Tamkang University

Big Data Mining

Deep Learning for Finance Big Data with TensorFlow

1071BDM10
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master’s Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)

Min-Yuh Day, Ph.D.
Assistant Professor
[Department of Information Management](http://mail.tku.edu.tw/myday)
Tamkang University

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

2018-11-26

Course Schedule (1/2)

Tamkang University

Week	Date	Subject/Topics
1	2018/09/10	Course Orientation for Big Data Mining
2	2018/09/17	ABC: AI, Big Data, Cloud Computing
3	2018/09/24	Mid-Autumn Festival (Day off)
4	2018/10/01	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data
5	2018/10/08	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem
6	2018/10/15	Foundations of Big Data Mining in Python
7	2018/10/22	Supervised Learning: Classification and Prediction
8	2018/10/29	Unsupervised Learning: Cluster Analysis
9	2018/11/05	Unsupervised Learning: Association Analysis

Course Schedule (2/2)

Tamkang University

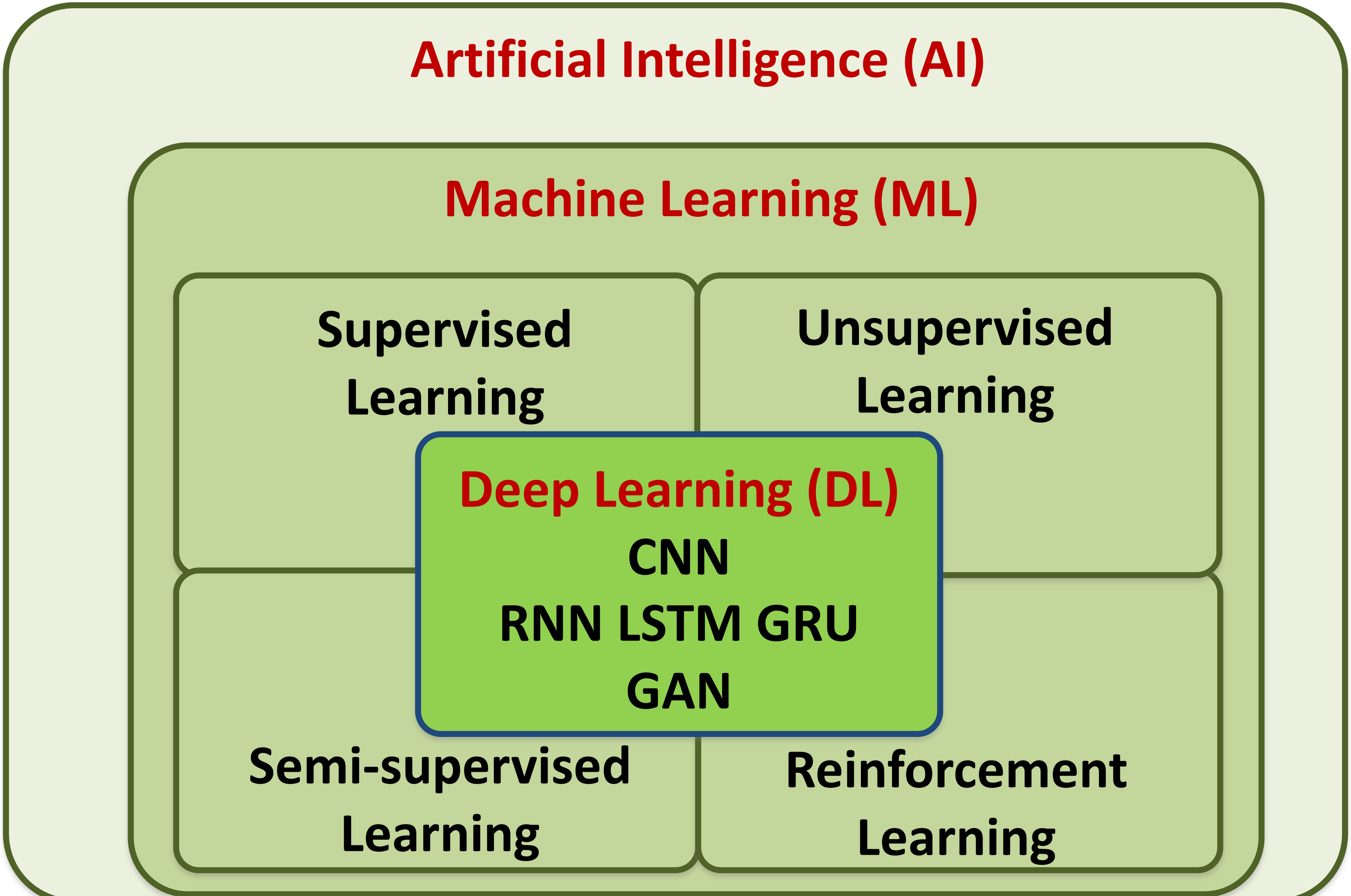
Week	Date	Subject/Topics
10	2018/11/12	Midterm Project Report
11	2018/11/19	Machine Learning with Scikit-Learn in Python
12	2018/11/26	Deep Learning for Finance Big Data with TensorFlow
13	2018/12/03	Convolutional Neural Networks (CNN)
14	2018/12/10	Recurrent Neural Networks (RNN)
15	2018/12/17	Reinforcement Learning (RL)
16	2018/12/24	Social Network Analysis (SNA)
17	2018/12/31	Bridge Holiday (Extra Day Off)
18	2019/01/07	Final Project Presentation

Deep Learning for Finance Big Data with TensorFlow

Outline

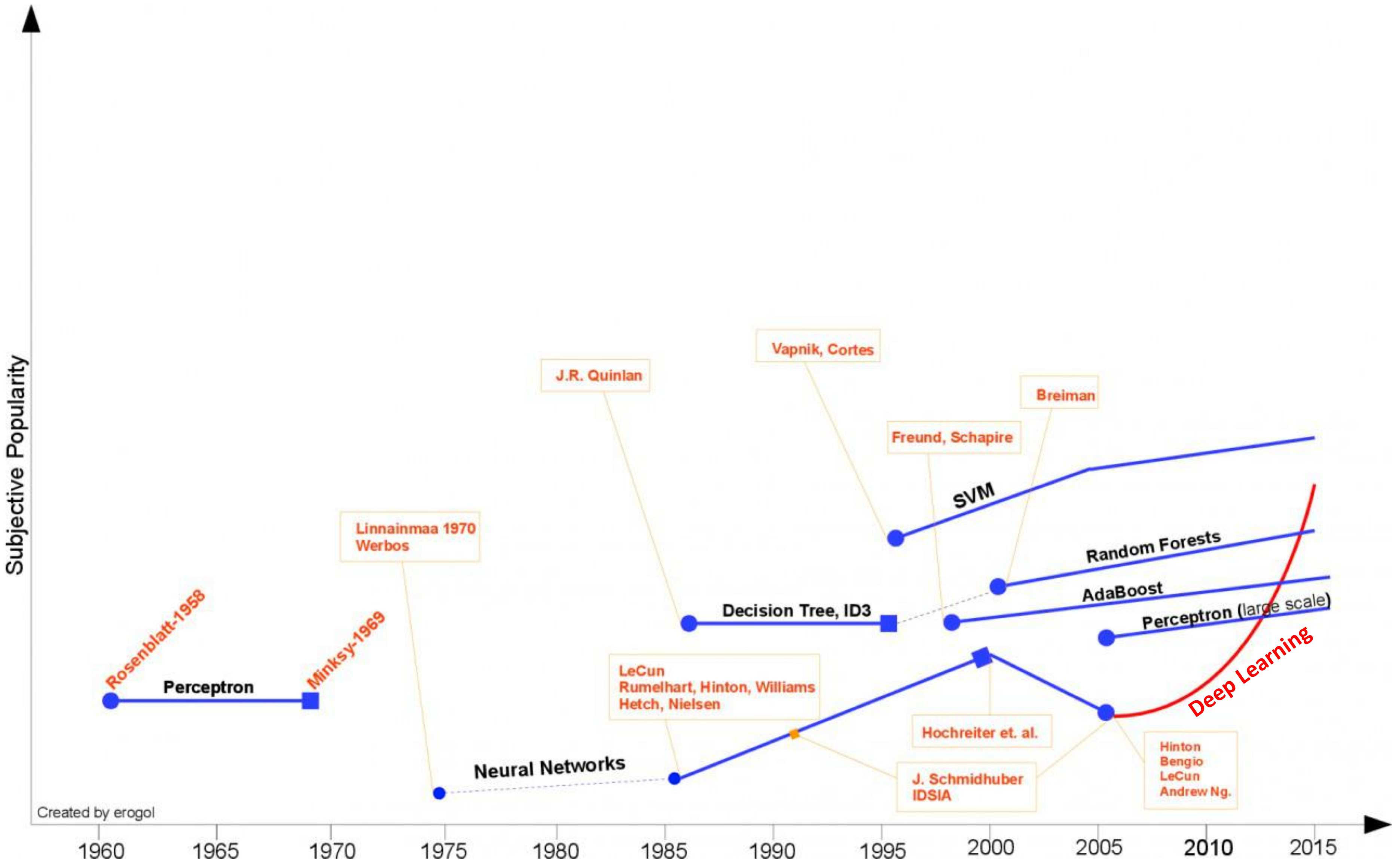
- Deep Learning for Finance Big Data with TensorFlow
  - Deep Learning
  - Finance Big Data
  - TensorFlow

AI, ML, DL



Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

Deep Learning Evolution



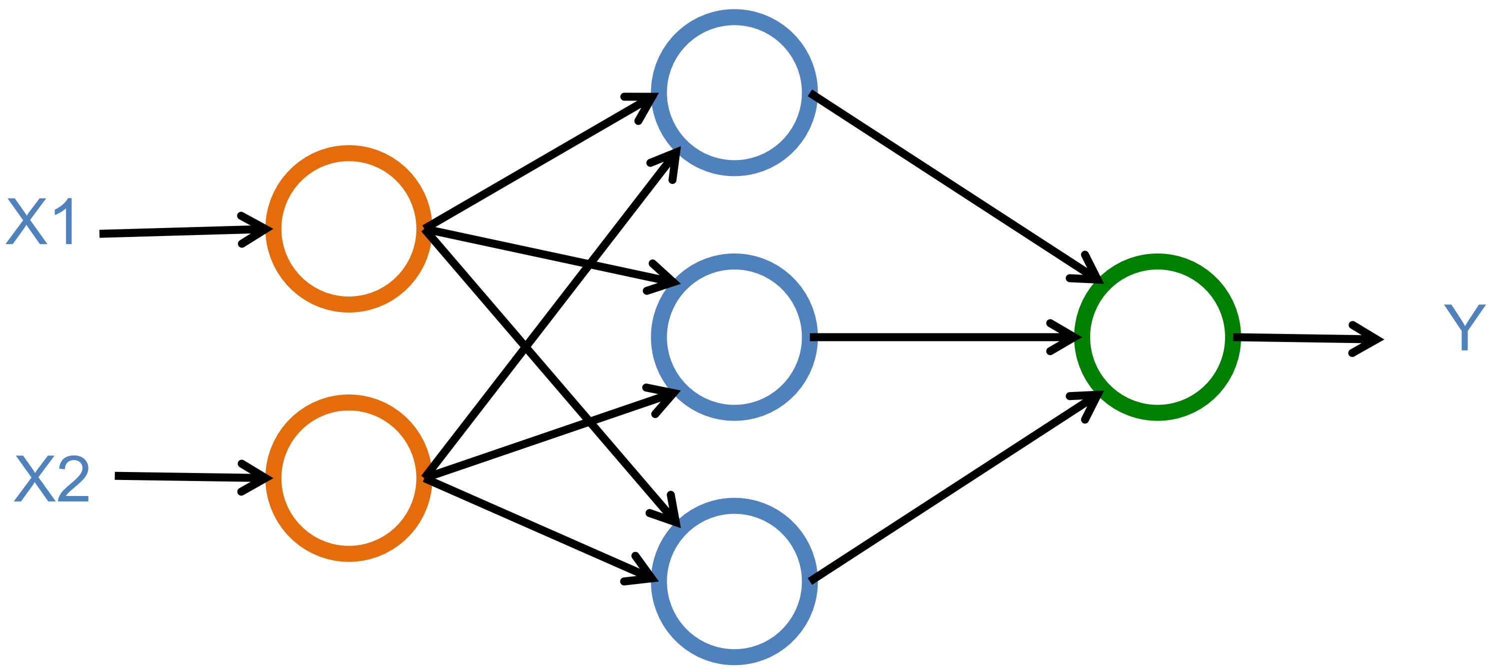
Source: http://www.erogol.com/brief-history-machine-learning/

Deep Learning
and
Neural Networks

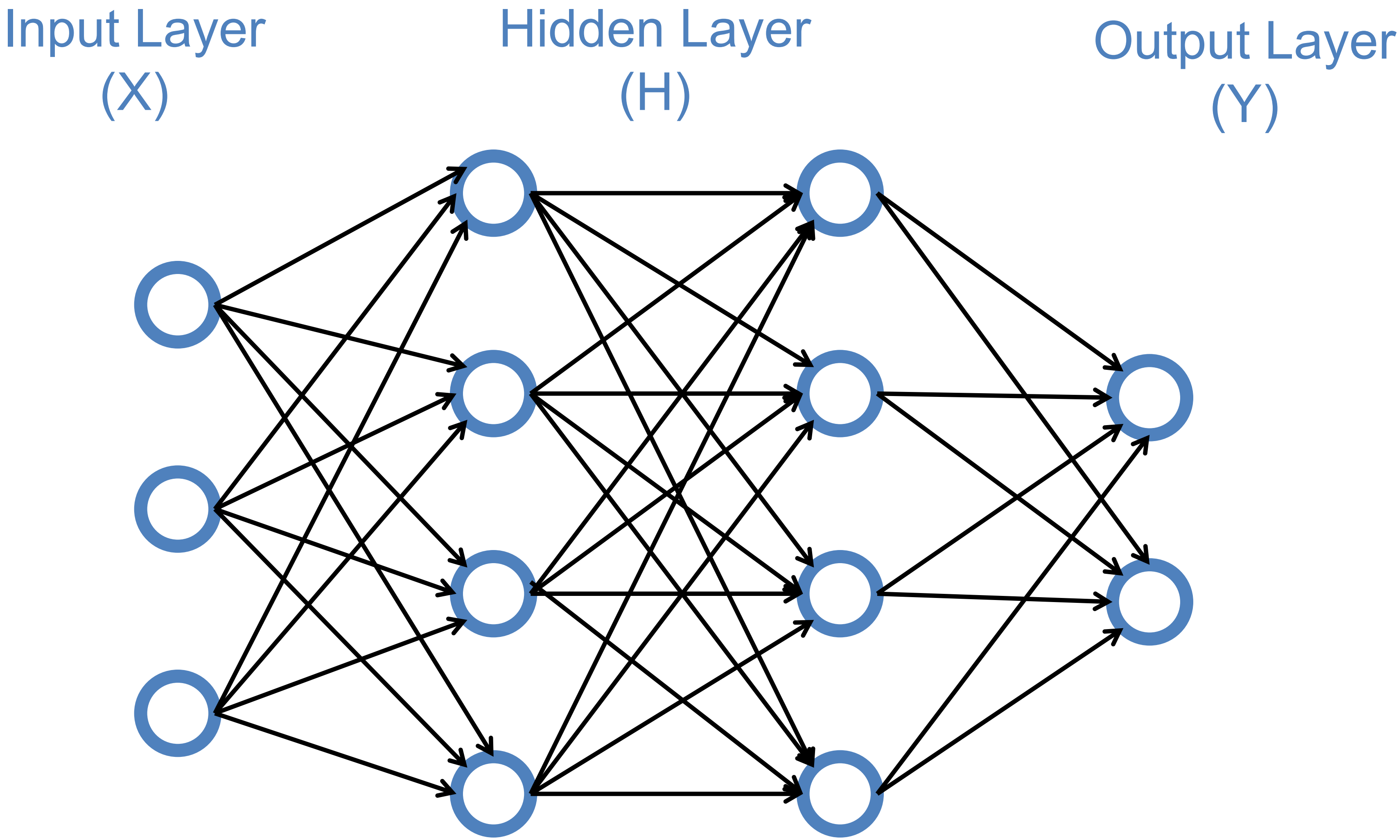
Deep Learning Foundations: Neural Networks

Deep Learning and Neural Networks

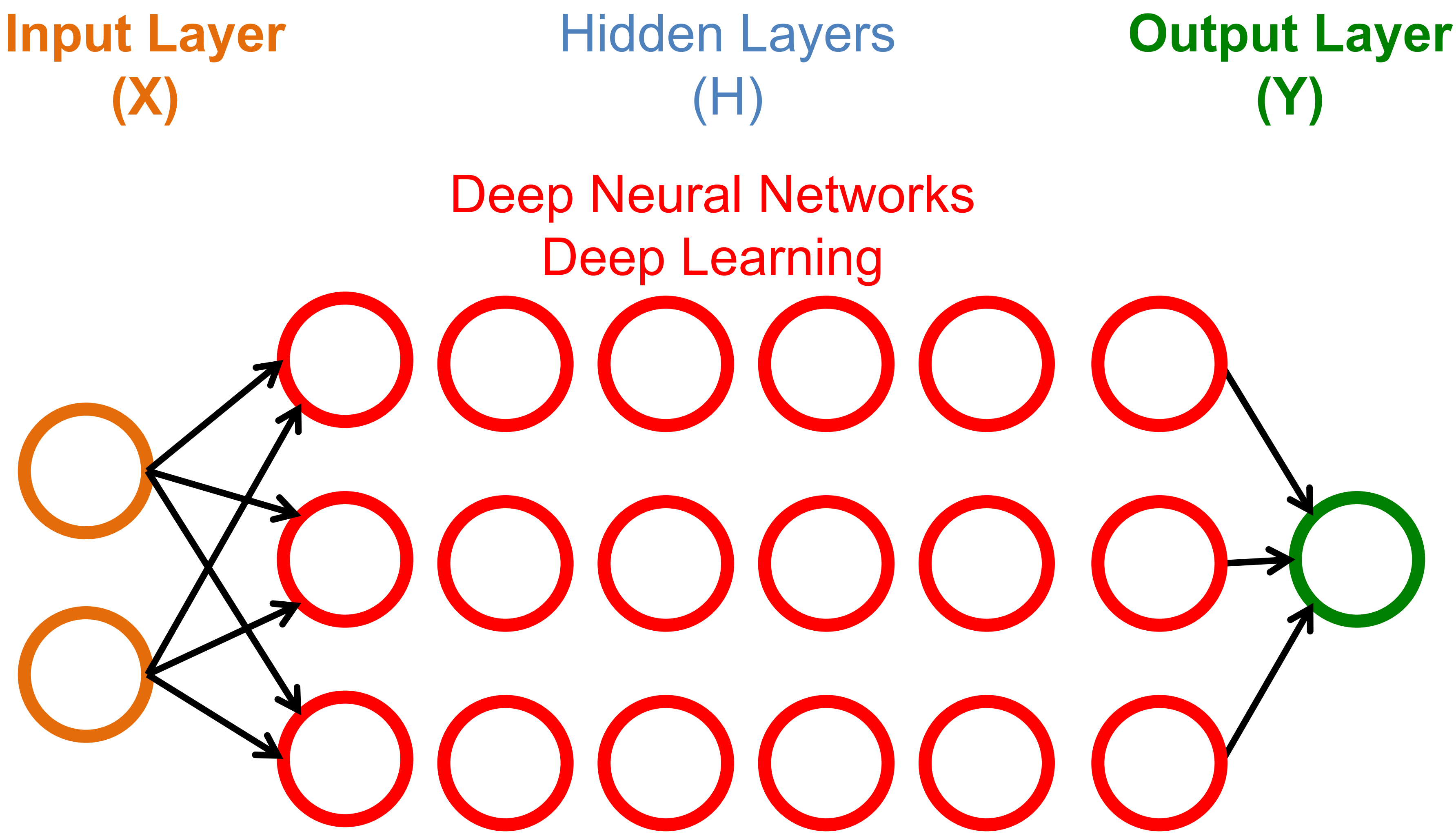
Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



Deep Learning and Neural Networks



Deep Learning and Neural Networks



Deep Learning
and
Deep Neural Networks

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

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

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

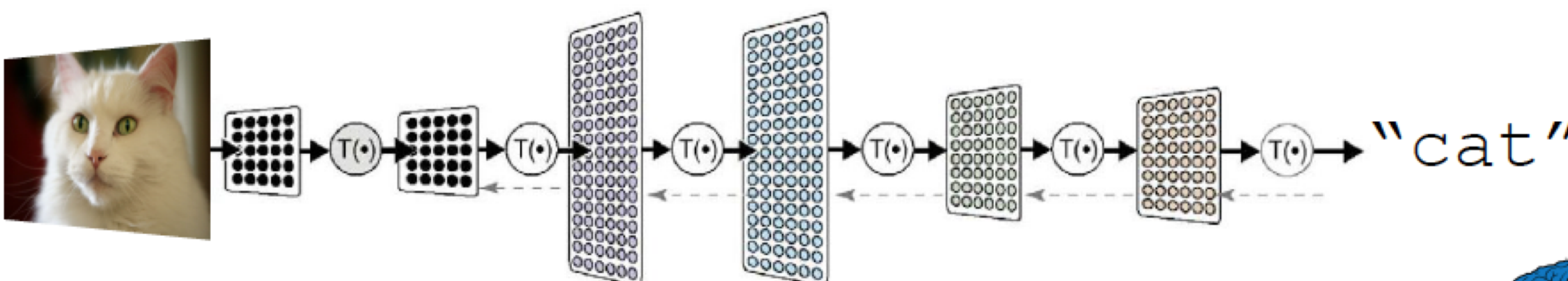
Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other 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, con- intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition[1–4] and speech recognition[5–7], it has beaten other machine-learning techniques at predicting the activity of potential drug 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].

Deep Learning

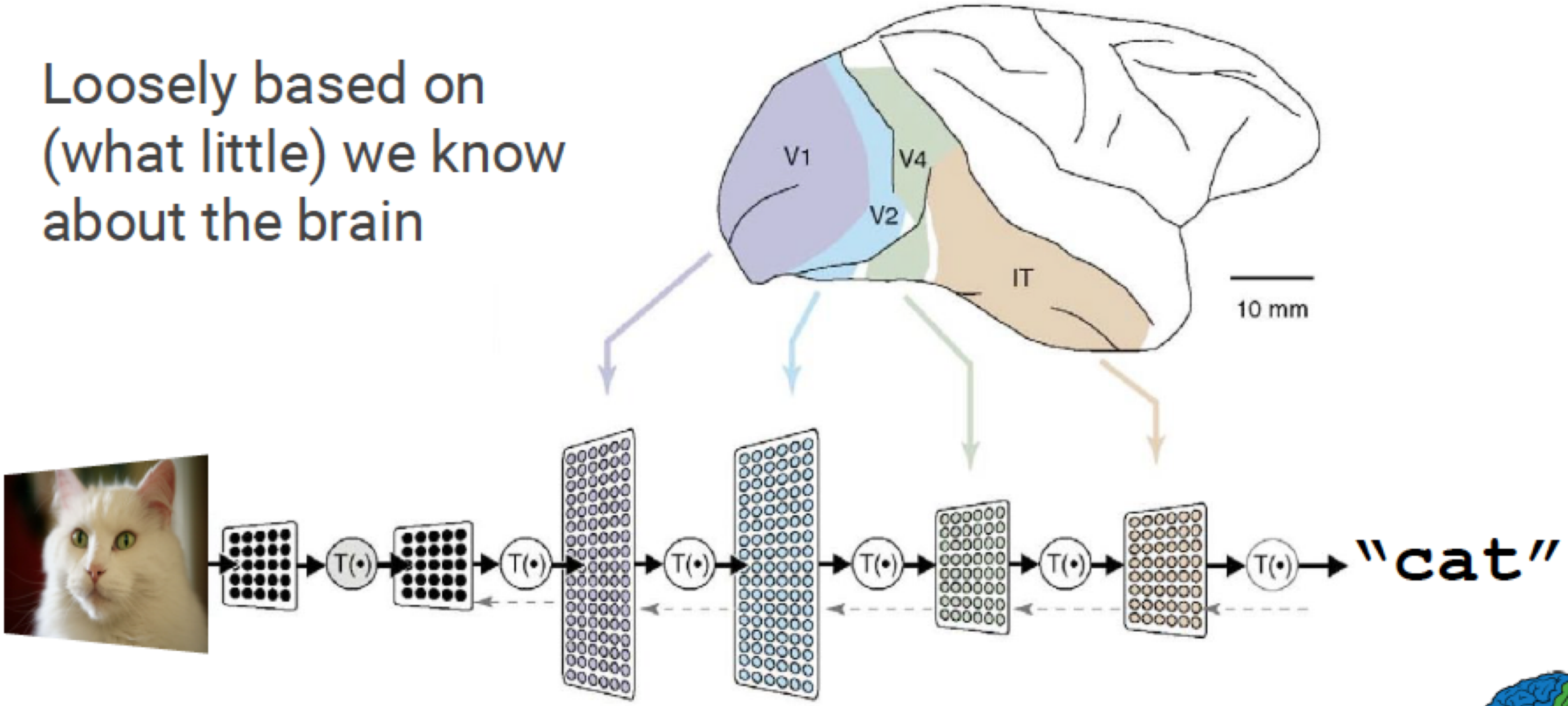
- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple,
  trainable mathematical functions
- Compatible with many variants of machine learning



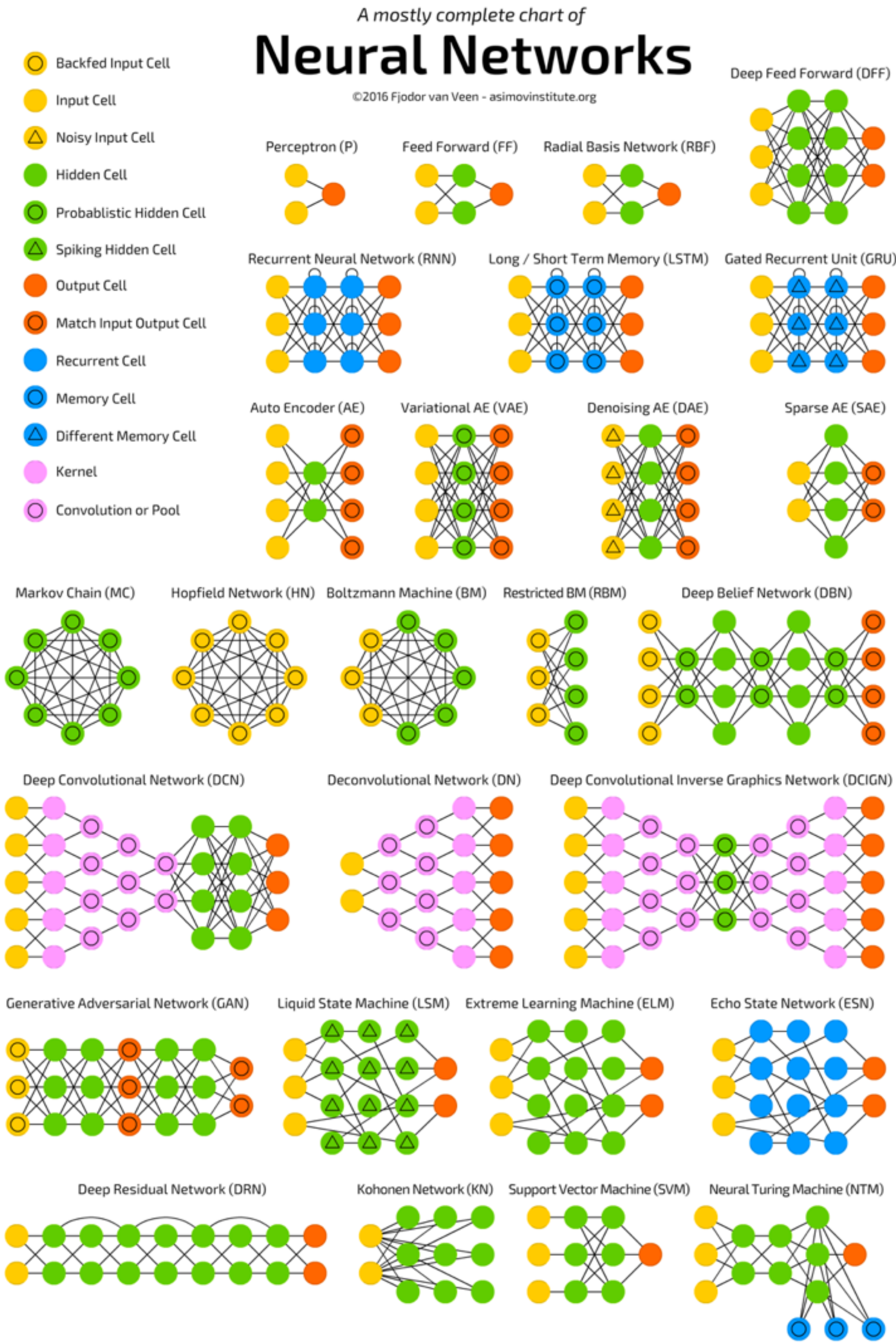
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

What is Deep Learning?

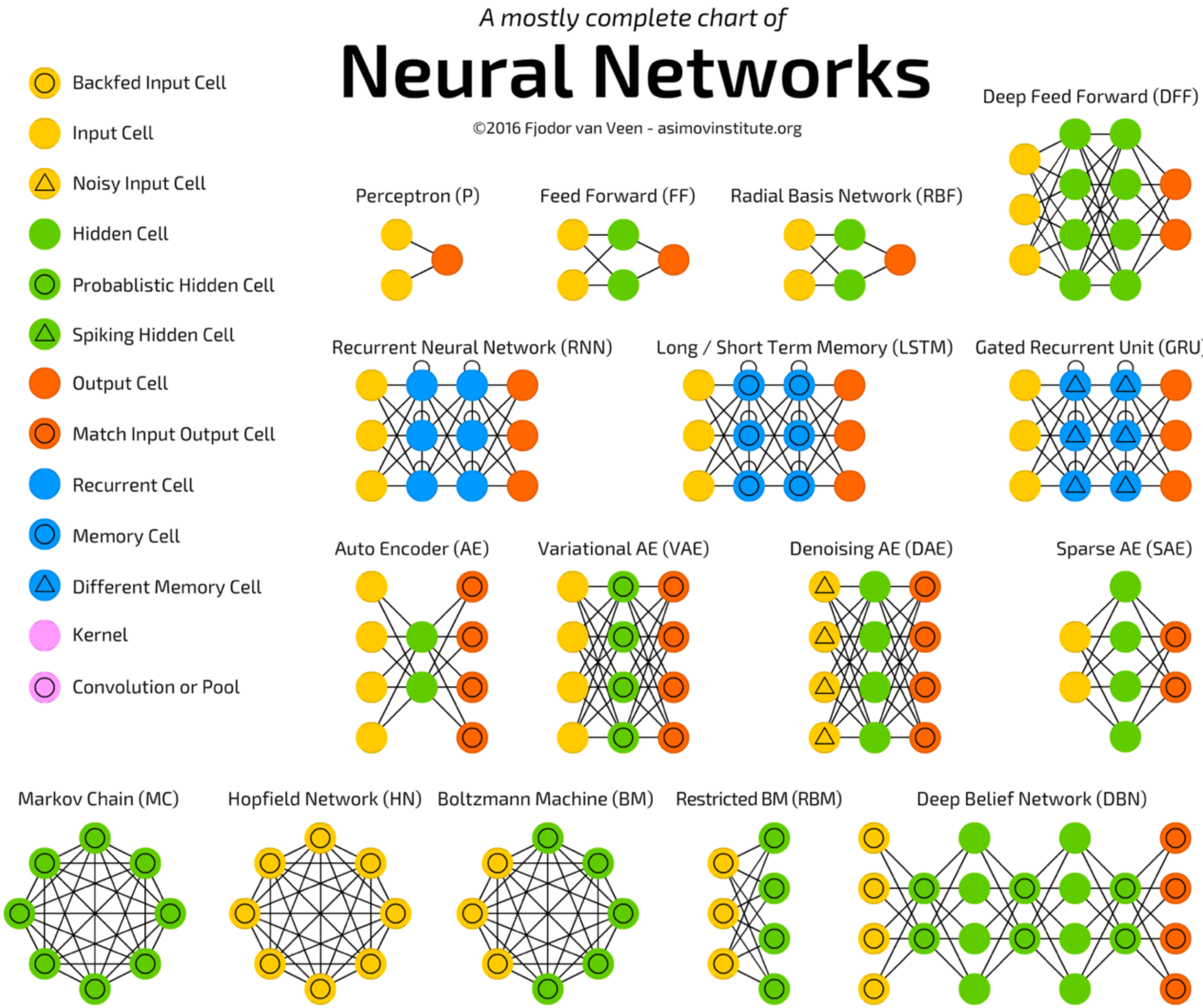
- Loosely based on (what little) we know about the brain



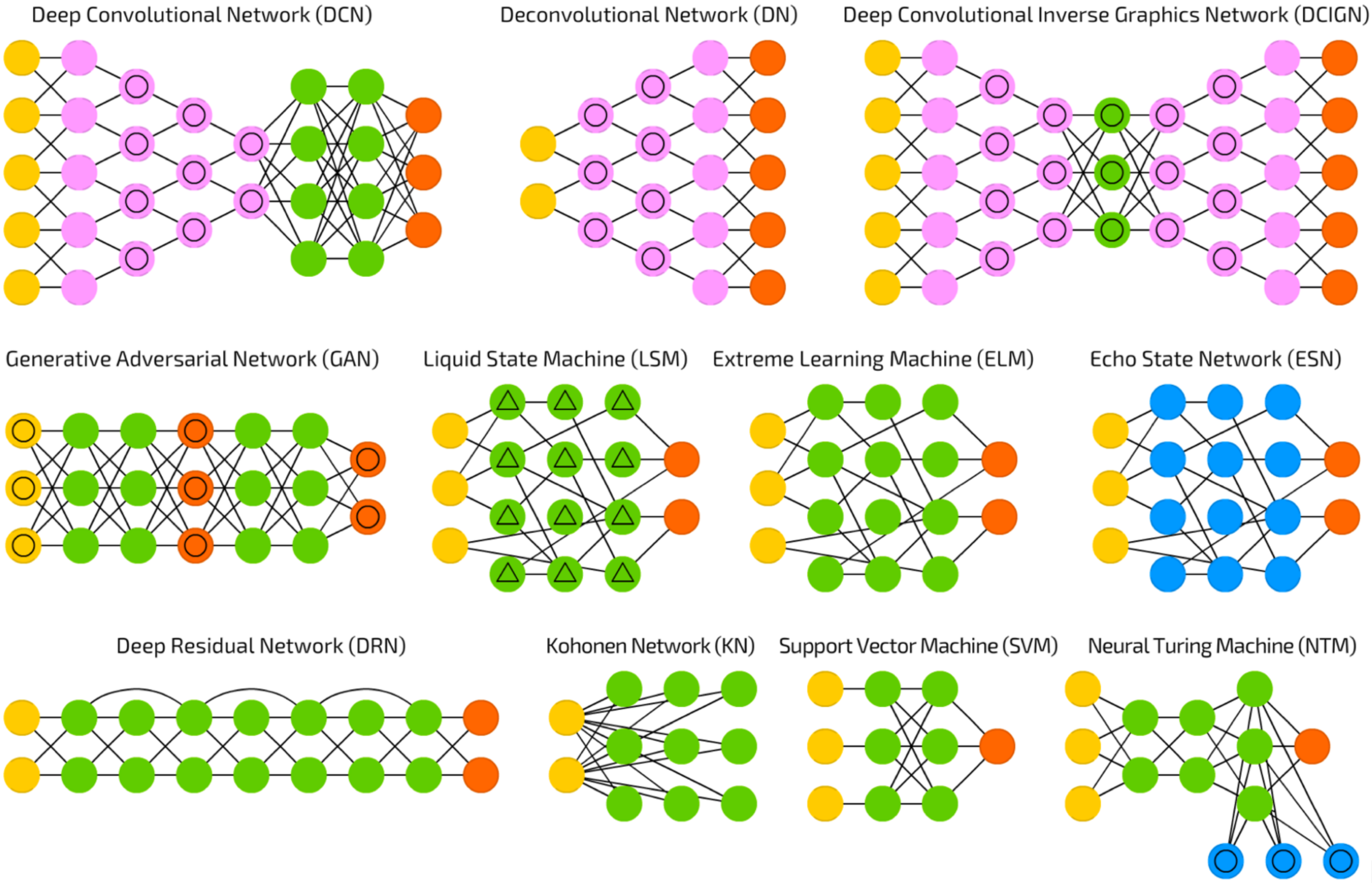
Neural Networks (NN)



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

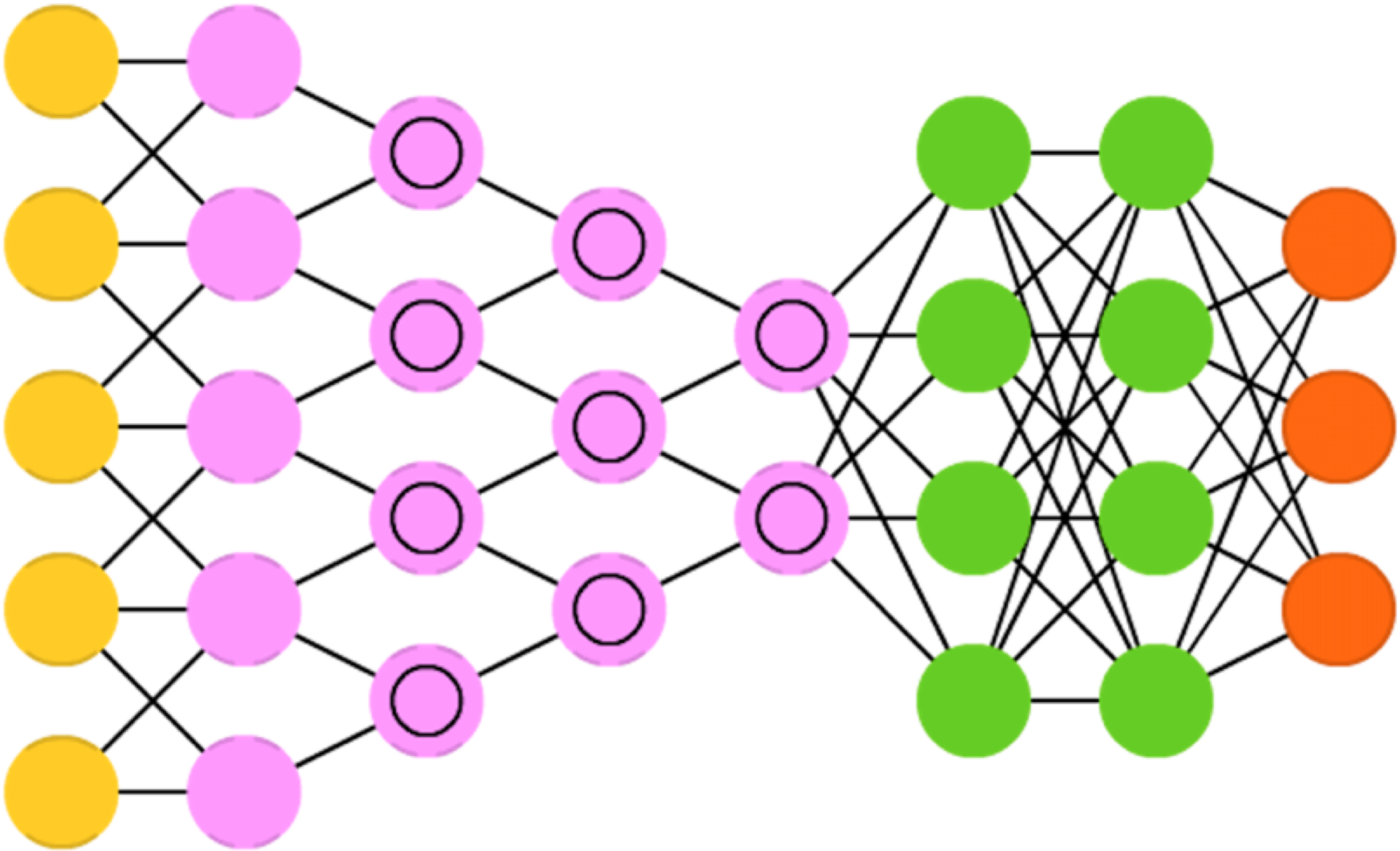


Source: http://www.asimovinstitute.org/neural-network-zoo/



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

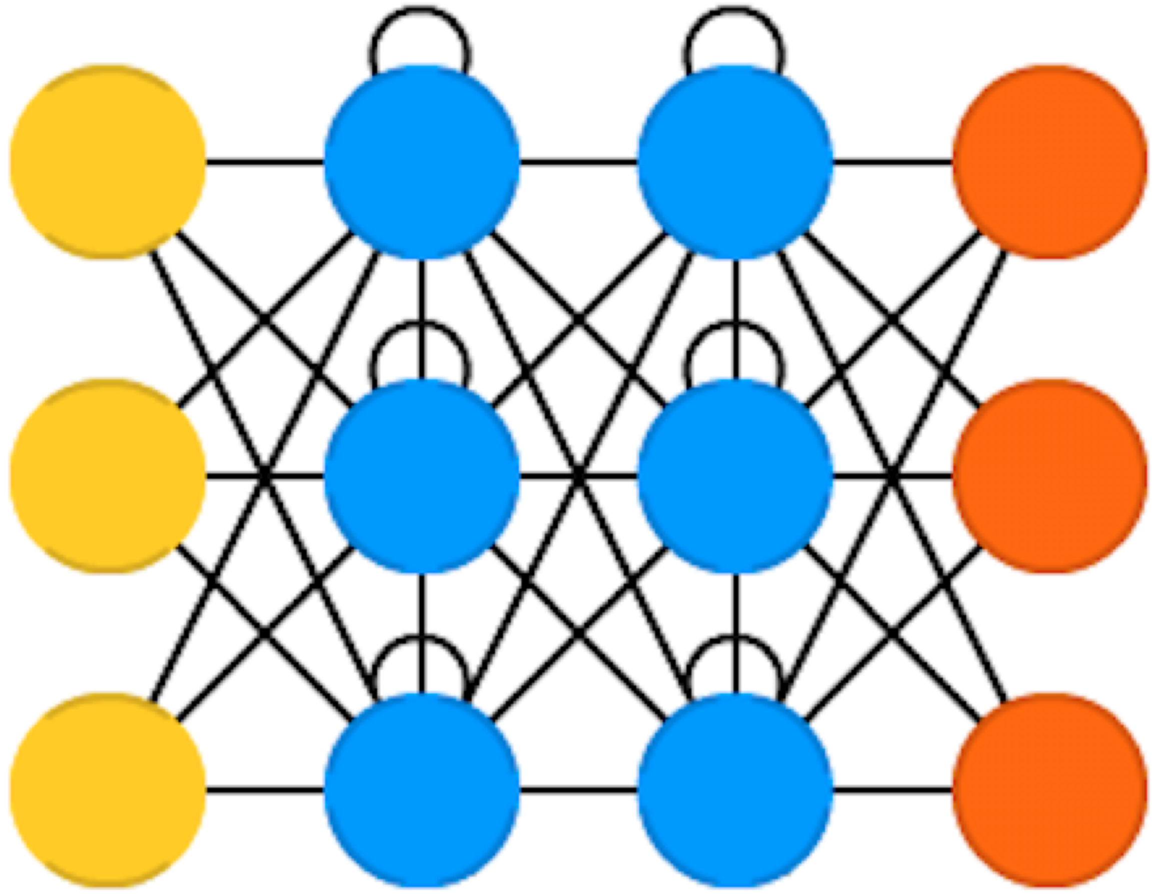
Convolutional Neural Networks
(CNN or Deep Convolutional Neural Networks, DCNN)



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

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

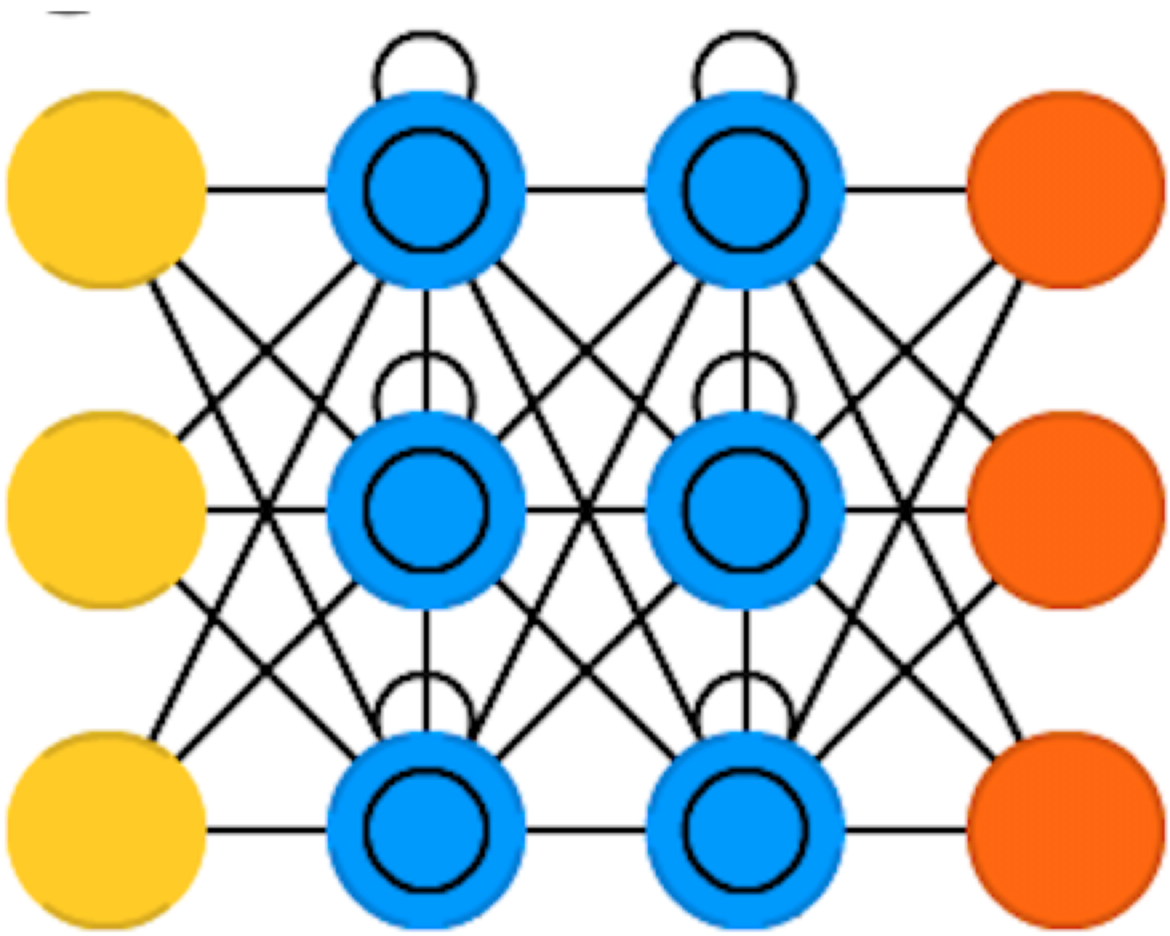
Recurrent Neural Networks (RNN)



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

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

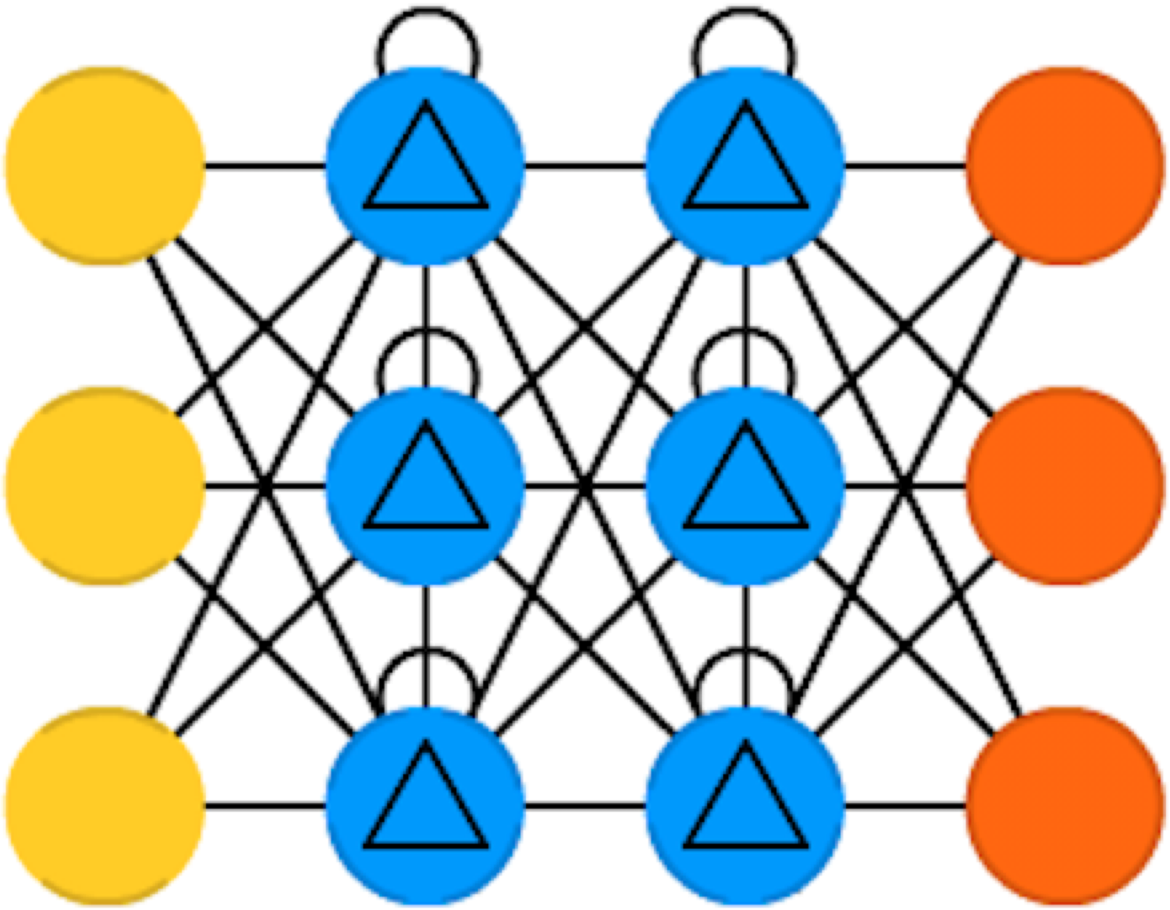
Long / Short Term Memory (LSTM)



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

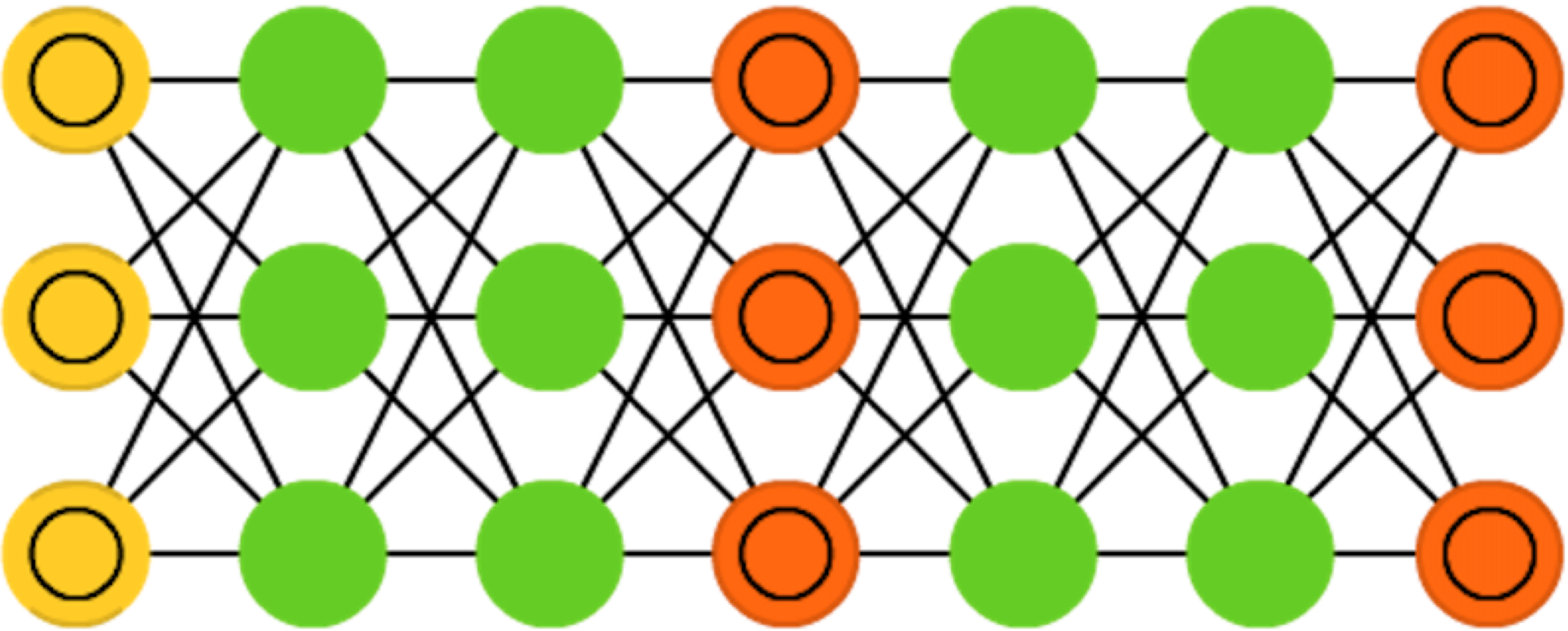
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



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

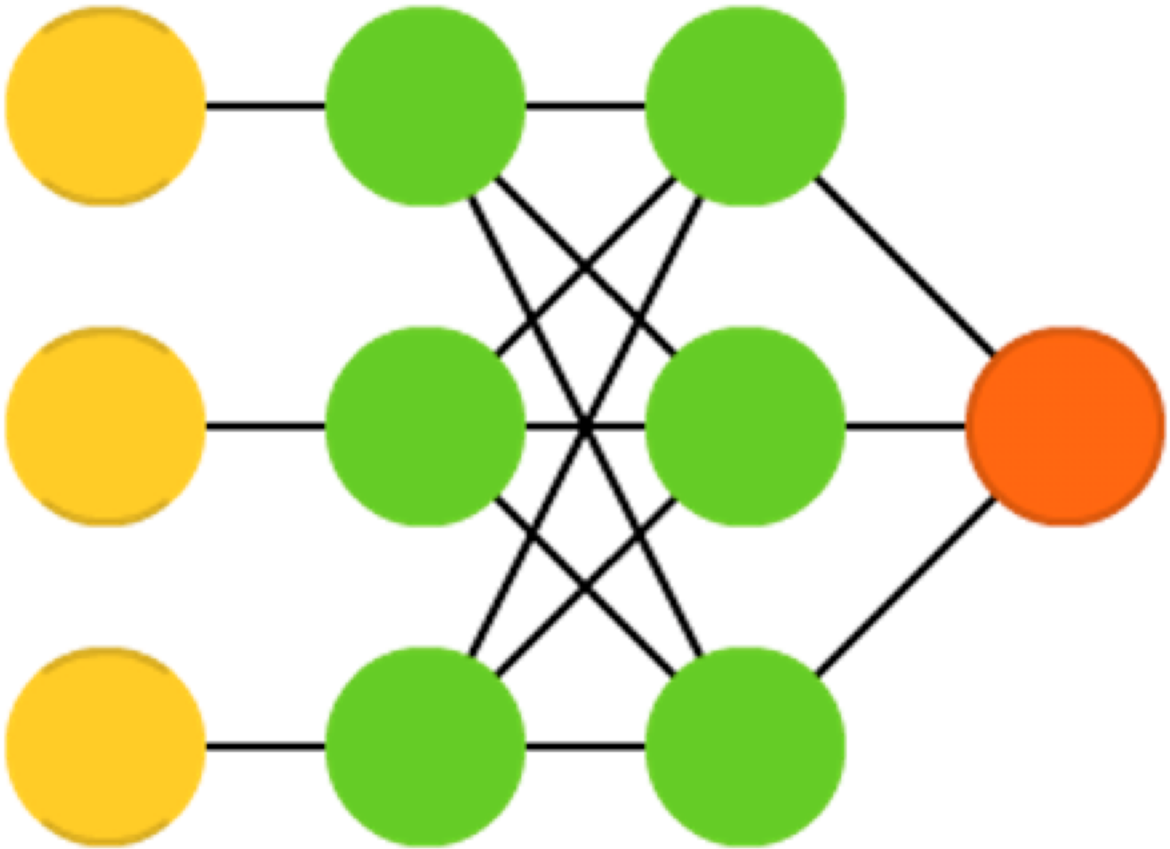
Generative Adversarial Networks (GAN)



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

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

Support Vector Machines (SVM)

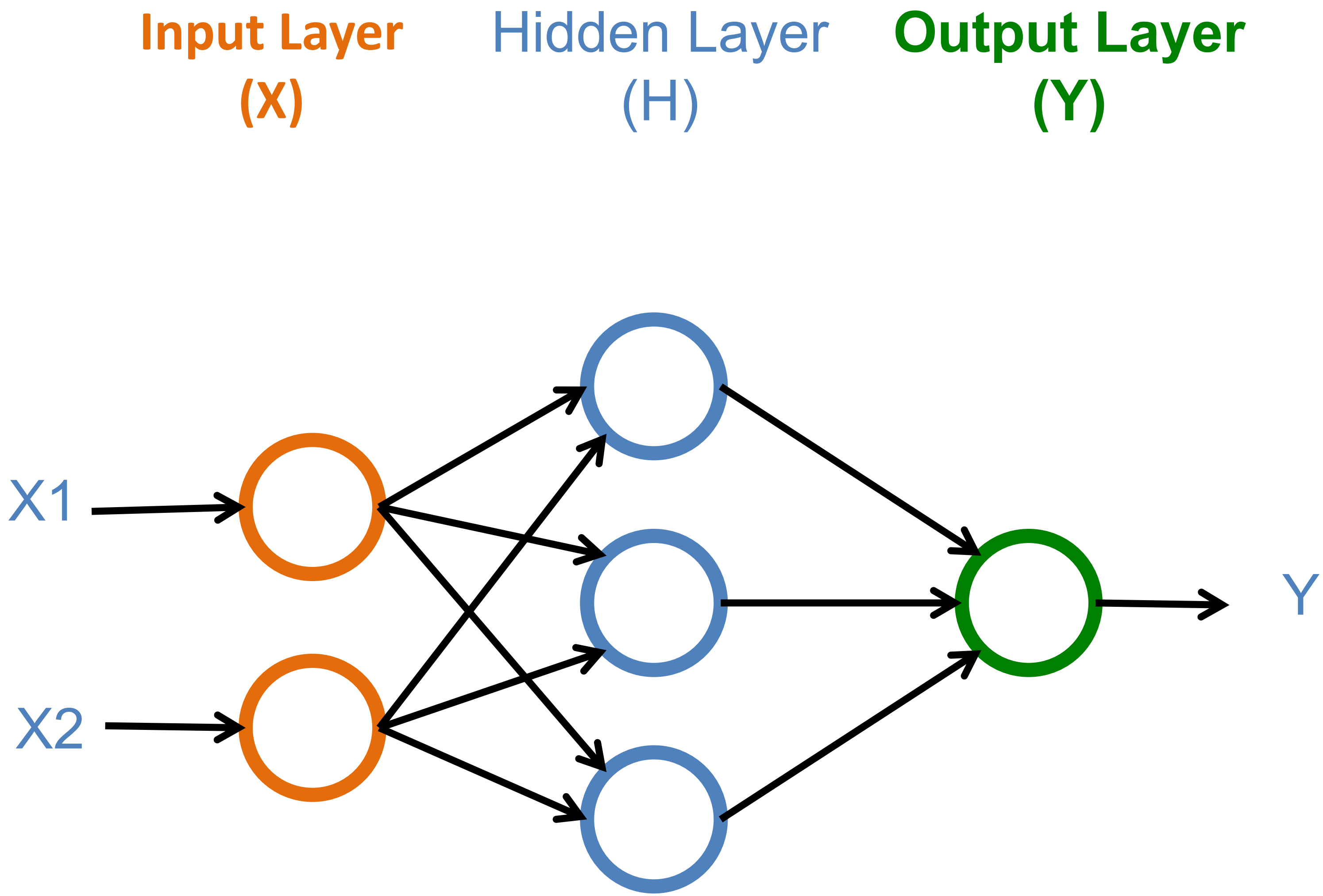


Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

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

Neural networks (NN) 1960

Neural Networks



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

Multilayer Perceptrons (MLP) 1985

Support Vector Machine (SVM) 1995

Hinton presents the

Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST

2005

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

Max-Pooling CDBN
2011

Deep Learning

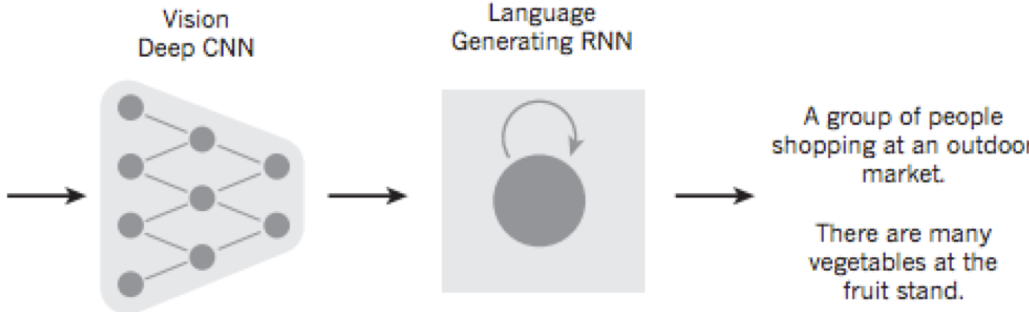
Geoffrey Hinton

Yann LeCun

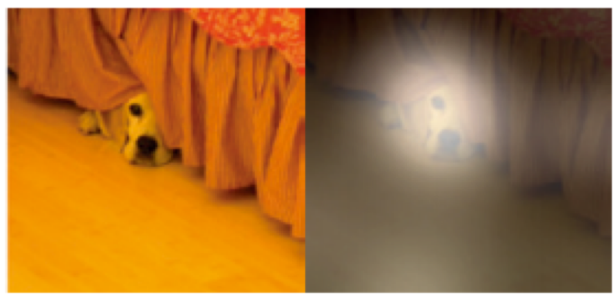
Yoshua Bengio

Andrew Y. Ng

From image to text

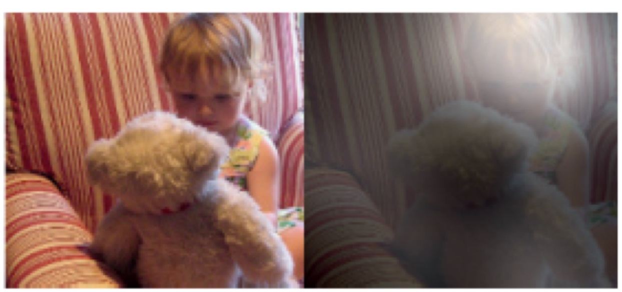


A woman is throwing a **frisbee** in a park.

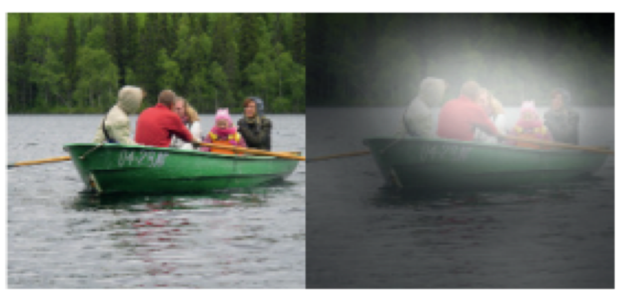


A **dog** is standing on a hardwood floor.

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



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



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

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

From image to text

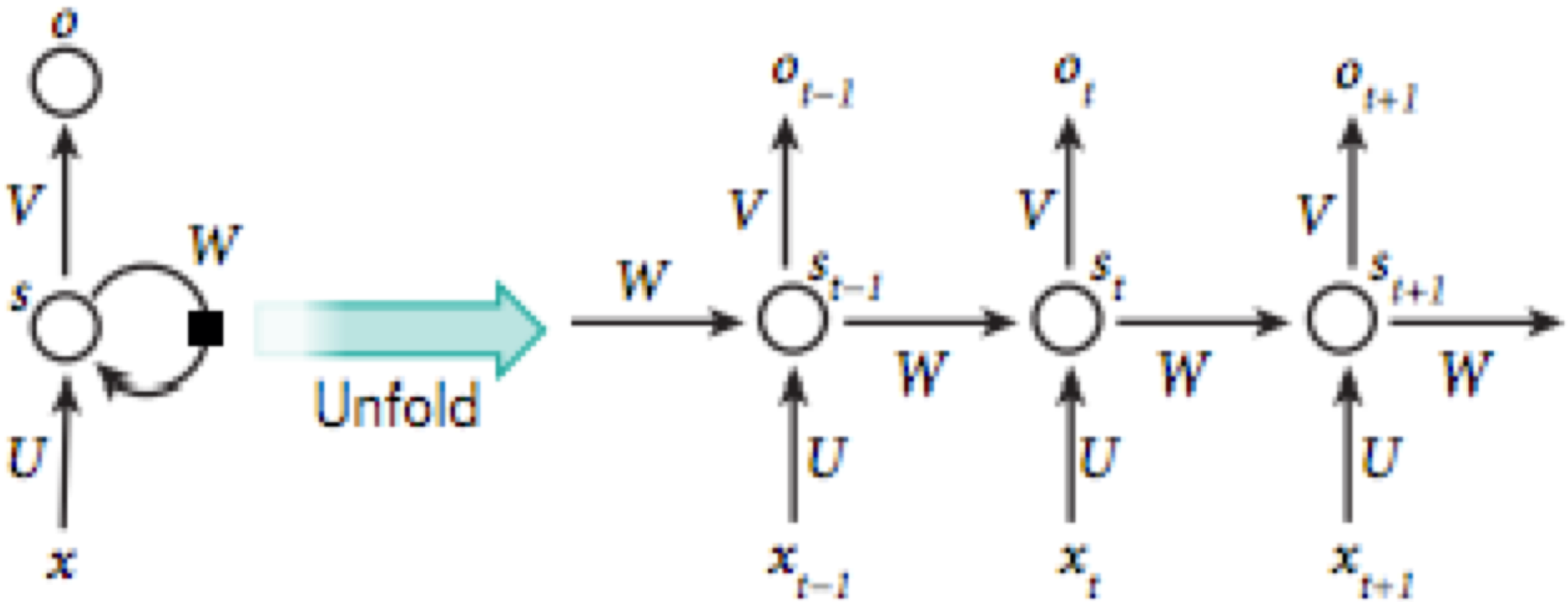
Image: deep convolution neural network (CNN)

Text: recurrent neural network (RNN)

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

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

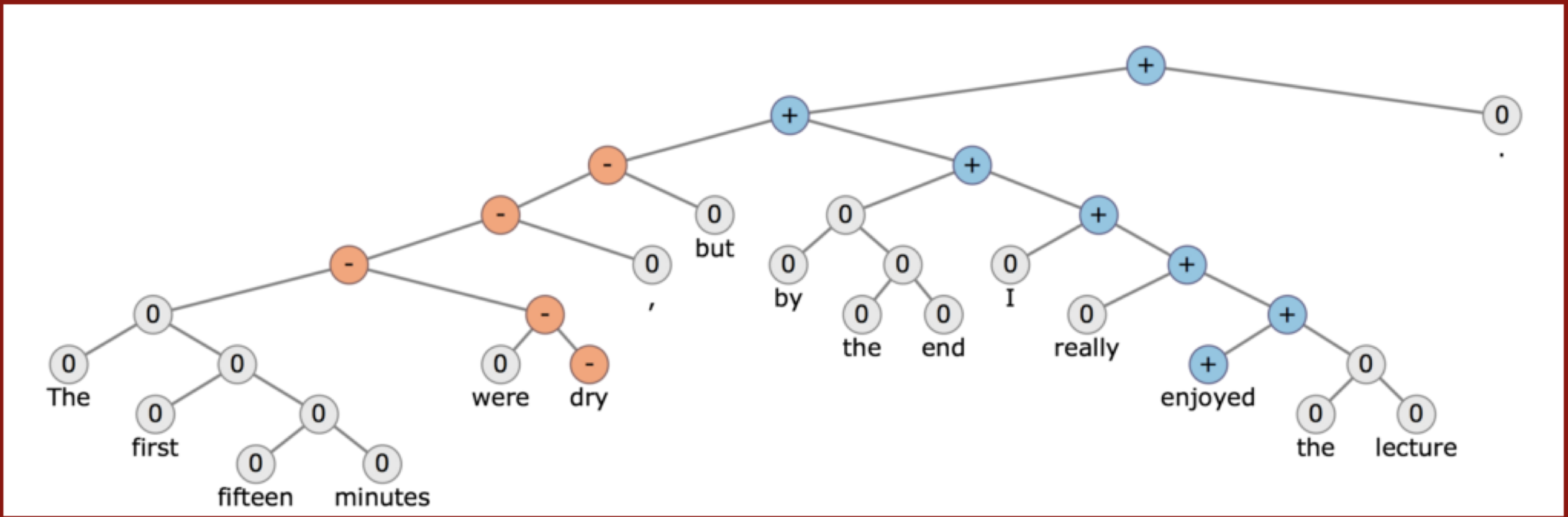
Recurrent Neural Network (RNN)



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

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

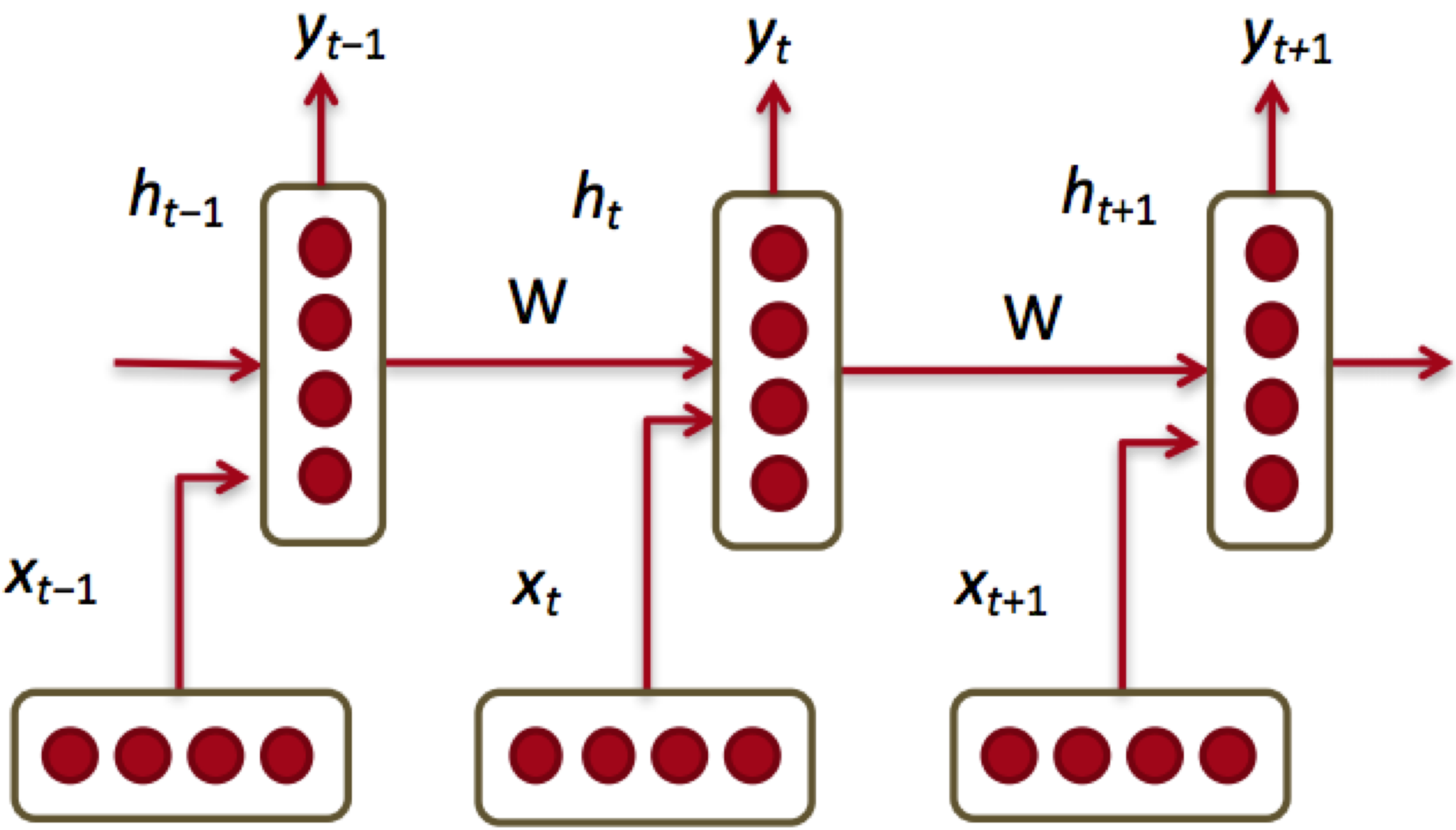


Course Description

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

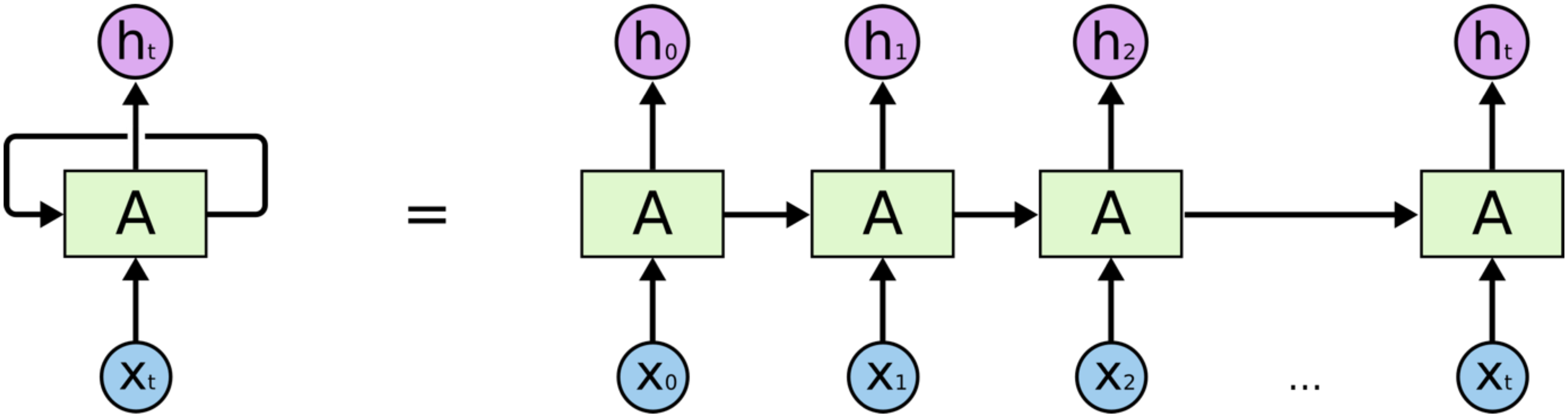
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



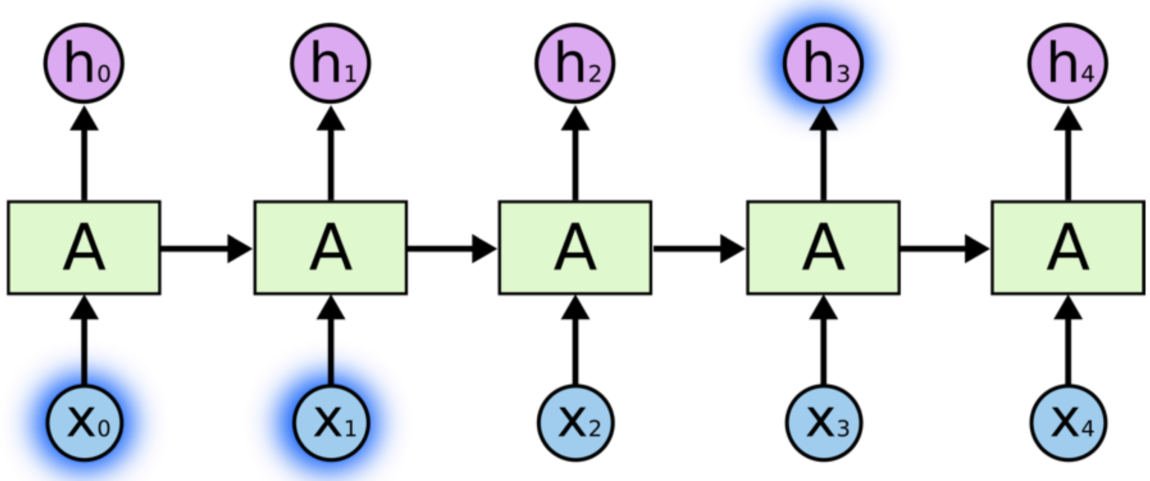
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

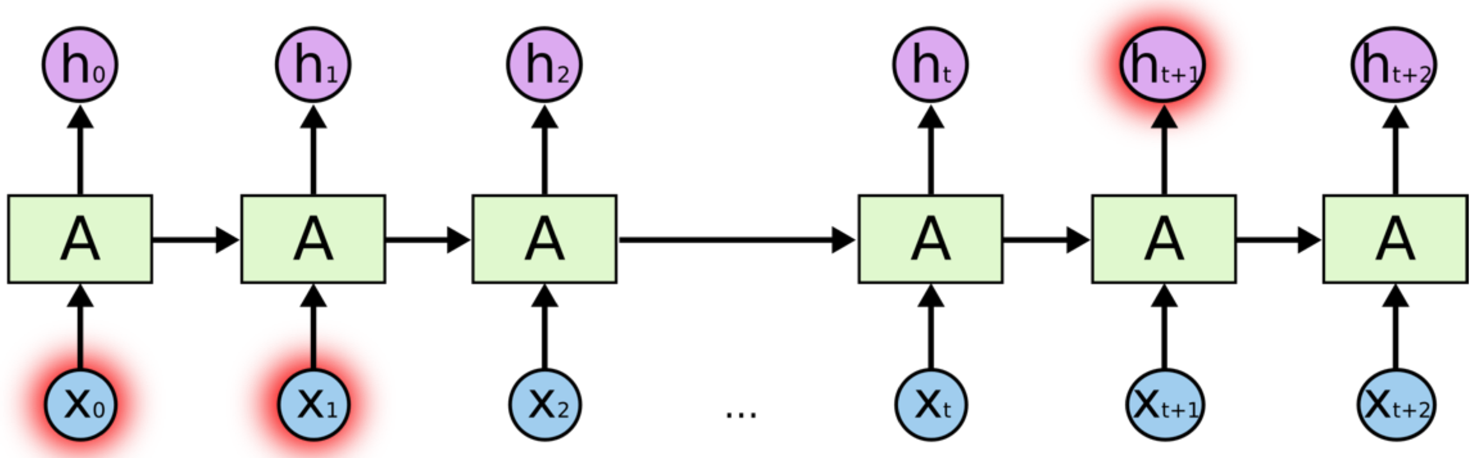
RNN



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

RNN long-term dependencies

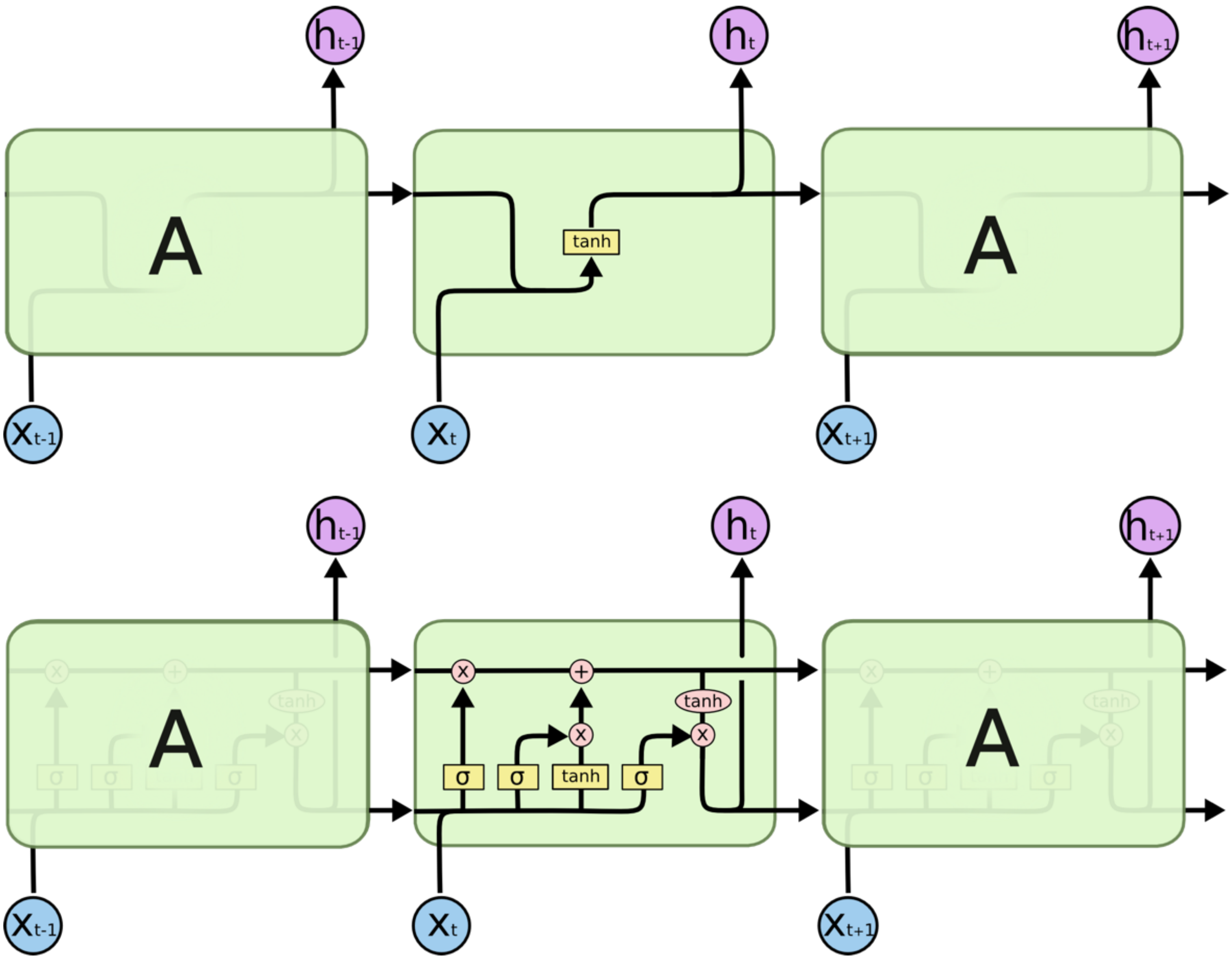




I grew up in France… I speak fluent French.

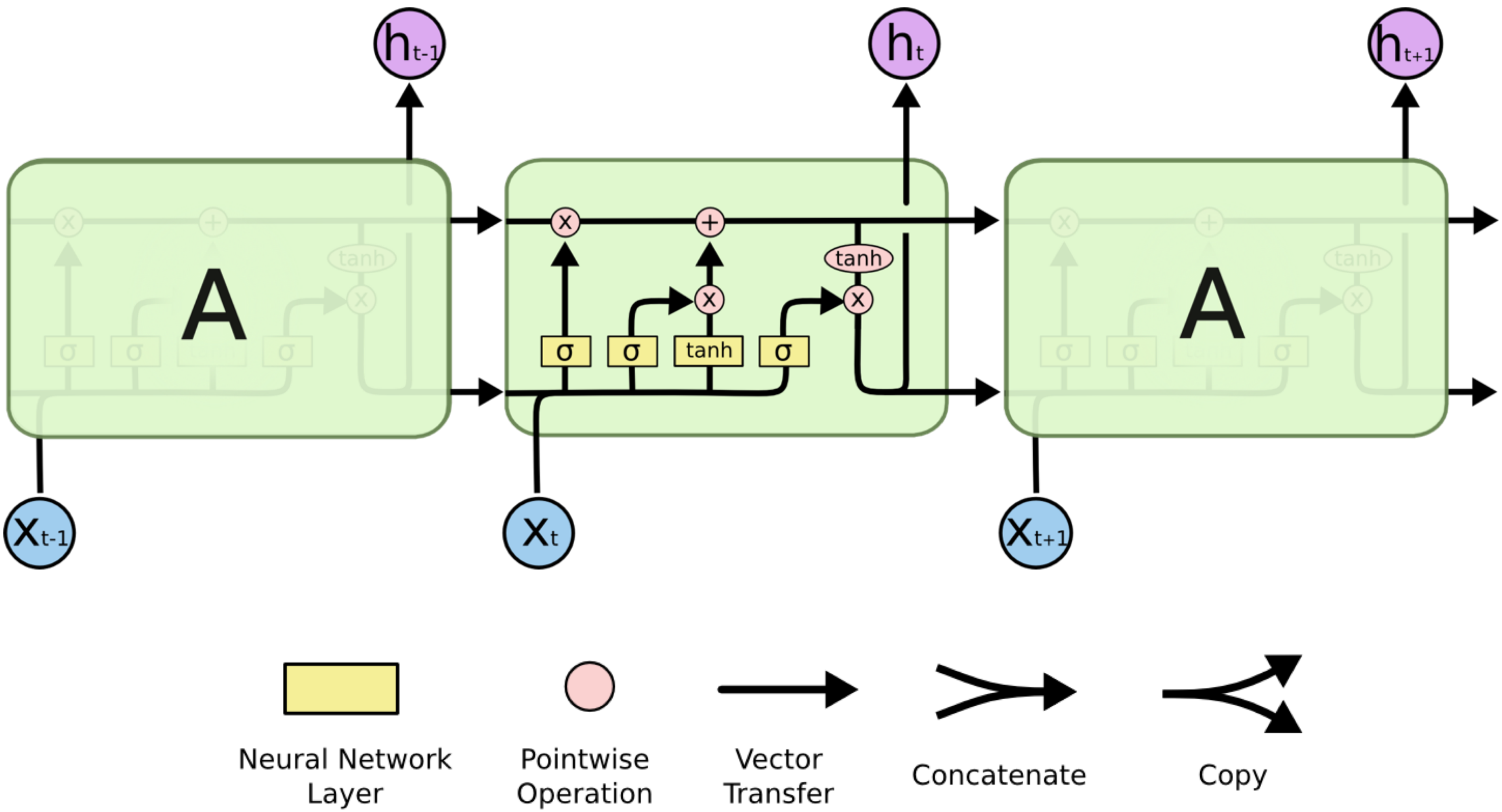
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN LSTM



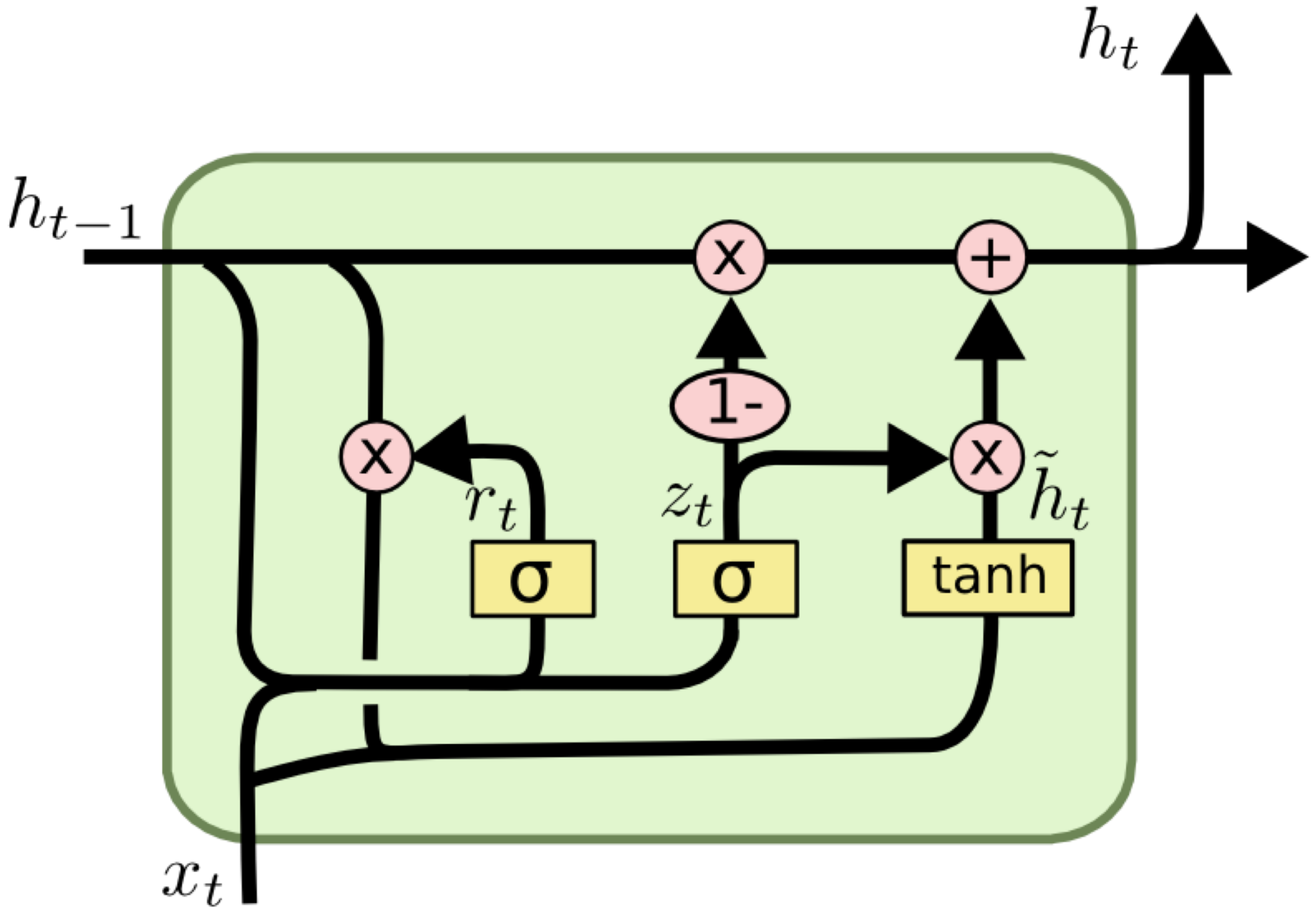
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



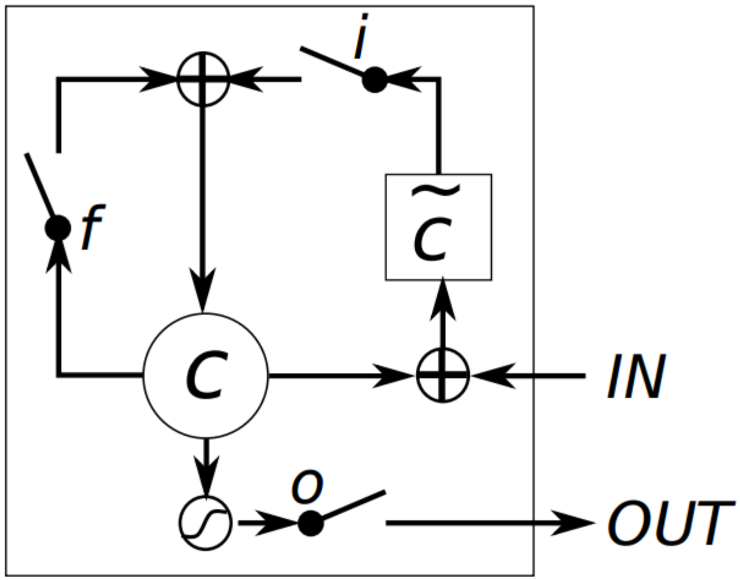
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

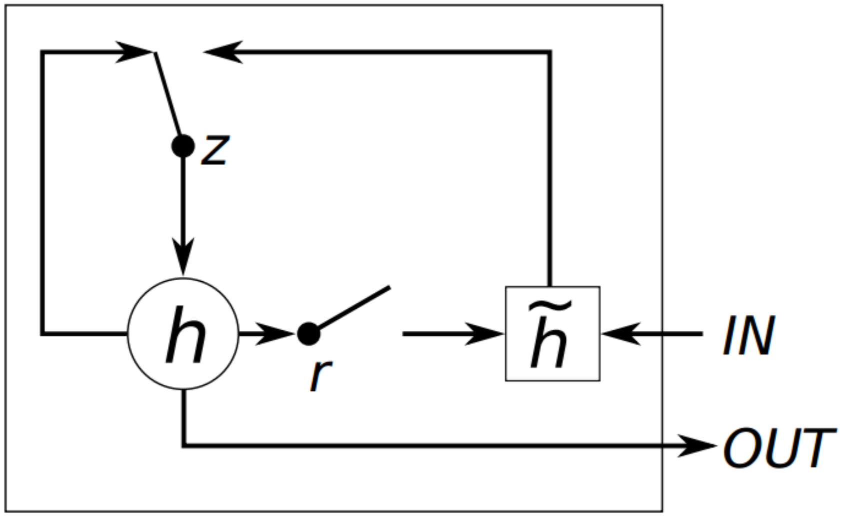
Gated Recurrent Unit (GRU)



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

LSTM vs GRU





LSTM

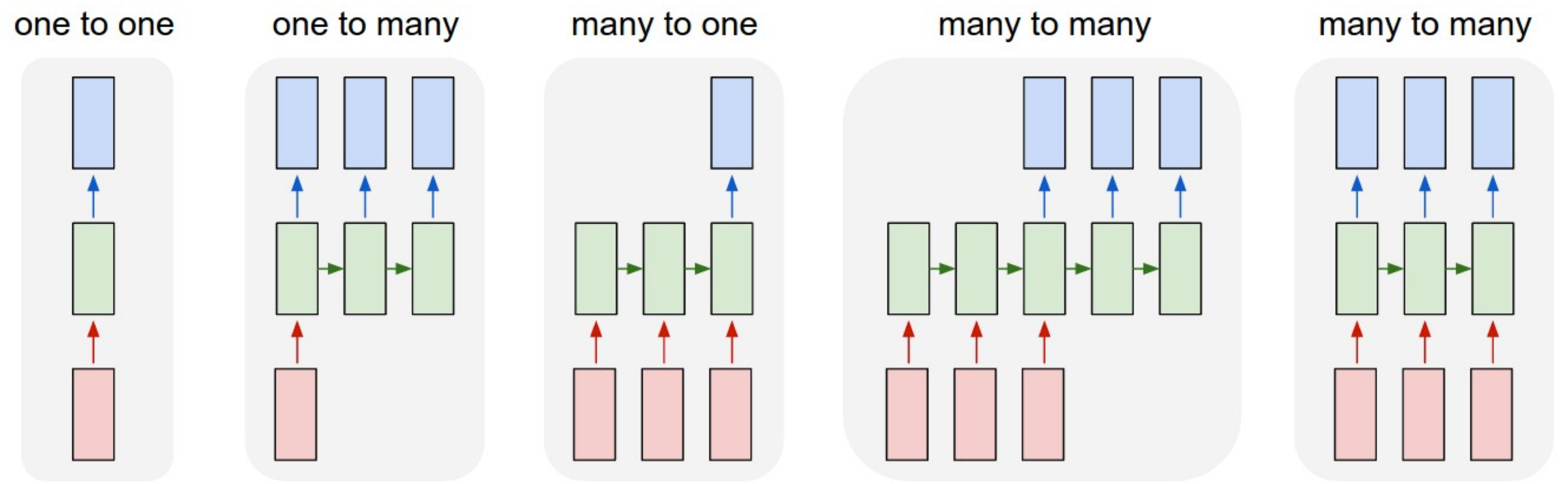
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

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

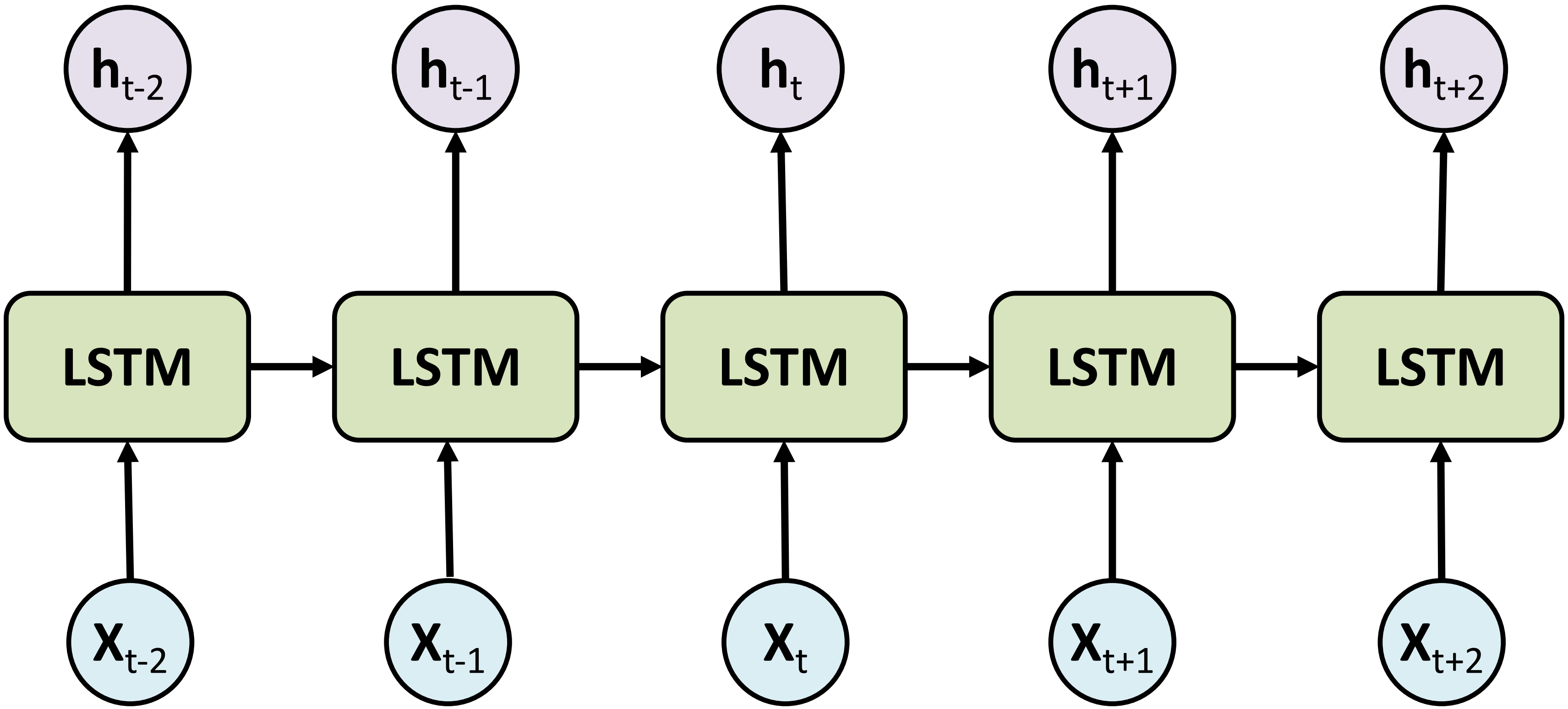
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

LSTM Recurrent Neural Network

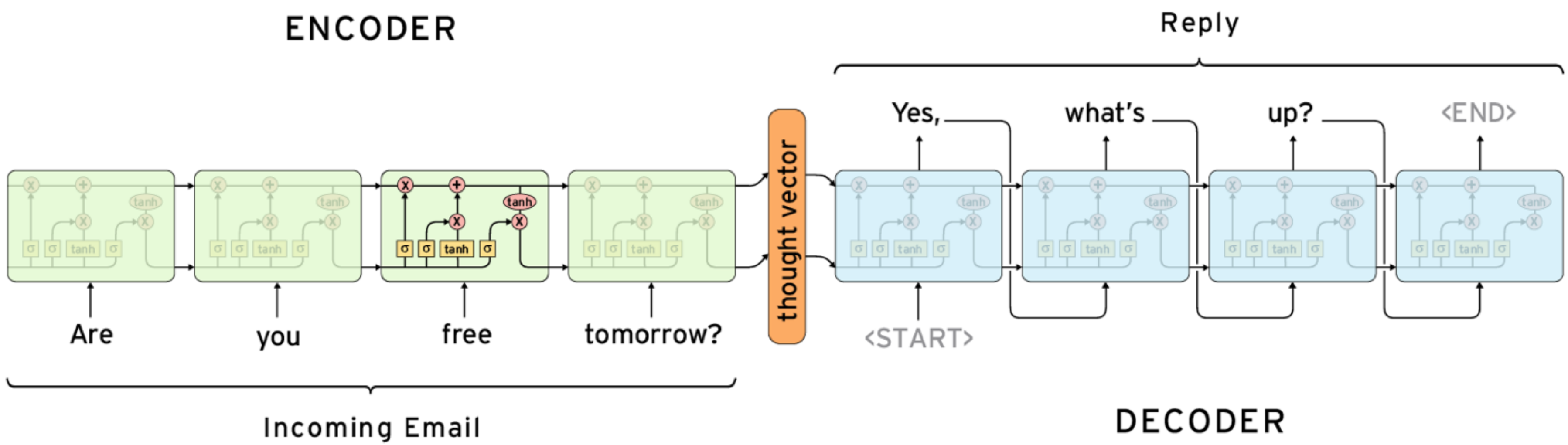


Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

Long Short Term Memory (LSTM) for Time Series Forecasting

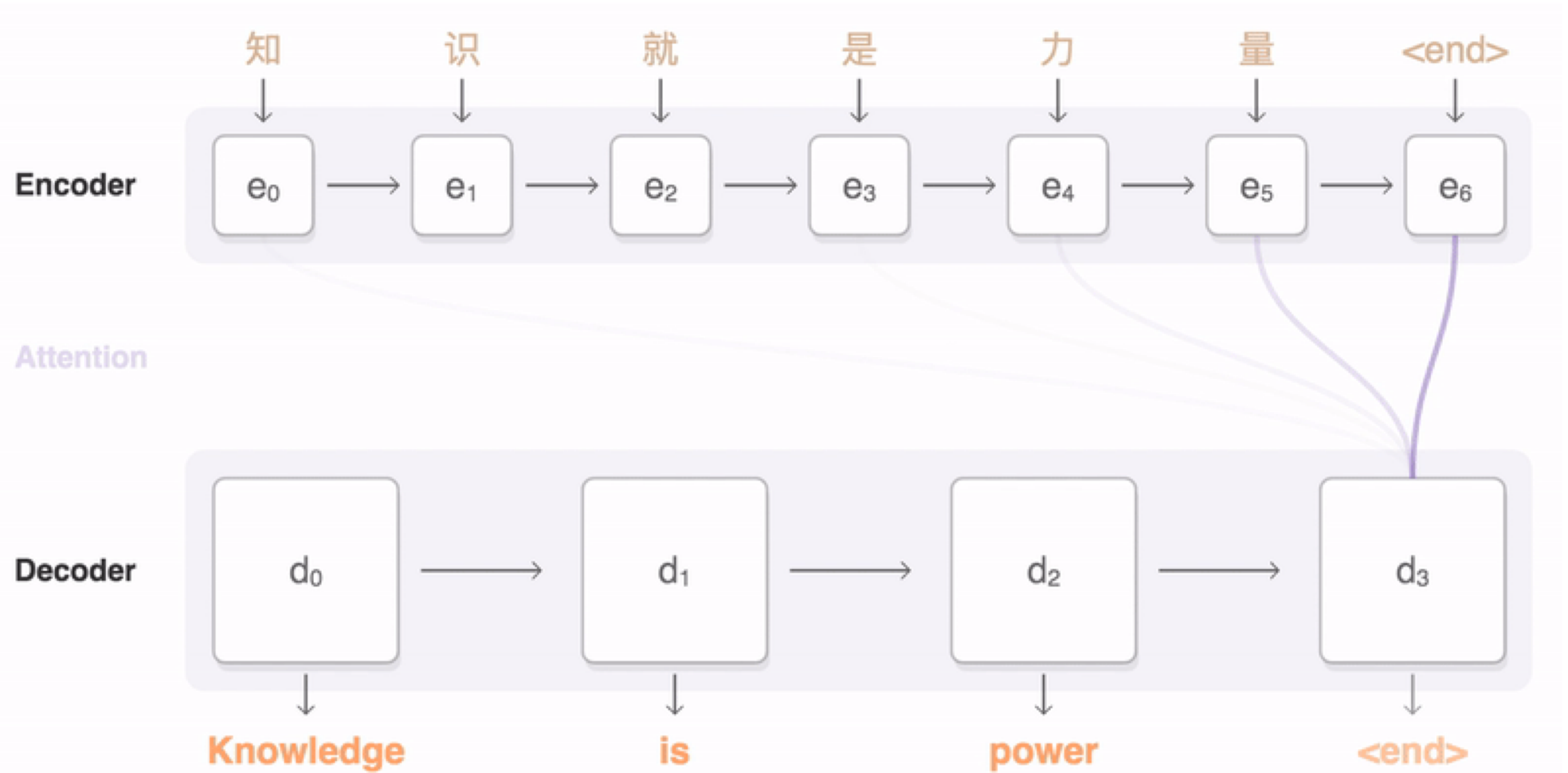


The Sequence to Sequence model (seq2seq)



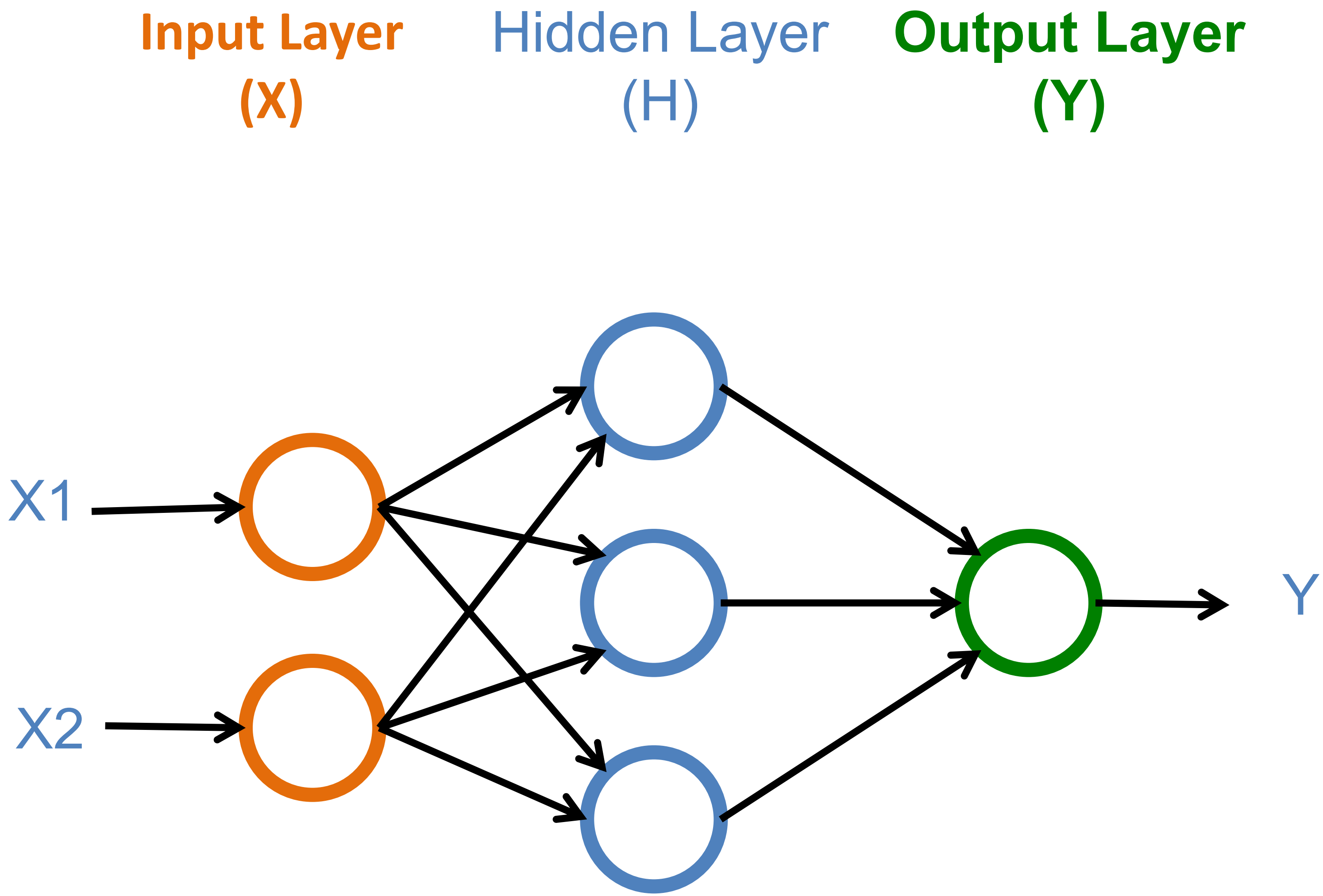
Source: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

Sequence to Sequence (Seq2Seq)



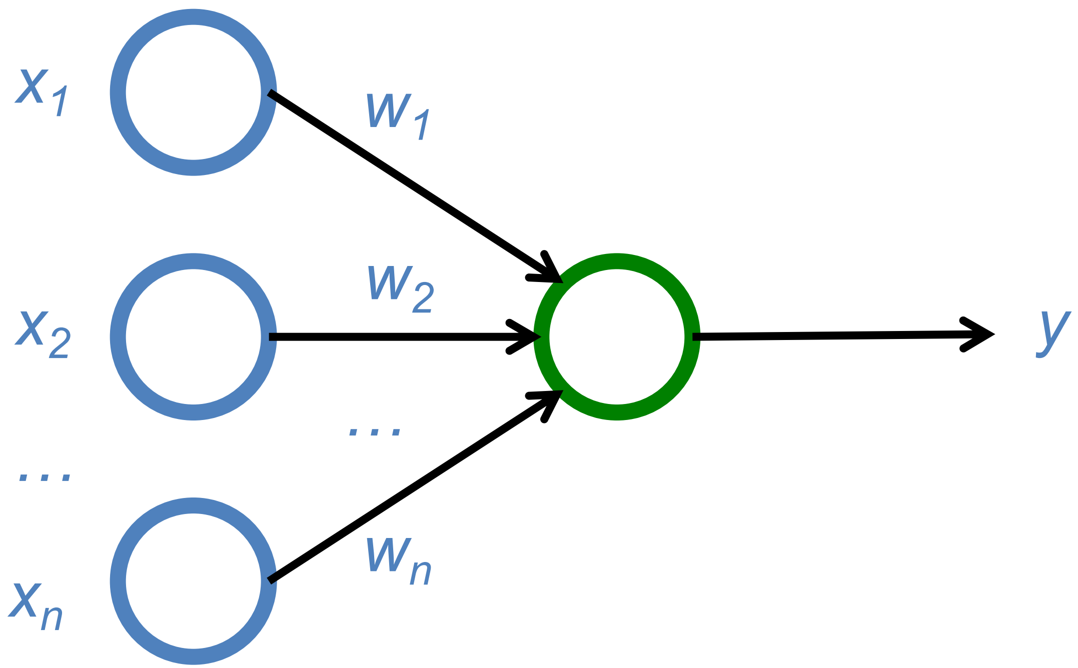
Source: https://google.github.io/seq2seq/

Neural Networks



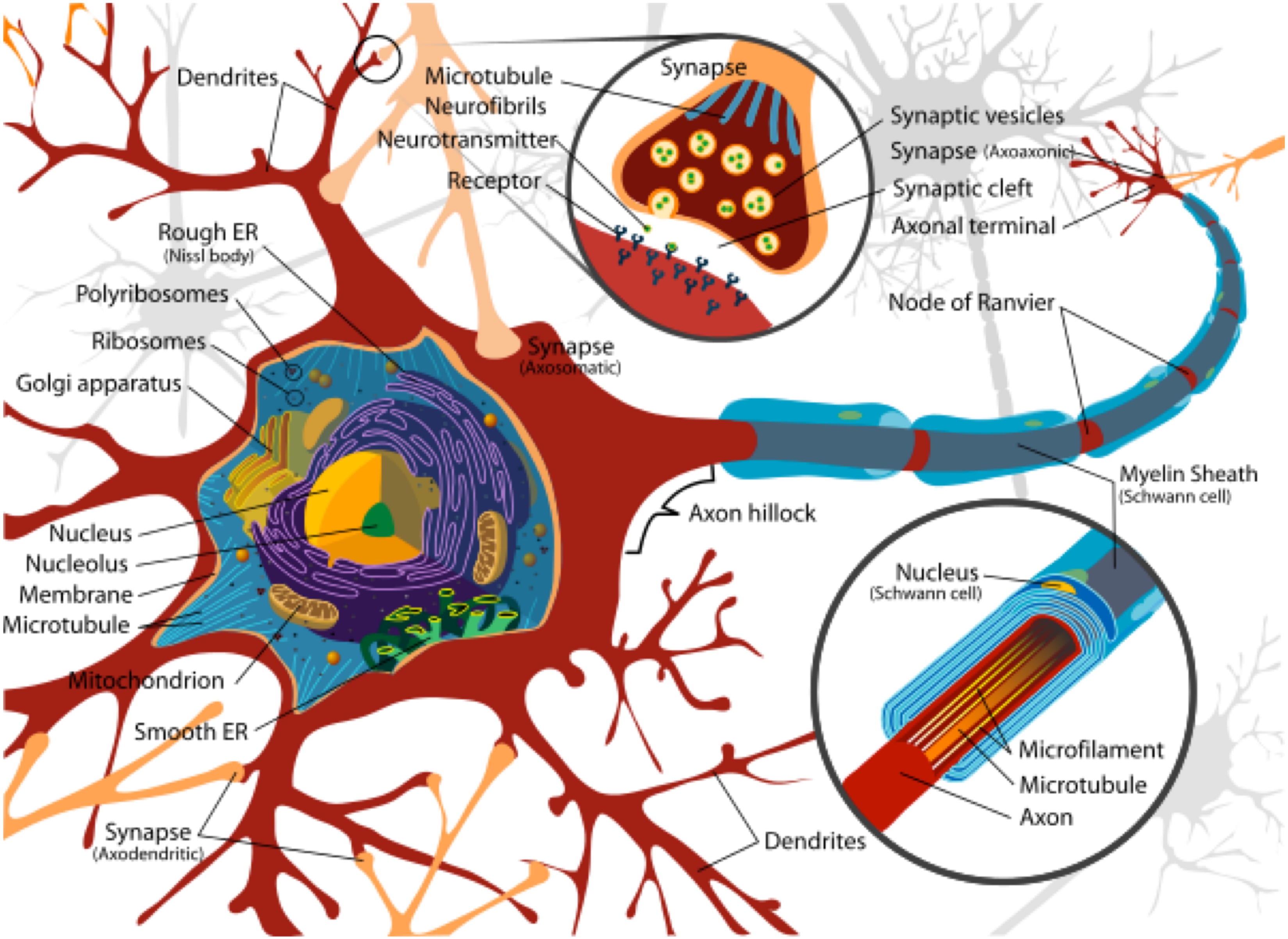
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

The Neuron



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

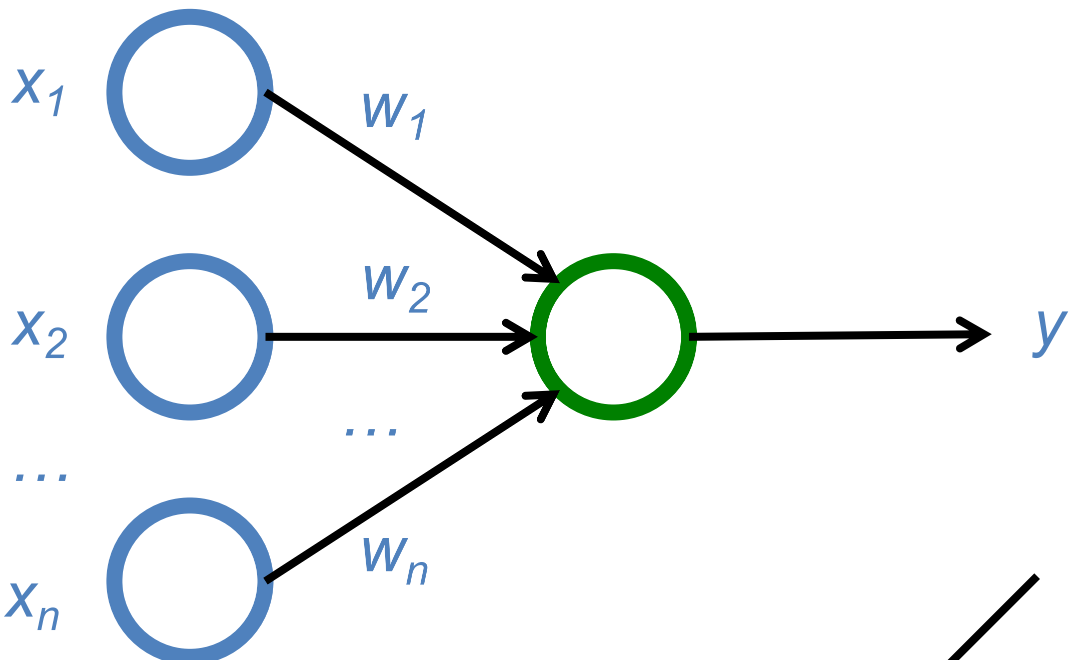
Neuron and Synapse

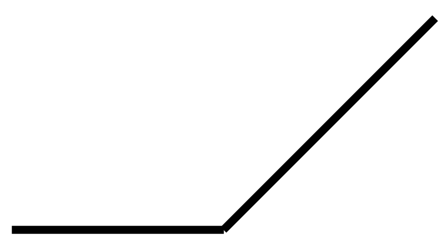


Source: https://en.wikipedia.org/wiki/Neuron

The Neuron

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

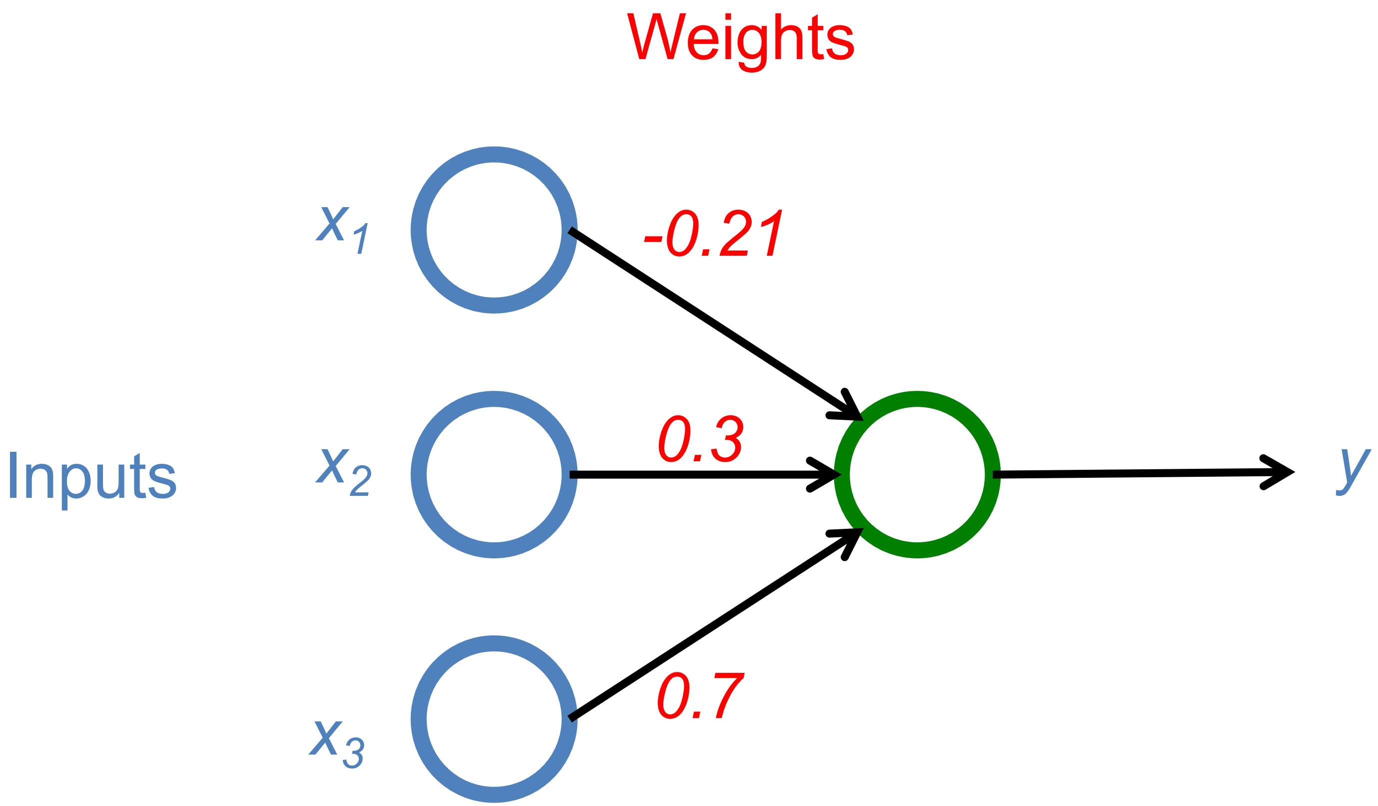




$$F(x) = \max(0, x)$$

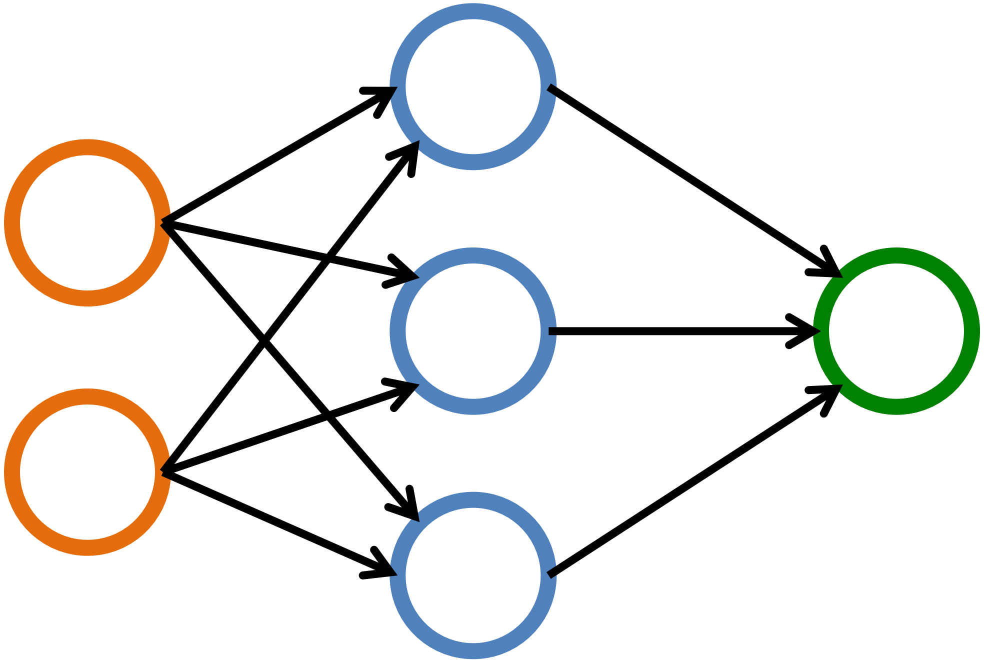
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

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



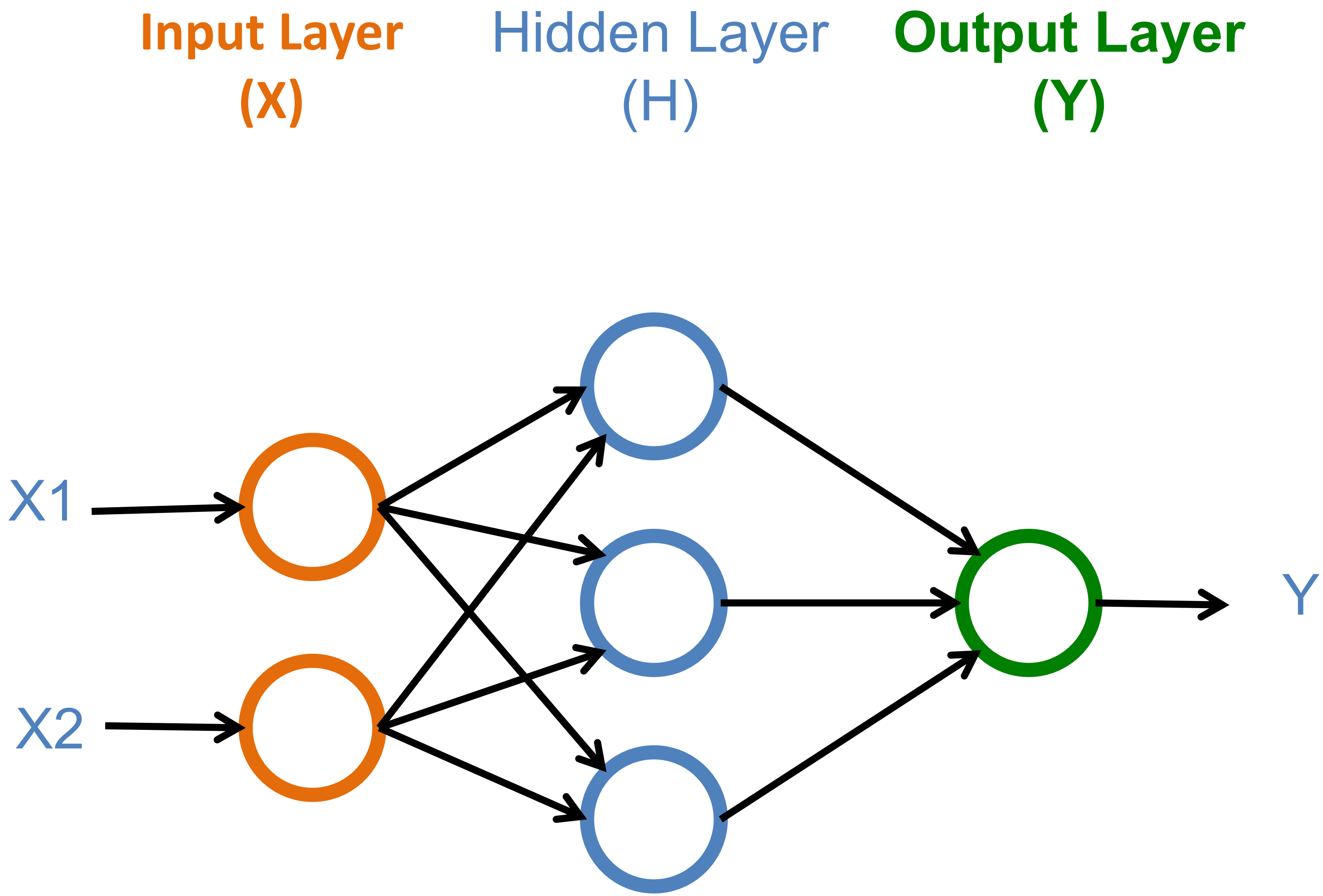
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Networks



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

Neural Networks



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

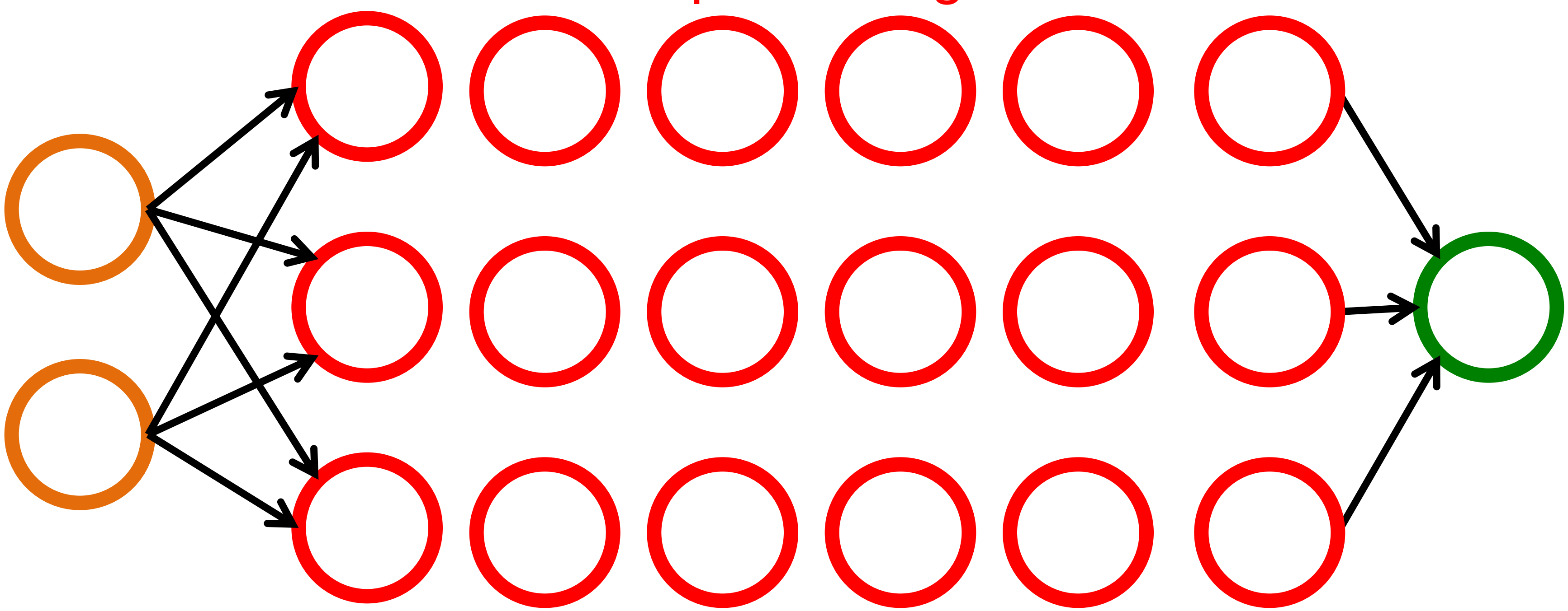
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

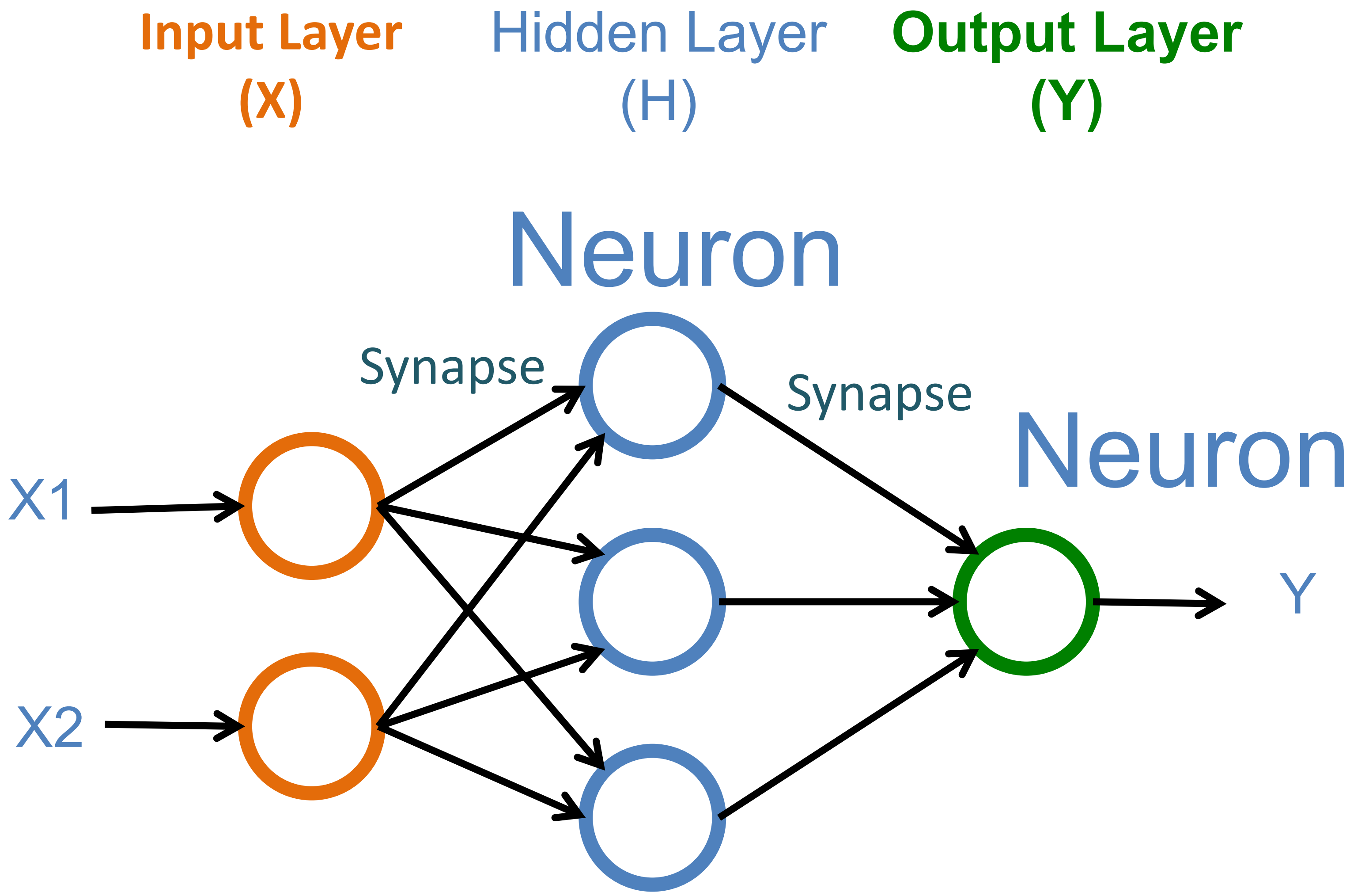
Output Layer
(Y)

Deep Neural Networks
Deep Learning



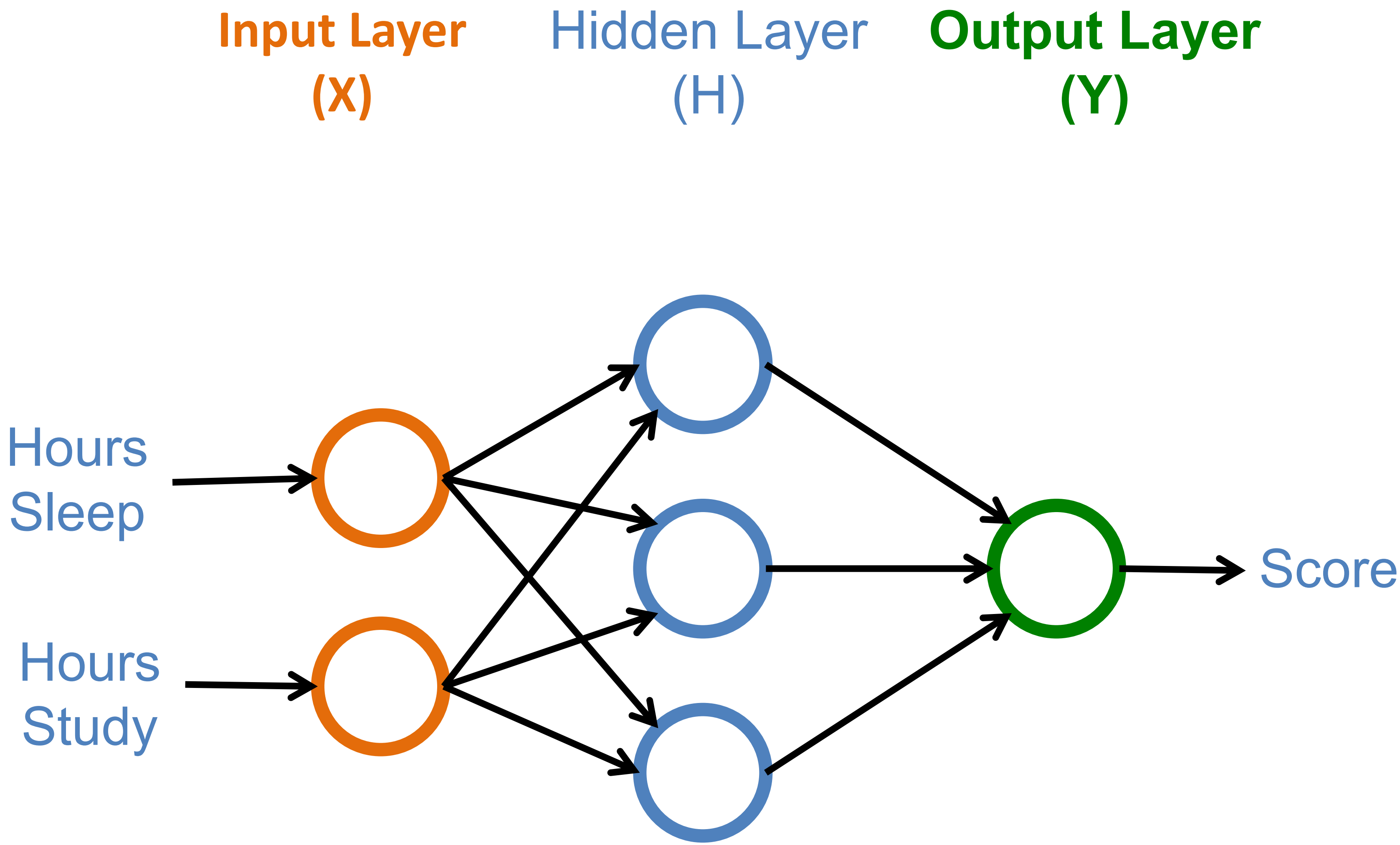
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



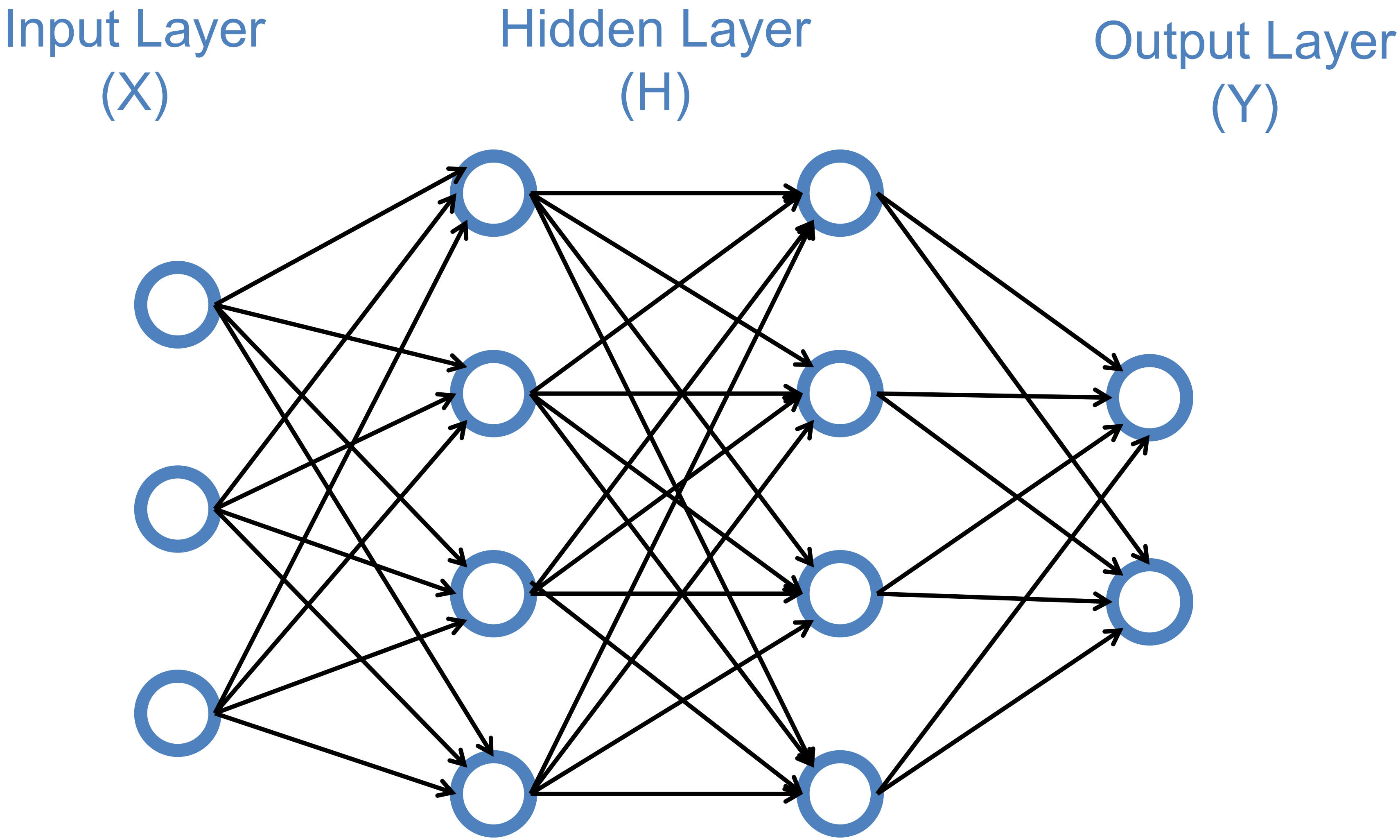
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



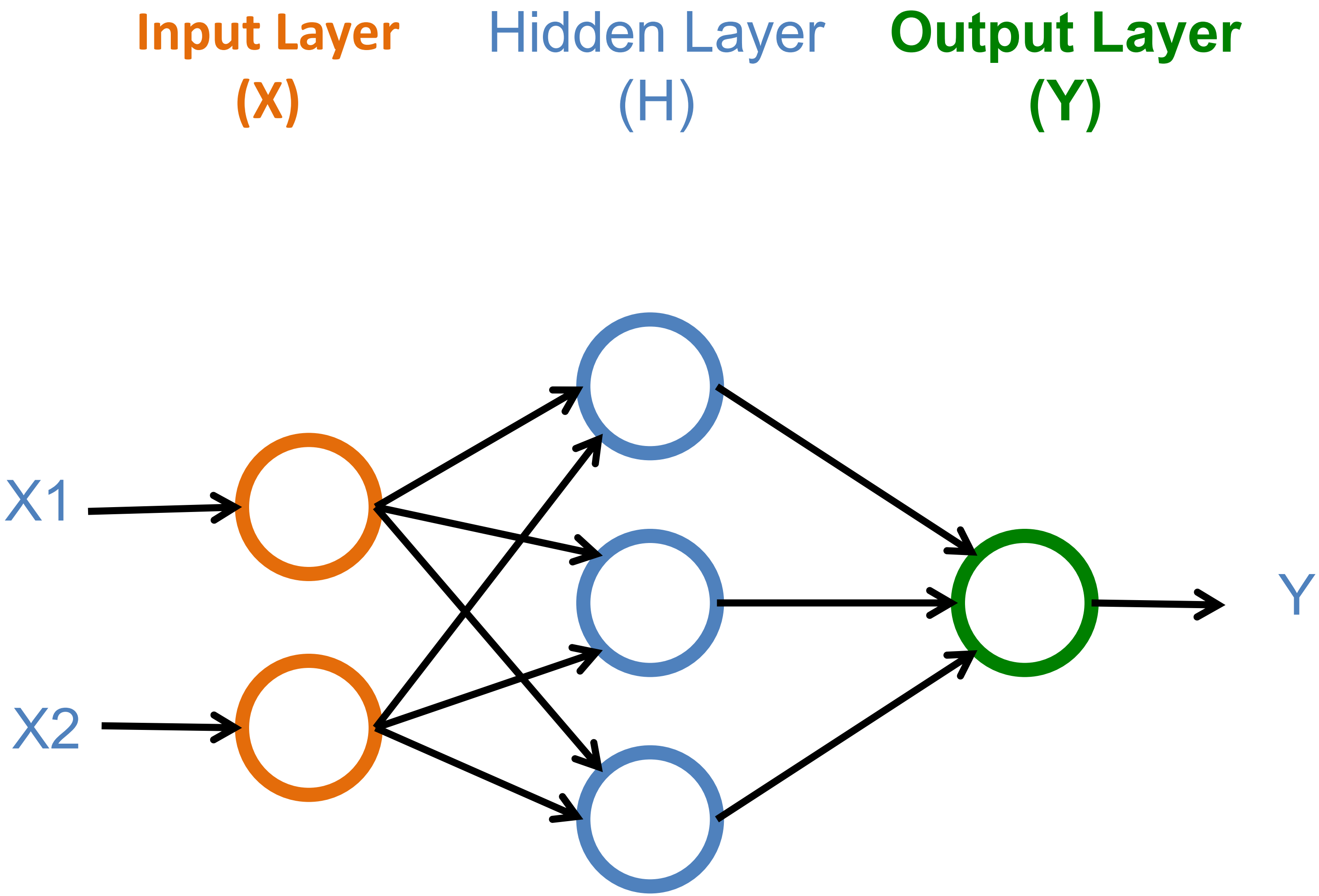
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



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

Neural Networks



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

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

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

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

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

$$Y = W X + b$$

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

Output

input

$$Y = W X + b$$

Weights

bias

Trained

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

$$W X + b = Y \quad \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \longrightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Scores $\longrightarrow$ Probabilities

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

SoftMAX

W X + b = Y

$$\begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \longrightarrow \boxed{S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}} \longrightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Logits

Scores ⟶ Probabilities

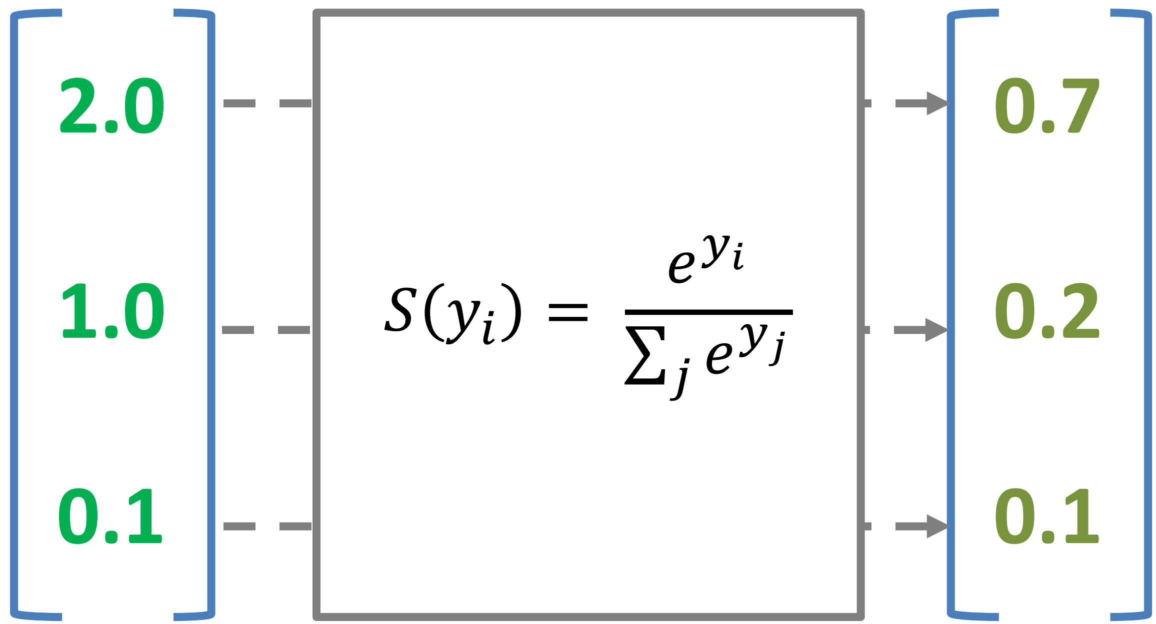
Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

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

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

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

W X + b = Y



Logits

Scores → Probabilities

Training a Network

=

Minimize the Cost Function

Training a Network

=

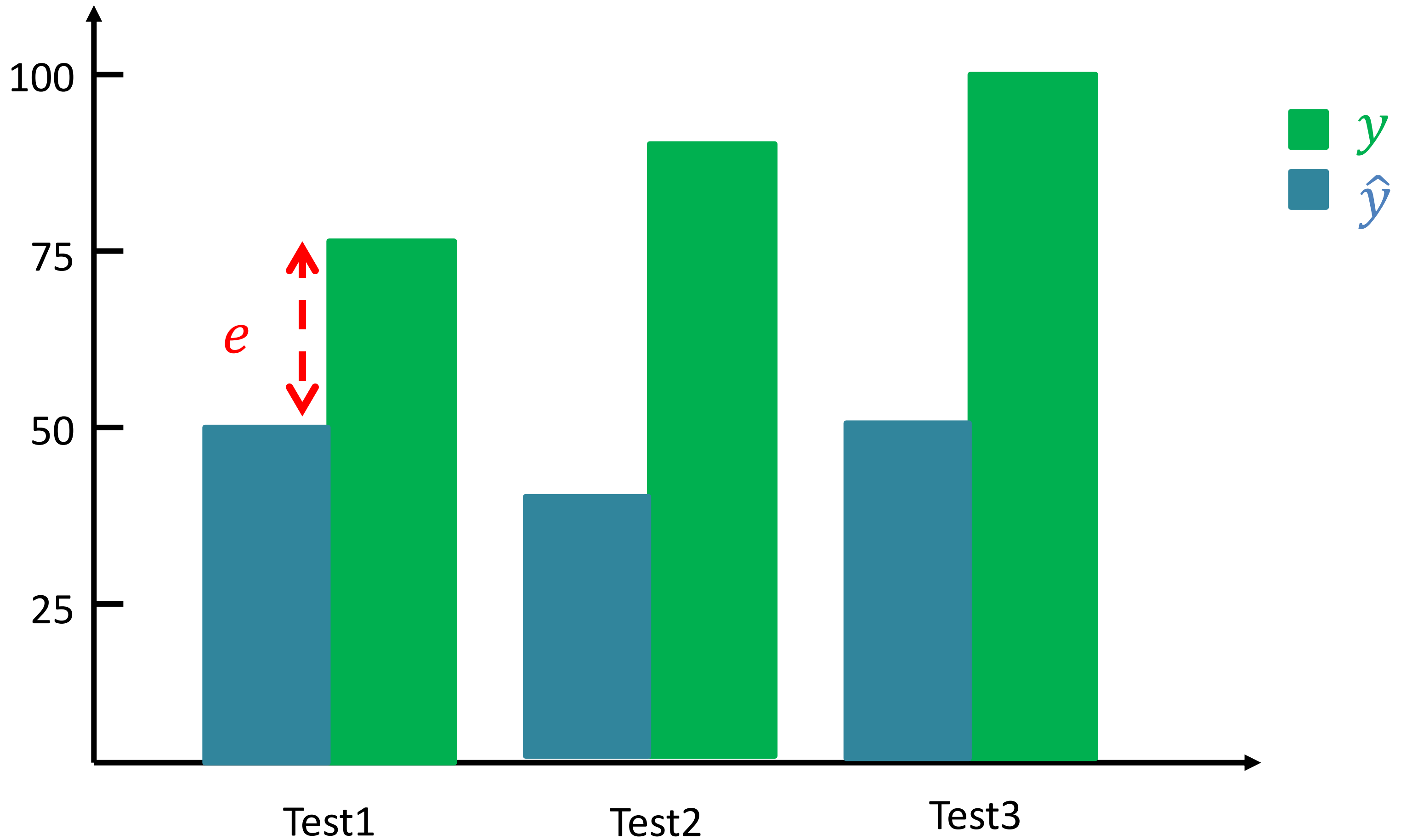
Minimize the **Cost** Function

Minimize the **Loss** Function

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

Error = Predict Y - Actual Y

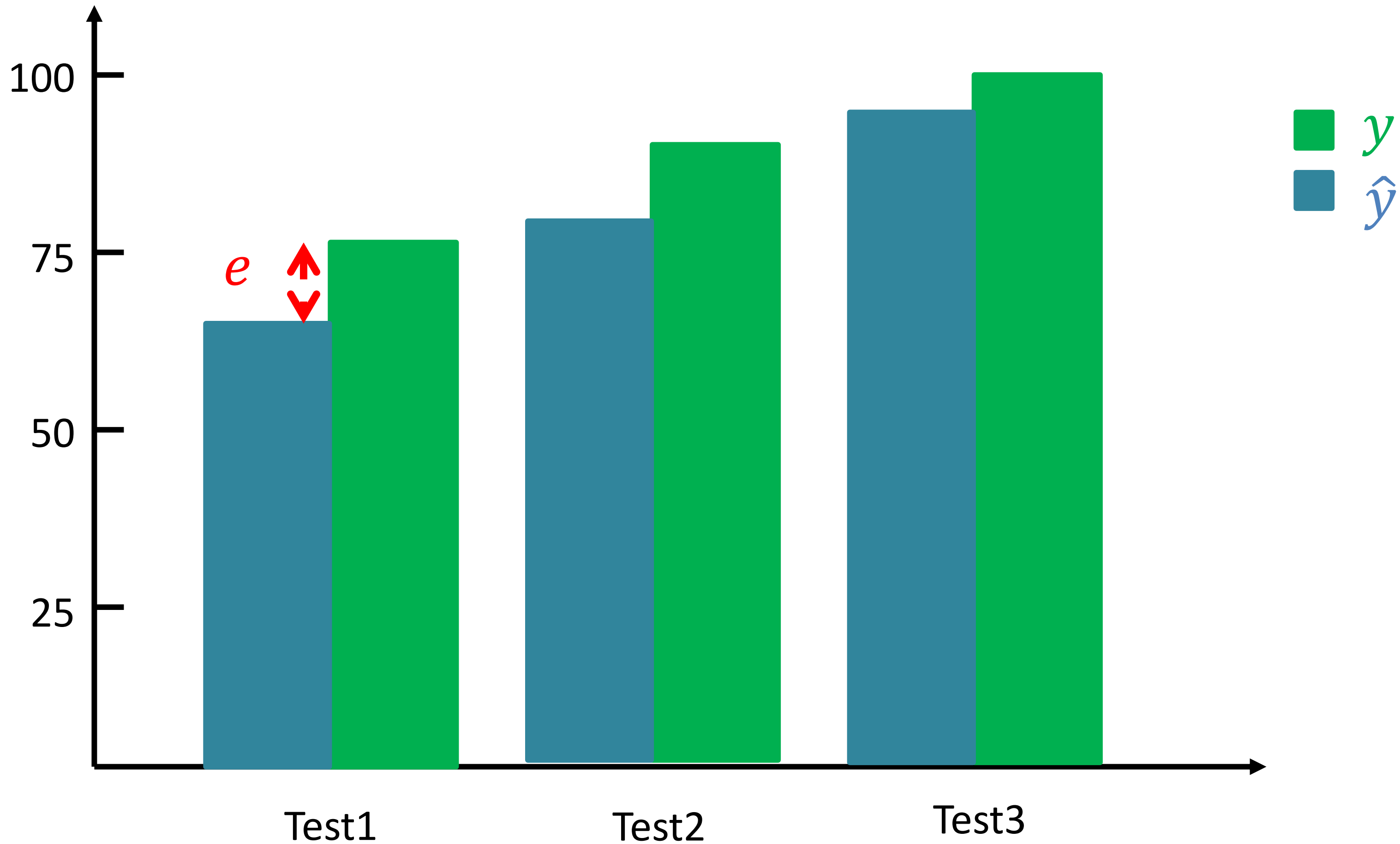
Error : Cost : Loss



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

Error = Predict Y - Actual Y

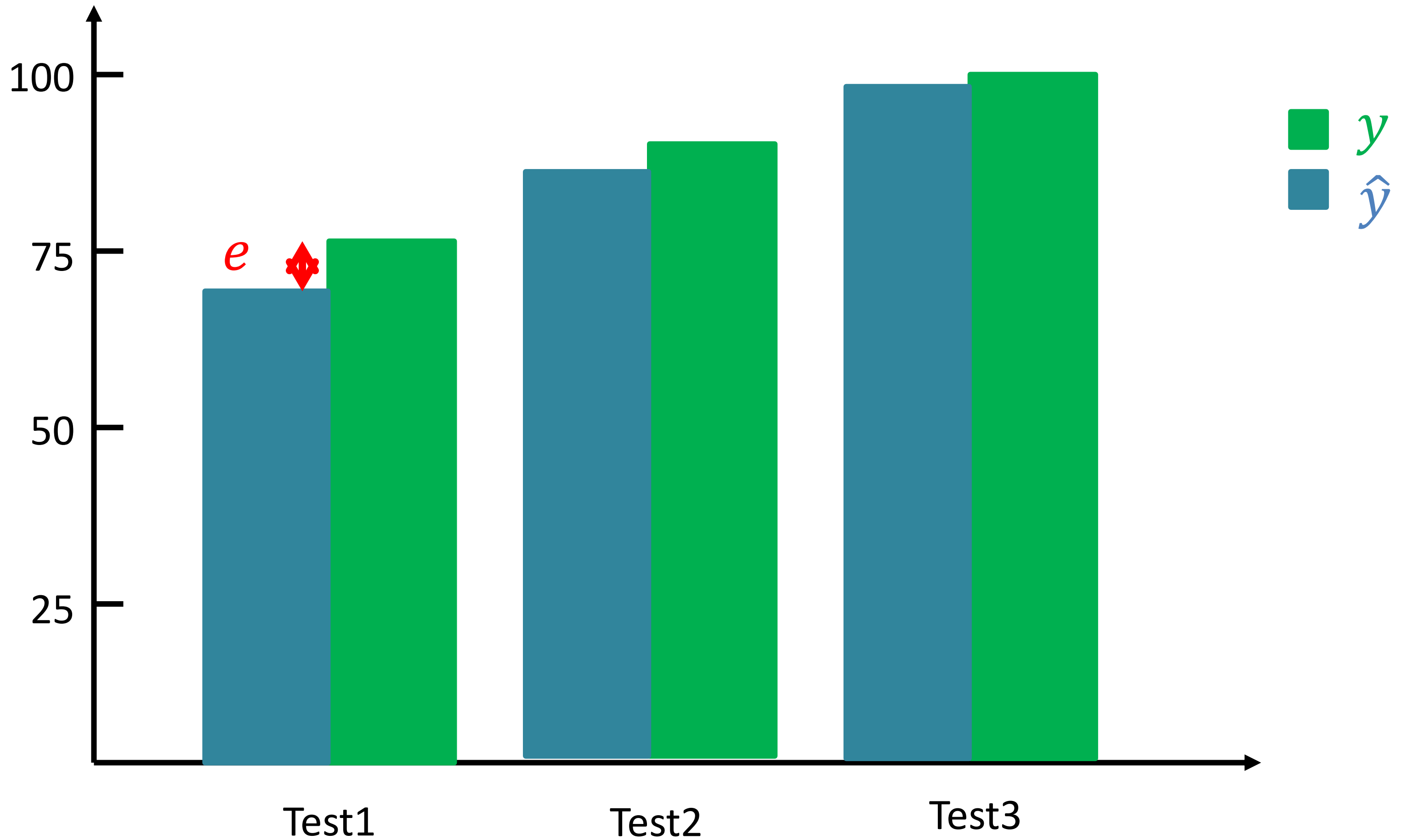
Error : Cost : Loss



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

Error = Predict Y - Actual Y

Error : Cost : Loss

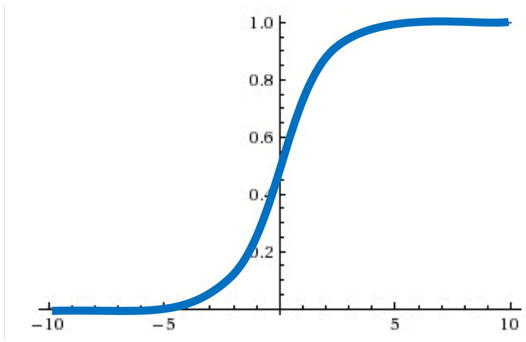


Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Activation Functions

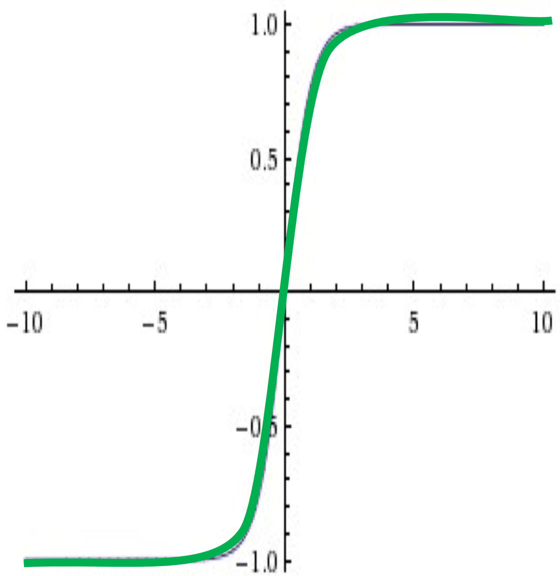
Activation Functions

Sigmoid



[0, 1]

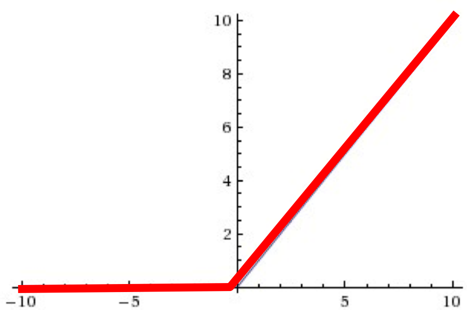
TanH



[-1, 1]

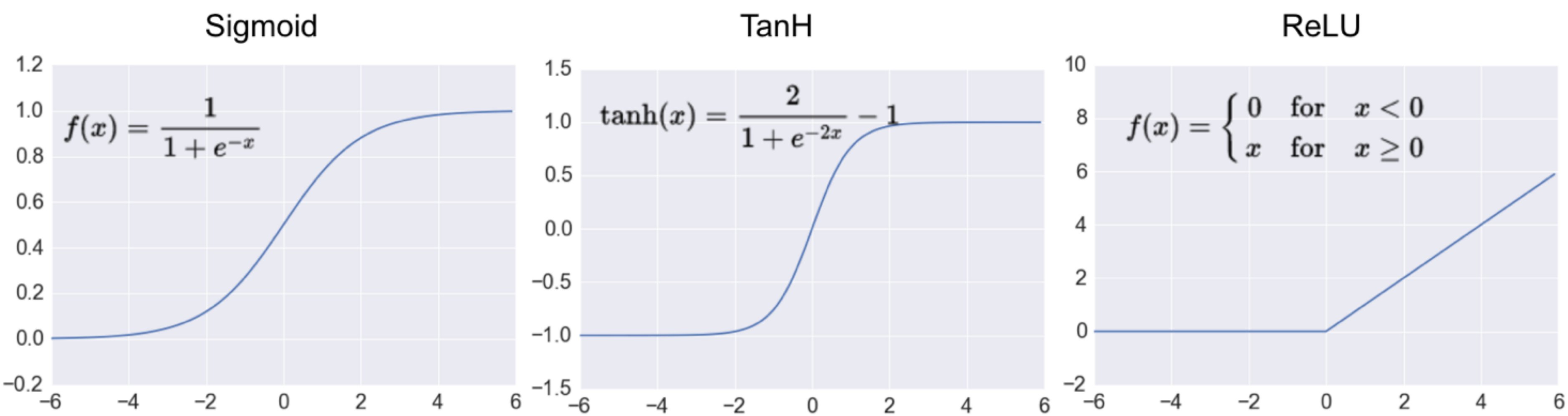
ReLU

(Rectified Linear Unit)



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

Activation Functions



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

Loss Function

Binary Classification: 2 Class

Activation Function:
Sigmoid

Loss Function:
Binary Cross-Entropy

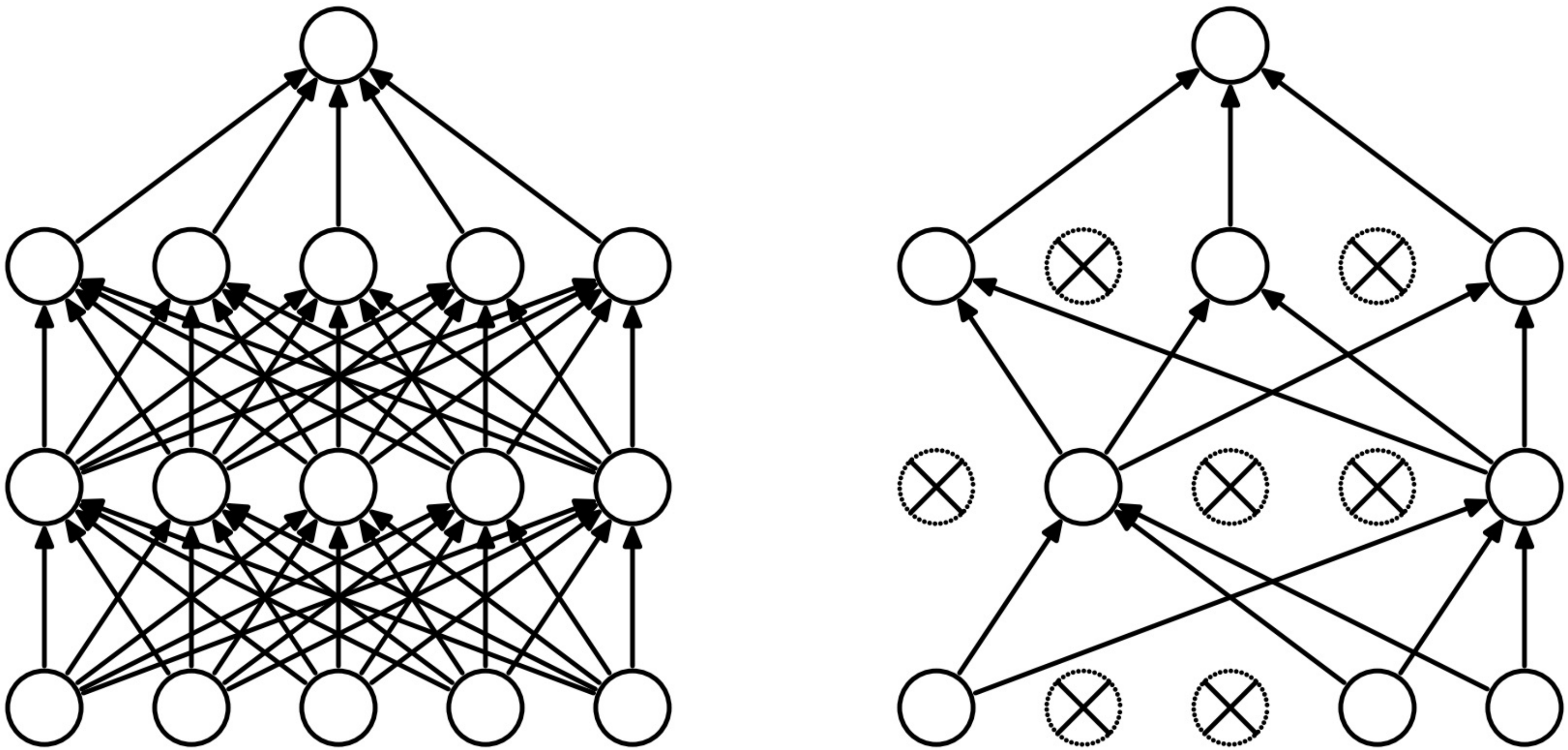
Multiple Classification: 10 Class

Activation Function:
SoftMAX

Loss Function:
Categorical Cross-Entropy

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net

(b) After applying dropout.

Source: Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research* 15, no. 1 (2014): 1929-1958.

Learning Algorithm

While not done:

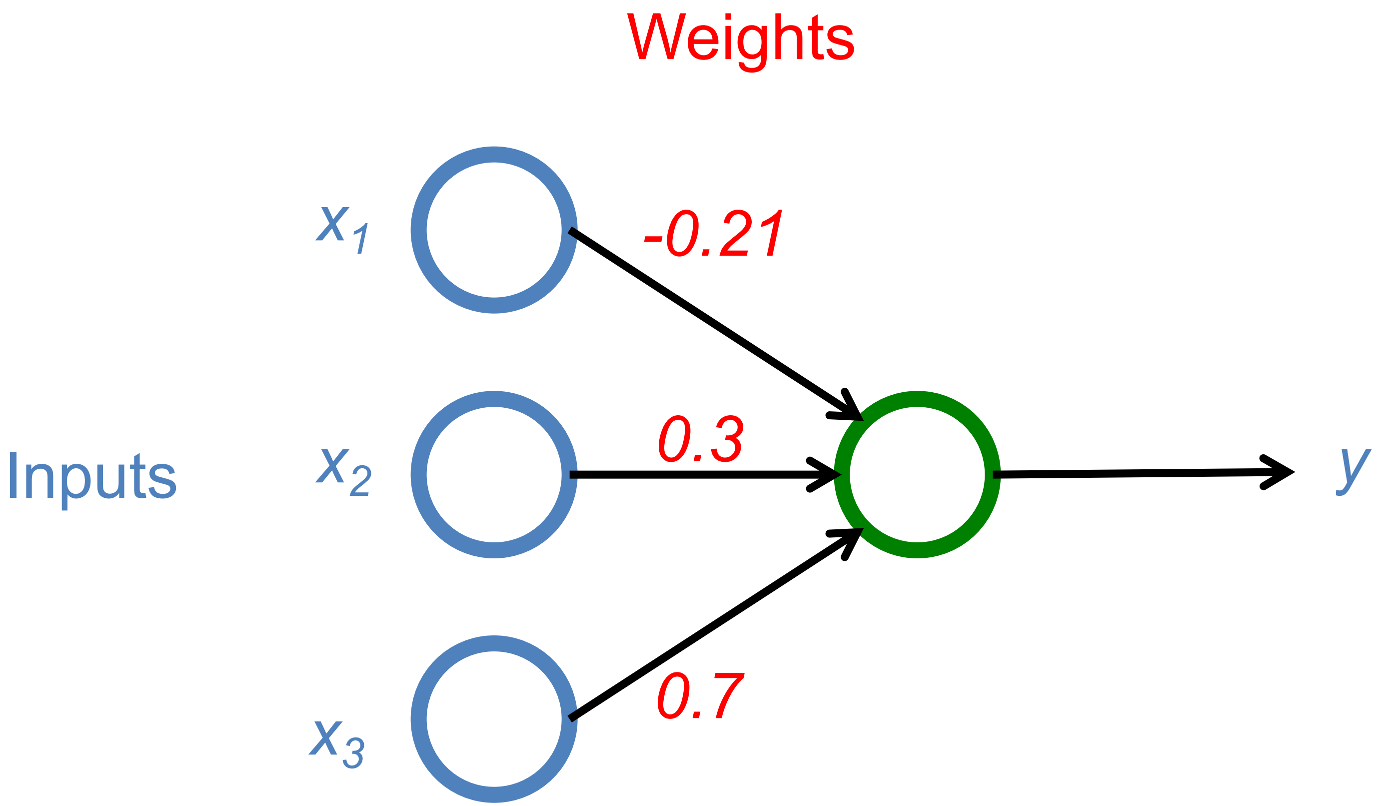
Pick a random training example “(input, label)”

Run neural network on “input”

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

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

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

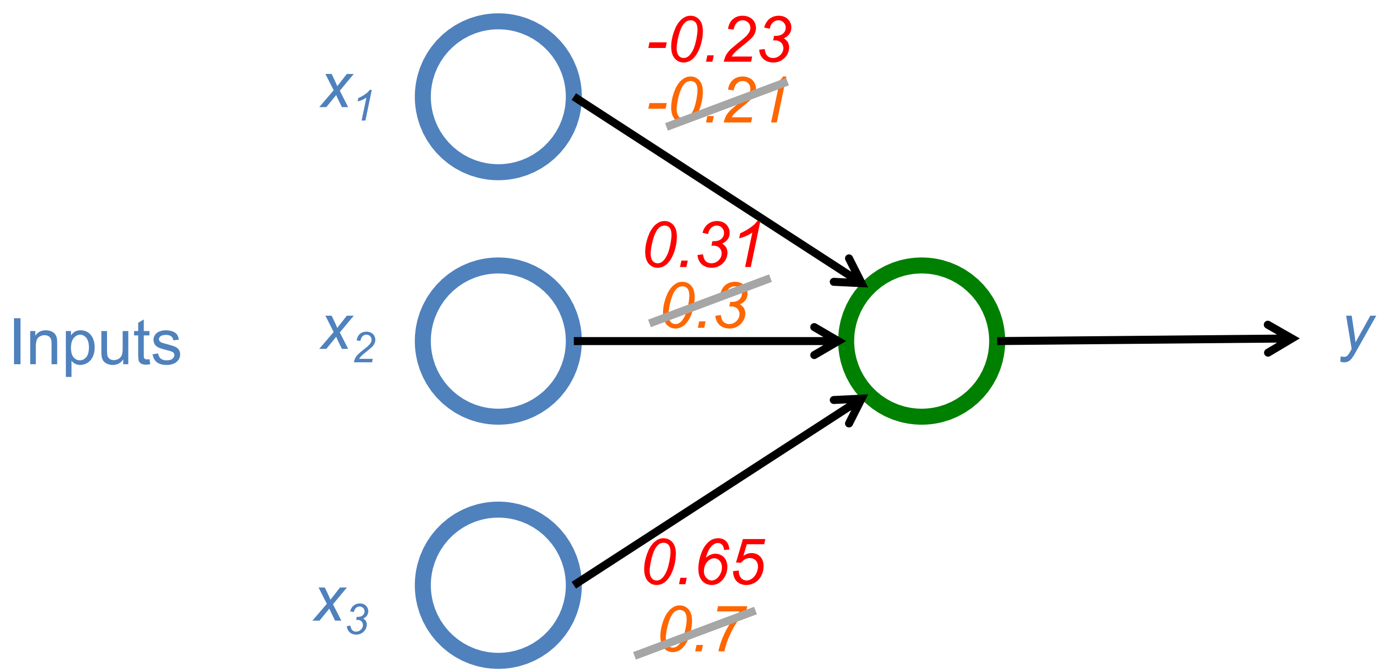


Next time:

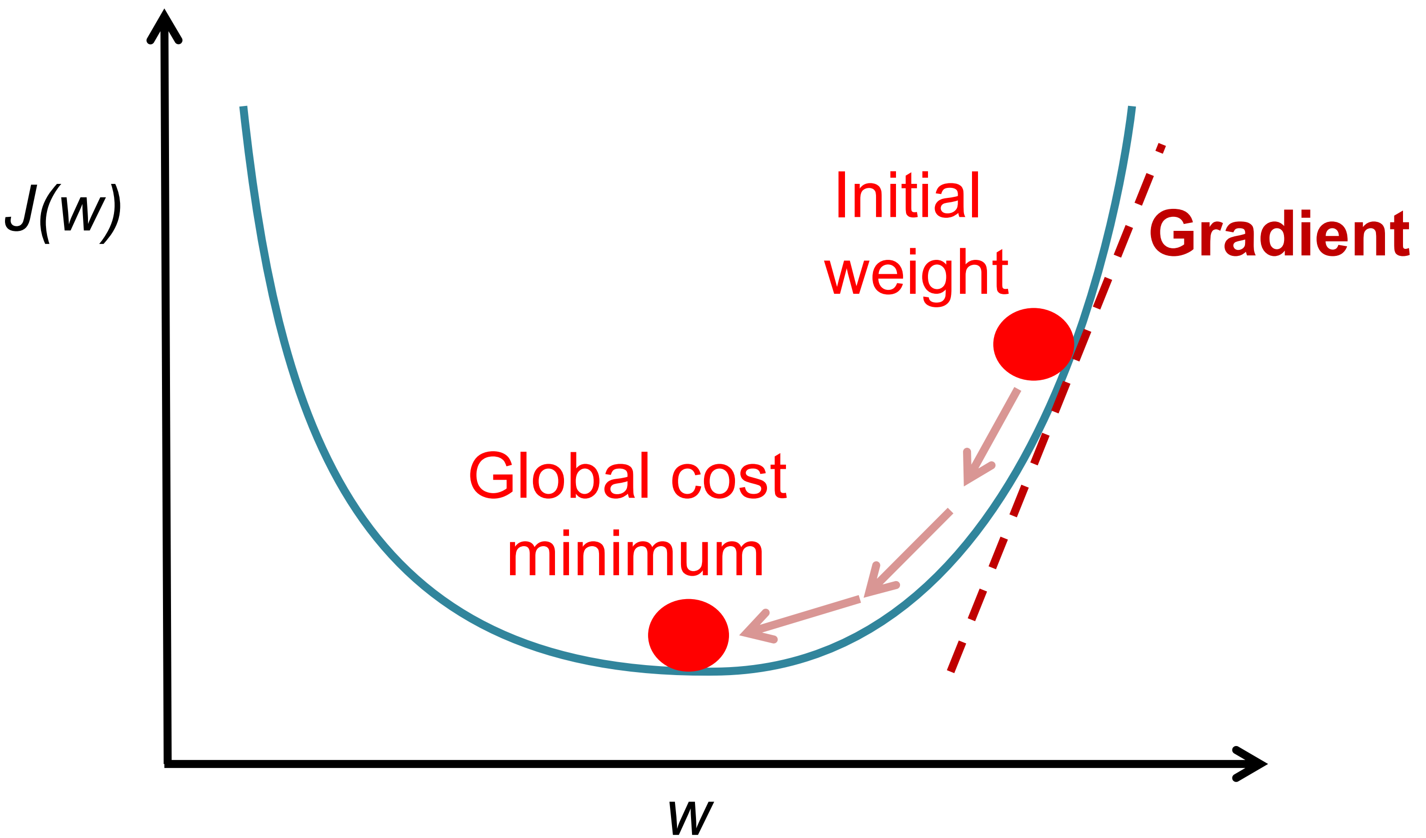
$$y = max\ (\ 0,\ -0.23\ *\ x_1 + 0.31\ *\ x_2 + 0.65\ *\ x_3\ )$$

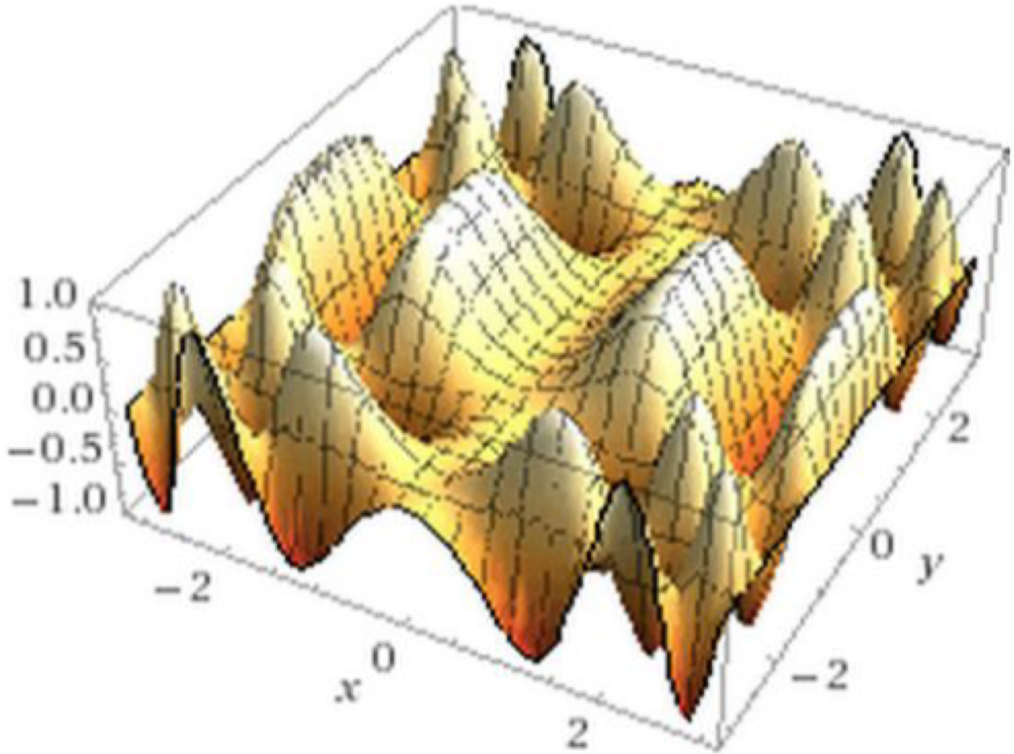
$$\cancel{y = max\ (\ 0,\ -0.21\ *\ x_1 + 0.3\ *\ x_2 + 0.7\ *\ x_3\ )}$$

Weights



Optimizer:
Stochastic Gradient Descent (SGD)

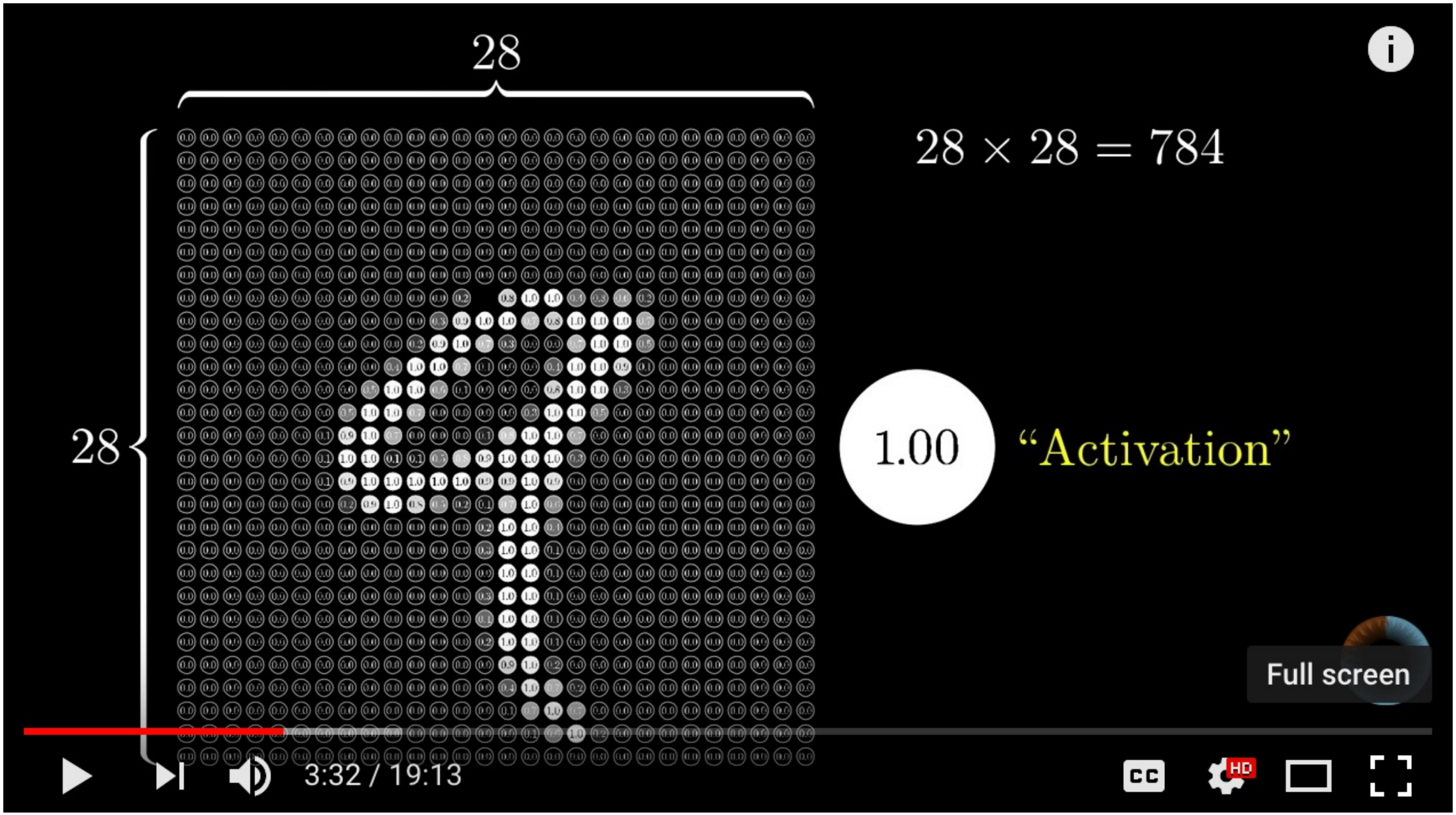




This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

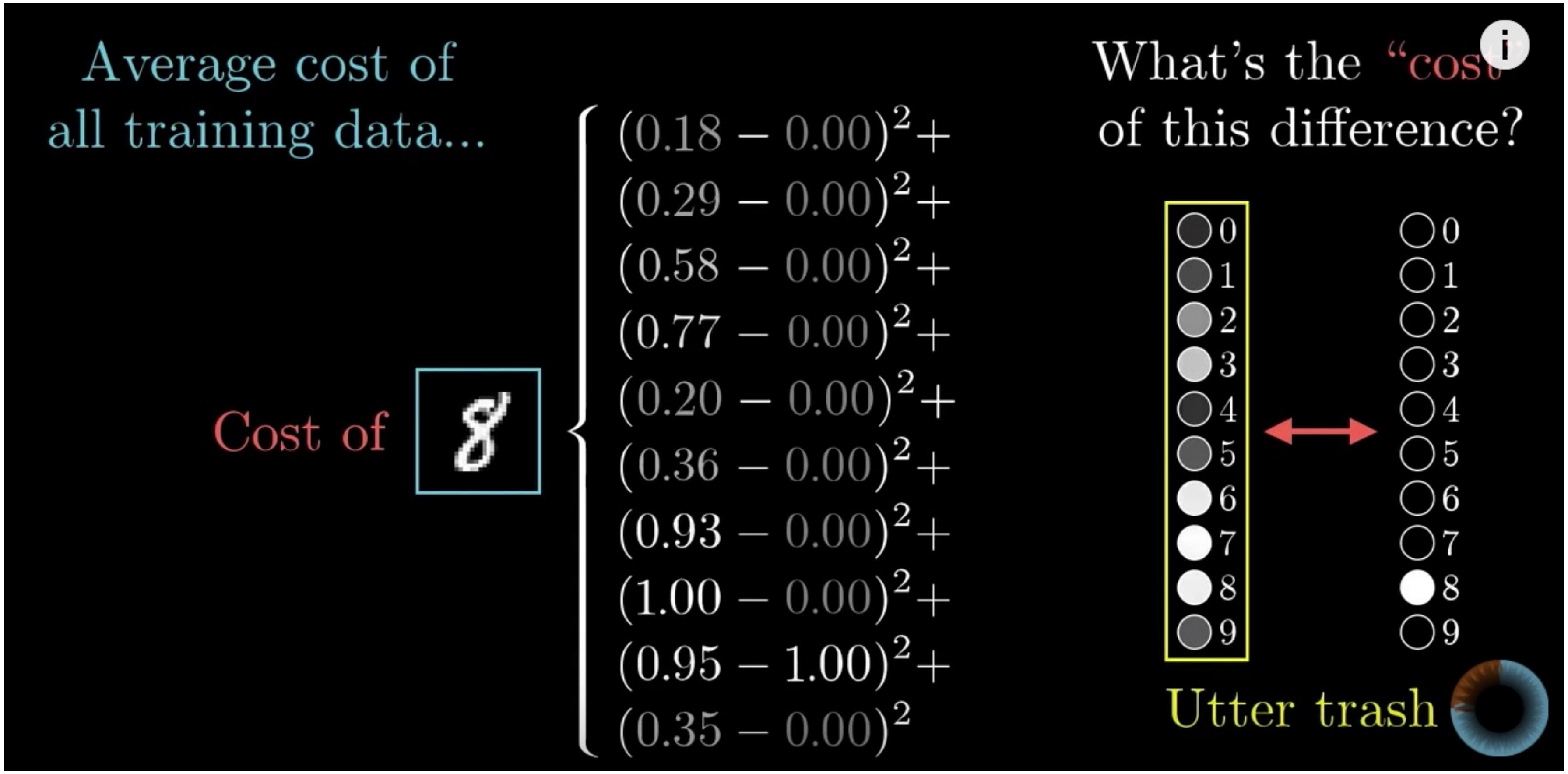
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Network and Deep Learning



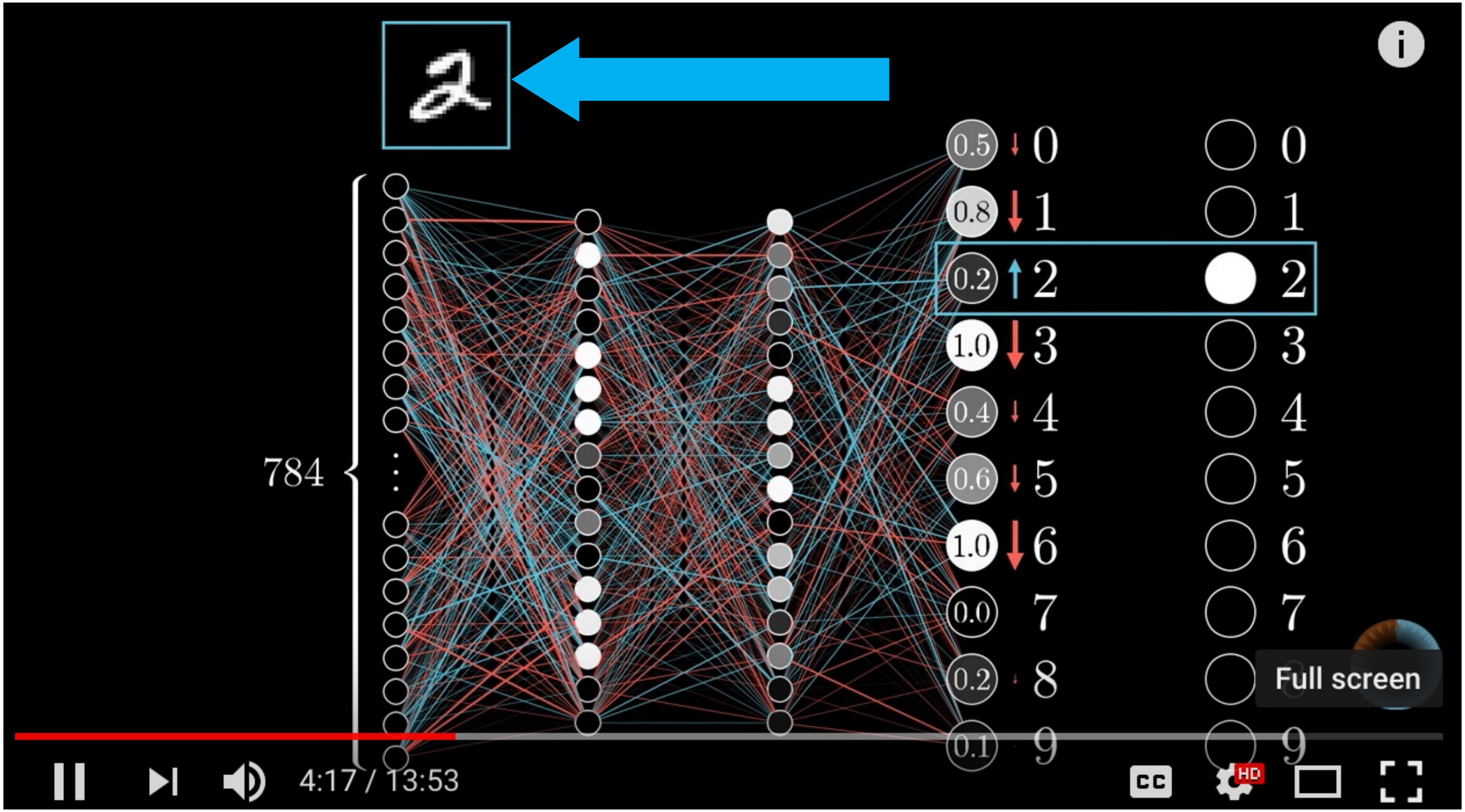
Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

Gradient Descent
how neural networks learn



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

Backpropagation



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

Learning Algorithm

While not done:

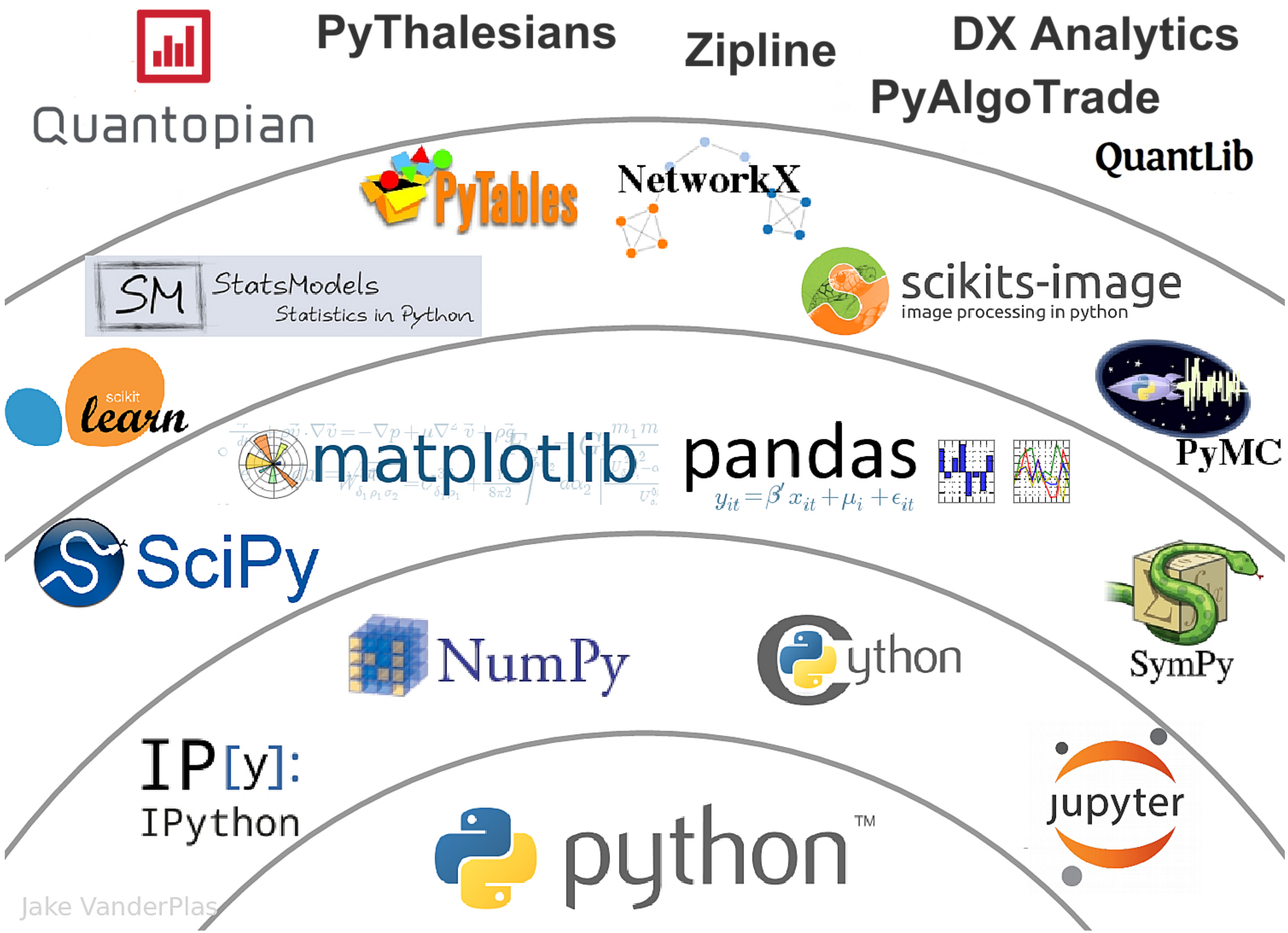
Pick a random training example “(input, label)”

Run neural network on “input”

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

Finance Big Data

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

AAPL
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

CO python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECTED EDITING

```
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```

pandas

Python Data Analysis Library

providing high-performance, easy-to-use
data structures and data analysis tools
for the Python programming language.

Source: http://pandas.pydata.org/

pandas Ecosystem

- Statistics and Machine Learning
  - Statsmodels
  - sklearn-pandas
- Visualization
  - Bokeh
  - yhat/ggplot
  - Seaborn
  - Vincent
  - IPython Vega
  - Plotly
  - Pandas-Qt
- IDE
  - IPython
  - quantopian/qgrid
  - Spyder
- API
  - pandas-datareader
  - quandl/Python
  - pydatastream
  - pandaSDMX
  - fredapi
- Domain Specific
  - Geopandas
  - xarray
- Out-of-core
  - Dask
  - Blaze
  - Odo

Source: http://pandas.pydata.org/pandas-docs/stable/index.html

pandas-datareader

pandas-datareader

latest

Search docs

What's New
Remote Data Access
Caching queries
Other Data Sources
Data Readers

Read the Docs v: latest

pandas-datareader

Up to date remote data access for pandas, works for multiple versions of pandas.

pypi v0.7.0 | build failing | coverage 79% | docs passing | health 95%

Warning

As of v0.6.0 Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated due to large changes in their API and no stable replacement.

Note

As of v0.6.0 Google finance is still functioning for historical price data, although there are frequent reports of failures. Failure is frequently encountered when bulk downloading historical price data.

Usage

Starting in 0.19.0, pandas no longer supports `pandas.io.data` or `pandas.io.wb`, so you must replace your imports from `pandas.io` with those from `pandas_datareader`:

```
from pandas.io import data, wb # becomes
from pandas_datareader import data, wb
```

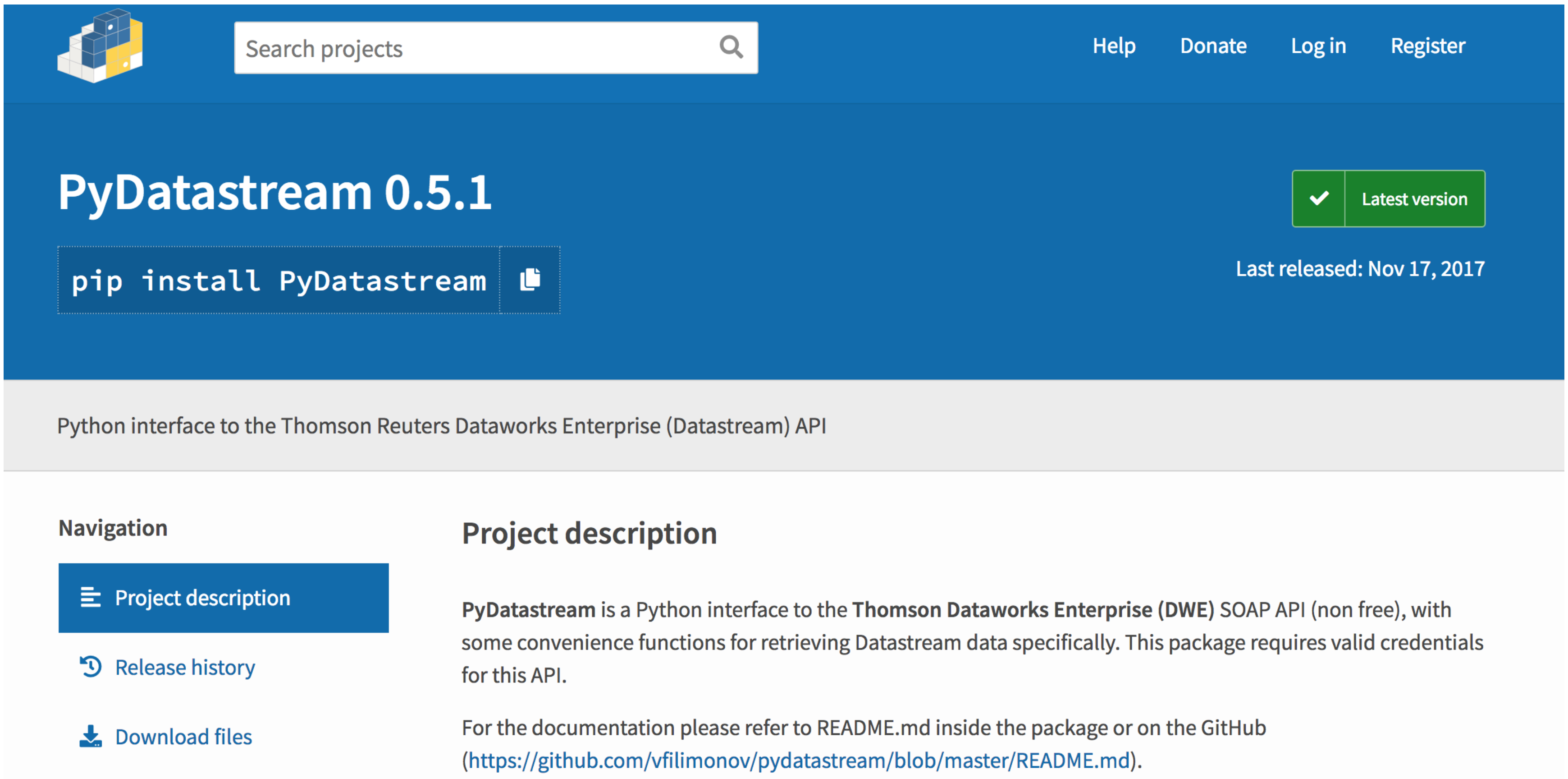
Many functions from the data module have been included in the top level API.

https://pandas-datareader.readthedocs.io/en/latest/

Quandl

https://www.quandl.com/tools/python

PyDatastream



Search projects

Help Donate Log in Register

PyDatastream 0.5.1

```
pip install PyDatastream
```

Latest version

Last released: Nov 17, 2017

Python interface to the Thomson Reuters Dataworks Enterprise (Datastream) API

Navigation

Project description

Release history

Download files

Project description

PyDatastream is a Python interface to the **Thomson Dataworks Enterprise (DWE)** SOAP API (non free), with some convenience functions for retrieving Datastream data specifically. This package requires valid credentials for this API.

For the documentation please refer to README.md inside the package or on the GitHub (https://github.com/vfilimonov/pydatastream/blob/master/README.md).

https://pypi.org/project/PyDatastream/

pandasSDMX

Table Of Contents

This Page

Show Source

Quick search

Go

pandaSDMX: Statistical Data and Metadata eXchange in Python

pandaSDMX is an Apache 2.0-licensed Python client to retrieve and acquire statistical data and metadata disseminated in SDMX 2.1, an ISO-standard widely used by institutions such as statistics offices, central banks, and international organisations. pandaSDMX exposes datasets and related structural metadata including dataflows, codelists, and datastructure definitions as pandas Series or multi-indexed DataFrames. Many other output formats and storage backends are available thanks to Odo.

Supported data providers

pandaSDMX ships with built-in support for the following agencies (others may be configured by the user):

- Australian Bureau of Statistics (ABS)
- European Central Bank (ECB)
- Eurostat
- French National Institute for Statistics (INSEE)
- Instituto Nacional de la Estadìstica y Geografia - INEGI (Mexico)
- International Monetary Fund (IMF) - SDMX Central only
- International Labour Organization (ILO)
- Italian statistics Office (ISTAT)
- Norges Bank (Norway)
- Organisation for Economic Cooperation and Development (OECD)
- United Nations Statistics Division (UNSD)
- UNESCO (free registration required)
- World Bank - World Integrated Trade Solution (WITS)

https://pandasdmx.readthedocs.io/en/latest/

Fred API

ECONOMIC RESEARCH
FEDERAL RESERVE BANK of ST. LOUIS

REGISTER | SIGN IN

Search

FRED® Economic Data | Information Services | Publications | Working Papers | Economists | About | St. Louis Fed Home

Home > Developer API > FRED API

API Keys | Terms of Use

FRED® API

General Documentation | API | Toolkits

The FRED® API is a web service that allows developers to write programs and build applications that retrieve economic data from the FRED® and ALFRED® websites hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis. Requests can be customized according to data source, release, category, series, and other preferences.

General Documentation

- Overview
- What is FRED®?
- What is ALFRED®?
- FRED® versus ALFRED®
- Real-Time Periods
- Errors

API

Categories

- fred/category – Get a category.
- fred/category/children – Get the child categories for a specified parent category.
- fred/category/related – Get the related categories for a category.
- fred/category/series – Get the series in a category.
- fred/category/tags – Get the tags for a category.
- fred/category/related_tags – Get the related tags for a category.

https://research.stlouisfed.org/docs/api/fred/

Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

! pip install pandas_datareader

```
1 ! pip install pandas_datareader
```

```
Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fb1e37fb45335ec01cceca199b8a79339137f5ed269e0/pandas_datarea
    100% |████████████████████████████████| 112kB 2.7MB/s
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5accedb0e63f5/lxml-4.2.5-cp3
    100% |████████████████████████████████| 5.8MB 7.5MB/s
Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (1.14.6)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.5)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas>=0.19.2->pandas_data
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
```

conda install pandas-datareader

imyday — -bash — 80×24

```
iMyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata ...........
Solving package specifications: .

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

The following NEW packages will be INSTALLED:

    pandas-datareader: 0.2.1-py36_0
    requests-file:     1.4.1-py36_0

Proceed ([y]/n)? y

requests-file- 100% |################################| Time: 0:00:00   1.55 MB/s
pandas-datarea 100% |################################| Time: 0:00:00 409.66 kB/s
iMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
_license                  1.1                      py36_1
alabaster                 0.7.9                    py36_0
anaconda                  4.3.1               np111py36_0
anaconda-client           1.6.0                    py36_0
anaconda-navigator        1.5.0                    py36_0
anaconda-project          0.4.1                    py36_0
```


AAPL
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

Finance Data from Yahoo Finance

```
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

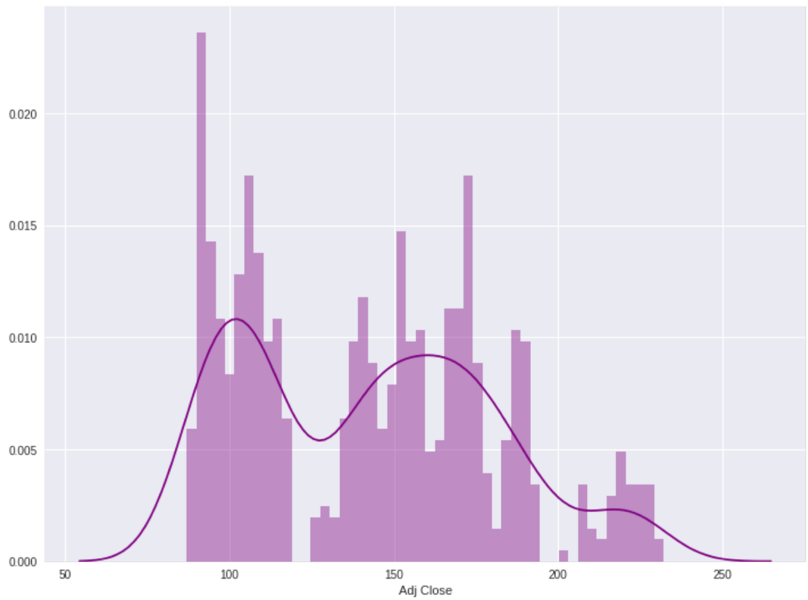
```
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month,
end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

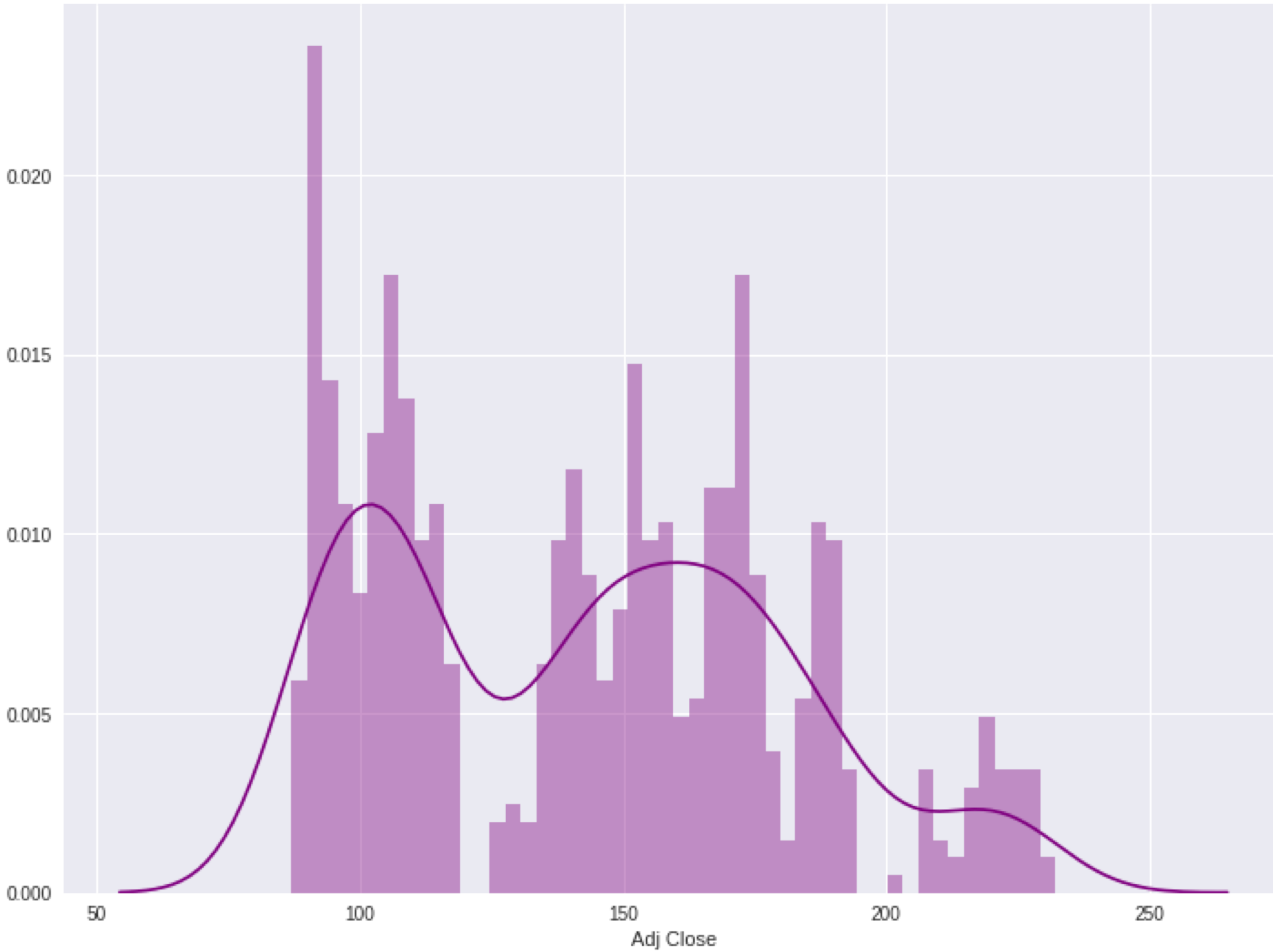
```python
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL’, label='Adj Close')
```

```
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0),
rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0),
rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'],
color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])
```


AAPL
Adj Close
220
200
180
160
140
120
100
80
1e8
Volume
1.0
0.5
0.0
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10

```
 # set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(),
bins=50, color='purple')
```




0.020
0.015
0.010
0.005
0.000
50
100
150
200
250
Adj Close

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
#5 days
df['MA20'] = df['Adj
Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj
Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj
Close'],'MA05': df['MA05'],'MA20': df['MA20'],
'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True,
title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```


AAPL
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

Finance Data from Quandl

```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

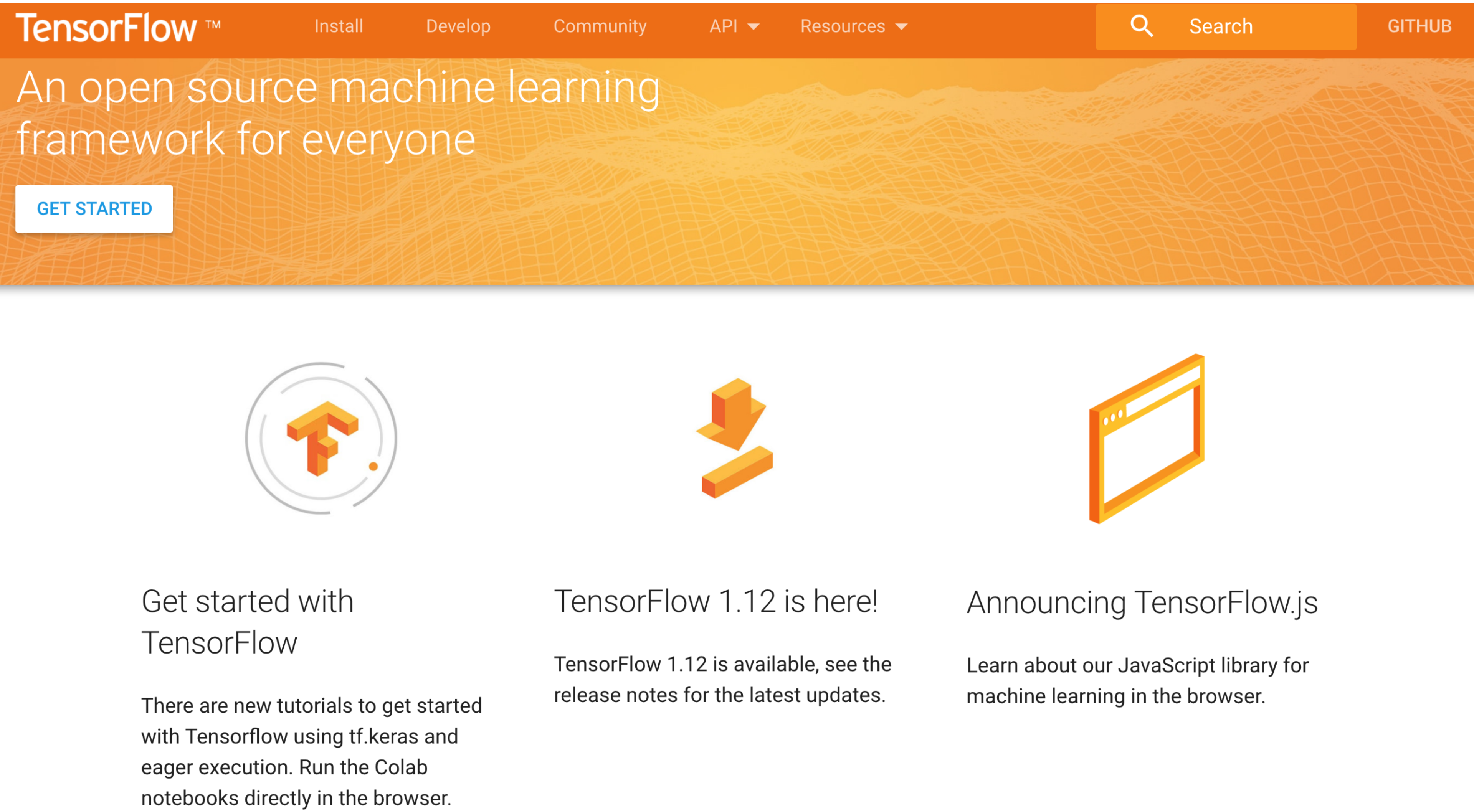
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: FutureWarning: from_csv is deprecated. Please use read_csv(...) instead
  """
```

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2017-12-22	174.68	175.424	174.500	175.01	16052615.0	0.0	1.0	174.68	175.424	174.500	175.01	16052615.0
2017-12-26	170.80	171.470	169.679	170.57	32968167.0	0.0	1.0	170.80	171.470	169.679	170.57	32968167.0
2017-12-27	170.10	170.780	169.710	170.60	21672062.0	0.0	1.0	170.10	170.780	169.710	170.60	21672062.0
2017-12-28	171.00	171.850	170.480	171.08	15997739.0	0.0	1.0	171.00	171.850	170.480	171.08	15997739.0
2017-12-29	170.52	170.590	169.220	169.23	25643711.0	0.0	1.0	170.52	170.590	169.220	169.23	25643711.0

Source: https://www.quandl.com/tools/python

Deep Learning with TensorFlow

TensorFlow



https://www.tensorflow.org/

Deep Learning Software

- TensorFlow
  - TensorFlow™ is an open source software library for high performance numerical computation.
- Keras
  - Deep Learning library for TensorFlow, CNTK
- PyTorch
  - An open source deep learning platform that provides a seamless path from research prototyping to production deployment.
- CNTK
  - Computational Network Toolkit by Microsoft Research

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

tf.keras

Keras:

High-level API for TensorFlow

Keras

Keras Documentation

Search docs

- Home
  - Keras: The Python Deep Learning library
  - You have just found Keras.
  - Guiding principles
  - Getting started: 30 seconds to Keras
  - Installation
  - Configuring your Keras backend
  - Support
  - Why this name, Keras?
- Activations
- Applications
- Backend
- Callbacks
- Constraints
- Contributing
- Datasets
- Initializers
- Losses
- Metrics

GitHub Next »

Docs » Home

Edit on GitHub

Keras: The Python Deep Learning library

You have just found Keras.

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

Use Keras if you need a deep learning library that:

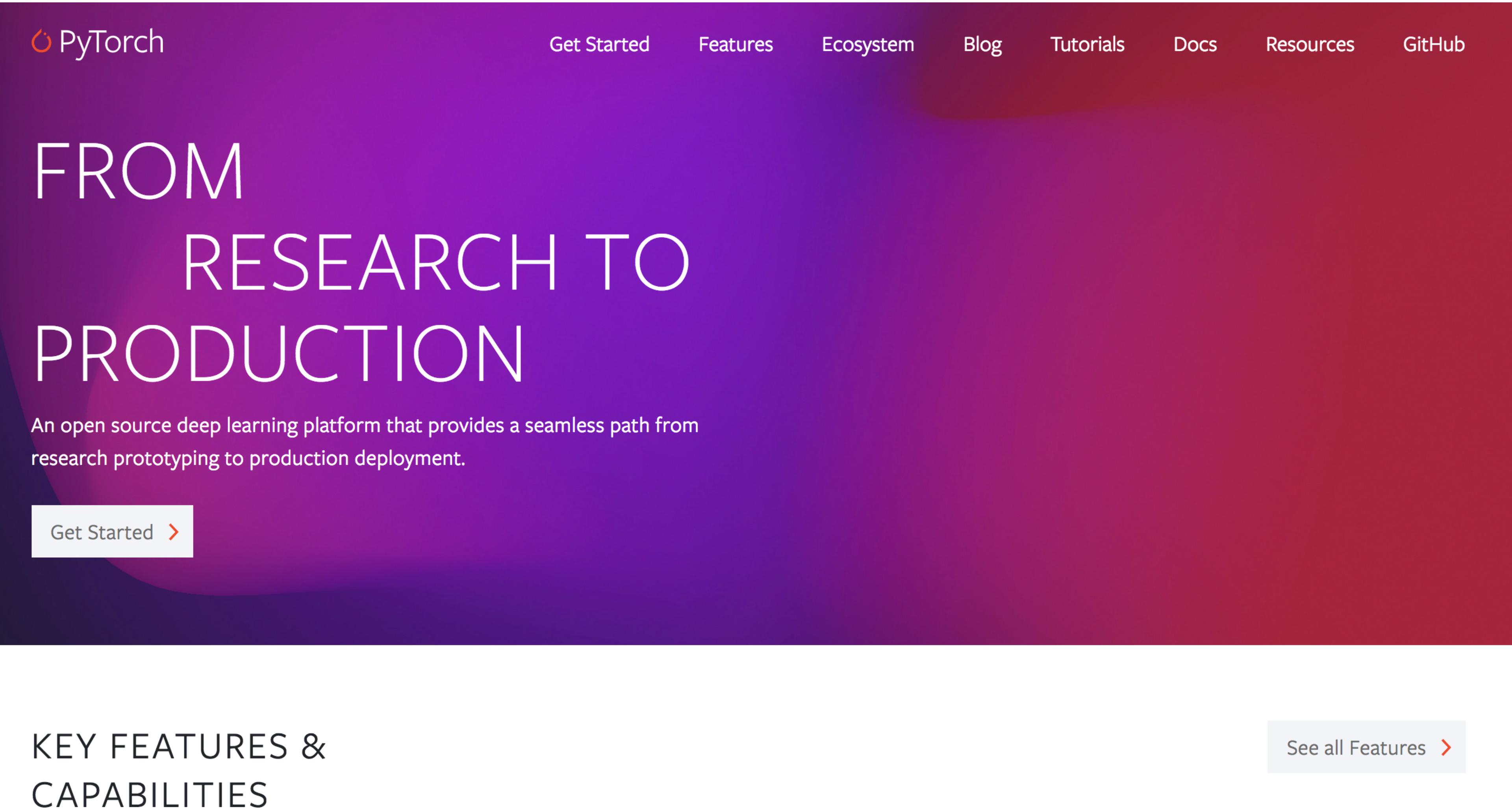
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

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

http://keras.io/

PyTorch



http://pytorch.org/

Keras

Keras

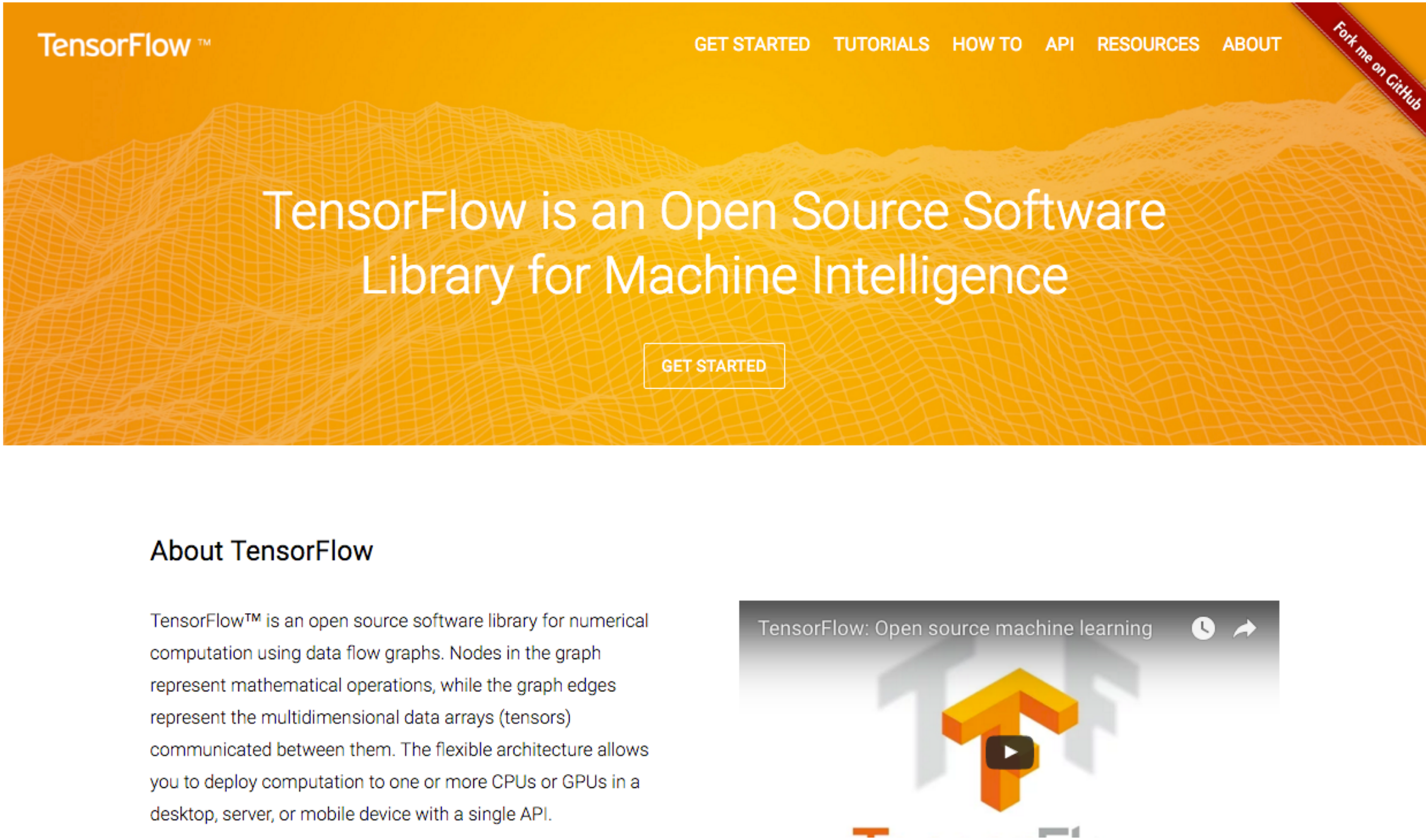
- Keras is a high-level neural networks API
- Written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- It was developed with a focus on enabling fast experimentation.
- Being able to go from idea to result with the least possible delay is key to doing good research.

Source: https://keras.io/

TensorFlow

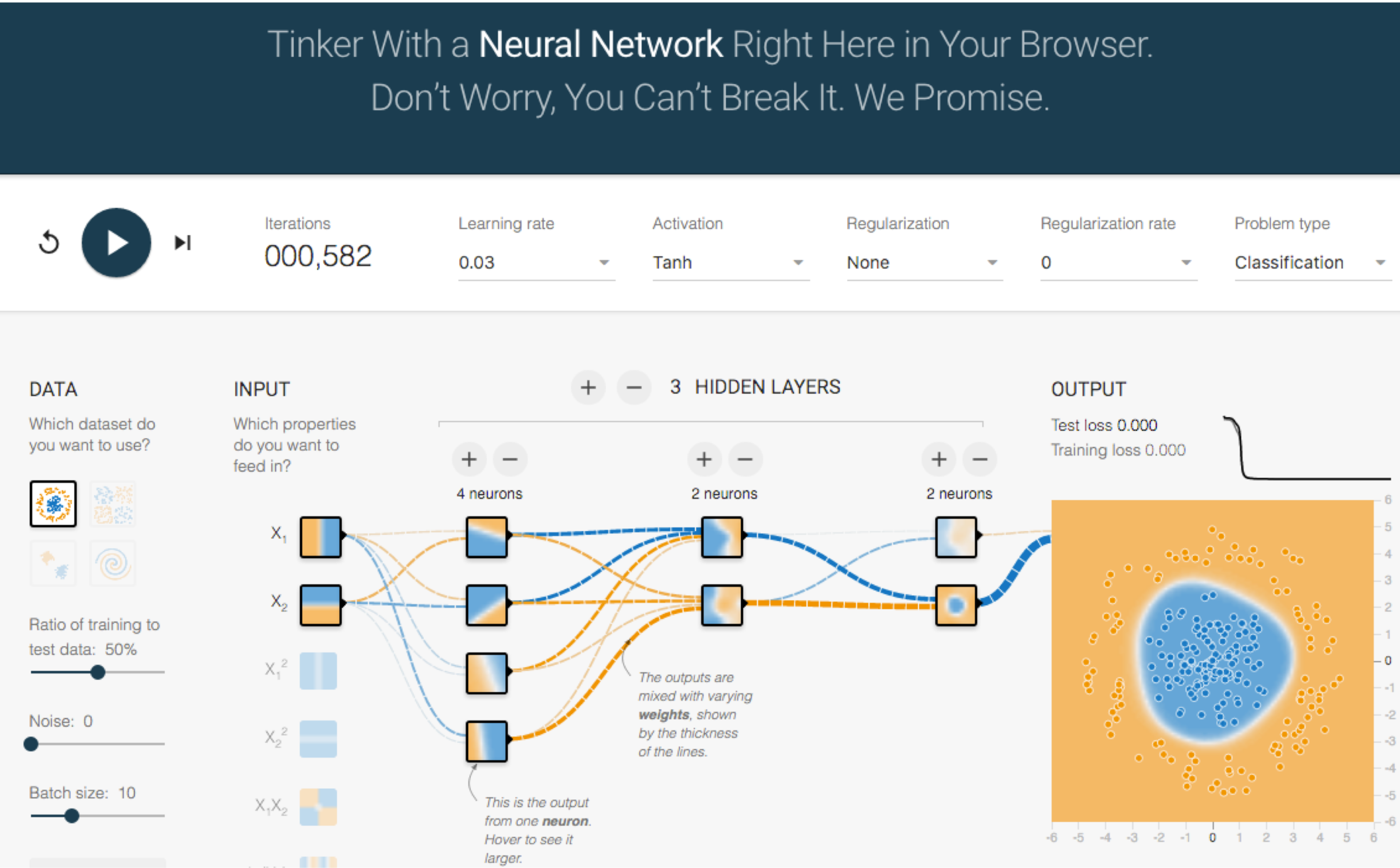
Source: https://github.com/tensorflow/tensorflow

Google TensorFlow



https://www.tensorflow.org/

TensorFlow Playground



http://playground.tensorflow.org/

TensorFlow

is an

Open Source Software Library

for

Machine Intelligence

https://www.tensorflow.org/

TensorFlow

numerical computation using data flow graphs

https://www.tensorflow.org/

Tensor

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

https://www.tensorflow.org/

Nodes:

mathematical operations

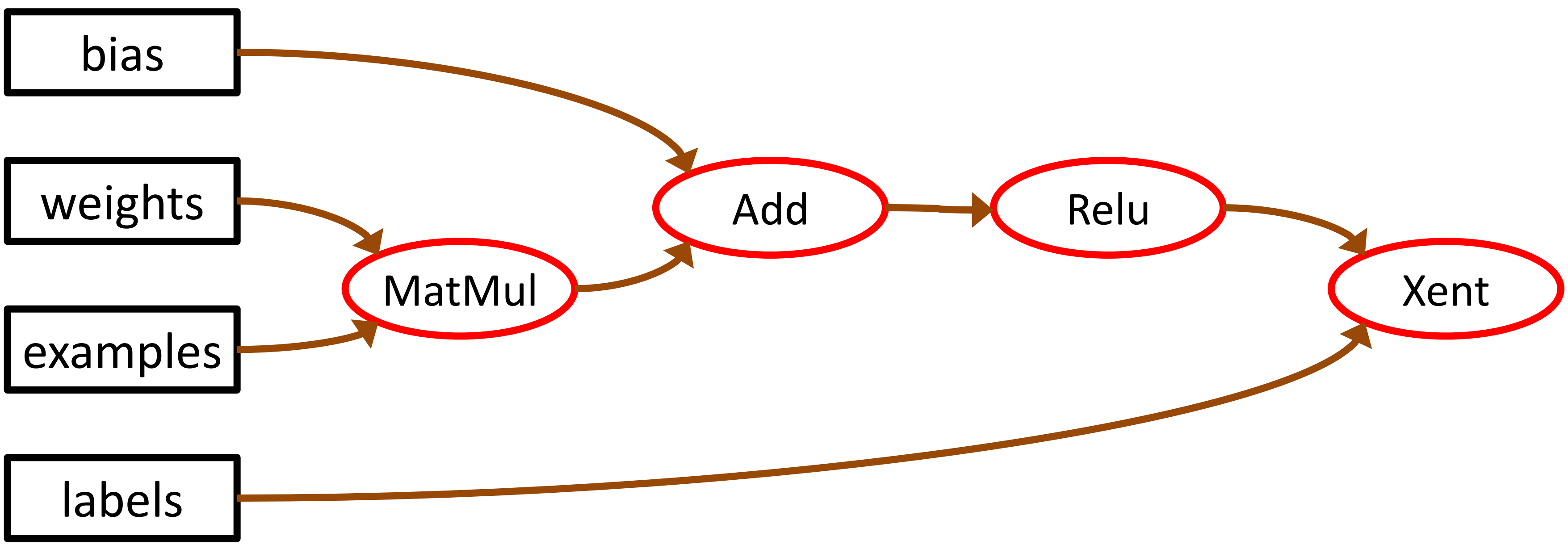
edges:

multidimensional data arrays (tensors)

communicated between nodes

Computation is a Dataflow Graph

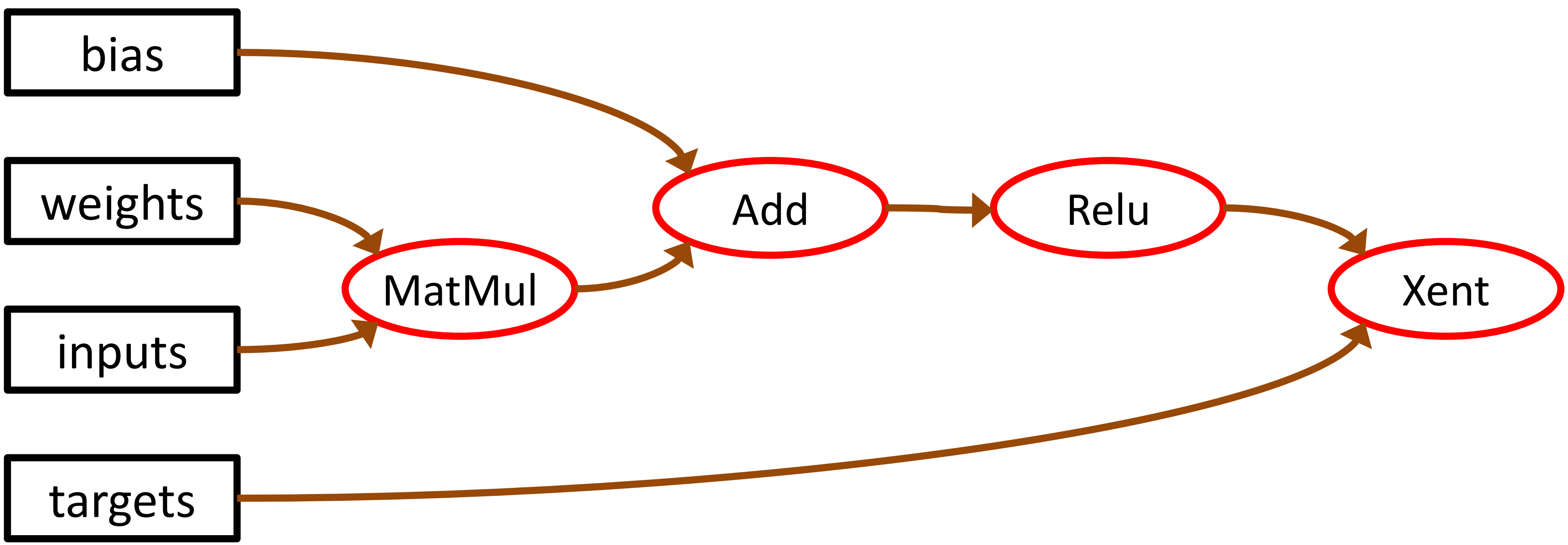
Graph of Nodes,
also called Operations or ops.



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

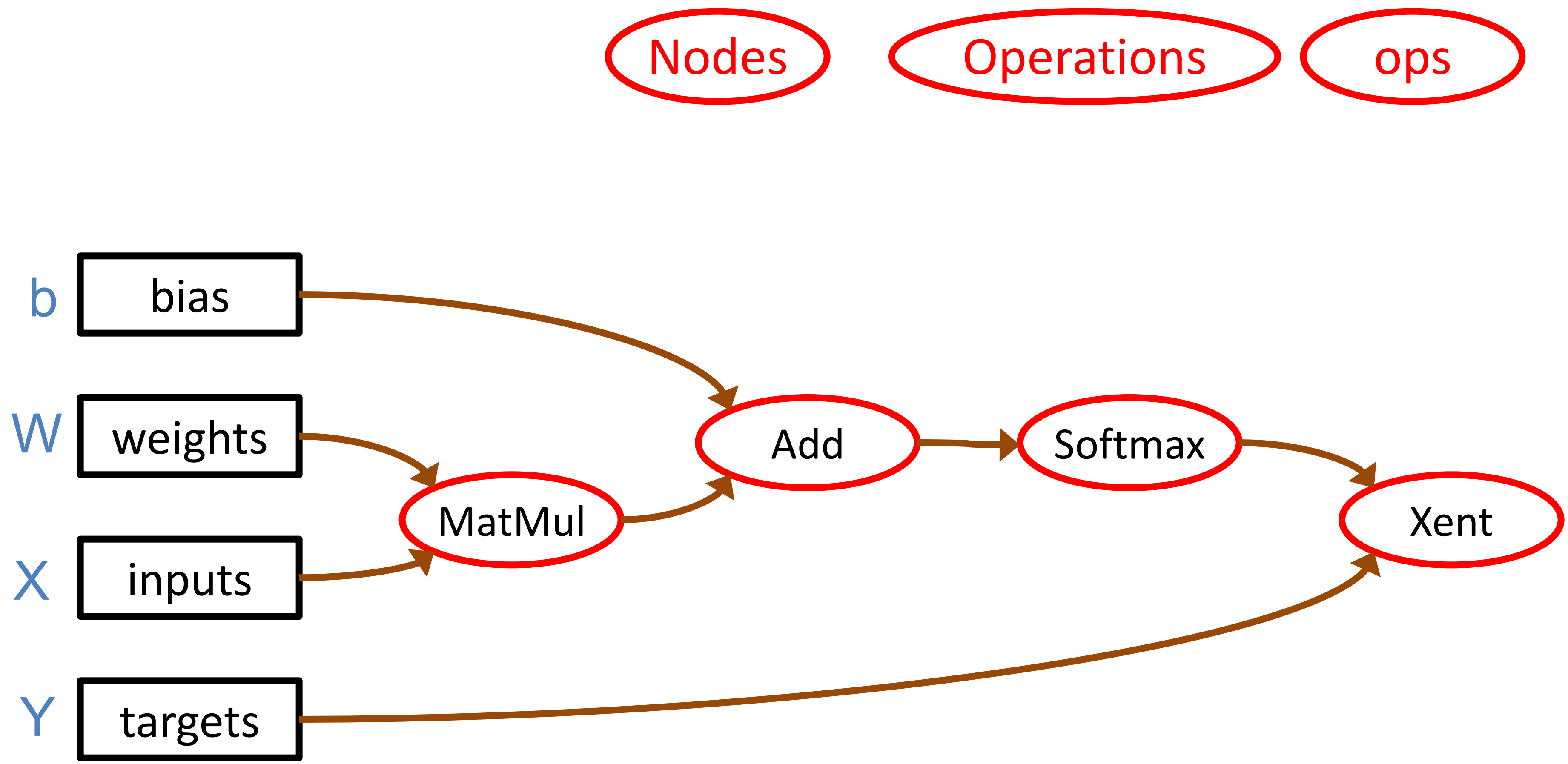
Computation is a Dataflow Graph

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

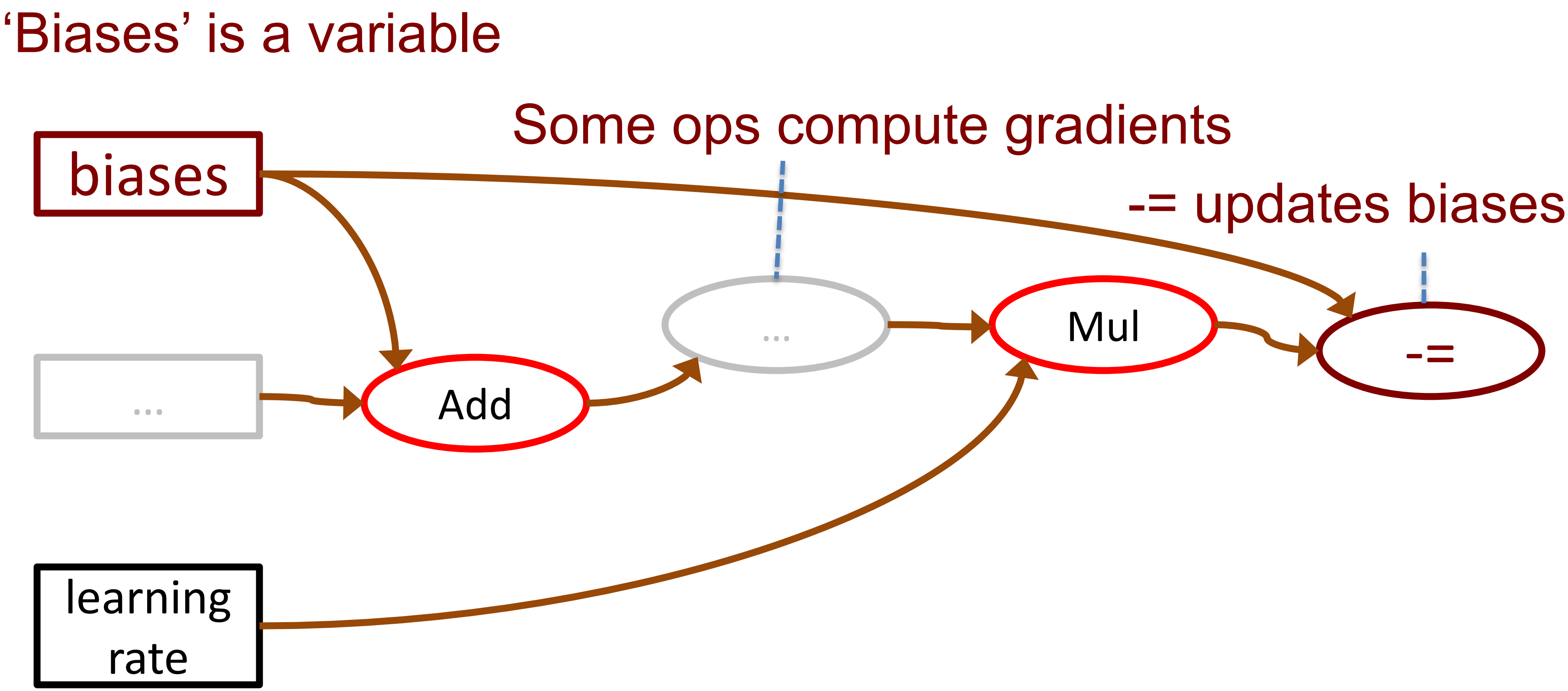


Edges are N-dimensional arrays: **Tensors**

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

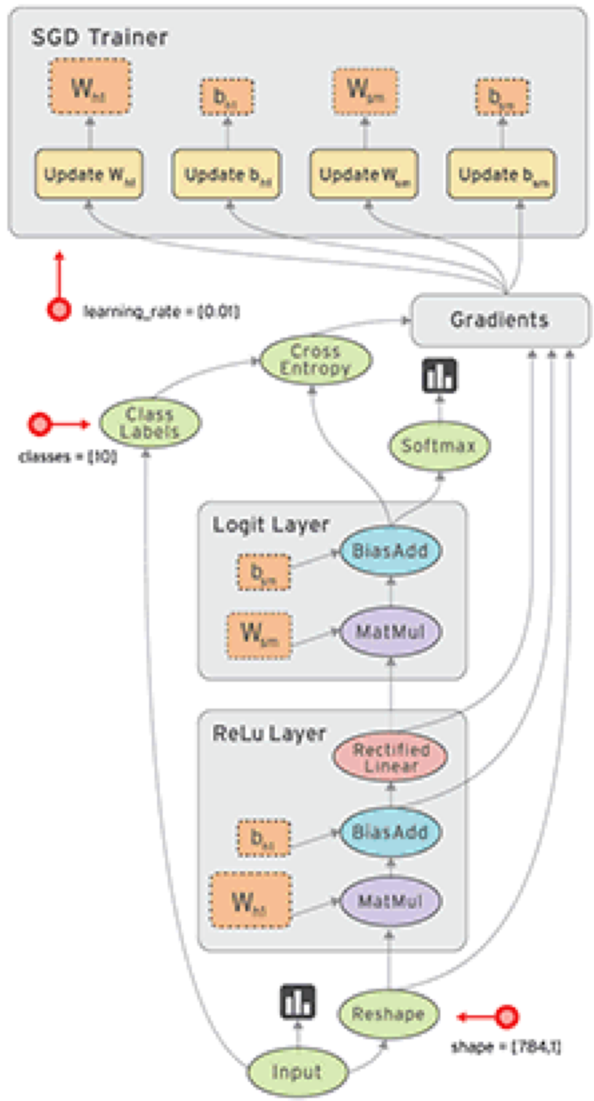
Computation is a Dataflow Graph

with state



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

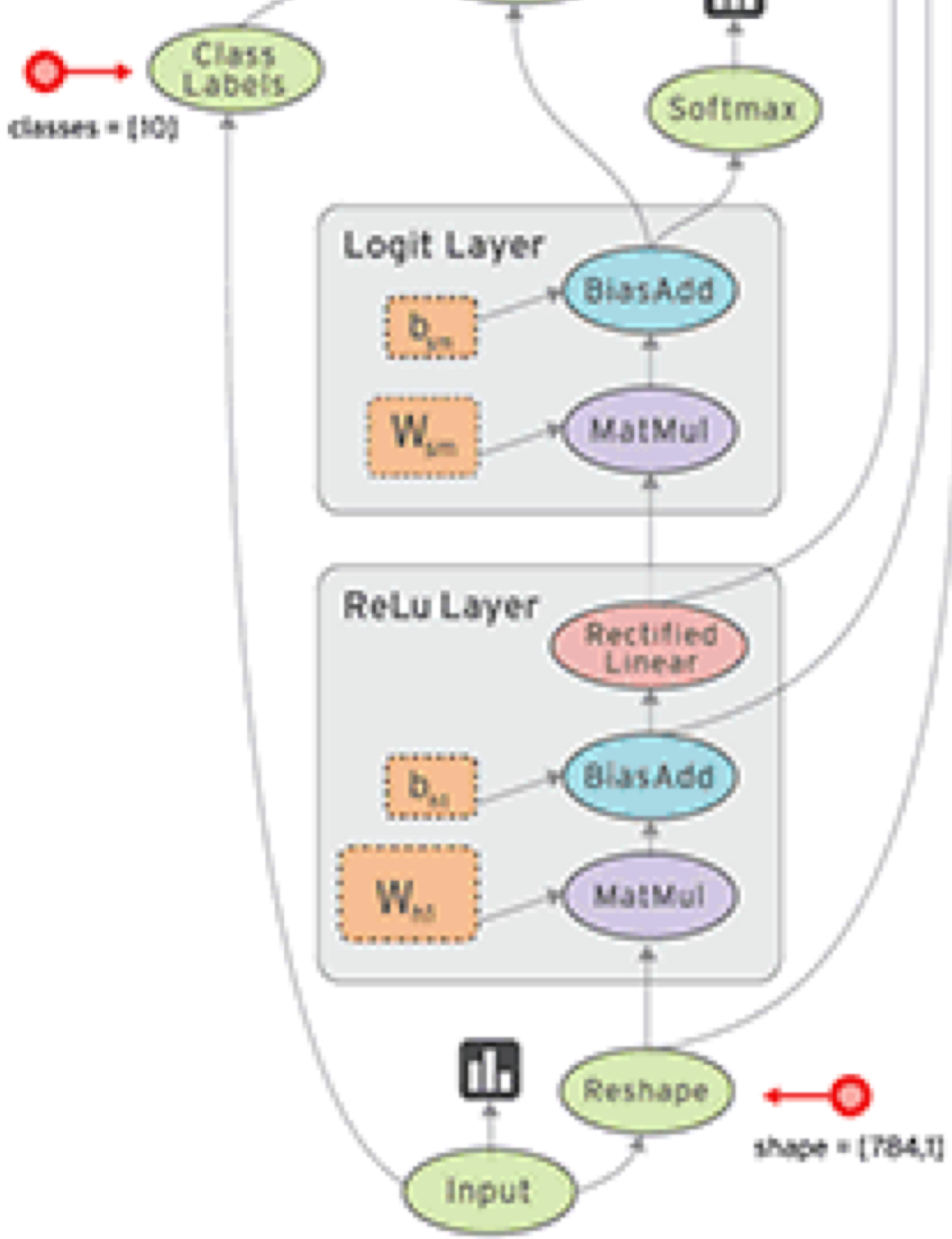
Data Flow Graph



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

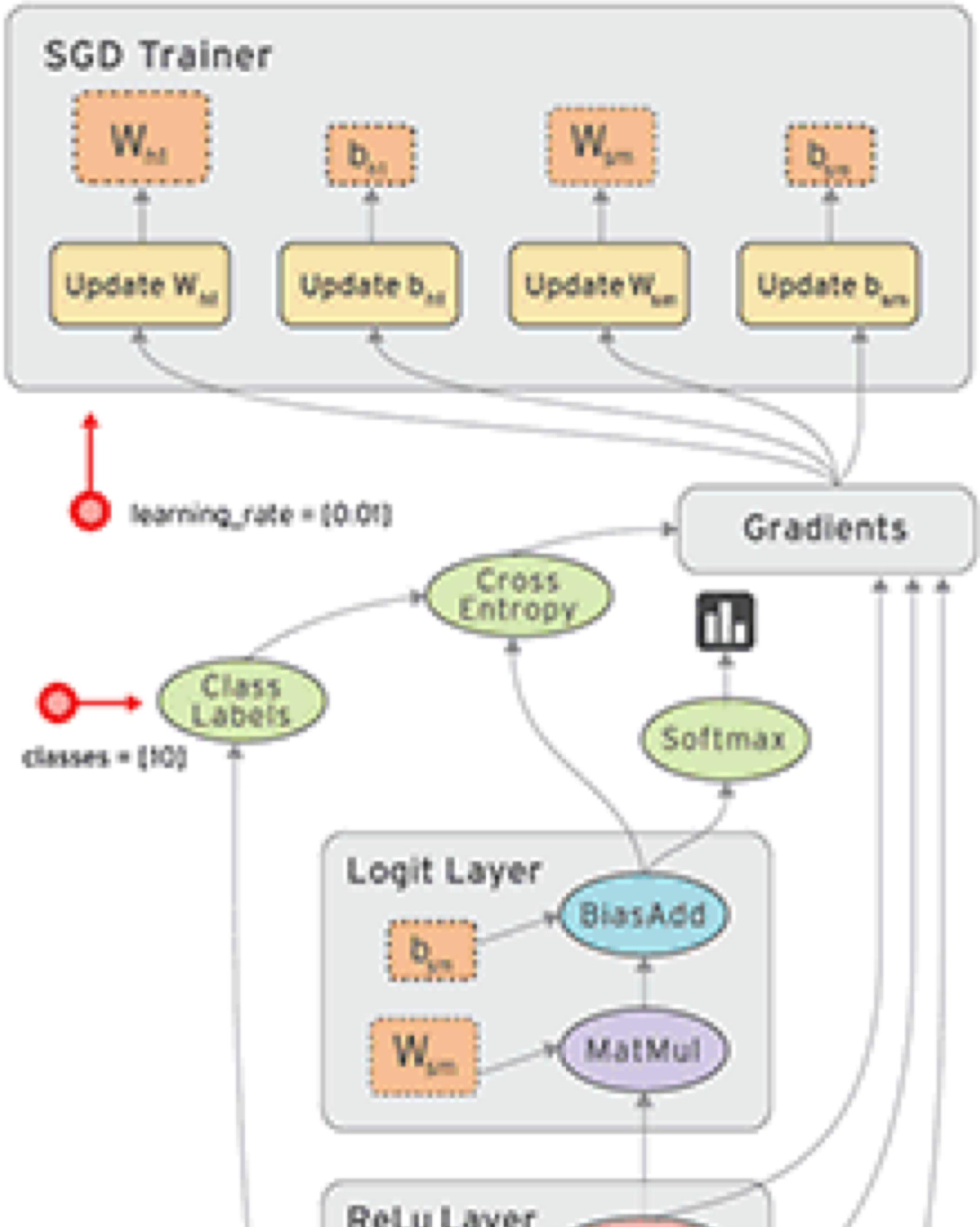
TensorFlow

Data Flow Graph



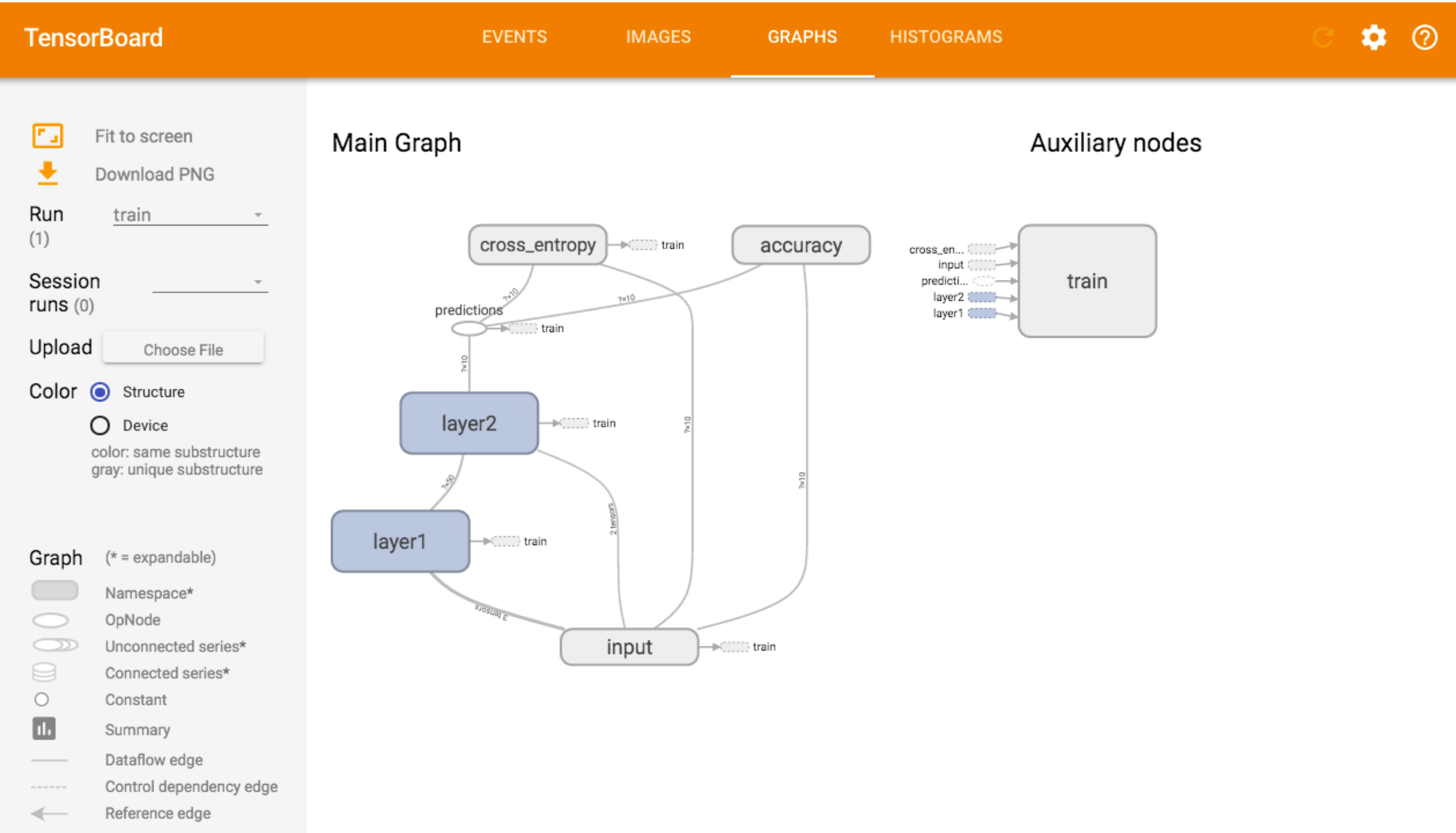
Source: https://www.tensorflow.org/

Data Flow Graph



TensorFlow

TensorBoard



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

Getting Started with TensorFlow

TensorFlow ™ Install Develop API r1.4 Deploy Extend Community Versions Search GITHUB

Develop

GET STARTED PROGRAMMER'S GUIDE TUTORIALS PERFORMANCE MOBILE

- Getting Started
- Getting Started With TensorFlow
- MNIST For ML Beginners
- Deep MNIST for Experts
- TensorFlow Mechanics 101
- tf.estimator Quickstart
- Building Input Functions with tf. estimator
- TensorBoard: Visualizing Learning
- TensorBoard: Graph Visualization
- TensorBoard Histogram Dashboard
- TensorFlow Versions

Getting Started With TensorFlow

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

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

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

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

Contents

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

Try your first TensorFlow

```
$ python

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

https://github.com/tensorflow/tensorflow

Hello TensorFlow

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

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

https://github.com/tensorflow/tensorflow

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

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

```
42
```

https://github.com/tensorflow/tensorflow

Linear Regression Model

```
import tensorflow as tf

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

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

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

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

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

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

tf.estimator

```
import numpy as np
import tensorflow as tf

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

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

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

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

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

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

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

Deep Learning for Financial Market Prediction

Deep Learning
for
Financial Market Prediction
Stock Market Prediction
Stock Price Prediction
Time Series Prediction

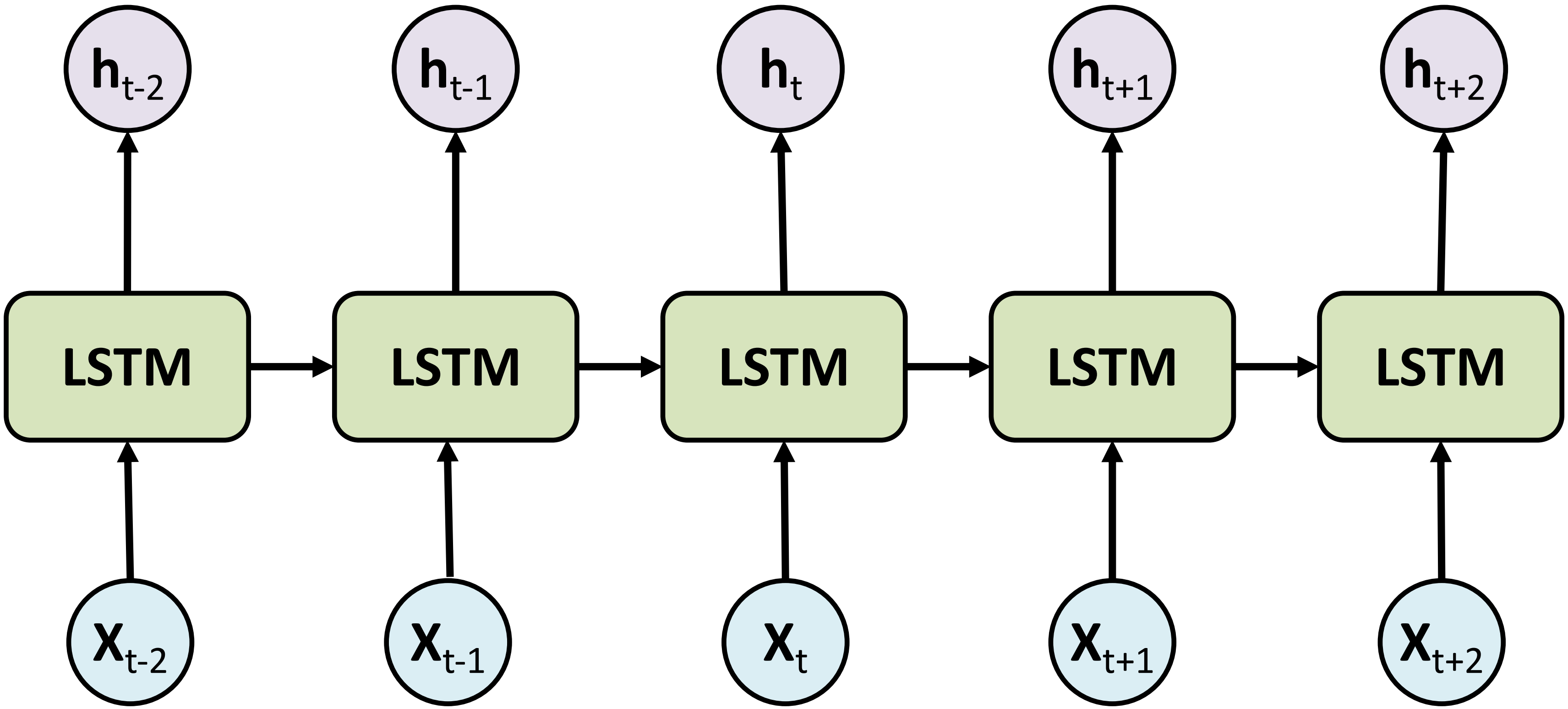
Time Series Data

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

[10, 20, 30, 40, 50, 60, 70, 80, 90]

X	Y
[10 20 30]	40
[20 30 40]	50
[30 40 50]	60
[40 50 60]	70
[50 60 70]	80
[60 70 80]	90

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

```python
# univariate data preparation
from numpy import array
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# summarize the data
for i in range(len(X)):
    print(X[i], y[i])
```

```
[10 20 30] 40
[20 30 40] 50
[30 40 50] 60
[40 50 60] 70
[50 60 70] 80
[60 70 80] 90
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

CO Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECTED EDITING

LSTM for Time Series Forecasting

```
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline

# define dataset
X = array([[100, 110, 120], [110, 120, 130], [120, 130, 140], [130, 140, 150], [140, 150, 160]])
y = array([130, 140, 150, 160, 170])
# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=2000, verbose=0)
# demonstrate prediction
x_input = array([150, 160, 170])
x_input = x_input.reshape((1, 3, 1))
yhat = model.predict(x_input, verbose=0)
print('yhat', yhat)
print(model.summary())
# list all data in history
print(history.history.keys())
# summarize history for loss
print('loss:', '%f'%history.history['loss'][-1])
print('loss:', history.history['loss'][-1])
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

```
yhat [[181.34615]]
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

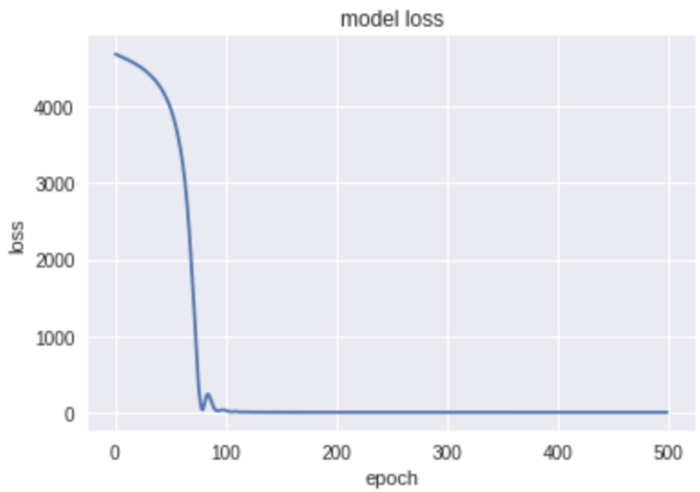
```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=500, verbose=0)
# demonstrate prediction
x_input = array([70, 80, 90])
x_input = x_input.reshape((1, n_steps, n_features))
yhat = model.predict(x_input, verbose=0)
print(yhat)
print('yhat', yhat)
print(model.summary())
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

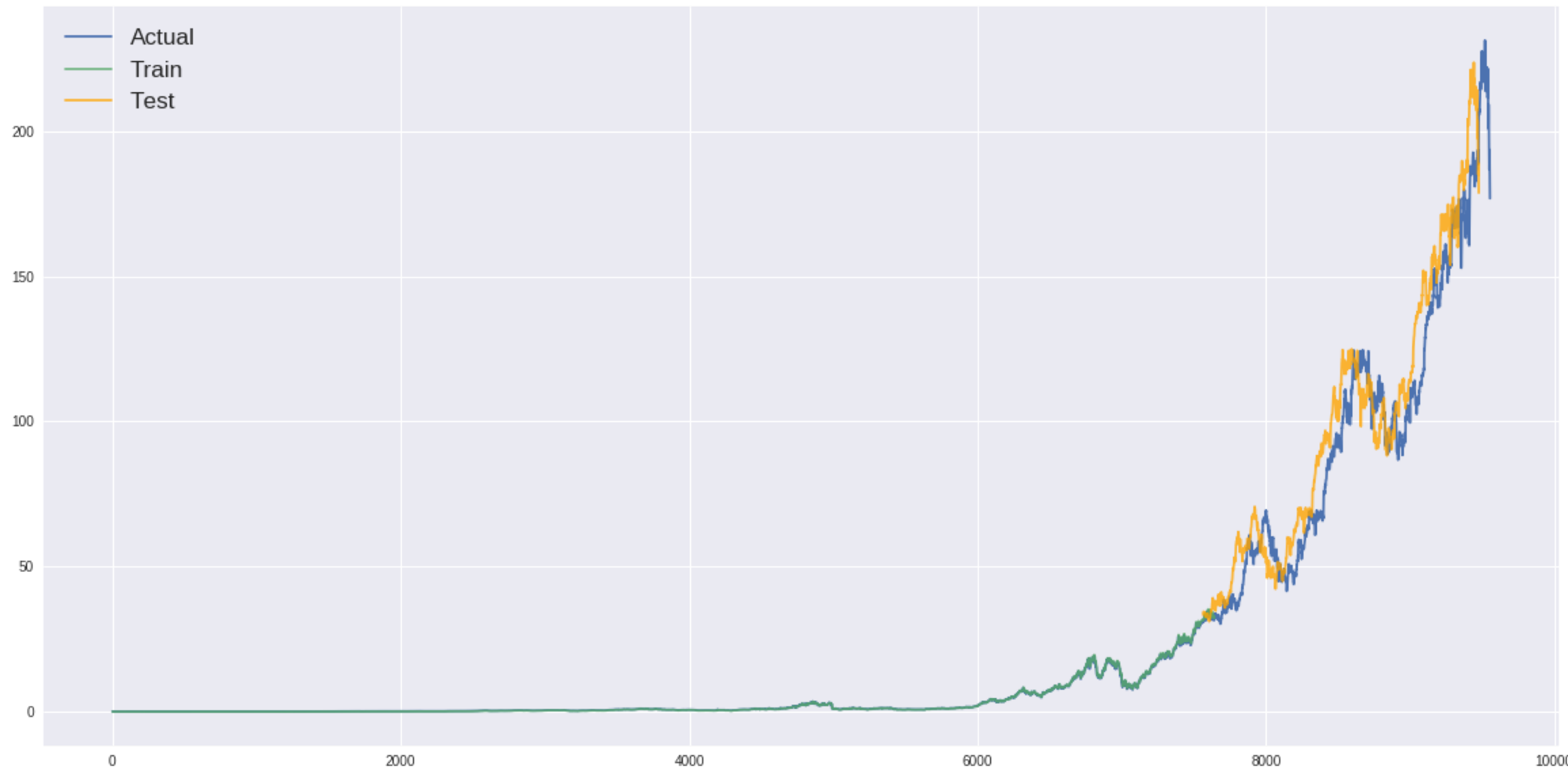
```
Using TensorFlow backend.
[[102.31296]]
yhat [[102.31296]]

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
None
dict_keys(['loss'])
loss: 0.000000
loss: 1.2578432517784677e-07
```



Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

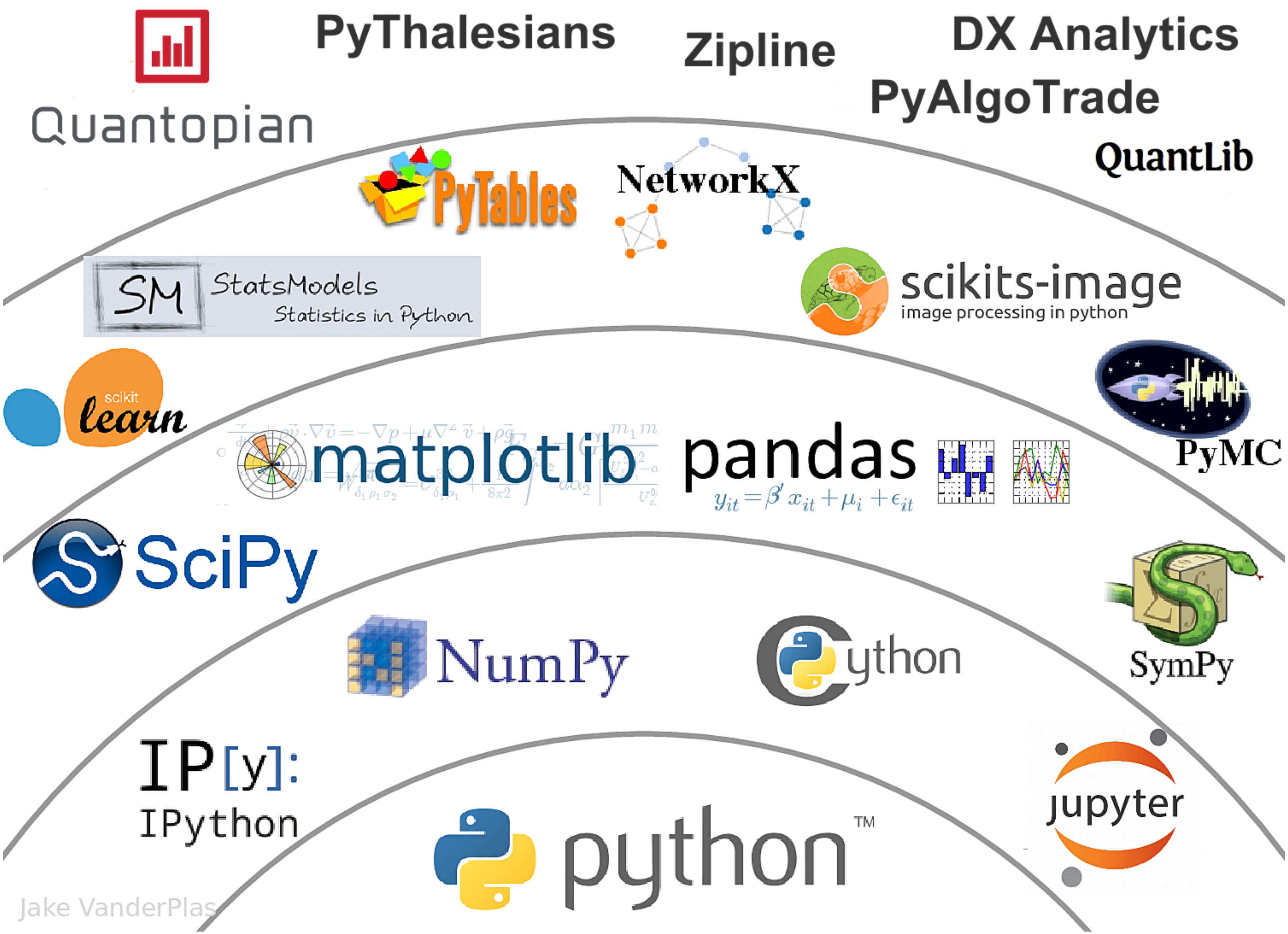
Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM



Source: https://github.com/yash-1337/AAPL_LSTM_Stock_Predictor/blob/master/AAPL_daily_LSTM_stock_predictor.ipynb

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

AI + VDI POC

AI + VDI POS TensorFlow Models

- M1: Basic Classification (Image Classification) (65 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb
- M2: Basic Text Classification (Text Classification) (46 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb
- M3: Basic Regression (Predict House Prices) (43 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb
- M4: Pix2Pix Eager (Option) (7-8 Hours)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/pix2pix/pix2pix_eager.ipynb
- M5. NMT with Attention (Option) (20-30 minutes)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Basic Classification

Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcutlzNHjRSLbeVl4kd5z

CO tf01_basic_classification.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents | Code snippets | Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Train your first neural network: basic classification

View on TensorFlow.org | Run in Google Colab | View source on GitHub

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format
printm()
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb

Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror

tf02_basic-text-classification.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

Copyright 2018 The TensorFlow Authors.

2 cells hidden

Text classification with movie reviews

View on TensorFlow.org Run in Google Colab View source on GitHub

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the MLCC Text Classification Guide.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdlj

CO tf03_basic-regression.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Predict house prices: regression

View on TensorFlow.org Run in Google Colab View source on GitHub

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the `tf.keras` API, see this guide for details.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format(gpu.memo
```

AI+VDI POC
ISAC+TKU Test

- AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  - https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxIKbGwJvXlrjtc
- run3models.ipynb
  - https://colab.research.google.com/drive/1HQ1GrIqQUUPCct7_AVgoMwMrh0UqMm0f
- AI+VDI POC ISAC+TKU Test Report (Example)
  - https://docs.google.com/spreadsheets/d/1meMwqn15PSuTk6d5TgendDpdDX6L3OfHM4E0SIkq1Zk/edit?usp=sharing

Summary

- Deep Learning for Finance Big Data with TensorFlow
  - Deep Learning
  - Finance Big Data
  - TensorFlow

References

- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.
 https://github.com/wesm/pydata-book
- Avinash Jain (2017), Introduction To Python Programming, Udemy, https://www.udemy.com/pythonforbeginnersintro/
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
- James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
- Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- Jason Brownlee (2018), How to Develop LSTM Models for Time Series Forecasting, https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow