財務金融大數據分析
Big Data Analytics in Finance

Tamkang University
淡江大學

Python Keras 深度學習
(Deep Learning with Keras in Python)

1061BDAF09
MIS EMBA (M2322) (8605)
Thu 12,13,14 (19:20-22:10) (D503)

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/

2017-11-30

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

1 2017/09/21 財務金融大數據分析課程介紹
(Course Orientation for Big Data Analytics in Finance)

2 2017/09/28 金融科技商業模式 (Business Models of Fintech)

3 2017/10/05 人工智慧投資分析與機器人理財顧問
(Artificial Intelligence for Investment Analysis and
Robo-Advisors)

4 2017/10/12 金融科技對話式商務與智慧型交談機器人
(Conversational Commerce and
Intelligent Chatbots for Fintech)

5 2017/10/19 事件研究法 (Event Study)

6 2017/10/26 財務金融大數據分析個案研究 I
(Case Study on Big Data Analytics in Finance I)

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

7 2017/11/02 Python 財務大數據分析基礎
(Foundations of Finance Big Data Analytics in Python)

8 2017/11/09 Python Numpy大數據分析
(Big Data Analytics with Numpy in Python)

9 2017/11/16 Python Pandas 財務大數據分析
(Finance Big Data Analytics with Pandas in Python)

10 2017/11/23 期中報告 (Midterm Project Report)

11 2017/11/30 Python Keras深度學習
(Deep Learning with Keras in Python)

12 2017/12/07 文字探勘分析技術與自然語言處理
(Text Mining Techniques and
Natural Language Processing)
[Invited Speaker: Irene Chen, Consultant, Teradata]

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2017/12/14	財務金融大數據分析個案研究 II (Case Study on Big Data Analytics in Finance II)
14	2017/12/21	TensorFlow深度學習 (Deep Learning with TensorFlow)
15	2017/12/28	財務金融大數據深度學習 (Deep Learning for Finance Big Data)
16	2018/01/04	社會網絡分析 (Social Network Analysis)
17	2018/01/11	期末報告 I (Final Project Presentation I)
18	2018/01/18	期末報告 II (Final Project Presentation II)

Deep Learning

with

Keras

in

Python

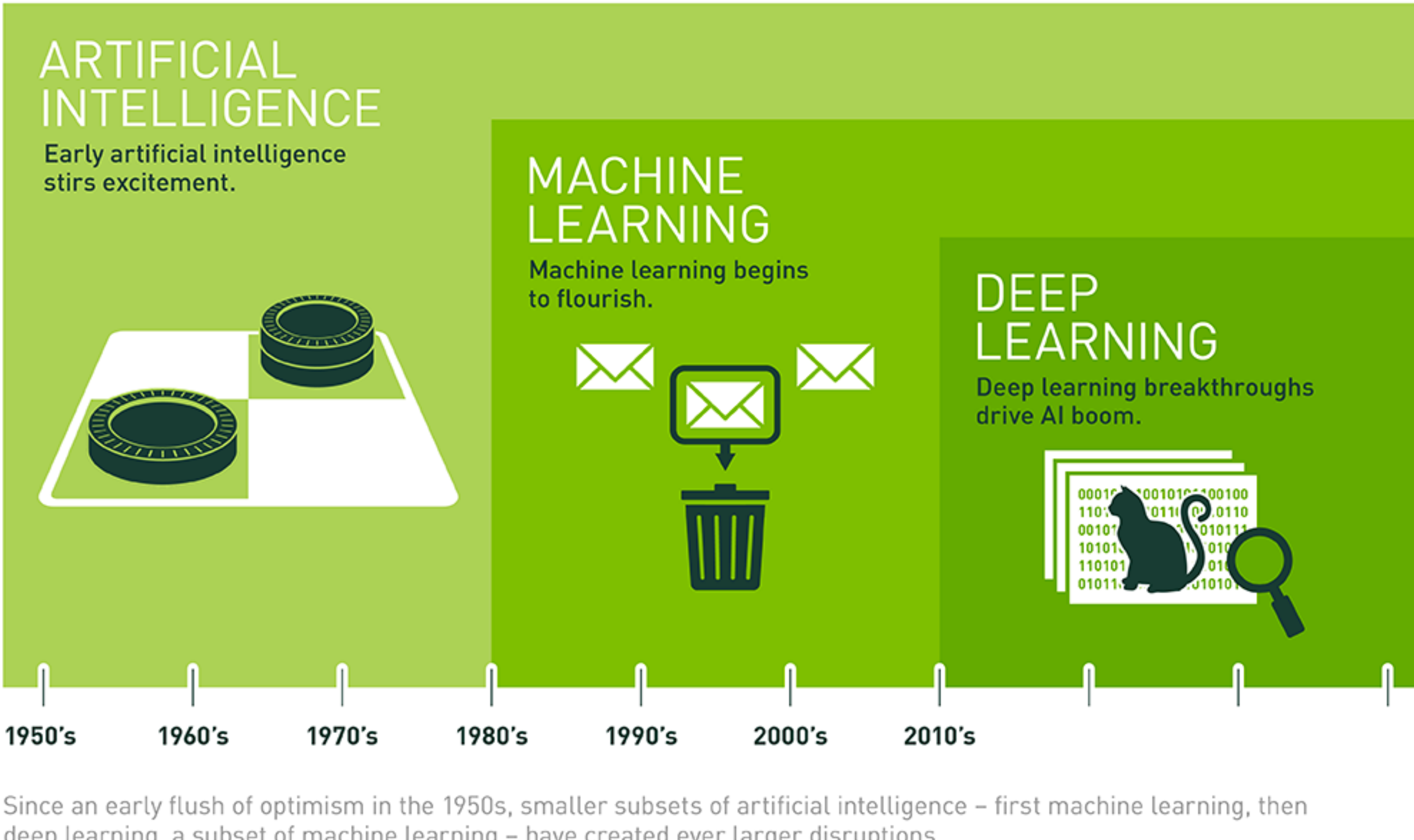
Keras + TensorFlow

Source: https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html

Outline

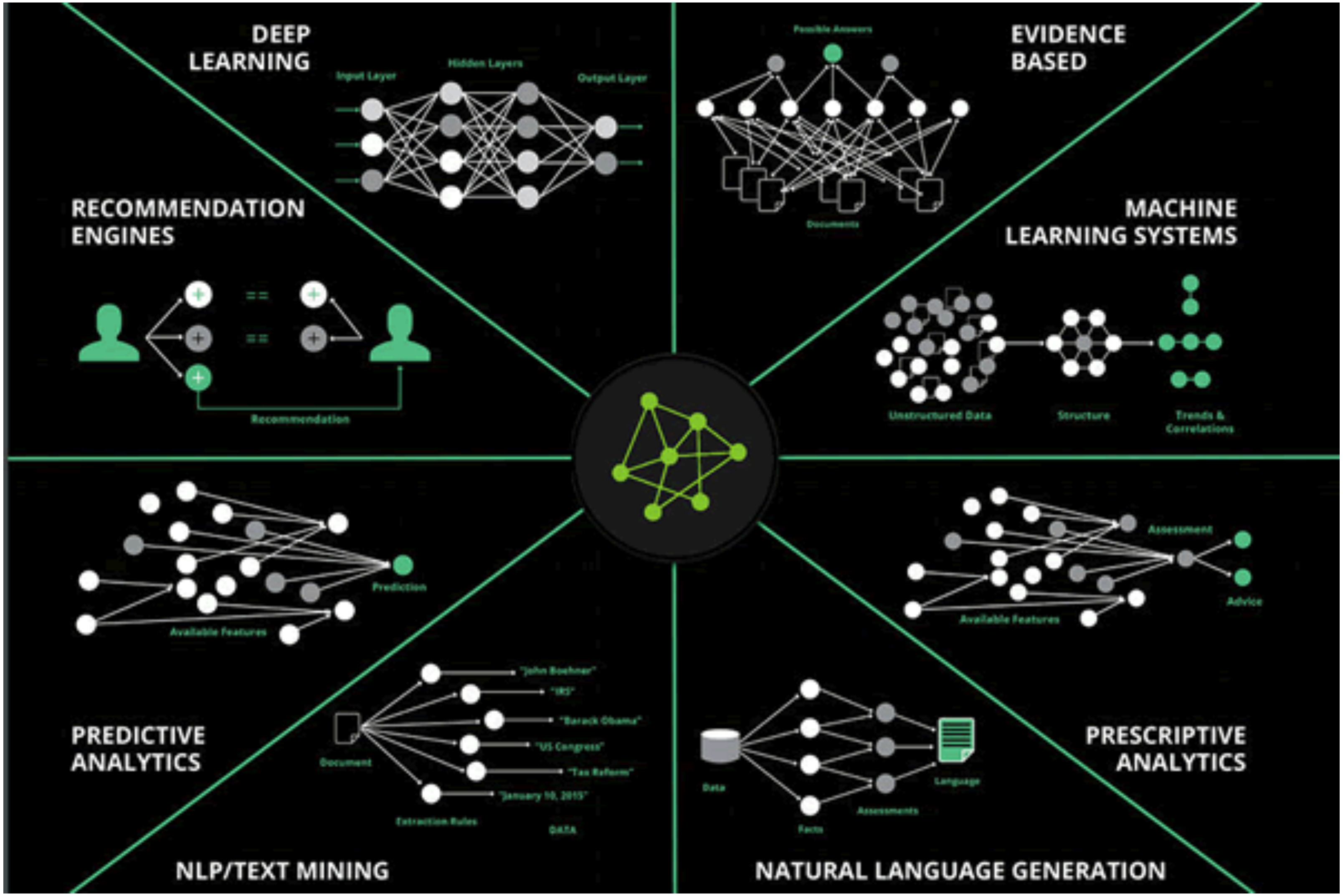
- AI, Machine Learning and Deep Learning
- Deep Learning Foundations: Neural Networks
- Keras: High-level API for TensorFlow

Artificial Intelligence
Machine Learning & Deep Learning



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

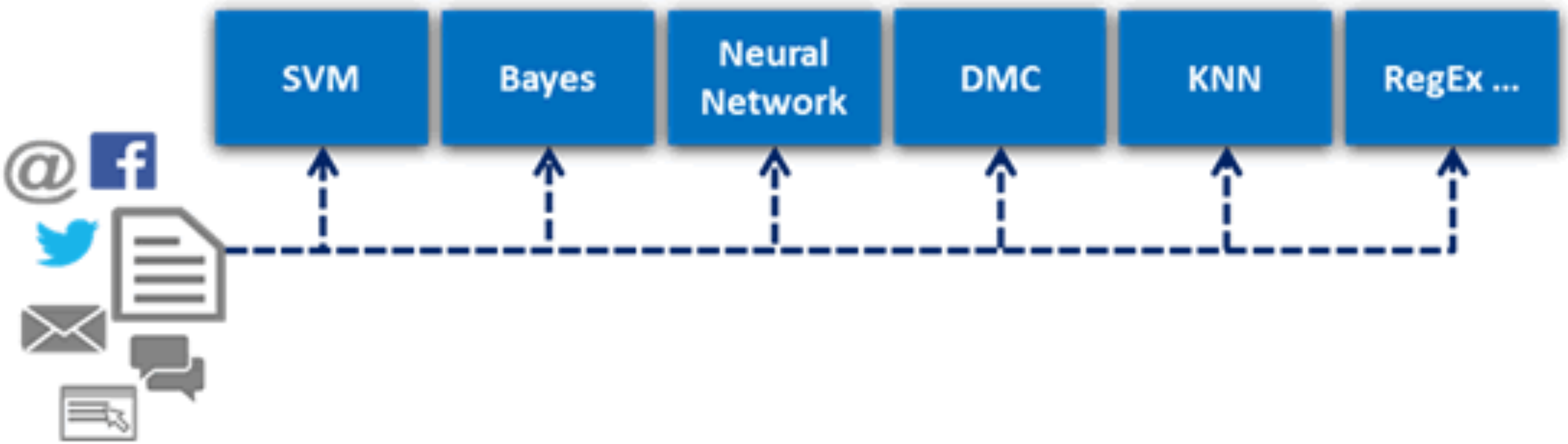
Artificial Intelligence (AI) is many things



Ecosystem of AI

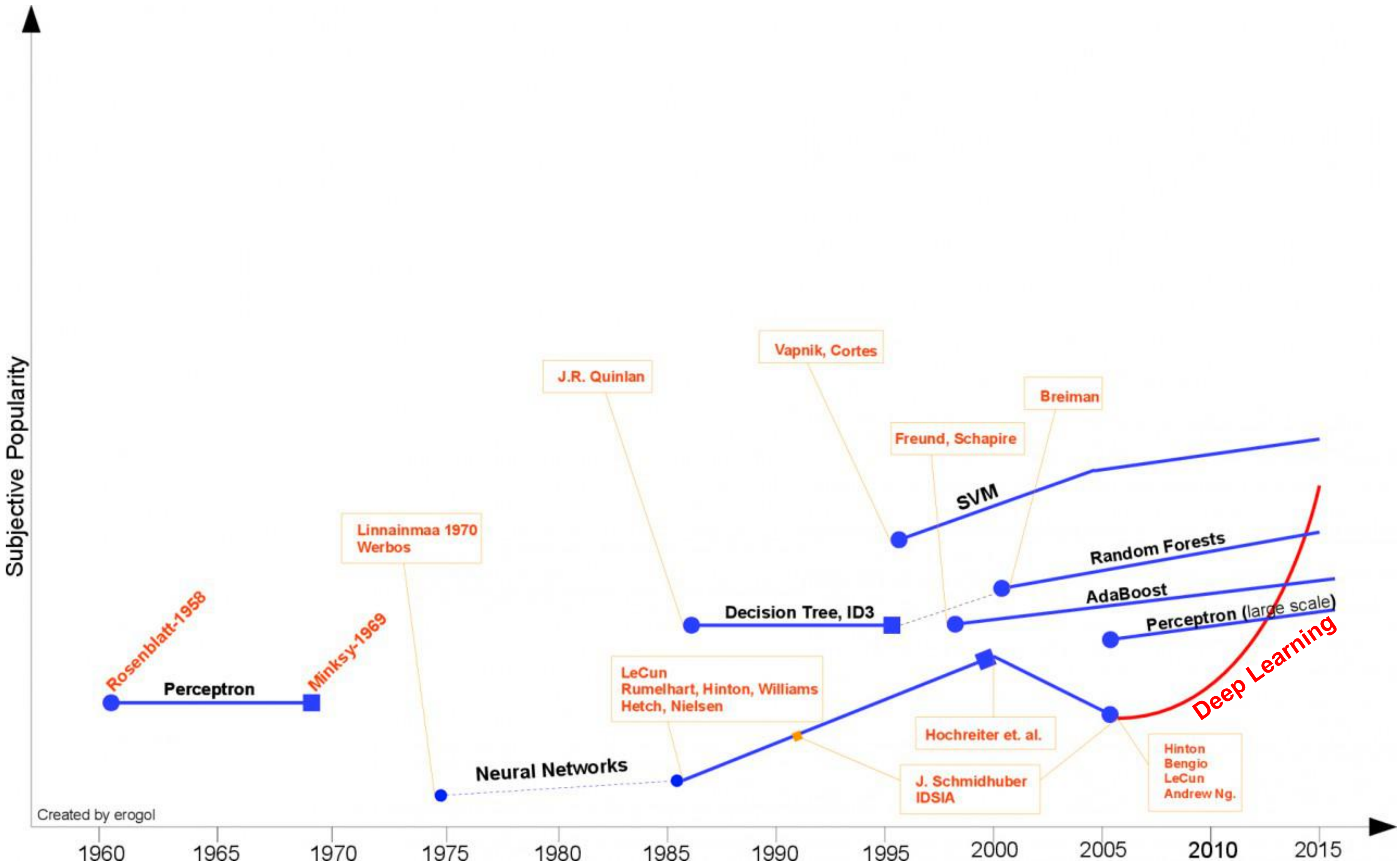
Source: https://www.i-scoop.eu/artificial-intelligence-cognitive-computing/

Artificial Intelligence (AI)
Intelligent Document Recognition algorithms



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

Deep Learning Evolution



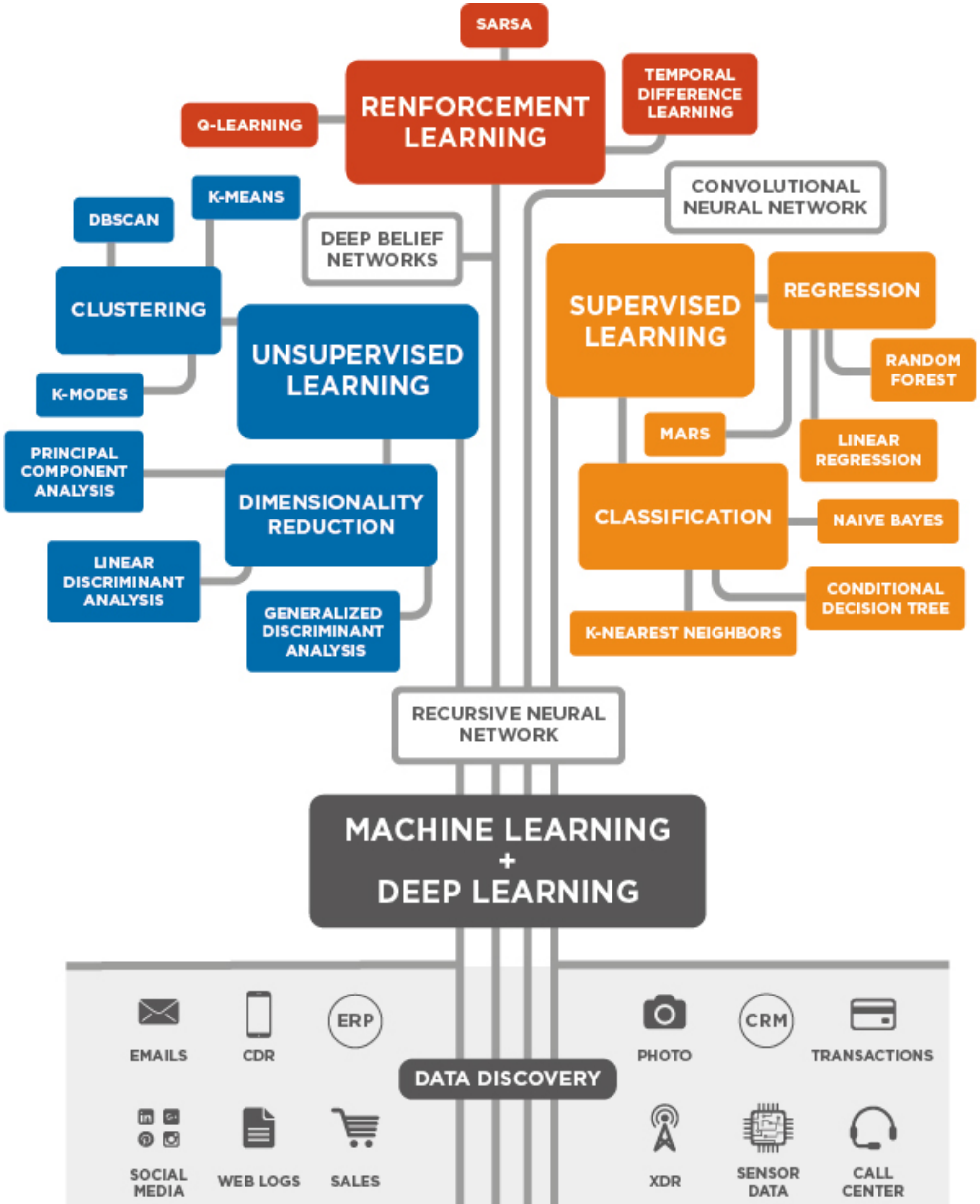
Source: http://www.erogol.com/brief-history-machine-learning/

Machine Learning Models

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

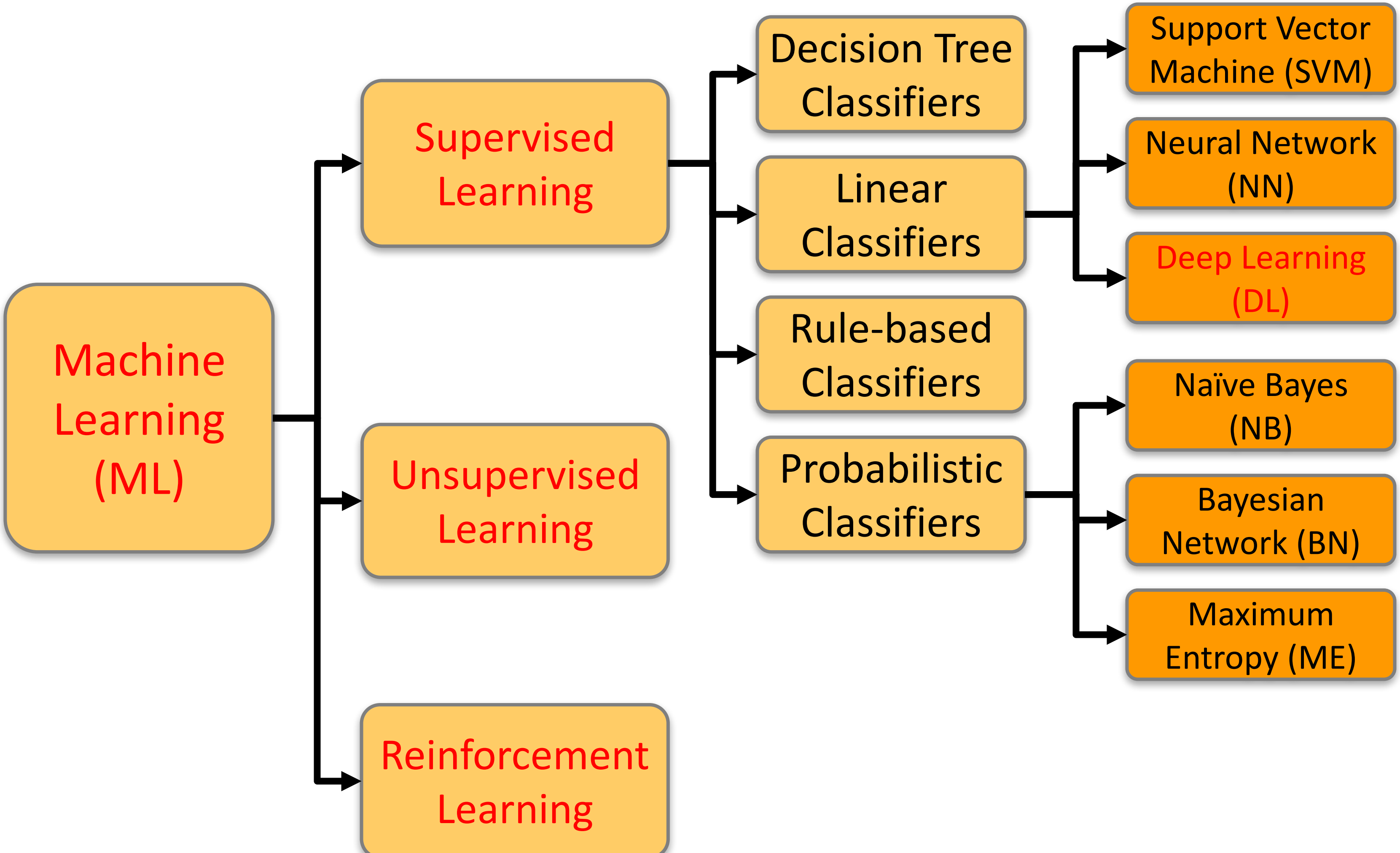
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

3 Machine Learning Algorithms



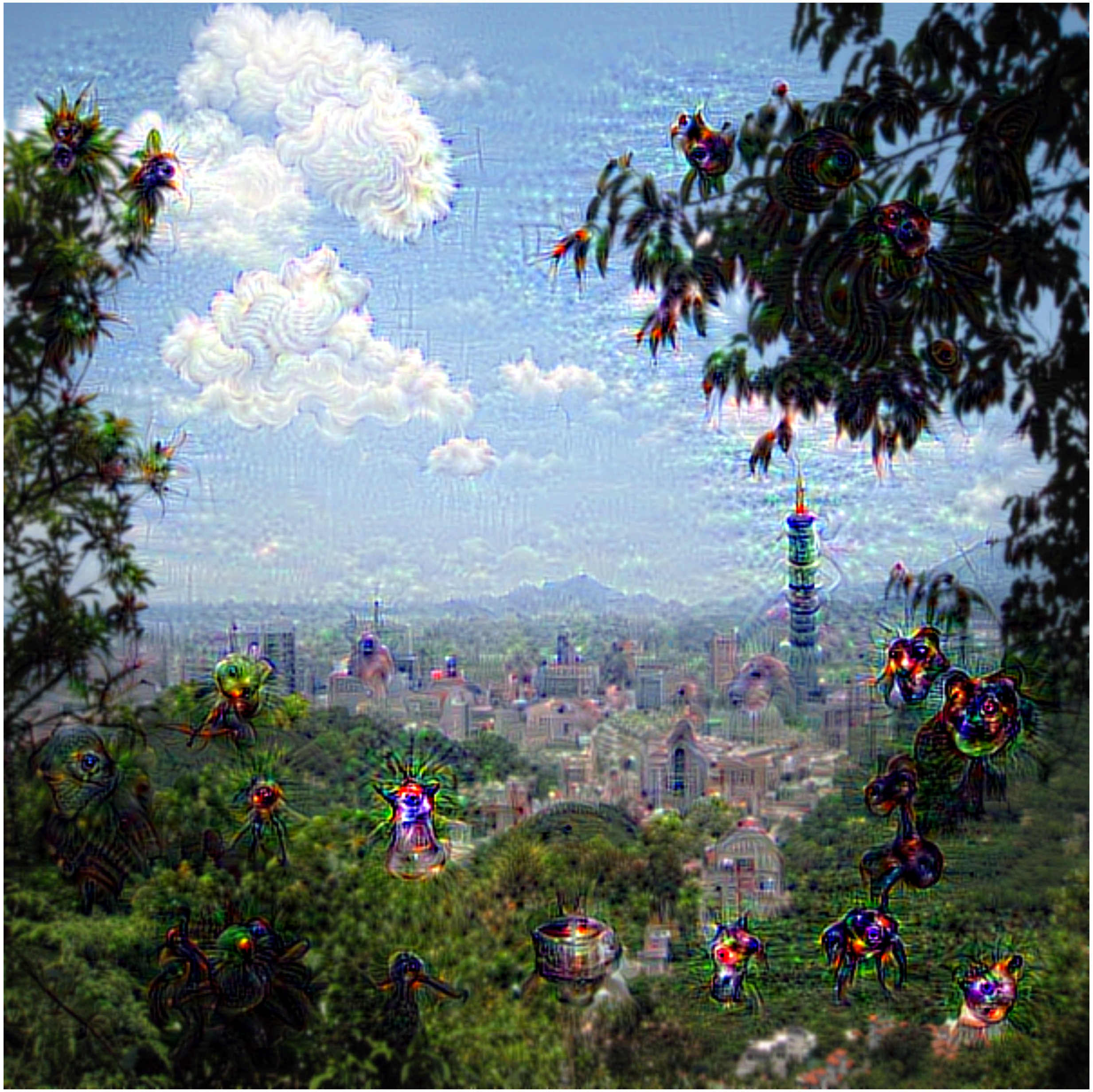
Source: Enrico Galimberti, http://blogs.teradata.com/data-points/tree-machine-learning-algorithms/

Machine Learning (ML) / Deep Learning (DL)



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

Deep Dream



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

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]

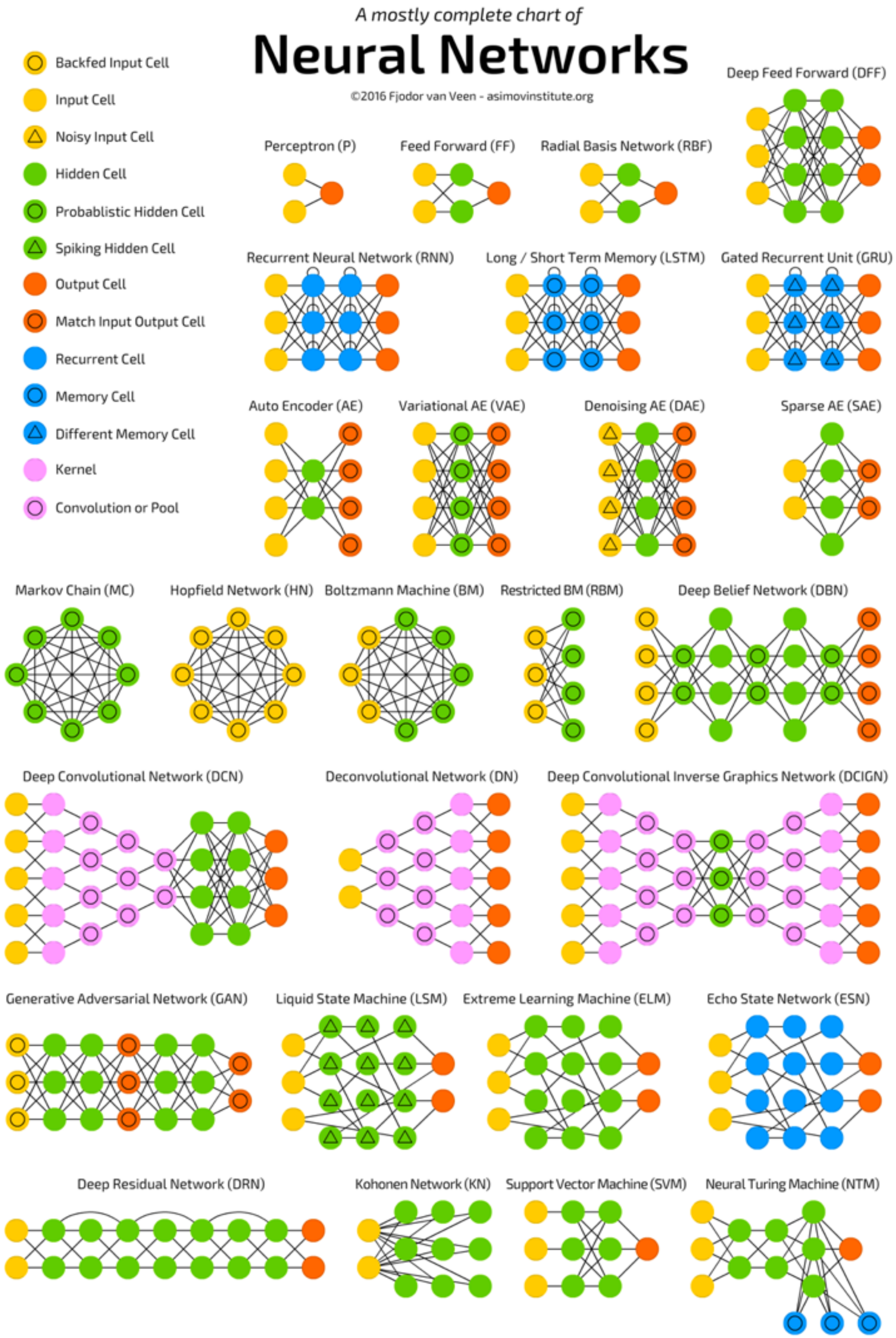
Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other 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.

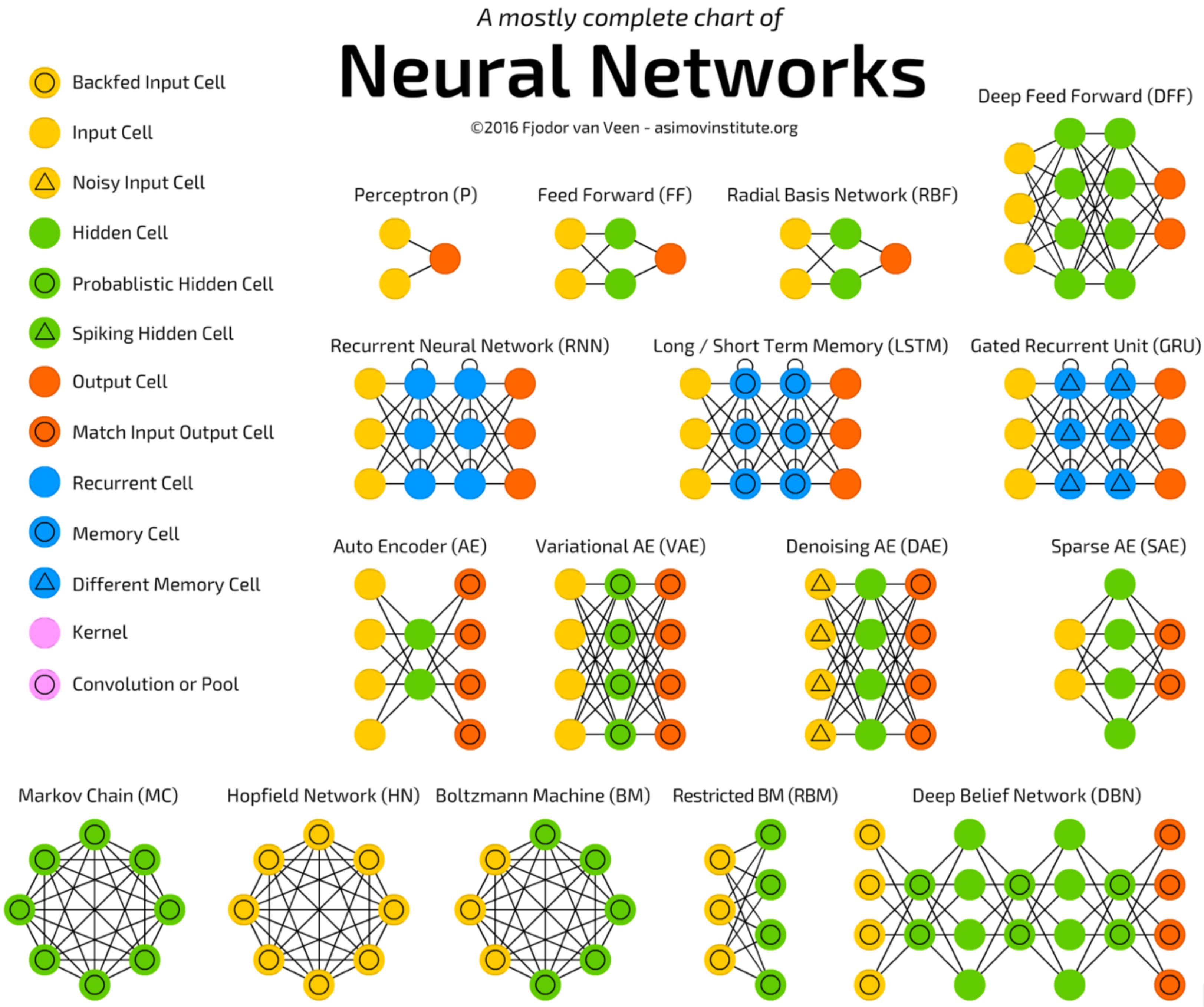
Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, con-

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].

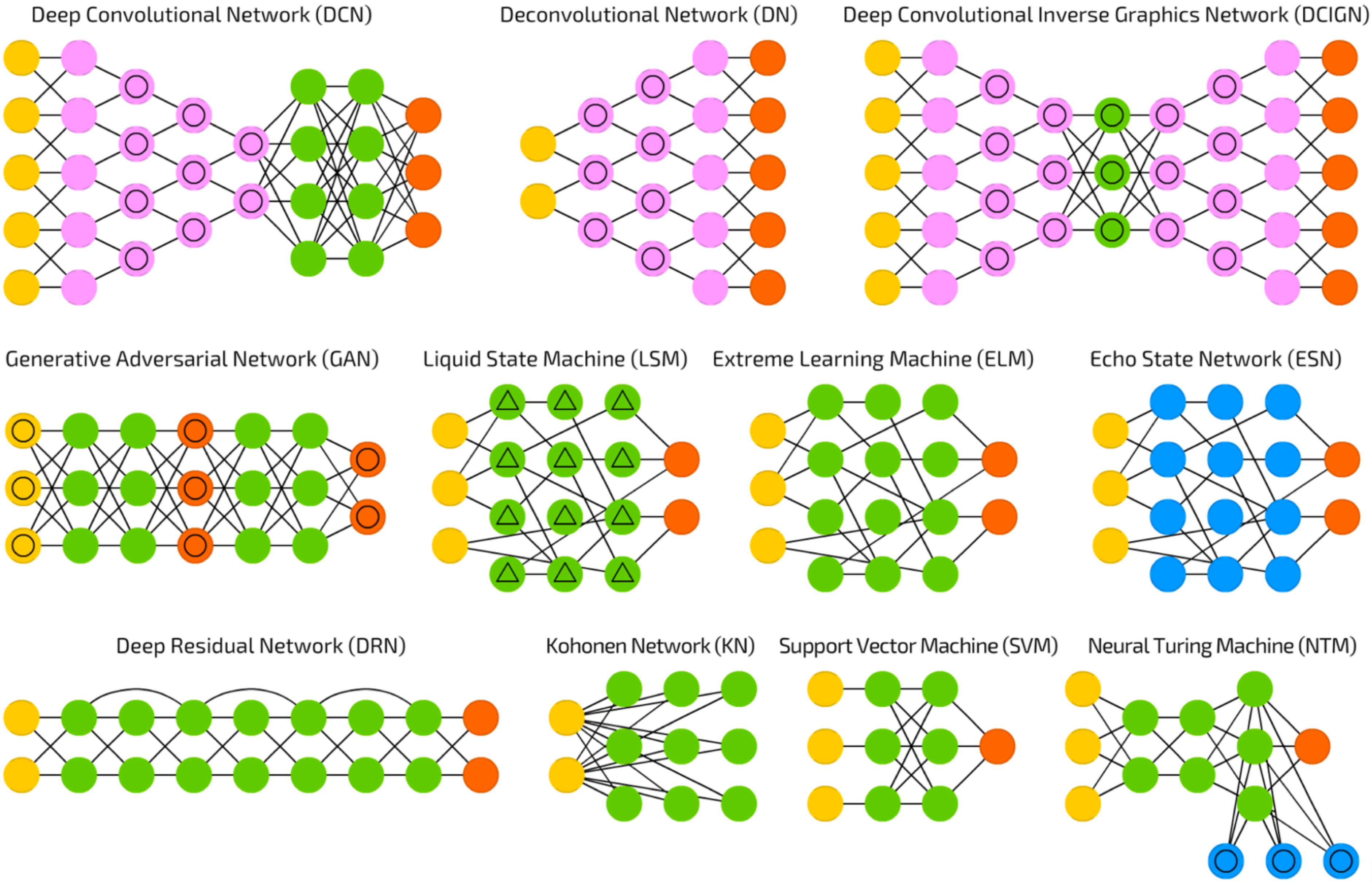
Neural Networks (NN)



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



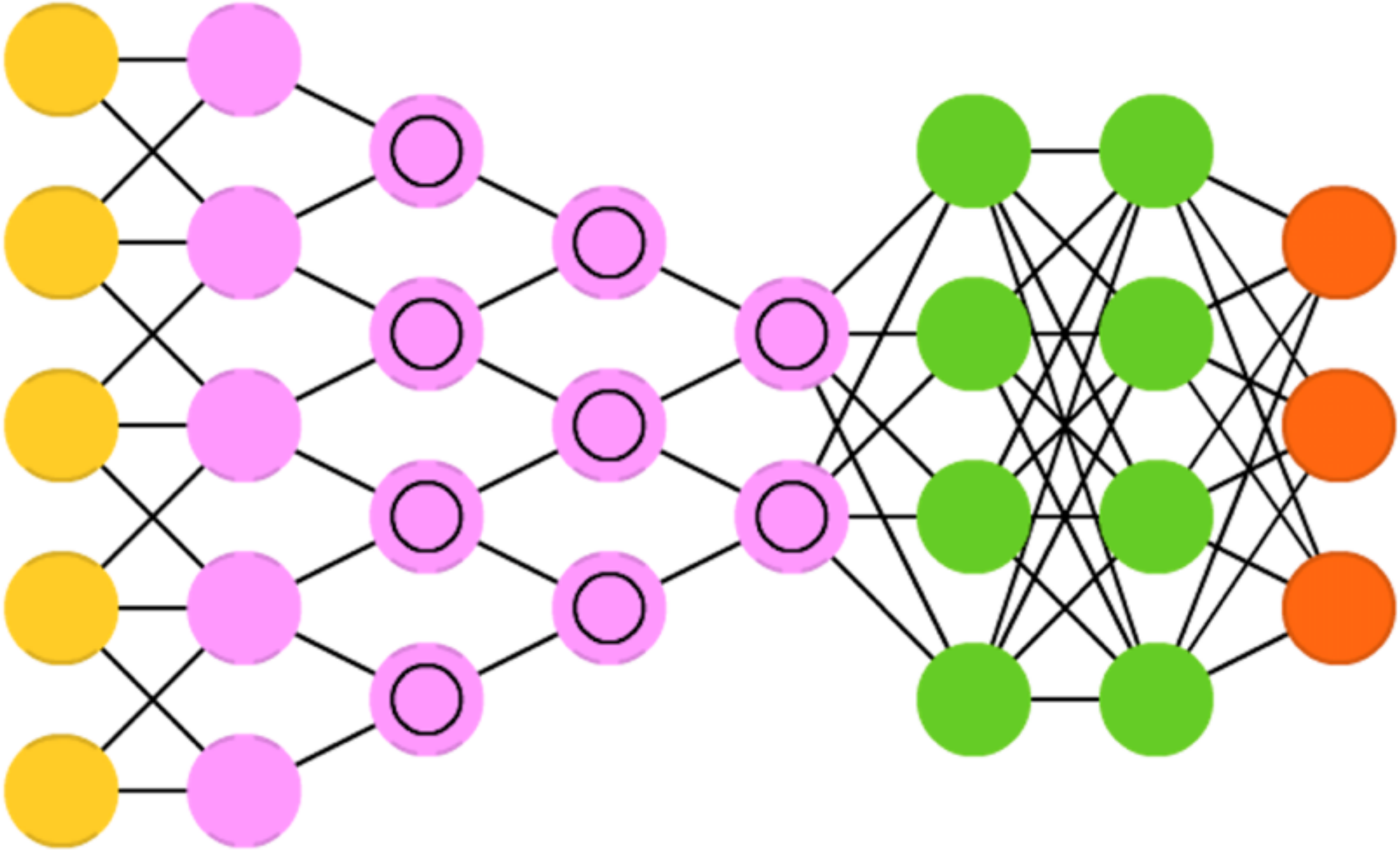
Source: http://www.asimovinstitute.org/neural-network-zoo/



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

Convolutional Neural Networks

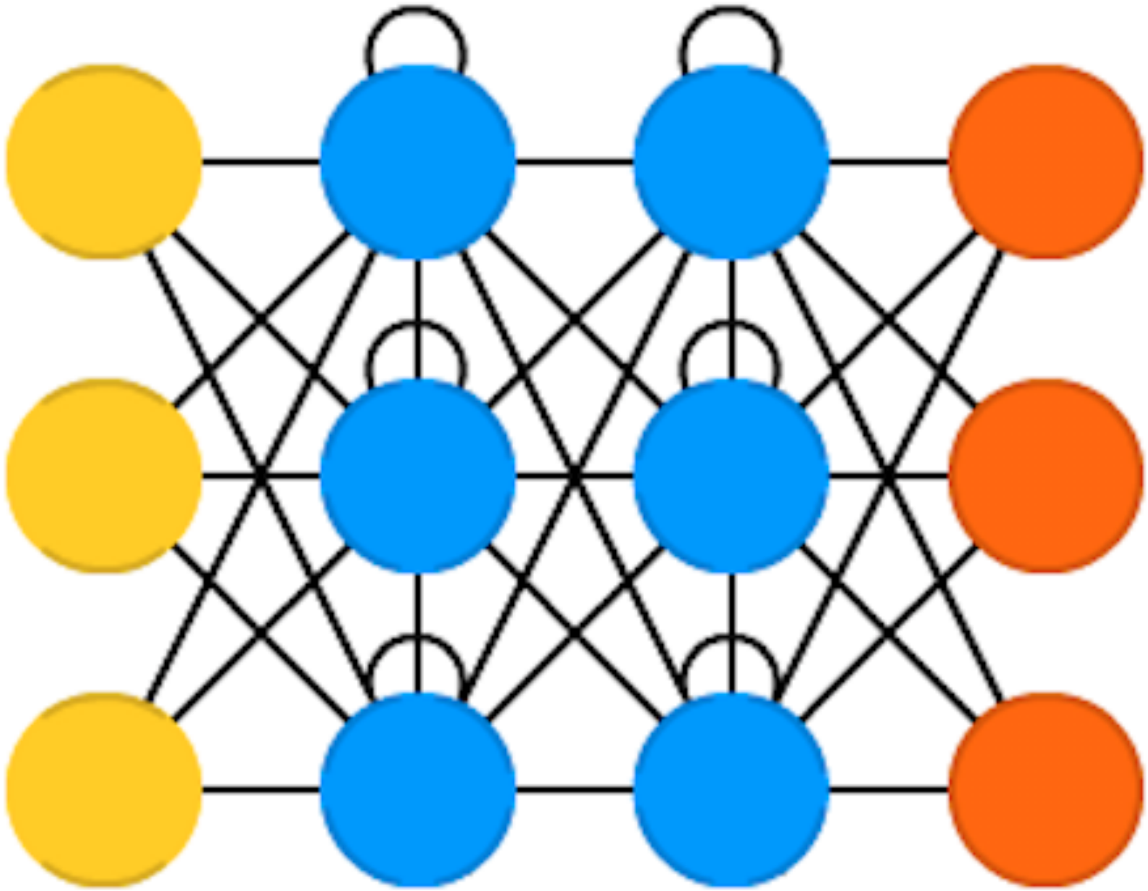
(CNN or Deep Convolutional Neural Networks, DCNN)



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

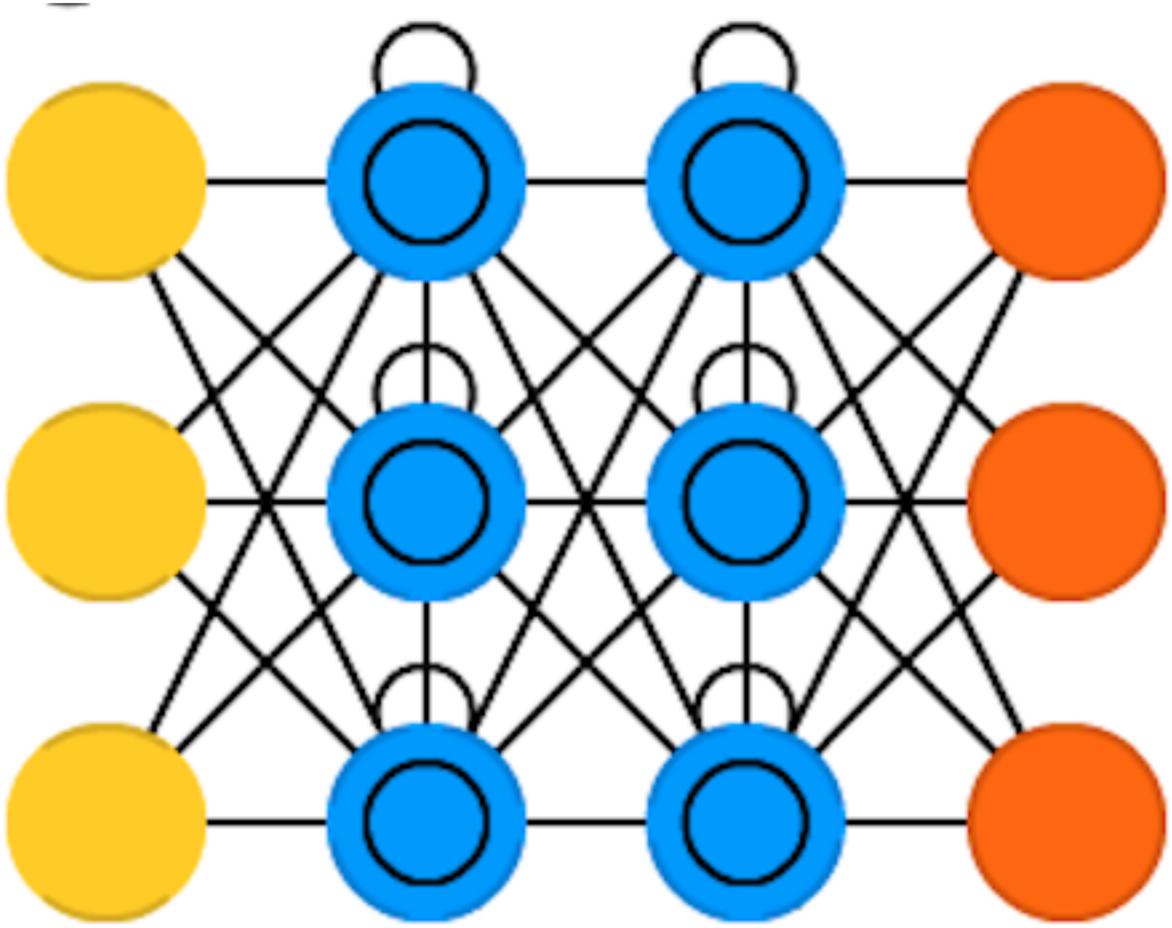
Source: http://www.asimovinstitute.org/neural-network-zoo/

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/

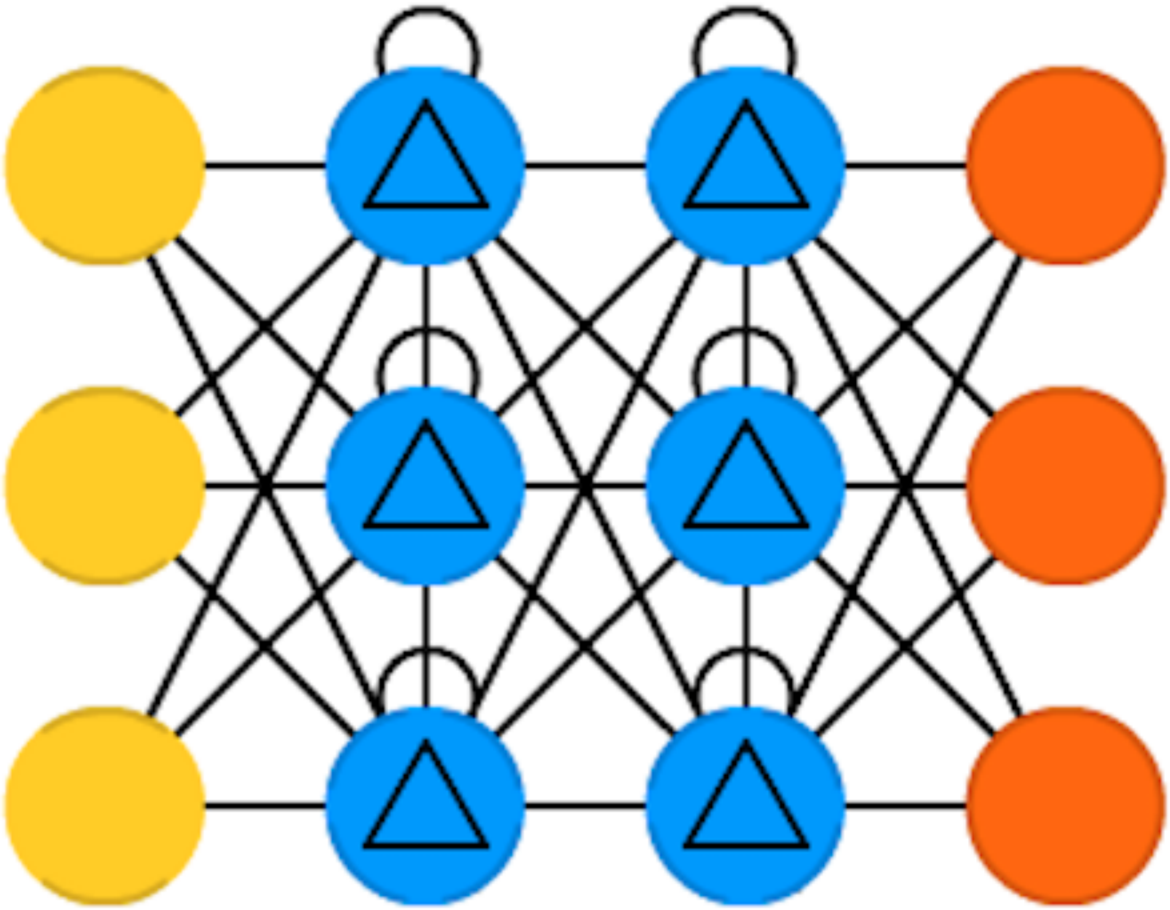
Long / Short Term Memory (LSTM)



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

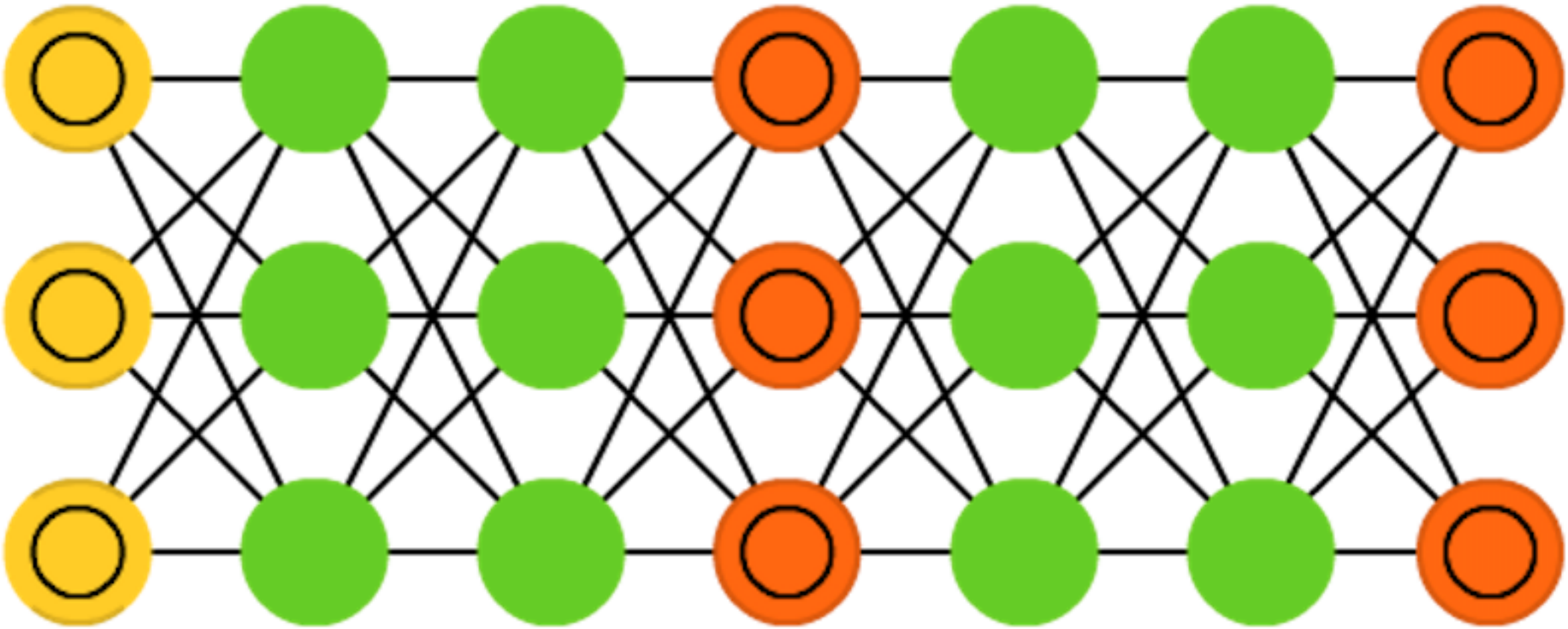
Source: http://www.asimovinstitute.org/neural-network-zoo/

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/

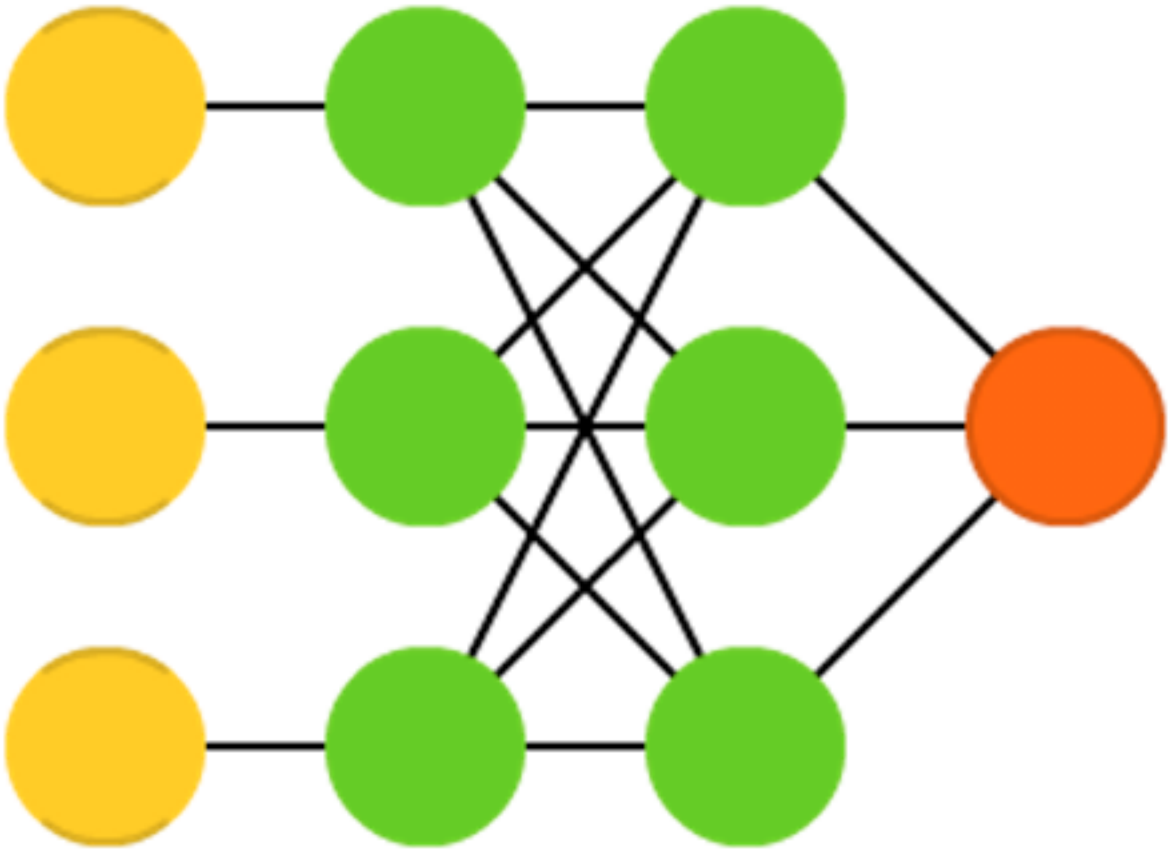
Generative Adversarial Networks (GAN)



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

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

Multilayer Perceptrons (MLP) 1985

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

Support Vector Machine (SVM) 1995

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

Hinton presents the

Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST

2005

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

Deep
Recurrent Neural Network
(RNN)
2009

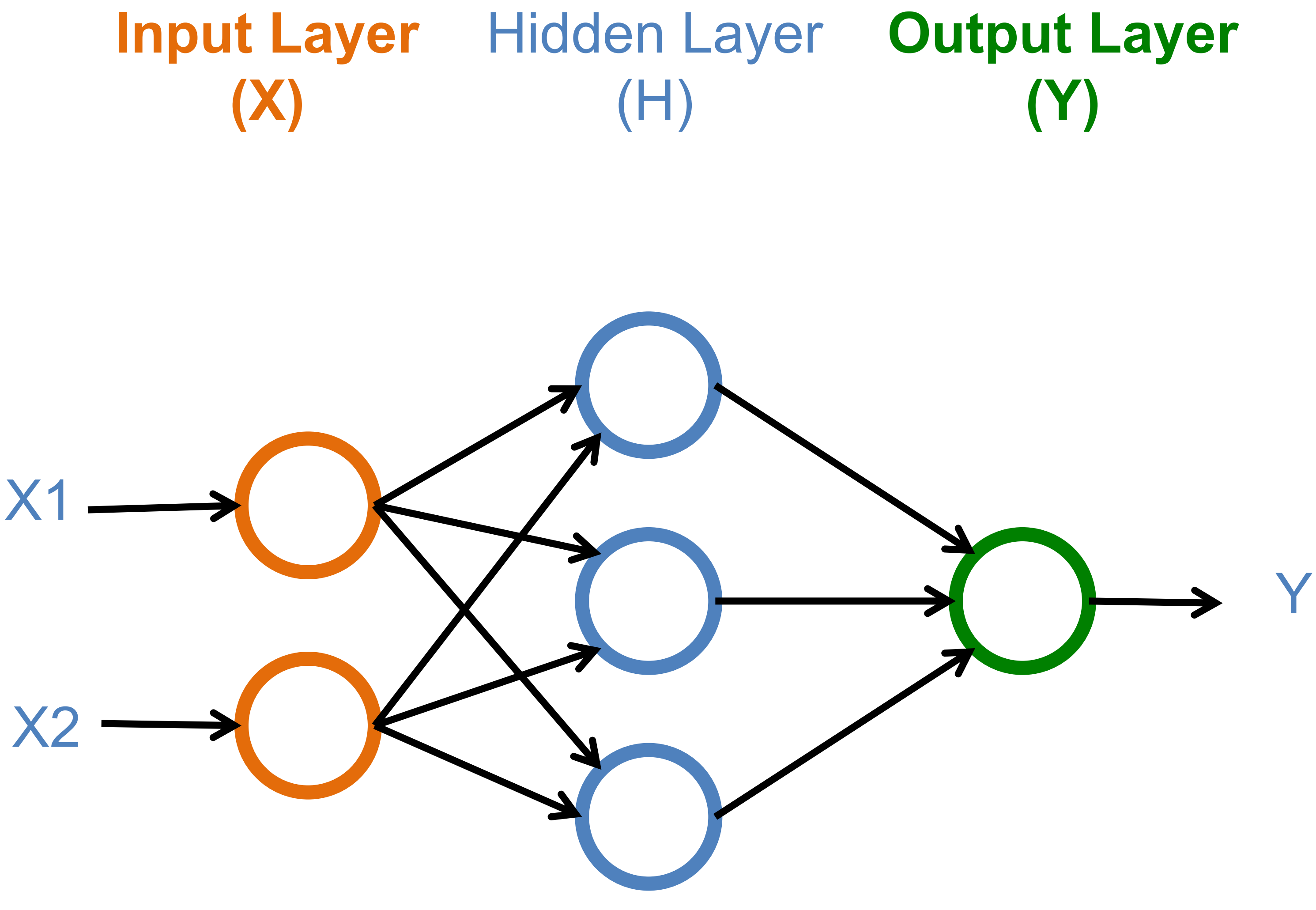
Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing

Convolutional DBN
2010

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

Max-Pooling CDBN
2011

Neural Networks

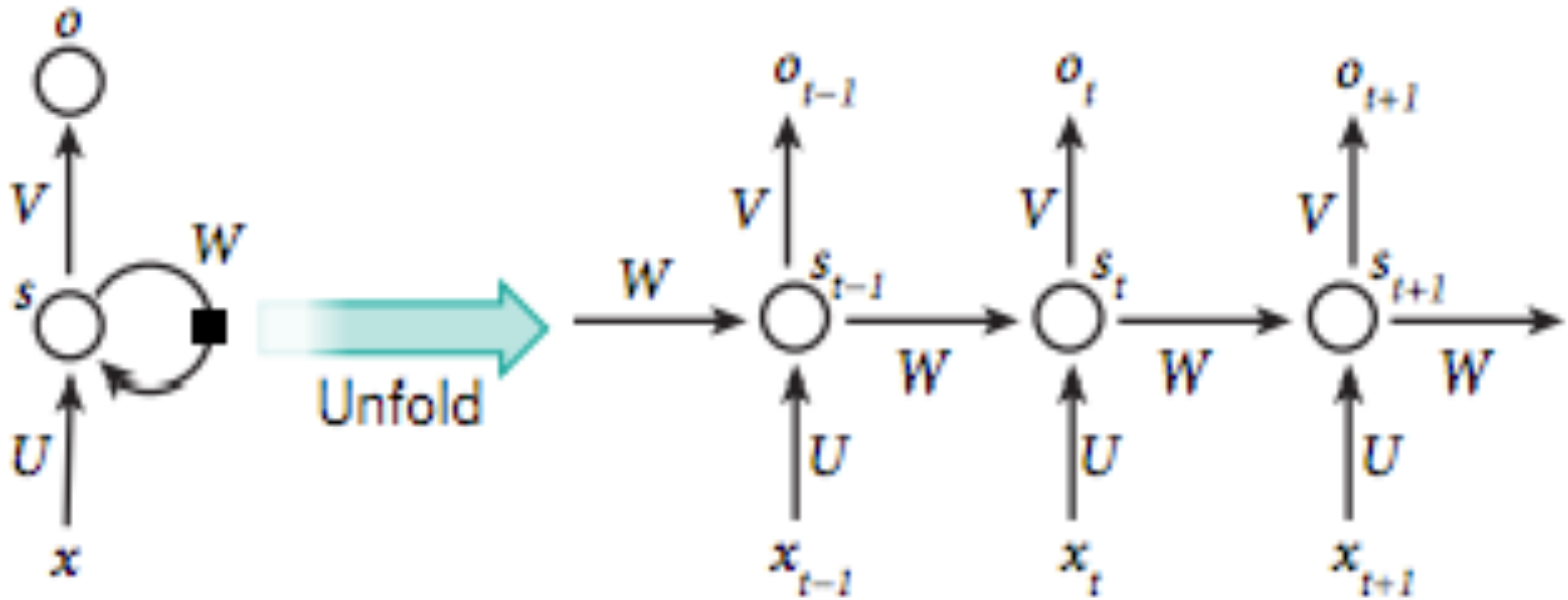


Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Deep Learning

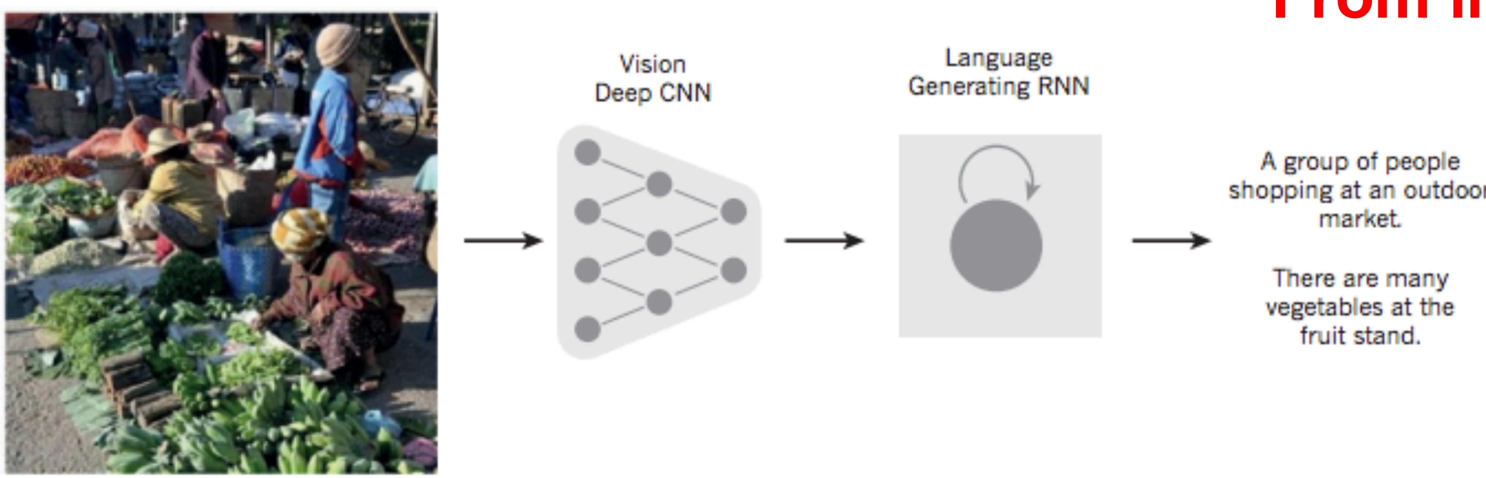
Geoffrey Hinton
Yann LeCun
Yoshua Bengio
Andrew Y. Ng

Recurrent Neural Network (RNN)



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

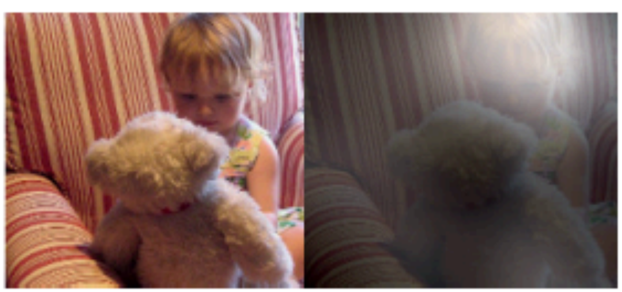
From image to text



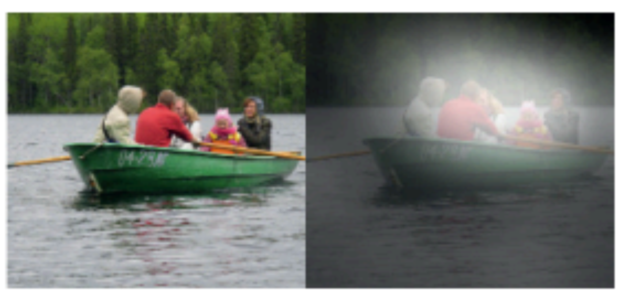
A woman is throwing a **frisbee** in a park.

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)

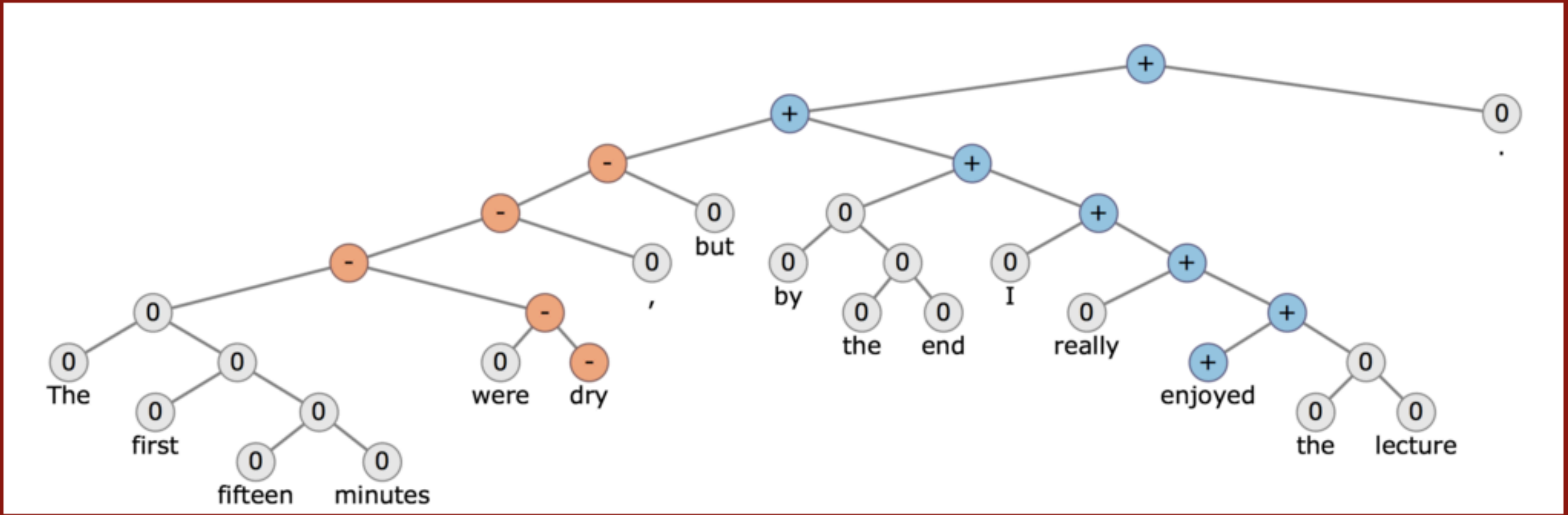
Text: recurrent neural network (RNN)

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

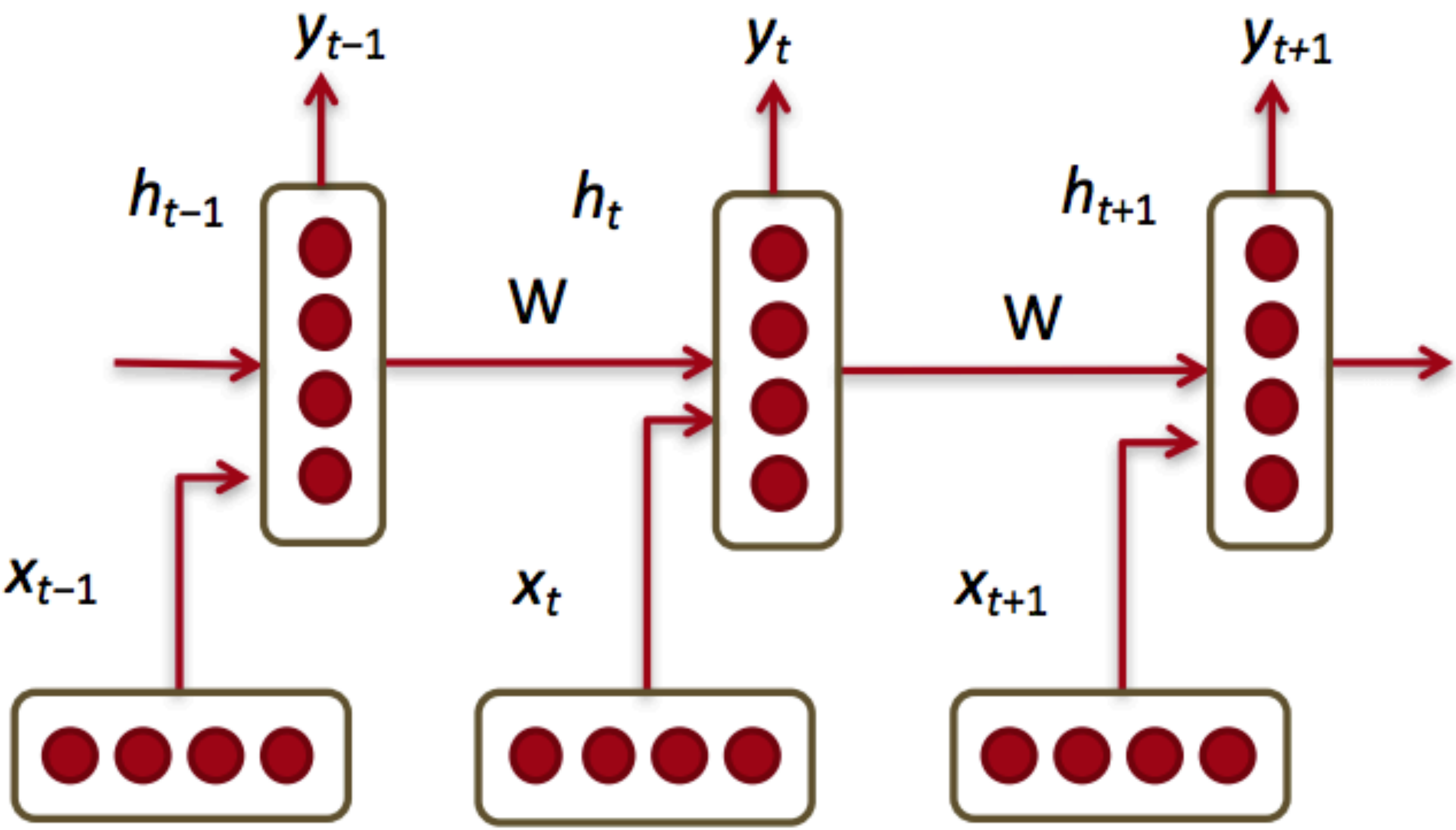


Course Description

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,

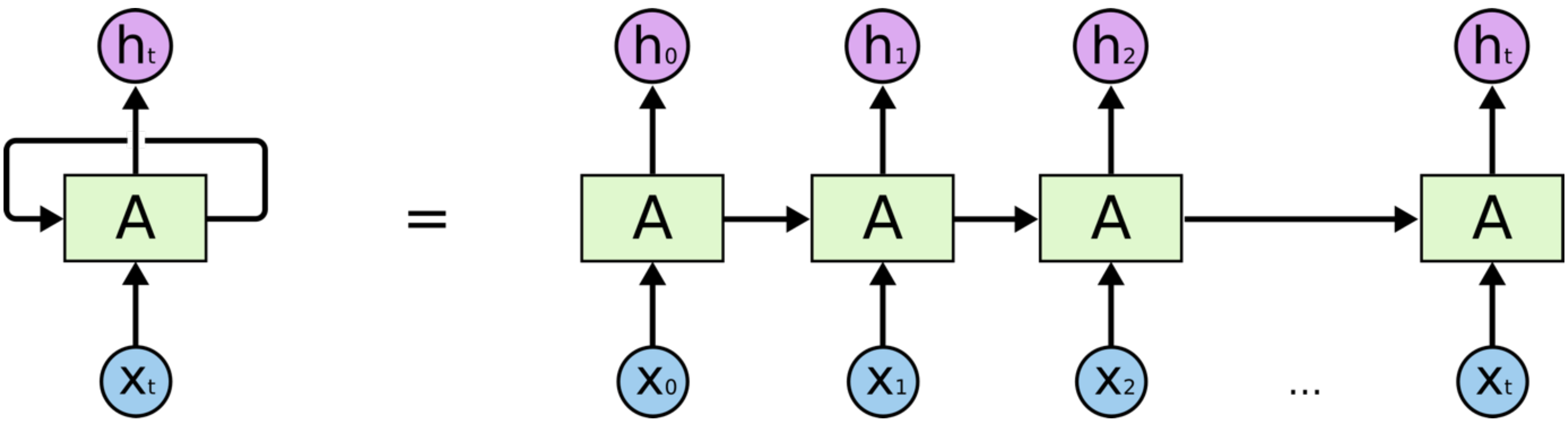
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



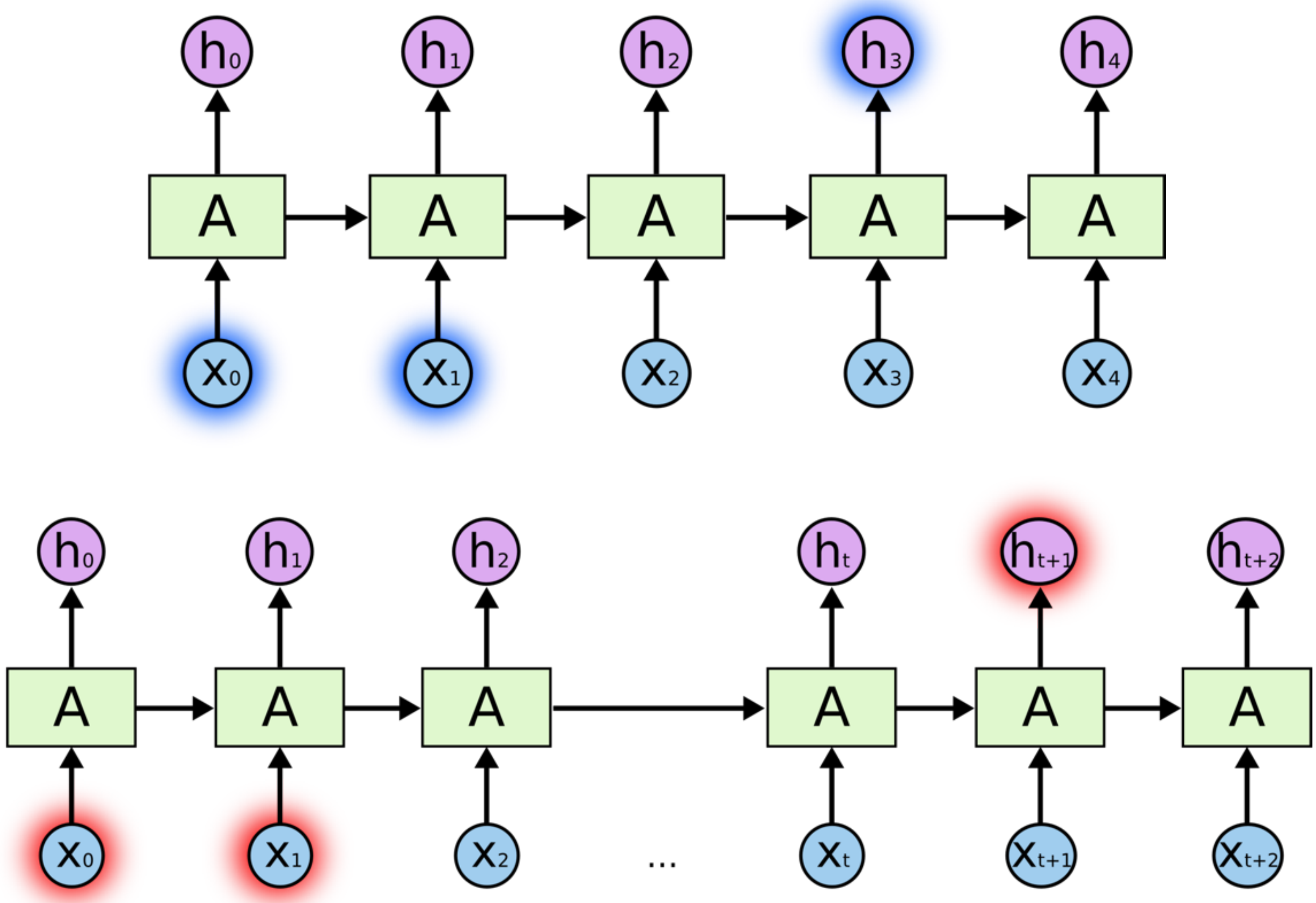
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN



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

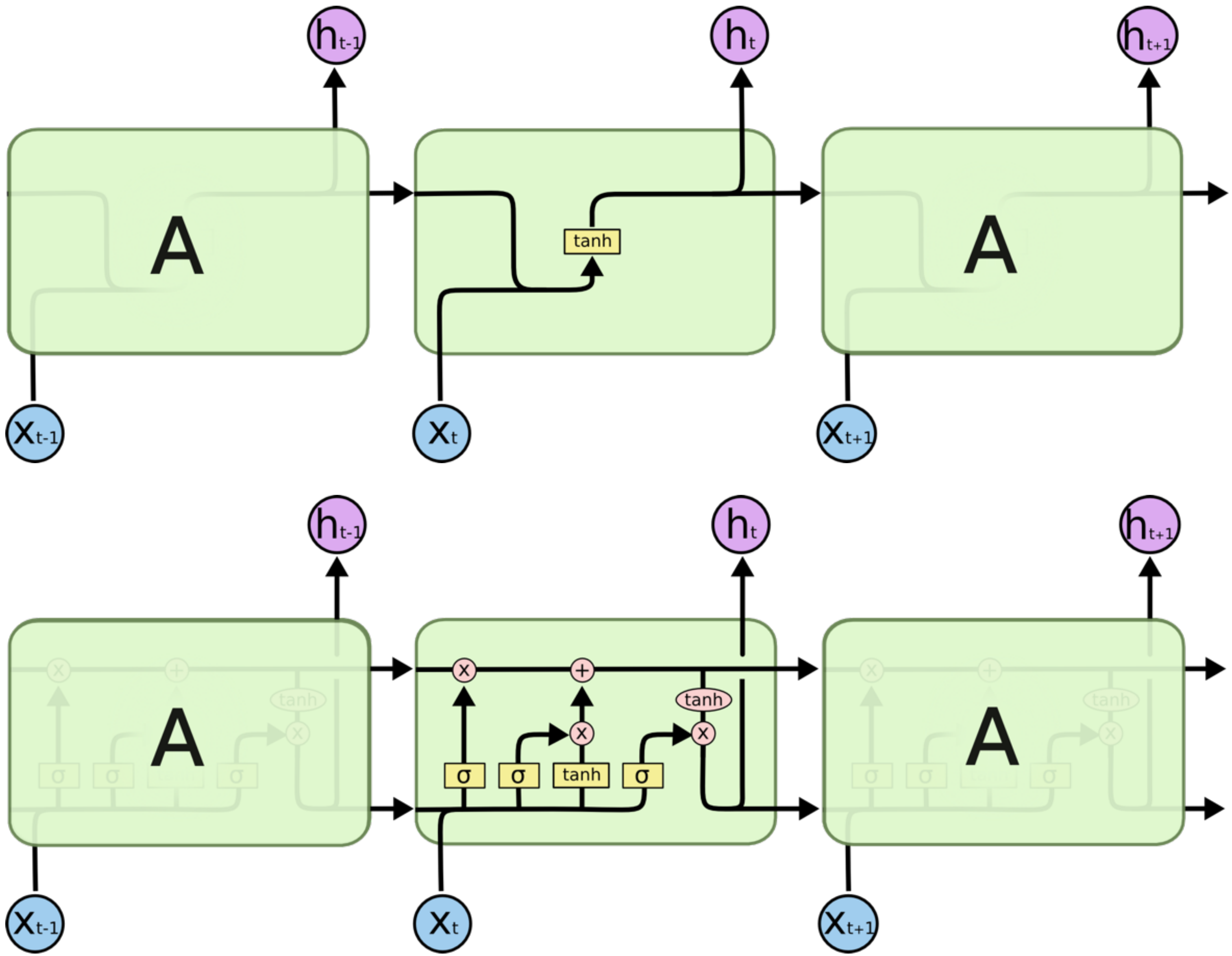
RNN long-term dependencies



I grew up in France… I speak fluent French.

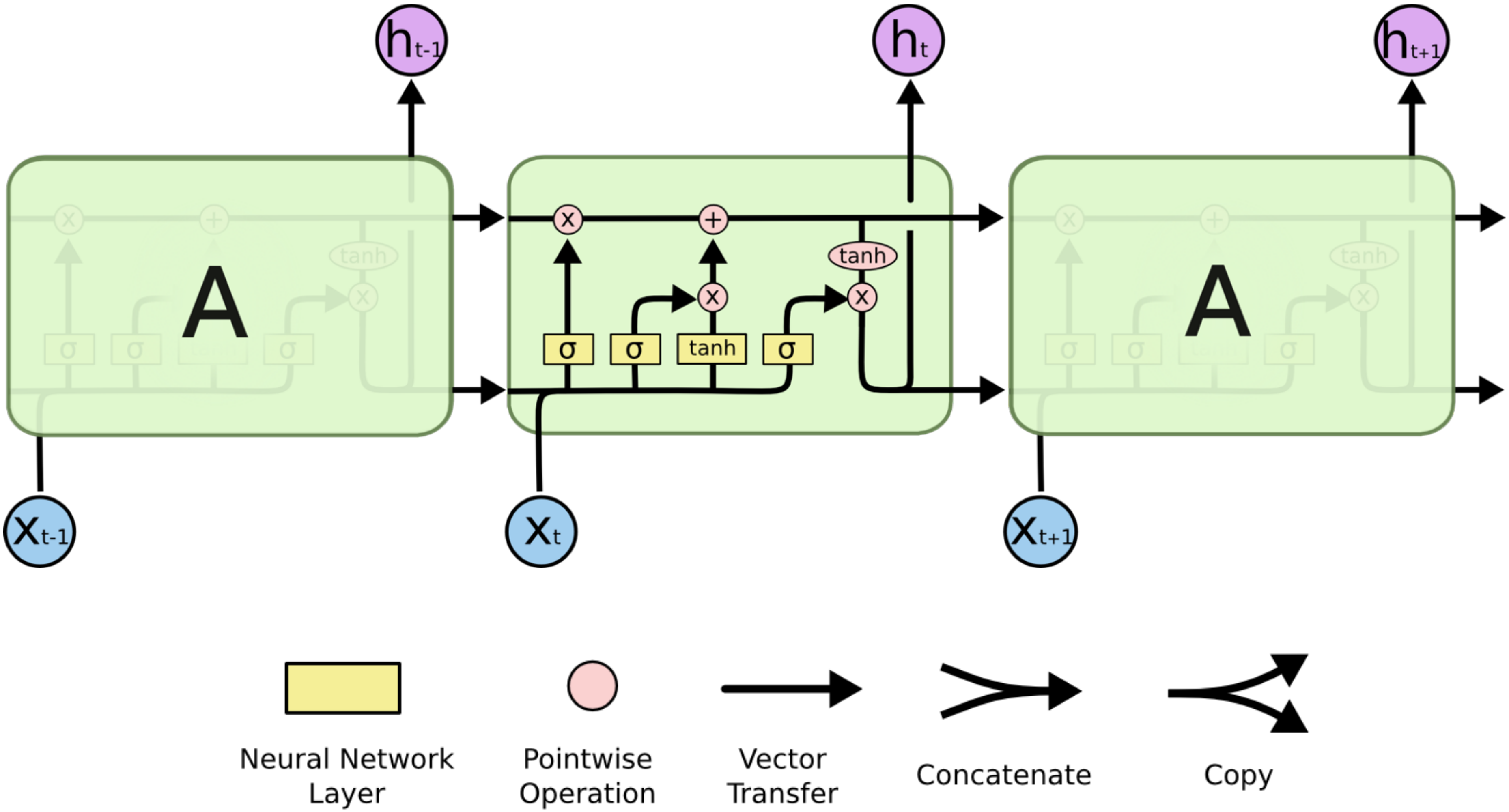
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN LSTM



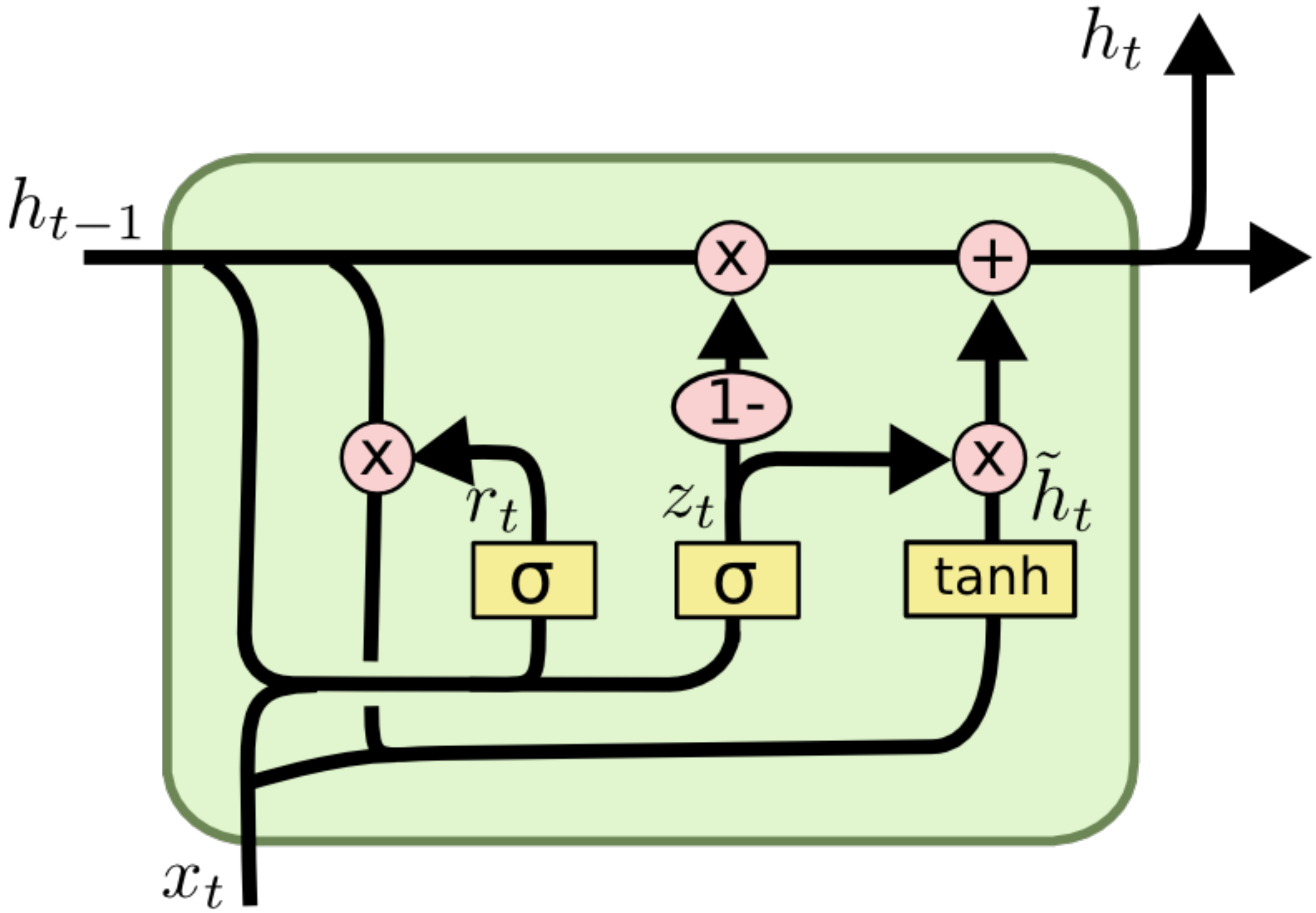
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



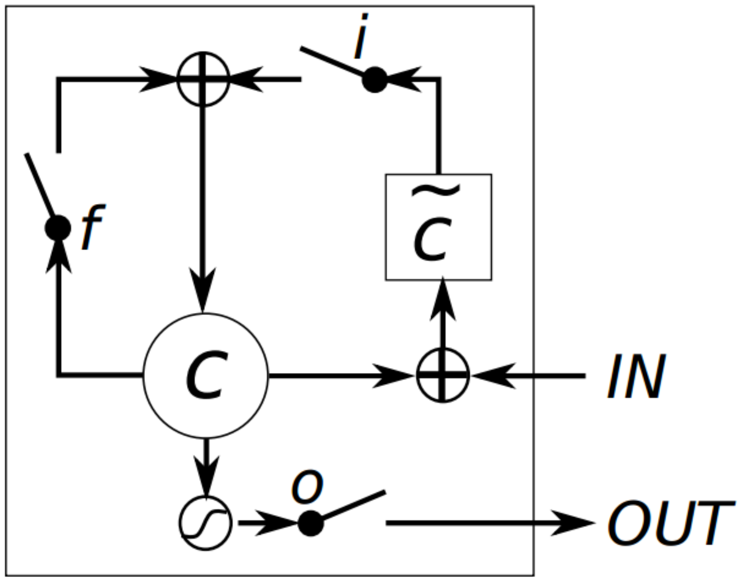
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

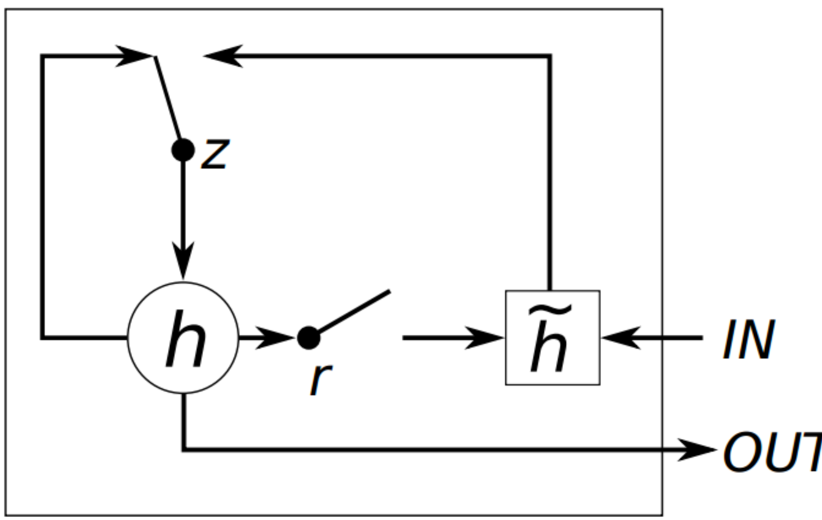
Gated Recurrent Unit (GRU)



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

LSTM vs GRU





LSTM

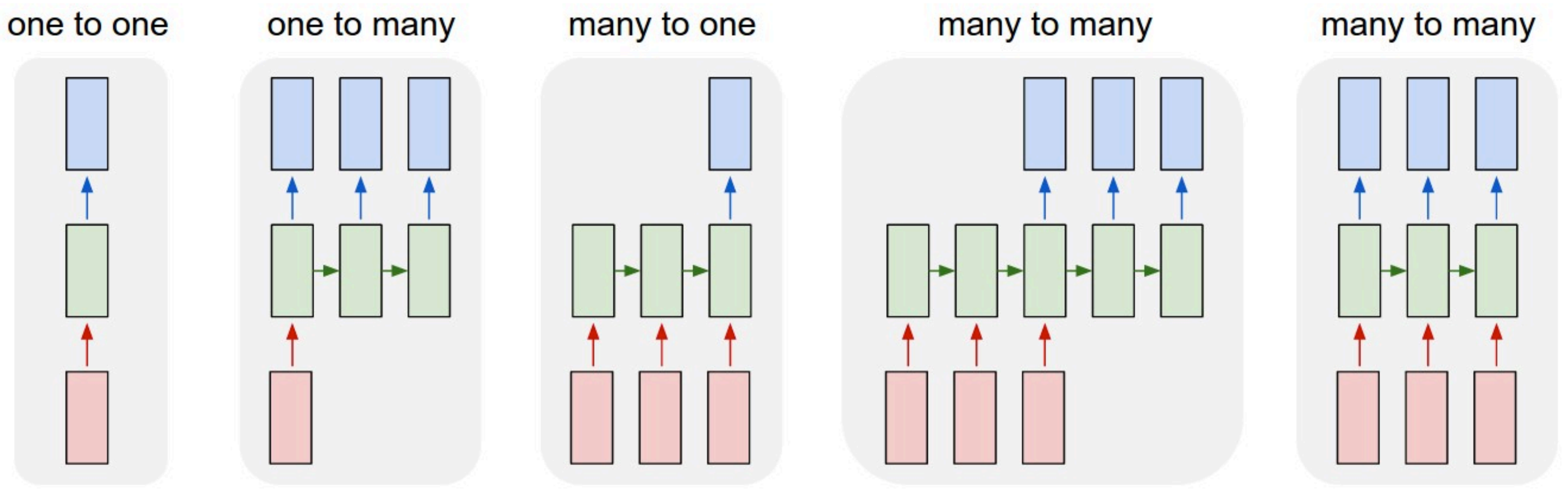
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

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

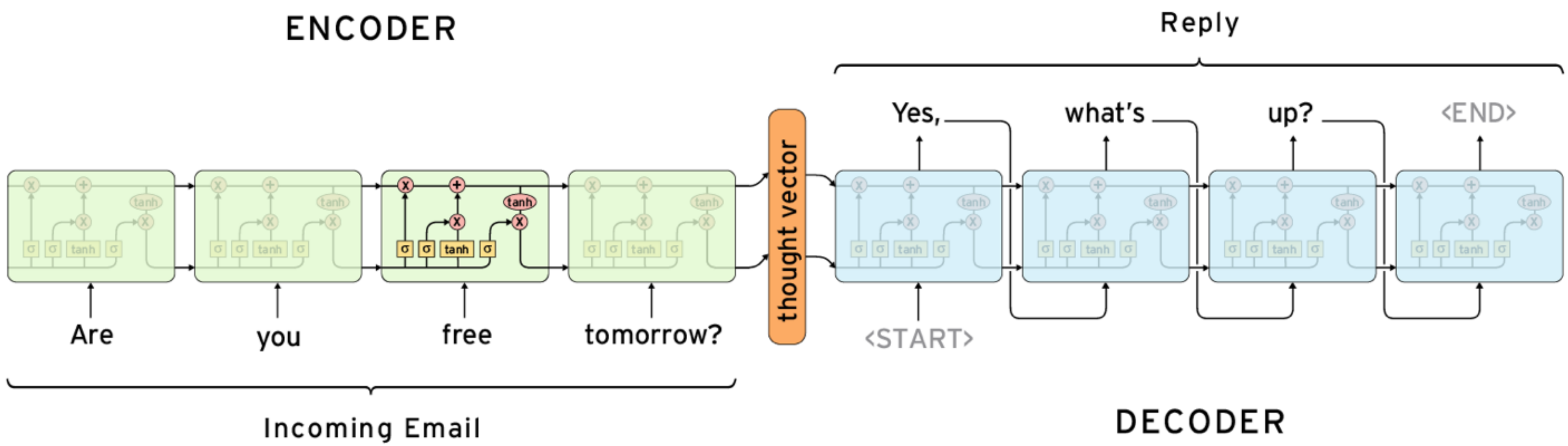
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

LSTM Recurrent Neural Network



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

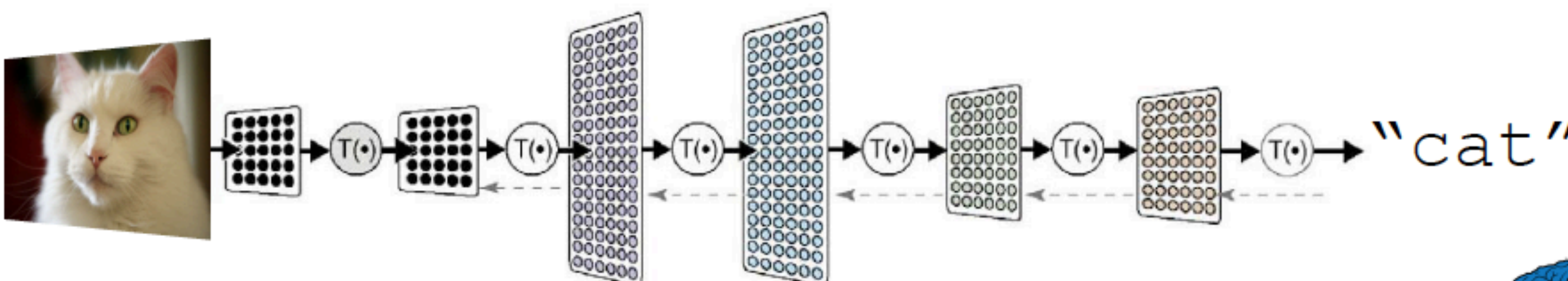
The Sequence to Sequence model (seq2seq)



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

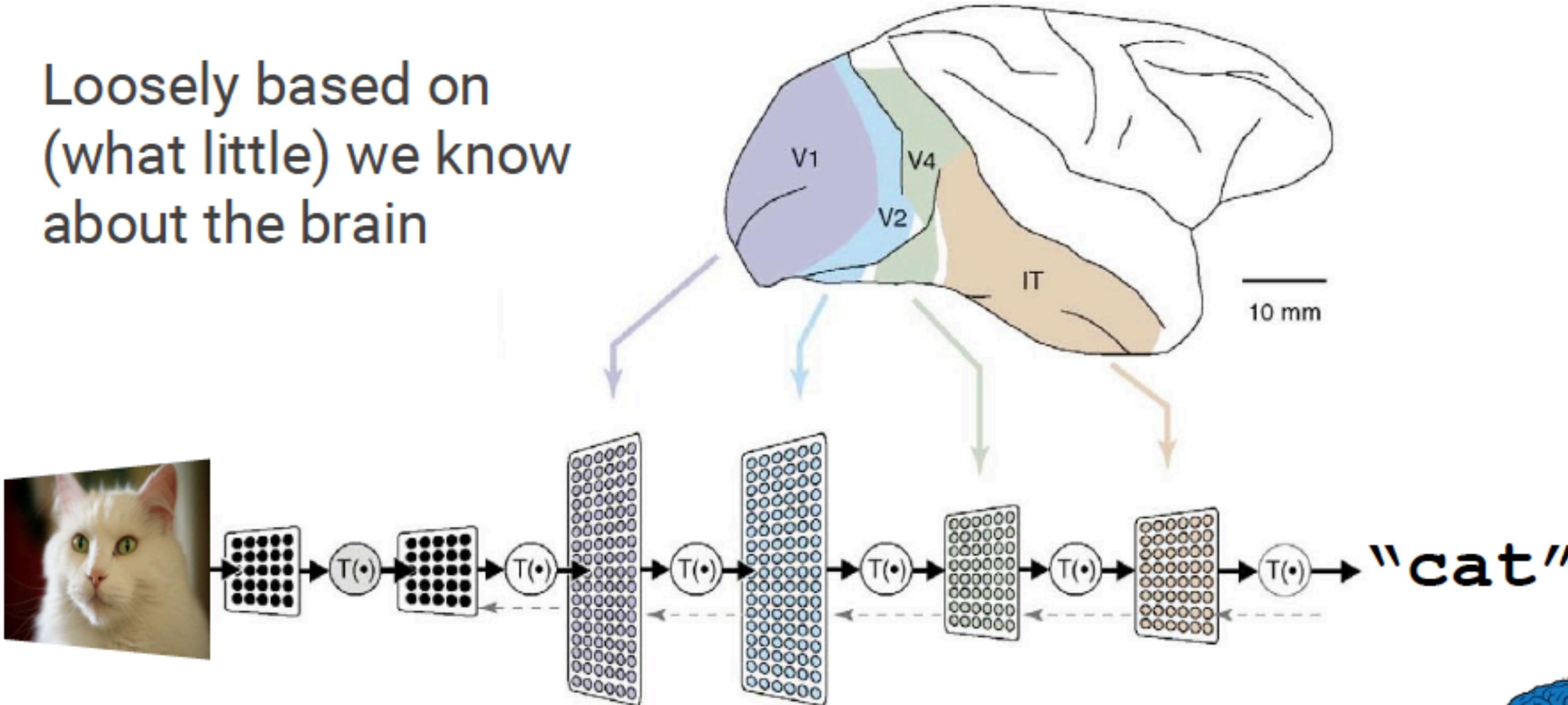
Deep Learning

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

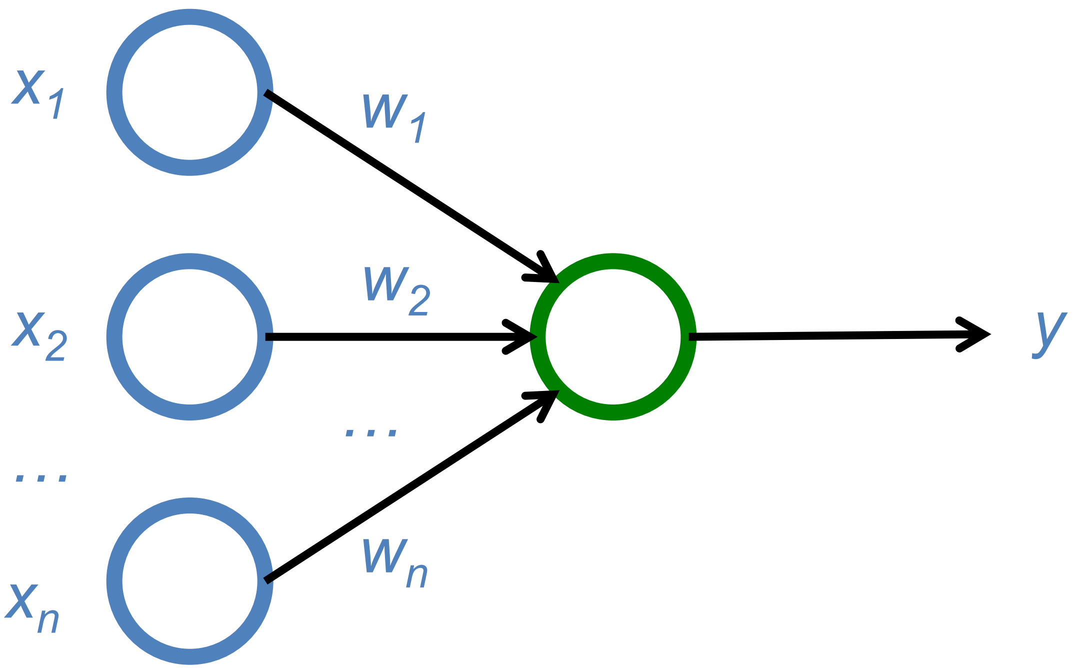


What is Deep Learning?

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

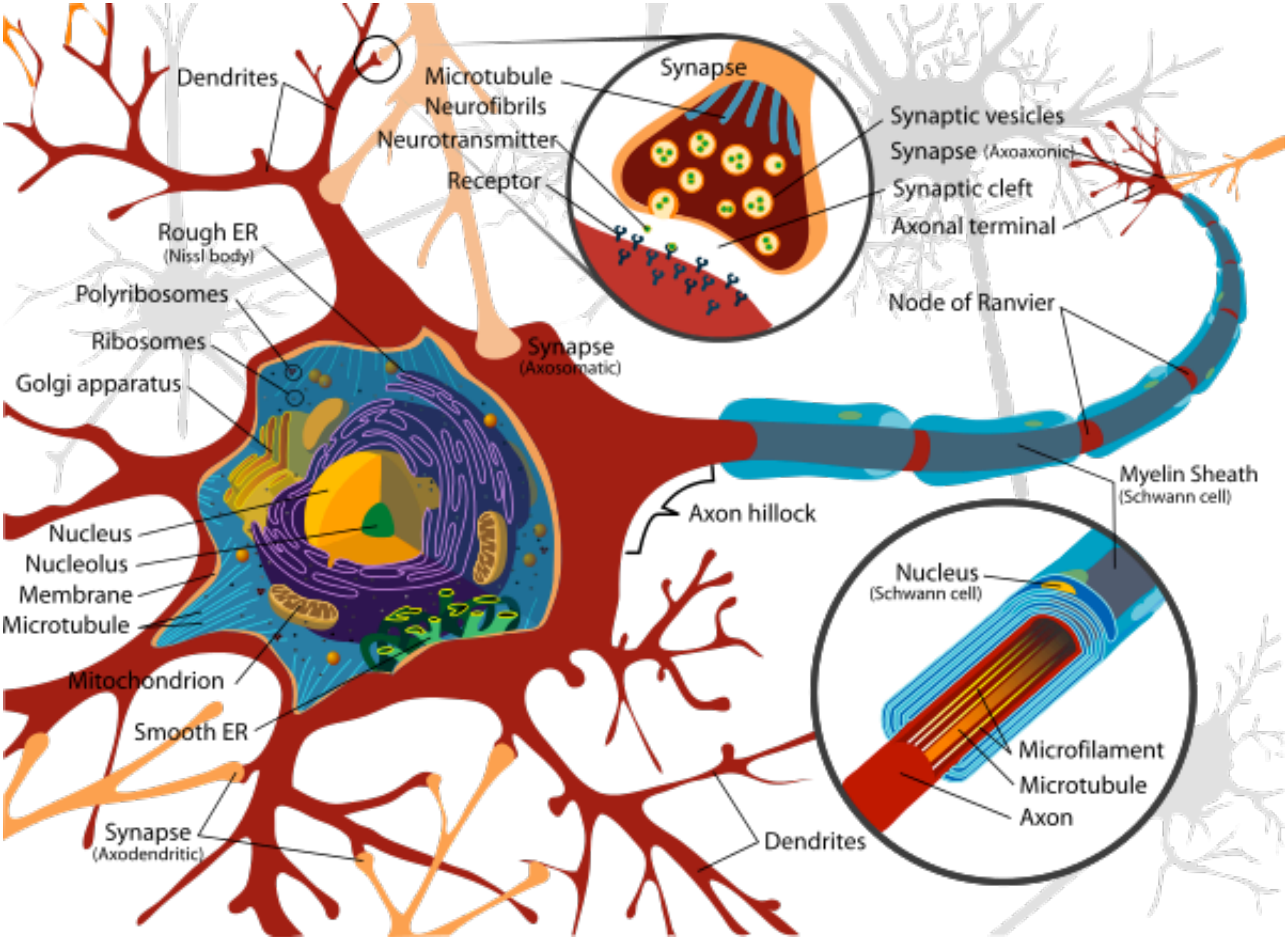


The Neuron



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

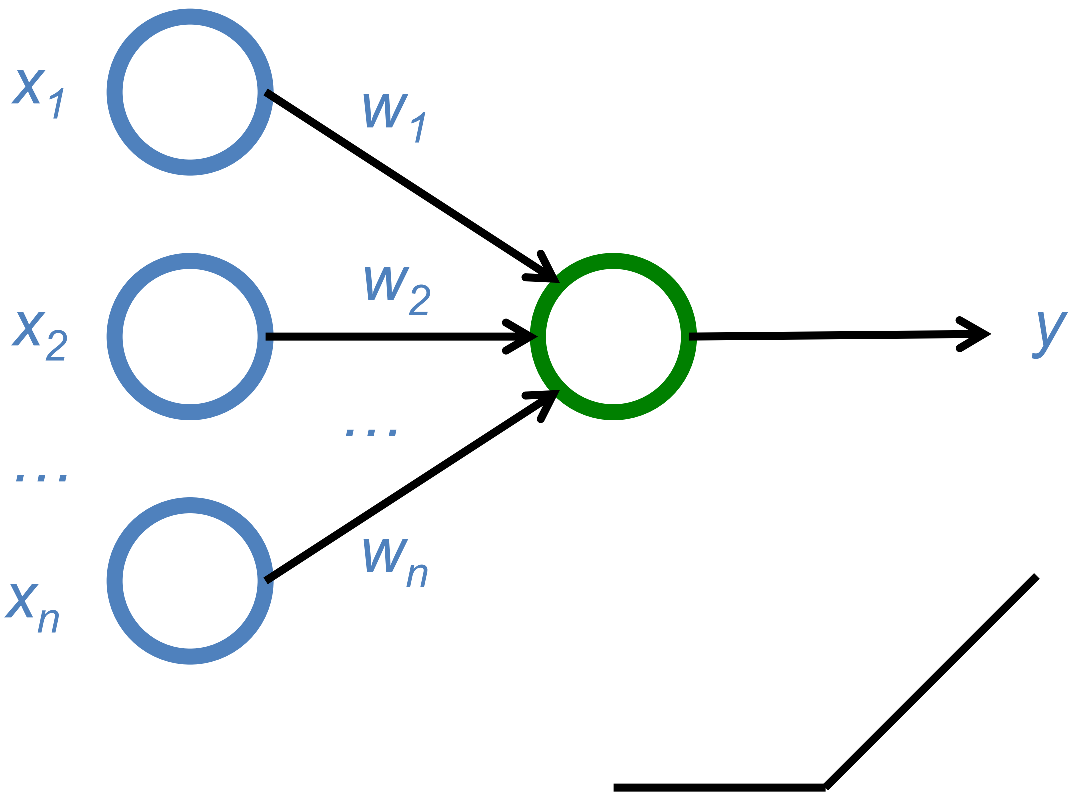
Neuron and Synapse



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

The Neuron

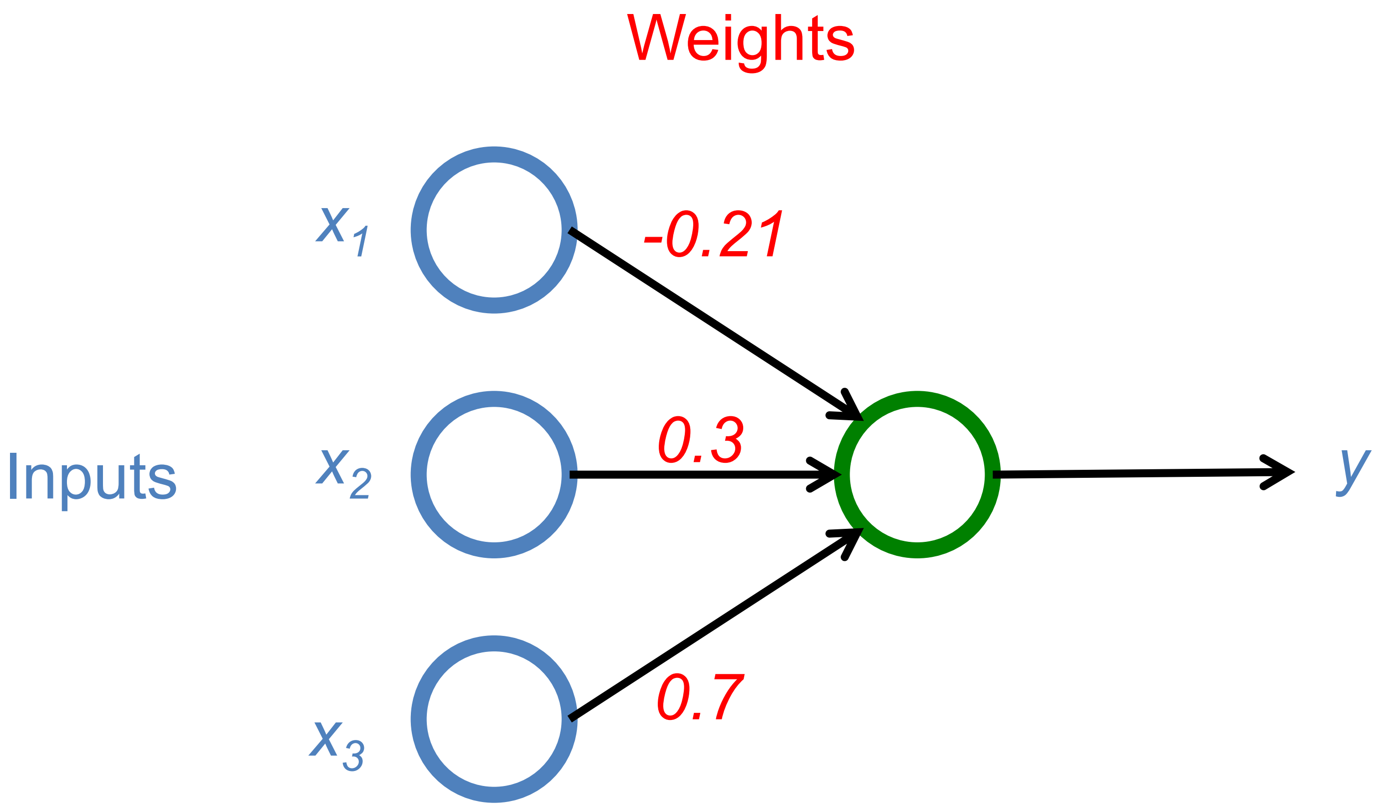
$$y = F\left(\sum_i w_i x_i\right)$$



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

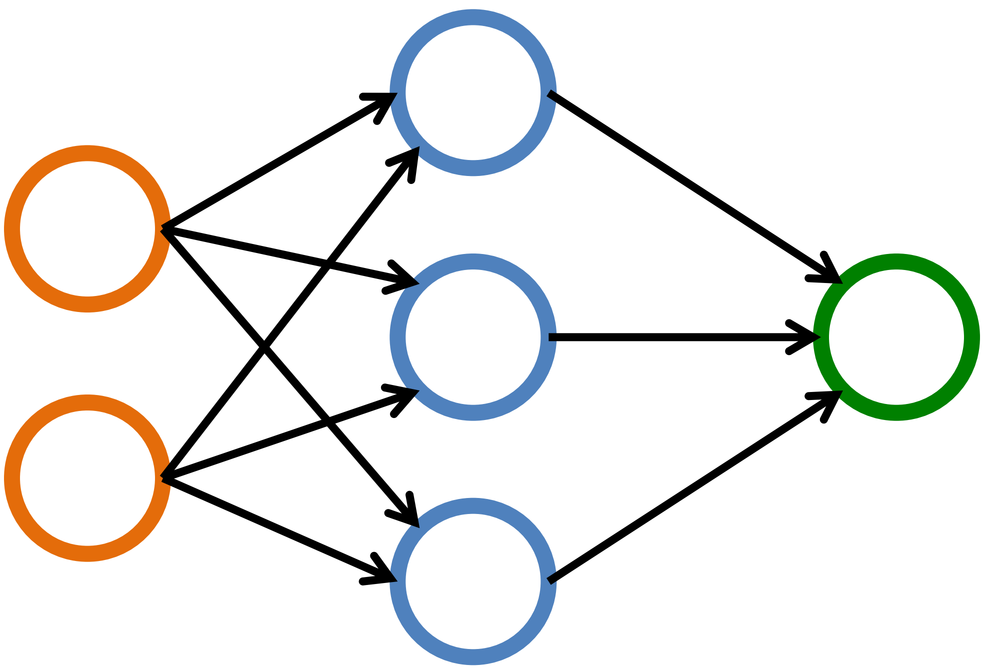
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

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



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

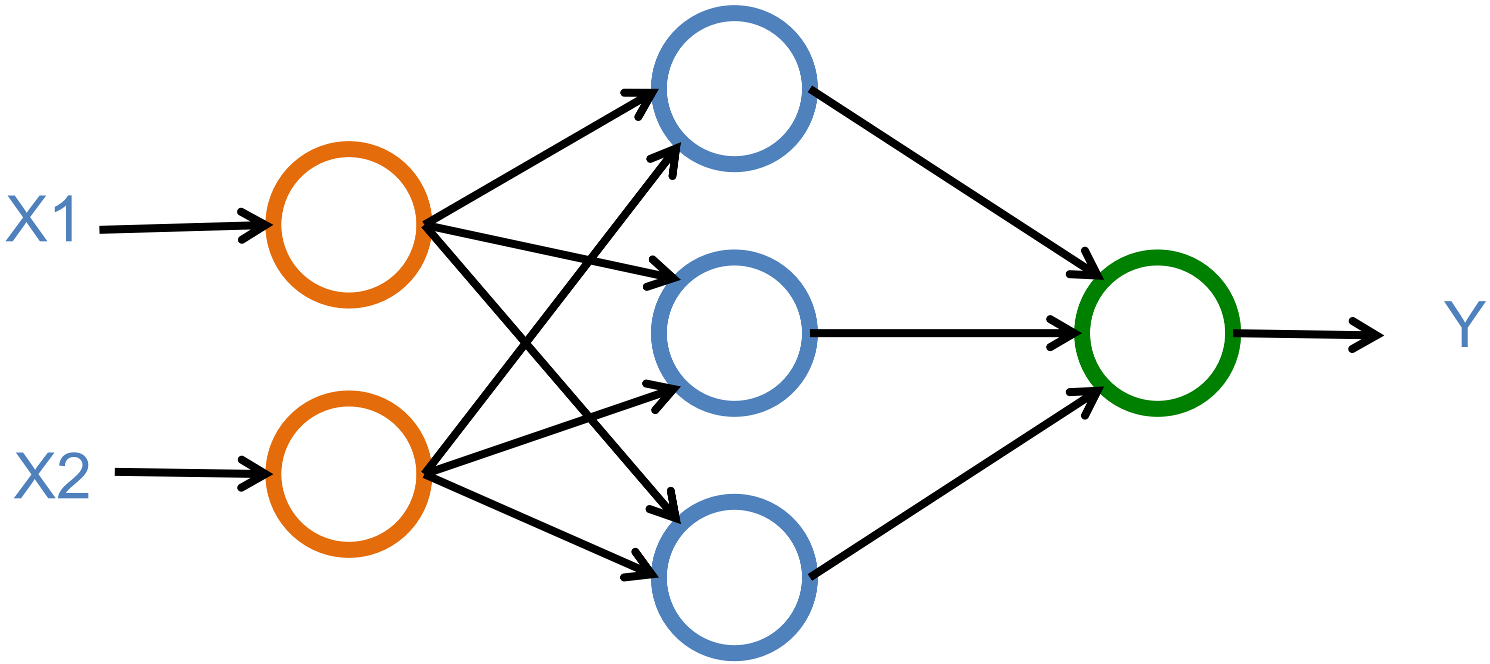
Neural Networks



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

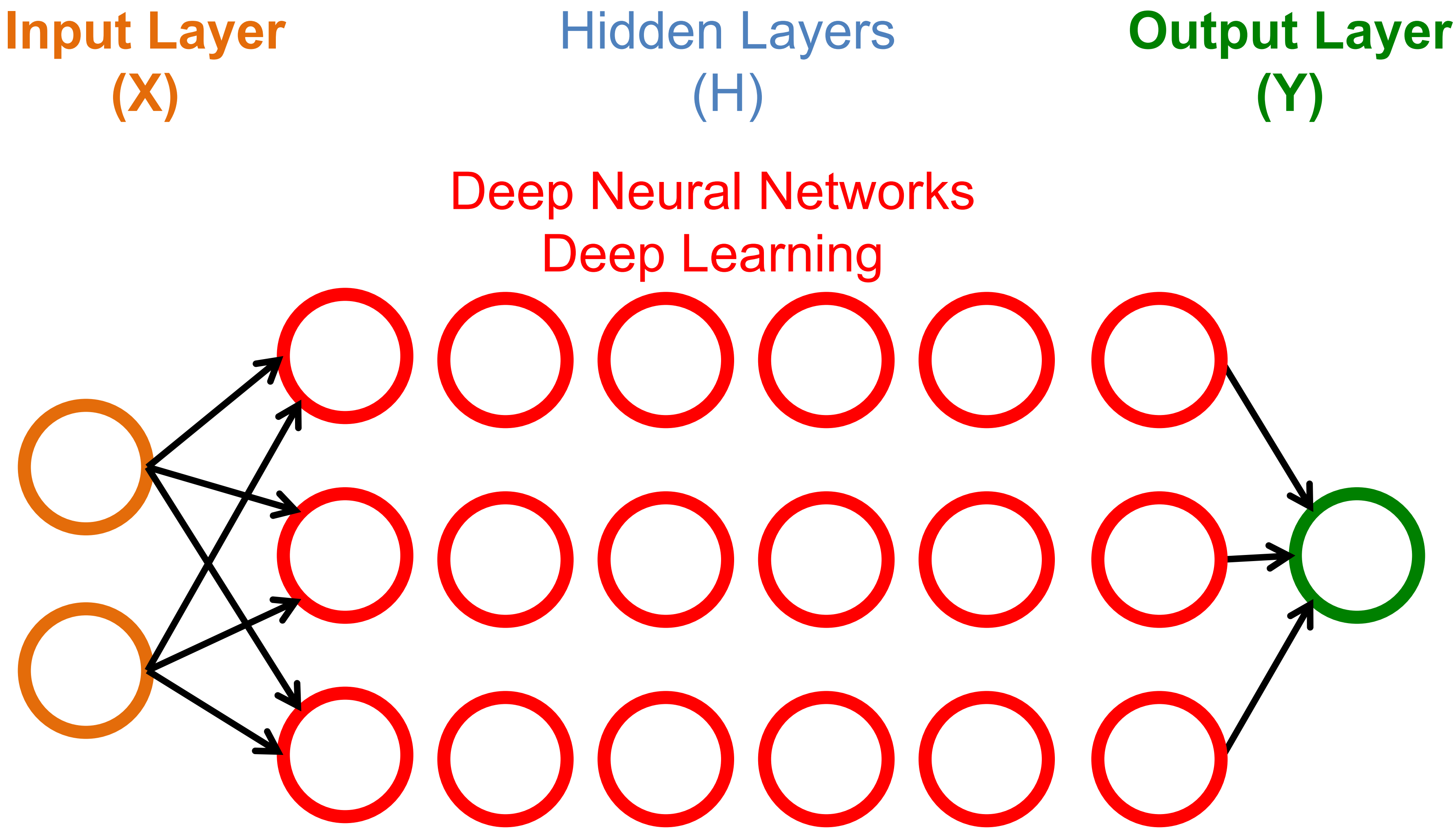
Neural Networks

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



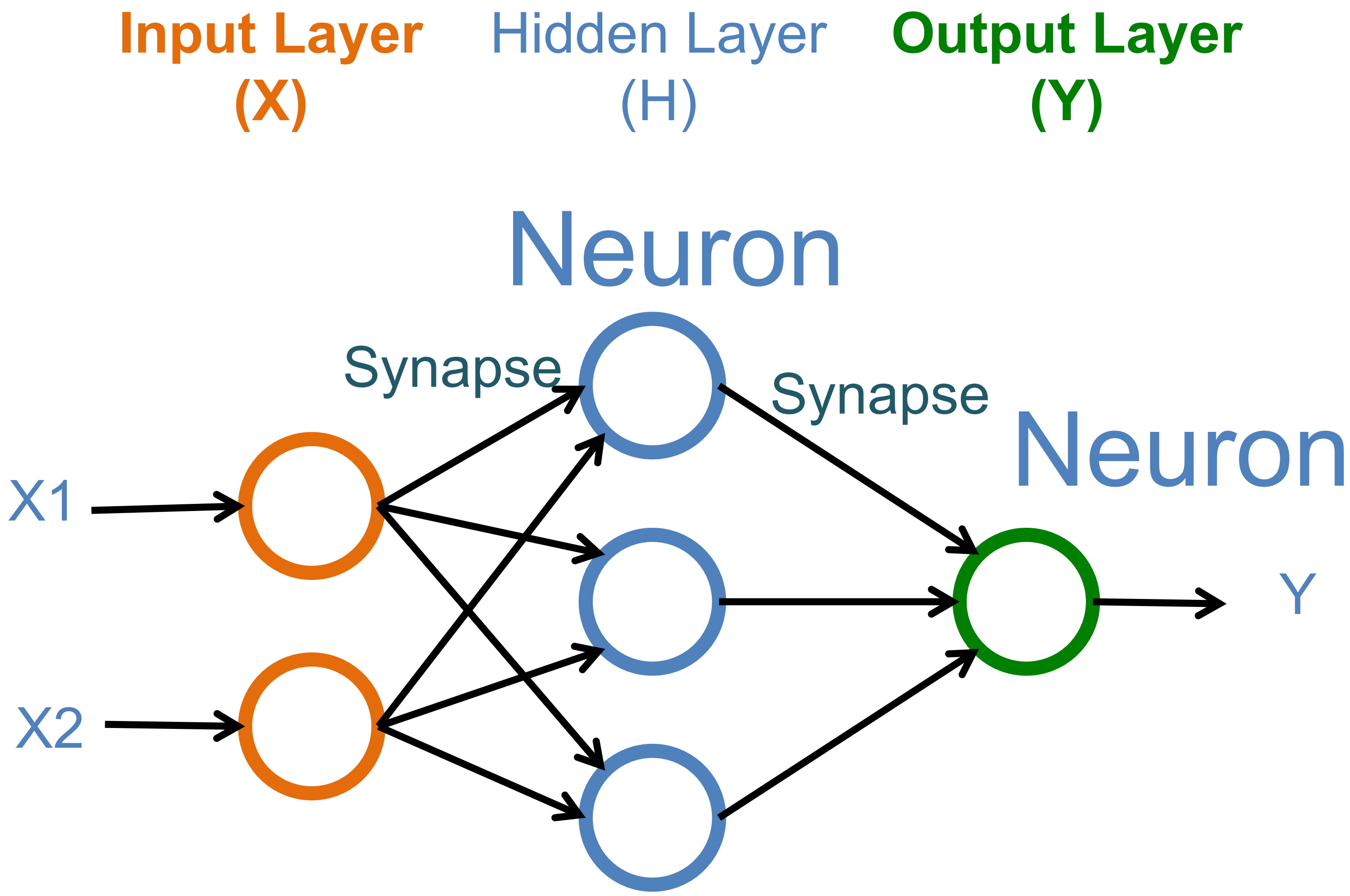
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



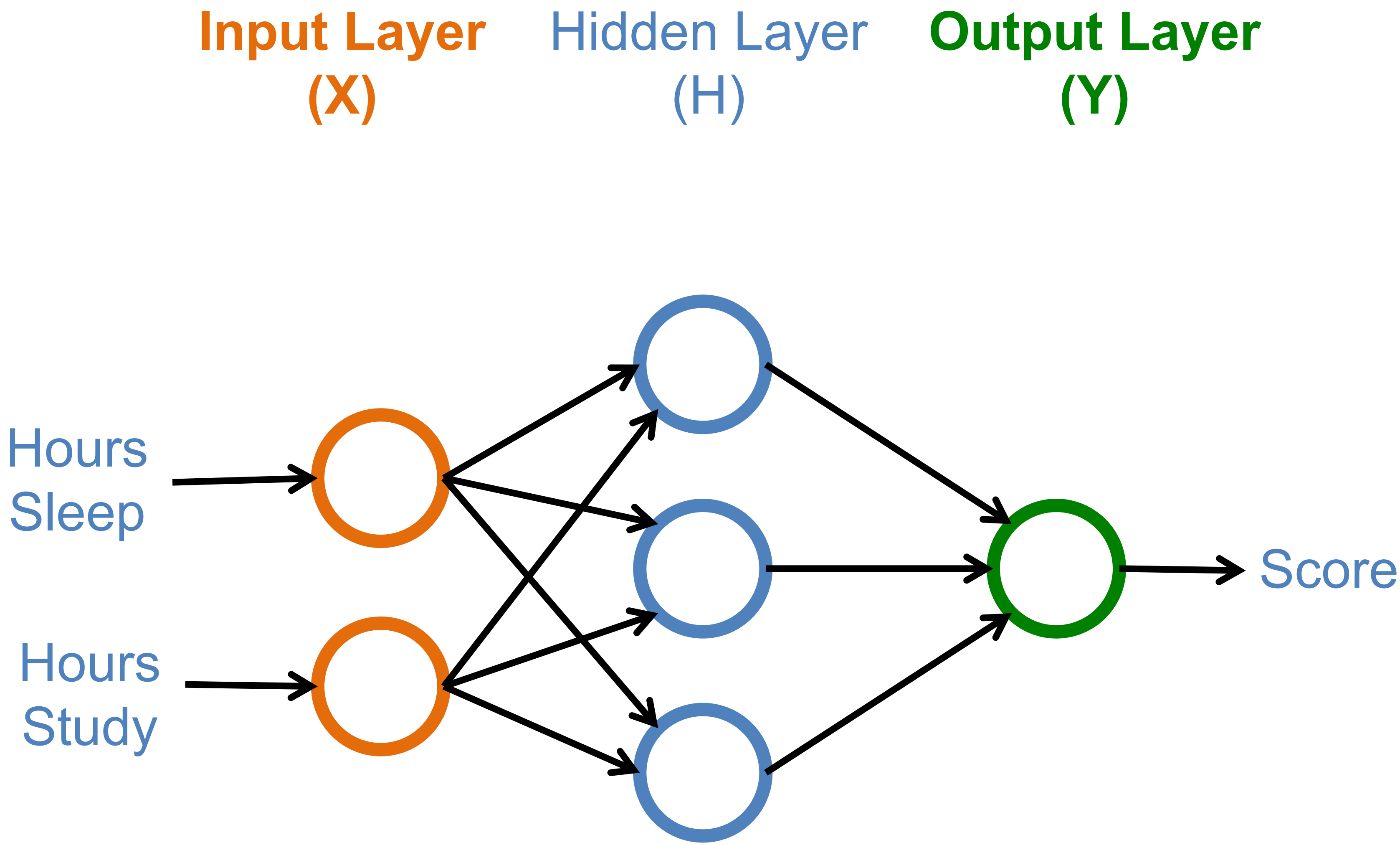
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



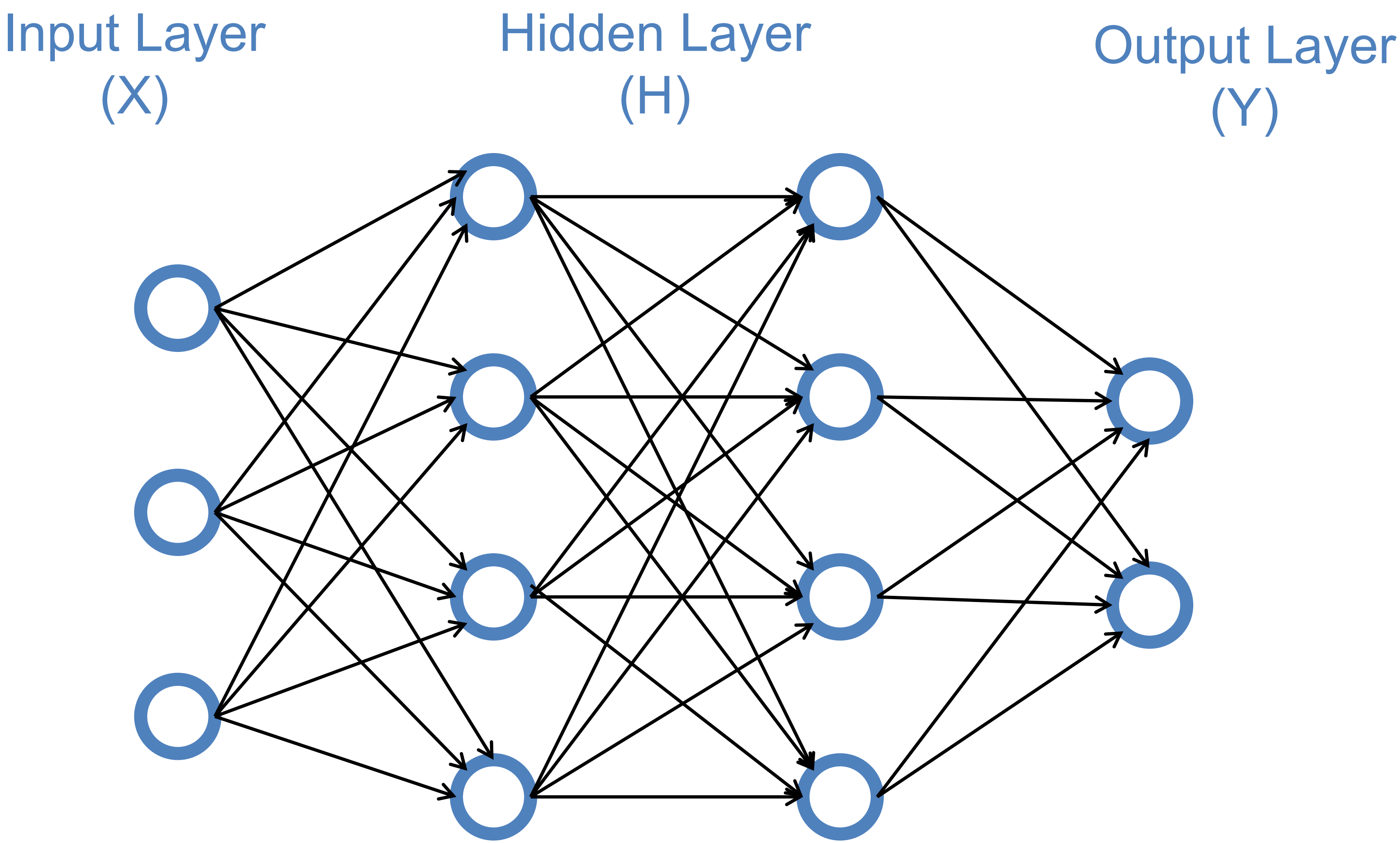
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



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

Neural Networks

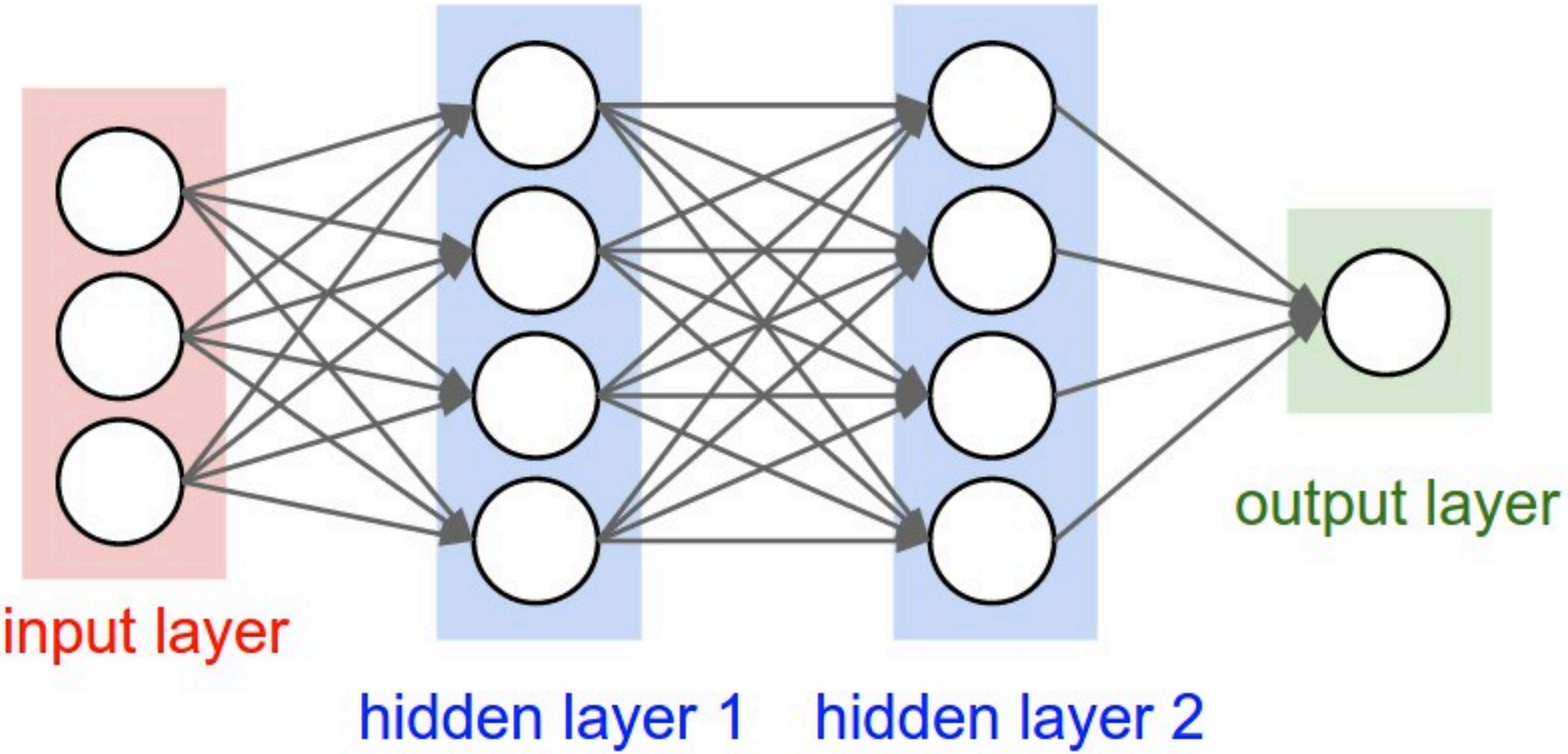


Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2

Convolutional Neural Networks
(CNNs / ConvNets)

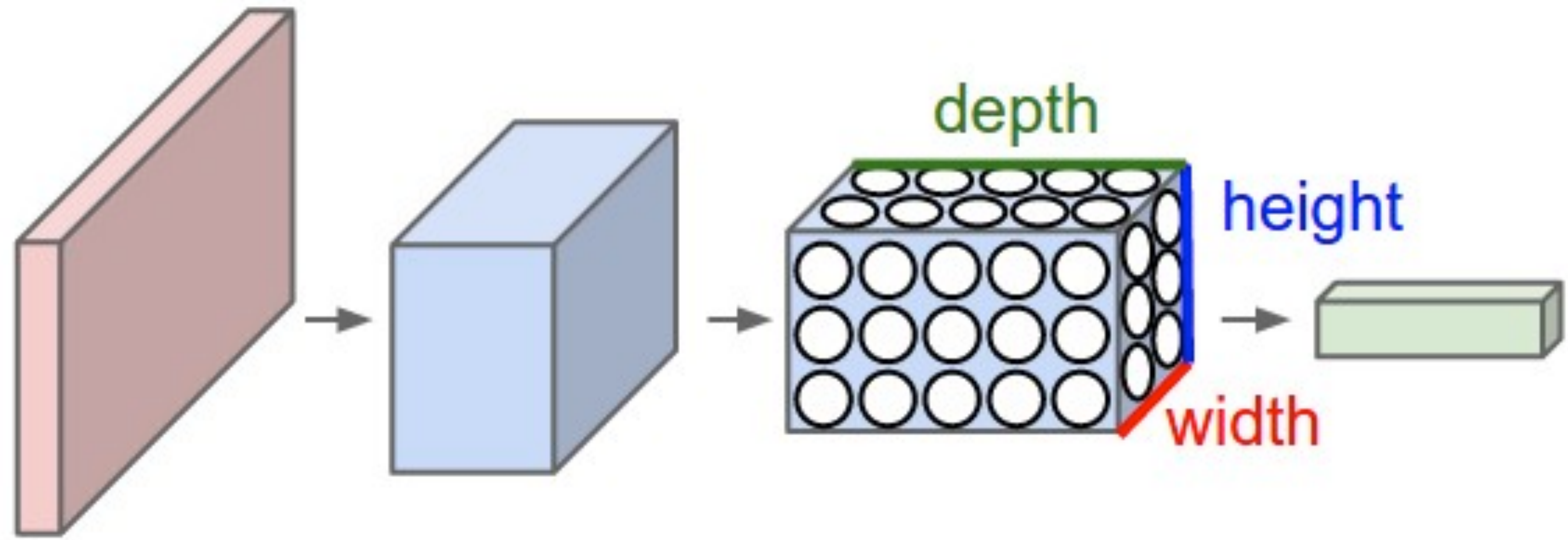
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



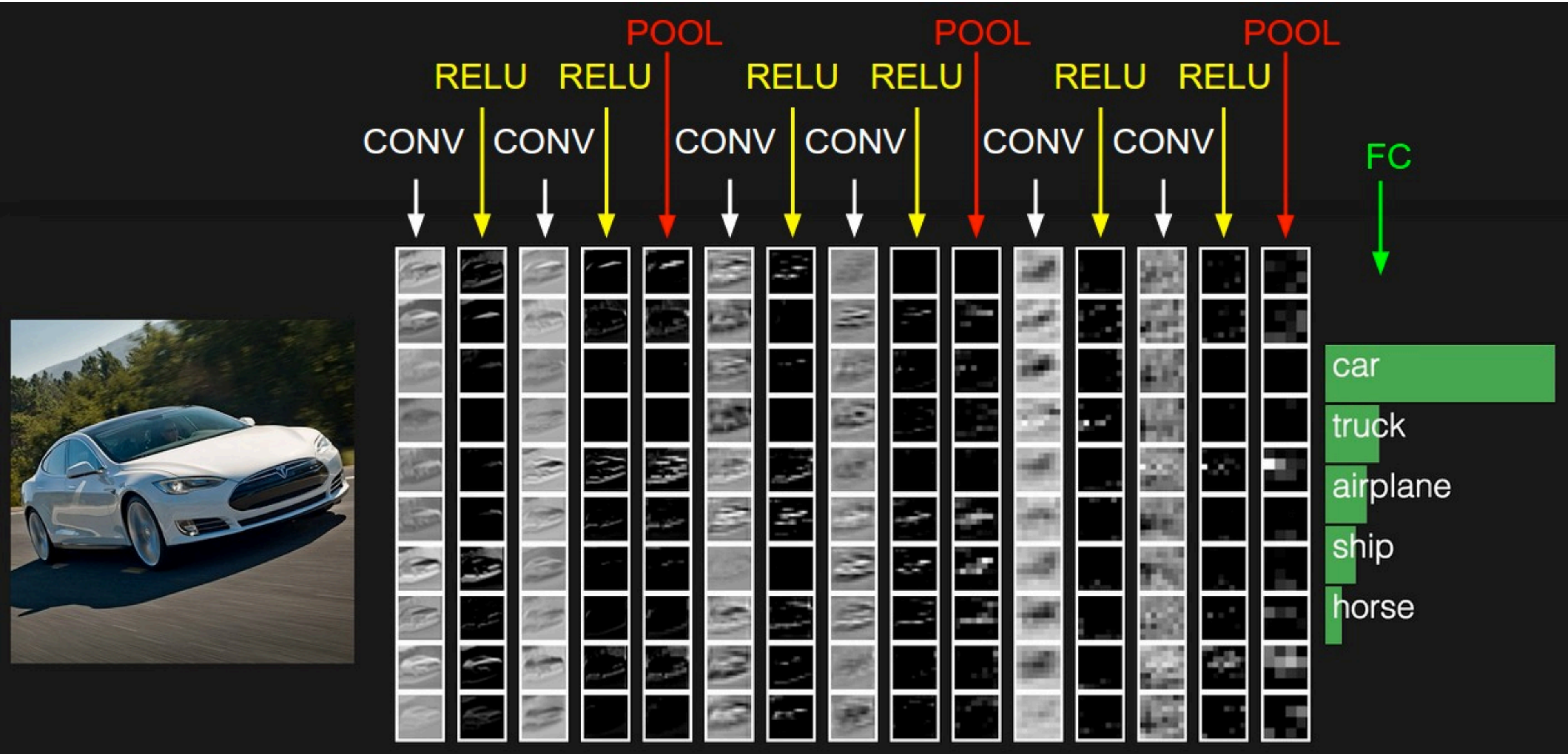
http://cs231n.github.io/convolutional-networks/

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



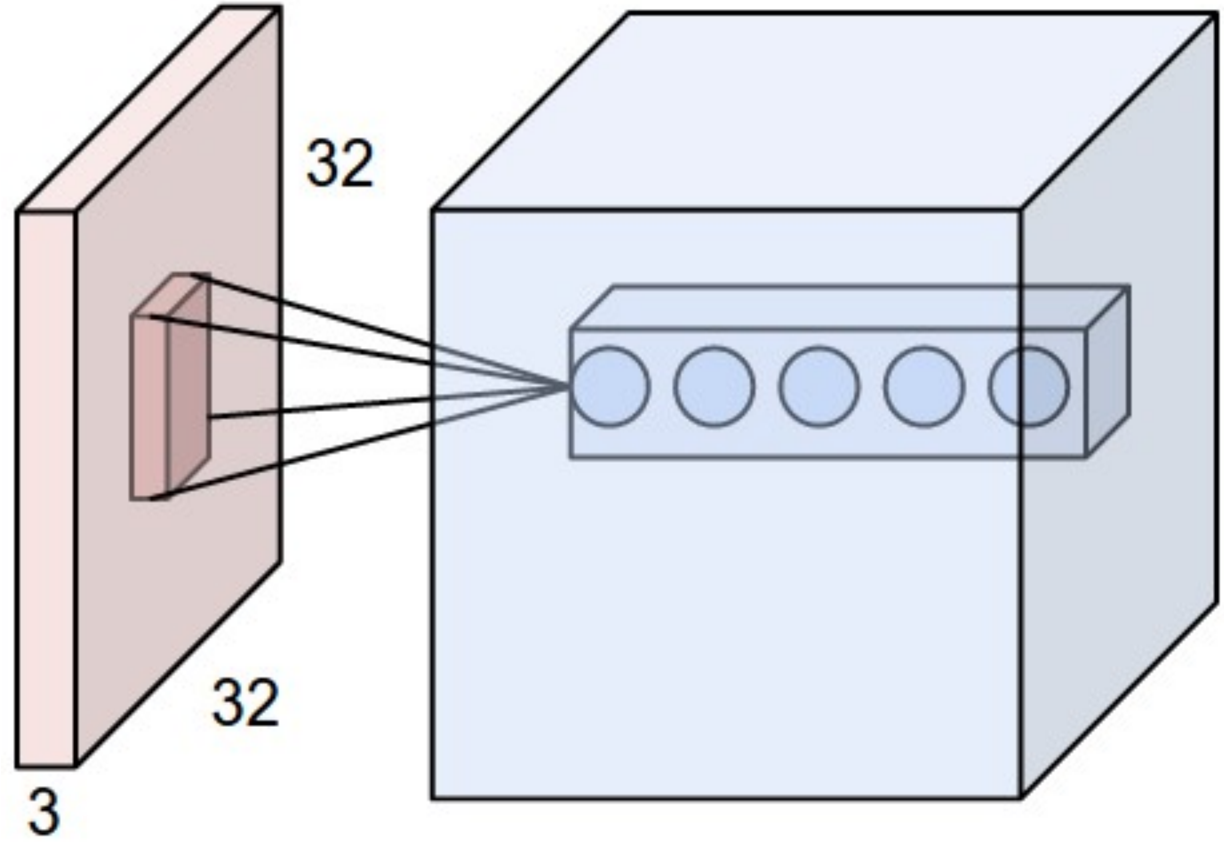
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



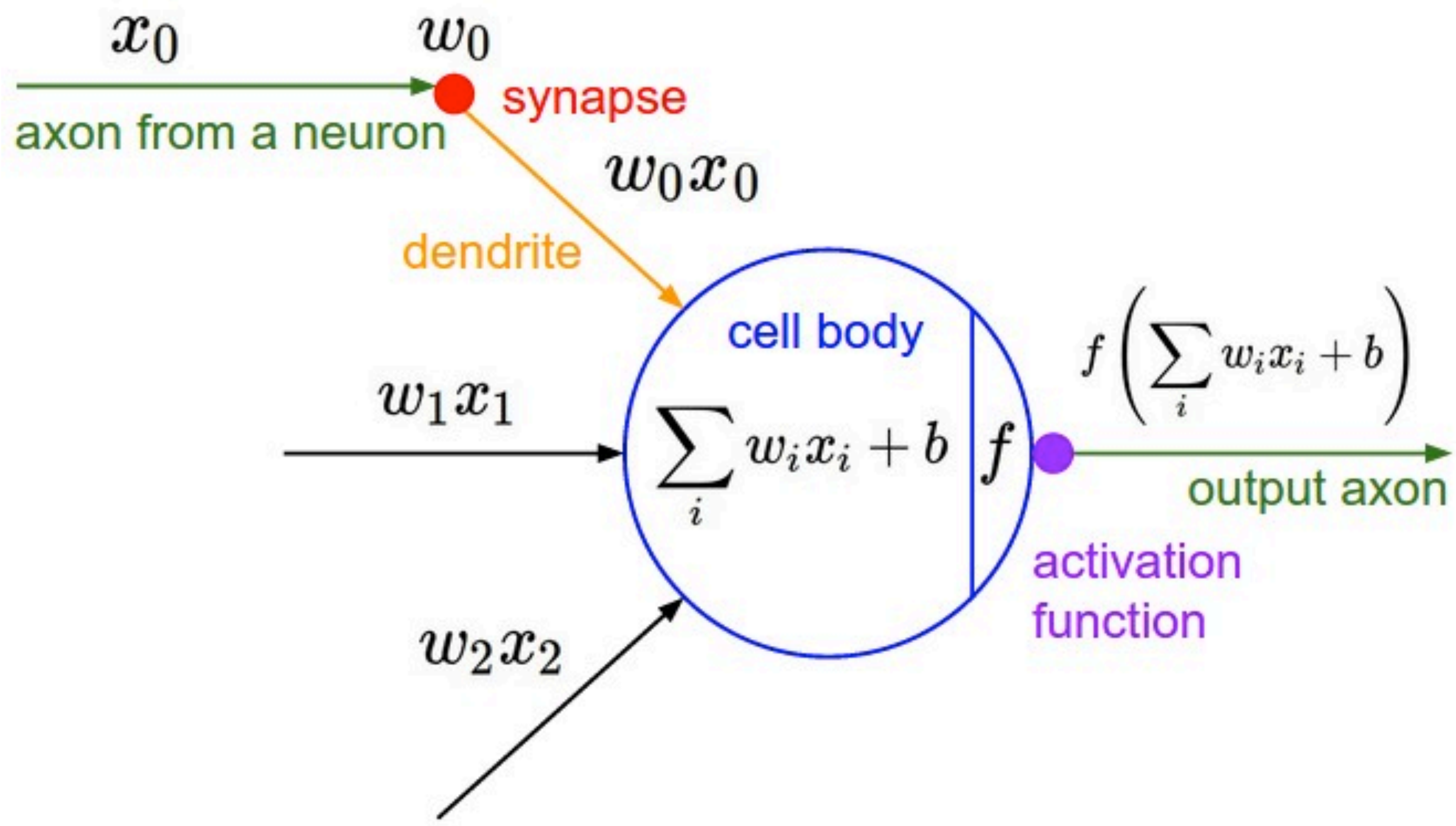
http://cs231n.github.io/convolutional-networks/

ConvNets



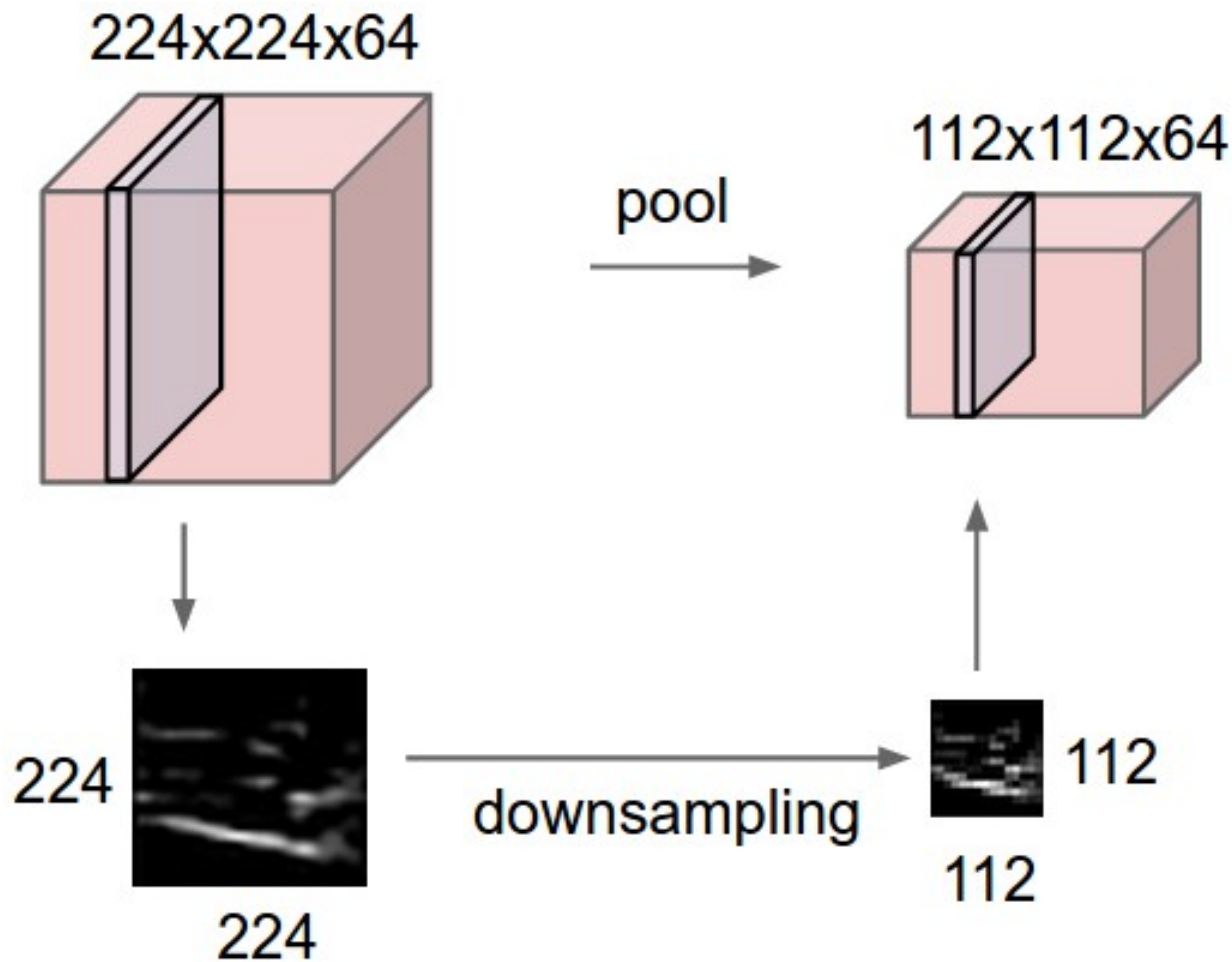
http://cs231n.github.io/convolutional-networks/

ConvNets



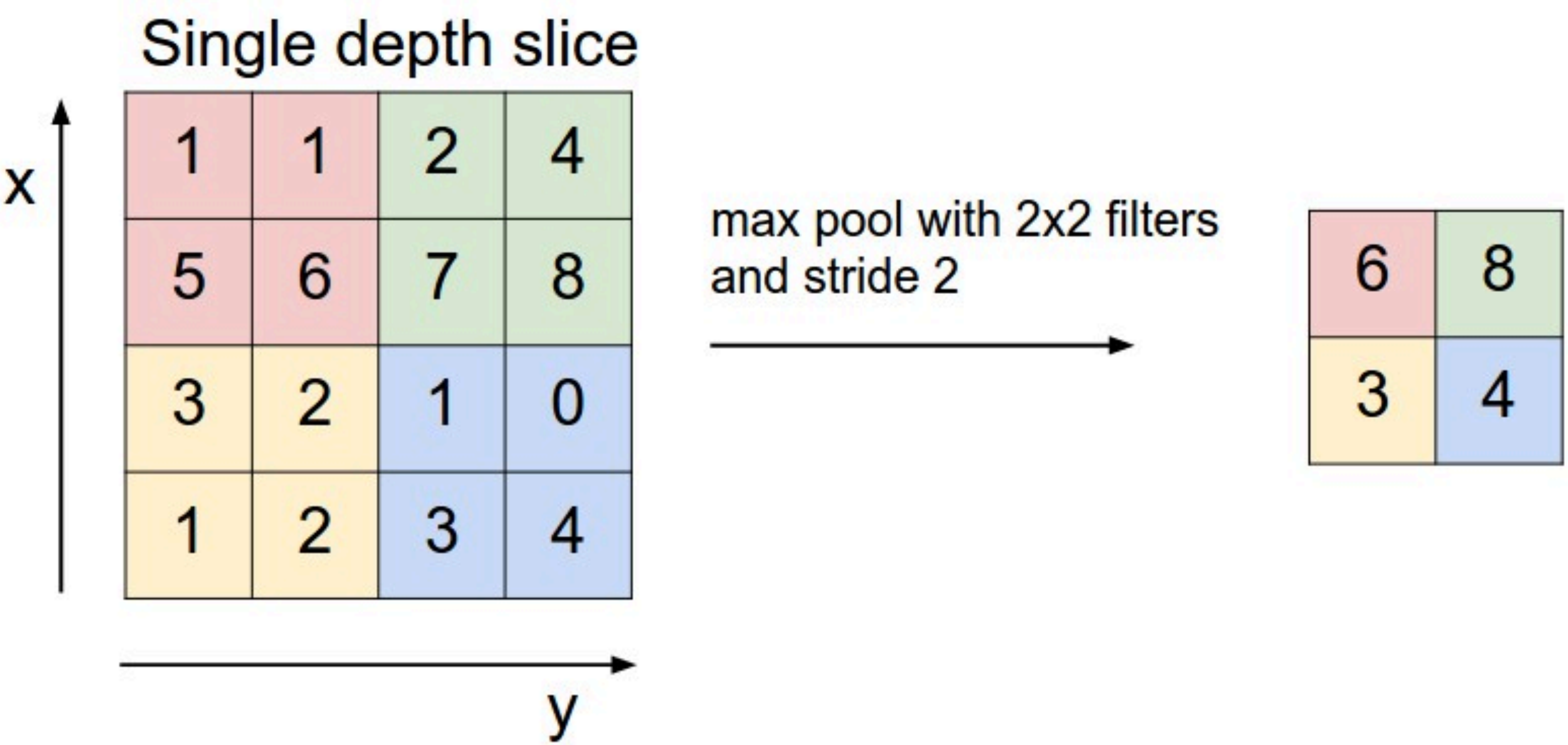
http://cs231n.github.io/convolutional-networks/

ConvNets



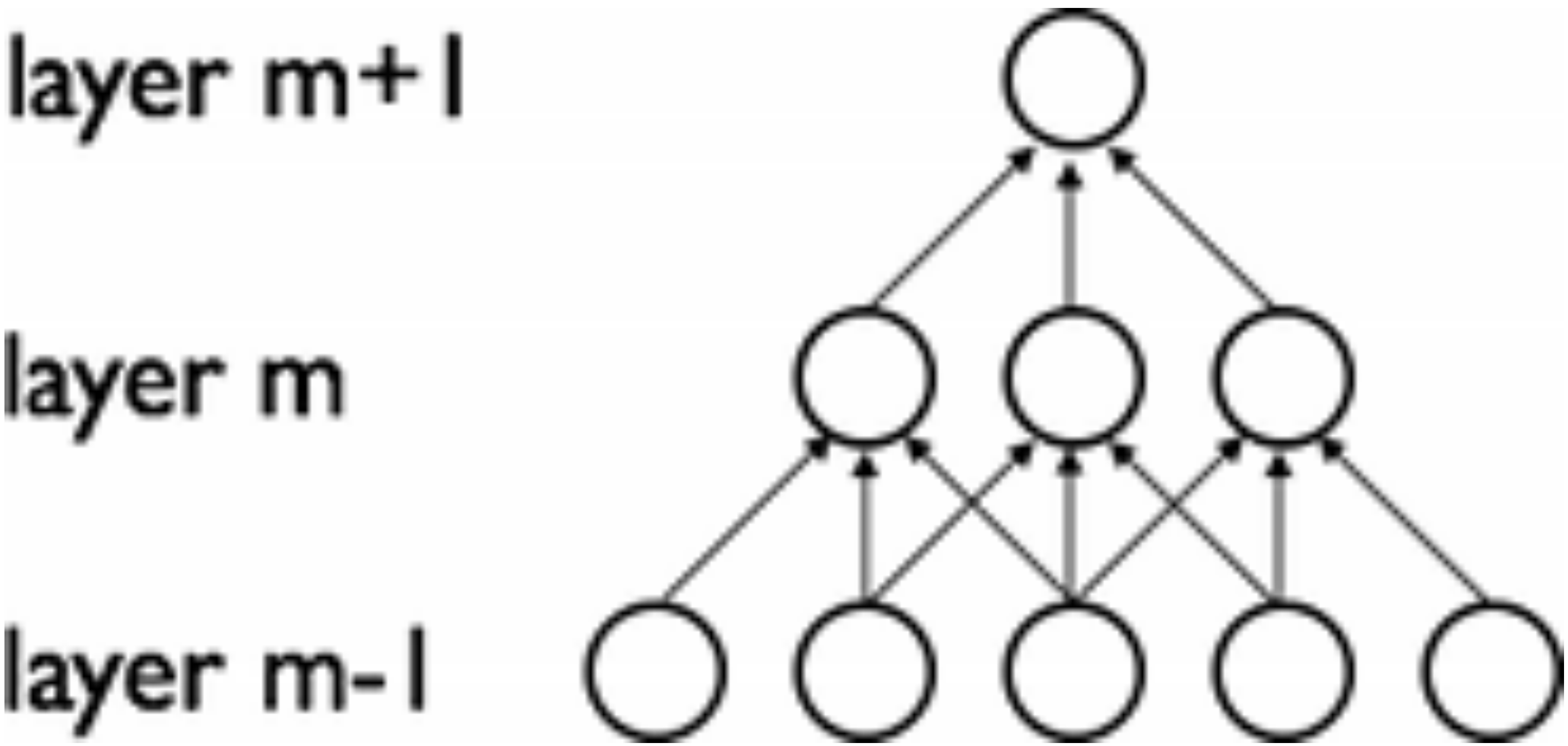
http://cs231n.github.io/convolutional-networks/

ConvNets
max pooling



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

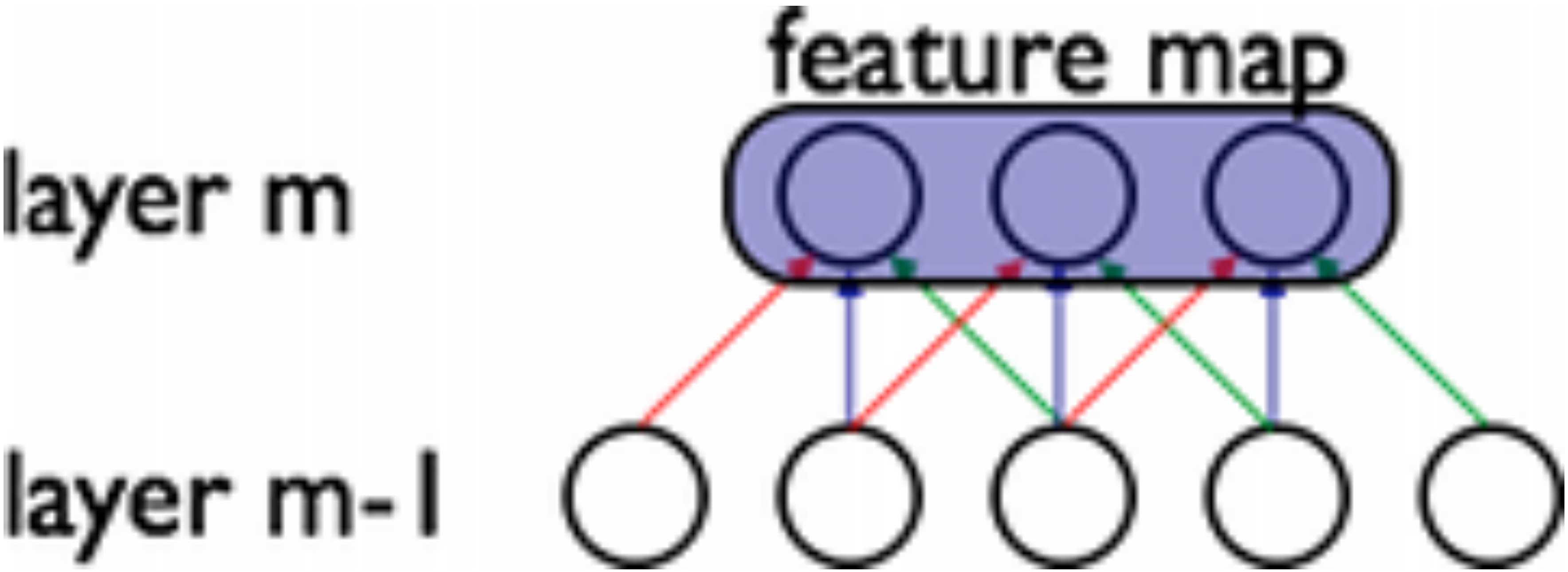
Convolutional Neural Networks (CNN) (LeNet)
Sparse Connectivity



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

Convolutional Neural Networks (CNN) (LeNet)

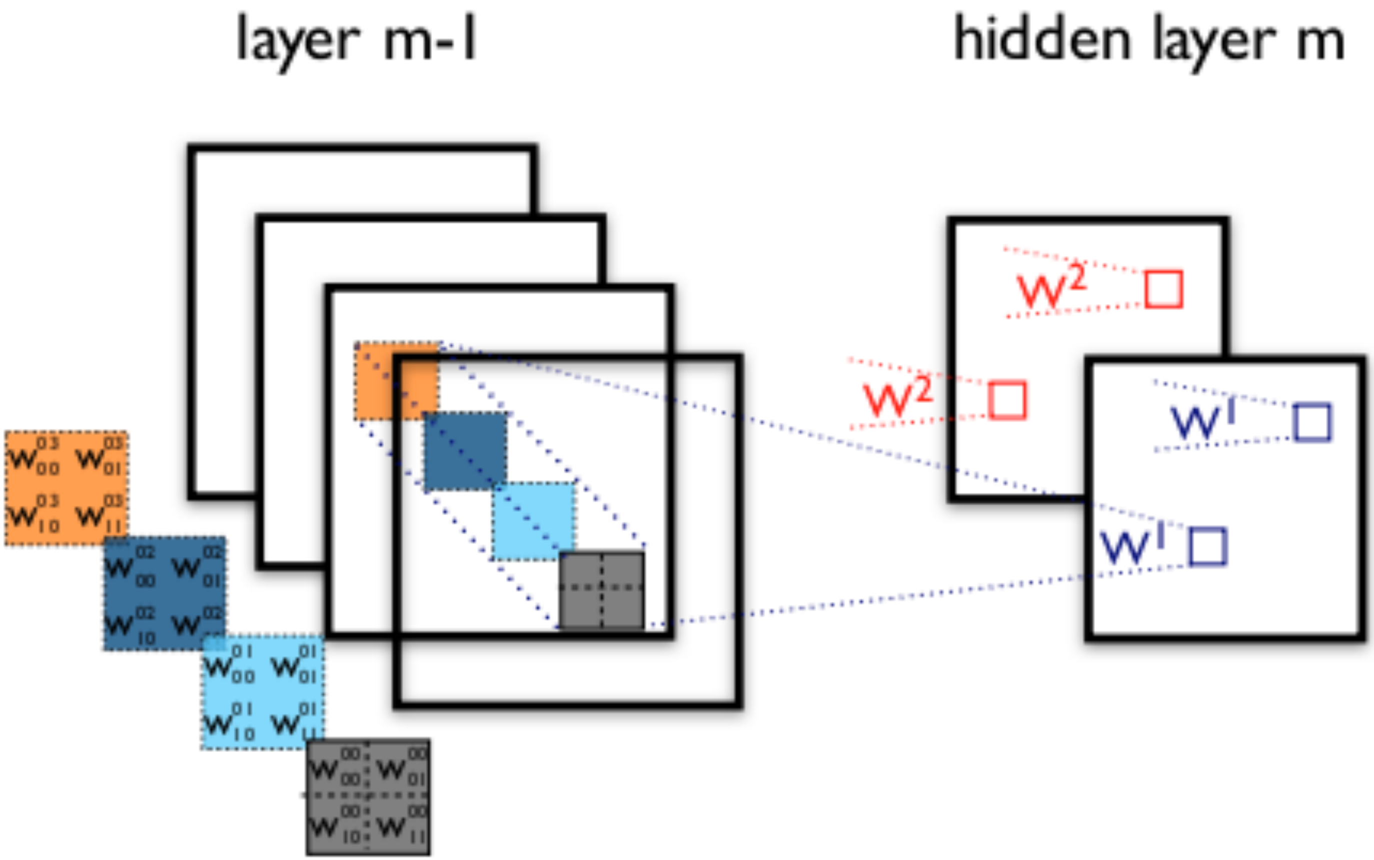
Shared Weights



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

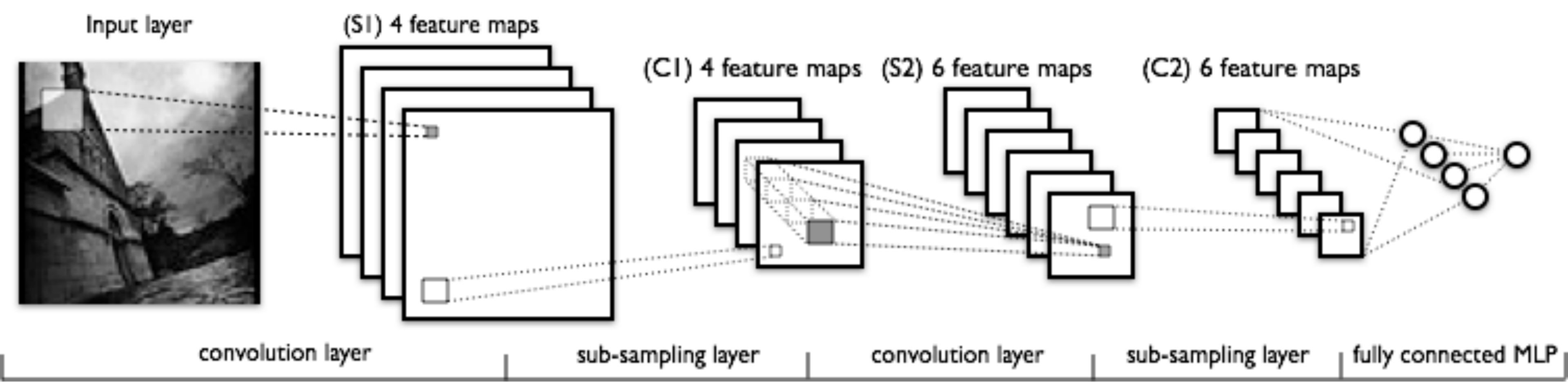
Convolutional Neural Networks (CNN) (LeNet)

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

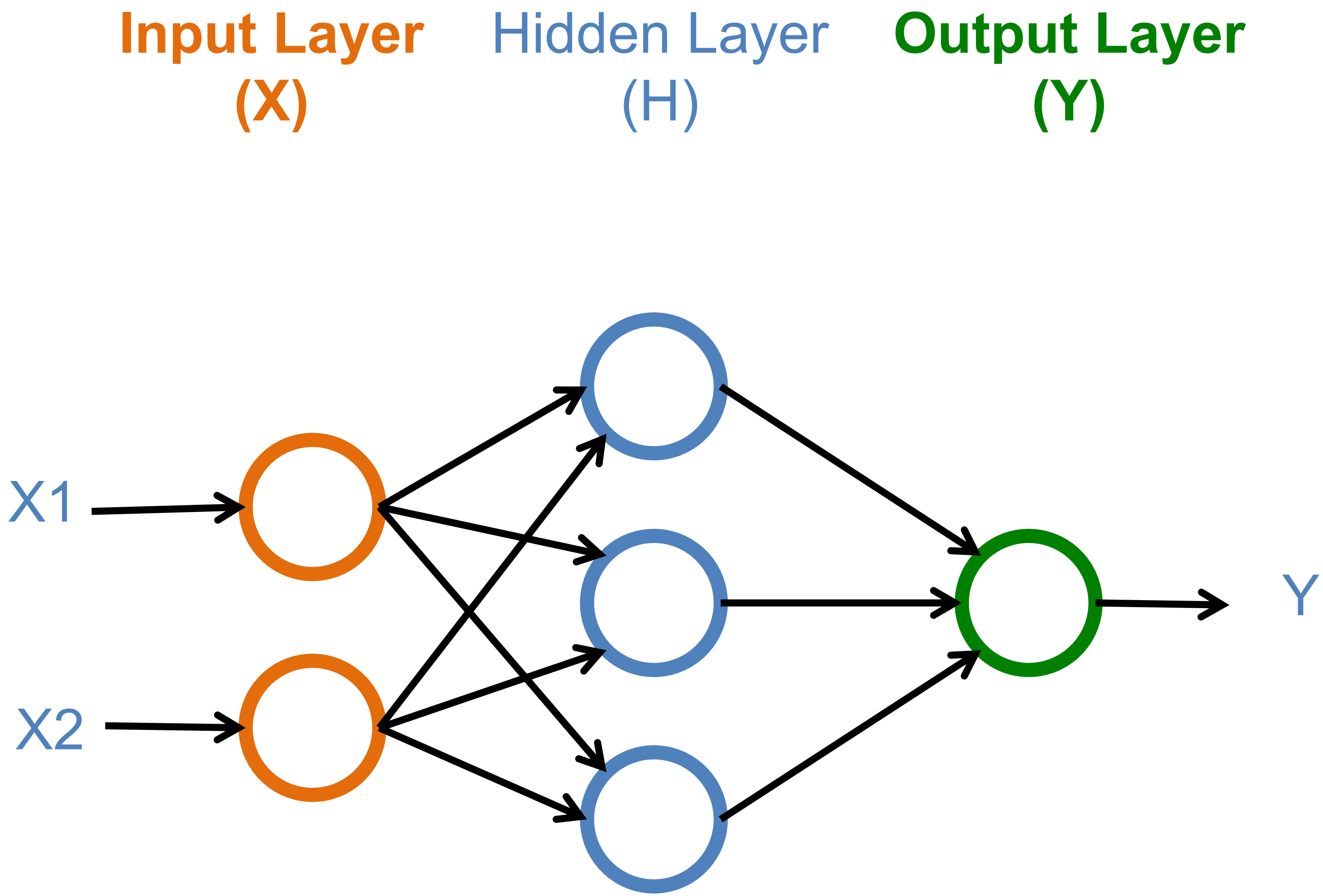
show flights from Boston to New York today

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



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	?

	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

$$Y = W X + b$$

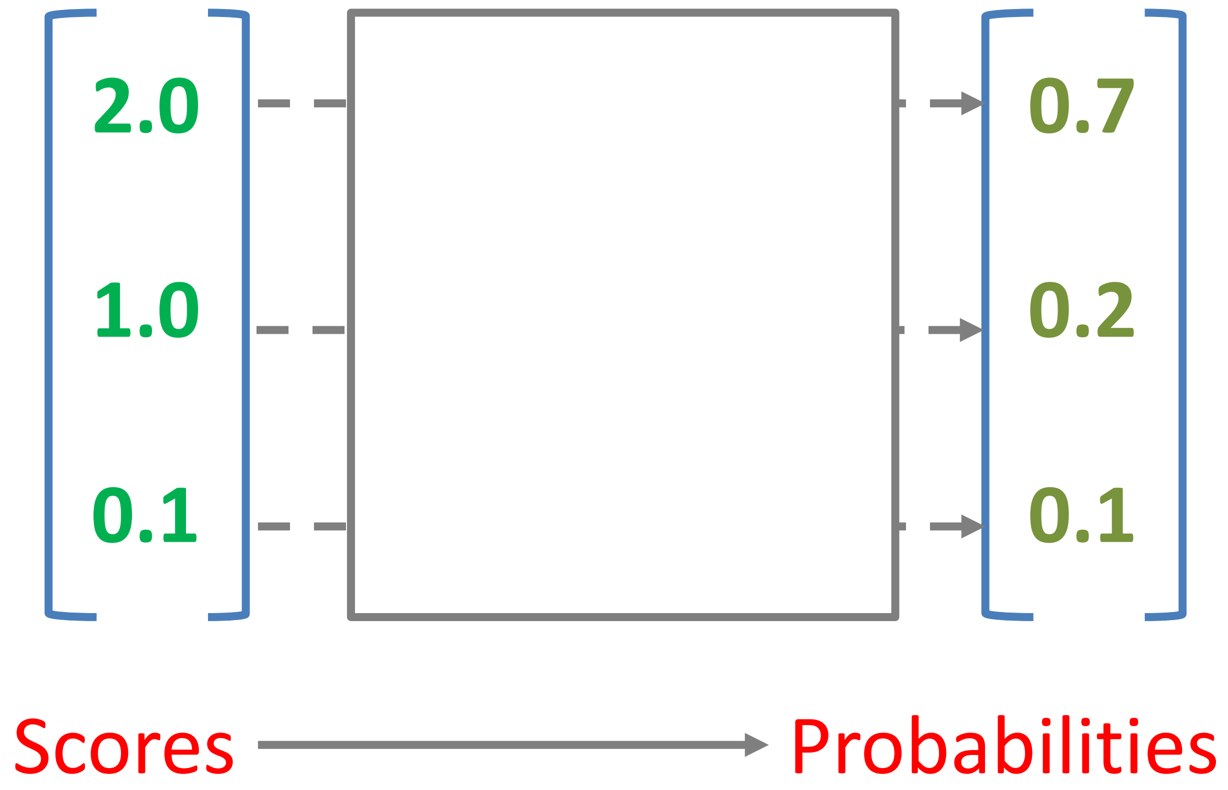
Weights

bias

Trained

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

$$WX + b = Y$$



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

SoftMAX

W X + b = Y

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

Logits

Scores → Probabilities

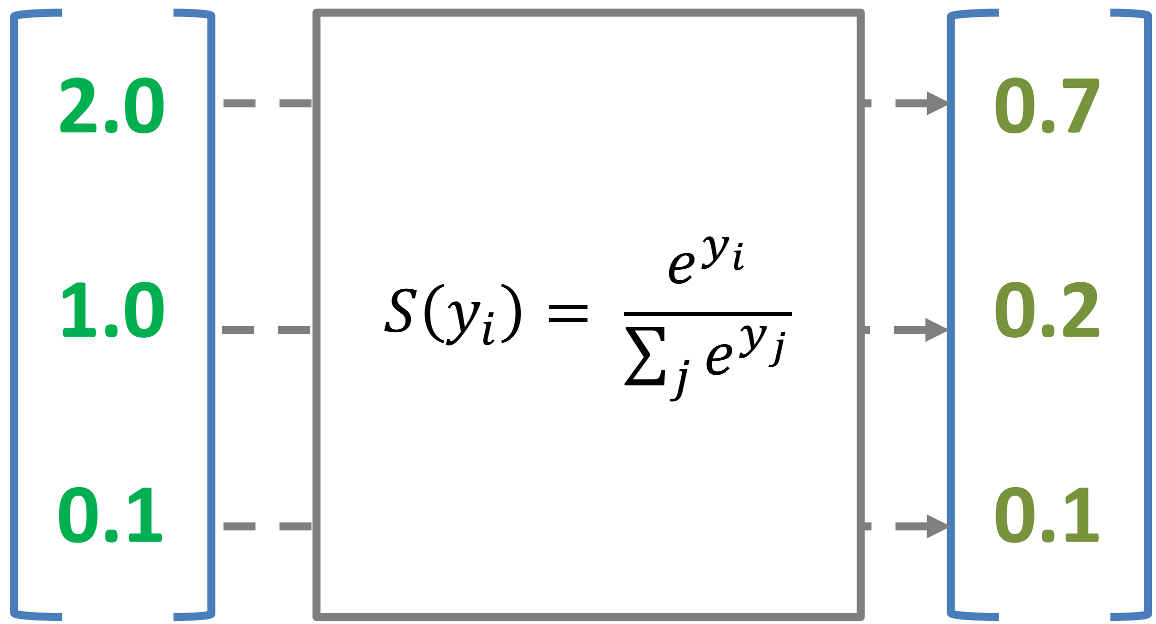
Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

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

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

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

W X + b = Y



Logits

Scores → Probabilities

Training a Network

=

Minimize the Cost Function

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

Training a Network

=

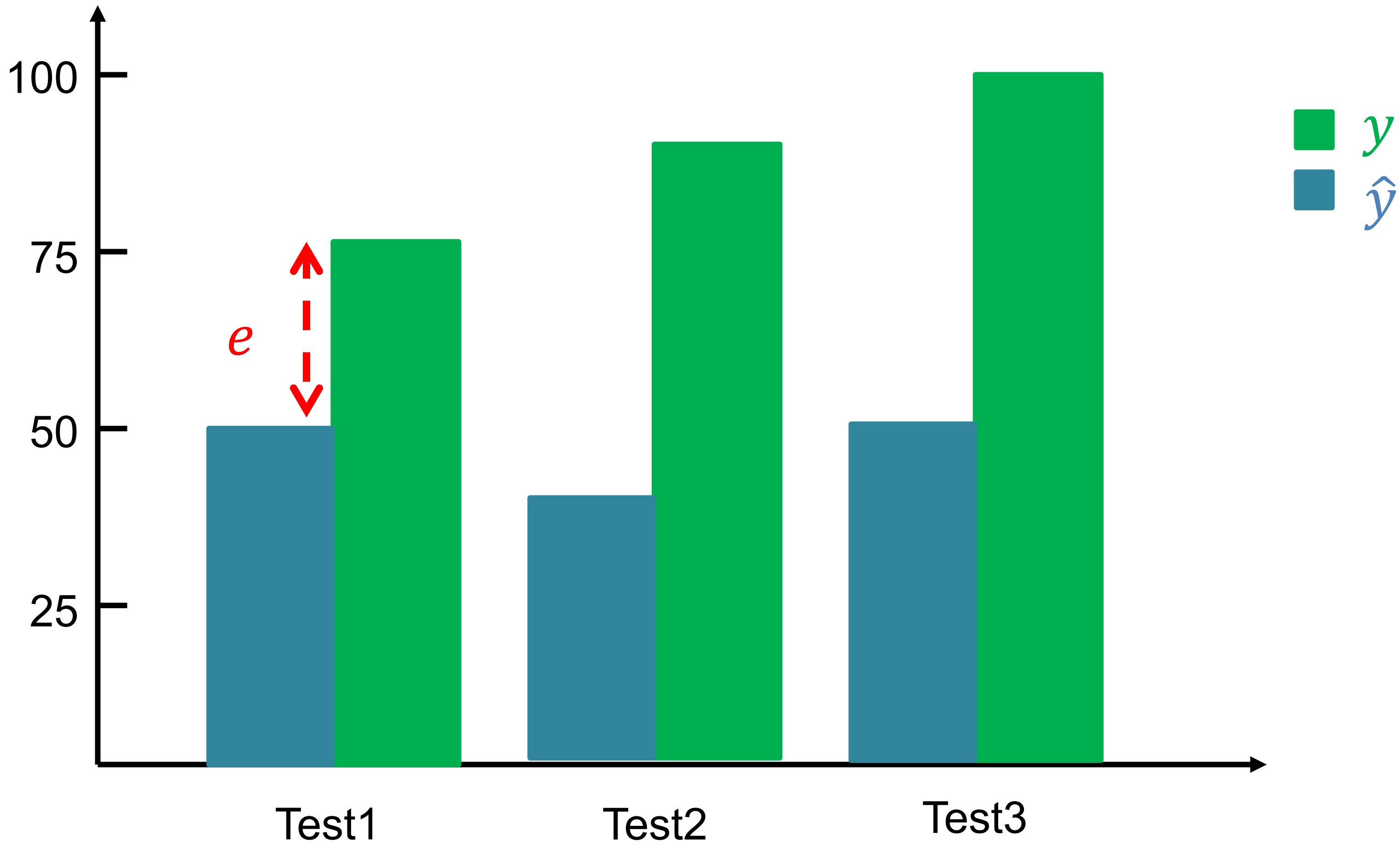
Minimize the Cost Function

Minimize the Loss Function

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

Error = Predict Y - Actual Y

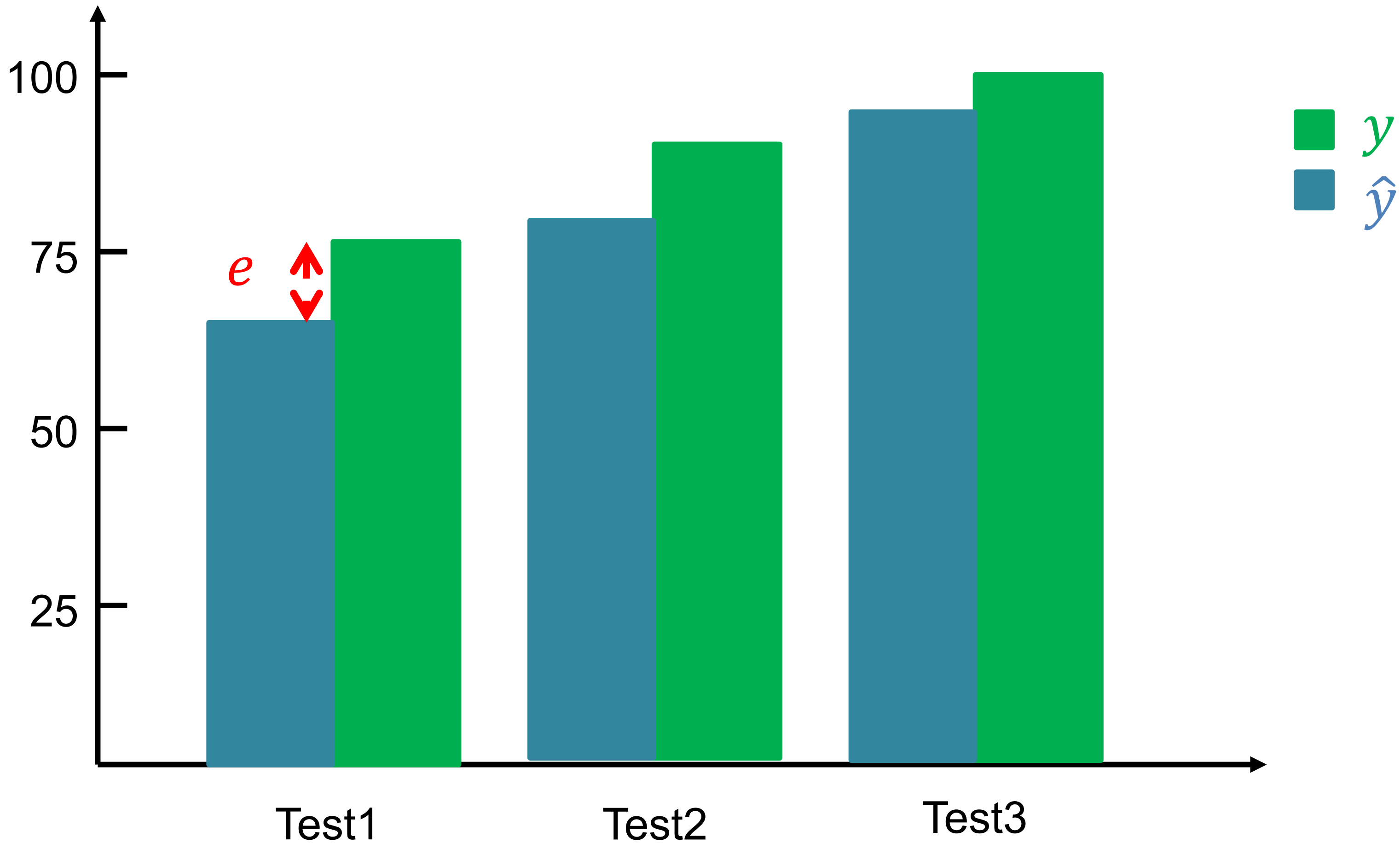
Error : Cost : Loss



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

Error = Predict Y - Actual Y

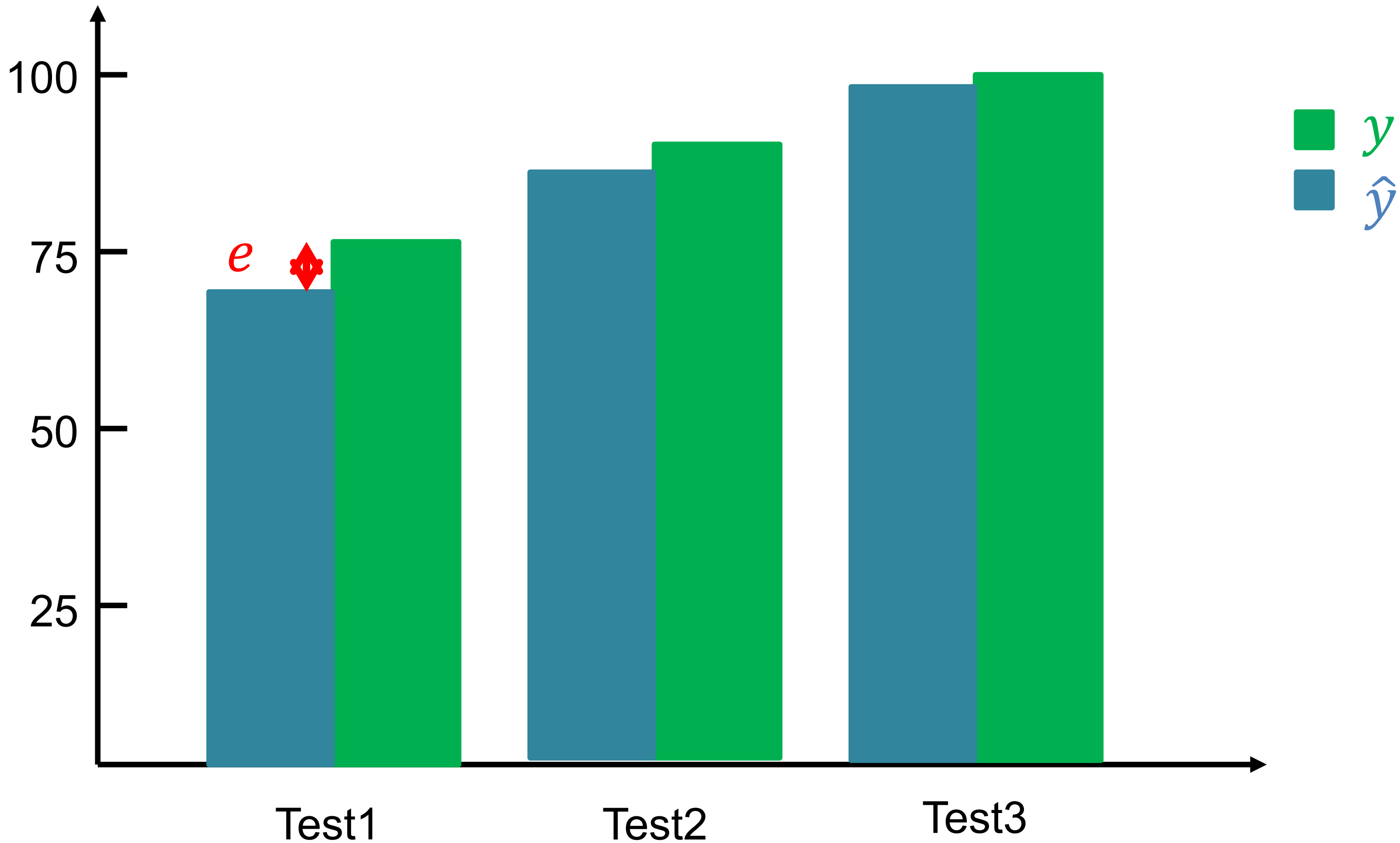
Error : Cost : Loss



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

Error = Predict Y - Actual Y

Error : Cost : Loss

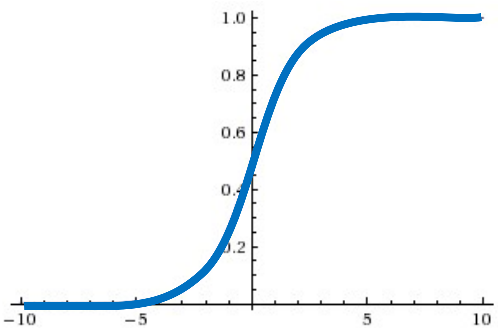


Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Activation Functions

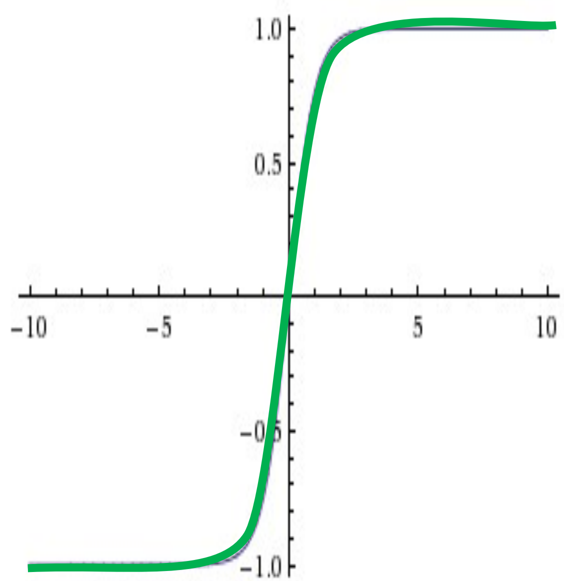
Activation Functions

Sigmoid



[0, 1]

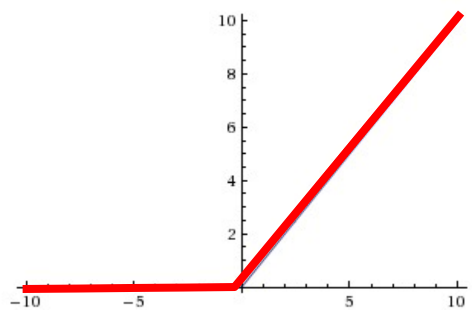
TanH



[-1, 1]

ReLU

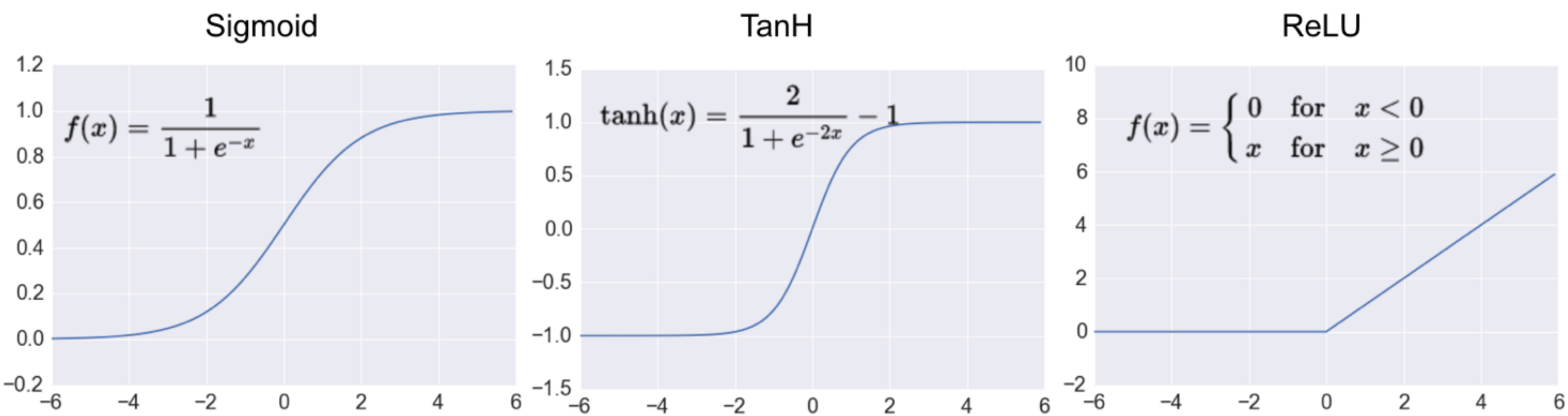
(Rectified Linear Unit)



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

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

Activation Functions



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

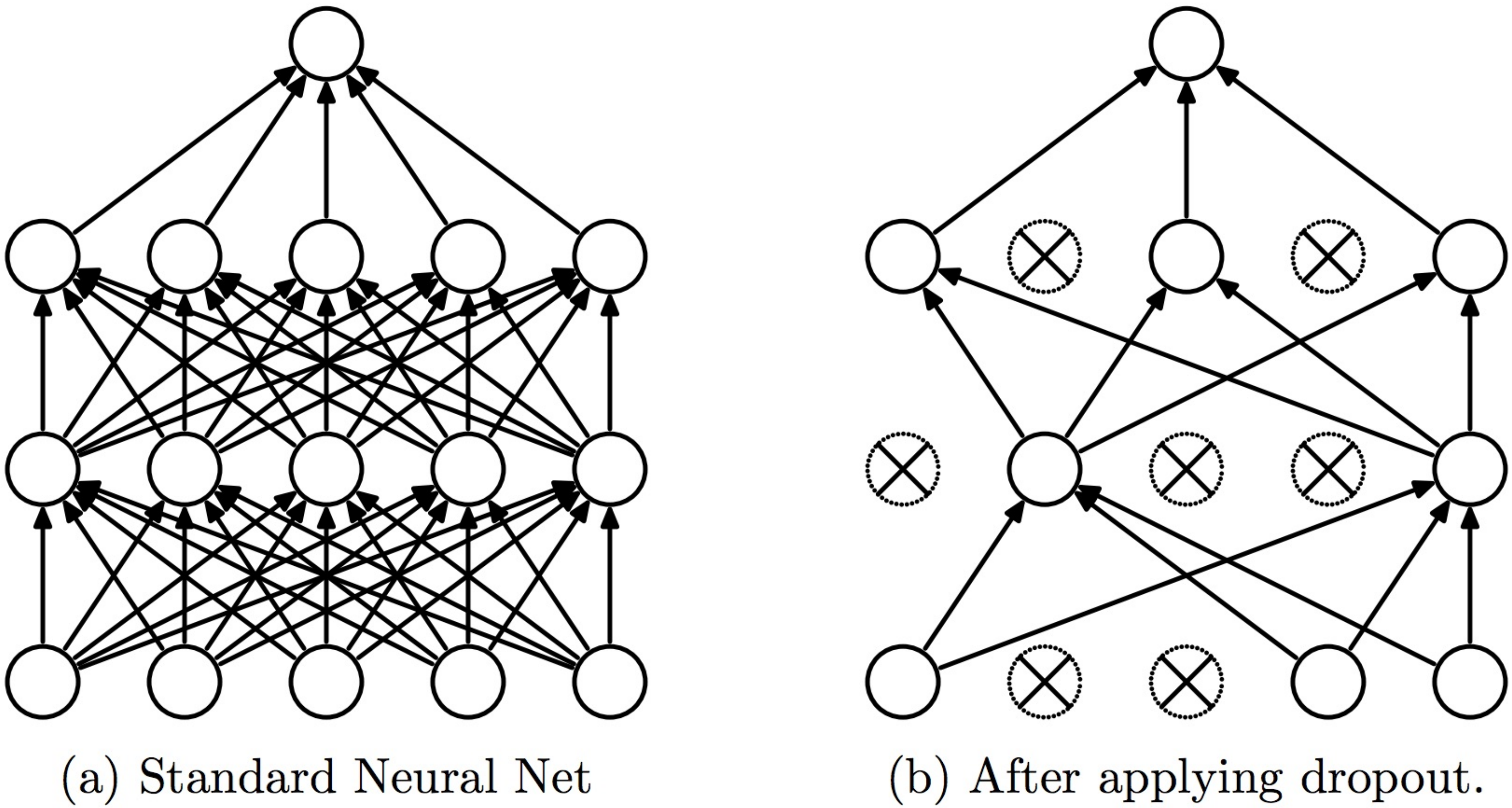
Multiple Classification: 10 Class

Activation Function:
SoftMAX

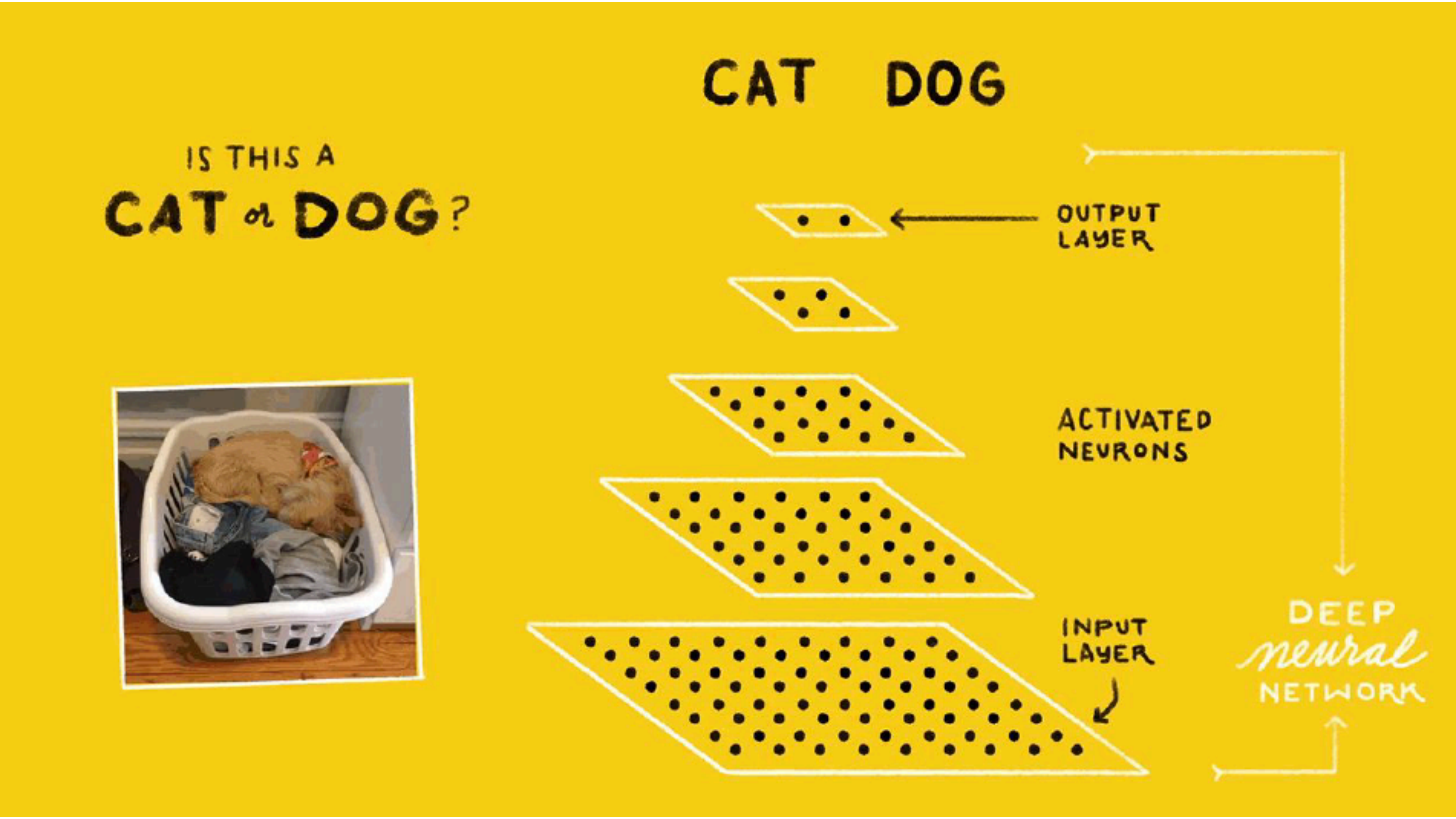
Loss Function:
Categorical Cross-Entropy

Dropout

Dropout: a simple way to prevent neural networks from overfitting



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.



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

Learning Algorithm

While not done:

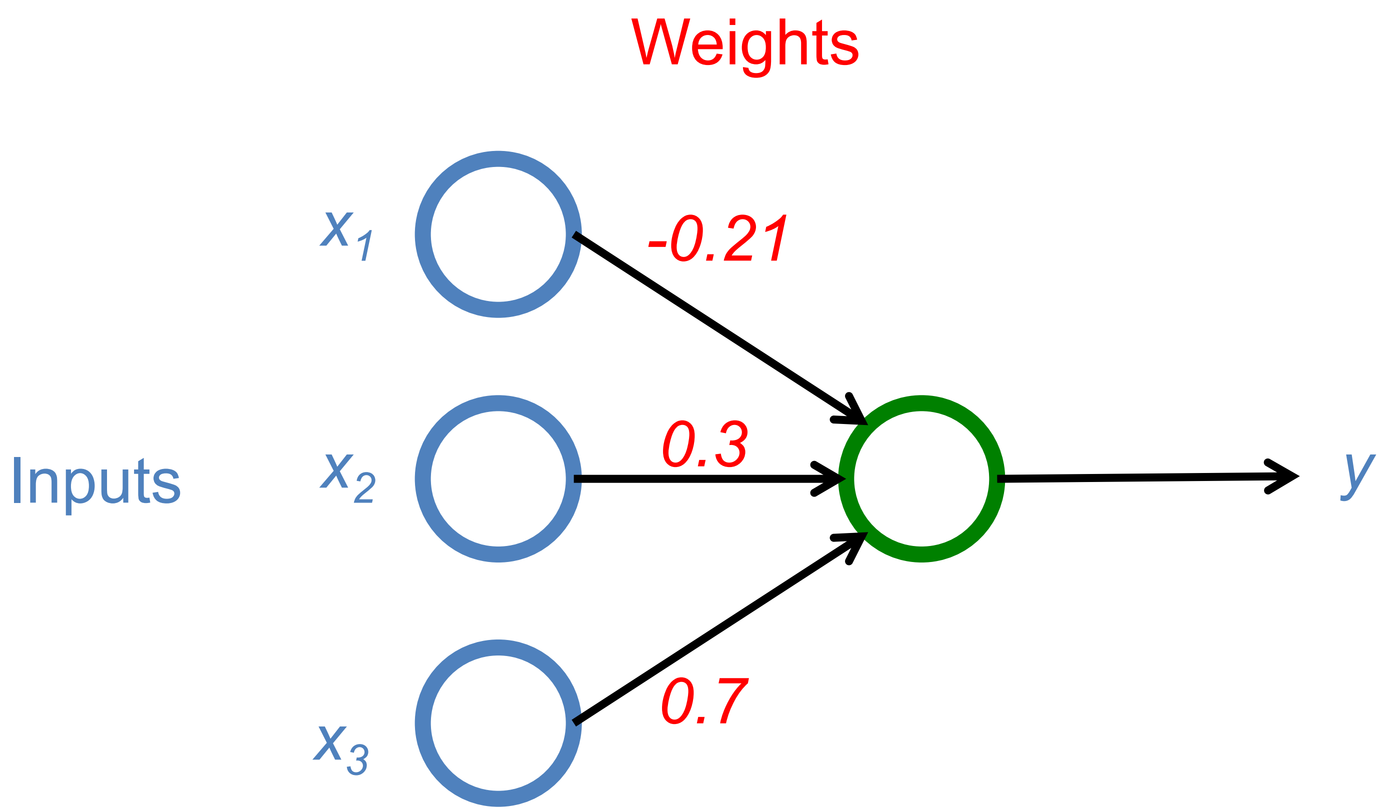
  Pick a random training example "(input, label)"

  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

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



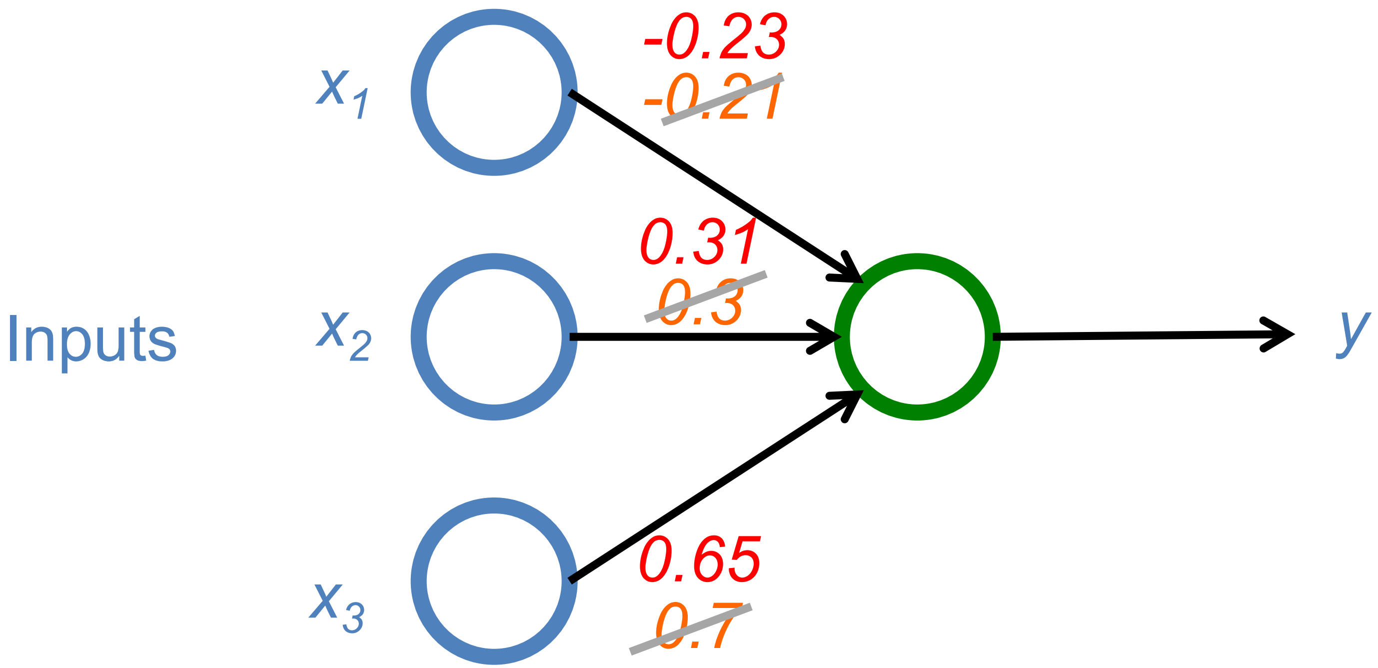
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Next time:

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

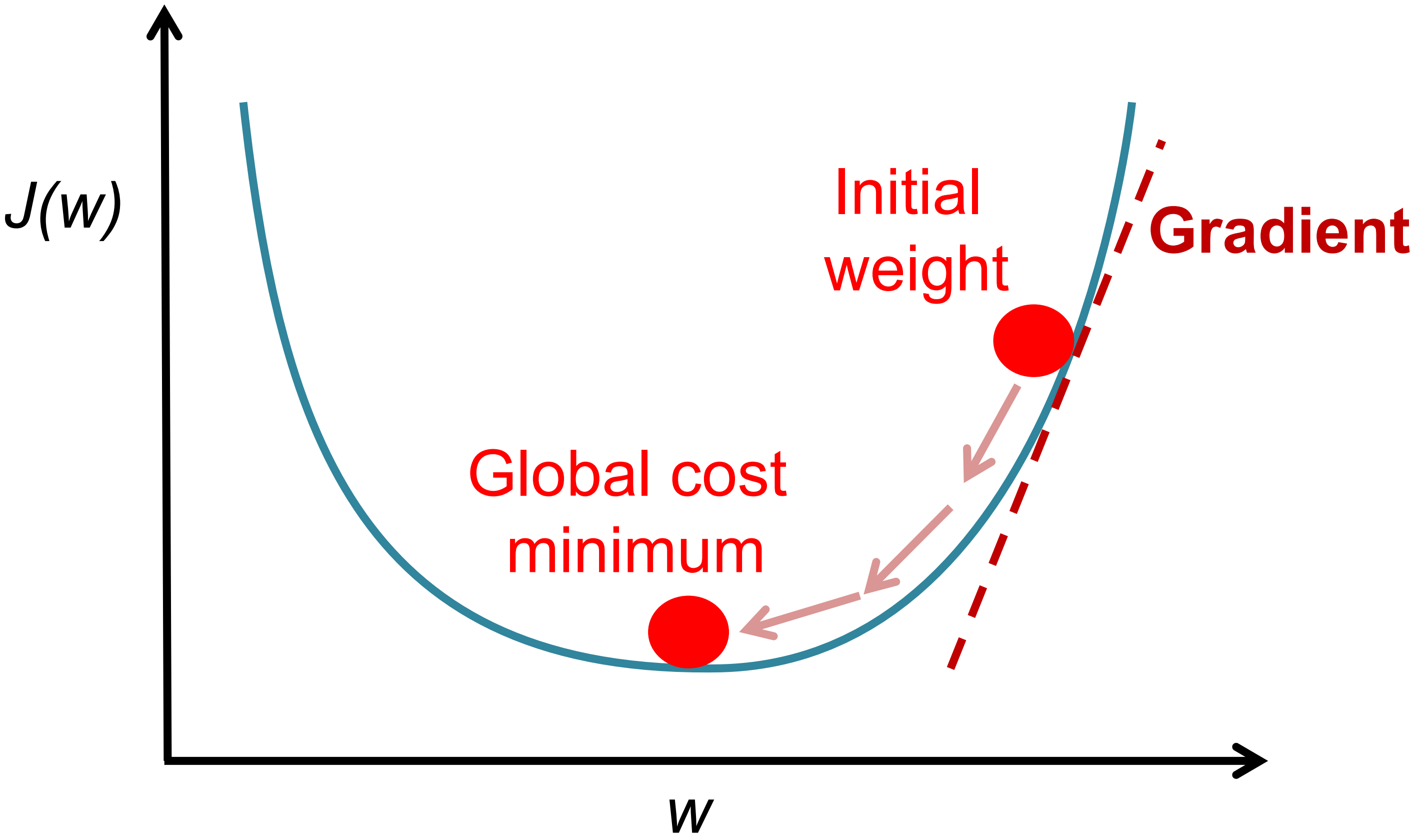
$$\cancel{y = max\ (\ 0,\ -0.21\ *\ x_1 + 0.3\ *\ x_2 + 0.7\ *\ x_3\ )}$$

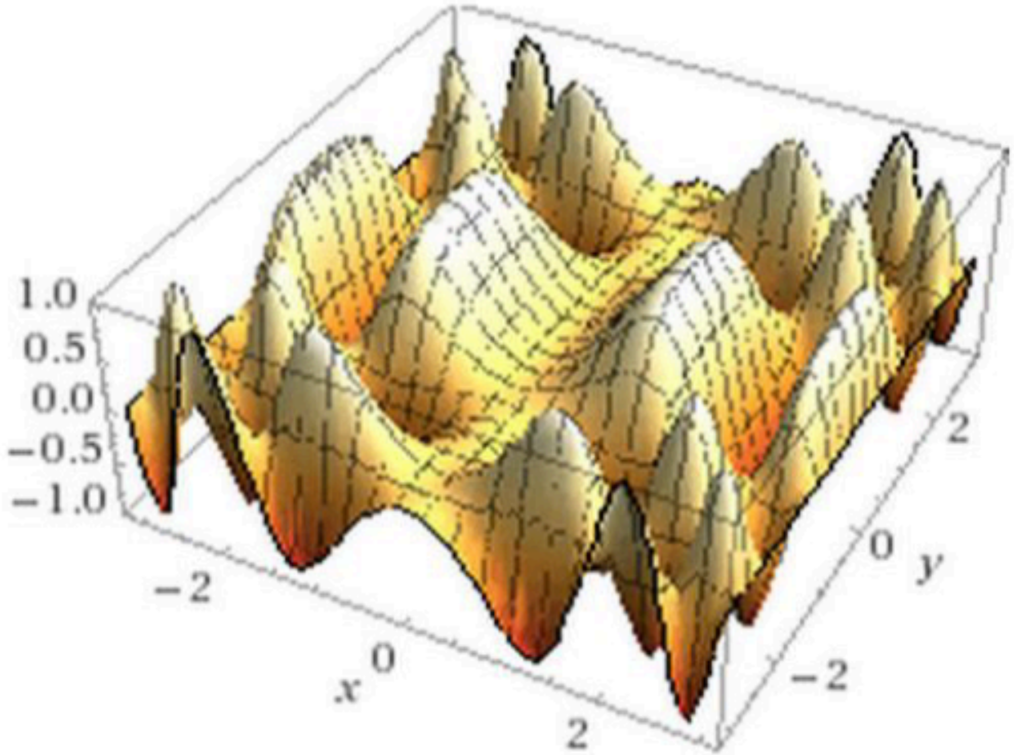
Weights



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

Optimizer:
Stochastic Gradient Descent (SGD)

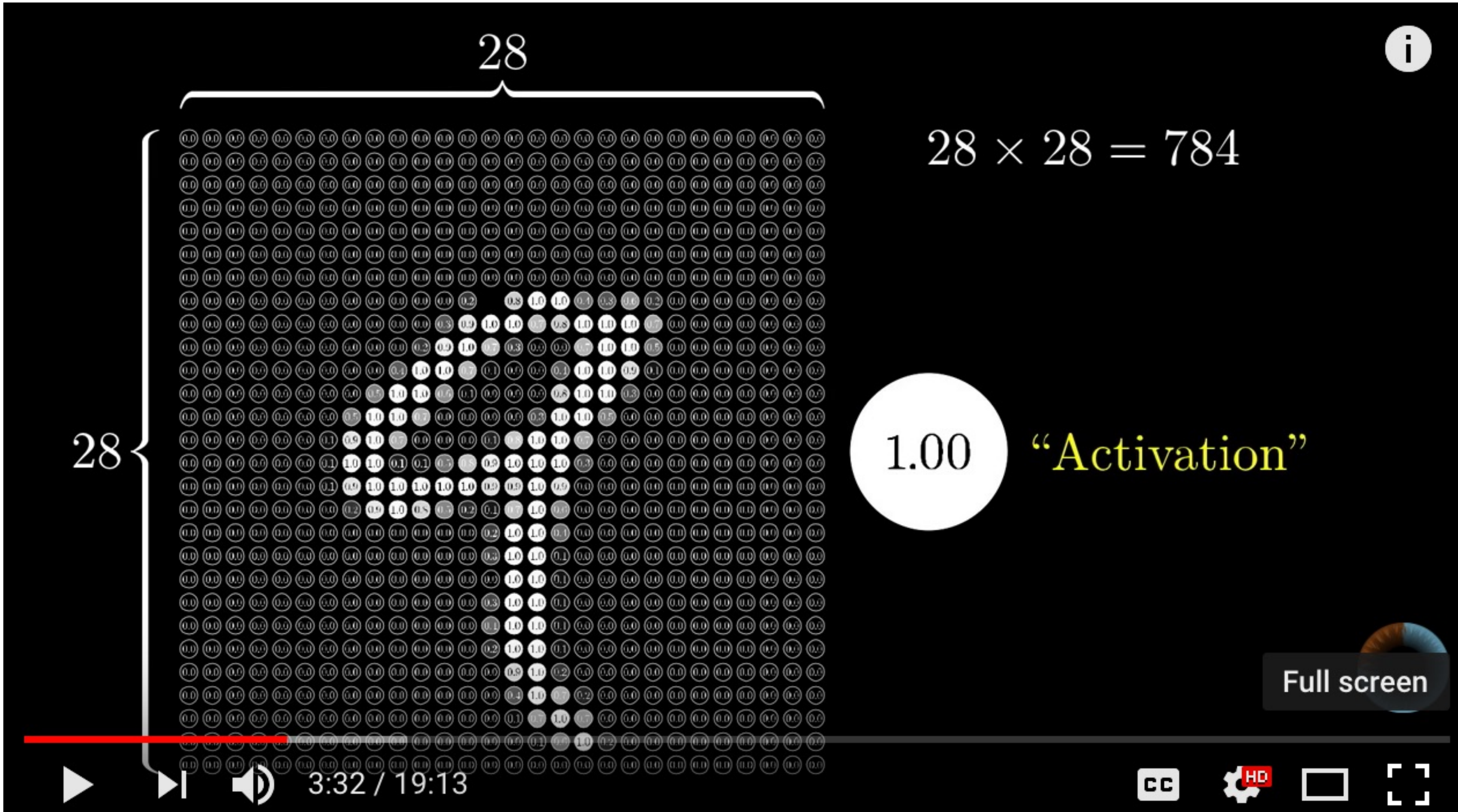




This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

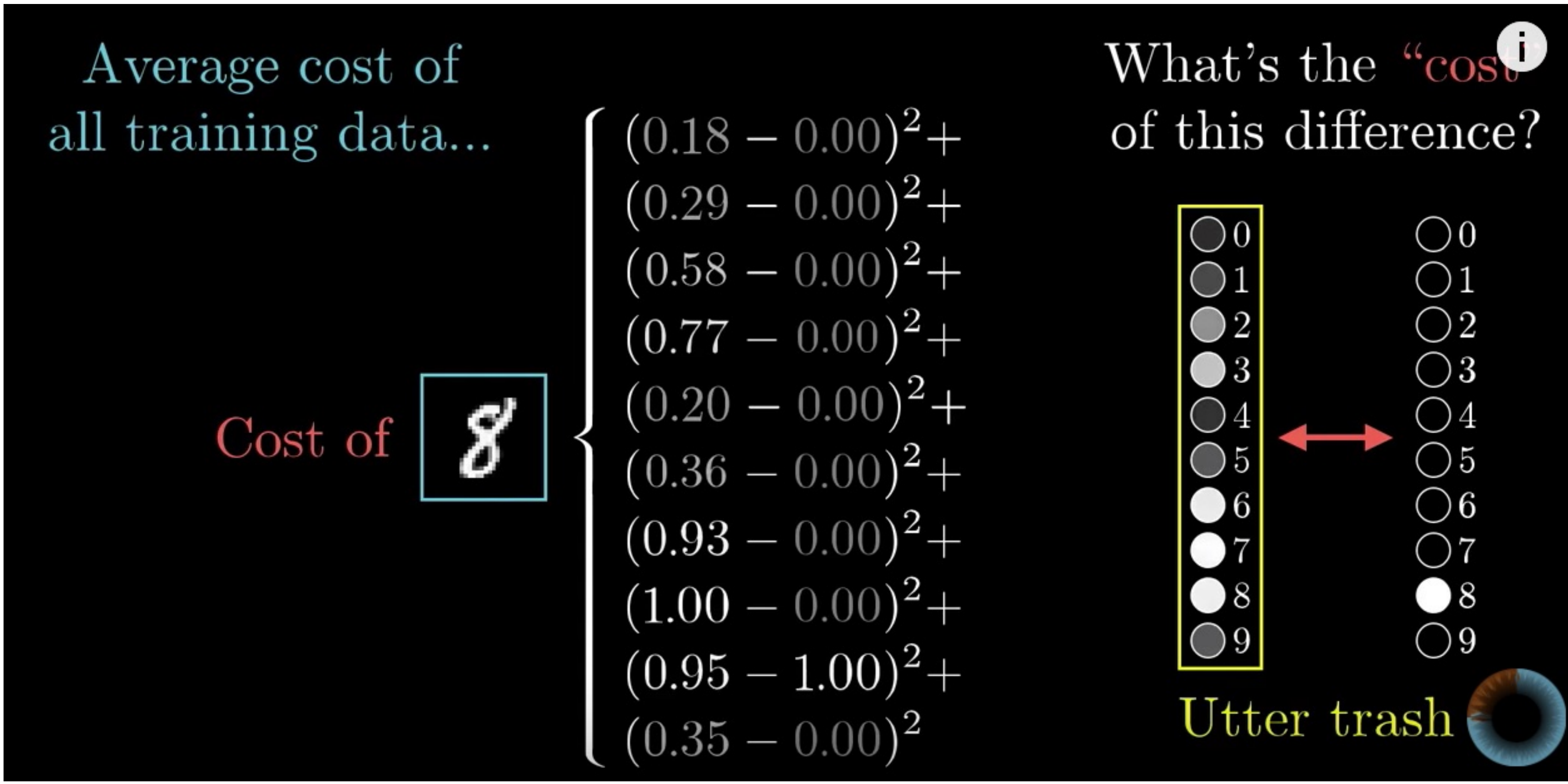
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Network and Deep Learning



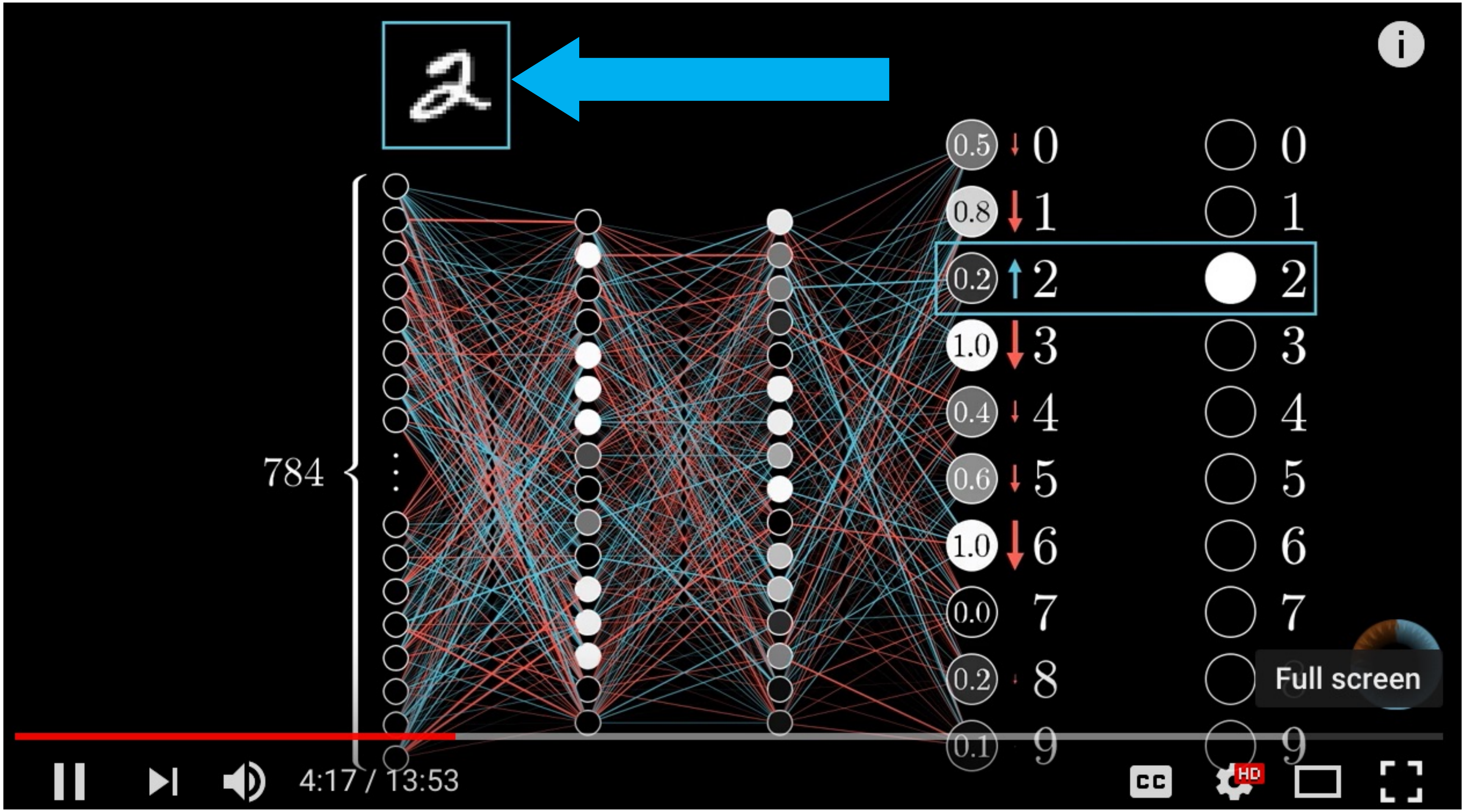
Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

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

Important Property of Neural Networks

Results get better with

More data +

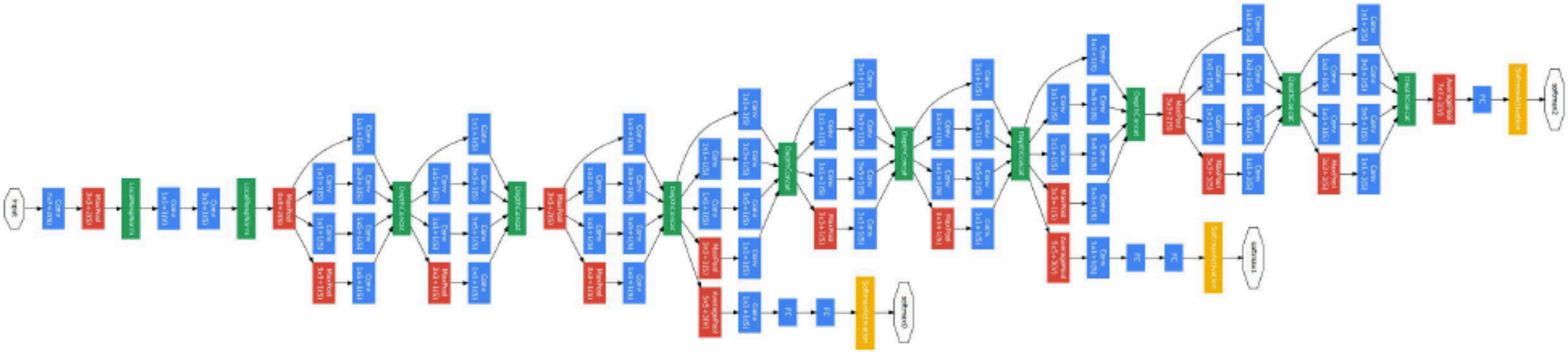
Bigger models +

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)



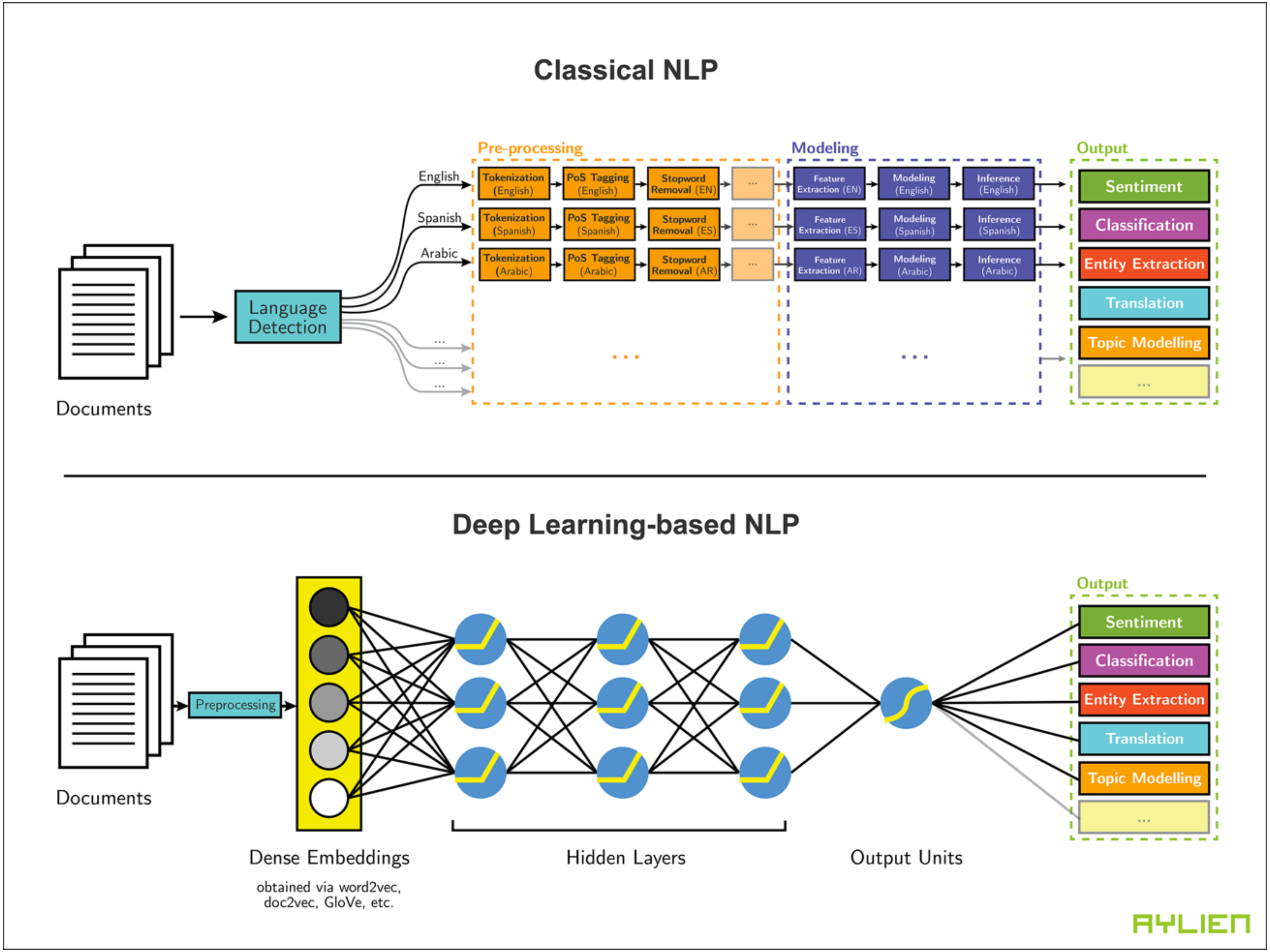
Going Deeper with Convolutions

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

ArXiv 2014, CVPR 2015

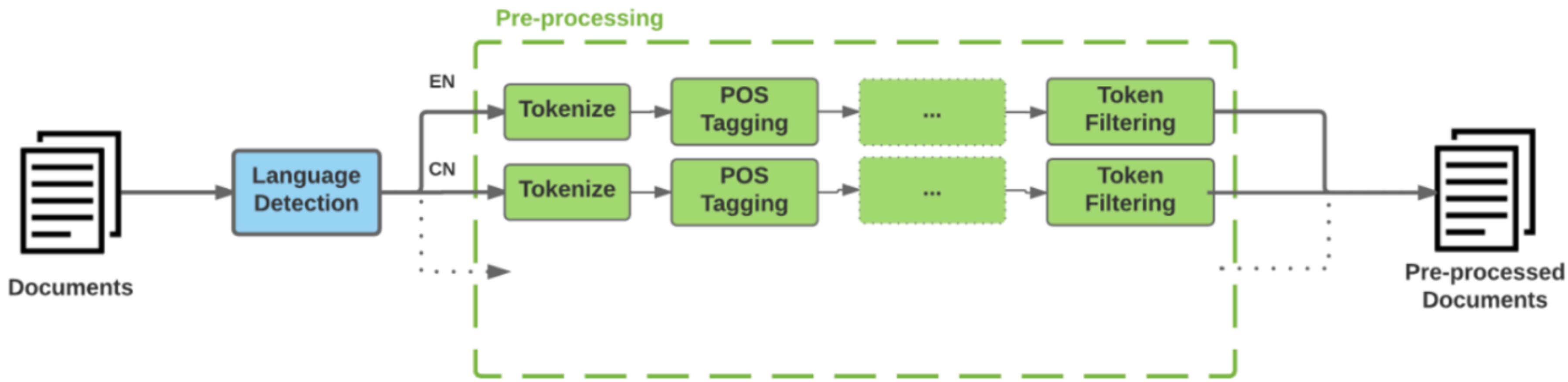
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

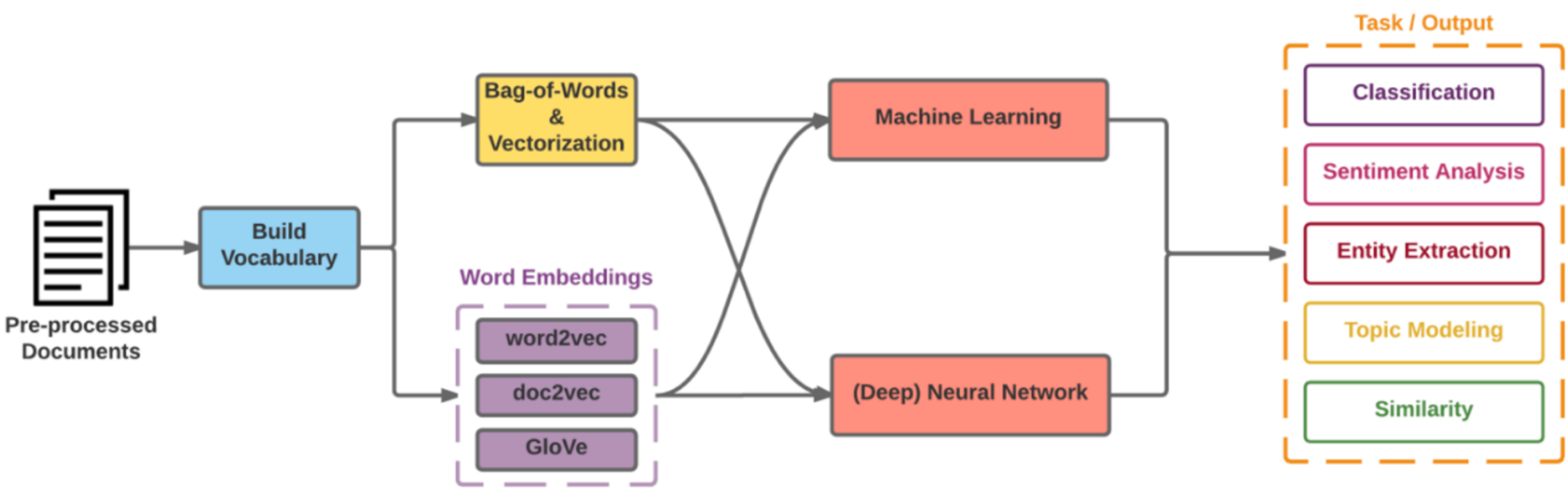
NLP



Source: http://blog.aylien.com/leveraging-deep-learning-for-multilingual/

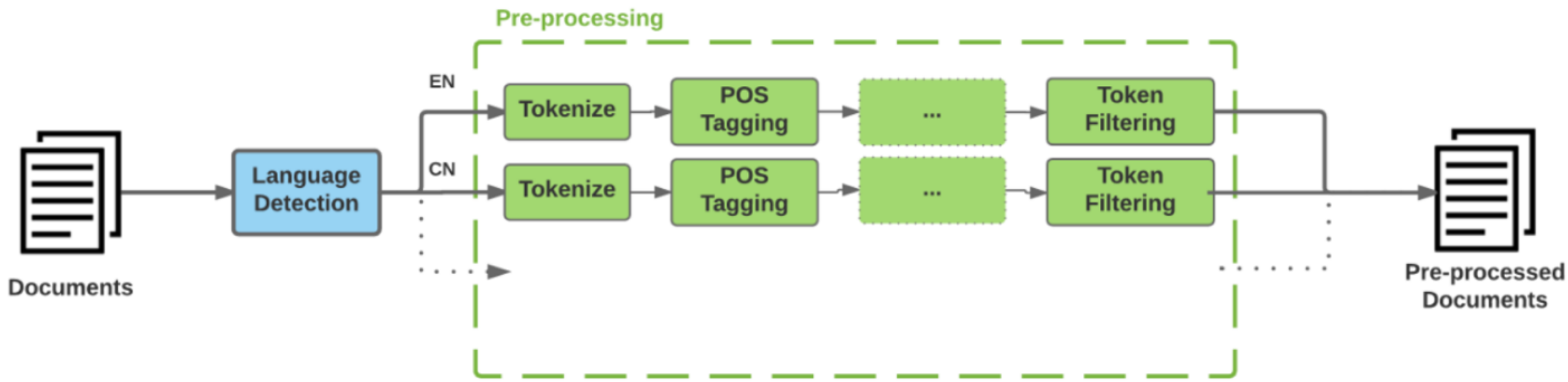
Modern NLP Pipeline

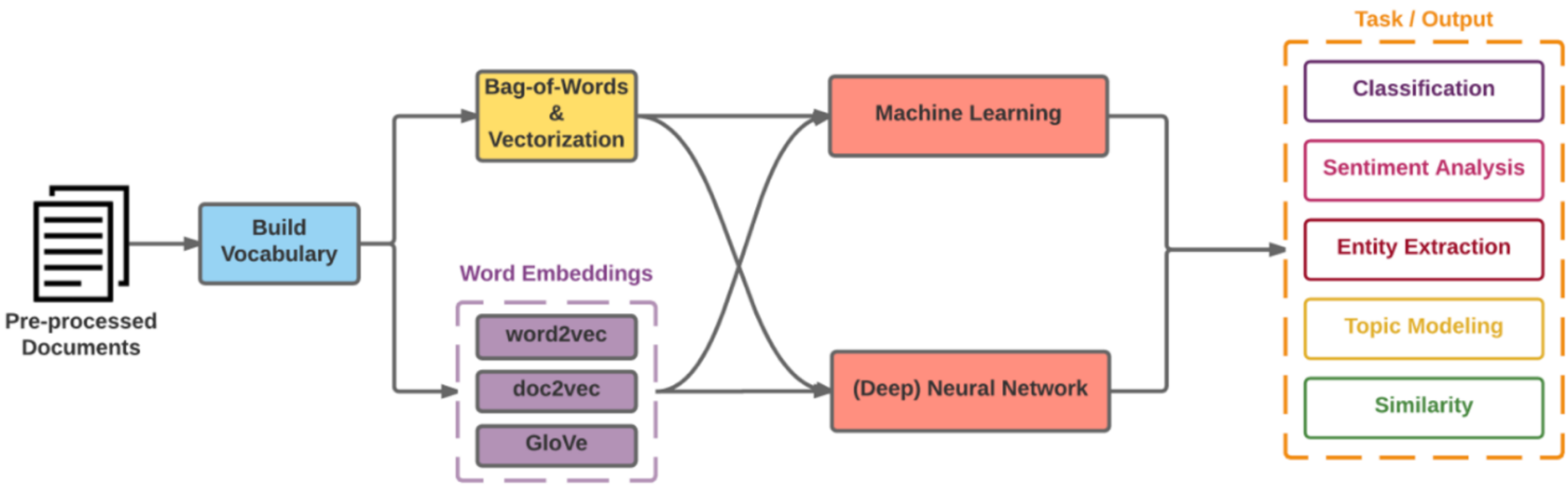




Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

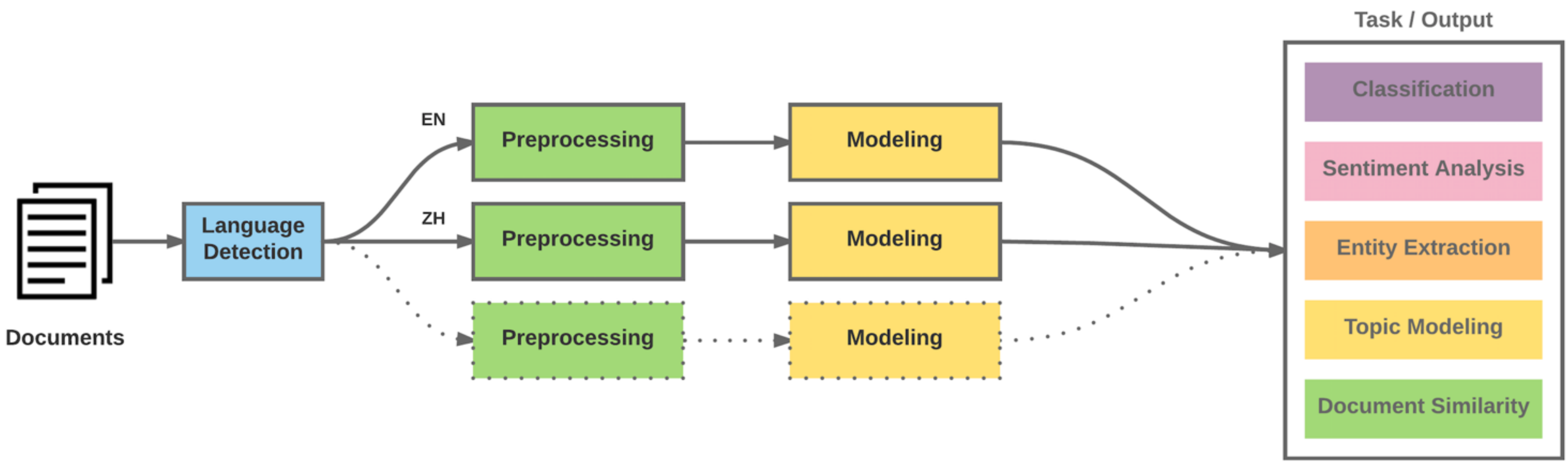
Modern NLP Pipeline





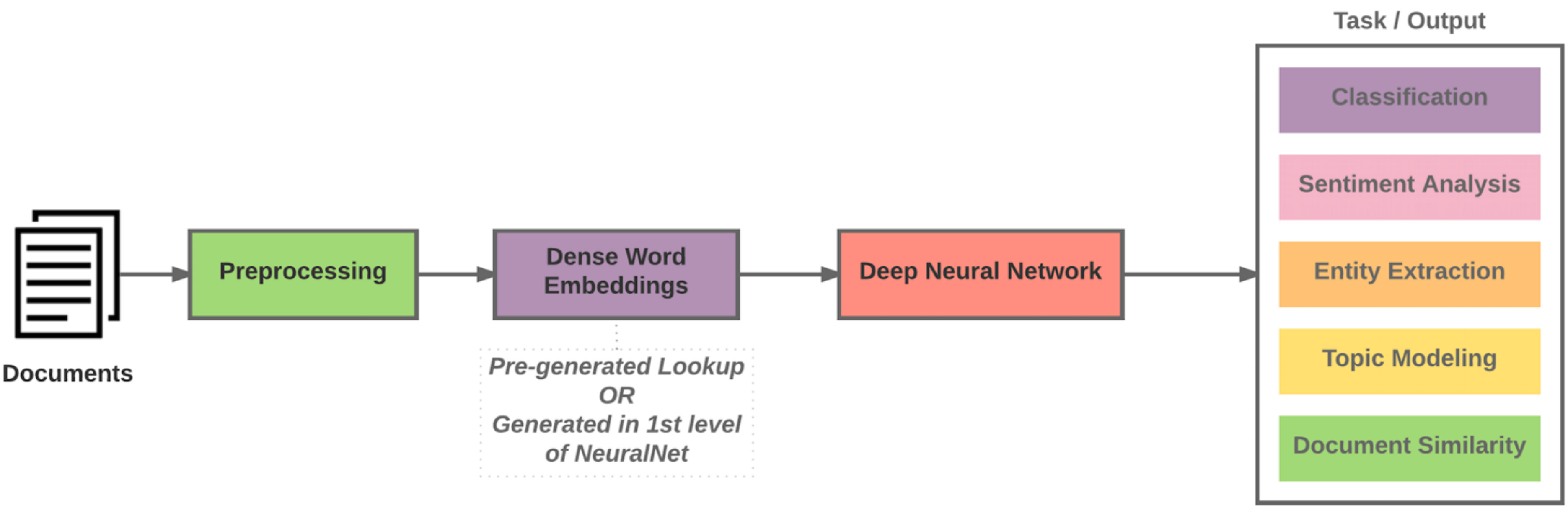
Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Modern NLP Pipeline



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Deep Learning NLP



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

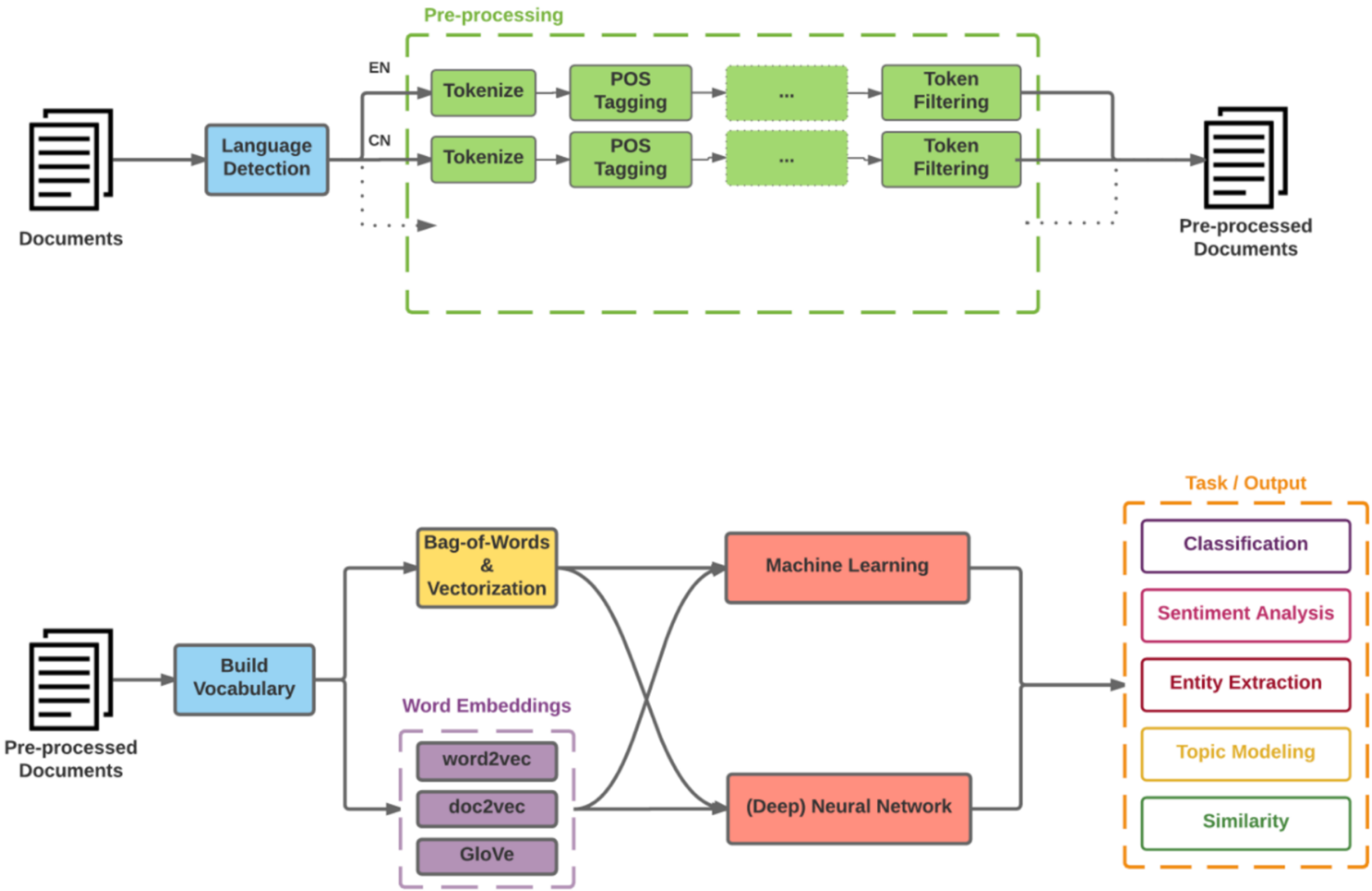
Vector Representations of Words

Word Embeddings

Word2Vec

GloVe

Modern NLP Pipeline



Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Facebook Research FastText

Pre-trained word vectors
Word2Vec
wiki.zh.vec (861MB)
332647 word
300 vec

Pre-trained word vectors for 90 languages, trained on Wikipedia using fastText.

These vectors in dimension 300 were obtained using the skip-gram model with default parameters.

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Source: Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." *arXiv preprint arXiv:1607.04606* (2016).

Facebook Research FastText Word2Vec: wiki.zh.vec

(861MB) (332647 word 300 vec)

wiki.zh.vec

```
yg -0.3978 0.49084 -0.54621 0.078991 0.8584 -0.26163 -0.45787 0.060828 0.36513 -0.03771 0.80791 0.16613 1.4828 -0.89862 0.085965
迴圈 -0.034834 0.71651 -0.4377 0.48344 0.31117 -0.51783 -0.40156 -0.057097 0.31535 -0.088301 0.23436 0.30884 1.2932 -0.6704 0.21
ぶっ -0.23267 0.39349 -0.90806 -0.53805 0.59308 -0.31819 -0.64229 0.16871 0.10086 0.09342 1.0914 -0.16019 1.6954 -0.70604 -0.218
三公 0.54129 0.55641 -0.4348 0.25094 0.1631 -0.10326 -0.54099 0.064742 0.13175 0.10217 0.84938 -0.10287 1.312 -0.74969 0.24025 -
水貨 -0.14451 0.80455 -0.6145 0.55905 0.58307 -0.02559 -0.41088 -0.19056 -0.09178 0.33935 1.1927
刚才 0.19347 0.553 -0.64736 0.26358 0.83816 -0.24098 -0.83997 -0.16232 -0.024786 -0.2483 0.69732
無知 -0.0089777 0.90866 -0.25306 0.72983 0.67791 -0.3285 -0.63835 0.075295 0.4774 -0.04134 0.721
好轉 -0.026068 0.92676 -0.47469 0.50129 0.67343 -0.32509 -0.32917 0.066499 0.3875 0.0011722 0.66
紀事 0.40541 0.67654 -0.5351 0.30329 0.43042 -0.24675 -0.19287 0.34207 0.35516 -0.076331 0.85916
變回 -0.089933 0.88136 -0.43524 0.59963 0.6403 -0.70981 -0.56788 -0.074018 0.16905 -0.086594 0.6
牟尼 -0.26578 0.6434 0.028982 -0.044001 0.88297 -0.17646 -0.64672 0.040483 0.43653 0.084908 0.74
埋藏 -0.0985 0.85082 -0.33363 0.24784 0.71518 -0.59054 -0.73731 0.050949 0.36726 -0.076886 0.817
正大 0.21069 0.27605 -0.83862 -0.099698 0.47894 -0.32196 -0.38288 -0.01892 0.40548 -0.029619 0.7
kis -0.30595 0.18482 -0.71287 -0.314 0.44776 -0.44245 -0.36447 -0.23723 0.00098801 -0.2528 0.608
合奏 0.1841 0.60874 -0.51376 -0.48002 0.21506 -0.55515 -0.71746 0.030735 0.39508 -0.40856 0.6226
精兵 0.25619 0.77186 -0.48847 0.23118 0.27254 0.21305 -0.3517 0.47305 0.24882 -0.34756 1.025 0.1
疲勞 -0.072521 1.0381 -0.51933 0.19421 0.67573 -0.45204 -0.20126 0.22704 0.44196 0.018401 0.3473
襯 -0.11771 1.4272 -1.0849 0.77532 0.87026 -0.6892 -0.3521 0.036517 0.42727 -0.1871 0.82789 -0.0
小貓 -0.21554 0.73988 -0.39628 0.044656 1.0602 -0.67047 -0.54102 0.11888 0.1693 0.19343 1.0841 0.
lai -0.25451 0.31596 -0.29228 -0.19144 0.99059 -0.24459 -0.66342 0.063093 -0.061142 -0.22749 0.6
偏東 -0.50835 1.0943 0.043918 0.29173 1.0161 -0.32493 -0.27305 0.026946 0.46811 -0.3874 1.4049 0.
大约是 -0.35726 -0.03476 -0.28672 0.075447 0.18175 -0.39421 -0.32088 0.025225 0.34808 0.074744 0.
franch -0.6046 -0.3235 0.024041 -0.2756 0.74761 -0.14654 0.0082566 -0.10071 0.53593 -0.17374 0.2
brazilian -0.54029 -0.63905 -0.094006 -0.68768 0.33263 -0.1583 -0.060424 0.20644 0.46234 -0.0764
夹竹桃 -0.4361 0.011429 -0.078896 -0.078186 0.37747 -0.052101 -0.096683 0.10769 0.62661 -0.37252
continent -0.37761 -0.72151 -0.42248 -0.81768 0.5016 -0.48569 0.13464 0.12644 0.32292 0.18099 0.
我还是 0.097443 0.28929 -0.14202 0.034027 0.50621 -0.1647 -0.45849 -0.16198 0.13965 -0.33451 0.61
vienna -0.25827 -0.050966 0.050502 -0.63466 0.4949 -0.17448 -0.59978 0.20269 0.37532 0.059419 0.
固态 -0.12678 0.4556 -0.27108 0.12506 0.52106 -0.058477 -0.69296 0.12162 0.26508 -0.089028 0.752
吉普 -0.33693 0.48335 -0.58455 0.13722 0.74856 -0.24529 -0.41125 -0.13832 0.33871 -0.12051 0.864
實物 0.030096 0.65756 -0.67982 0.2203 0.38492 -0.19001 -0.53136 -0.10322 0.24523 0.15287 0.92591
教职 0.11559 0.67087 -0.5111 0.14955 0.61417 -0.51571 -0.47901 0.29445 0.37629 -0.24232 0.4608 -
惕 0.50469 1.5357 -0.64393 0.48668 0.69479 -0.23443 -0.47863 0.16288 0.3347 -0.51673 0.86777 0.0
岸上 0.088323 0.85815 -0.485 0.30383 0.75965 -0.25031 -0.76678 0.12805 0.37641 -0.088752 0.65012
议和 0.26835 0.94854 -0.27972 0.097623 0.43305 -0.031361 -0.57406 0.21608 0.3324 -0.36823 0.6987
aka -0.21332 0.11216 -0.48872 -0.18531 0.79093 -0.34221 -0.51122 0.10067 0.29963 -0.075253 0.642
滑鐵盧 -0.28726 0.88014 -0.39751 -0.056992 0.37408 -0.16967 -0.20673 -0.048533 -0.1978 -0.13107 0
```

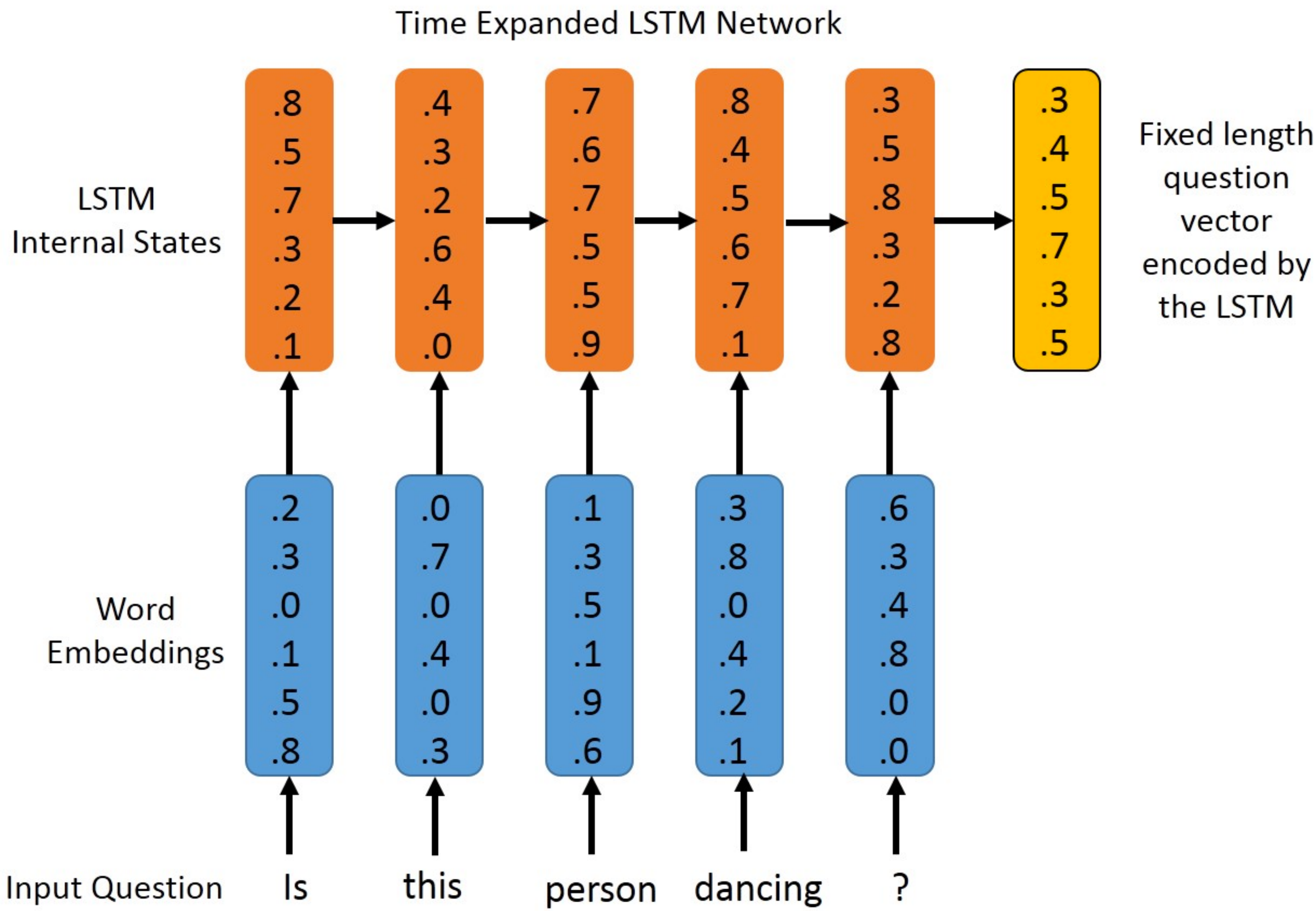
Models

The models can be downloaded from:

- Afrikaans: *bin+text*, *text*
- Albanian: *bin+text*, *text*
- Arabic: *bin+text*, *text*
- Armenian: *bin+text*, *text*
- Asturian: *bin+text*, *text*
- Azerbaijani: *bin+text*, *text*
- Bashkir: *bin+text*, *text*
- Basque: *bin+text*, *text*
- Belarusian: *bin+text*, *text*
- Bengali: *bin+text*, *text*
- Bosnian: *bin+text*, *text*
- Breton: *bin+text*, *text*
- Bulgarian: *bin+text*, *text*
- Burmese: *bin+text*, *text*
- Catalan: *bin+text*, *text*
- Cebuano: *bin+text*, *text*
- Chechen: *bin+text*, *text*
- Chinese: *bin+text*, *text*
- Chuvash: *bin+text*, *text*
- Croatian: *bin+text*, *text*
- Czech: *bin+text*, *text*

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Word Embeddings in LSTM RNN



Source: https://avisingh599.github.io/deeplearning/visual-qa/

Deep Learning

with

Keras

Deep Learning Software

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

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

Keras

Secure | https://keras.io

K Keras Documentation

Search docs

- Home
  - Keras: The Python Deep Learning library
  - You have just found Keras.
  - Guiding principles
  - Getting started: 30 seconds to Keras
  - Installation
  - Switching from TensorFlow to CNTK or Theano
  - Support
  - Why this name, Keras?
- Why use Keras
- Getting started
  - Guide to the Sequential model
  - Guide to the Functional API
  - FAQ
- Models
  - About Keras models
  - Sequential
  - Model (functional API)

GitHub Next »

Docs » Home Edit on GitHub

Keras: The Python Deep Learning library

Keras

You have just found Keras.

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

Use Keras if you need a deep learning library that:

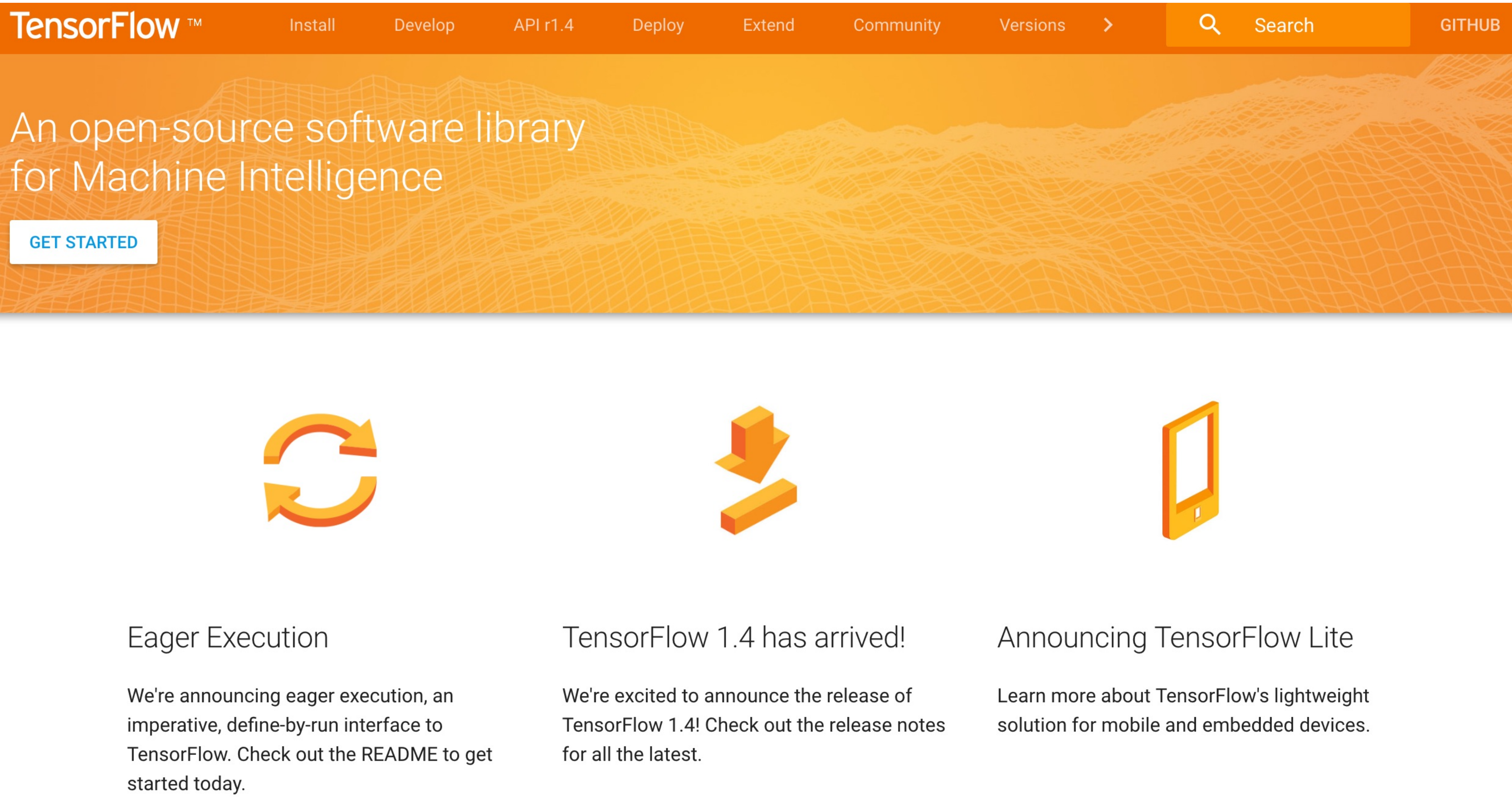
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

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

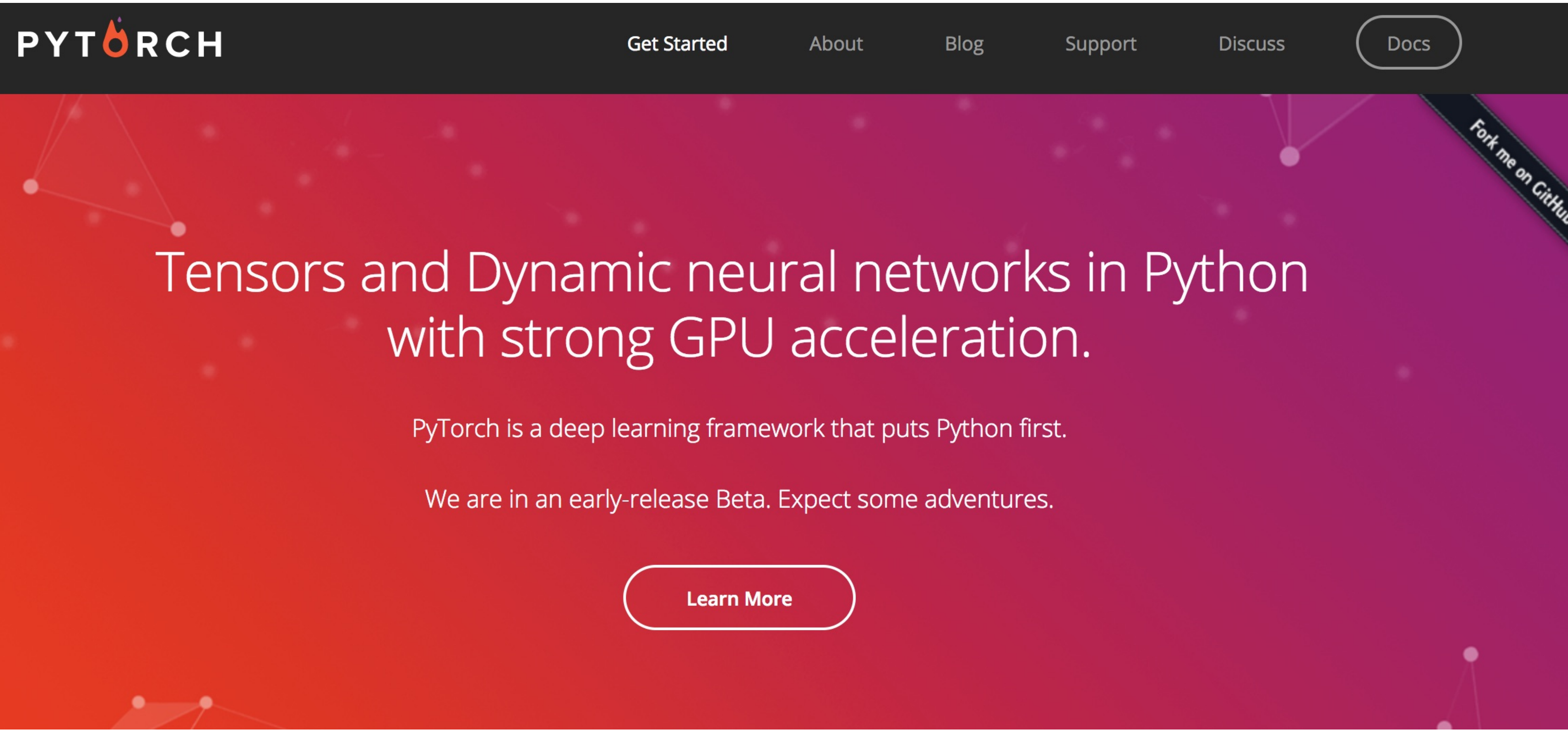
http://keras.io/

Tensorflow



https://www.tensorflow.org/

PyTorch



http://pytorch.org/

Keras

Keras

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

Source: https://keras.io/

Install Keras

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

https://keras.io/#installation

TensorFlow Installation

Secure https://www.tensorflow.org/install/

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

Install

Installing TensorFlow
Installing TensorFlow on Ubuntu
Installing TensorFlow on macOS
Installing TensorFlow on Windows
Installing TensorFlow from Sources

Transitioning to TensorFlow 1.0

Installing TensorFlow for Java
Installing TensorFlow for Go
Installing TensorFlow for C

Installing TensorFlow

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

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

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

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

- Installing TensorFlow on Ubuntu
- Installing TensorFlow on macOS
- Installing TensorFlow on Windows
- Installing TensorFlow from Sources

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

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

Keras Installation

Keras Documentation

Search docs

Home

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Switching from TensorFlow to Theano

Support

Why this name, Keras?

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

Layers

Installation

Keras uses the following dependencies:

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

When using the TensorFlow backend:

- TensorFlow
  - See installation instructions.

When using the Theano backend:

- Theano
  - See installation instructions.

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

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

You can also install Keras from PyPI:

```
sudo pip install keras
```

https://keras.io/#installation

```
conda info --envs
```

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

```
conda list
```

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

```
source activate tensorflow
```

```
activate tensorflow
```

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

```
sudo pip install tensorflow
```

```
pip install tensorflow
```

```
sudo pip install keras
```

```
pip install keras
```

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

```
jupyter notebook
```

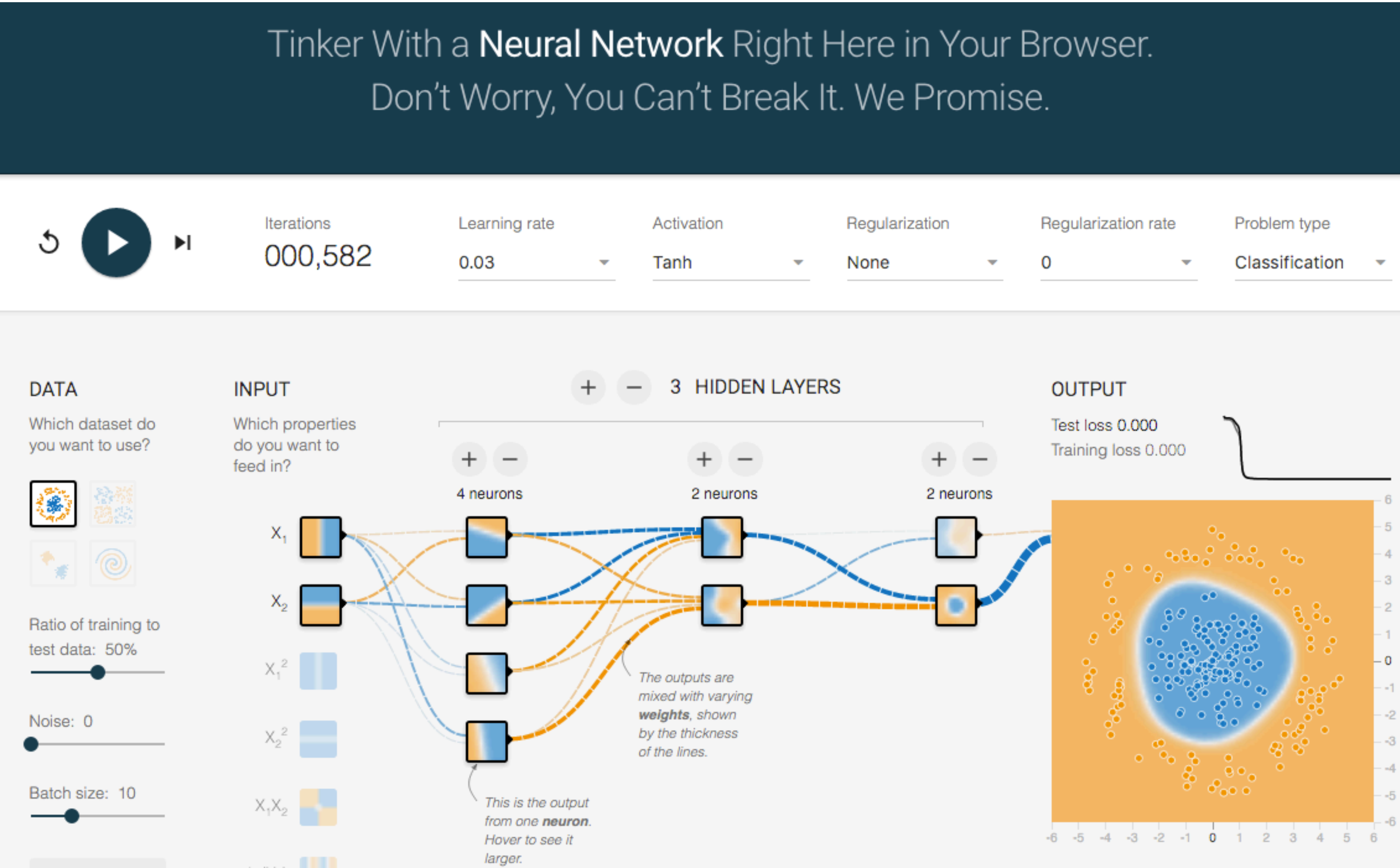
Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt

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

pip install tensorflow

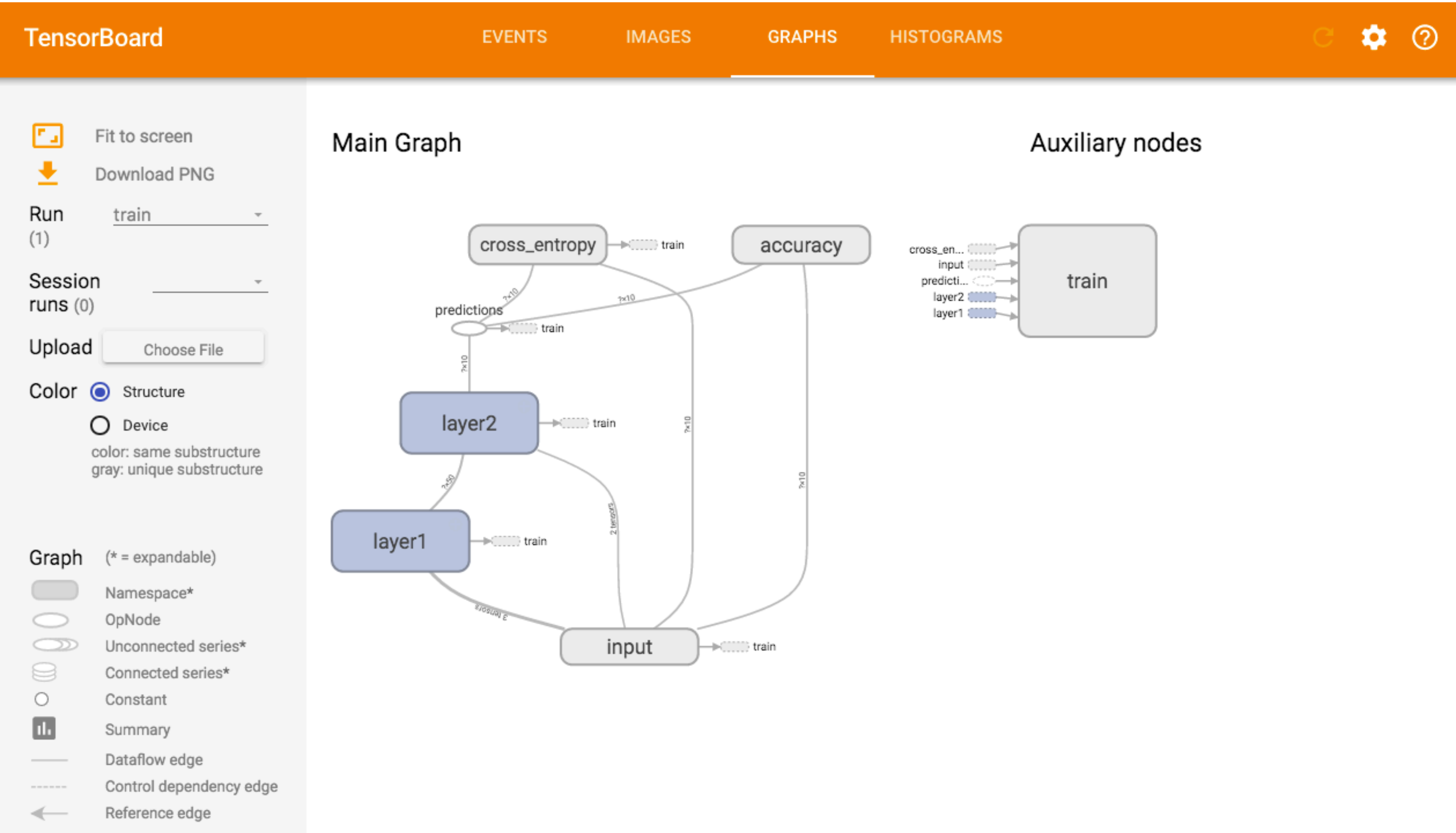
```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement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

TensorFlow Playground



http://playground.tensorflow.org/

TensorBoard



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

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

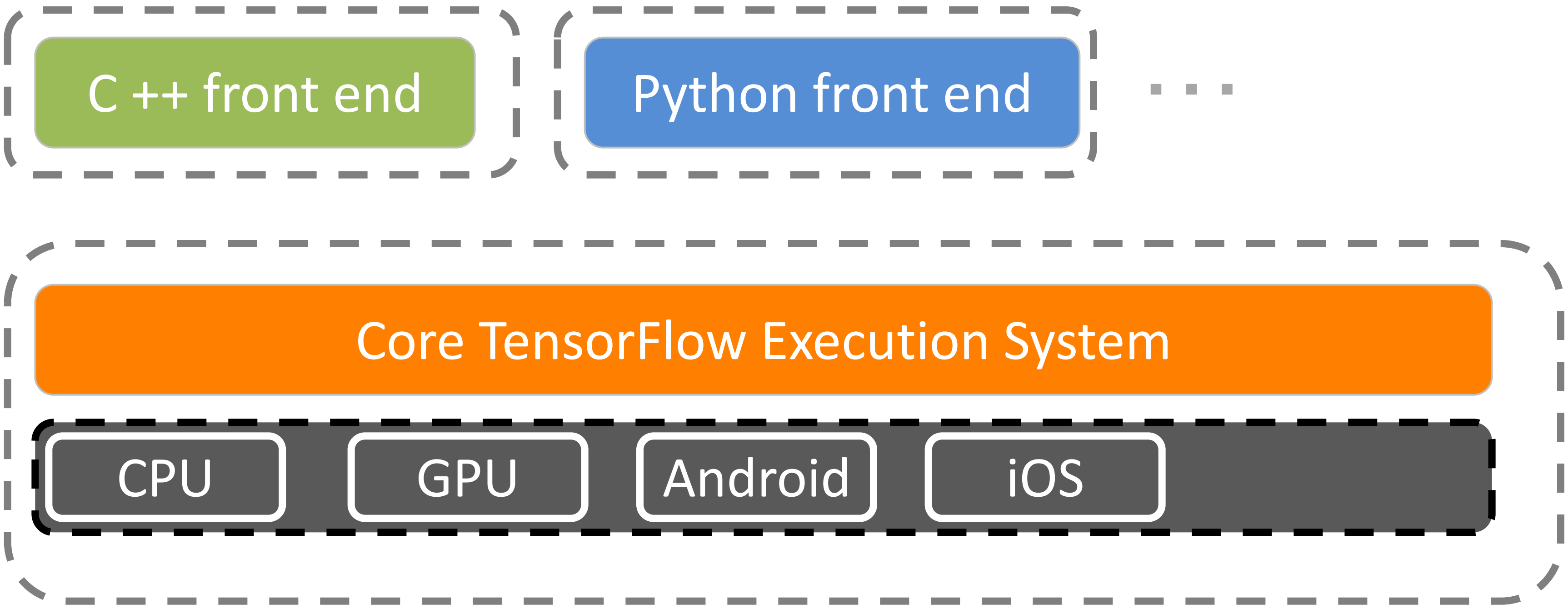
Try your first TensorFlow

```
$ 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

Keras

Secure | https://keras.io

Keras Documentation

Search docs

- Home
  - Keras: The Python Deep Learning library
  - You have just found Keras.
  - Guiding principles
  - Getting started: 30 seconds to Keras
  - Installation
  - Switching from TensorFlow to CNTK or Theano
  - Support
  - Why this name, Keras?
- Why use Keras
- Getting started
  - Guide to the Sequential model
  - Guide to the Functional API
  - FAQ
- Models
  - About Keras models
  - Sequential
  - Model (functional API)

GitHub Next »

Docs » Home

Edit on GitHub

Keras: The Python Deep Learning library

You have just found Keras.

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

Use Keras if you need a deep learning library that:

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

Read the documentation at Keras.io.

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

http://keras.io/

Keras Installation

K 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

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_x86_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 (from 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_x86_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_x86_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/02/44/43/68e963ce2b45baefa913a4e558bcd787403458afddffcf45ca
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 and will be removed in a future version. This is due to the fact that uninstalling a distutils 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$
```

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

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$
```

```
source activate tensorflow
```

```
jupyter notebook
```

```
import tensorflow as tf
print(tf.__version__)
```

```
import keras
print(keras.__version__)
```

```
import tensorflow as tf
print(tf.__version__)
```

```
1.4.0
```

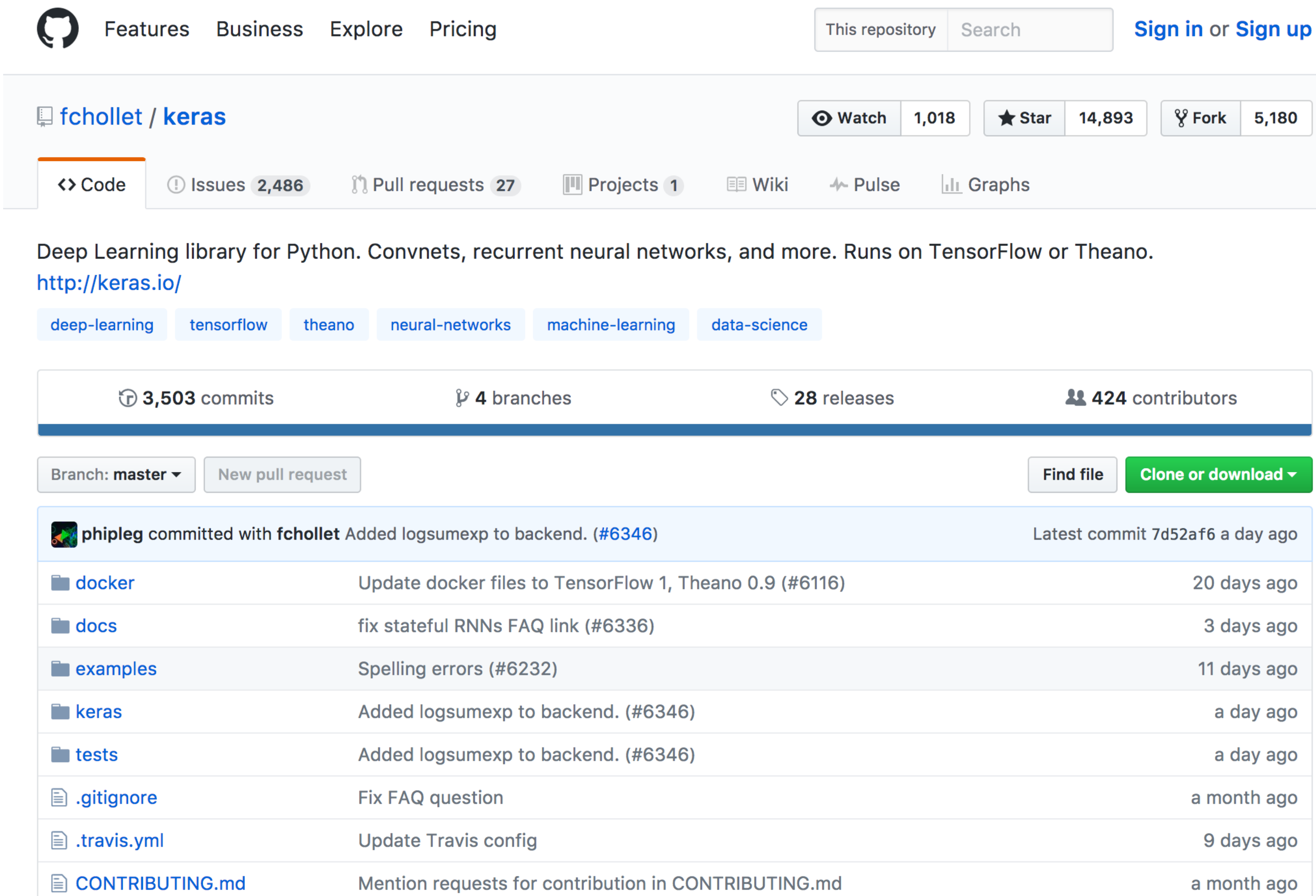
```
import keras
```

```
Using TensorFlow backend.
```

```
print(keras.__version__)
```

```
2.1.2
```

Keras Github



https://github.com/fchollet/keras

Keras Examples

Features Business Explore Pricing | This repository | Search | **Sign in** or **Sign up**

fchollet / **keras** | Watch 1,018 | Star 14,893 | Fork 5,181

<> Code | Issues 2,486 | Pull requests 27 | Projects 1 | Wiki | Pulse | Graphs

Branch: master | keras / **examples** / | Create new file | Find file | History

Mohanson committed with **fchollet** Spelling errors (#6232) — Latest commit 5bd3976 11 days ago

..

README.md	Adding mnist_acgan.py example link in README (#4876)	4 months ago
addition_rnn.py	Spelling errors (#6232)	11 days ago
antirectifier.py	Style fix for examples. (#5980)	28 days ago
babi_memnn.py	Style fixes in example scripts	a month ago
babi_rnn.py	Style fixes in example scripts	a month ago
cifar10_cnn.py	fix rmsprop learning rate for convergence (#6182)	17 days ago
conv_filter_visualization.py	Finish updating examples.	a month ago
conv_lstm.py	Update a number of example scripts.	2 months ago
deep_dream.py	Finish updating examples.	a month ago
image_ocr.py	Fixed URL for wordlist.tgz in image_ocr.py (#6136)	20 days ago
imdb_bidirectional_lstm.py	Finish updating examples.	a month ago
imdb_cnn.py	Finish updating examples.	a month ago
imdb_cnn_lstm.py	Style fix for examples. (#5980)	28 days ago

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

Keras Examples

- imdb_bidirectional_lstm.py Trains a Bidirectional LSTM on the IMDB sentiment classification task.
- imdb_cnn.py Demonstrates the use of Convolution1D for text classification.
- imdb_cnn_lstm.py Trains a convolutional stack followed by a recurrent stack network on the IMDB sentiment classification task.
- imdb_fasttext.py Trains a FastText model on the IMDB sentiment classification task.
- imdb_lstm.py Trains a LSTM on the IMDB sentiment classification task.
- lstm_benchmark.py Compares different LSTM implementations on the IMDB sentiment classification task.
- lstm_text_generation.py Generates text from Nietzsche's writings.

Source: 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

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
```

IMDB
Large Movie Review Dataset

- This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets.
- We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing.
- There is additional unlabeled data for use as well.
- Raw text and already processed bag of words formats are provided.
- Large Movie Review Dataset v1.0
  - http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

Source: http://ai.stanford.edu/~amaas/data/sentiment/

IMDB Dataset (Mass et al., 2011)

Features	PL04	Our Dataset	Subjectivity
Bag of Words (bnc)	85.45	87.80	87.77
Bag of Words (bΔt'c)	85.80	88.23	85.65
LDA	66.70	67.42	66.65
LSA	84.55	83.96	82.82
Our Semantic Only	87.10	87.30	86.65
Our Full	84.65	87.44	86.19
Our Full, Additional Unlabeled	87.05	87.99	87.22
Our Semantic + Bag of Words (bnc)	88.30	88.28	88.58
Our Full + Bag of Words (bnc)	87.85	88.33	88.45
Our Full, Add'l Unlabeled + Bag of Words (bnc)	88.90	88.89	88.13
Bag of Words SVM (Pang and Lee, 2004)	87.15	N/A	90.00
Contextual Valence Shifters (Kennedy and Inkpen, 2006)	86.20	N/A	N/A
tf.Δidf Weighting (Martineau and Finin, 2009)	88.10	N/A	N/A
Appraisal Taxonomy (Whitelaw et al., 2005)	90.20	N/A	N/A

Table 2: Classification accuracy on three tasks. From left to right the datasets are: A collection of 2,000 movie reviews often used as a benchmark of sentiment classification (Pang and Lee, 2004), 50,000 reviews we gathered from IMDB, and the sentence subjectivity dataset also released by (Pang and Lee, 2004). All tasks are balanced two-class problems.

Source: Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*

Keras IMDB Movie reviews sentiment classification

- Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative).
- Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers).
- For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data.
- This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".
- As a convention, "0" does not stand for a specific word, but instead is used to encode any unknown word.

Source: https://keras.io/datasets/

Keras IMDB load_data

```
def load_data(path='imdb.npz',
              num_words=None,
              skip_top=0,
              maxlen=None,
              seed=113,
              start_char=1,
              oov_char=2,
              index_from=3):
  path = get_file(
      path, origin='https://s3.amazonaws.com/text-datasets/imdb.npz')
  f = np.load(path)
  x_train = f['x_train']
  labels_train = f['y_train']
  x_test = f['x_test']
  labels_test = f['y_test']
  f.close()
```

Source: keras/datasets/imdb.py

Keras IMDB get_word_index

```
def get_word_index(path='imdb_word_index.json'):
  path = get_file(
      path,
      origin='https://s3.amazonaws.com/text-datasets/imdb_word_index.json')
  f = open(path)
  data = json.load(f)
  f.close()
  return data
```

Source: keras/datasets/imdb.py

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

```python
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

```python
# 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
```

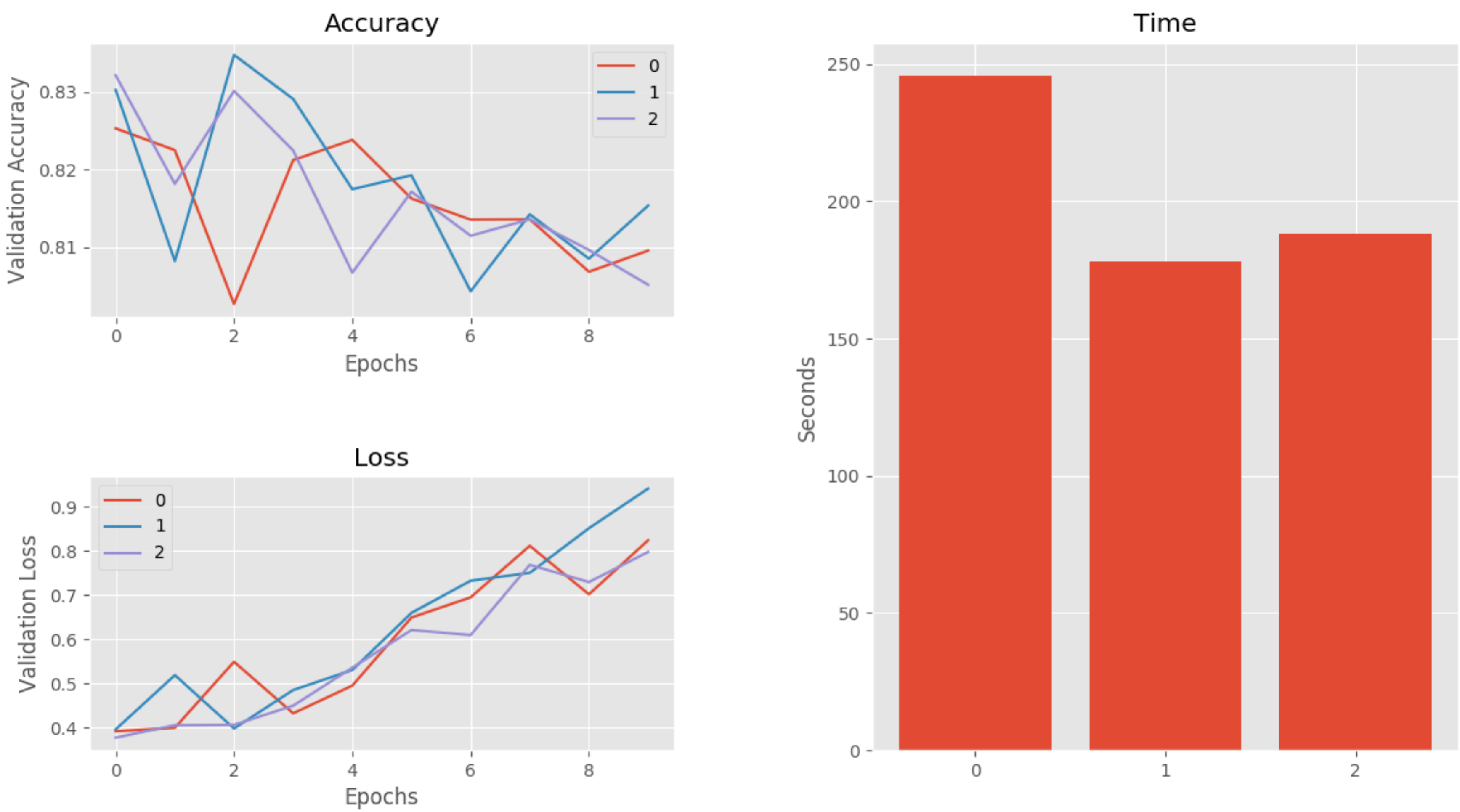
Keras IMDB FastText

```
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
```

Keras LSTM Benchmark



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

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

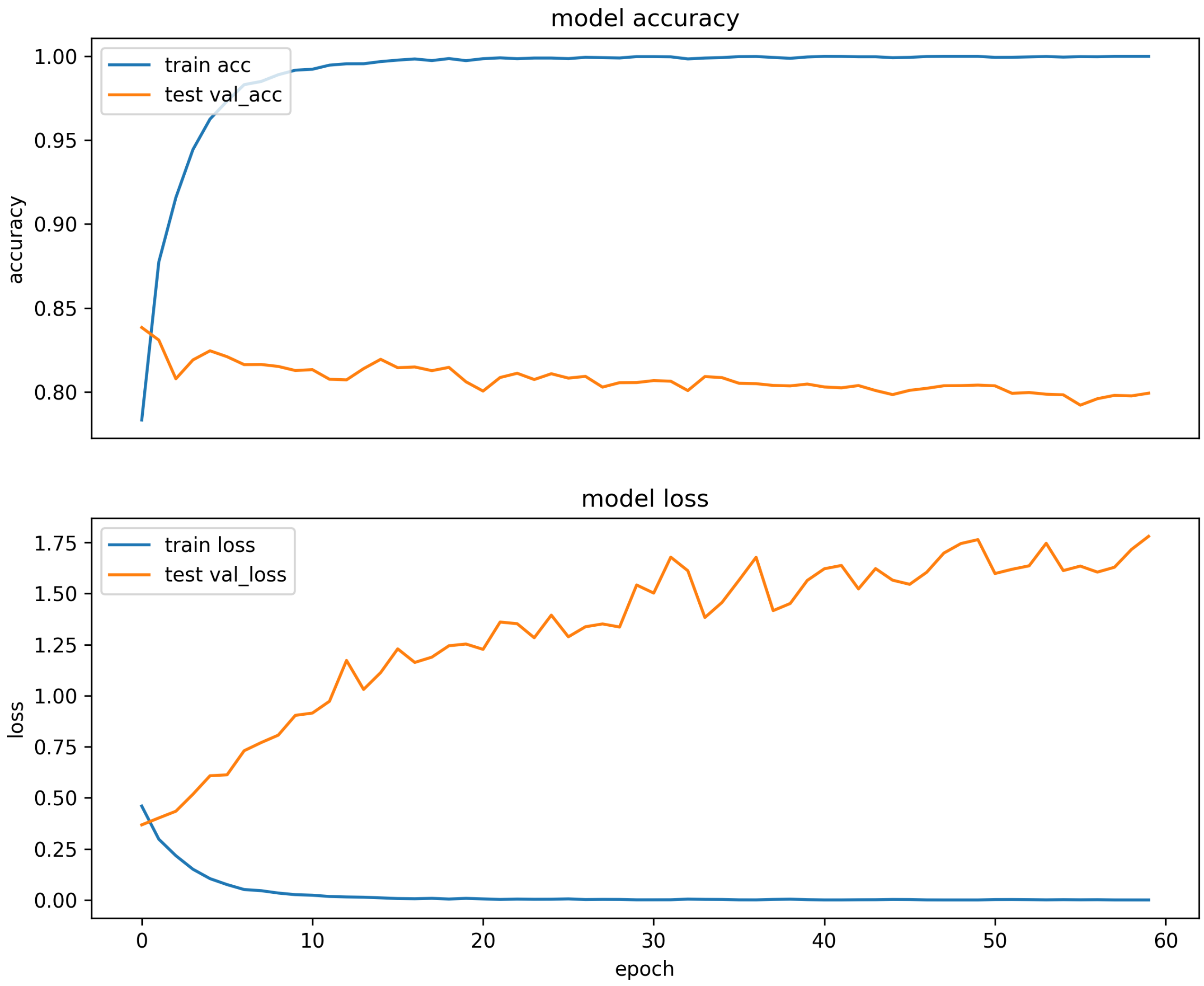
```
python imdb_fasttext_2.py
python imdb_cnn_2.py
python imdb_lstm_2.py
python imdb_cnn_lstm_2.py
python imdb_bidirectional_lstm_2.py
```

Deep Learning Summary

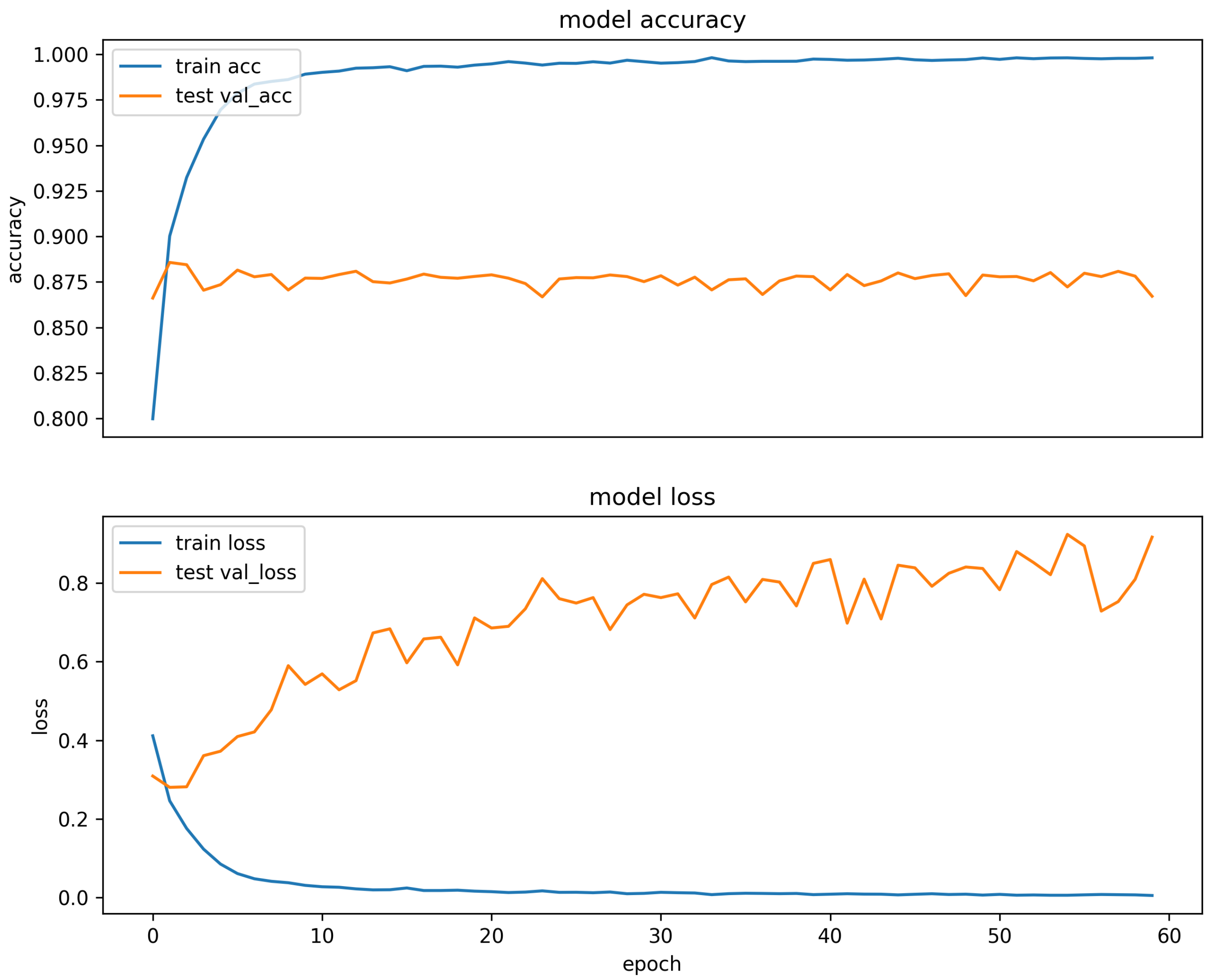
```
#Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\t'  + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t ' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))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

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

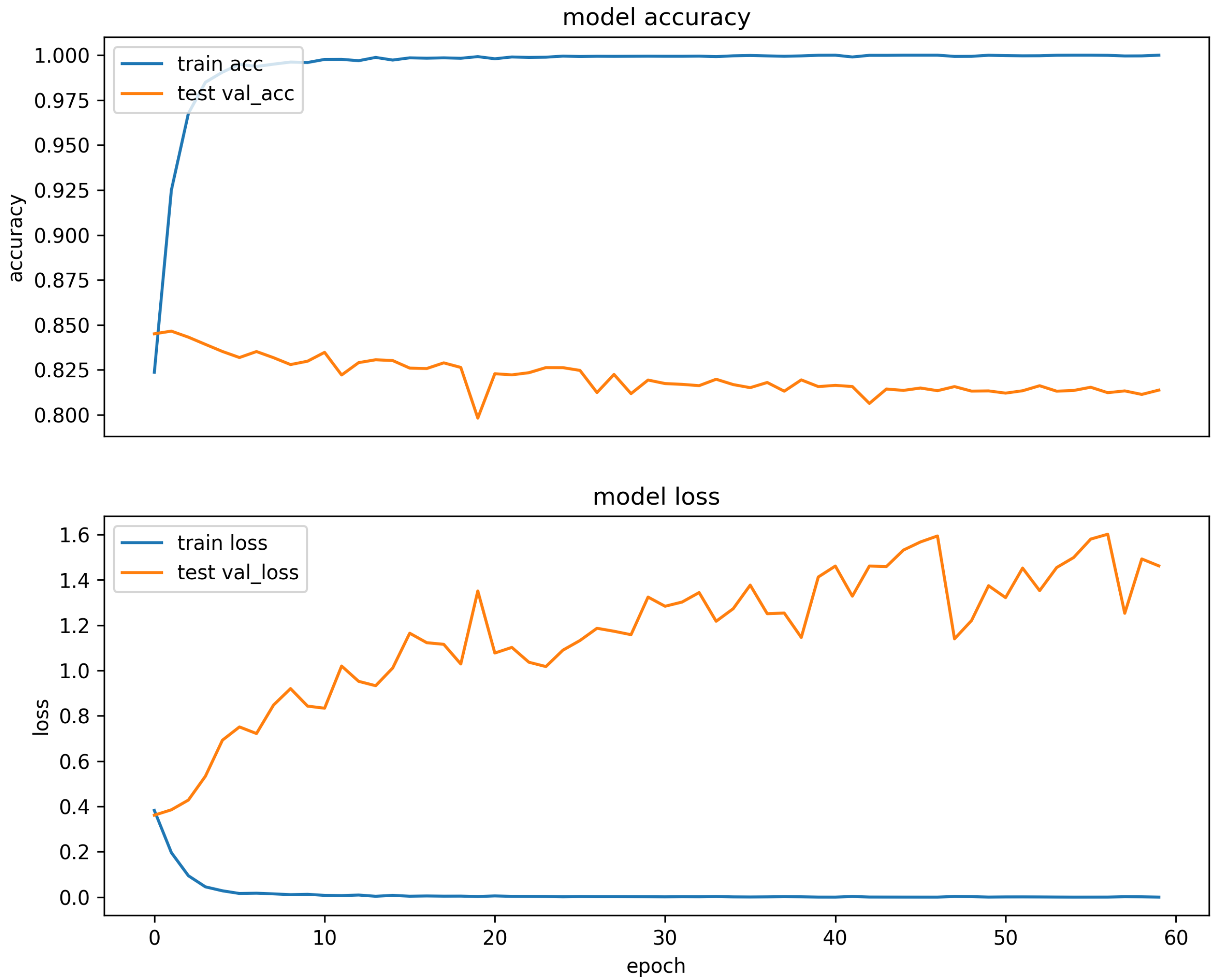


imdb_cnn_2.py

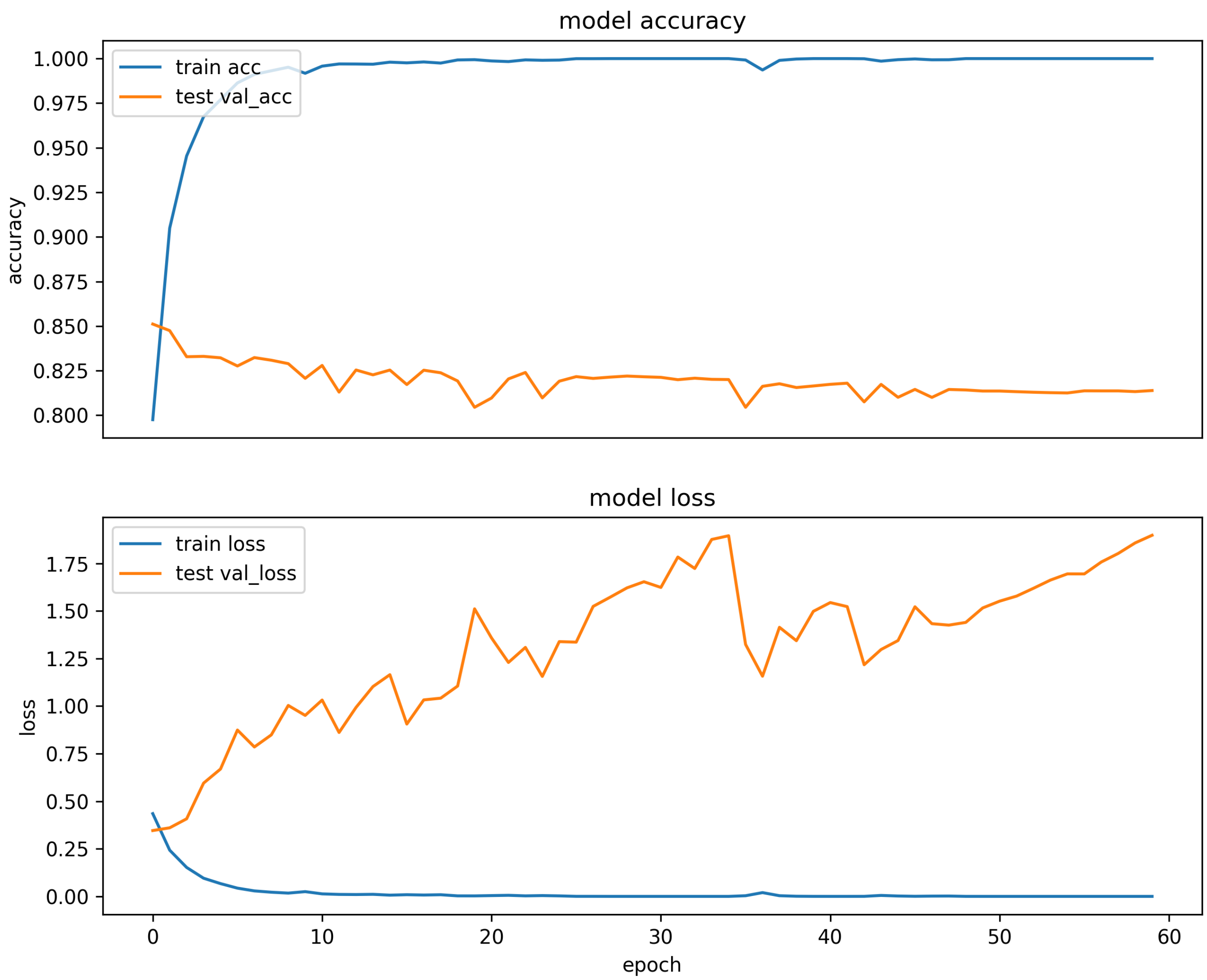


model accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
accuracy
train acc
test val_acc
model loss
0.8
0.6
0.4
0.2
0.0
loss
train loss
test val_loss
0
10
20
30
40
50
60
epoch

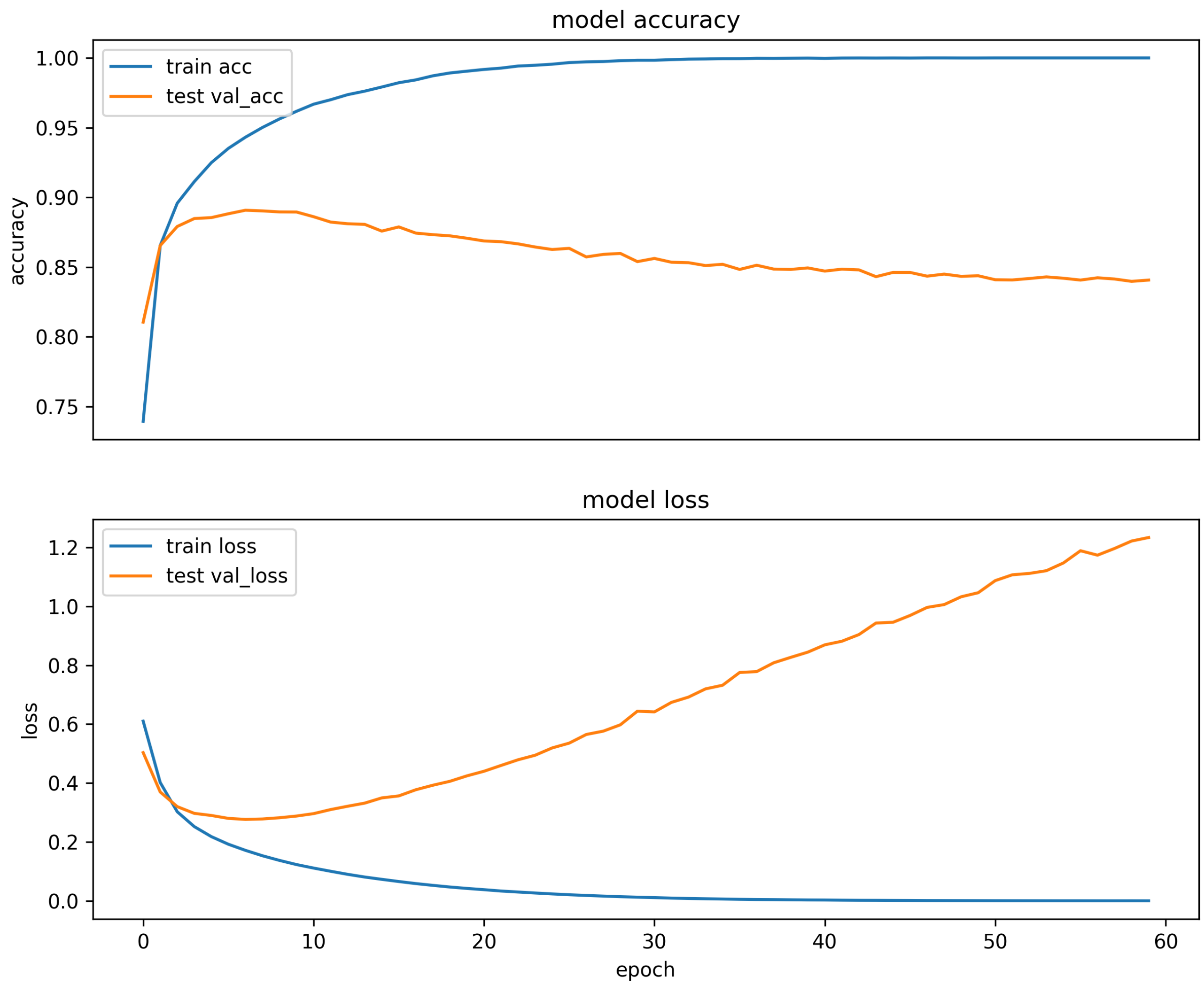
imdb_cnn_lstm_2.py



imdb_bidirectional_lstm_2.py



imdb_fasttext_2.py



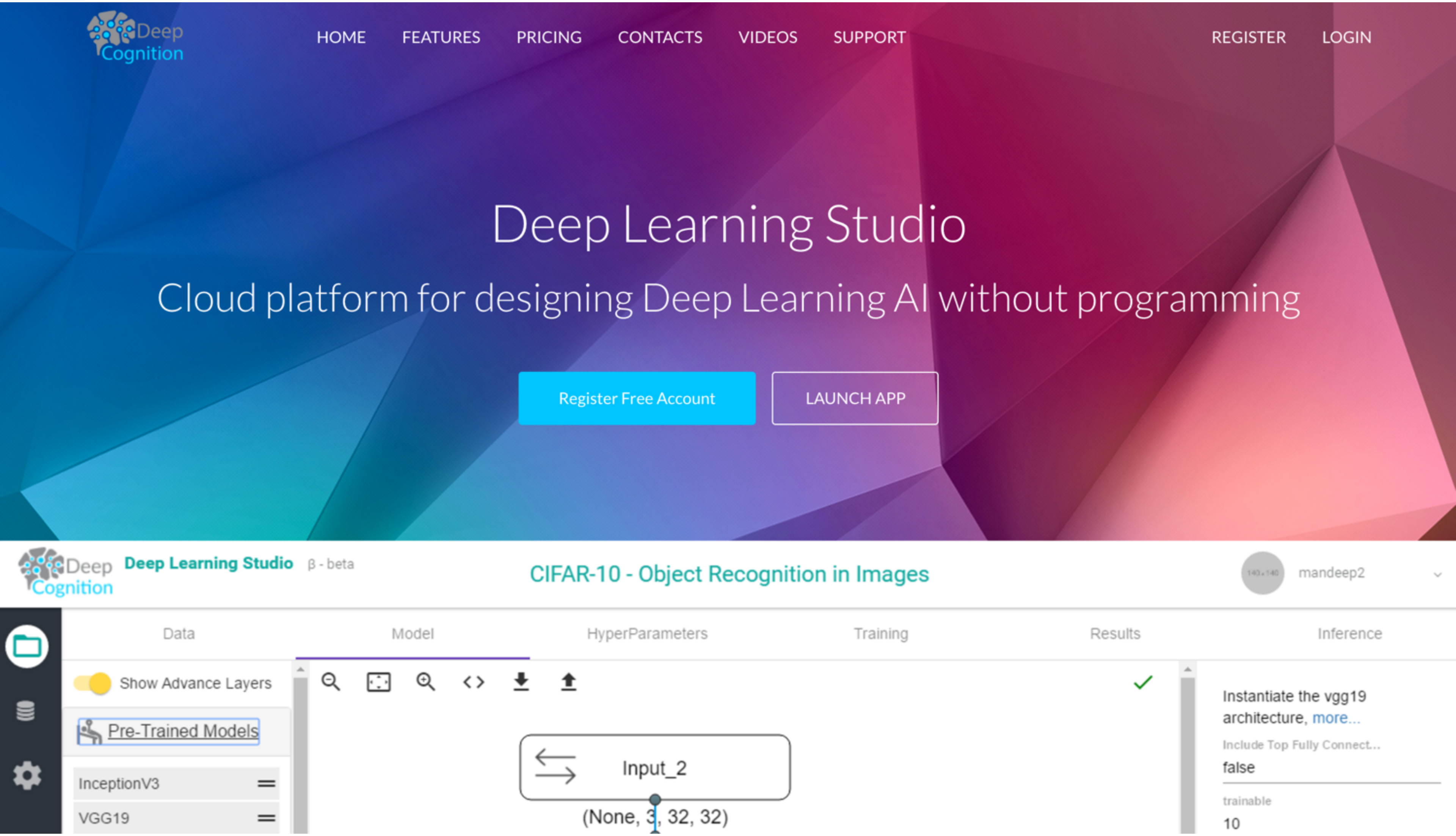
Deep Learning with CPU vs. GPU

```
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

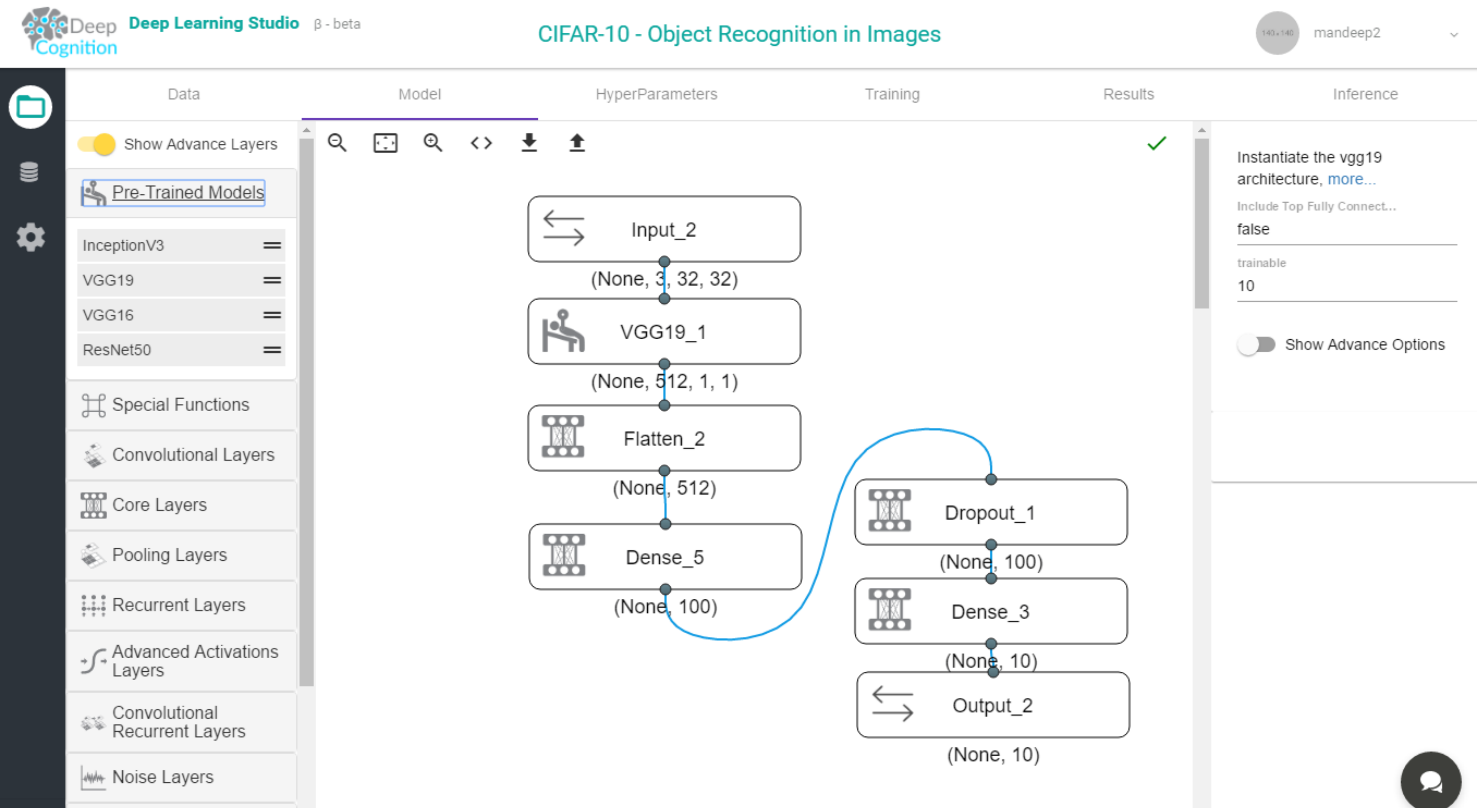
Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

Deep Learning Studio

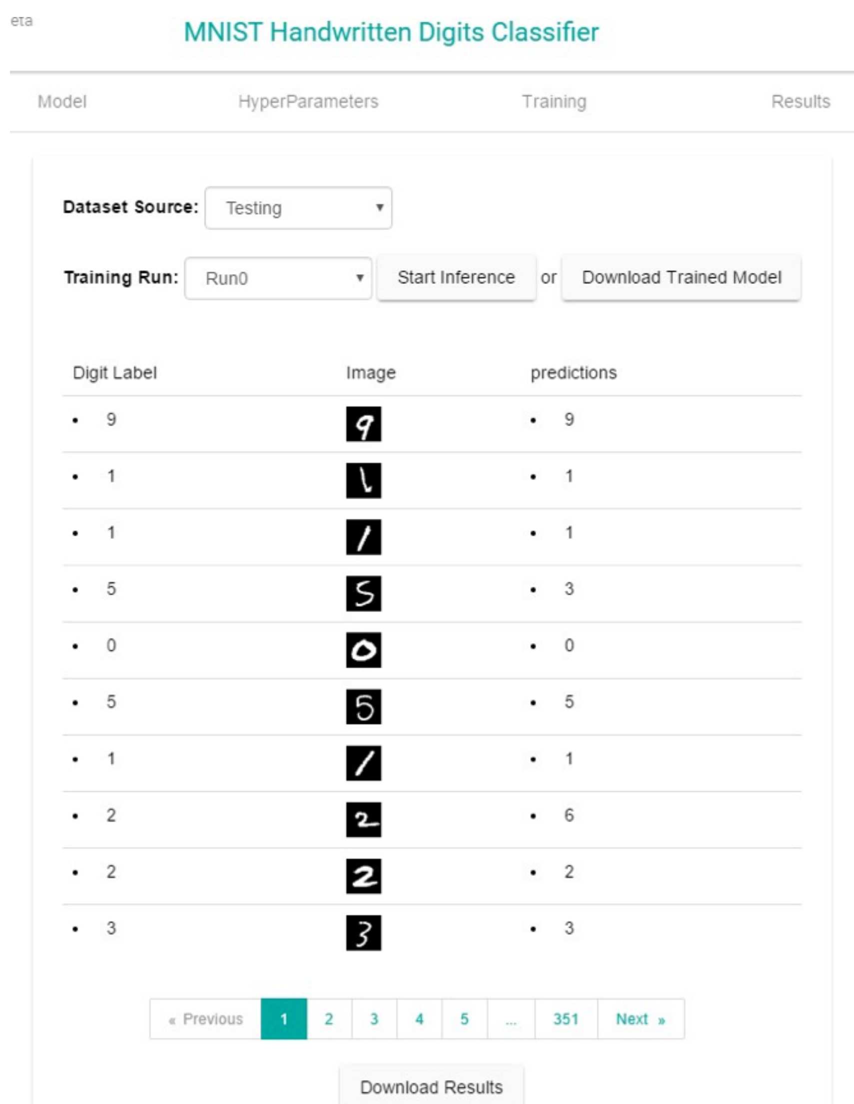
Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming



http://deepcognition.ai/

Summary

- AI, Machine Learning and Deep Learning
- Deep Learning Foundations: Neural Networks
- Keras: High-level API for TensorFlow

References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), https://www.youtube.com/watch?v=u4alGiomYP4
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFfI8
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, https://goo.gl/pHeXe7, https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Deep Learning Studio: Cloud platform for designing Deep Learning AI without programming, http://deepcognition.ai/
- Natural Language Processing with Deep Learning (Winter 2017), https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow