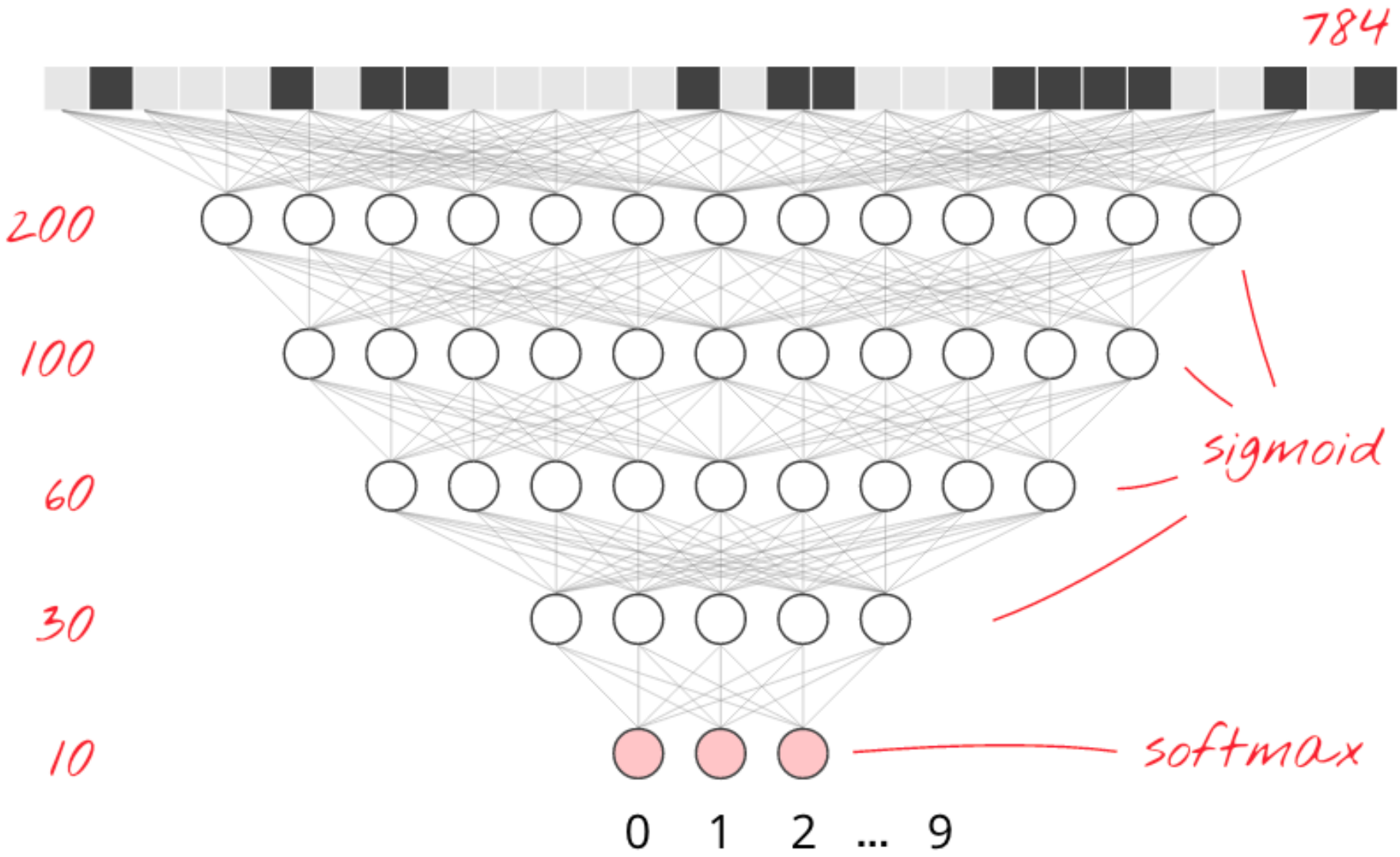


92%

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

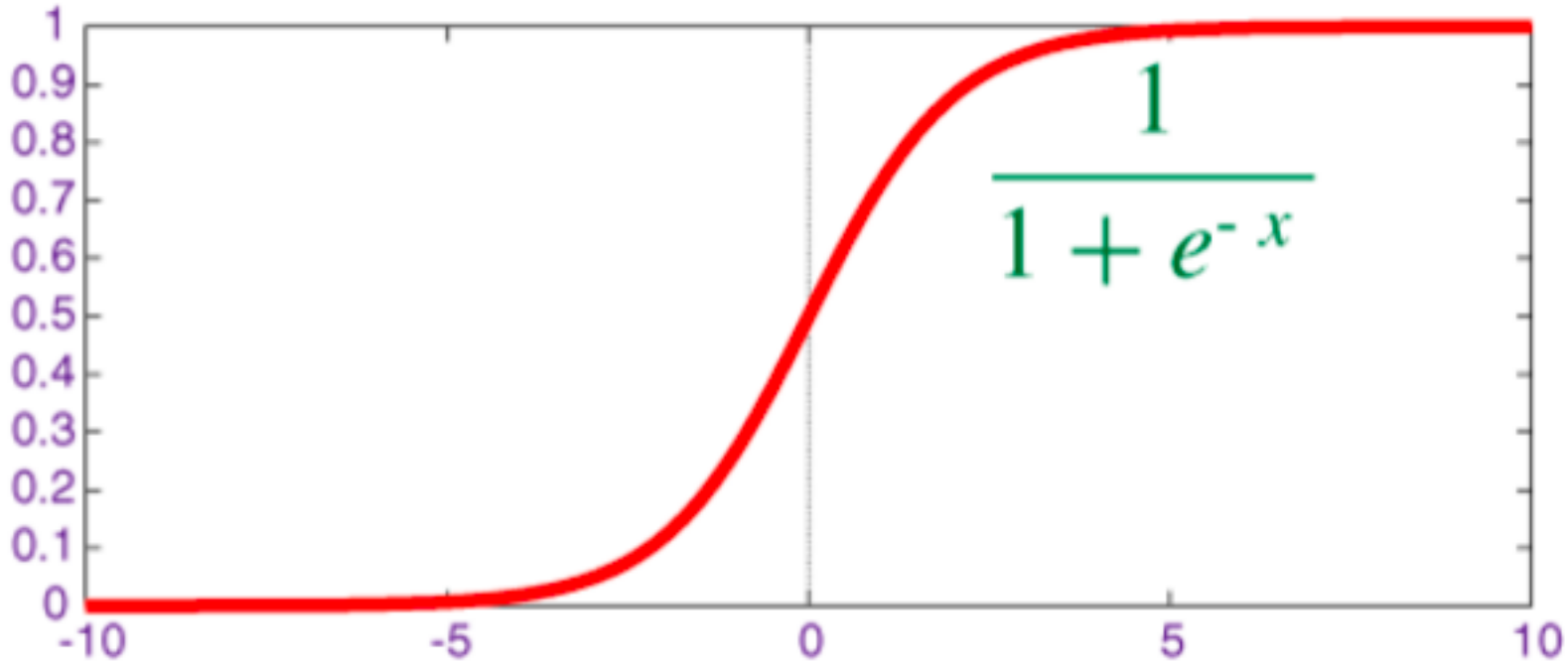
Deep Learning

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

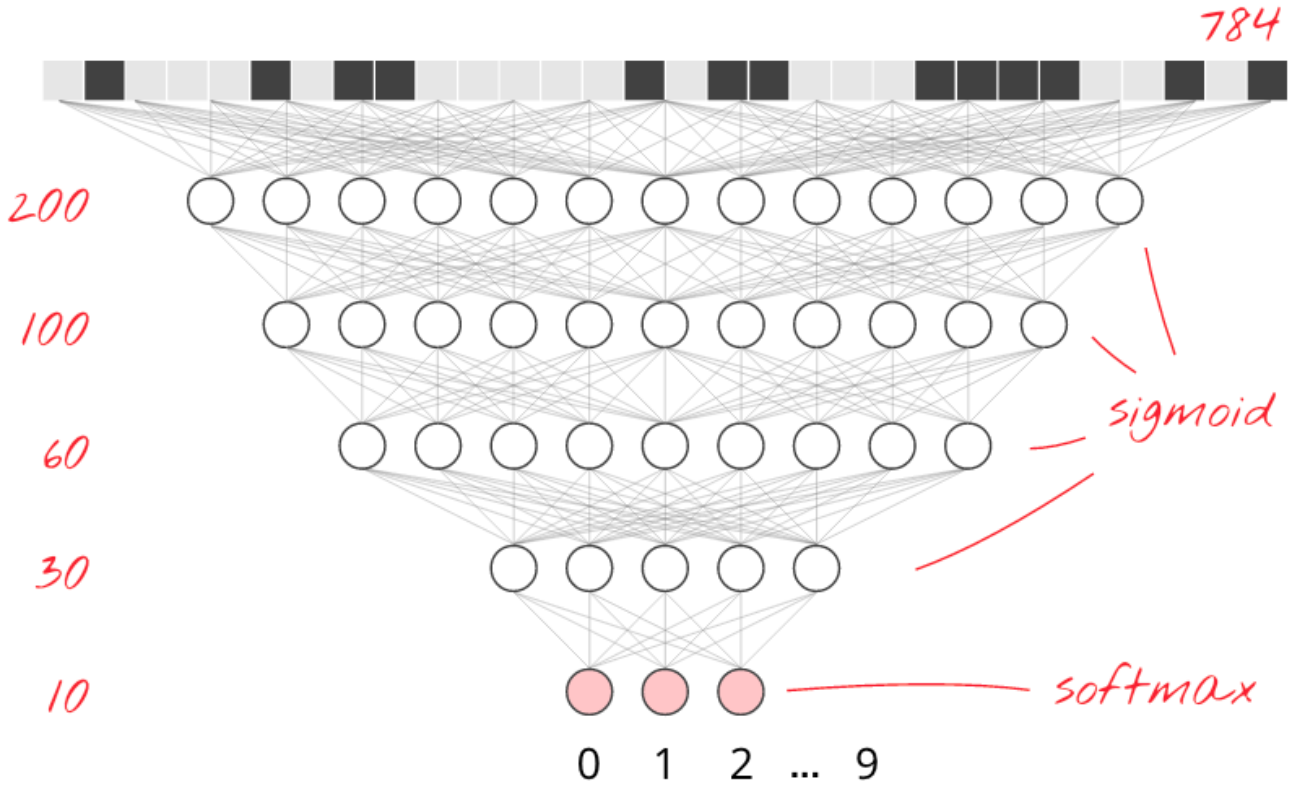


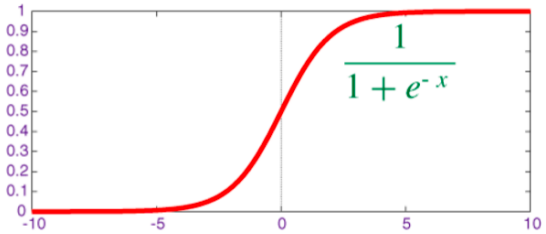
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Sigmoid

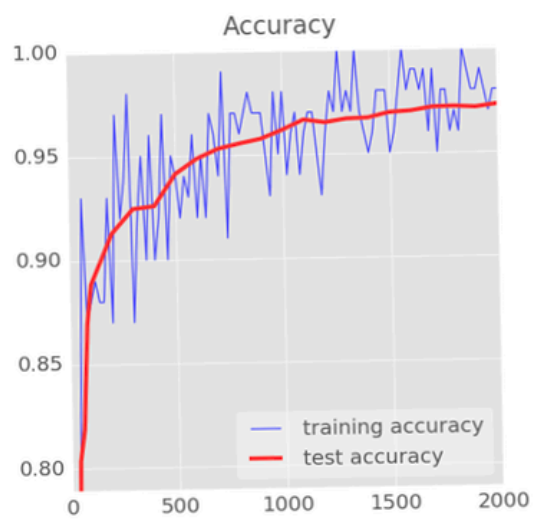


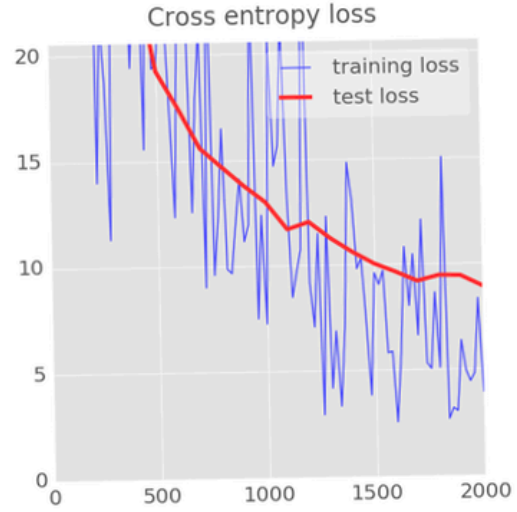
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

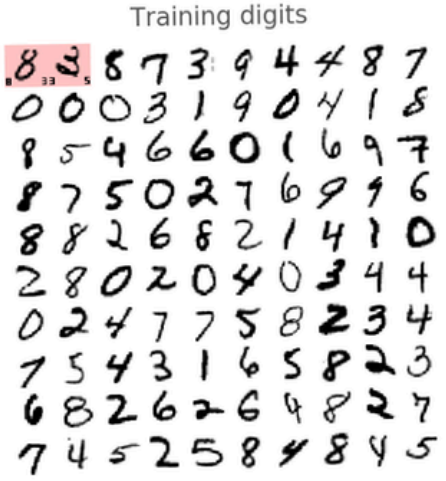


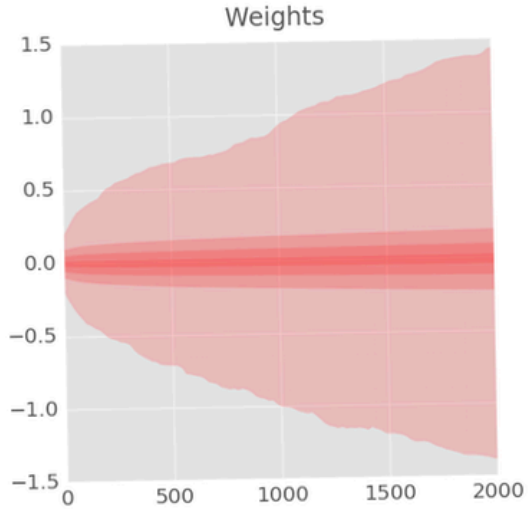


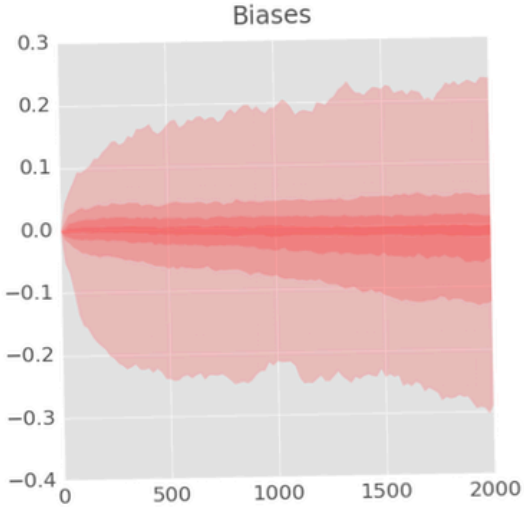
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/


Accuracy
1.00
0.95
0.90
0.85
0.80
0
500
1000
1500
2000
training accuracy
test accuracy


Cross entropy loss
20
15
10
5
0
0
500
1000
1500
2000
training loss
test loss


Training digits


Weights
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
500
1000
1500
2000

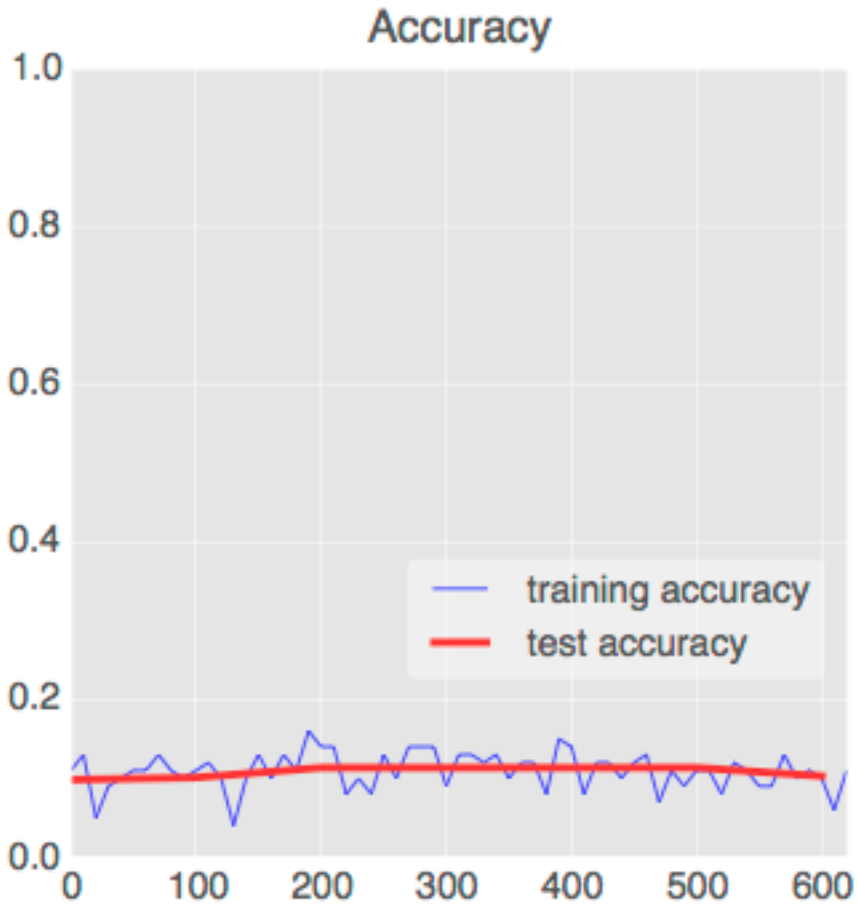

Biases
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
500
1000
1500
2000


Test digits
100%
98%
96%
94%
92%
90%

97%

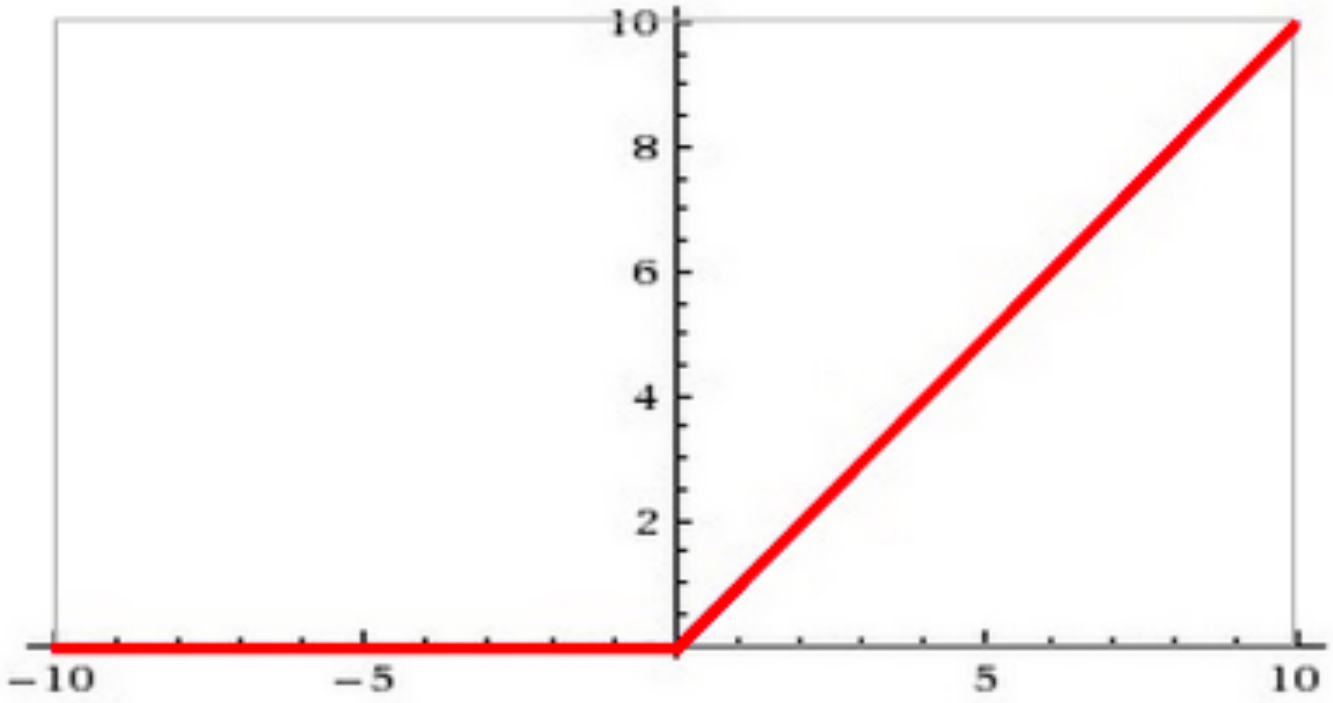
ReLU

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

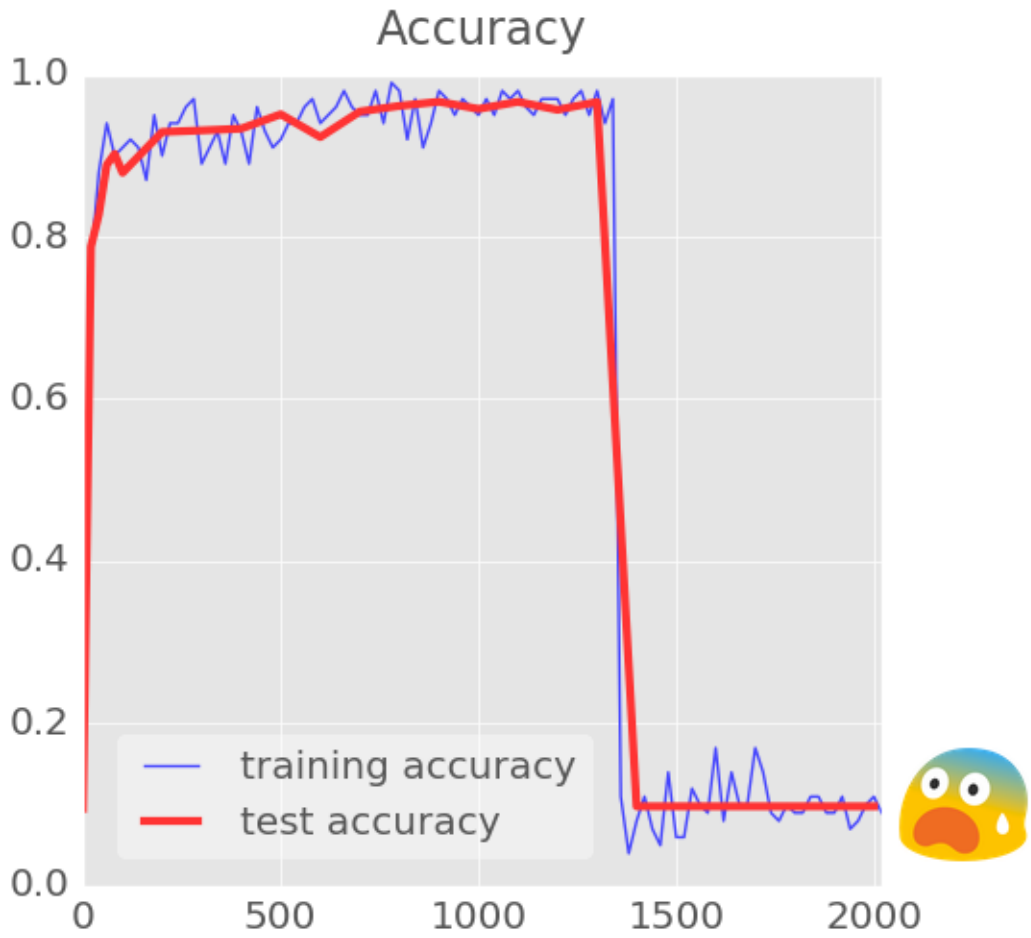


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

ReLU



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

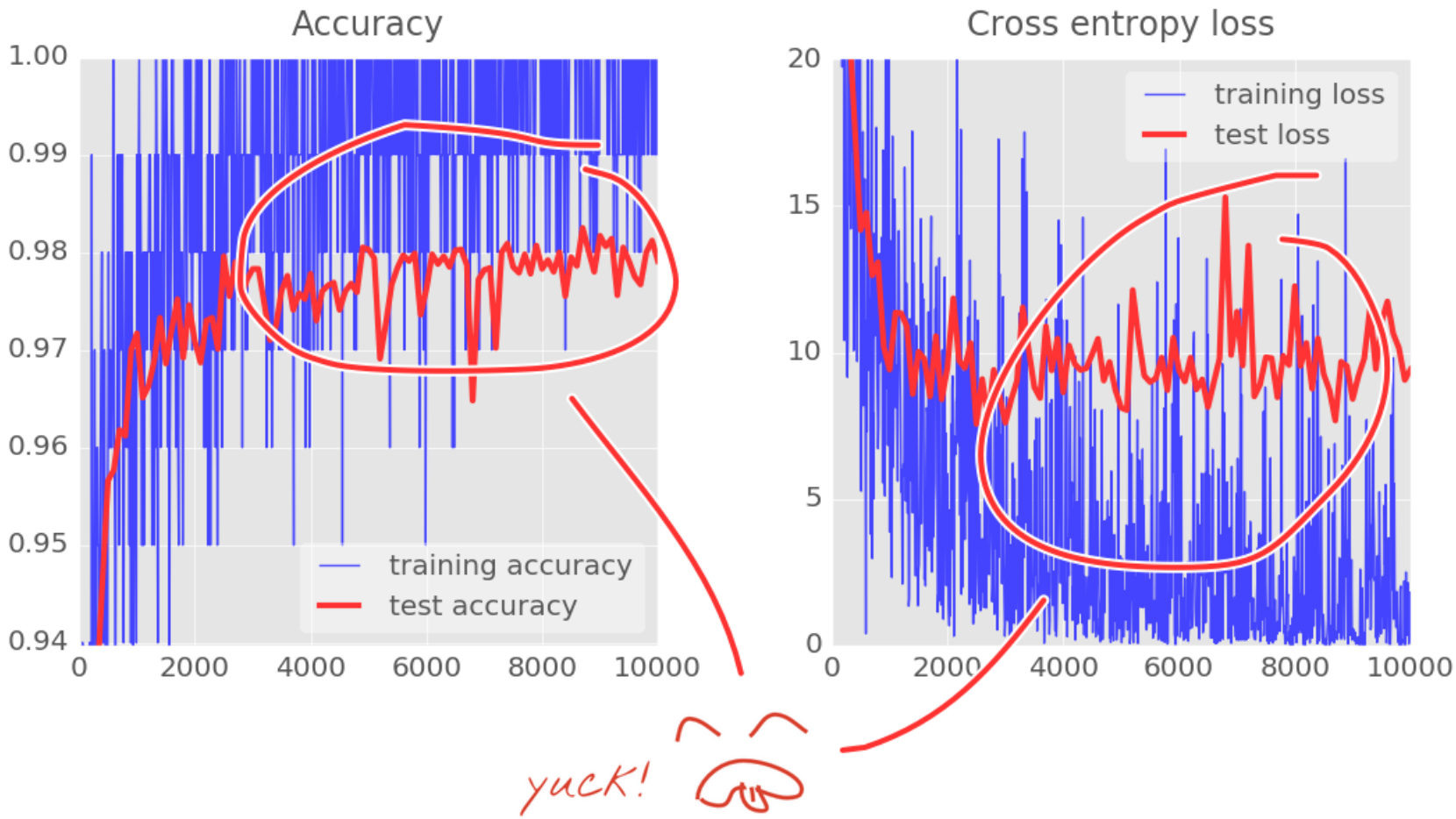


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Learning Rate

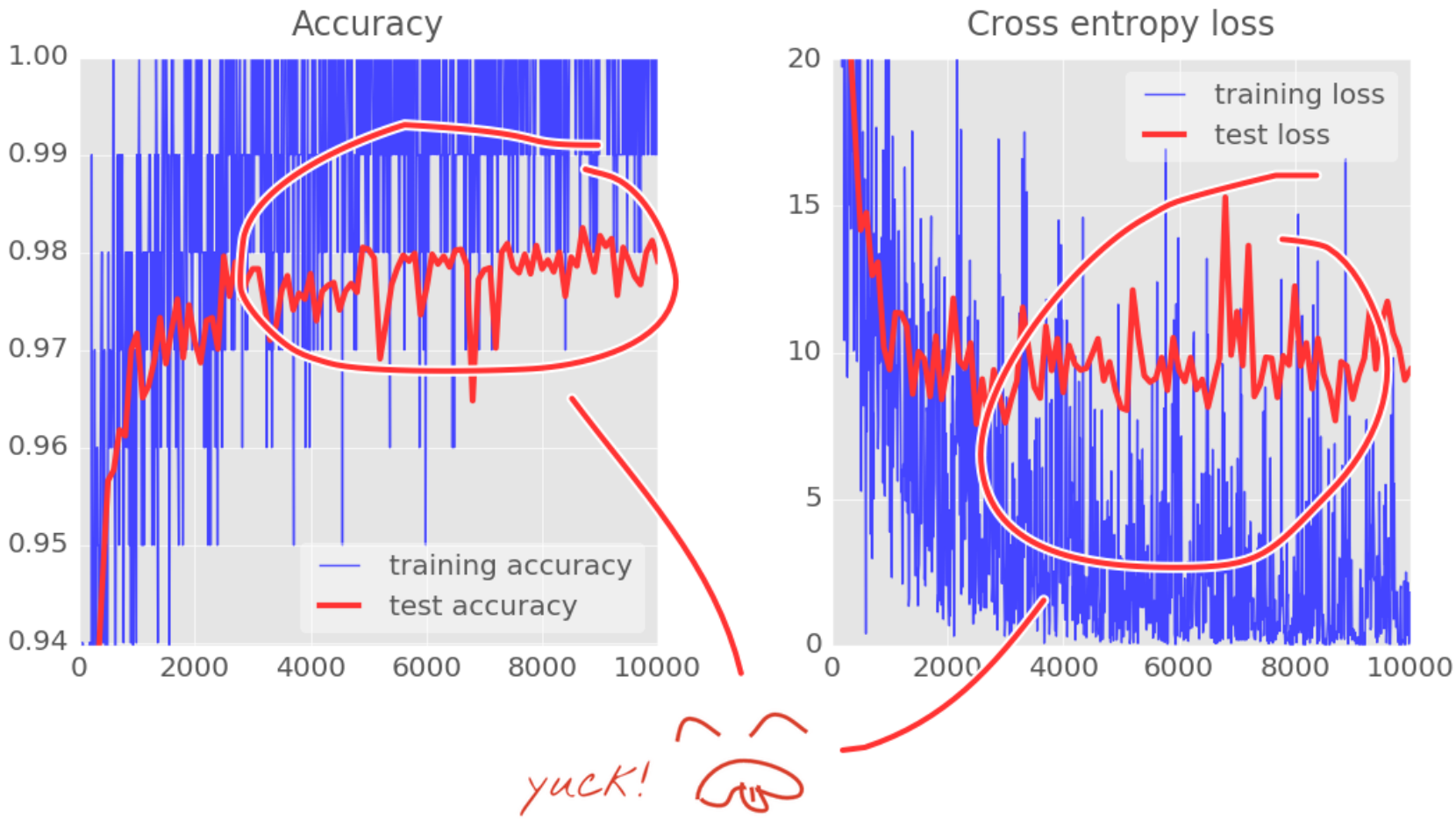
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

LR = 0.003

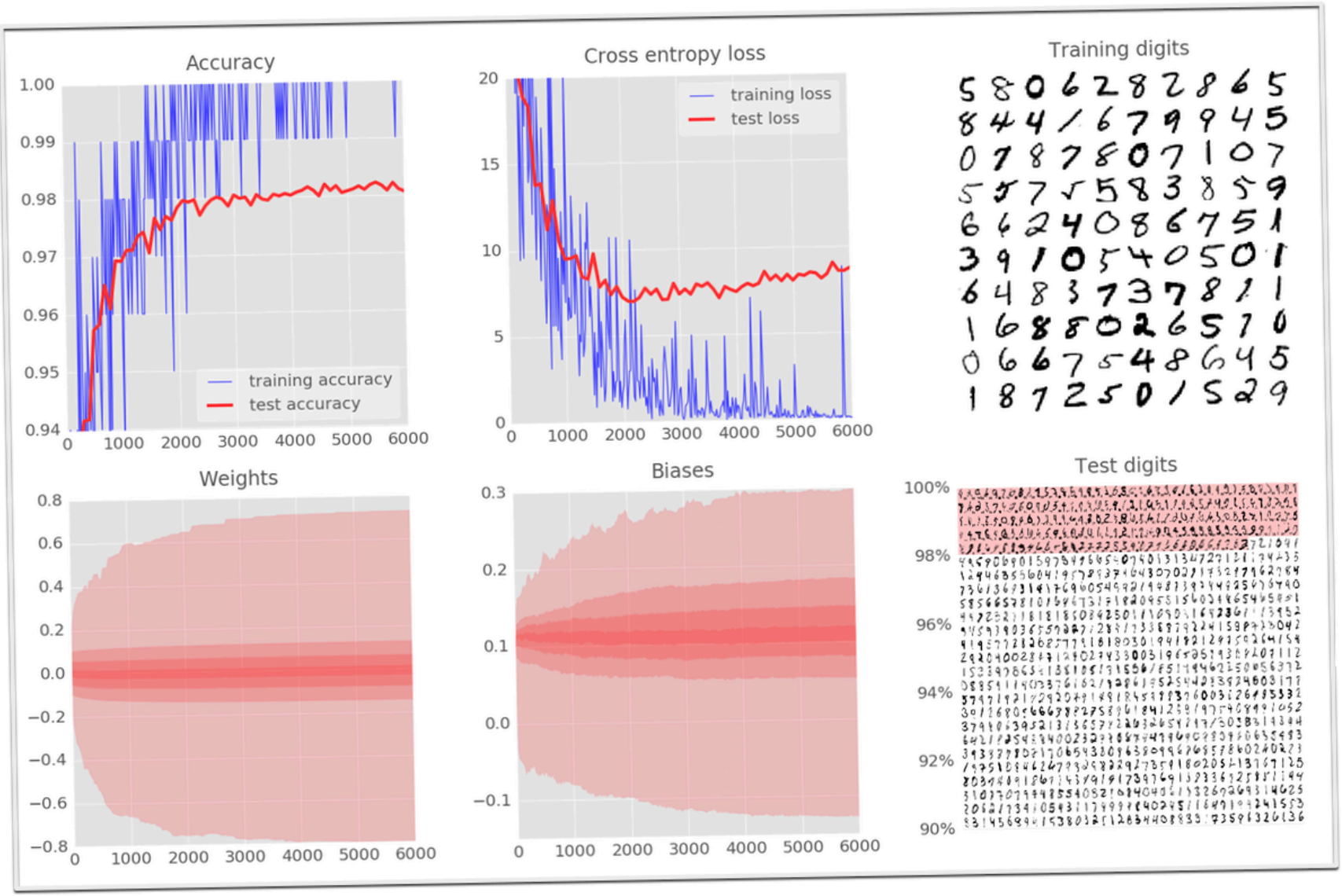


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

LR = 0.003

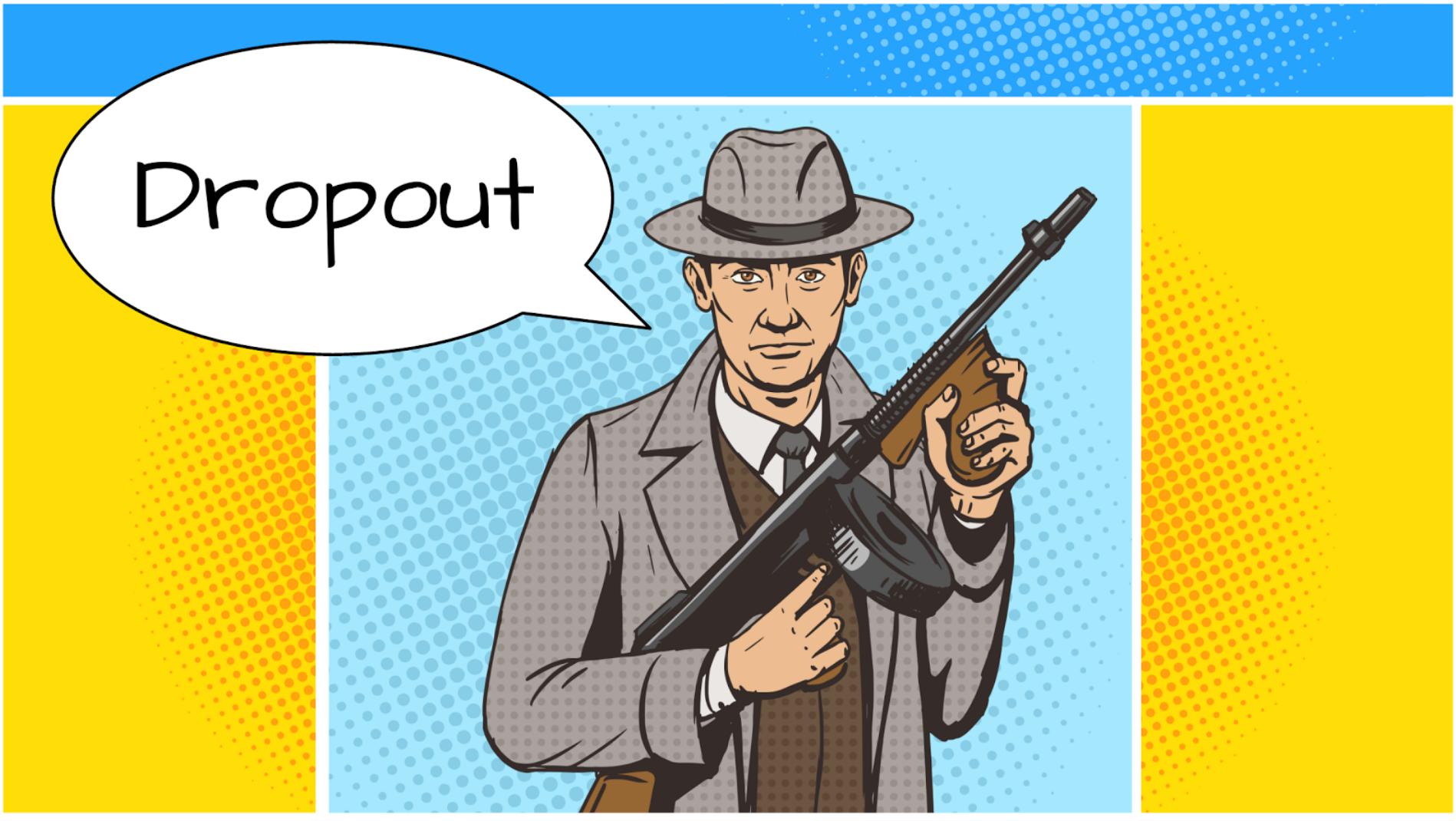


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

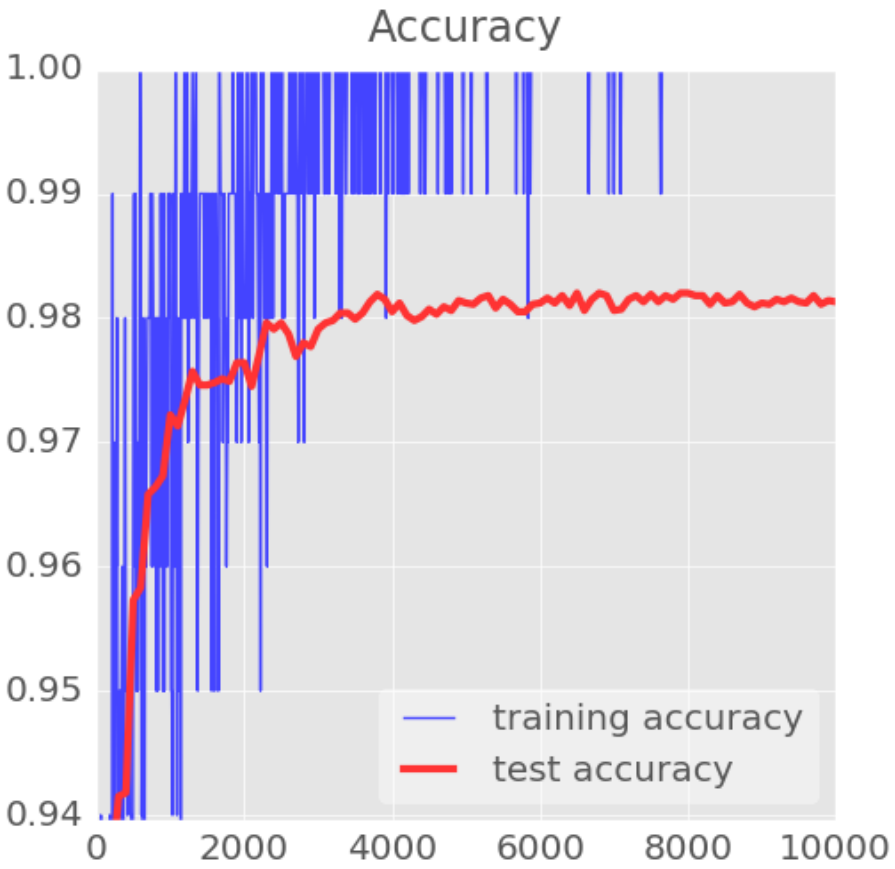

Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

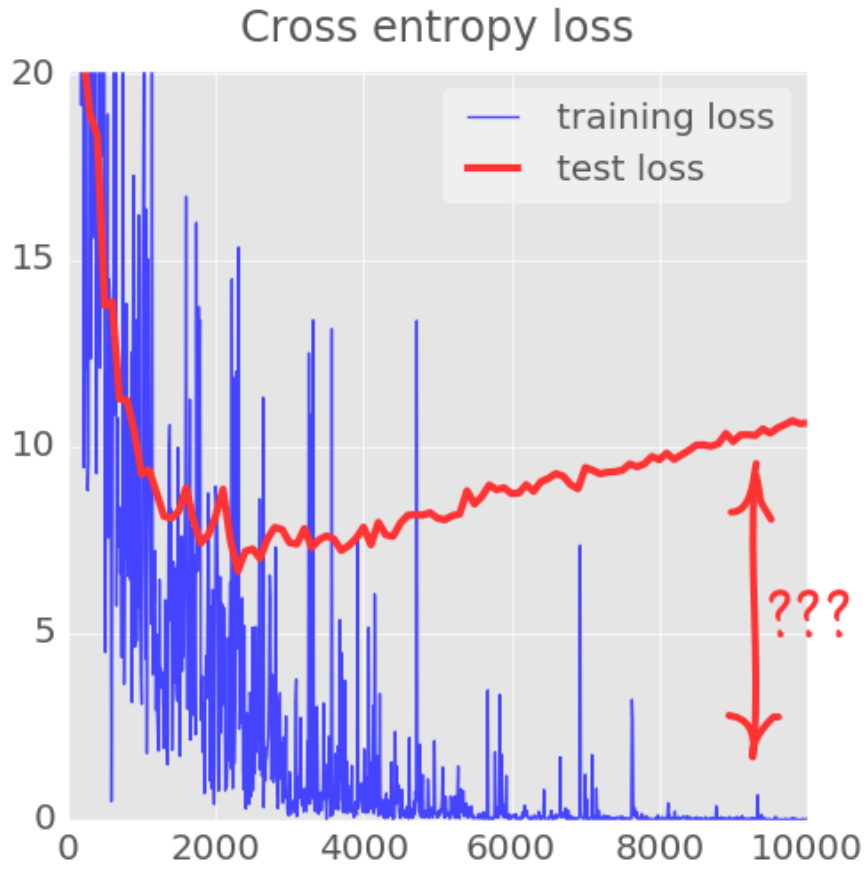
98%

Dropout



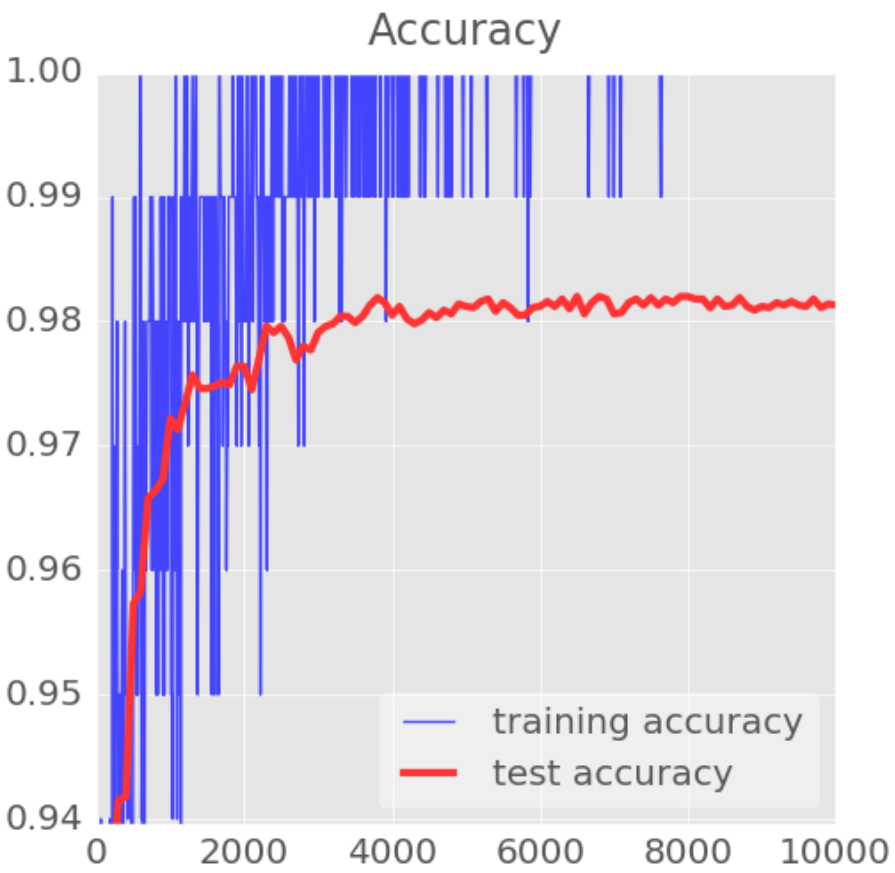
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

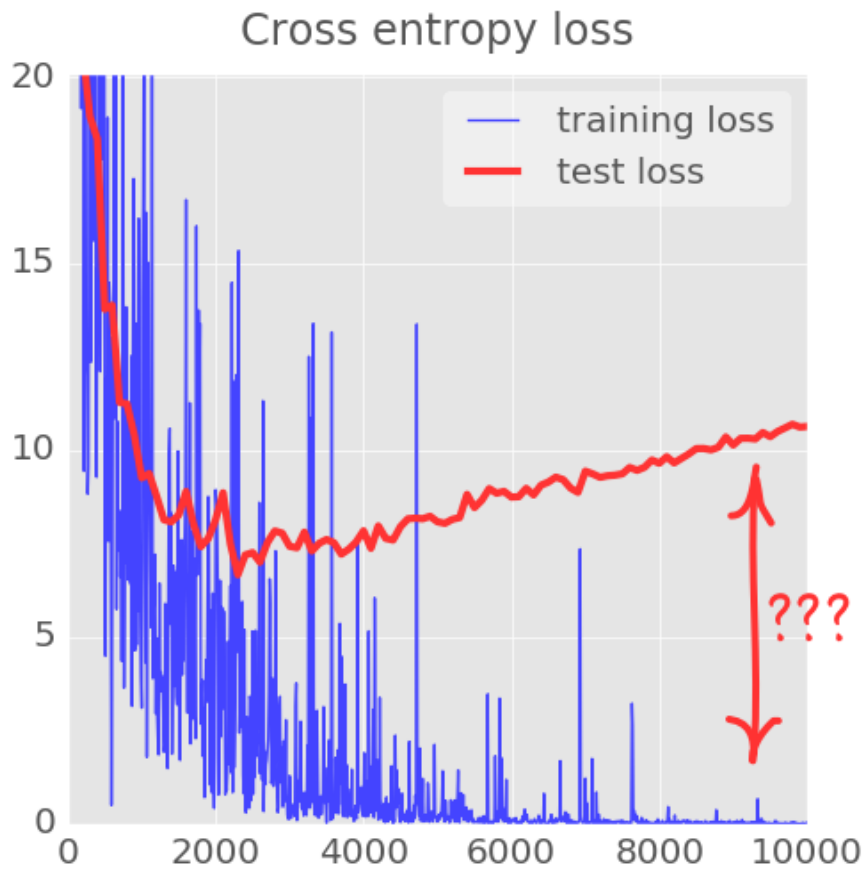




Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

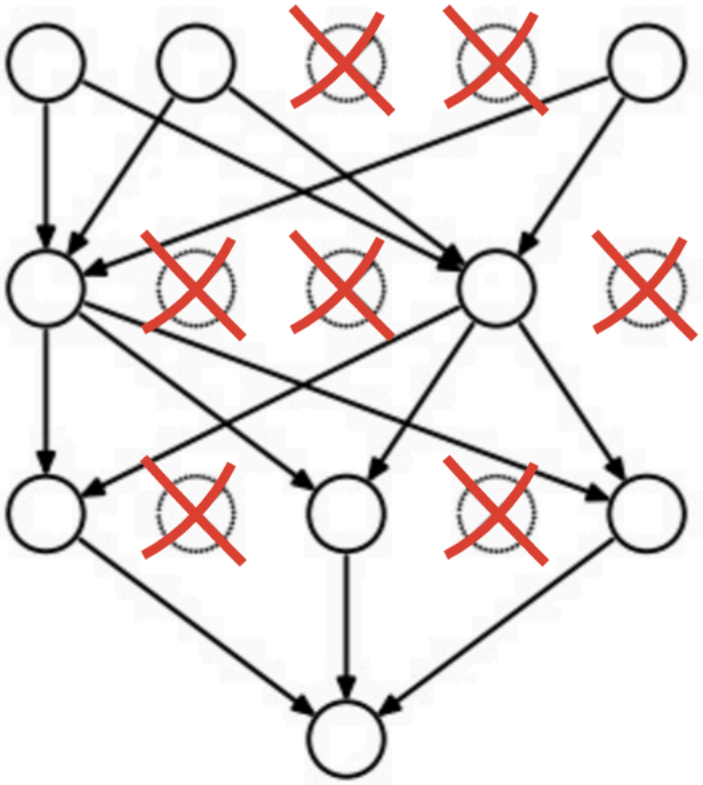
Overfitting



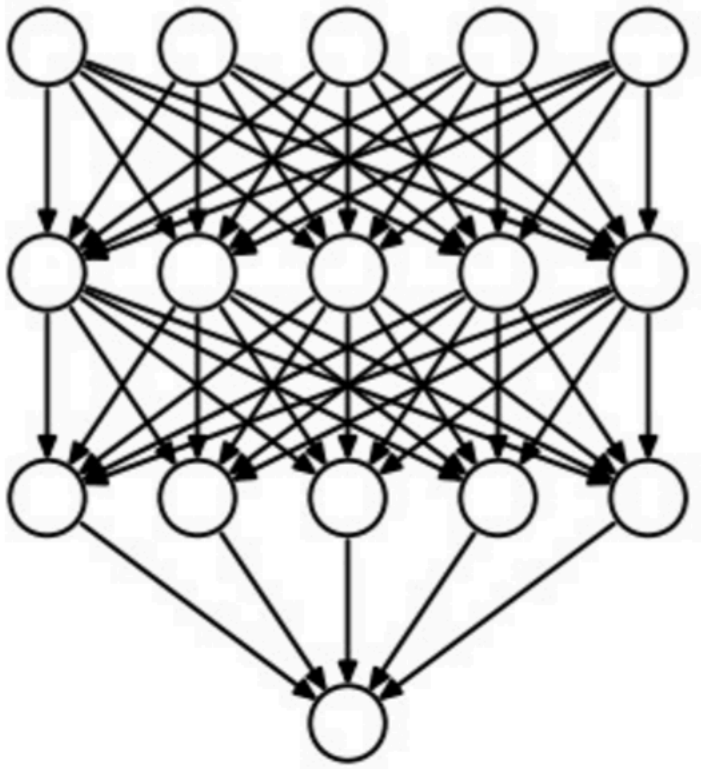


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Dropout



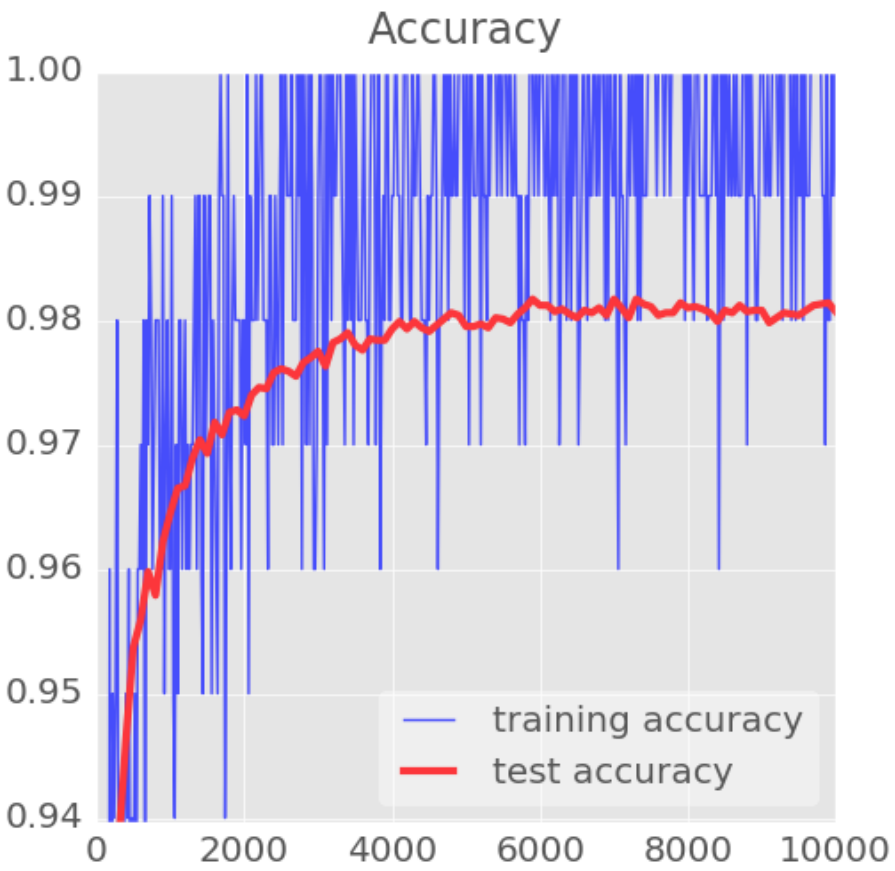
Training
pkeep = 0.75

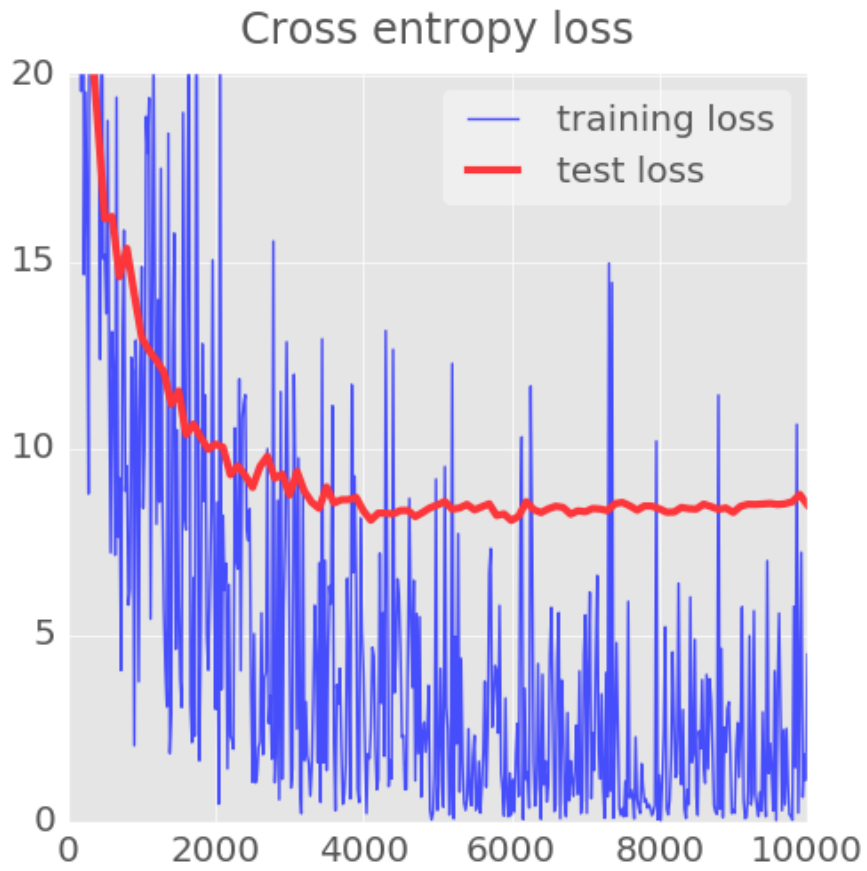


Test
pkeep = 1.0

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

TensorFlow MNIST Tutorial

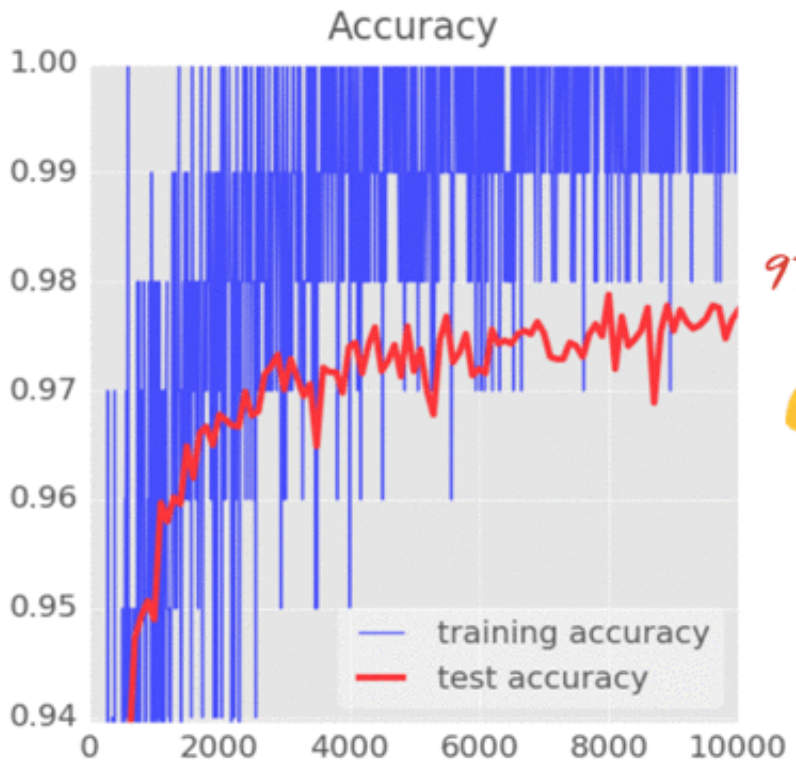




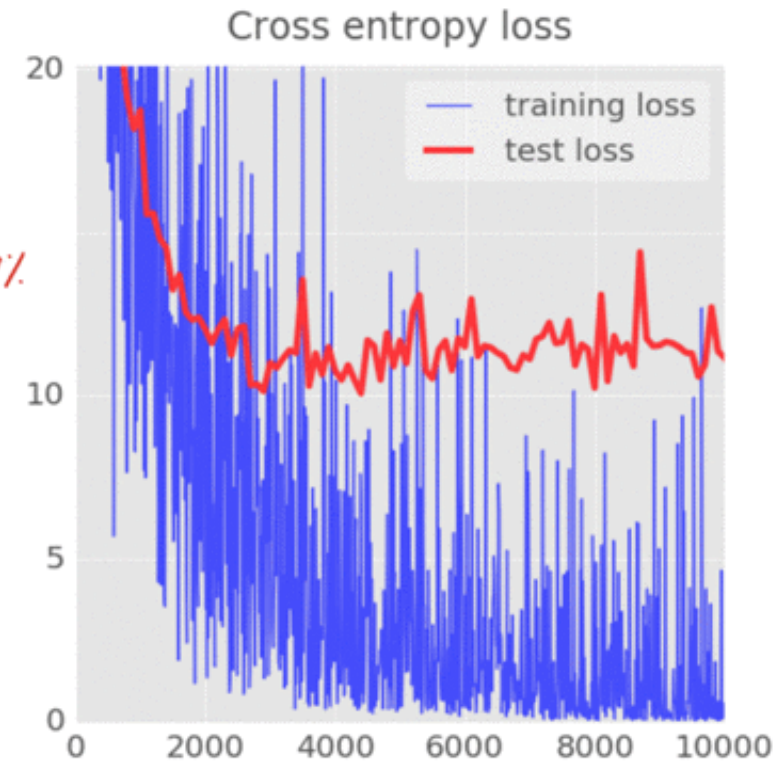
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

5 layers
sigmoid

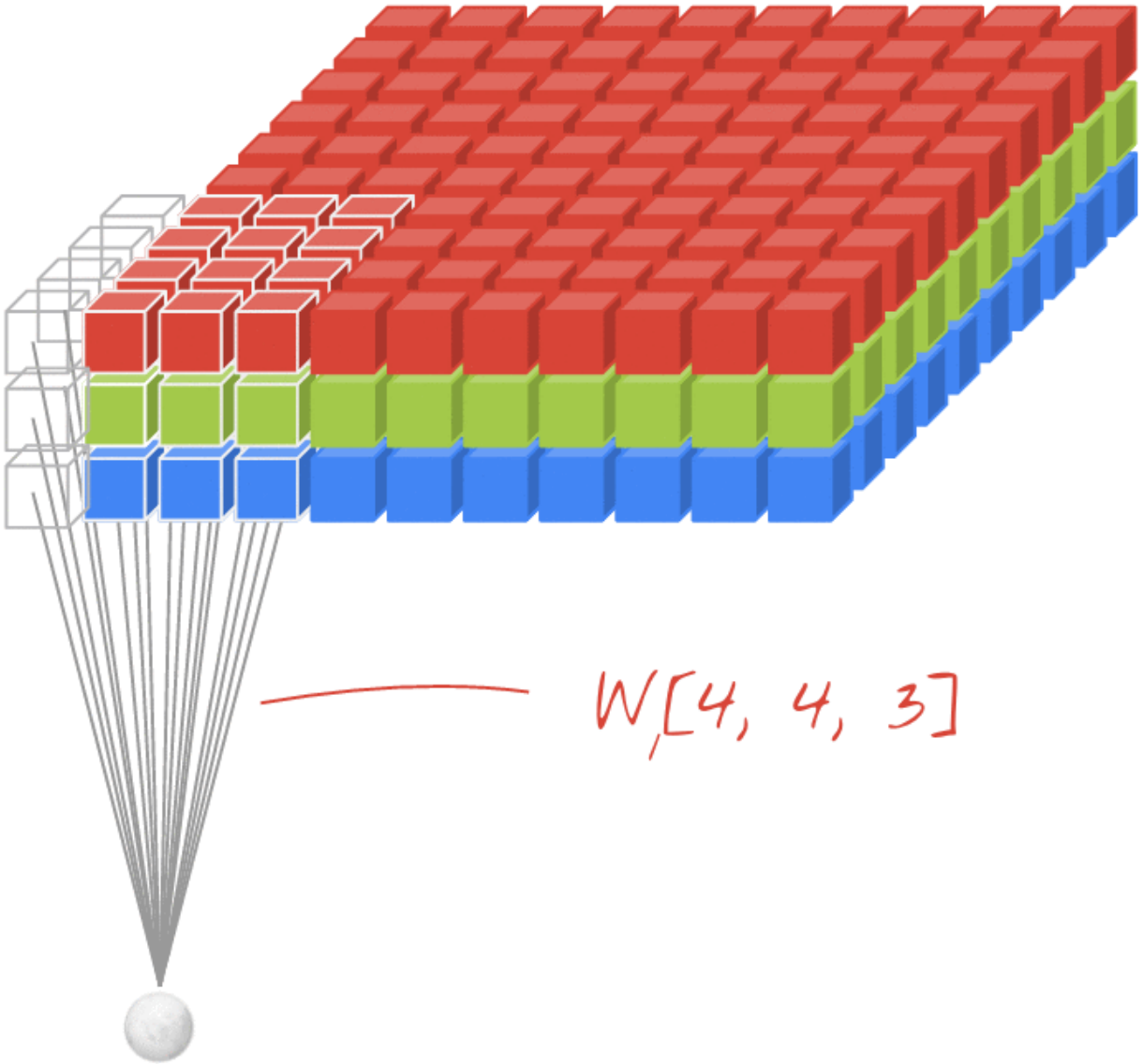


97.9%

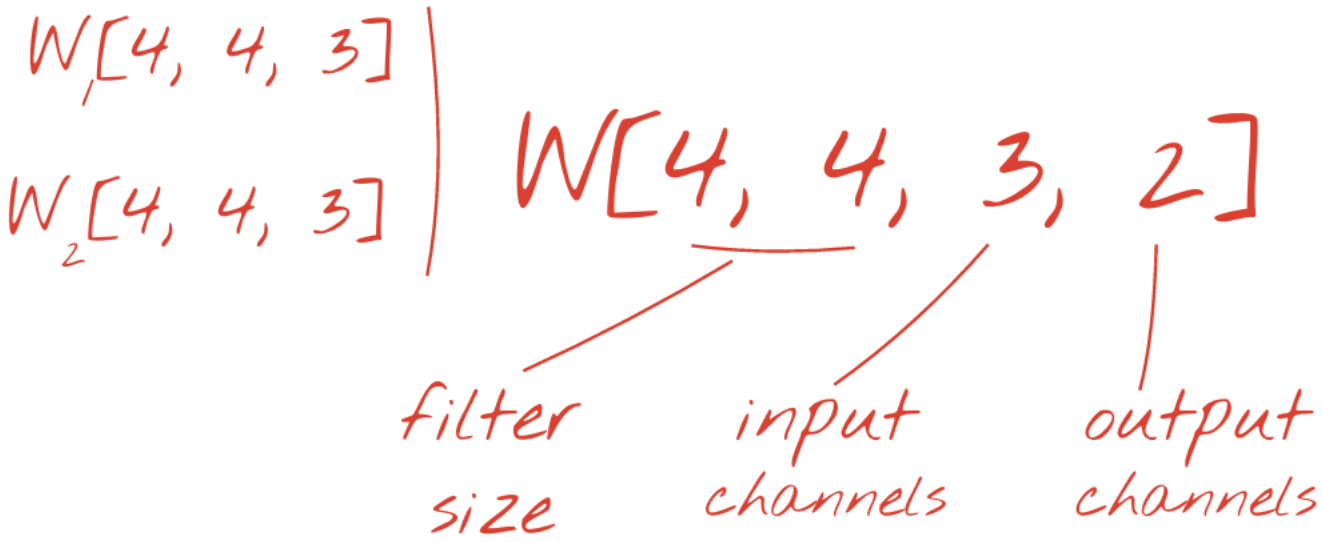


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

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

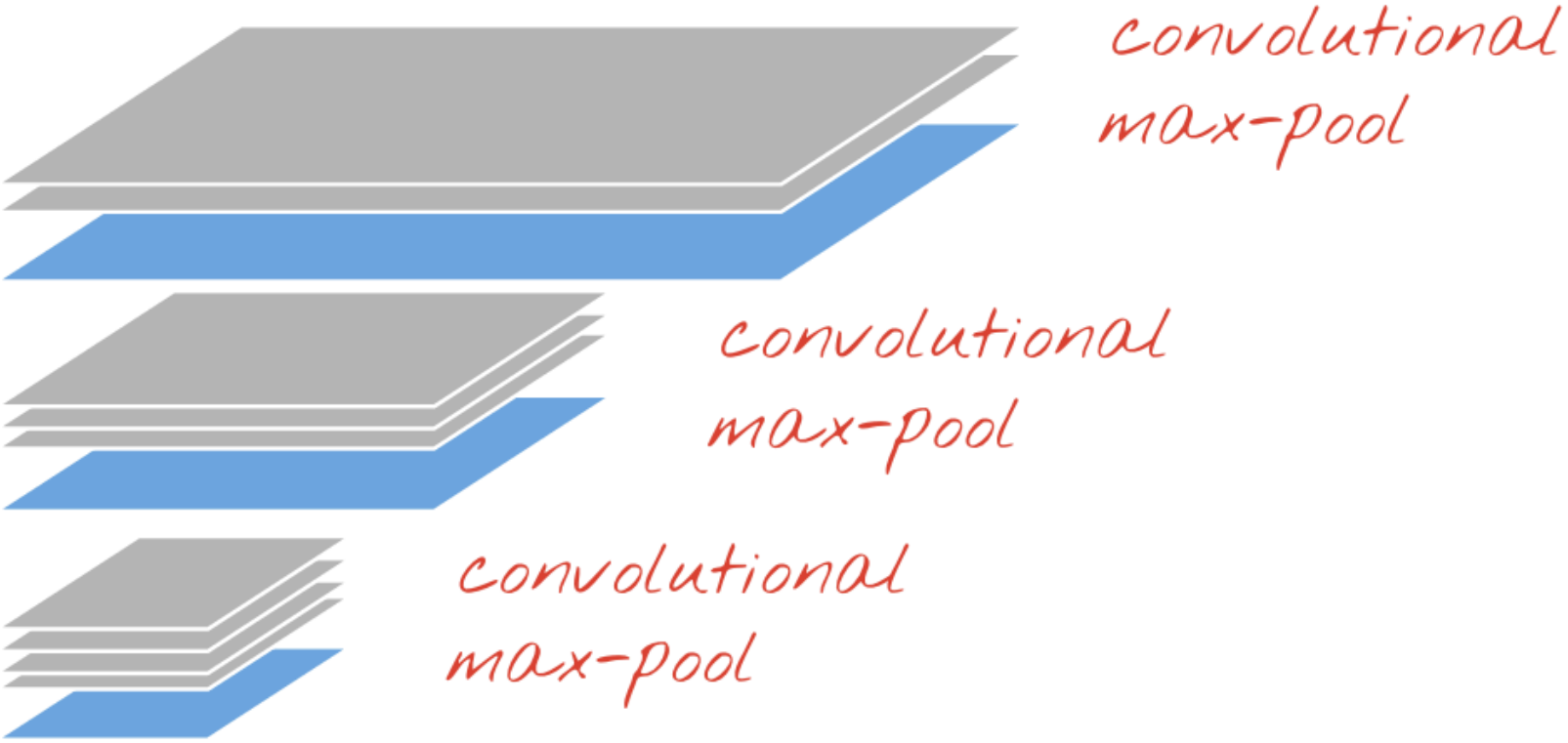


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



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

Convolutional Max-Pool

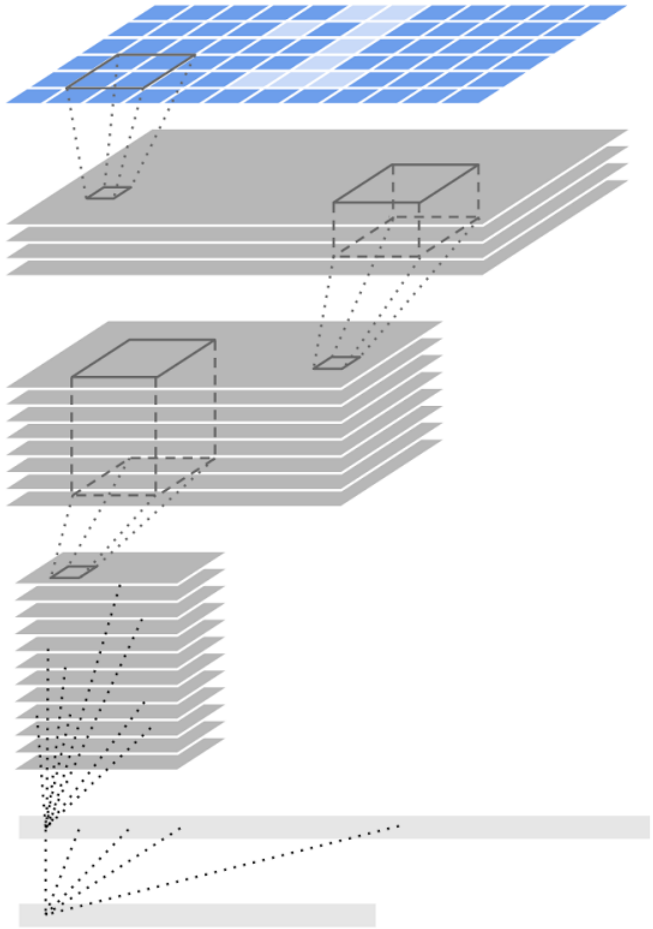


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

All Convolutional

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

+ biases on all layers



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

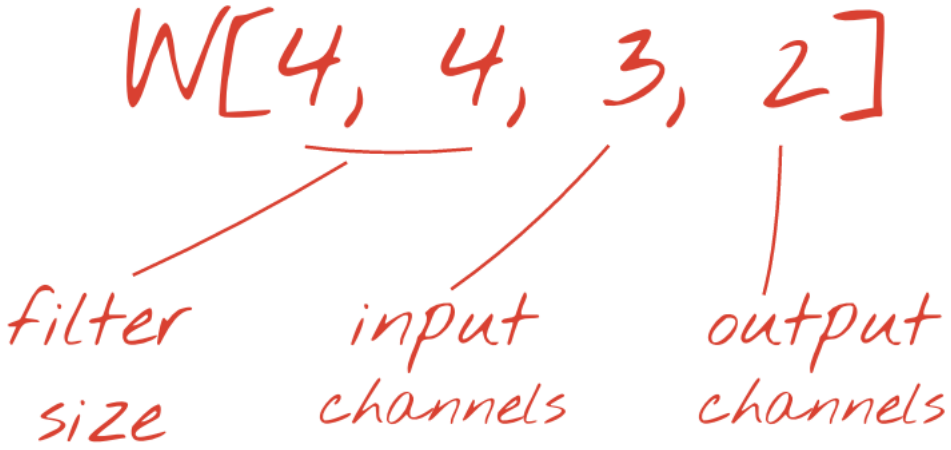
convolutional layer, 8 channels
W2[4, 4, 4, 8] stride 2

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

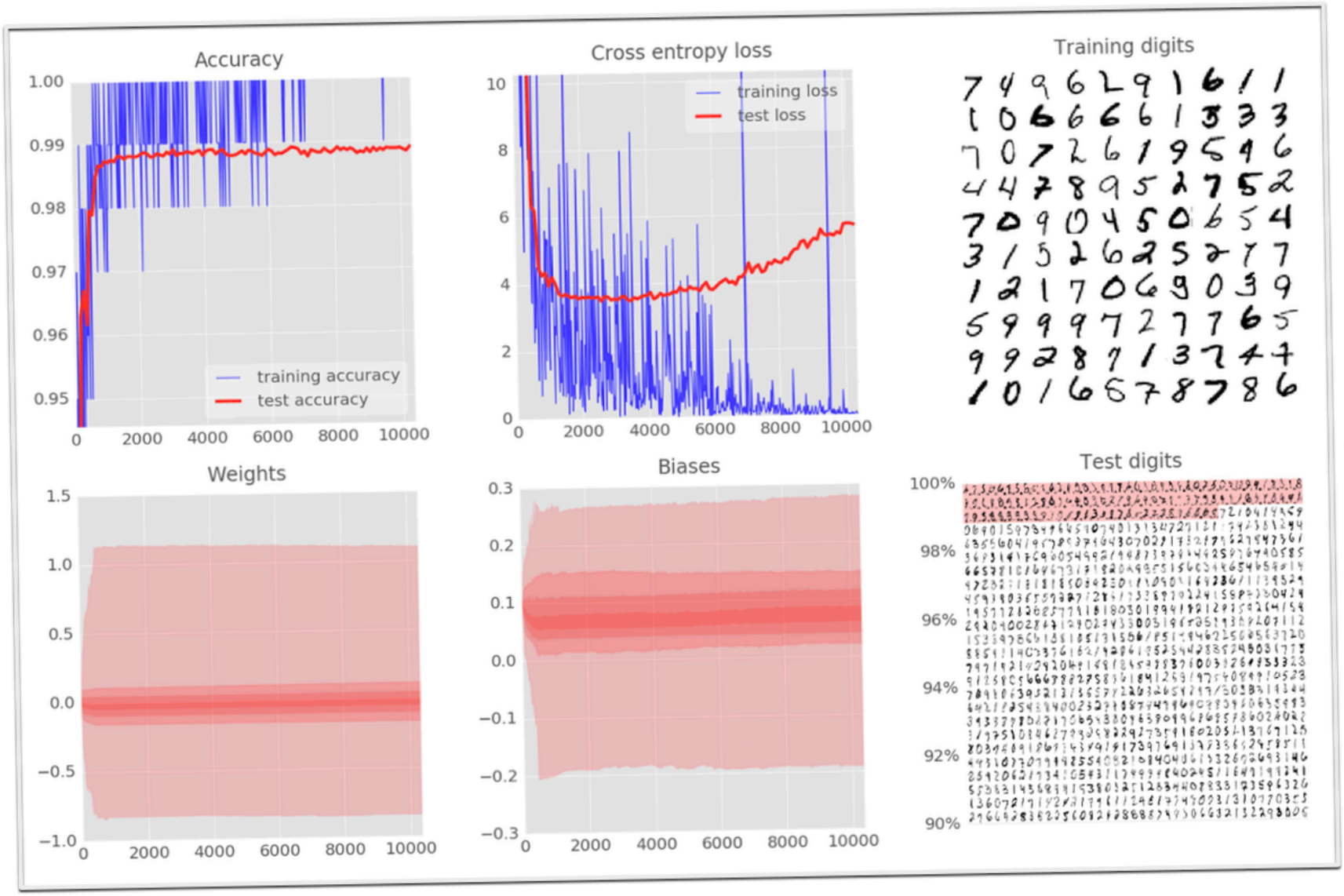
fully connected layer
W4[7x7x12, 200]

softmax readout layer
W5[200, 10]

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



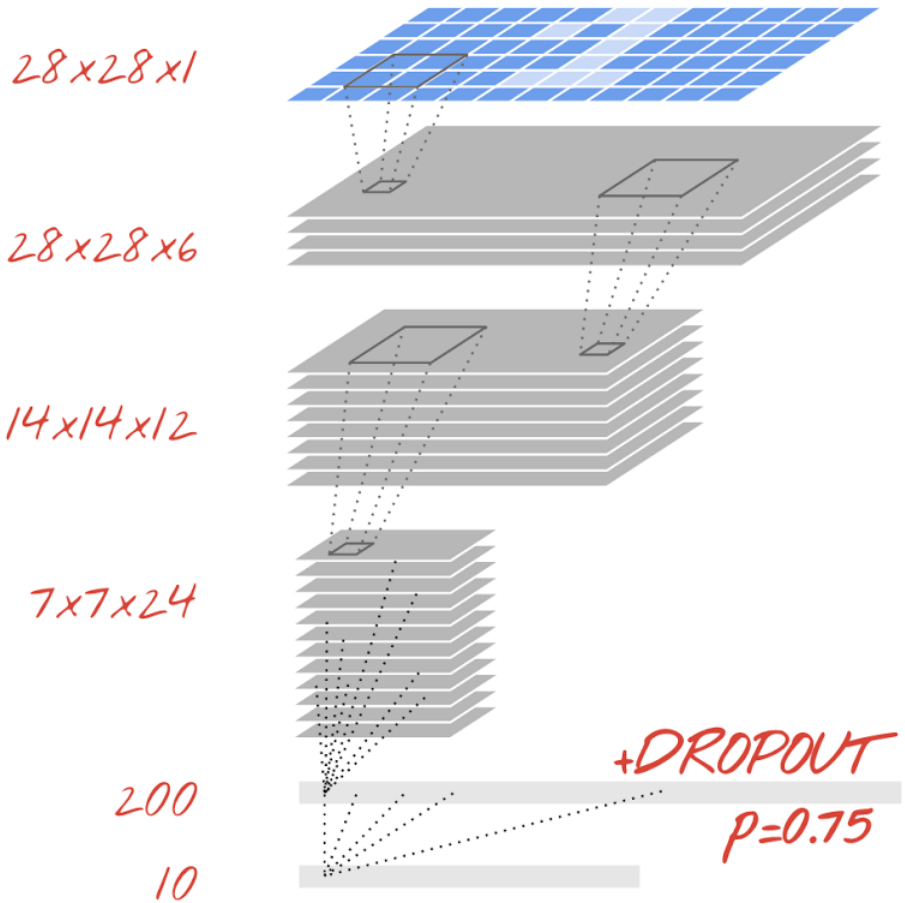
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



98.9%

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

+ 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

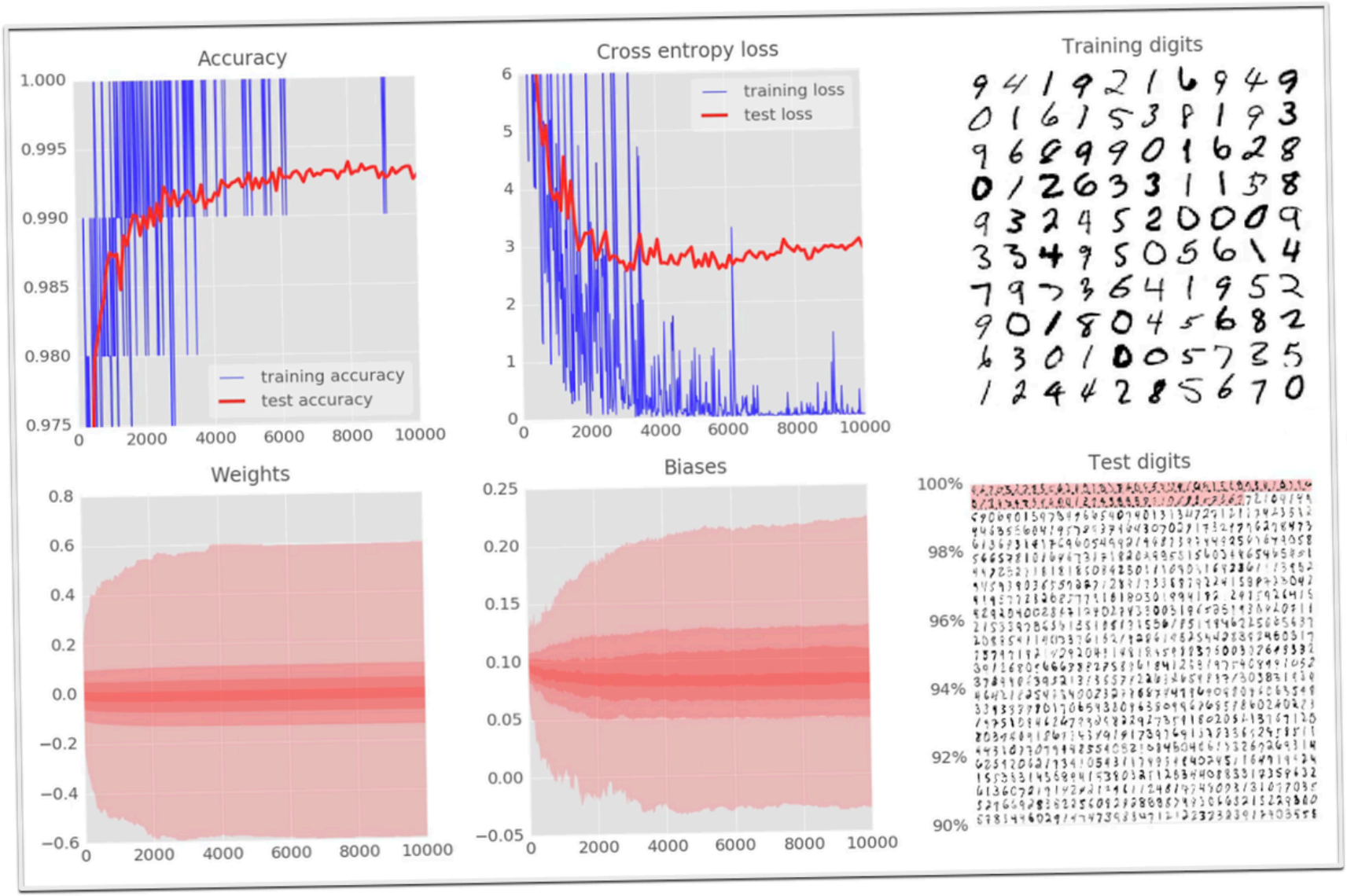
convolutional layer, **24 channels**
W3[**4**, **4**, 12, 24] stride 2

fully connected layer
W4[7x7x24, 200]

softmax readout layer
W5[200, 10]

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

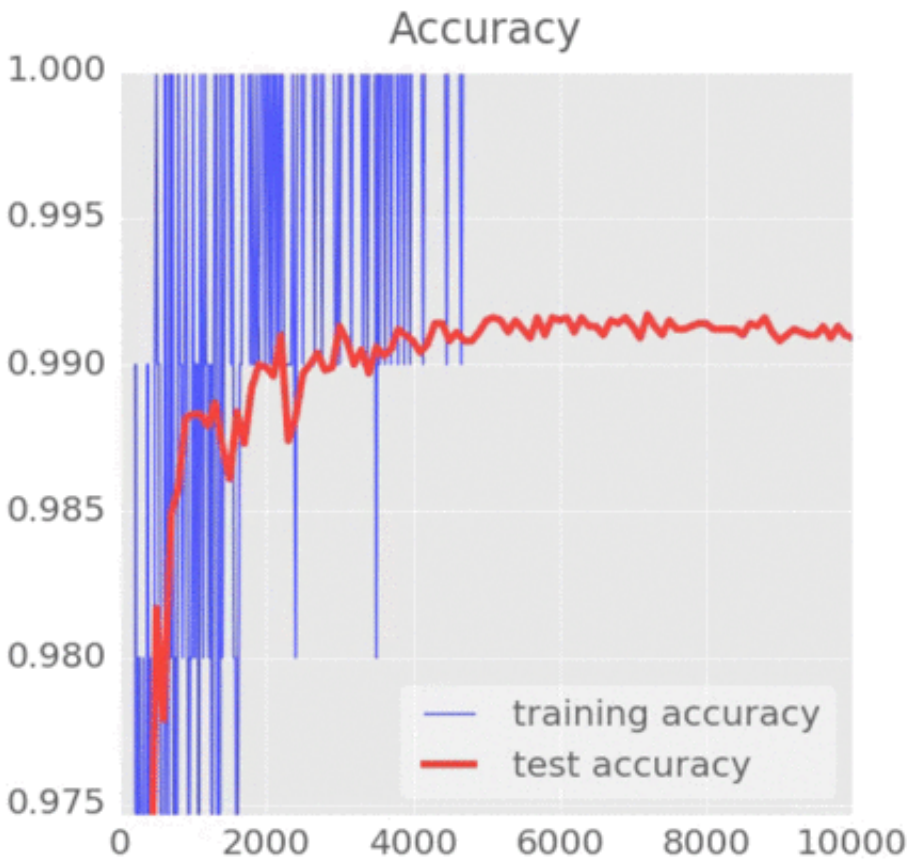
TensorFlow MNIST Tutorial

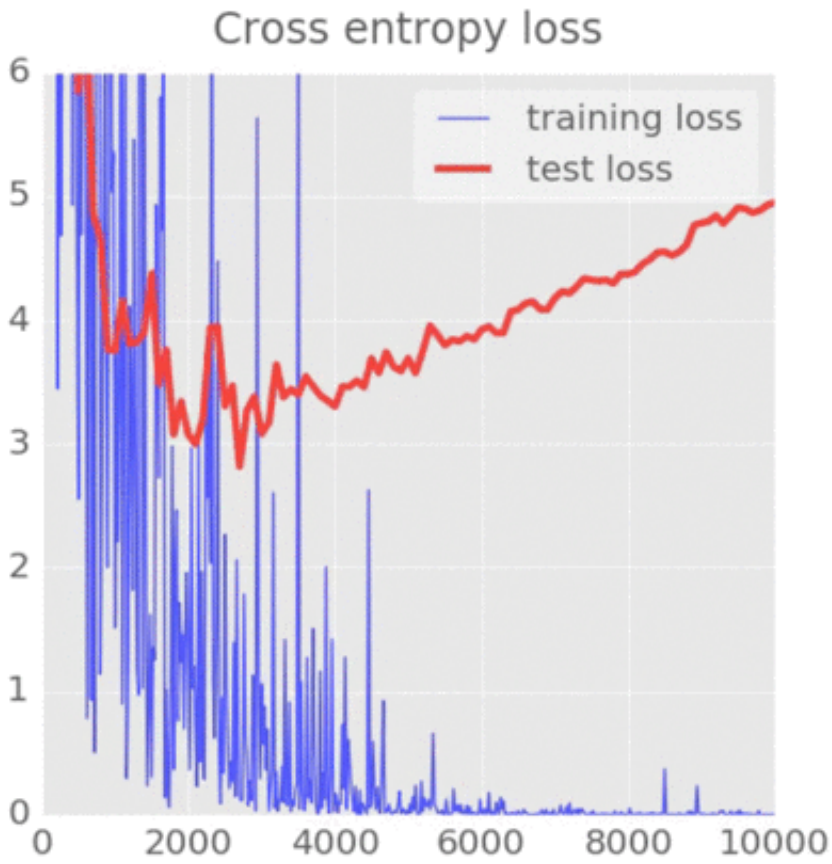


99.3%

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

TensorFlow MNIST Tutorial

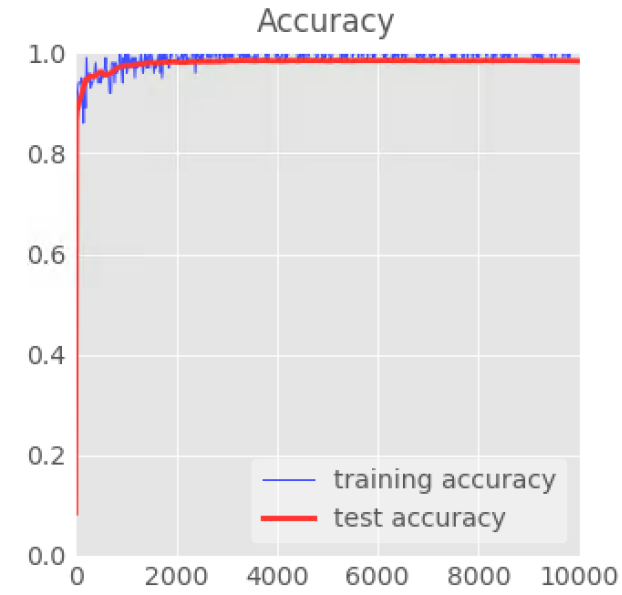


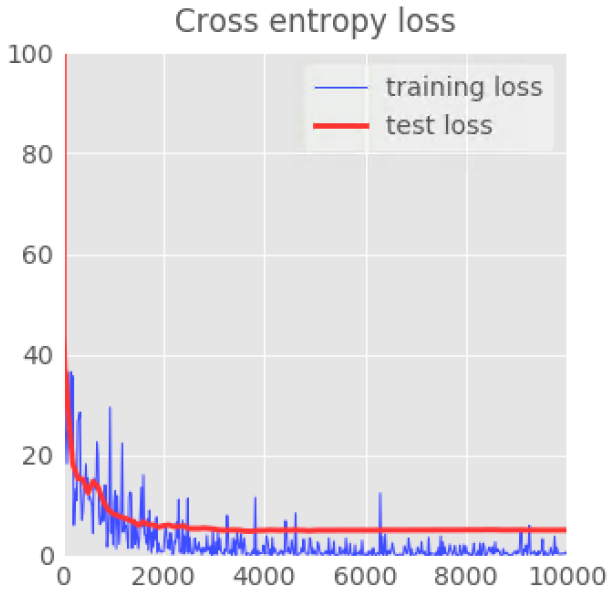


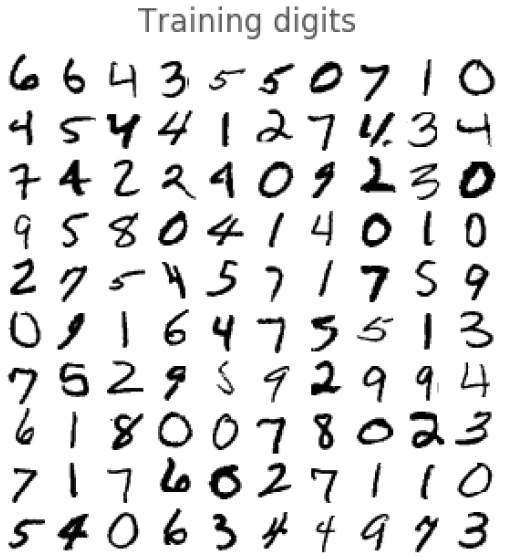
larger convolutional network

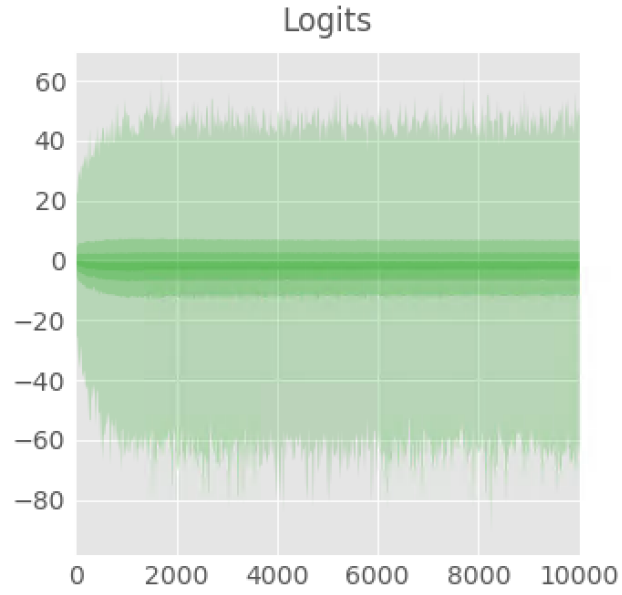
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

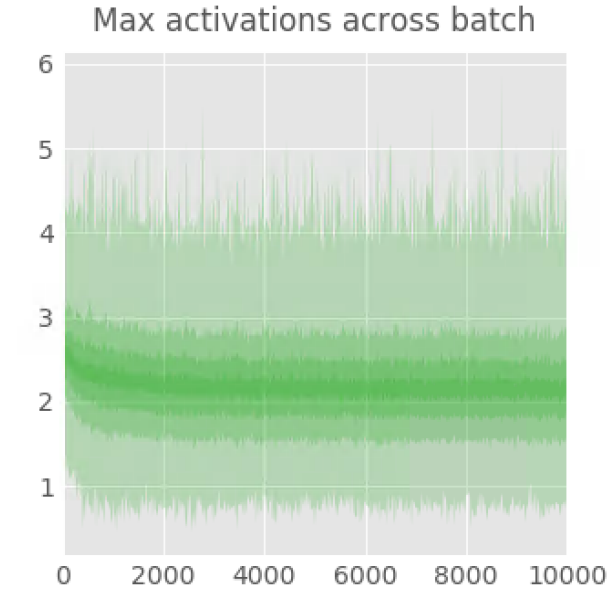
TensorFlow MNIST Tutorial

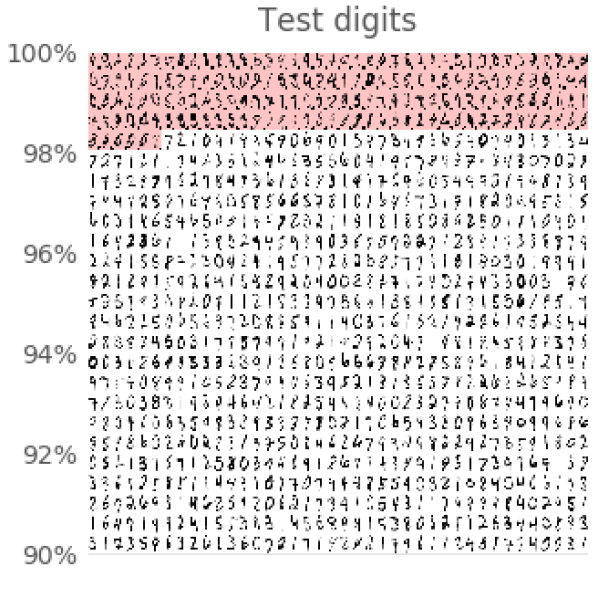












Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/

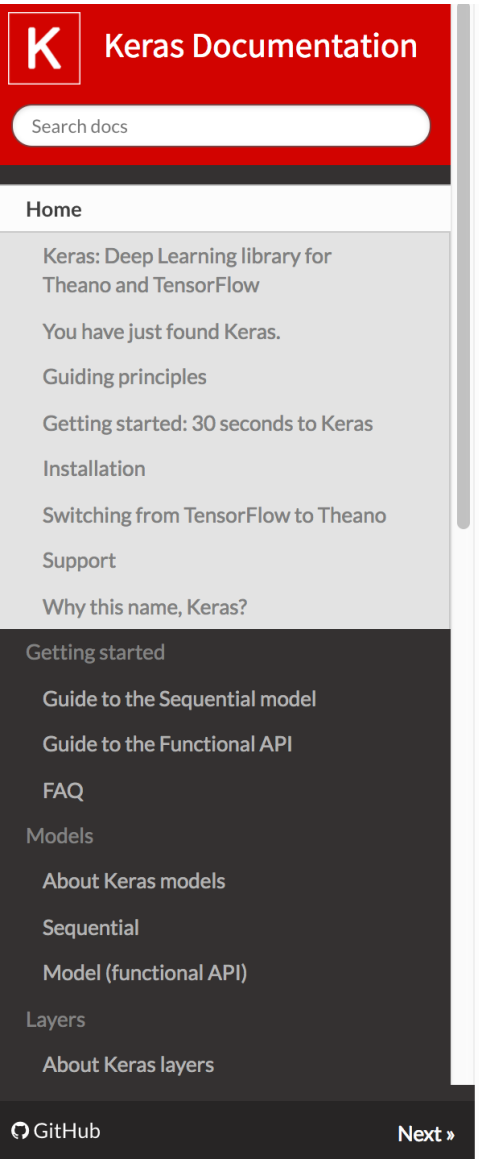
Keras

Keras

- Keras is a high-level neural networks API
- Written in Python and capable of running on top of either TensorFlow 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/

Keras



Docs » Home

Edit on GitHub

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow 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 2.7-3.5**.

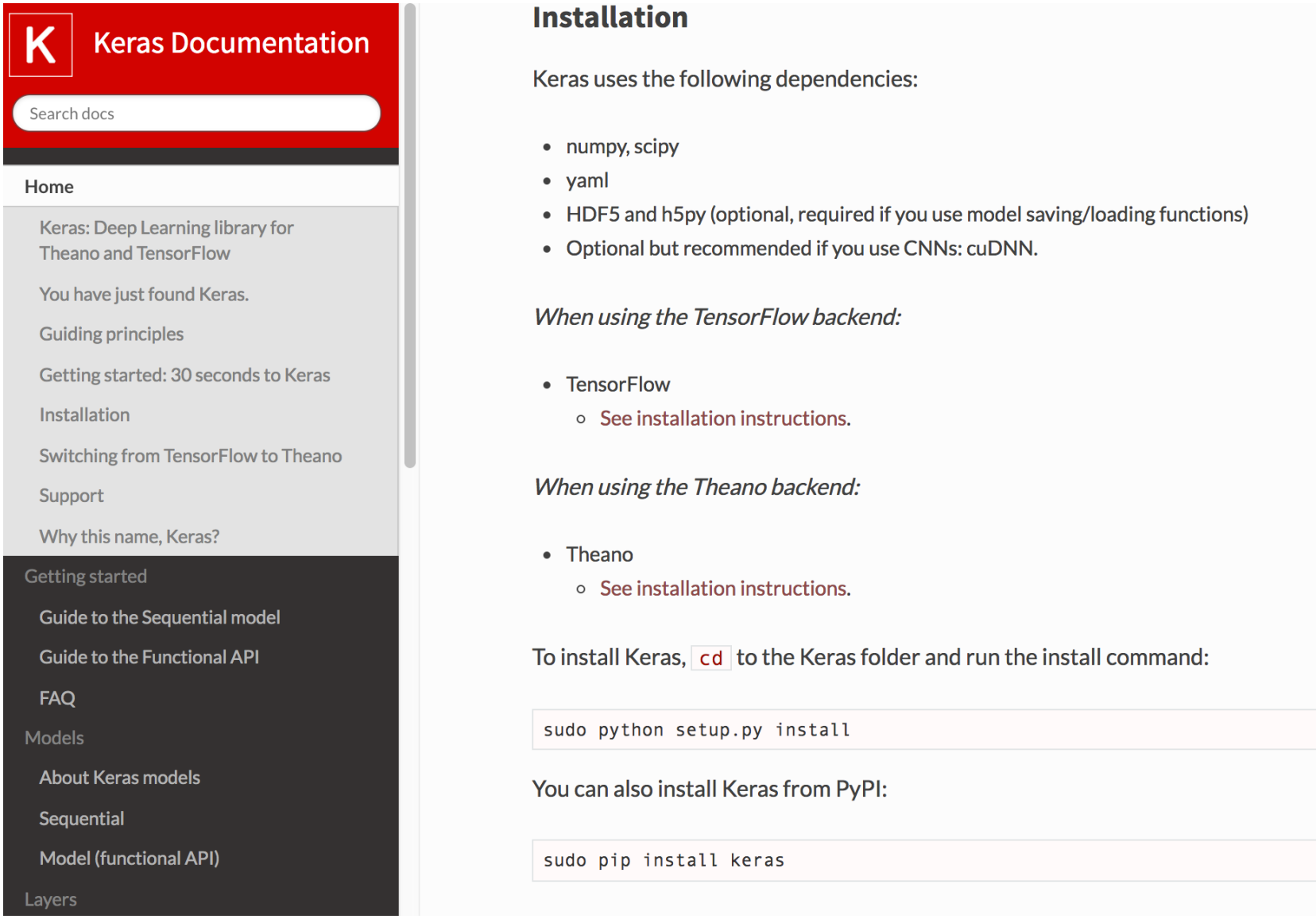
Guiding principles

- **User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity.** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can

http://keras.io/

Deep Learning with Keras

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

```
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

```
pip install Theano
```

```
conda install pydot
```

```
sudo pip install keras
```

```
pip install keras
```

```
pip install tensorflow
```

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

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

Gensim

pip install -U gensim

```
bash-3.2$ pip install -U gensim
Collecting gensim
  Downloading gensim-2.0.0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (5.6MB)
    100% |████████████████████████████████| 5.6MB 126kB/s
Requirement already up-to-date: six>=1.5.0 in ./anaconda/lib/python3.6/site-packages (fr
om gensim)
Collecting scipy>=0.7.0 (from gensim)
  Downloading scipy-0.19.0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (16.2MB)
    100% |████████████████████████████████| 16.2MB 43kB/s
Collecting smart-open>=1.2.1 (from gensim)
  Downloading smart_open-1.5.2.tar.gz
Collecting numpy>=1.3 (from gensim)
  Downloading numpy-1.12.1-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (4.4MB)
    100% |████████████████████████████████| 4.4MB 148kB/s
Collecting boto>=2.32 (from smart-open>=1.2.1->gensim)
  Downloading boto-2.46.1-py2.py3-none-any.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 372kB/s
Requirement already up-to-date: bz2file in ./anaconda/lib/python3.6/site-packages (from
smart-open>=1.2.1->gensim)
Collecting requests (from smart-open>=1.2.1->gensim)
  Downloading requests-2.13.0-py2.py3-none-any.whl (584kB)
    100% |████████████████████████████████| 593kB 632kB/s
Building wheels for collected packages: smart-open
  Running setup.py bdist_wheel for smart-open ... done
  Stored in directory: /Users/imyday/Library/Caches/pip/wheels/02/44/43/68e963ce2b45baef
a913a4e558bcd787403458afddffcf45ca
Successfully built smart-open
Installing collected packages: numpy, scipy, boto, requests, smart-open, gensim
  Found existing installation: numpy 1.11.3
    Uninstalling numpy-1.11.3:
      Successfully uninstalled numpy-1.11.3
  Found existing installation: scipy 0.18.1
    Uninstalling scipy-0.18.1:
      Successfully uninstalled scipy-0.18.1
  Found existing installation: boto 2.45.0
    DEPRECATION: Uninstalling a distutils installed project (boto) has been deprecated a
nd will be removed in a future version. This is due to the fact that uninstalling a dist
utils project will only partially uninstall the project.
    Uninstalling boto-2.45.0:
      Successfully uninstalled boto-2.45.0
  Found existing installation: requests 2.12.4
    Uninstalling requests-2.12.4:
      Successfully uninstalled requests-2.12.4
  Found existing installation: smart-open 1.4.0
    Uninstalling smart-open-1.4.0:
      Successfully uninstalled smart-open-1.4.0
  Found existing installation: gensim 1.0.1
    Uninstalling gensim-1.0.1:
      Successfully uninstalled gensim-1.0.1
Successfully installed boto-2.46.1 gensim-2.0.0 numpy-1.12.1 requests-2.13.0 scipy-0.19.
0 smart-open-1.5.2
bash-3.2$
```

Keras

sudo pip install keras

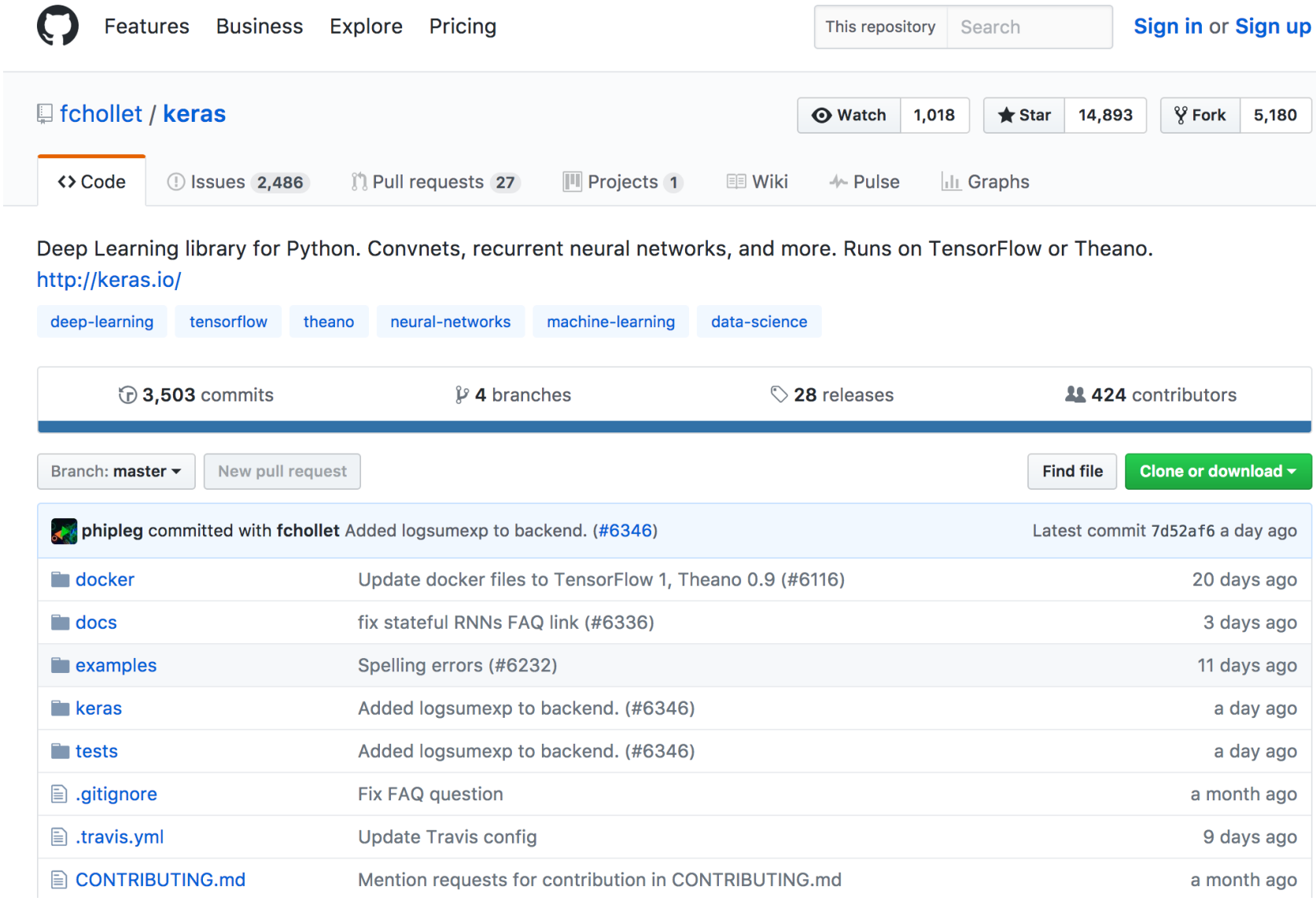
```
bash-3.2$ sudo pip install keras
Password:
The directory '/Users/imyday/Library/Caches/pip/http' or its parent directory is not owned by the current us
er and the cache has been disabled. Please check the permissions and owner of that directory. If executing p
ip with sudo, you may want sudo's -H flag.
The directory '/Users/imyday/Library/Caches/pip' or its parent directory is not owned by the current user an
d caching wheels has been disabled. check the permissions and owner of that directory. If executing pip with
 sudo, you may want sudo's -H flag.
Collecting keras
  Downloading Keras-2.0.3.tar.gz (196kB)
    100% |████████████████████████████████| 204kB 365kB/s
Collecting theano (from keras)
  Downloading Theano-0.9.0.tar.gz (3.1MB)
    100% |████████████████████████████████| 3.1MB 148kB/s
Requirement already satisfied: pyyaml in ./anaconda/lib/python3.6/site-packages (from keras)
Requirement already satisfied: six in ./anaconda/lib/python3.6/site-packages (from keras)
Requirement already satisfied: numpy>=1.9.1 in ./anaconda/lib/python3.6/site-packages (from theano->keras)
Requirement already satisfied: scipy>=0.14 in ./anaconda/lib/python3.6/site-packages (from theano->keras)
Installing collected packages: theano, keras
  Running setup.py install for theano ... done
  Running setup.py install for keras ... done
Successfully installed keras-2.0.3 theano-0.9.0
bash-3.2$
```

TensorFlow

pip install tensorflow

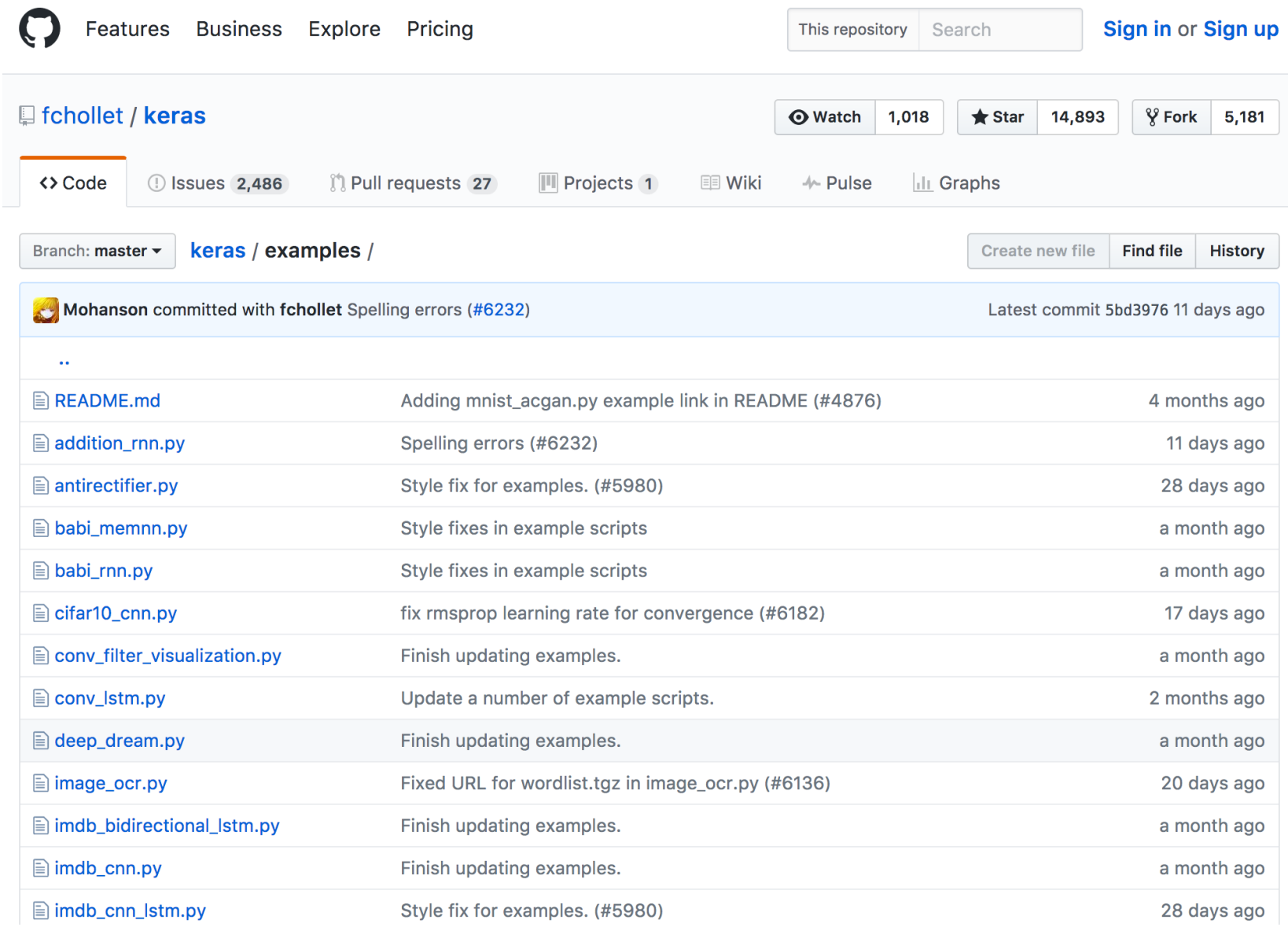
```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement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

Keras Github



https://github.com/fchollet/keras

Keras Examples



https://github.com/fchollet/keras/tree/master/examples

Keras MINST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Using TensorFlow backend.
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
Using TensorFlow backend.

Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 200s - loss: 0.3155 - acc: 0.9028 - val_loss: 0.0756 - val_acc: 0.9761
Epoch 2/12
60000/60000 [==============================] - 209s - loss: 0.1106 - acc: 0.9681 - val_loss: 0.0523 - val_acc: 0.9837
Epoch 3/12
60000/60000 [==============================] - 220s - loss: 0.0834 - acc: 0.9749 - val_loss: 0.0416 - val_acc: 0.9852
Epoch 4/12
60000/60000 [==============================] - 224s - loss: 0.0700 - acc: 0.9795 - val_loss: 0.0392 - val_acc: 0.9879
Epoch 5/12
60000/60000 [==============================] - 229s - loss: 0.0614 - acc: 0.9818 - val_loss: 0.0358 - val_acc: 0.9871
Epoch 6/12
60000/60000 [==============================] - 227s - loss: 0.0558 - acc: 0.9828 - val_loss: 0.0345 - val_acc: 0.9880
Epoch 7/12
60000/60000 [==============================] - 217s - loss: 0.0498 - acc: 0.9850 - val_loss: 0.0337 - val_acc: 0.9883
Epoch 8/12
60000/60000 [==============================] - 217s - loss: 0.0473 - acc: 0.9865 - val_loss: 0.0294 - val_acc: 0.9899
Epoch 9/12
60000/60000 [==============================] - 217s - loss: 0.0439 - acc: 0.9872 - val_loss: 0.0316 - val_acc: 0.9889
Epoch 10/12
60000/60000 [==============================] - 217s - loss: 0.0415 - acc: 0.9871 - val_loss: 0.0319 - val_acc: 0.9897
Epoch 11/12
60000/60000 [==============================] - 217s - loss: 0.0380 - acc: 0.9889 - val_loss: 0.0275 - val_acc: 0.9904
Epoch 12/12
60000/60000 [==============================] - 215s - loss: 0.0376 - acc: 0.9889 - val_loss: 0.0285 - val_acc: 0.9905
Test loss: 0.0285460013417
Test accuracy: 0.9905
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Keras MINST CNN

```
python mnist_cnn.py
Using TensorFlow backend.
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 108s - loss: 0.3510 - acc: 0.8921 - val_loss: 0.0880 - val_acc: 0.9738
Epoch 2/12
60000/60000 [==============================] - 106s - loss: 0.1200 - acc: 0.9649 - val_loss: 0.0567 - val_acc: 0.9820
Epoch 3/12
60000/60000 [==============================] - 104s - loss: 0.0889 - acc: 0.9735 - val_loss: 0.0438 - val_acc: 0.9856
Epoch 4/12
60000/60000 [==============================] - 106s - loss: 0.0744 - acc: 0.9783 - val_loss: 0.0392 - val_acc: 0.9862
Epoch 5/12
60000/60000 [==============================] - 106s - loss: 0.0648 - acc: 0.9807 - val_loss: 0.0363 - val_acc: 0.9873
Epoch 6/12
60000/60000 [==============================] - 109s - loss: 0.0574 - acc: 0.9840 - val_loss: 0.0348 - val_acc: 0.9884
Epoch 7/12
60000/60000 [==============================] - 104s - loss: 0.0522 - acc: 0.9842 - val_loss: 0.0324 - val_acc: 0.9890
Epoch 8/12
60000/60000 [==============================] - 104s - loss: 0.0484 - acc: 0.9856 - val_loss: 0.0315 - val_acc: 0.9894
Epoch 9/12
60000/60000 [==============================] - 104s - loss: 0.0447 - acc: 0.9870 - val_loss: 0.0296 - val_acc: 0.9902
Epoch 10/12
60000/60000 [==============================] - 109s - loss: 0.0419 - acc: 0.9877 - val_loss: 0.0338 - val_acc: 0.9894
Epoch 11/12
60000/60000 [==============================] - 104s - loss: 0.0405 - acc: 0.9879 - val_loss: 0.0301 - val_acc: 0.9896
Epoch 12/12
60000/60000 [==============================] - 127s - loss: 0.0391 - acc: 0.9883 - val_loss: 0.0304 - val_acc: 0.9899
Test loss: 0.030424870987
Test accuracy: 0.9899
```

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 15 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.datasets import imdb

# set parameters:
max_features = 5000
maxlen = 400
batch_size = 32
embedding_dims = 50
filters = 250
kernel_size = 3
hidden_dims = 250
epochs = 2

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

print('Build model...')
model = Sequential()

# we start off with an efficient embedding layer which maps
# our vocab indices into embedding_dims dimensions
model.add(Embedding(max_features,
                    embedding_dims,
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_cnn.py

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 19 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
model.add(Embedding(max_features,
                    embedding_dims,
                    input_length=maxlen))
model.add(Dropout(0.2))

# we add a Convolution1D, which will learn filters
# word group filters of size filter_length:
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))
```

```
Using TensorFlow backend.
```

```
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz
25000 train sequences
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_cnn.py

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 13 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

```
# we use max pooling:
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))
```

```
Using TensorFlow backend.
```

```
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [==============================] - 266s - loss: 0.4110 - acc: 0.8012 - val_loss: 0.2965 - val_acc: 0.8739
Epoch 2/2
25000/25000 [==============================] - 286s - loss: 0.2429 - acc: 0.9020 - val_loss: 0.2726 - val_acc: 0.8862
```

Out[1]: <keras.callbacks.History at 0x11dc37b00>

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_cnn.py

Keras IMDB CNN

```
python imdb_cnn.py
Using TensorFlow backend.
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [==============================] - 157s - loss: 0.4050 - acc: 0.8065 - val_loss: 0.2924 - val_acc: 0.8750
Epoch 2/2
25000/25000 [==============================] - 128s - loss: 0.2433 - acc: 0.9040 - val_loss: 0.2701 - val_acc: 0.8865
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at 0x0000019F153C2A20>>
Traceback (most recent call last):
  File "C:\Program Files\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 587, in __del__
AttributeError: 'NoneType' object has no attribute 'TF_NewStatus'
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_cnn.py

Keras IMDB LSTM

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb

max_features = 20000
maxlen = 80  # cut texts after this number of words (among top max_features most common words)
batch_size = 32

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=15,
          validation_data=(x_test, y_test))
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

Keras IMDB LSTM

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

Keras IMDB LSTM

```
max_features = 20000
maxlen = 80  # cut texts after this number of words (among top
max_features most common words)
batch_size = 32

print('Loading data...')
(x_train, y_train), (x_test, y_test) =
imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

Keras IMDB LSTM

```
print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

Keras IMDB LSTM

```
print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=15,
          validation_data=(x_test, y_test))
score, acc = model.evaluate(x_test, y_test,

batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

Keras IMDB LSTM

```
python imdb_lstm.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 80)
x_test shape: (25000, 80)
Build model...
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/15
25000/25000 [==============================] - 111s - loss: 0.4561 - acc: 0.7837 - val_loss: 0.3892 - val_acc: 0.8275
Epoch 2/15
25000/25000 [==============================] - 112s - loss: 0.2947 - acc: 0.8792 - val_loss: 0.4266 - val_acc: 0.8353
Epoch 3/15
25000/25000 [==============================] - 111s - loss: 0.2122 - acc: 0.9178 - val_loss: 0.4133 - val_acc: 0.8284
Epoch 4/15
25000/25000 [==============================] - 112s - loss: 0.1461 - acc: 0.9450 - val_loss: 0.4670 - val_acc: 0.8260
Epoch 5/15
25000/25000 [==============================] - 113s - loss: 0.1038 - acc: 0.9633 - val_loss: 0.5580 - val_acc: 0.8203
Epoch 6/15
25000/25000 [==============================] - 113s - loss: 0.0739 - acc: 0.9749 - val_loss: 0.6738 - val_acc: 0.8174
Epoch 7/15
25000/25000 [==============================] - 113s - loss: 0.0542 - acc: 0.9810 - val_loss: 0.7463 - val_acc: 0.8154
Epoch 8/15
25000/25000 [==============================] - 113s - loss: 0.0428 - acc: 0.9856 - val_loss: 0.8131 - val_acc: 0.8157
Epoch 9/15
25000/25000 [==============================] - 115s - loss: 0.0334 - acc: 0.9889 - val_loss: 0.8566 - val_acc: 0.8165
Epoch 10/15
25000/25000 [==============================] - 114s - loss: 0.0248 - acc: 0.9920 - val_loss: 0.9186 - val_acc: 0.8165
Epoch 11/15
25000/25000 [==============================] - 116s - loss: 0.0156 - acc: 0.9955 - val_loss: 0.9016 - val_acc: 0.8082
Epoch 12/15
25000/25000 [==============================] - 117s - loss: 0.0196 - acc: 0.9942 - val_loss: 0.9720 - val_acc: 0.8124
Epoch 13/15
25000/25000 [==============================] - 120s - loss: 0.0152 - acc: 0.9957 - val_loss: 1.0064 - val_acc: 0.8148
Epoch 14/15
25000/25000 [==============================] - 121s - loss: 0.0128 - acc: 0.9961 - val_loss: 1.1103 - val_acc: 0.8121
Epoch 15/15
25000/25000 [==============================] - 114s - loss: 0.0110 - acc: 0.9970 - val_loss: 1.0173 - val_acc: 0.8132
25000/25000 [==============================] - 23s
Test score: 1.01734088922
Test accuracy: 0.8132
```

Keras IMDB FastText

```
python imdb_fasttext.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Average train sequence length: 238
Average test sequence length: 230
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [==============================] - 14s - loss: 0.6102 - acc: 0.7397 - val_loss: 0.5034 - val_acc: 0.8105
Epoch 2/5
25000/25000 [==============================] - 14s - loss: 0.4019 - acc: 0.8656 - val_loss: 0.3697 - val_acc: 0.8654
Epoch 3/5
25000/25000 [==============================] - 14s - loss: 0.3025 - acc: 0.8959 - val_loss: 0.3199 - val_acc: 0.8791
Epoch 4/5
25000/25000 [==============================] - 14s - loss: 0.2521 - acc: 0.9113 - val_loss: 0.2971 - val_acc: 0.8848
Epoch 5/5
25000/25000 [==============================] - 14s - loss: 0.2181 - acc: 0.9249 - val_loss: 0.2899 - val_acc: 0.8855
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at
0x000001E3257DB438>>
Traceback (most recent call last):
  File "C:\Program Files\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 587, in __del__
AttributeError: 'NoneType' object has no attribute 'TF_NewStatus'
```

Keras IMDB CNN LSTM

```
python imdb_cnn_lstm_2.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 100)
x_test shape: (25000, 100)
Build model...
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [==============================] - 64s - loss: 0.3824 - acc: 0.8238 - val_loss: 0.3591 - val_acc: 0.8467
Epoch 2/2
25000/25000 [==============================] - 63s - loss: 0.1953 - acc: 0.9261 - val_loss: 0.3827 - val_acc: 0.8488
24990/25000 [============================>.] - ETA: 0s
Test score: 0.382728585386
Test accuracy: 0.848799994493
```

imdb_lstm_2.py

```
from __future__ import print_function

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb

py_filename = 'imdb_lstm_2.py'
max_features = 20000
maxlen = 80  # cut texts after this number of words (among top max_features
most common words)
batch_size = 32
epochs = 20 #60

#%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

import codecs
import datetime
import timeit
timer_start = timeit.default_timer()
#timer_end = timeit.default_timer()
#print('timer_end - timer_start', timer_end - timer_start)
```

imdb_lstm_2.py

```
def getDateTimeNow():
    strnow = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    return strnow

def read_file_utf8(filename):
    with codecs.open(filename,'r',encoding='utf-8') as f:
        text = f.read()
    return text

def write_file_utf8(filename,text):
    with codecs.open(filename, 'w', encoding='utf-8') as f:
        f.write(text)
        f.close()

def log_file_utf8(filename, text):
    with codecs.open(filename, 'a', encoding='utf-8') as f:
        #append file
        f.write(text + '\n')
        f.close()

log_file_utf8("logfile.txt", '***** ' + py_filename + ' *****')
log_file_utf8("logfile.txt", '***** Start DateTime: ' + getDateTimeNow())

print('Start: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
```

imdb_lstm_2.py

```
print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

imdb_lstm_2.py

```
print('Train...')
print('model.fit: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
history = model.fit(x_train, y_train,
          batch_size = batch_size,
          epochs = epochs,
          validation_data = (x_test, y_test))

score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

imdb_lstm_2.py

```
timer_end = timeit.default_timer()
print('Timer: ', str(round(timer_end - timer_start, 2)), 's')
print('DateTime: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
log_file_utf8("logfile.txt", 'Timer: ' + str(round(timer_end - timer_start, 2))
+ ' s')
log_file_utf8("logfile.txt", '***** End Datetime: ' +
datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))

# summarize history for accuracy
#http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
print('history.history.keys():', history.history.keys())
print('history.history:', history.history)
log_file_utf8("logfile.txt", 'history.history:' + str(history.history))
```

imdb_lstm_2.py

```
# Deep Learning Training Visualization
plt.figure(figsize=(10, 8))  # make separate figure
ax1 = plt.subplot(2, 1, 1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
ax1.xaxis.set_major_locator(plt.NullLocator())
#plt.xlabel('epoch')
plt.legend(['train acc', 'test val_acc'], loc='upper left')
#plt.show()
ax2 = plt.subplot(2, 1, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train loss', 'test val_loss'], loc='upper left')
plt.savefig("training_accuacy_loss_" + py_filename + "_" + str(epochs) +
".png", dpi= 300)
```

imdb_lstm_2.py

```
#Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\t'  + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t ' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))
#plt.show()
```

python filename.py

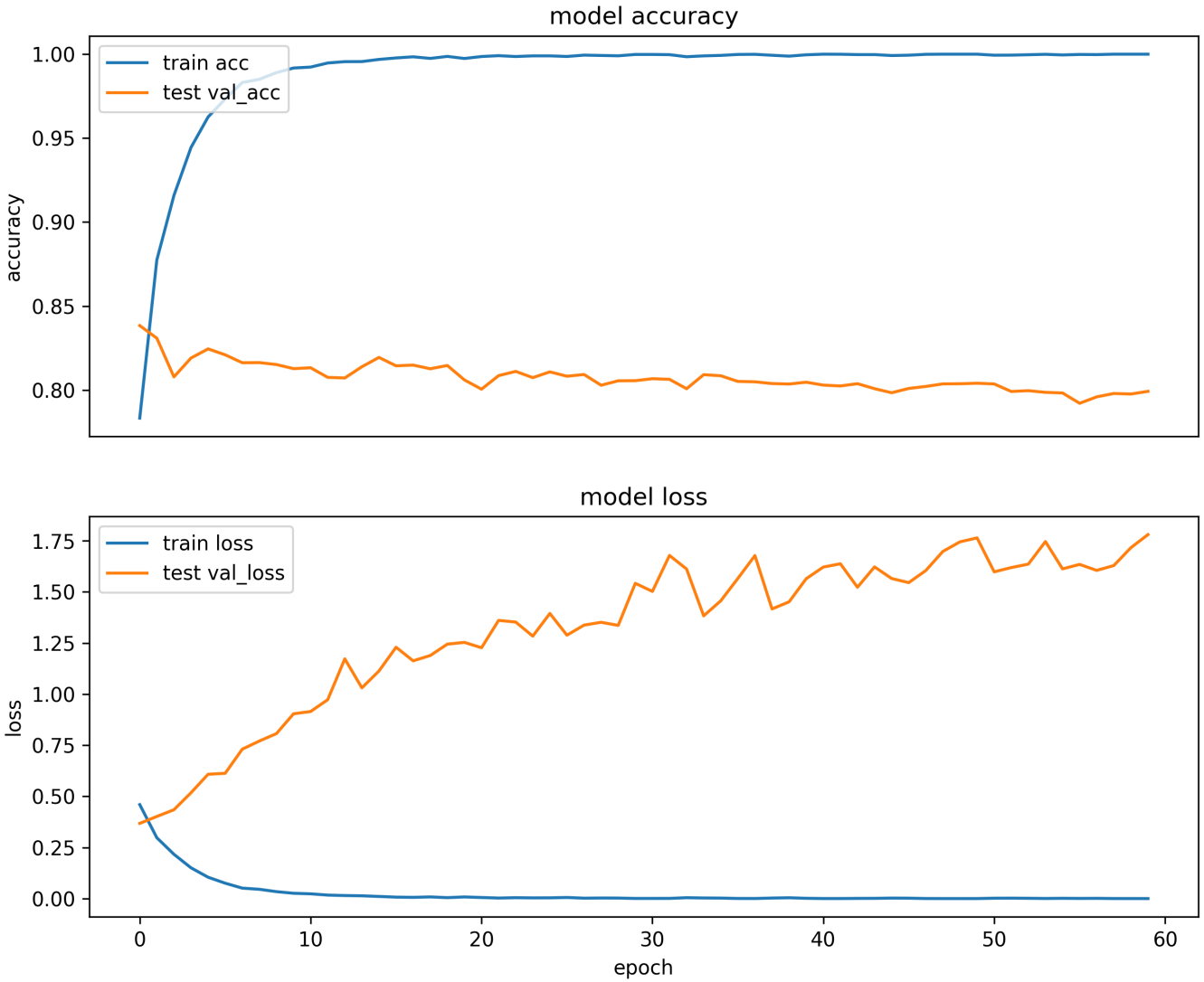
```
python imdb_fasttext_2.py
python imdb_cnn_2.py
python imdb_lstm_2.py
python imdb_cnn_lstm_2.py
python imdb_bidirectional_lstm_2.py
```

Deep Learning Summary

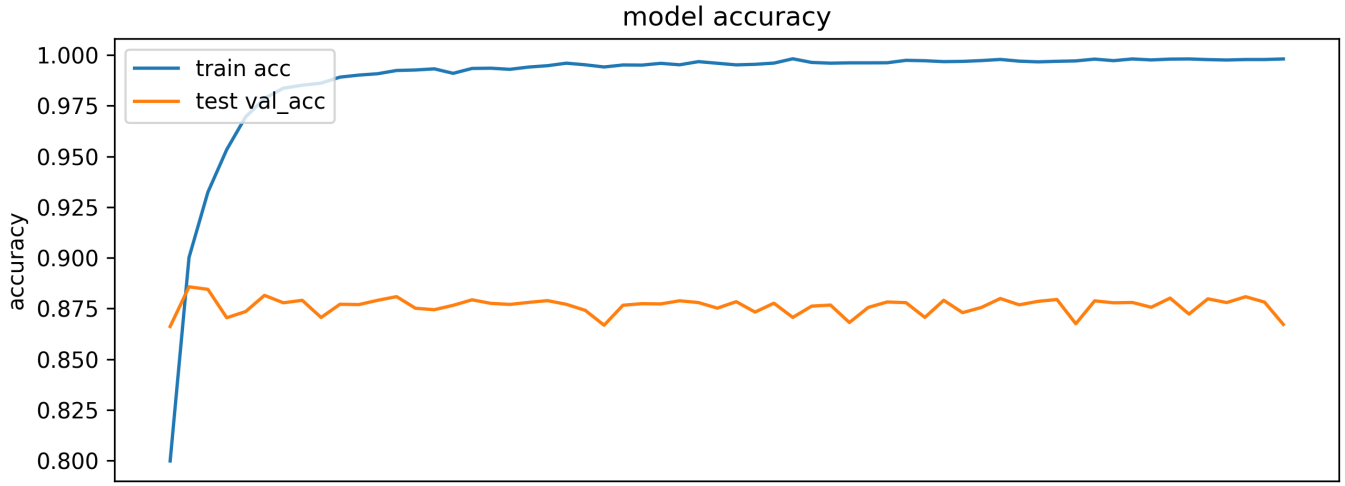
```
 #Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\t'  + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t ' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))
```

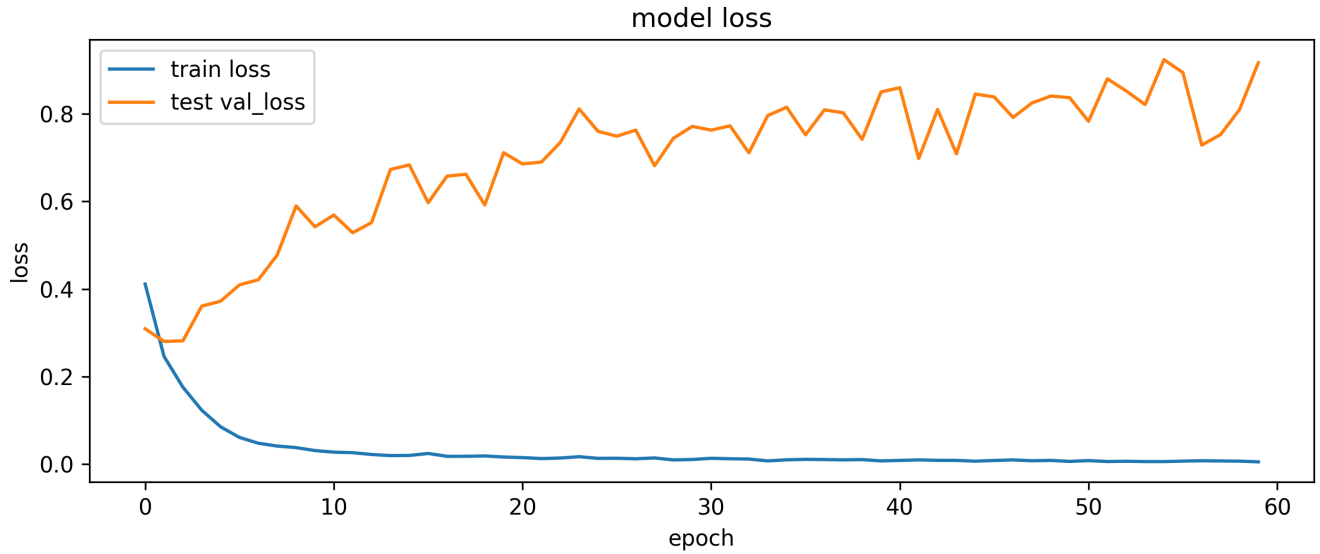
Model	epochs	Score	Accuracy	Timer (s)
imdb_lstm_2.py	30	0.6440	0.8540	**682.57**
imdb_cnn_2.py	30	0.7186	**0.8775**	4320.38
imdb_lstm_2.py	30	1.5716	0.8052	3958.93
imdb_cnn_lstm_2.py	30	1.3105	0.8240	2471.65
imdb_bidirectional_lstm_2.py	30	1.4083	0.8255	4344.36
imdb_fasttext_2.py	30	**0.6439**	0.8540	1117.78
imdb_fasttext_2.py	60	1.2335	0.8407	1297.02
imdb_cnn_2.py	60	0.9170	0.8672	8507.48
imdb_lstm_2.py	60	1.7803	0.7992	8039.67
imdb_cnn_lstm_2.py	60	1.4623	0.8137	4912.25
imdb_bidirectional_lstm_2.py	60	1.8975	0.8138	8589.17

imdb_lstm_2.py

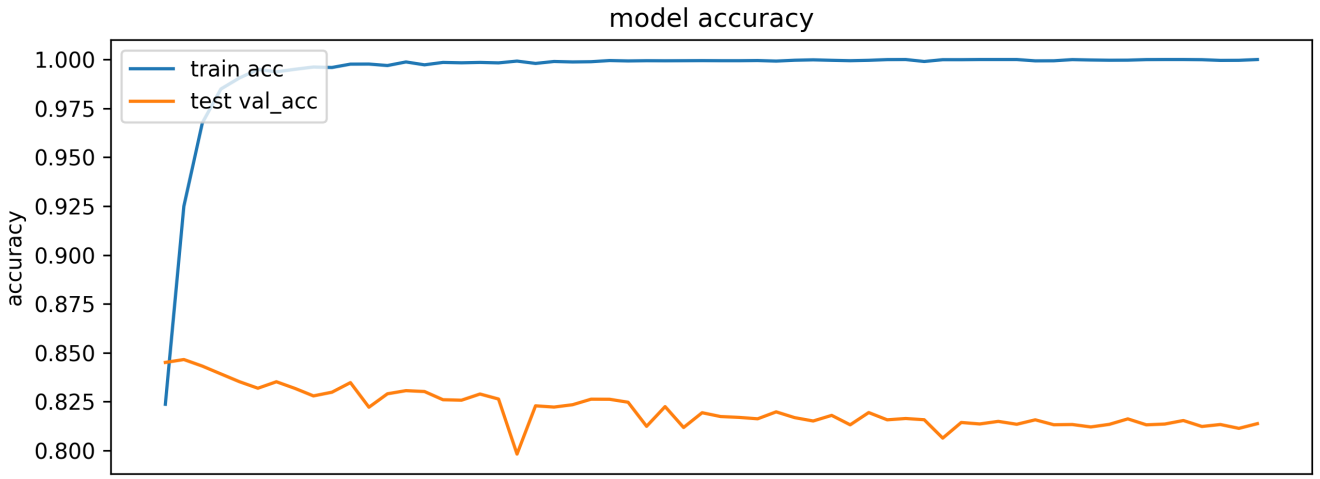


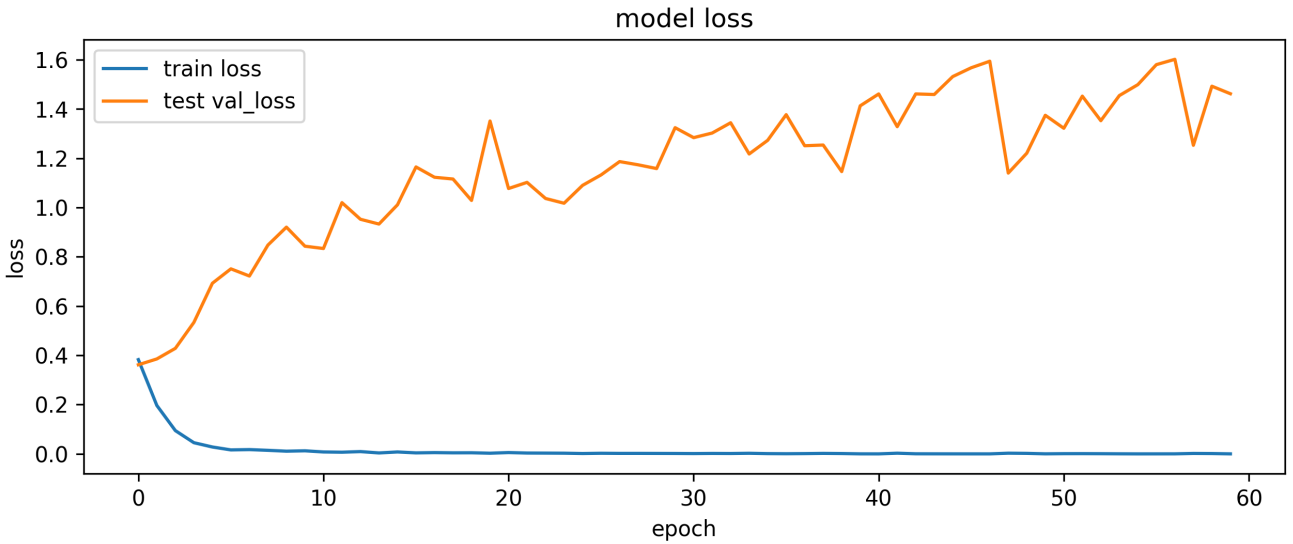
imdb_cnn_2.py


model accuracy
train acc
test val_acc
accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800

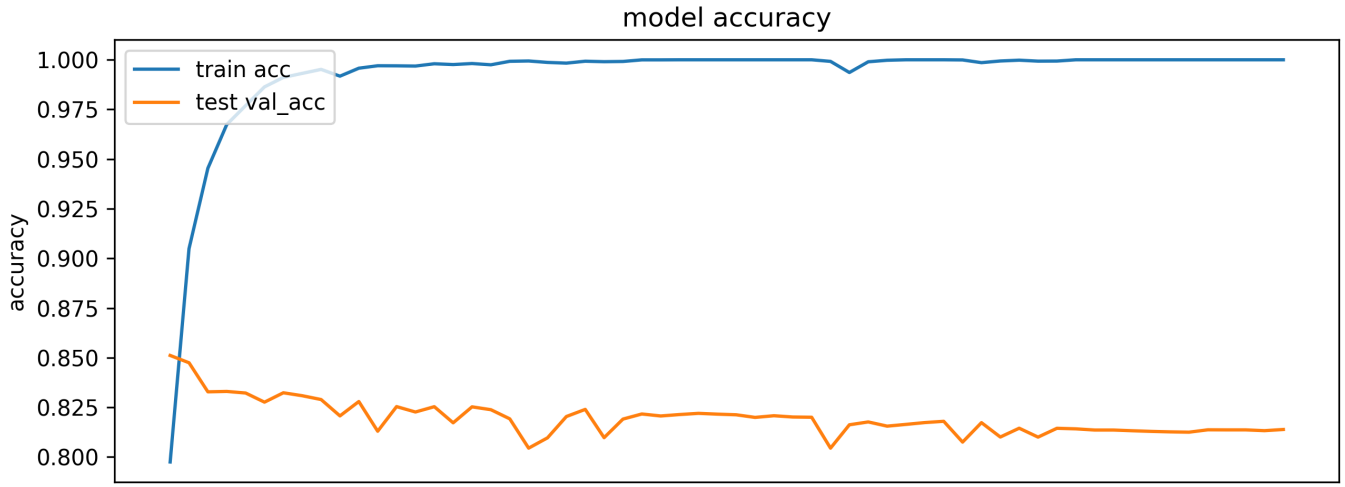

model loss
train loss
test val_loss
loss
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
epoch

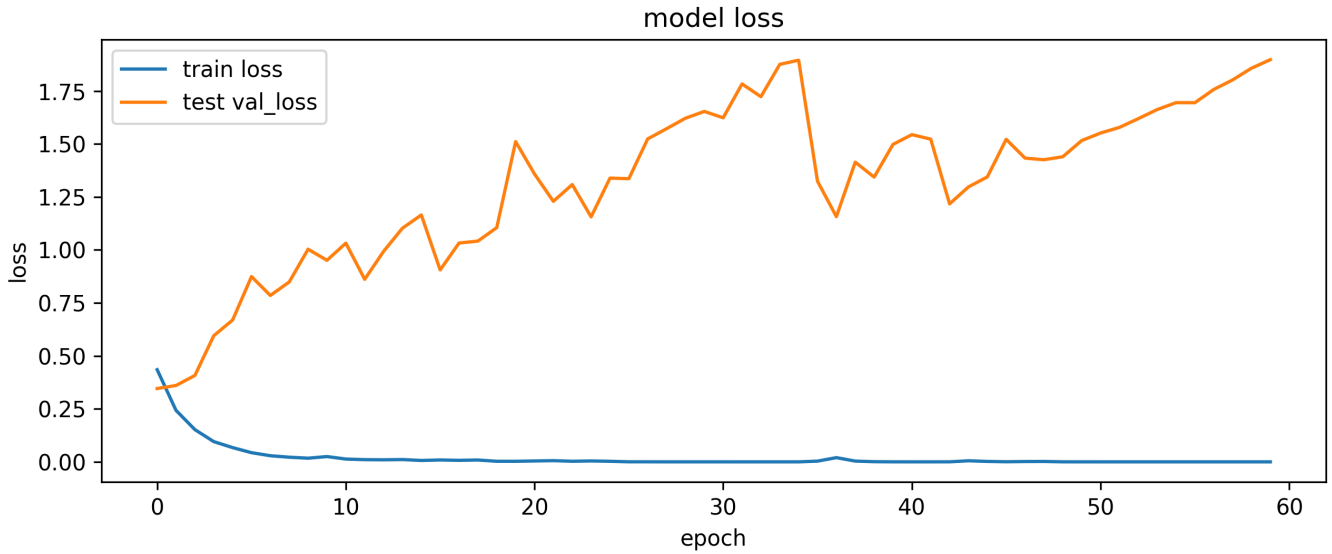
imdb_cnn_lstm_2.py



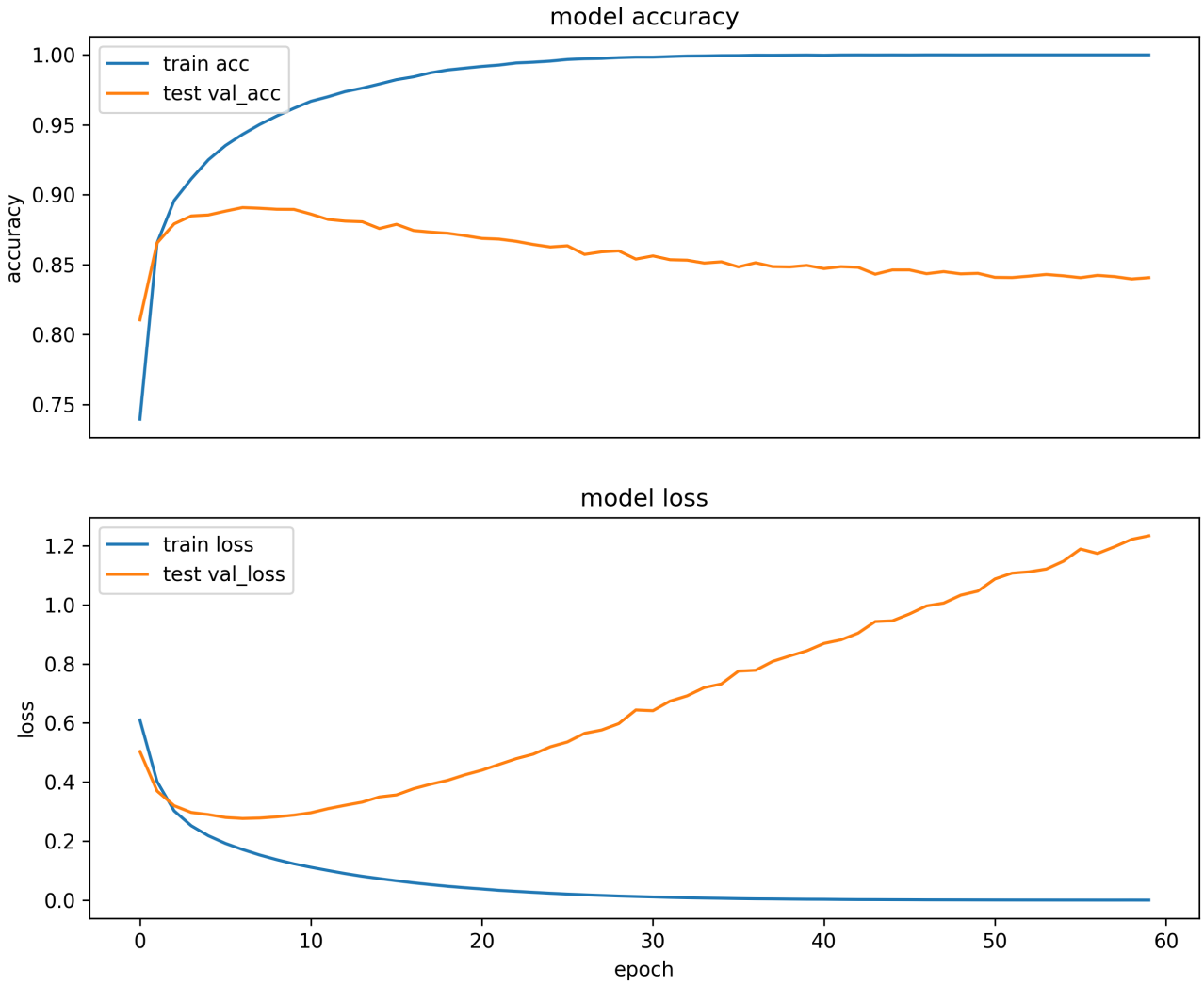


imdb_bidirectional_lstm_2.py





imdb_fasttext_2.py



Deep Learning with CPU vs. GPU

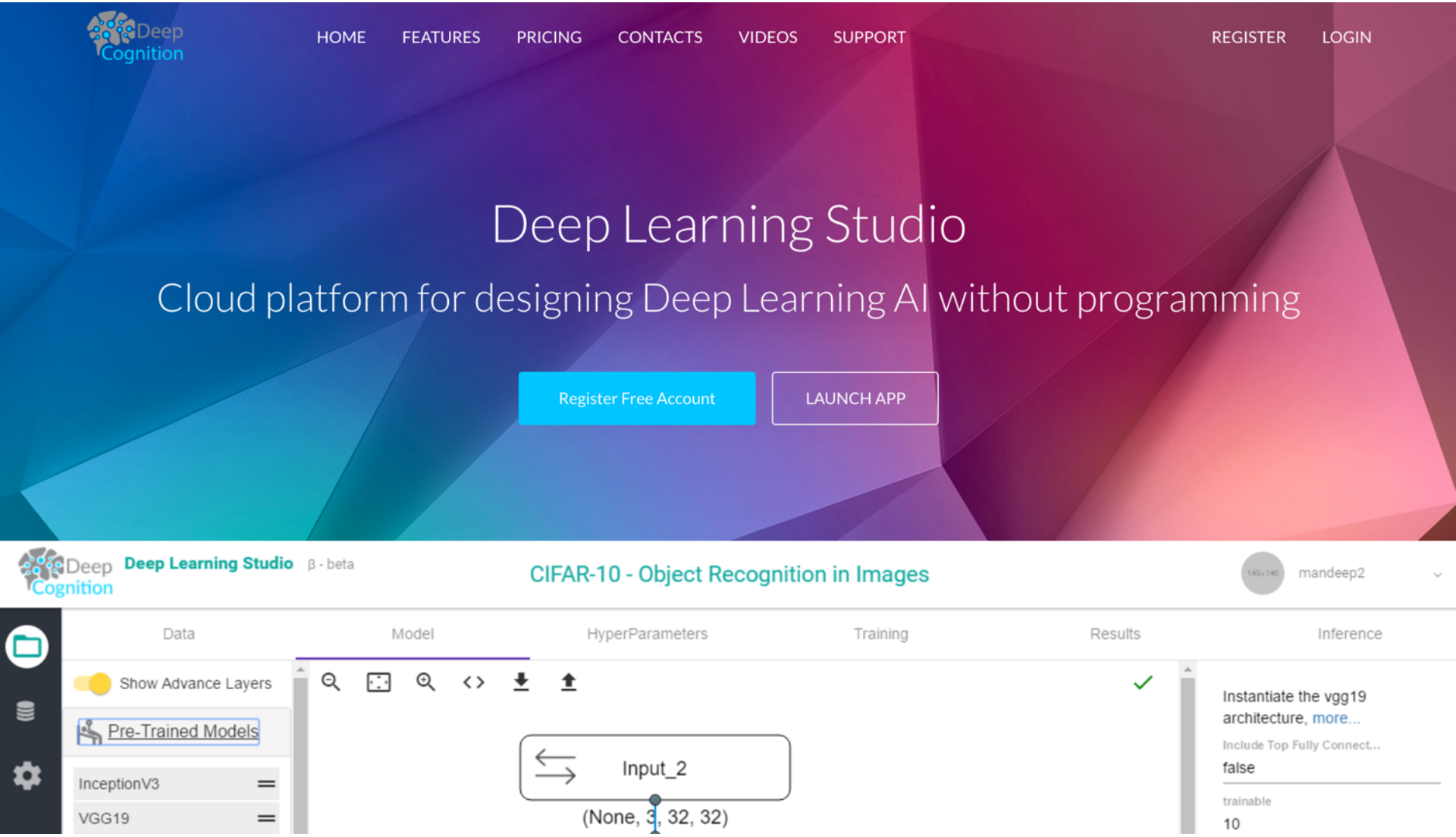
Timings:

Hardware	Backend	Time / Epoch
CPU	TF	3 hrs
Titan X (maxwell)	TF	4 min
Titan X (maxwell)	TH	7 min

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_acgan.py

Deep Learning Studio

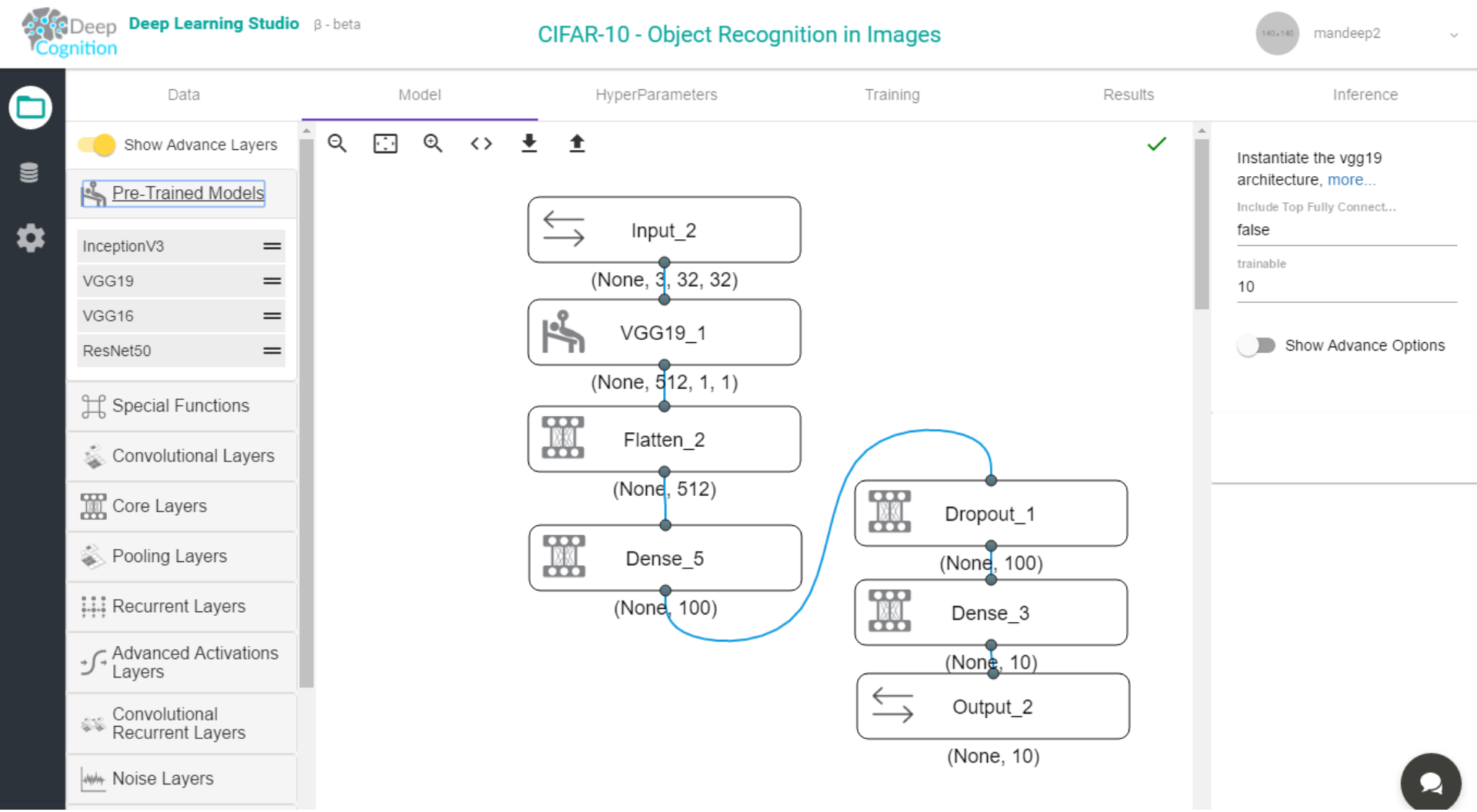
Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

Deep Learning Studio

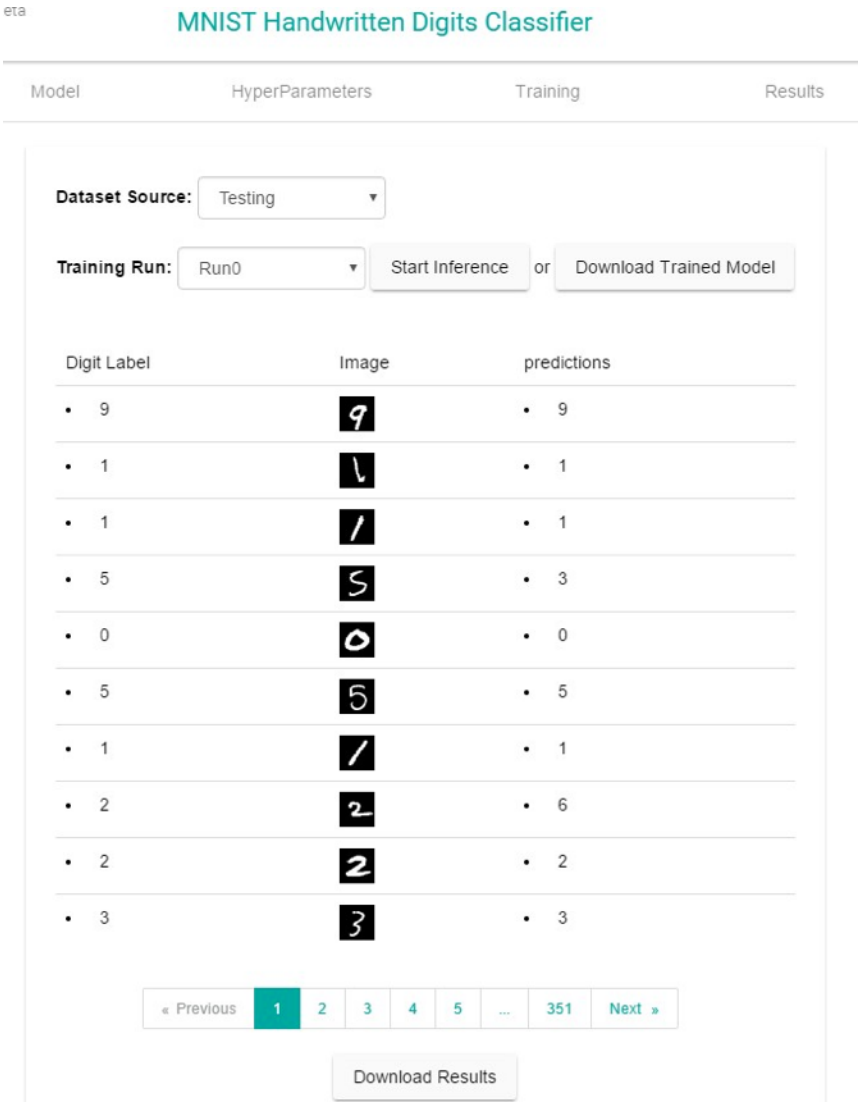
Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

References

- Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
- Sebastian Raschka (2015), Python Machine Learning, Packt Publishing
- 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
- TensorFlow: https://www.tensorflow.org/
- Theano: http://deeplearning.net/software/theano/
- Keras: http://keras.io/
- Deep Learning Studio: Cloud platform for designing Deep Learning AI without programming, http://deepcognition.ai/
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow