人工智慧財務金融應用

AI in Financial Application

Tamkang University
淡江大學

TensorFlow 深度學習財務金融應用

(Deep Learning for Finance Application with TensorFlow)

1081AIFA08
EMBA, IMTKU (M2457) (8413) (Fall 2019)
Fri 12,13,14 (19:20-22:10) (D301)

[Min-Yuh Day](http://mail.tku.edu.tw/myday/)
戴敏育
Associate Professor
副教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

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

2019-11-29, 12-13, 12-20

課程大綱 (Syllabus)

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

1 2019/09/13 中秋節 (Mid-Autumn Festival) 放假一天 (Day off)

2 2019/09/20 人工智慧財務金融應用課程介紹
(Course Orientation for AI in Financial Application)

3 2019/09/27 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 2019/10/04 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 2019/10/11 國慶日補假 (Bridge Holiday for National Day, Extra Day Off)

6 2019/10/18 財務金融事件研究法 (Event Studies in Finance)

課程大綱 (Syllabus)

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

7　2019/10/25　人工智慧財務金融應用個案研究 I
(Case Study on AI in Financial Application I)

8　2019/11/01　Python AI智慧金融分析基礎
(Foundations of AI in Finance Big Data Analytics with Python)

9　2019/11/08　Python Pandas 量化投資分析
(Quantitative Investing with Pandas in Python)

10　2019/11/15　期中報告 (Midterm Project Report)

11　2019/11/22　Python Scikit-Learn 機器學習財務金融應用
(Machine Learning in Finance Application with Scikit-Learn In Python)

12　2019/11/29　TensorFlow 深度學習財務金融應用 I
(Deep Learning for Finance Application with TensorFlow I)

課程大綱 (Syllabus)

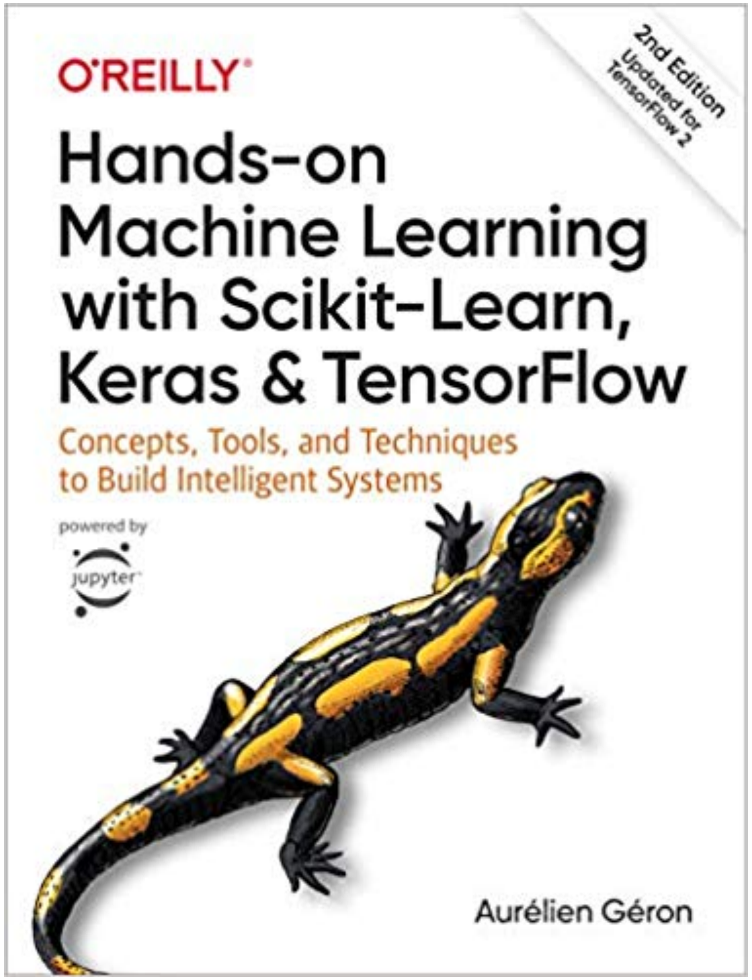
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2019/12/06	人工智慧財務金融應用個案研究 II (Case Study on AI in Financial Application II)
14	2019/12/13	TensorFlow 深度學習財務金融應用 II (Deep Learning for Finance Application with TensorFlow II)
15	2019/12/20	TensorFlow 深度學習財務金融應用 III (Deep Learning for Finance Application with TensorFlow III)
16	2019/12/27	社會網絡分析財務金融應用 (Social Network Analysis for Finance Application)
17	2020/01/03	期末報告 I (Final Project Presentation I)
18	2020/01/10	期末報告 II (Final Project Presentation II)

Deep Learning
for
Financial Application
with TensorFlow

Outline

- Deep Learning for Financial Application with TensorFlow
  - Deep Learning
  - Financial Application
  - TensorFlow

Aurélien Géron (2019),
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition
O'Reilly Media, 2019



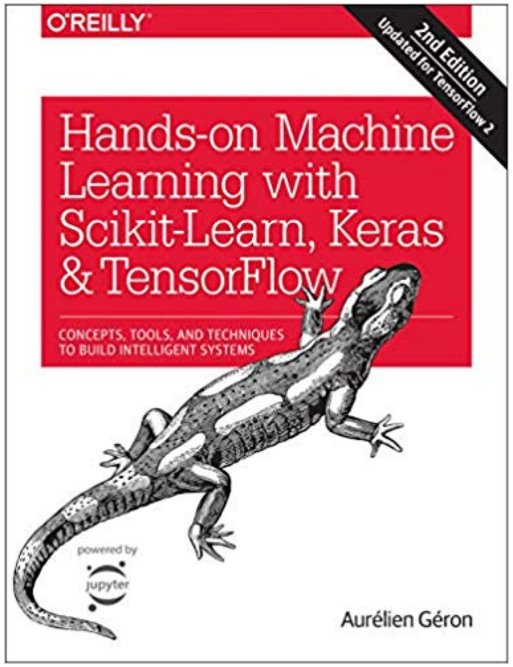
https://github.com/ageron/handson-ml2

Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1492032646/

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml2

Sequences using RNNs and CNNs

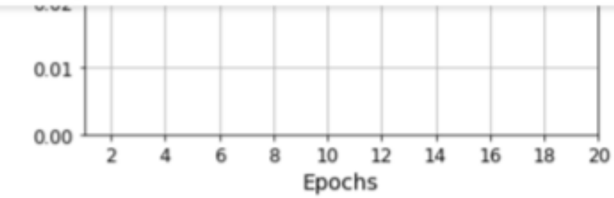
15_processing_sequences_using_rnns_and_cnns.ipynb

File Edit View Insert Runtime Tools Help Last edited on November 6 by ageron

Share

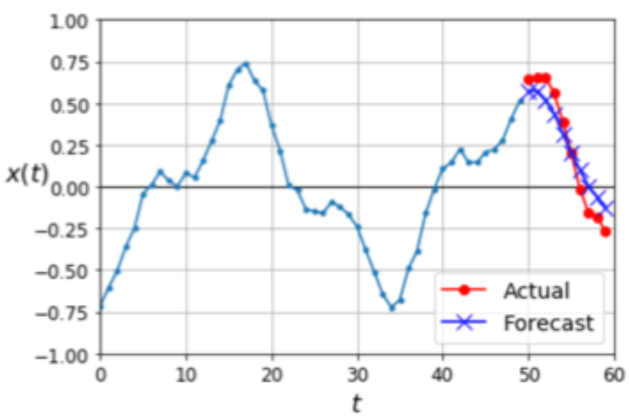
+ Code + Text Copy to Drive

Connect Editing




```
np.random.seed(43)

series = generate_time_series(1, 50 + 10)
X_new, Y_new = series[:, :50, :], series[:, 50:, :]
Y_pred = model.predict(X_new)[:, -1][..., np.newaxis]
```

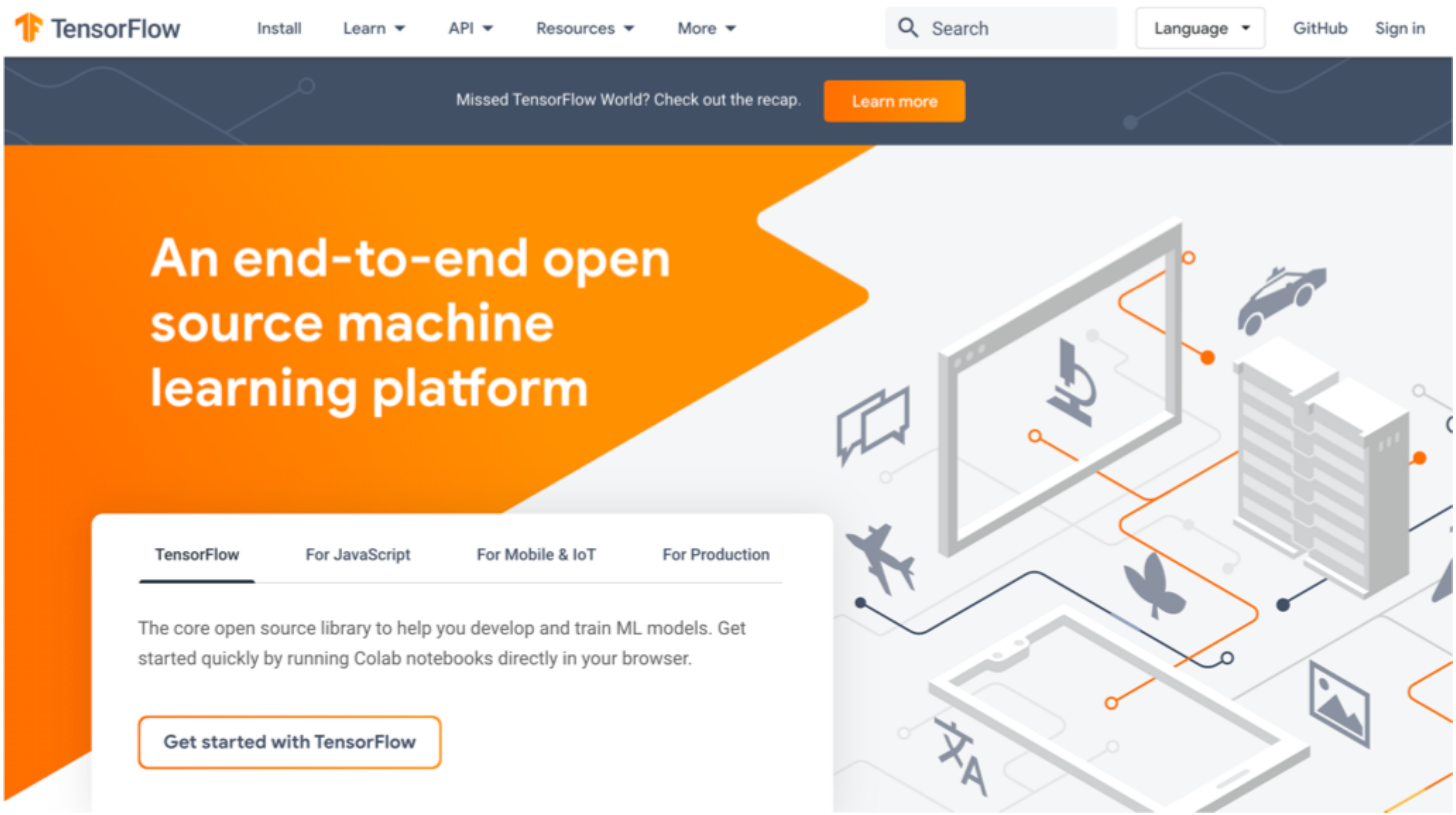
```
plot_multiple_forecasts(X_new, Y_new, Y_pred)
plt.show()
```



https://colab.research.google.com/github/ageron/handson-

TensorFlow

TensorFlow



https://www.tensorflow.org/

TensorFlow

- An end-to-end open source machine learning platform.
- The core open source library to help you develop and train ML models.
- Get started quickly by running Colab notebooks directly in your browser.

https://www.tensorflow.org/

TensorFlow

Why TensorFlow 2.0

Why TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

About →

Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

Robust ML production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

TensorFlow 2.0 vs. 1.X

```
# TensorFlow 2.0
outputs = f(input)

# TensorFlow 1.X
outputs = session.run(f(placeholder), feed_dict={placeholder: input})
```

Source: https://www.tensorflow.org/guide/effective_tf2

TensorFlow 2.0

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/overview/

TensorFlow

TensorFlow 2 Quick Start

beginner.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 2 by lamberta

Share A

+ Code + Text Copy to Drive Connect Editing

TensorFlow 2 quickstart for beginners

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

This short introduction uses Keras to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

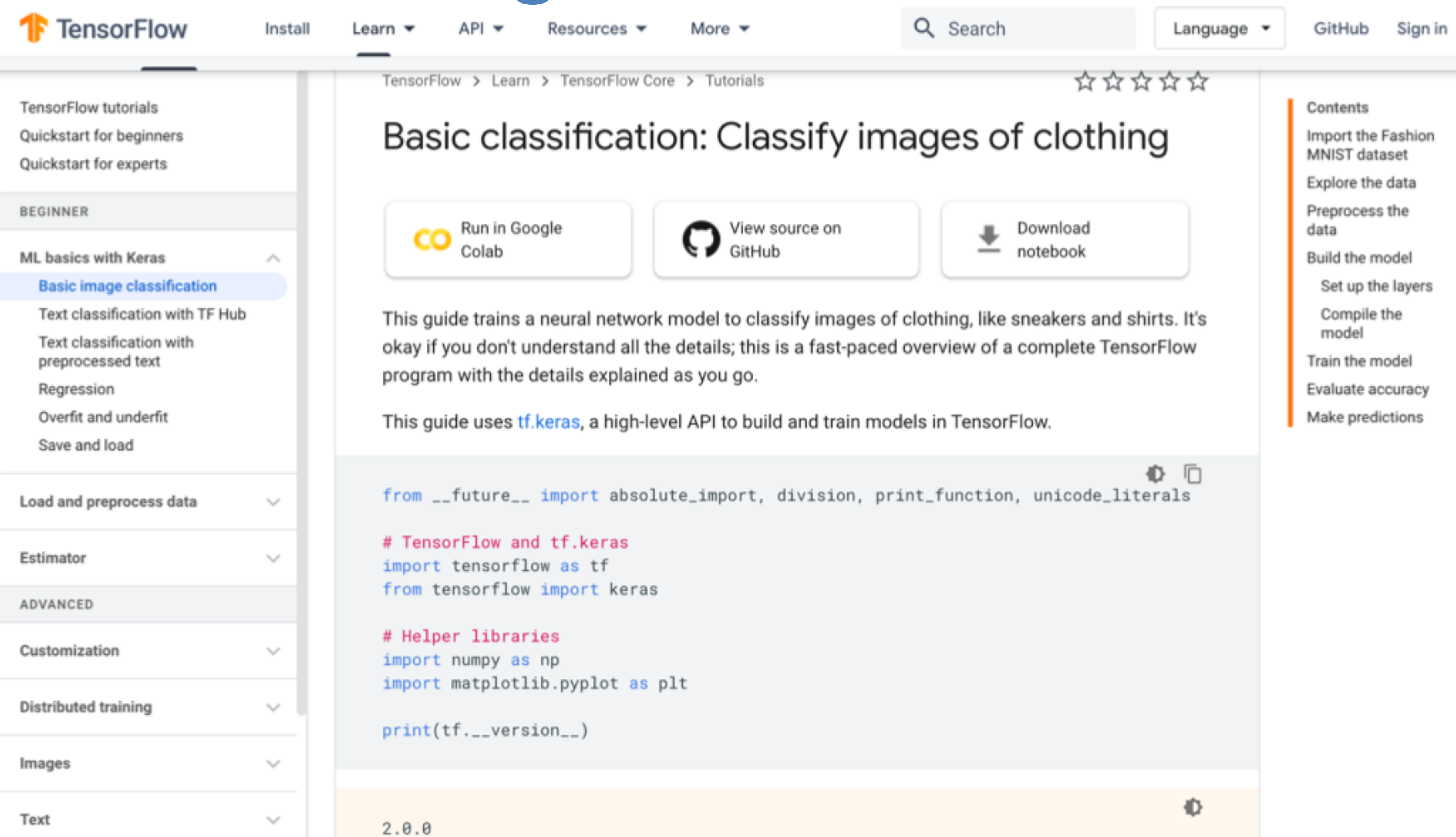
Download and install the TensorFlow 2 package. Import TensorFlow into your program:

```
from __future__ import absolute_import, division, print_function, unicode_literals

# Install TensorFlow
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb

TensorFlow

TensorFlow
Image Classification



TensorFlow · Install · Learn · API · Resources · More · Search · Language · GitHub · Sign in

TensorFlow > Learn > TensorFlow Core > Tutorials

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
Estimator

ADVANCED

Customization
Distributed training
Images
Text

Basic classification: Classify images of clothing

Run in Google Colab · View source on GitHub · Download notebook

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

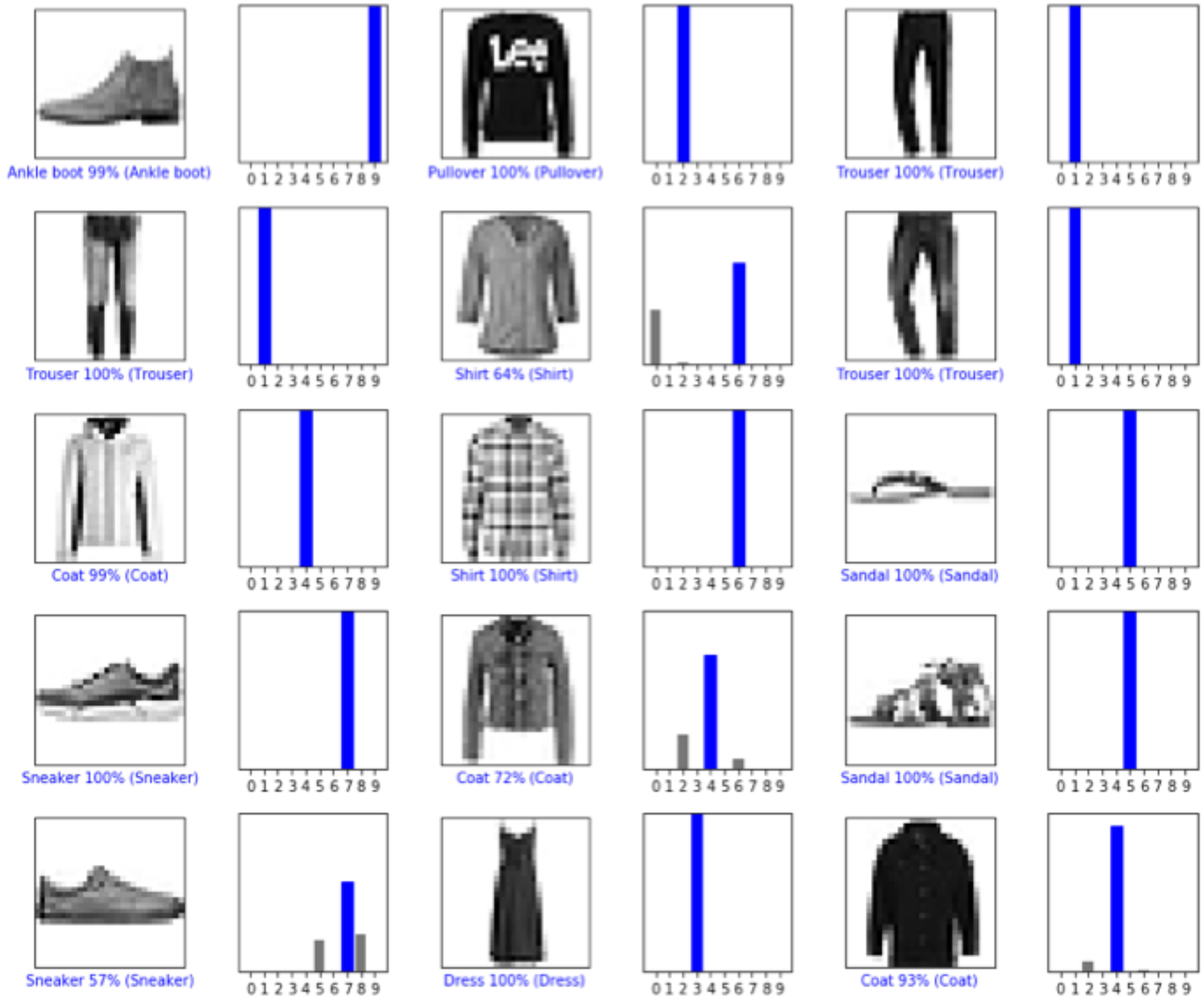
This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.0.0
```


https://www.tensorflow.org/tutorials/keras/classification

Image Classification
Fashion MNIST dataset



https://www.tensorflow.org/tutorials/keras/classification

TensorFlow

Text Classification with TF Hub

TensorFlow | Install | Learn | API | Resources | More | Search | Language | GitHub | Sign in

Overview | Tutorials | Guide | TF 1

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
- CSV
- NumPy
- pandas.DataFrame
- Images
- Text
- Unicode
- TF.Text
- TFRecord and tf.Example
- Additional formats with tf.io

TensorFlow > Learn > TensorFlow Core > Tutorials

Text classification with TensorFlow Hub: Movie reviews

Run in Google Colab | View source on GitHub | Download notebook

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

The tutorial demonstrates the basic application of transfer learning with TensorFlow Hub and Keras.

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

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

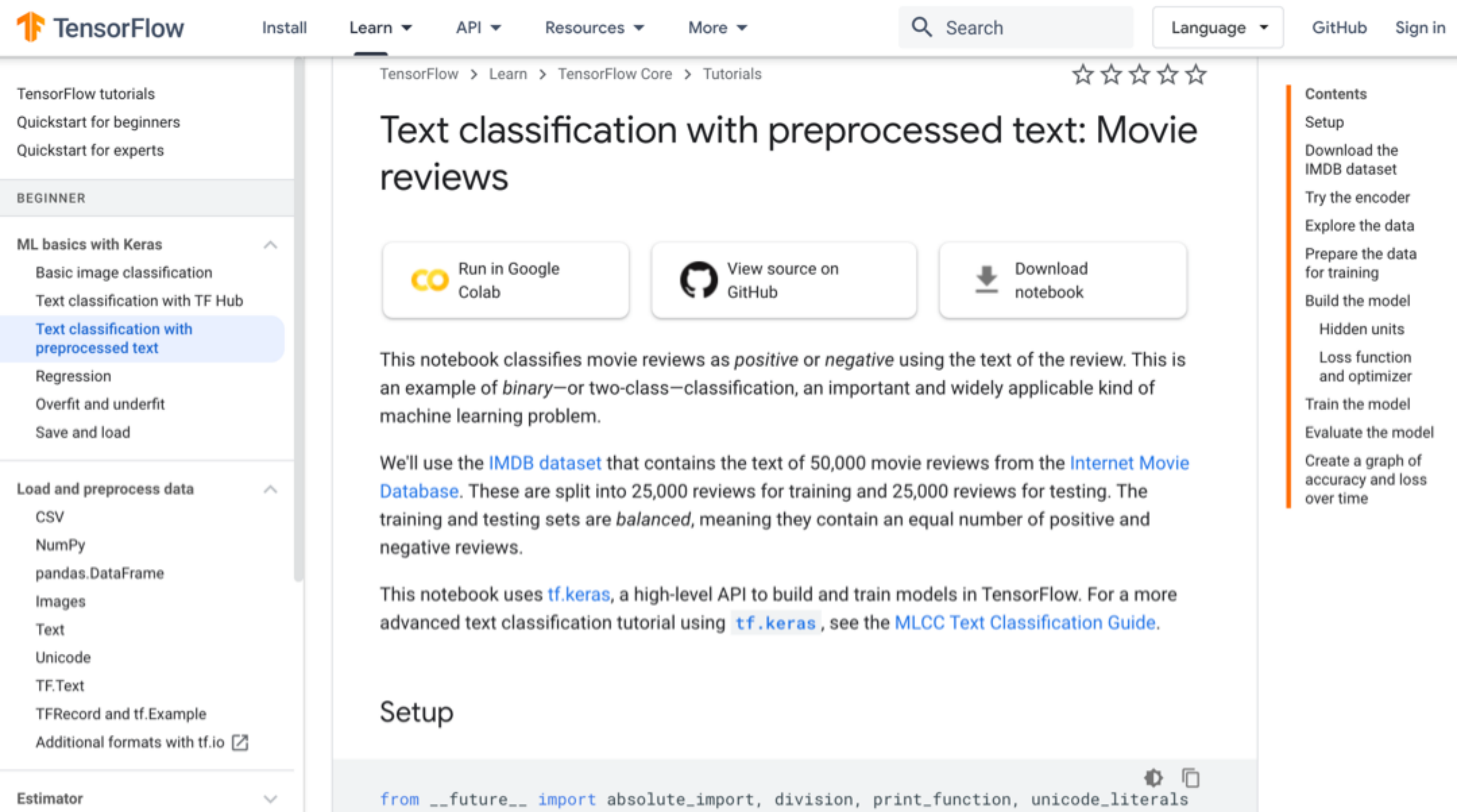
```
from __future__ import absolute_import, division, print_function, unicode_literals
```

Contents

https://www.tensorflow.org/tutorials/keras/text_classification_with_hub

TensorFlow

Text Classification with Pre Text



TensorFlow | Install | Learn | API | Resources | More | Search | Language | GitHub | Sign in

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
- CSV
- NumPy
- pandas.DataFrame
- Images
- Text
- Unicode
- TF.Text
- TFRecord and tf.Example
- Additional formats with tf.io

Estimator

TensorFlow > Learn > TensorFlow Core > Tutorials

Text classification with preprocessed text: Movie reviews

Run in Google Colab | View source on GitHub | Download notebook

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

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

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

Setup

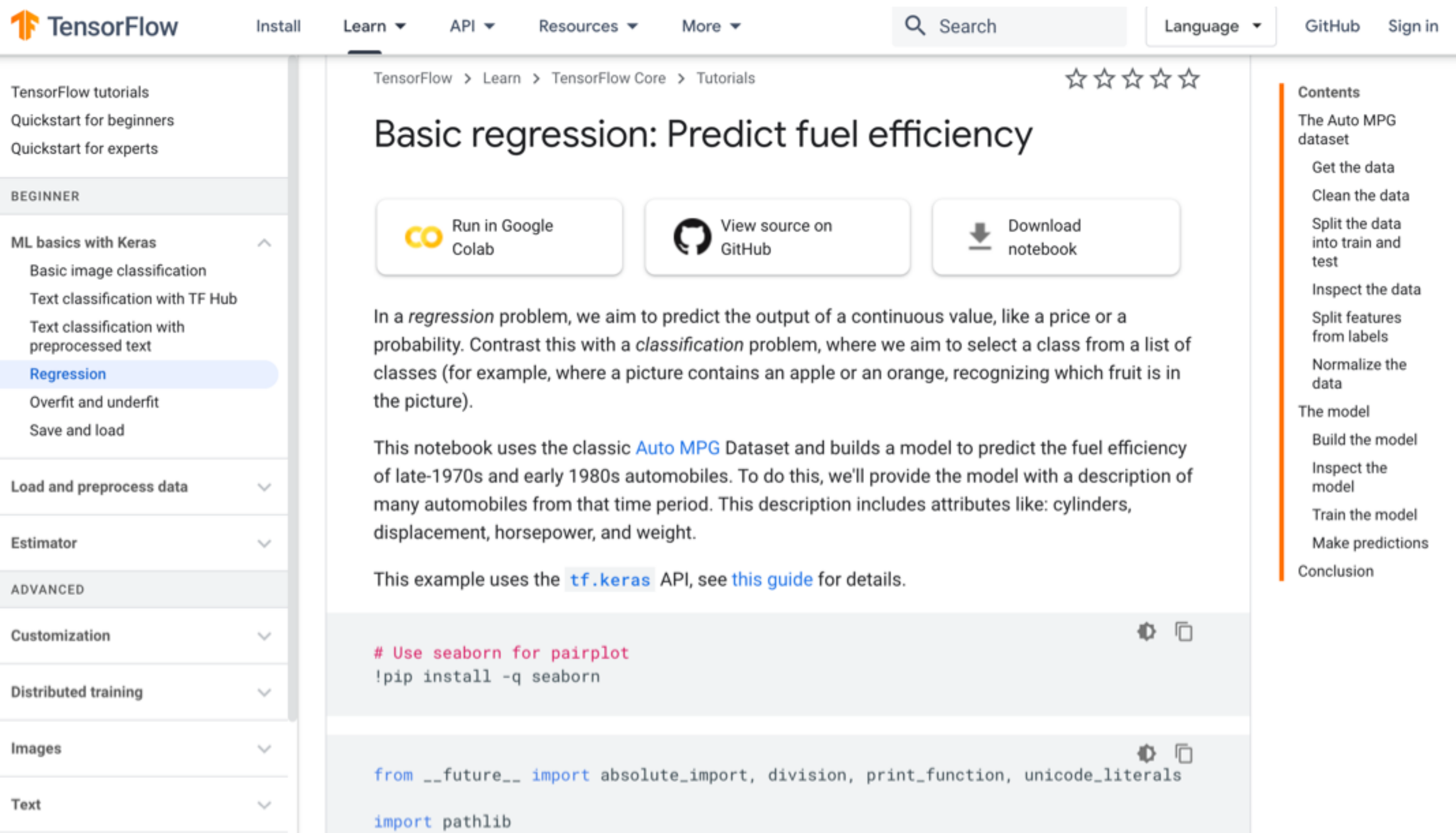
```
from __future__ import absolute_import, division, print_function, unicode_literals
```

Contents

https://www.tensorflow.org/tutorials/keras/text_classificati

TensorFlow

Regression



TensorFlow | Install | Learn | API | Resources | More | Search | Language | GitHub | Sign in

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data

Estimator

ADVANCED

Customization

Distributed training

Images

Text

TensorFlow > Learn > TensorFlow Core > Tutorials

Basic regression: Predict fuel efficiency

Run in Google Colab | View source on GitHub | Download notebook

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to select a class from a list of classes (for example, where a picture contains an apple or an orange, recognizing which fruit is in the picture).

This notebook uses the classic Auto MPG Dataset and builds a model to predict the fuel efficiency of late-1970s and early 1980s automobiles. To do this, we'll provide the model with a description of many automobiles from that time period. This description includes attributes like: cylinders, displacement, horsepower, and weight.

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

```
# Use seaborn for pairplot
!pip install -q seaborn
```

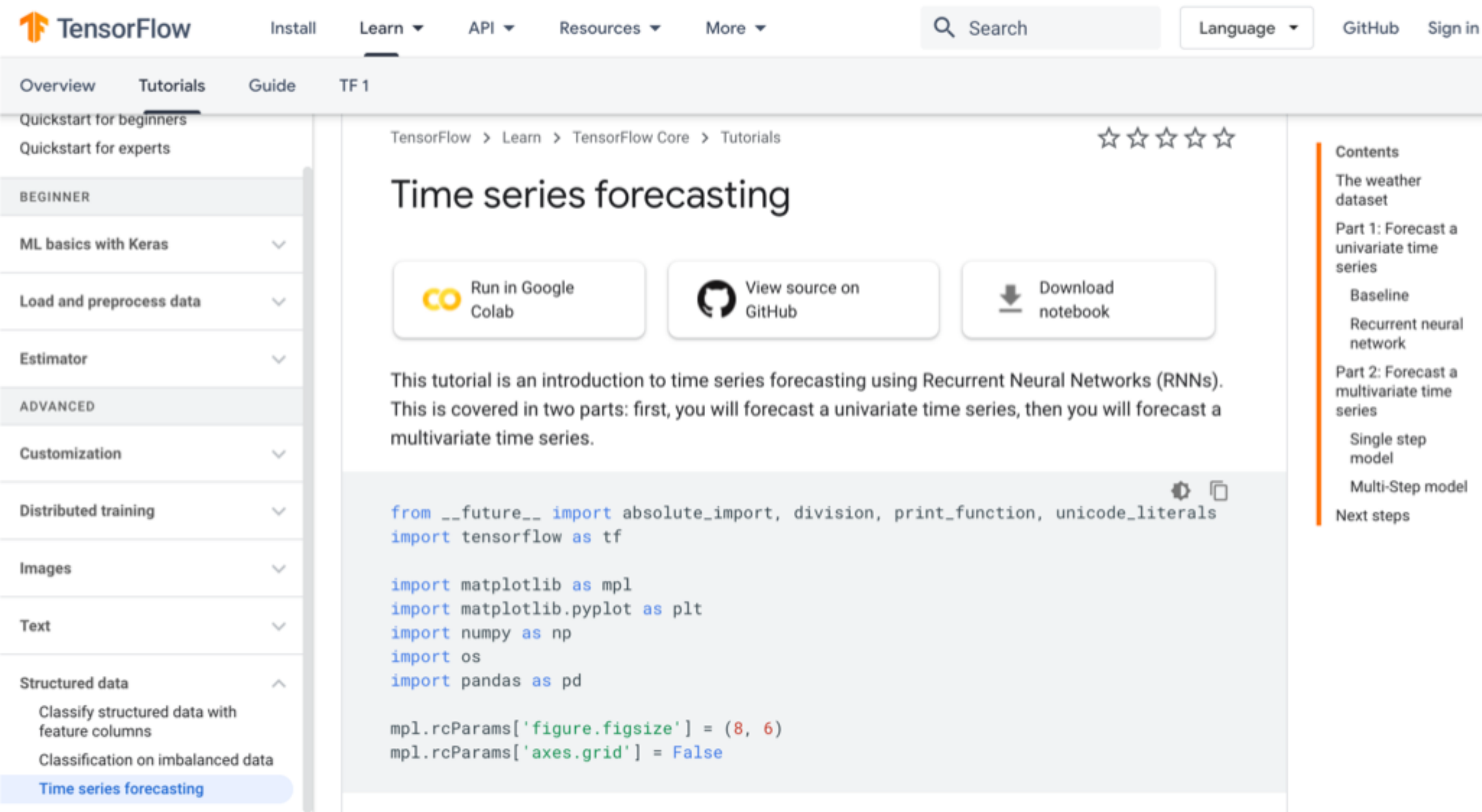
```
from __future__ import absolute_import, division, print_function, unicode_literals

import pathlib
```

Contents

https://www.tensorflow.org/tutorials/keras/regression

TensorFlow

TensorFlow 2.0
Time Series Forecasting



https://www.tensorflow.org/tutorials/structured_data/time_series

Time Series Data

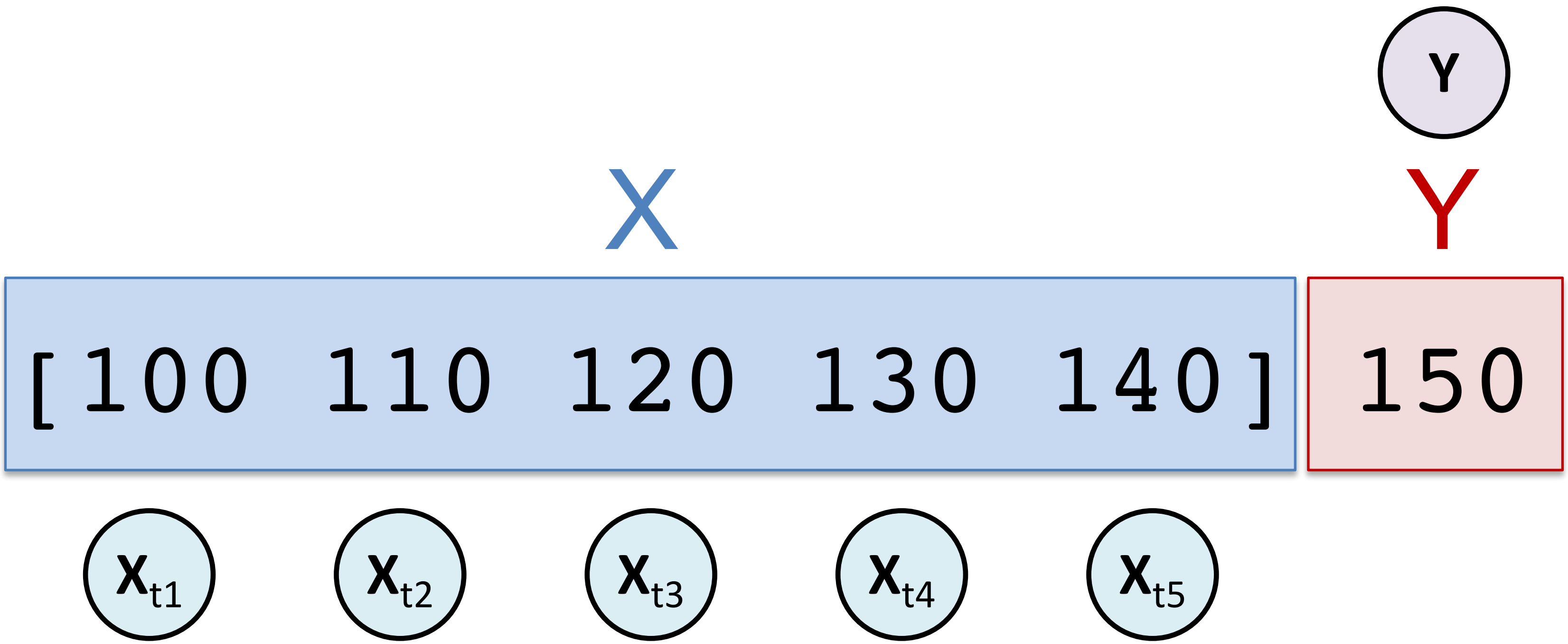
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

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

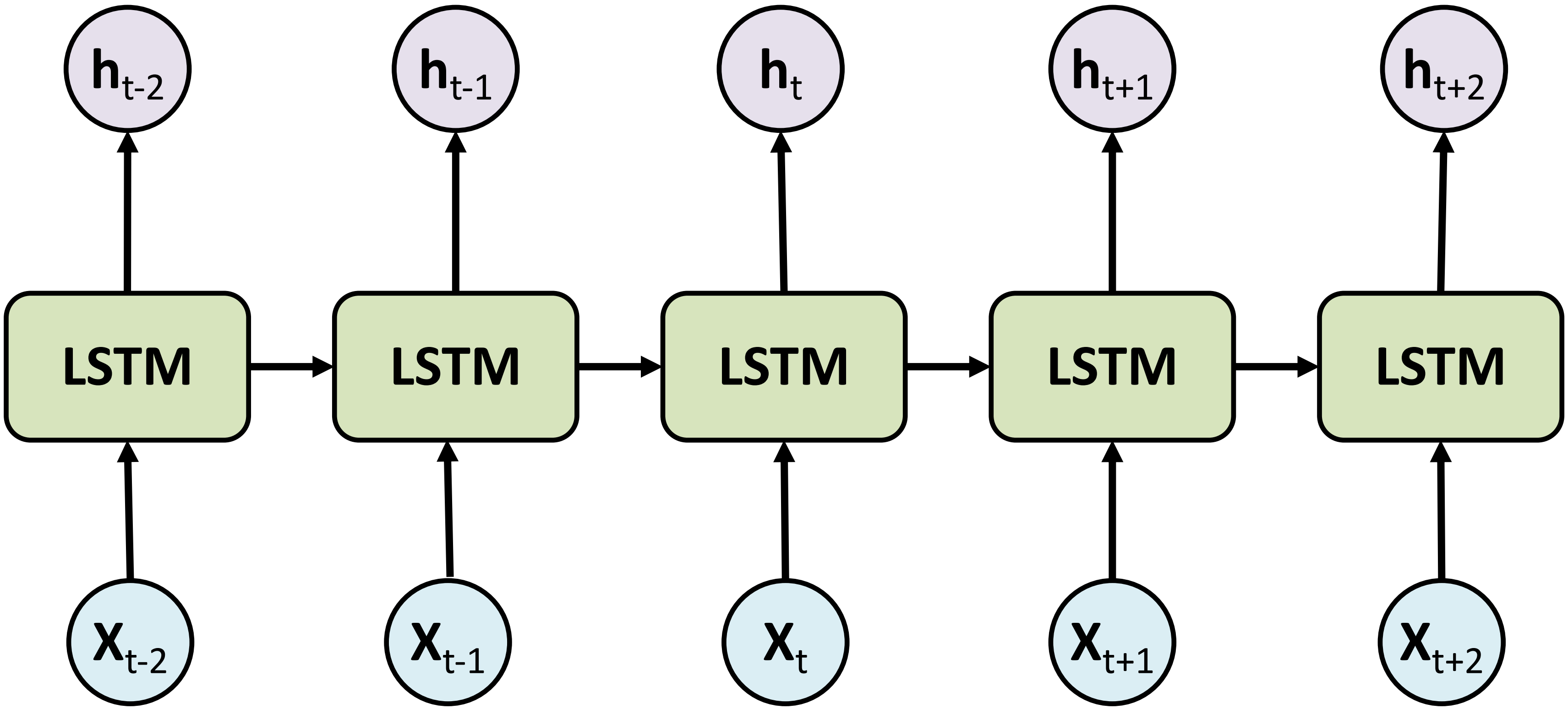
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Time Series Data

[100, 110, 120, 130, 140, 150]



Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

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

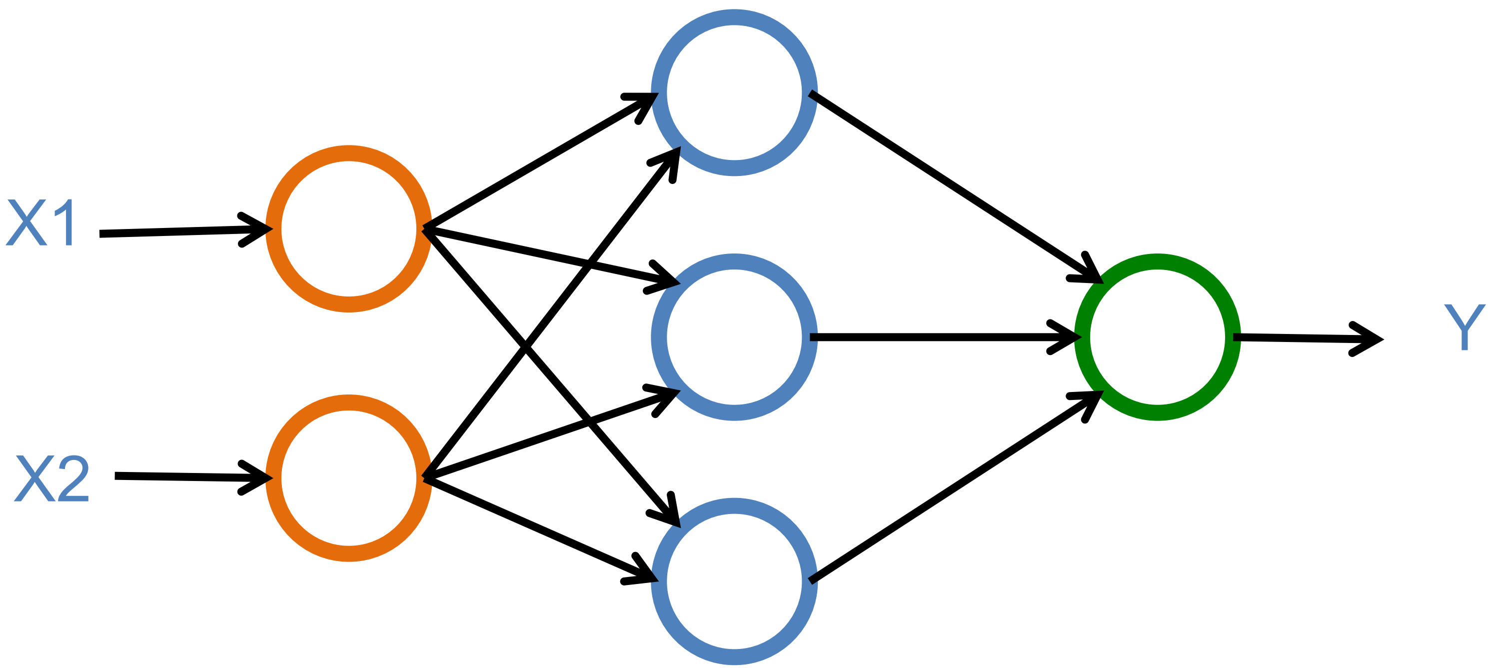
X	Y
[10 20 30]	40
[20 30 40]	50
[30 40 50]	60
[40 50 60]	70
[50 60 70]	80
[60 70 80]	90

Deep Learning
and
Neural Networks

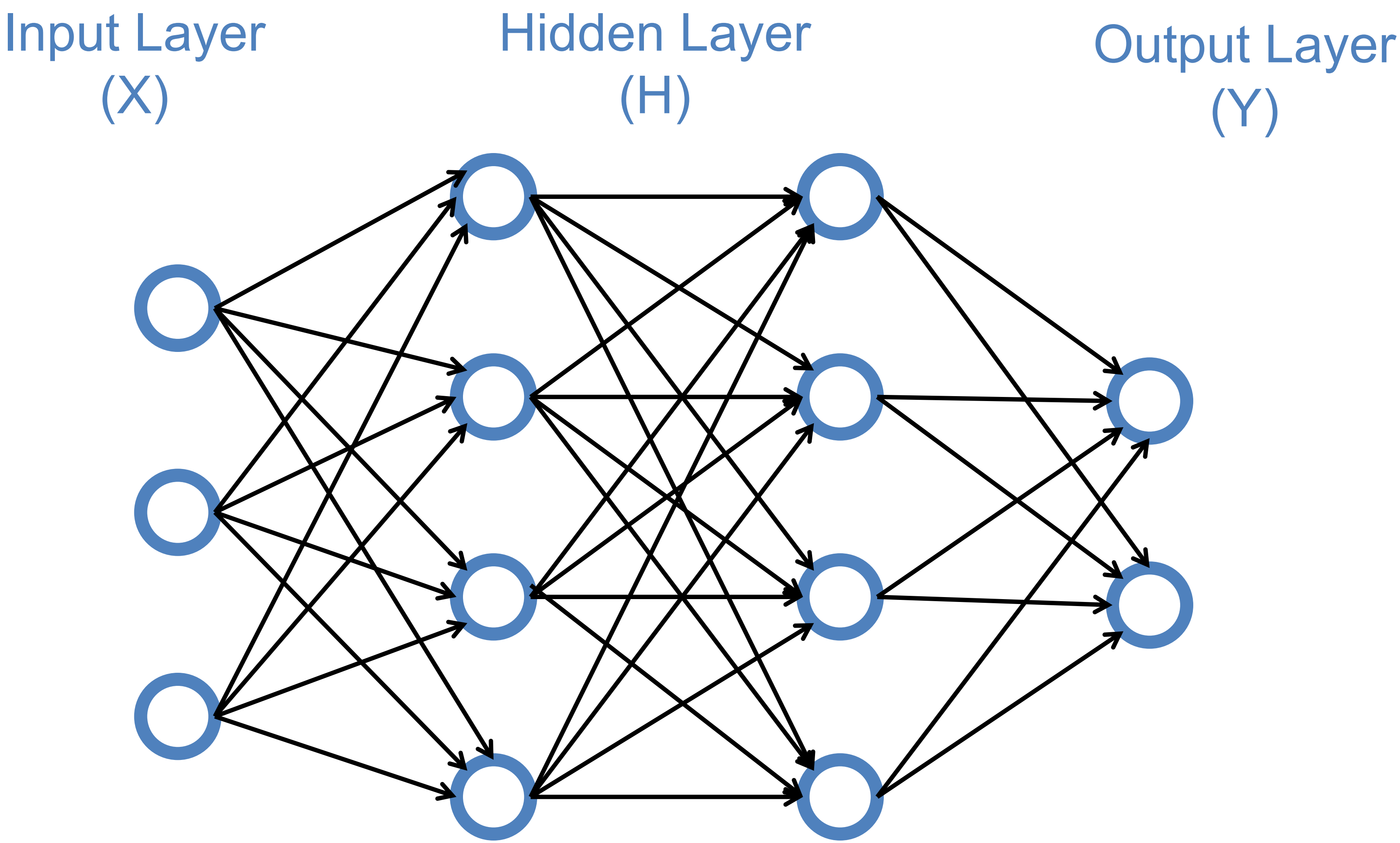
Deep Learning Foundations: Neural Networks

Deep Learning and Neural Networks

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



Deep Learning and Neural Networks



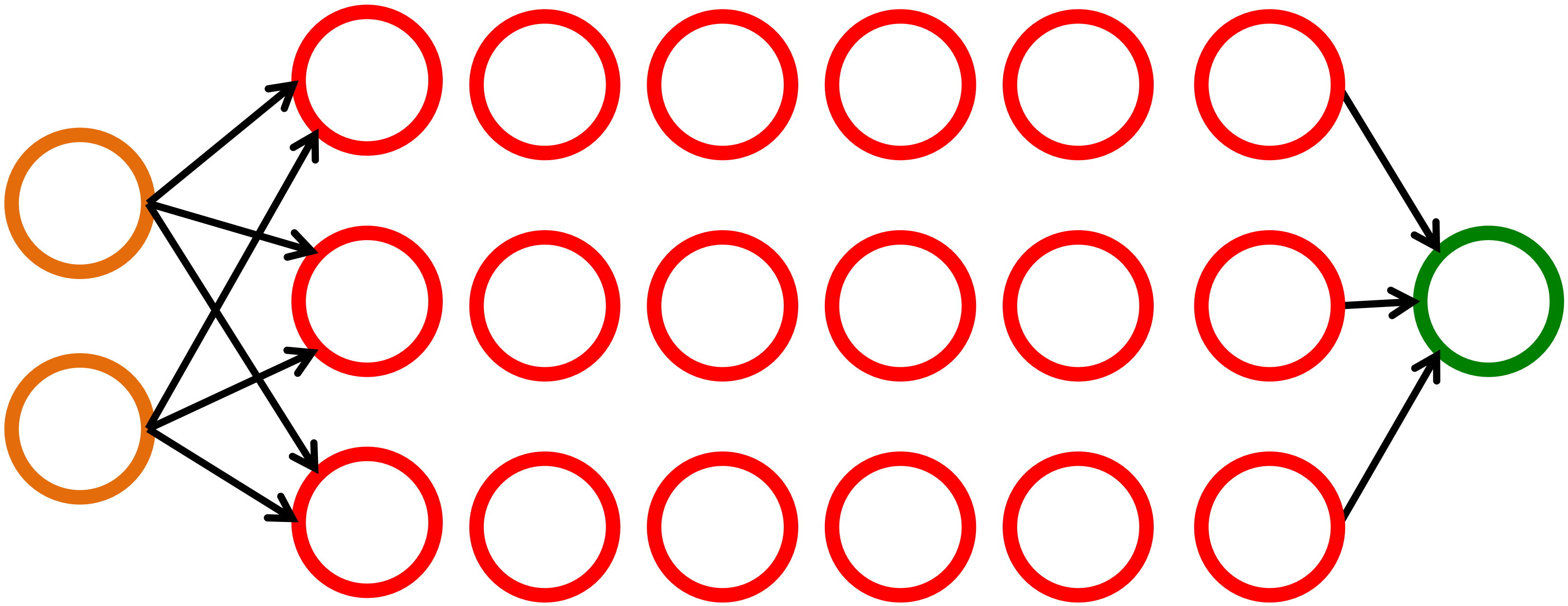
Deep Learning and Neural Networks

Input Layer
(X)

Hidden Layers
(H)

Output Layer
(Y)

Deep Neural Networks
Deep Learning



Deep Learning
and
Deep Neural Networks

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

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

REVIEW

doi:10.1038/nature14539

Deep learning

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

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

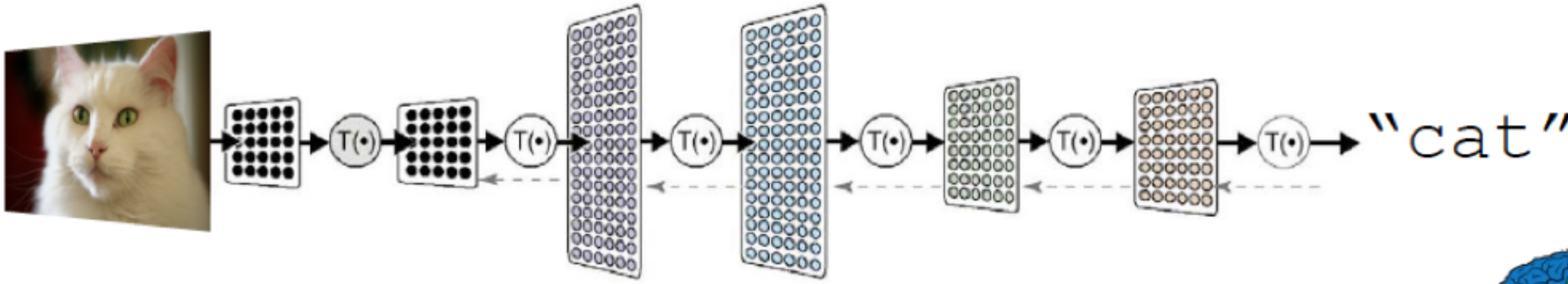
Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects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

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

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

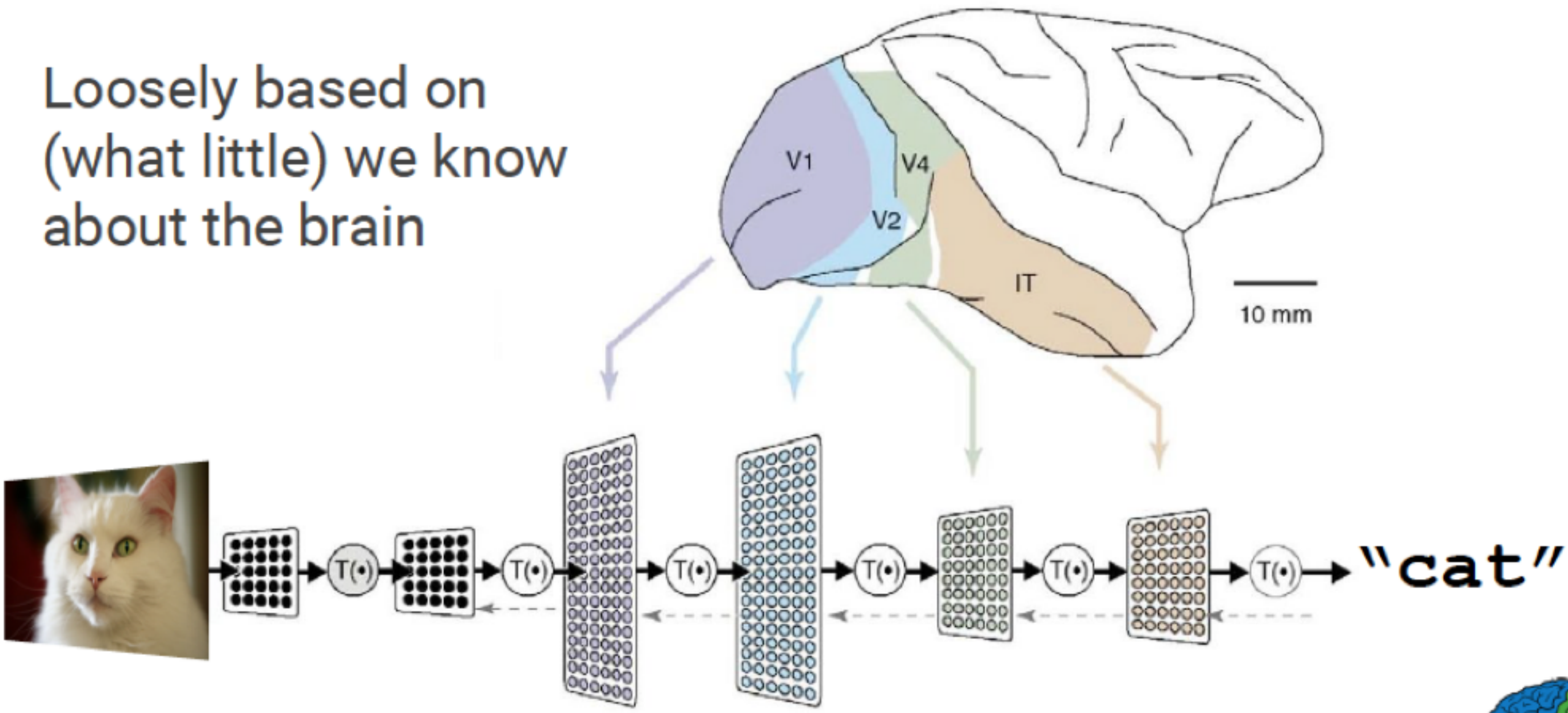
Deep Learning

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

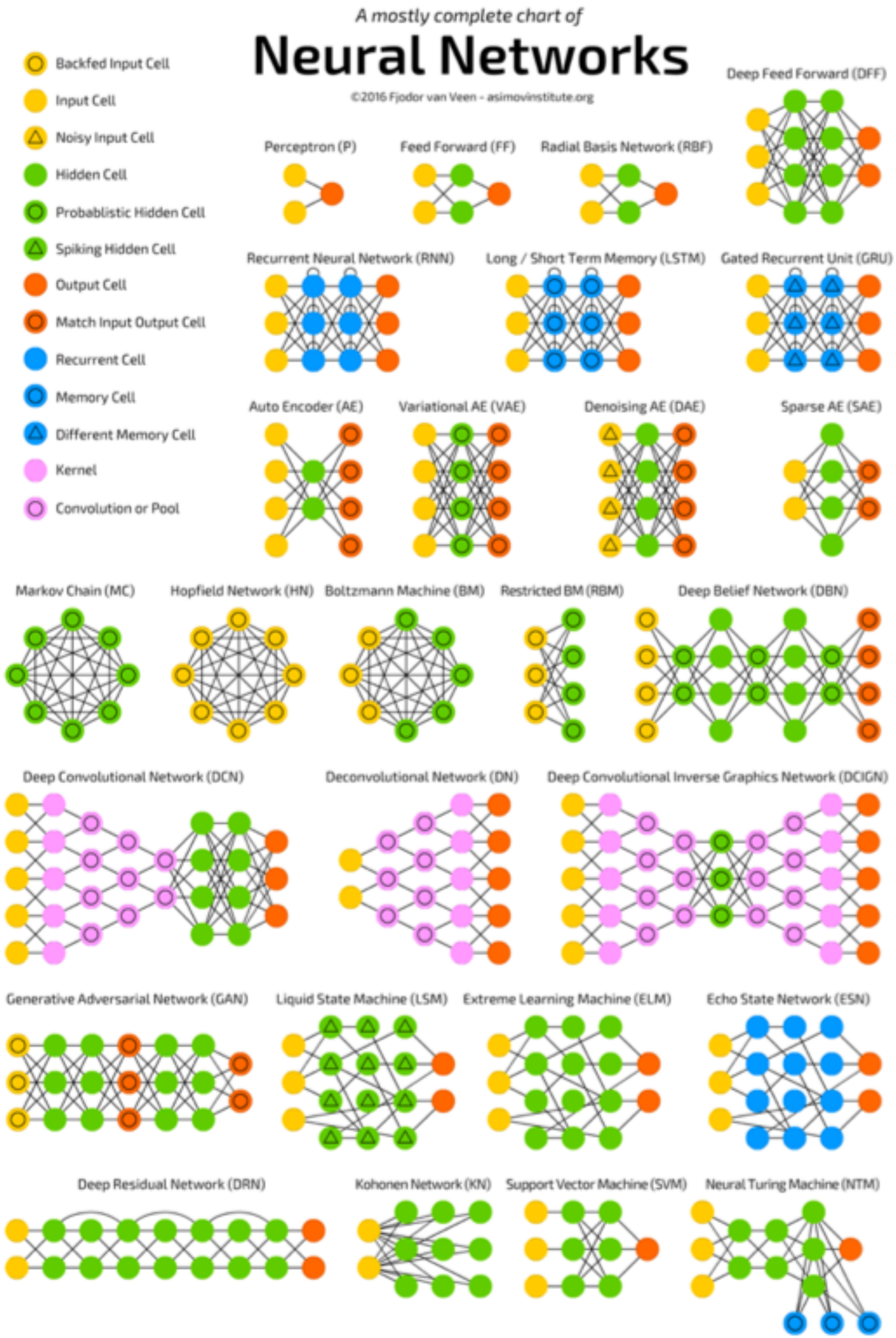


What is Deep Learning?

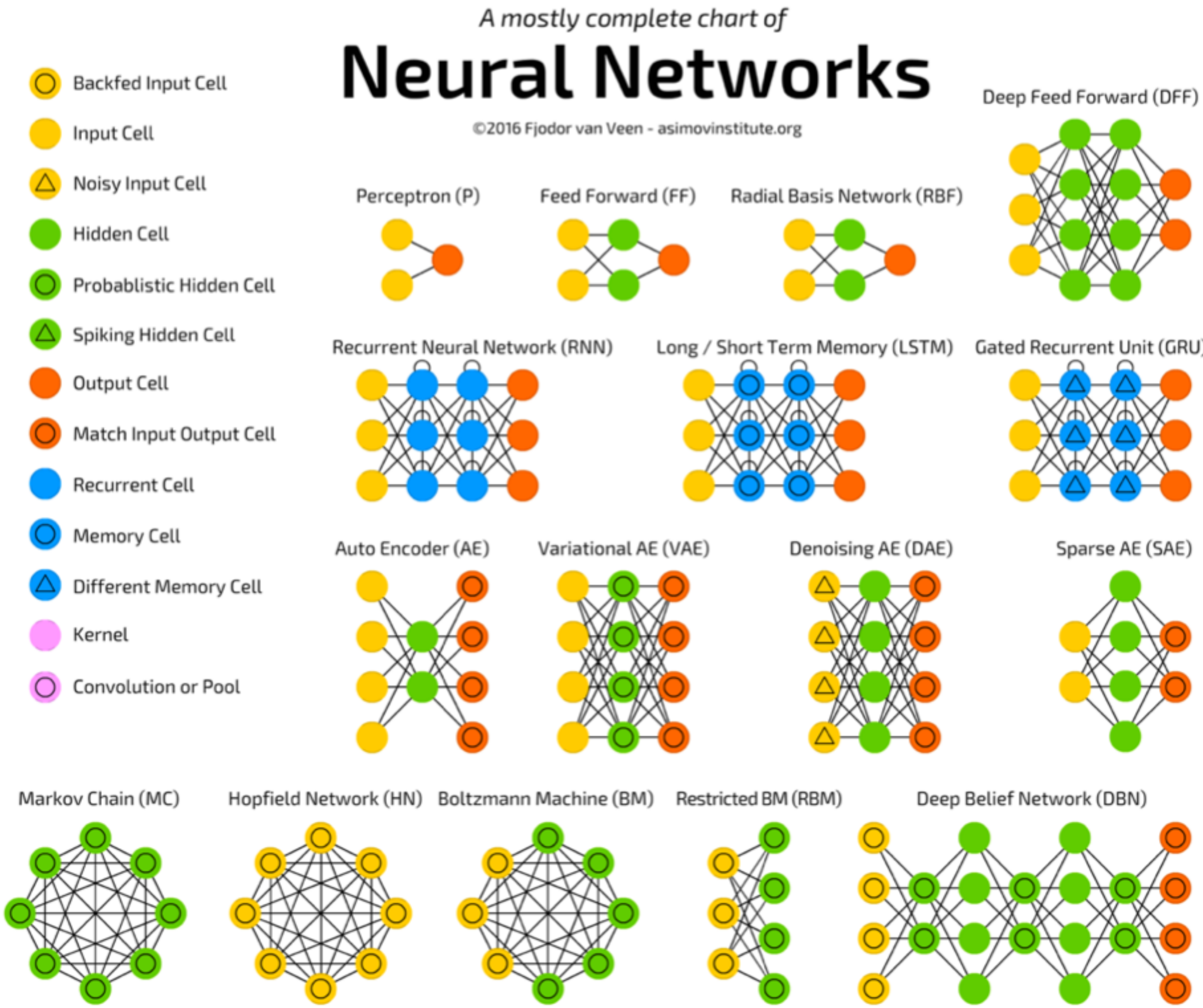
- Loosely based on (what little) we know about the brain



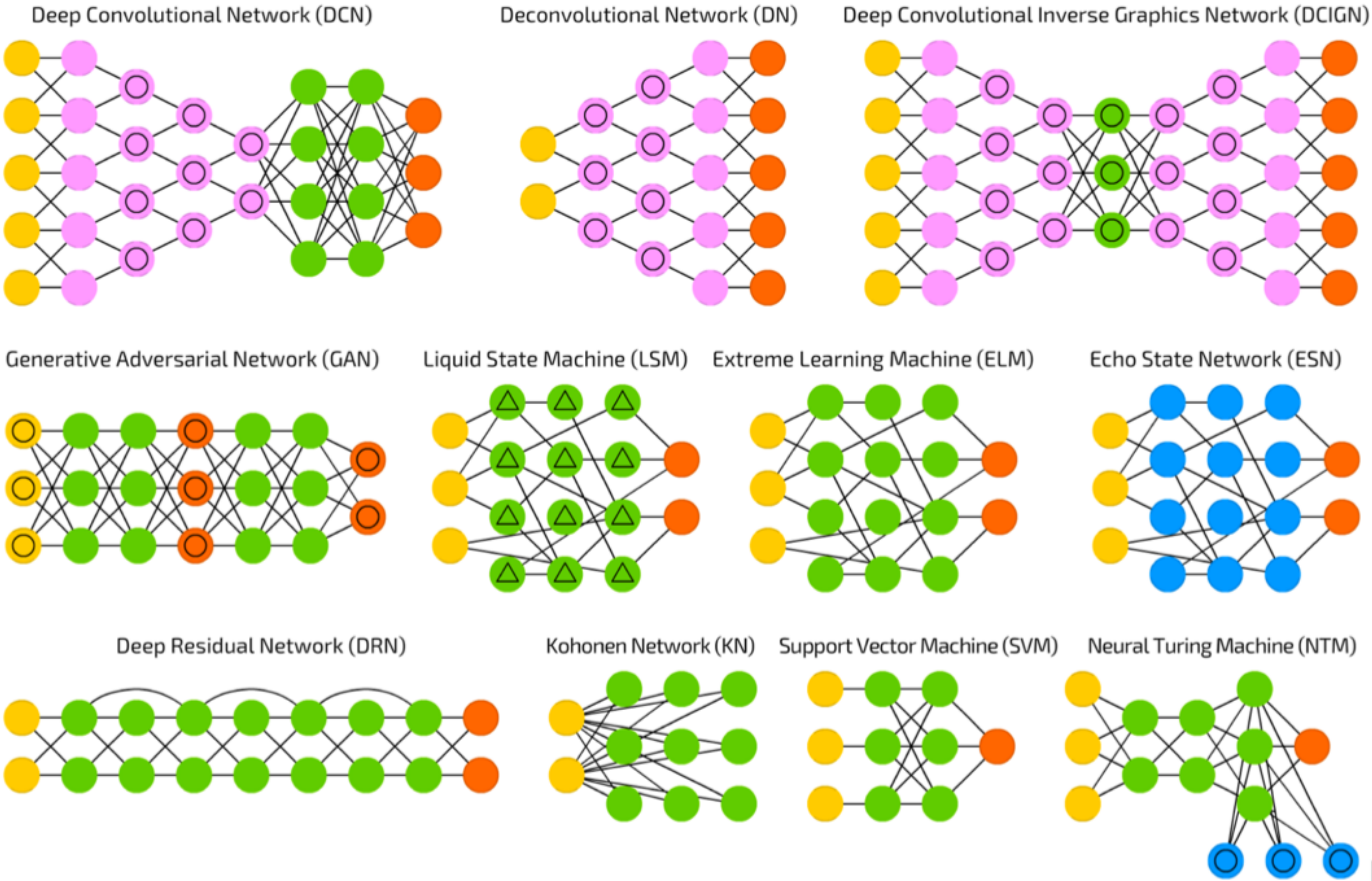
Neural Networks (NN)



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



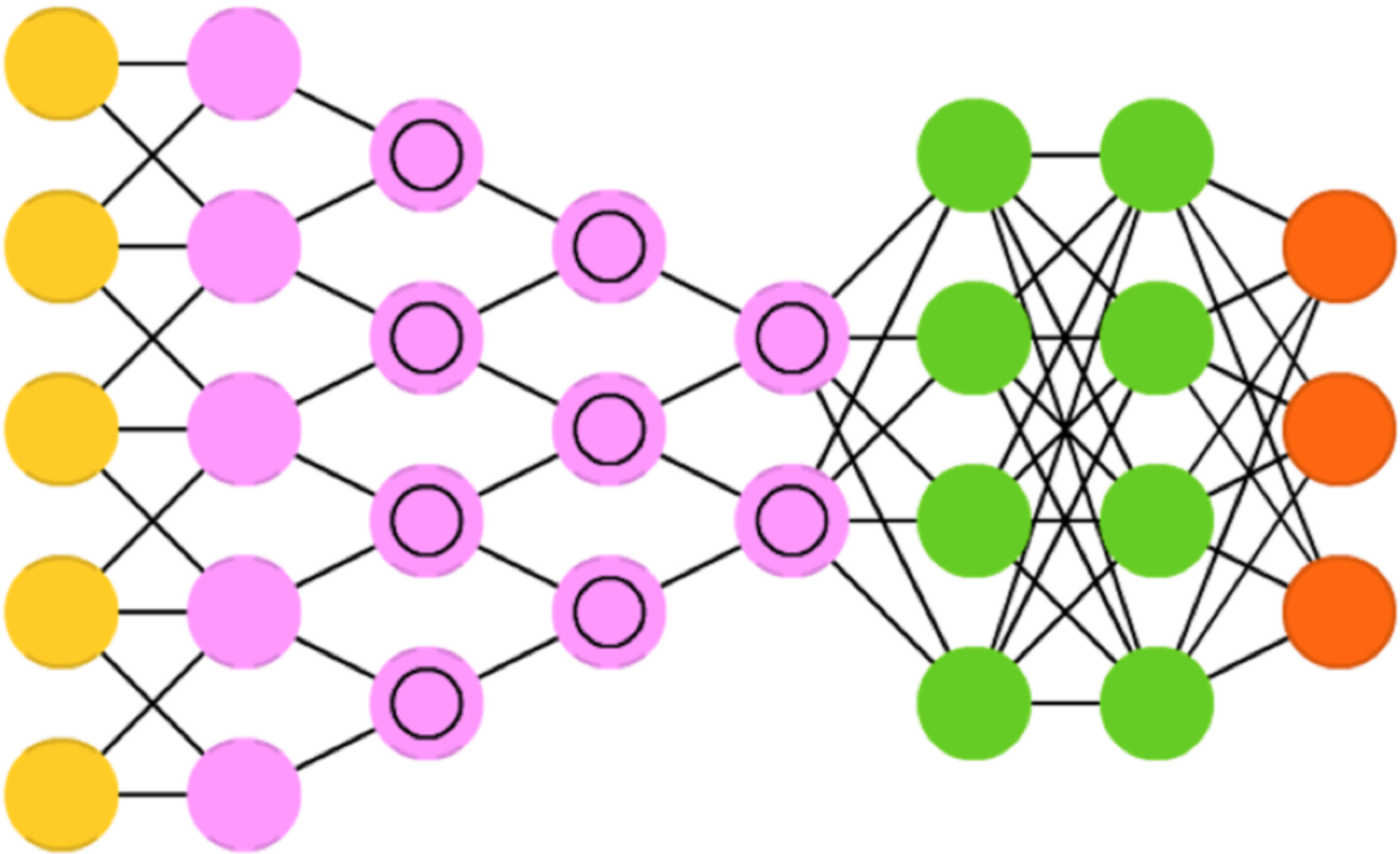
Source: http://www.asimovinstitute.org/neural-network-zoo/



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

Convolutional Neural Networks

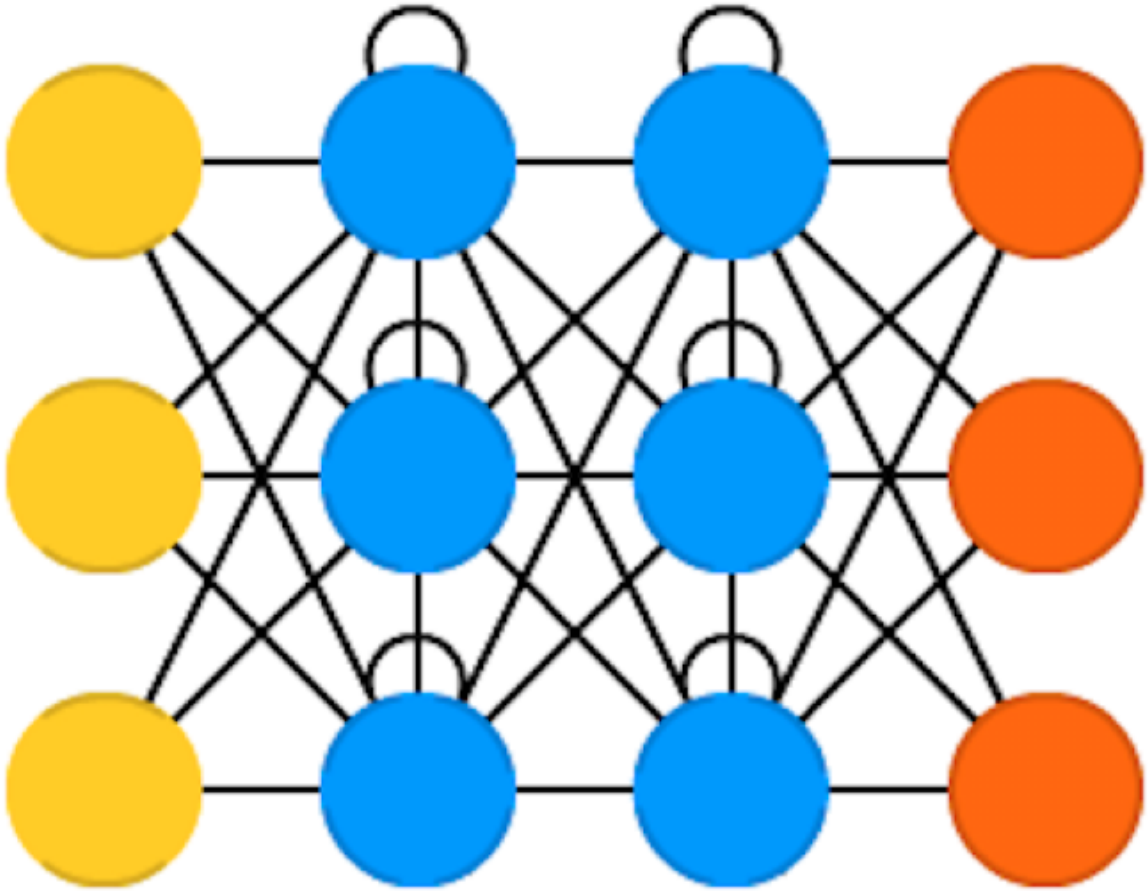
(CNN or Deep Convolutional Neural Networks, DCNN)



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

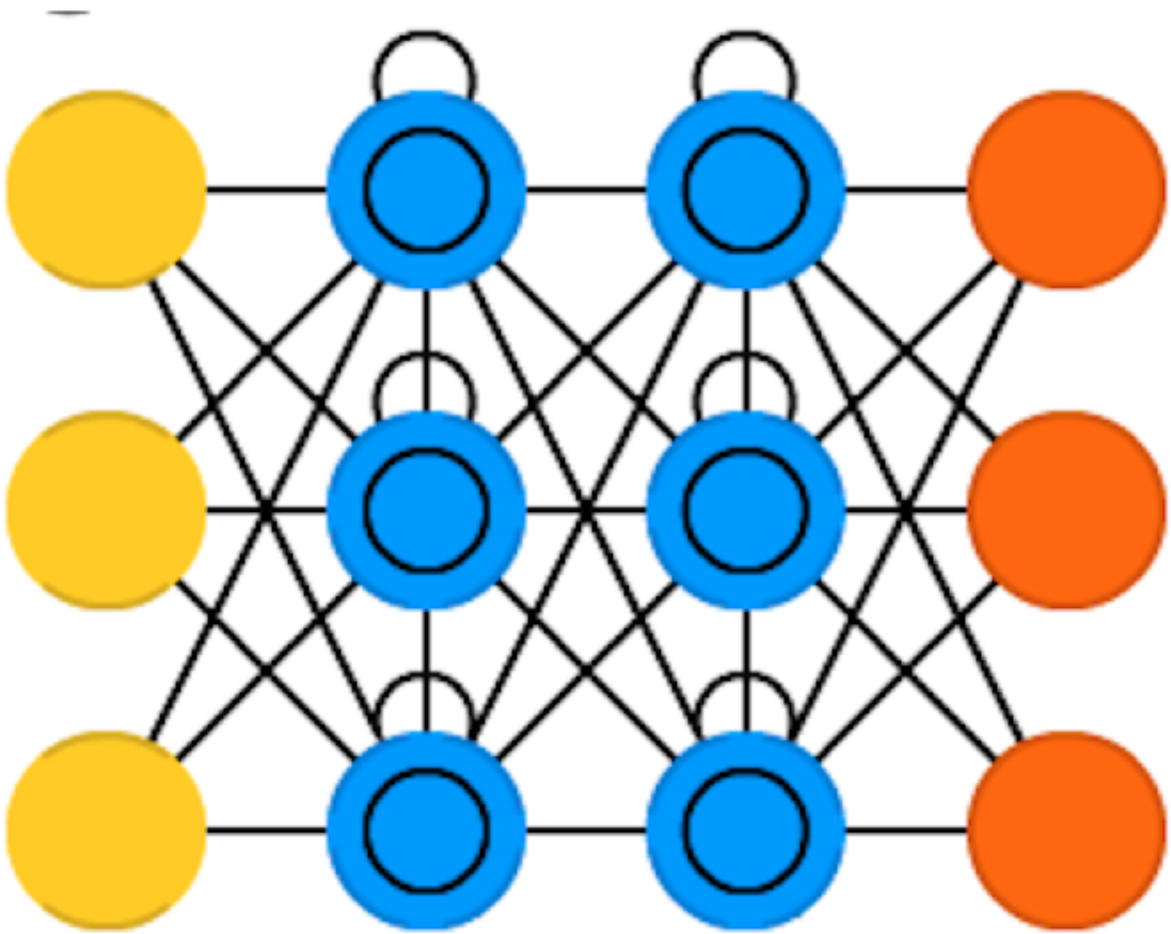
Source: http://www.asimovinstitute.org/neural-network-zoo/

Recurrent Neural Networks (RNN)



Elman, Jeffrey L. “Finding structure in time.” Cognitive science 14.2 (1990): 179-211
Source: http://www.asimovinstitute.org/neural-network-zoo/

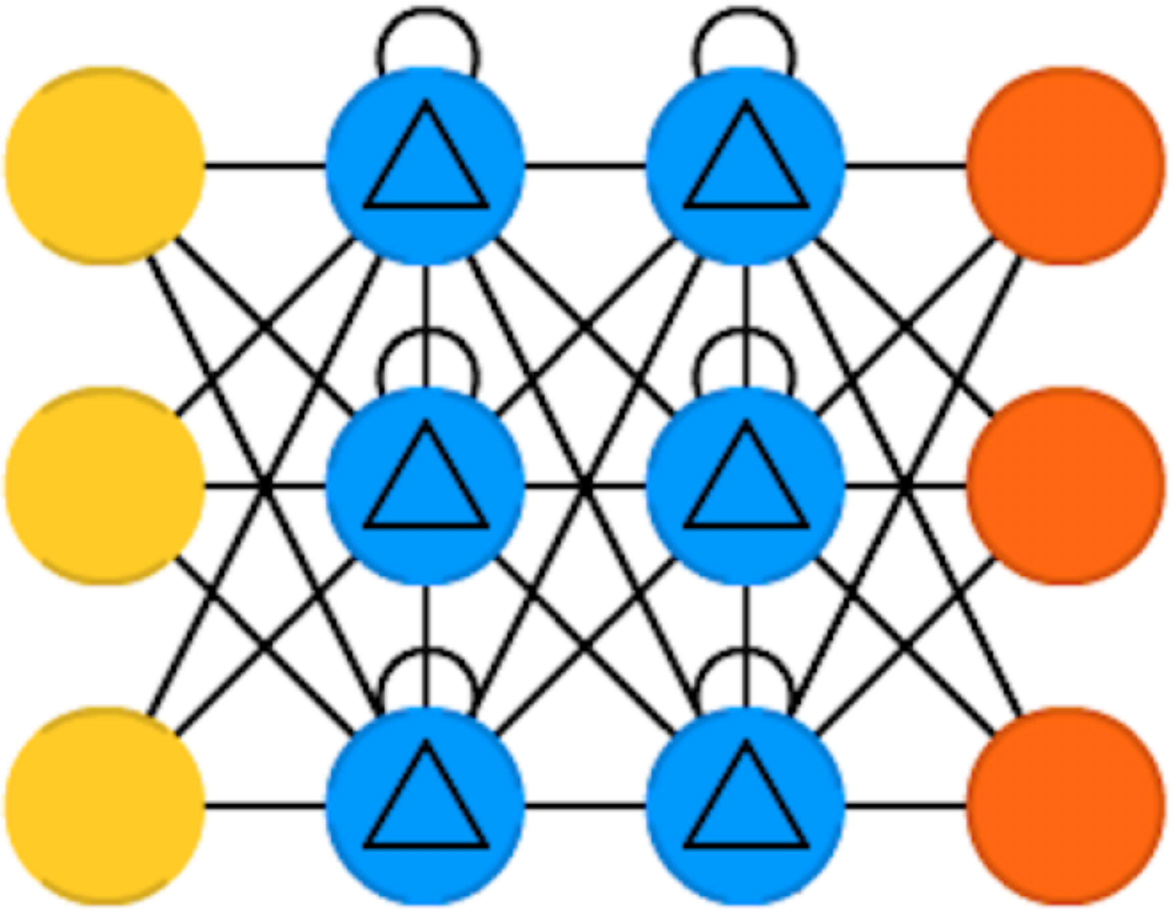
Long / Short Term Memory (LSTM)



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

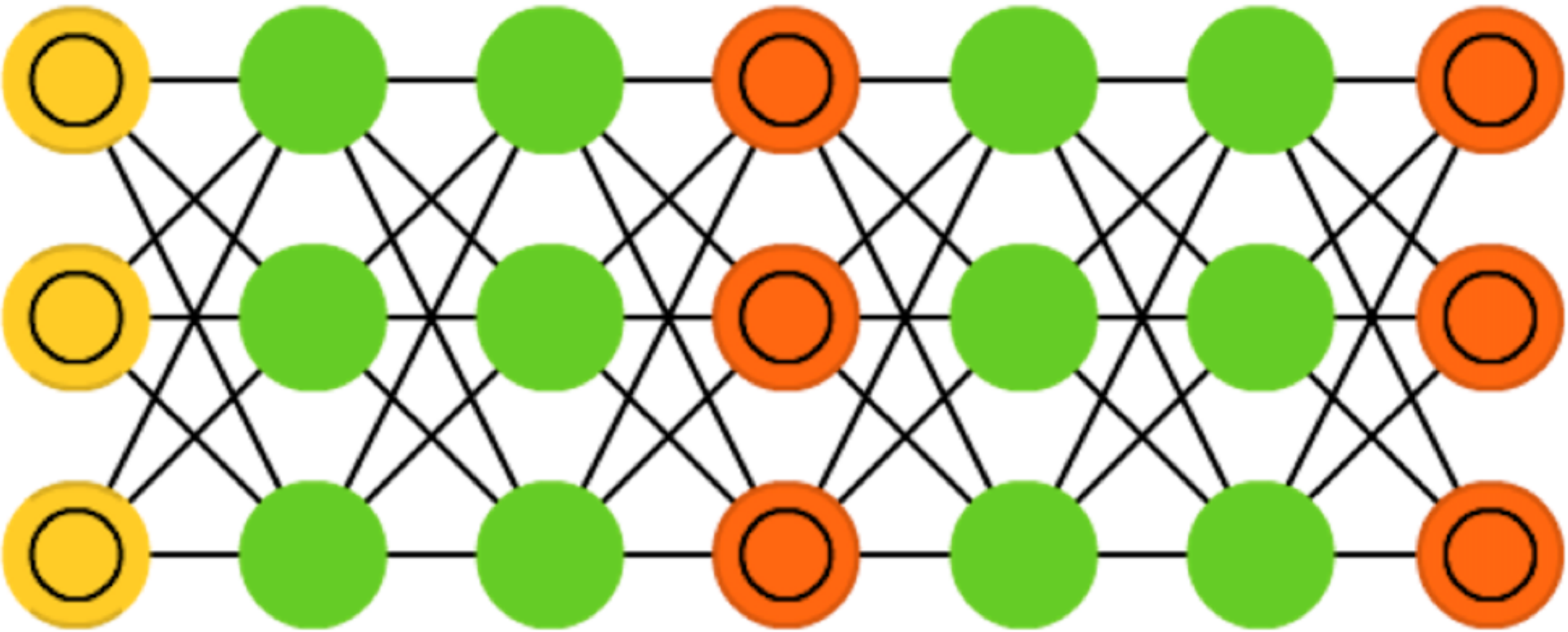
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



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

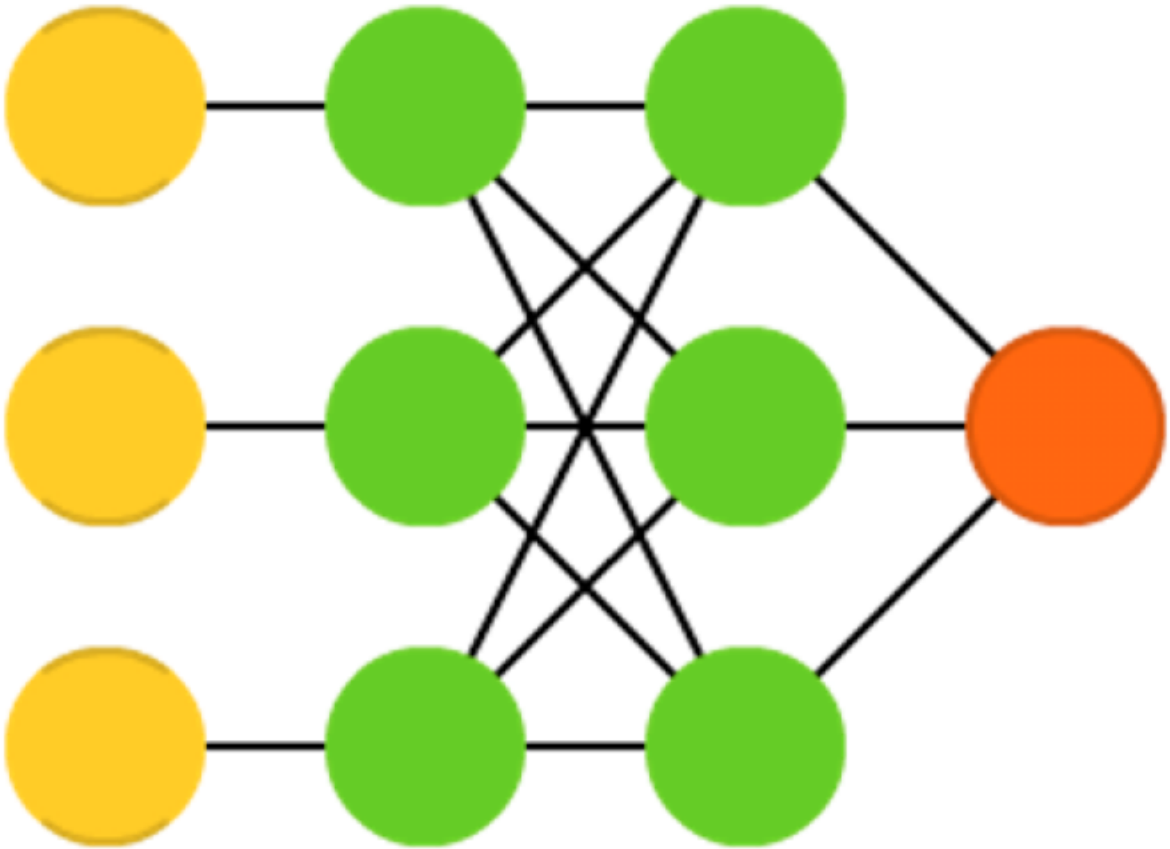
Generative Adversarial Networks (GAN)



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

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

Support Vector Machines (SVM)



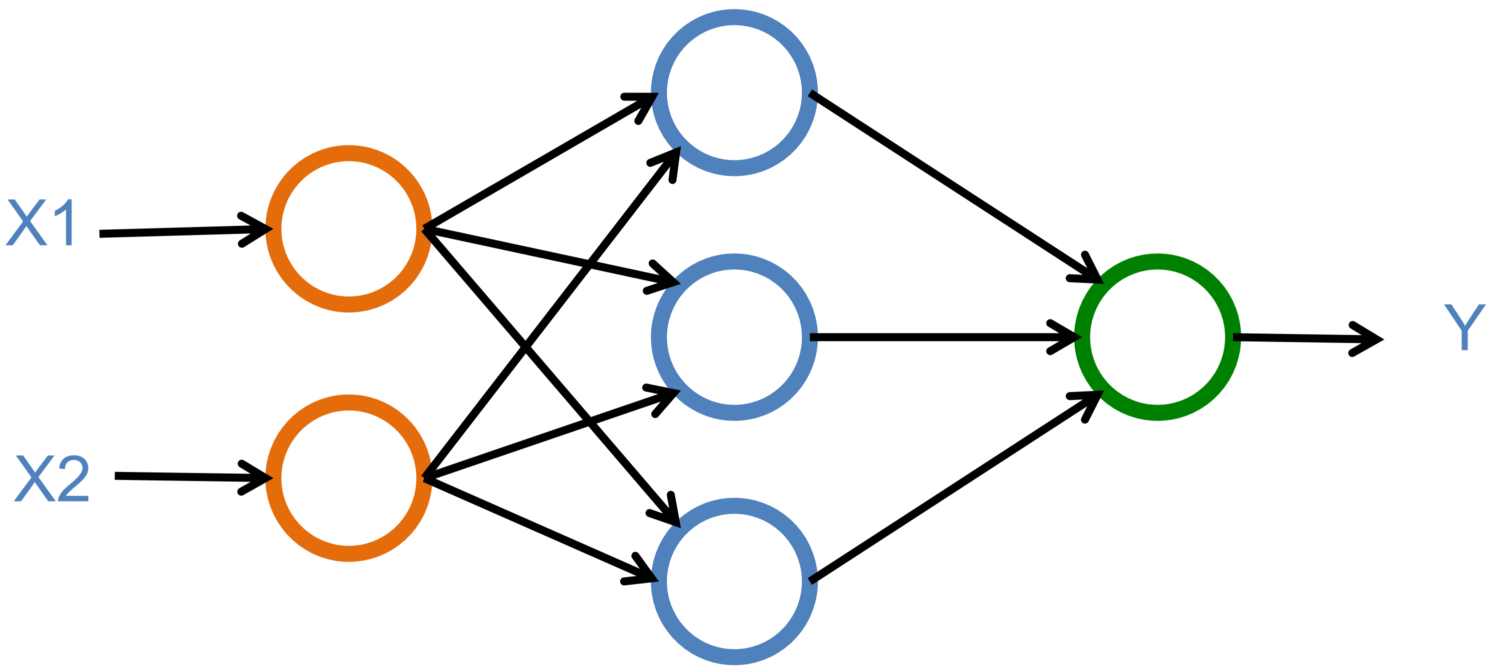
Cortes, Corinna, and Vladimir Vapnik. “Support-vector networks.” Machine learning 20.3 (1995): 273-297.

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

Neural networks (NN) 1960

Neural Networks

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



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

Multilayer Perceptrons (MLP) 1985

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

Support Vector Machine (SVM) 1995

Hinton presents the

Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST

2005

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

Deep
Recurrent Neural Network
(RNN)
2009

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

Convolutional DBN 2010

Max-Pooling CDBN
2011

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

Deep Learning

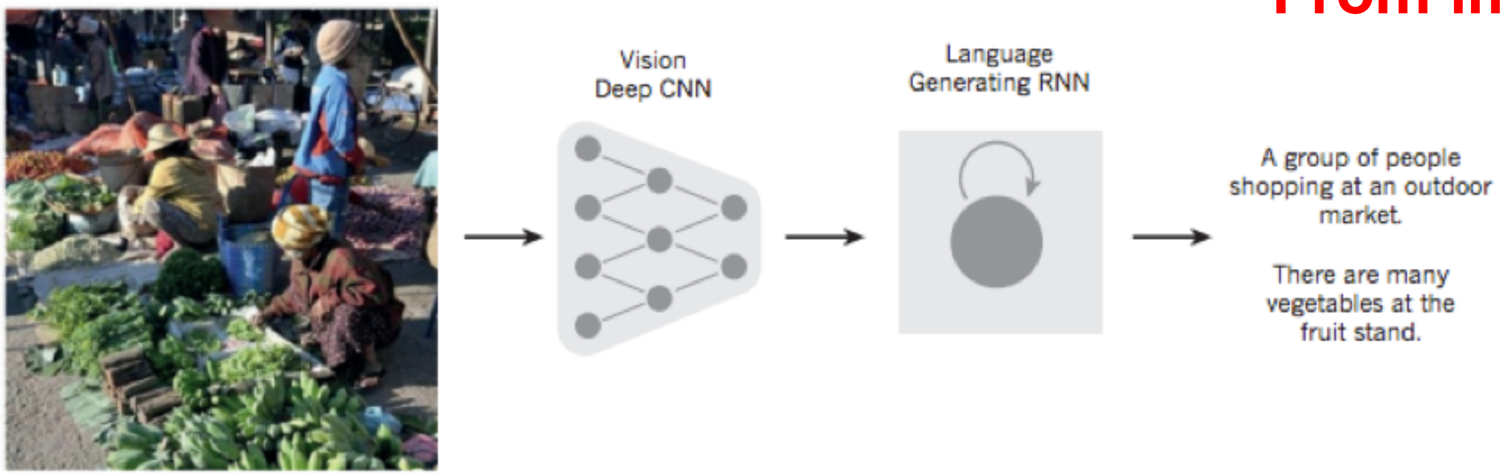
Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng

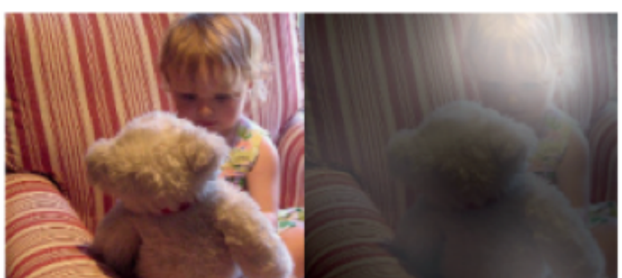
From image to text



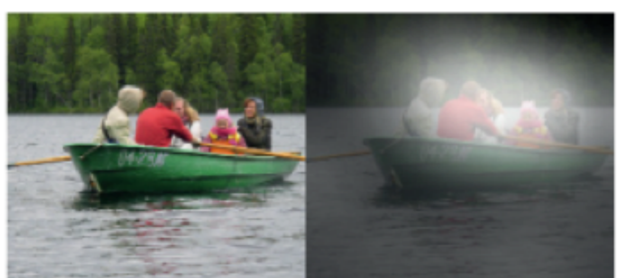
A woman is throwing a **frisbee** in a park.

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

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



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



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

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

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

From image to text

Image: deep convolution neural network (CNN)

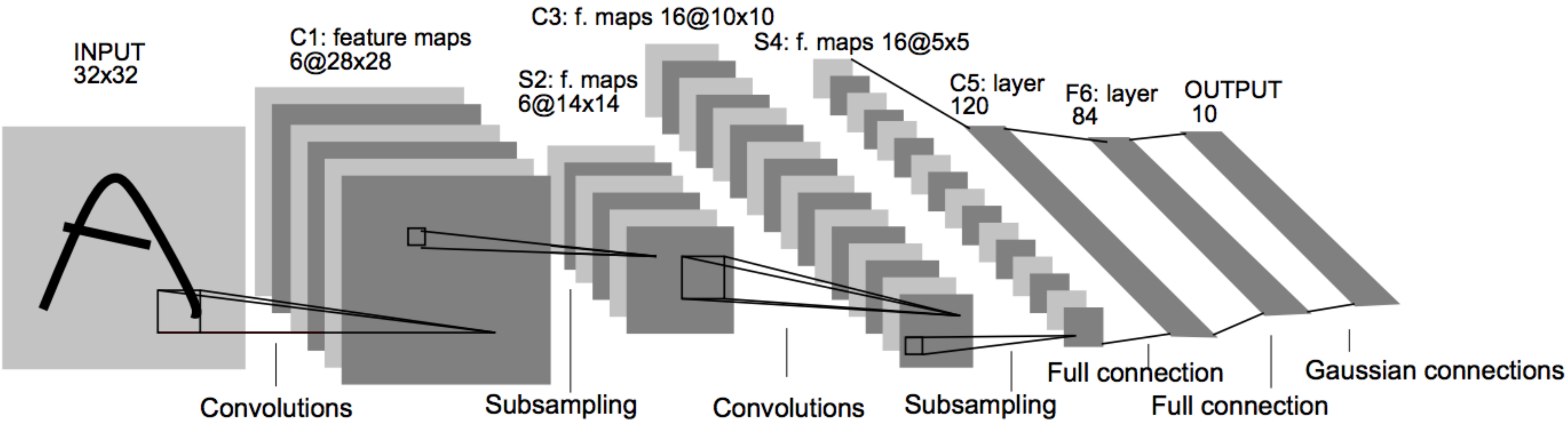
Text: recurrent neural network (RNN)

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

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

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN)



Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)

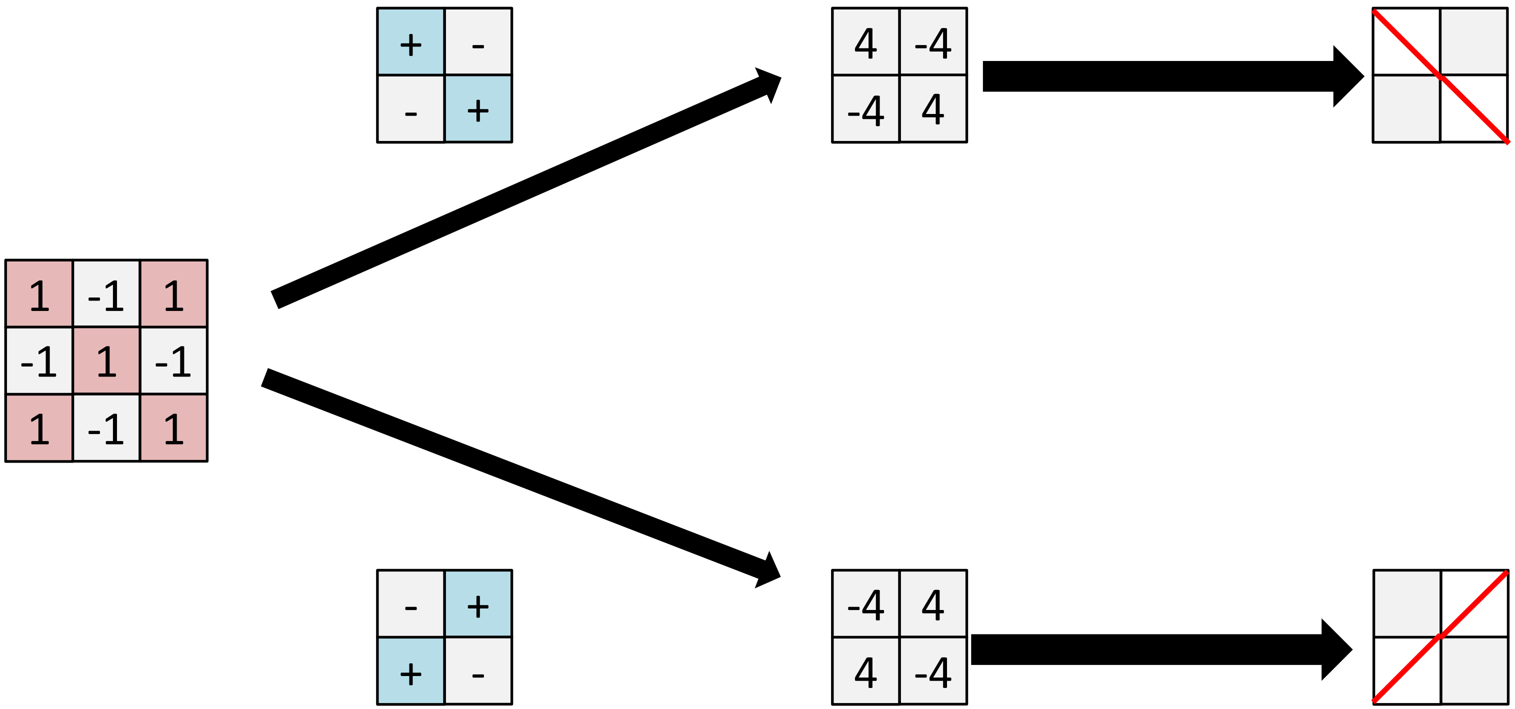
Source: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Source: LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)

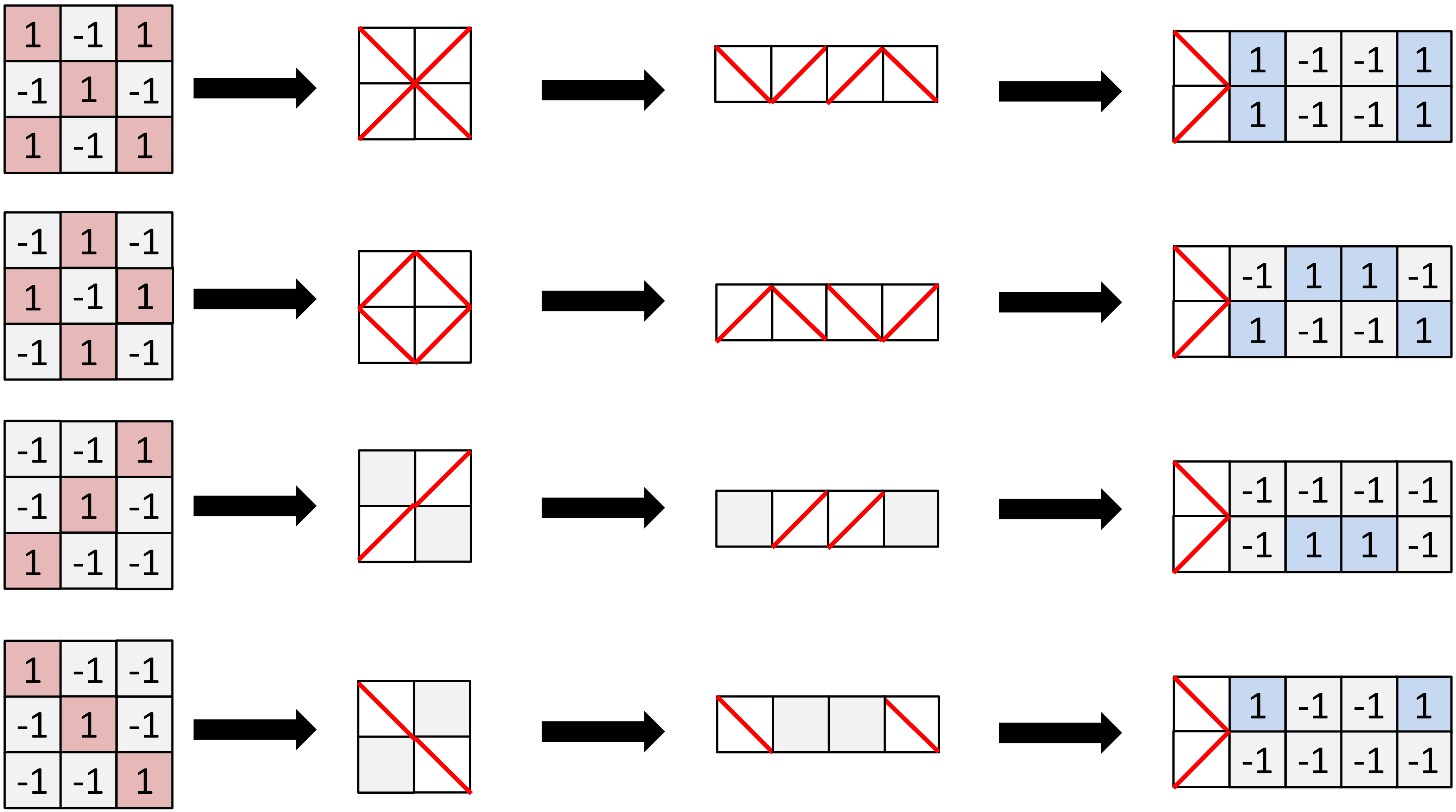
A friendly introduction to Convolutional Neural Networks and Image Recognition



Convolution Layer Pooling Layer

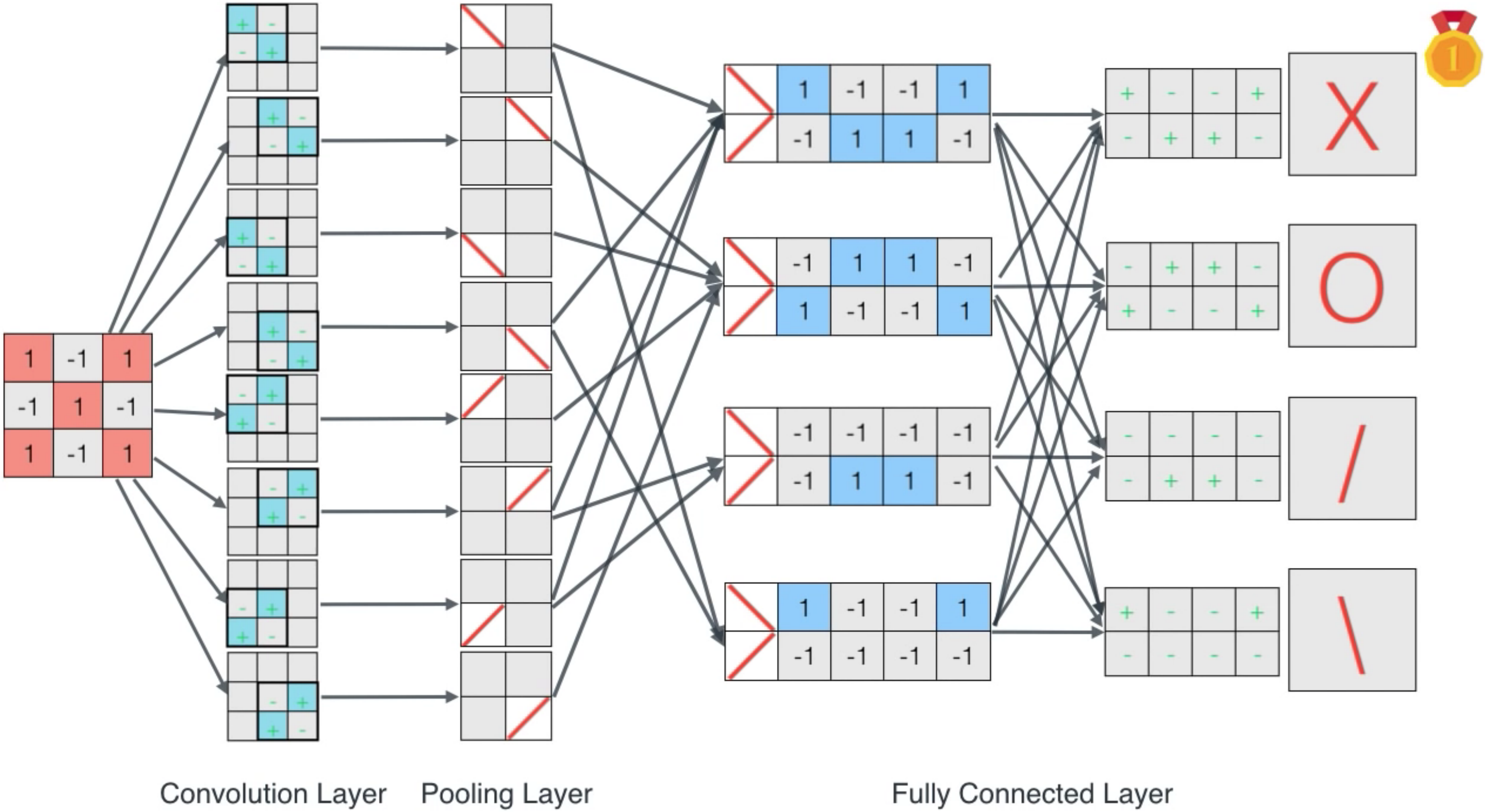
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



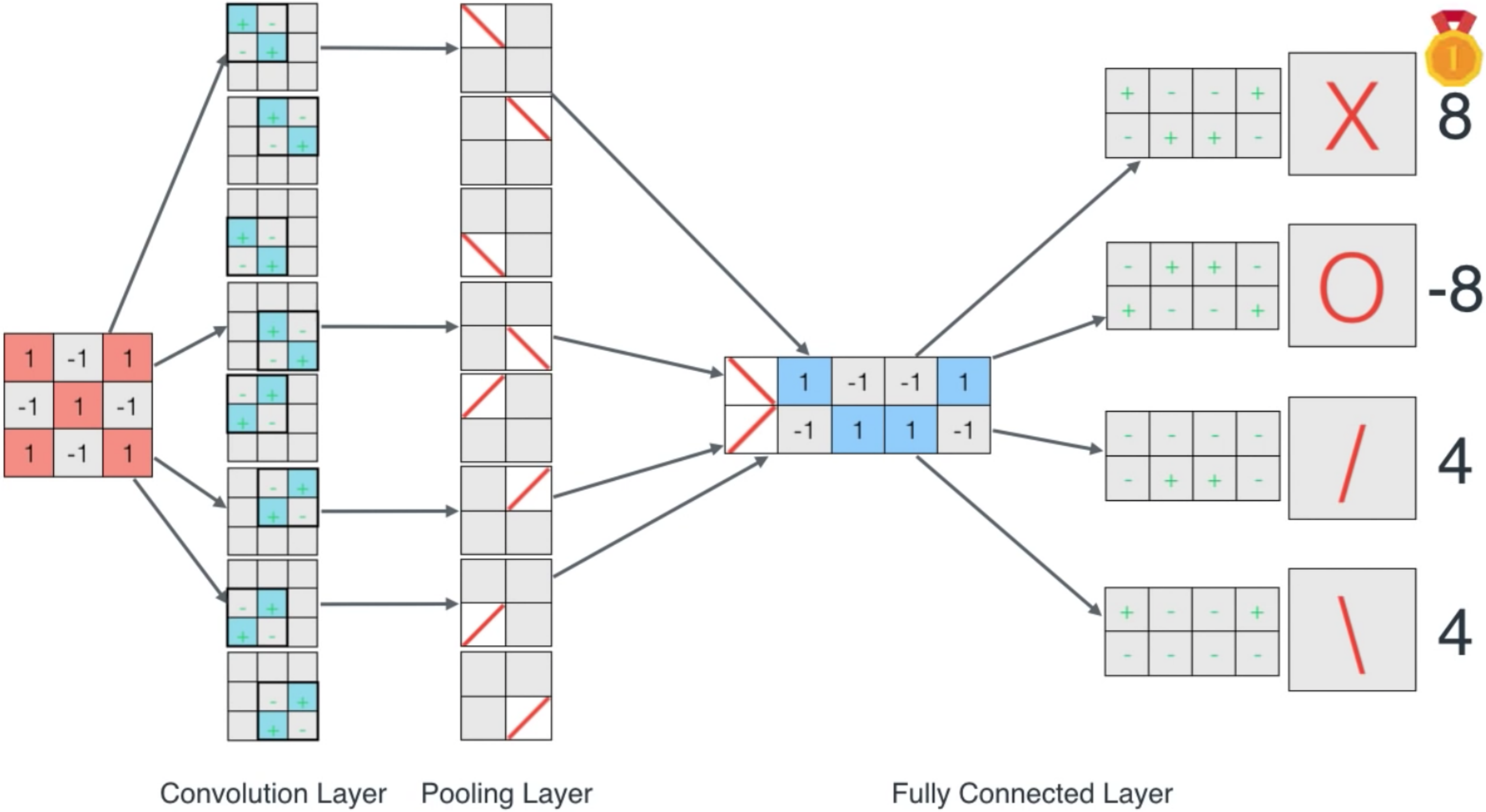
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



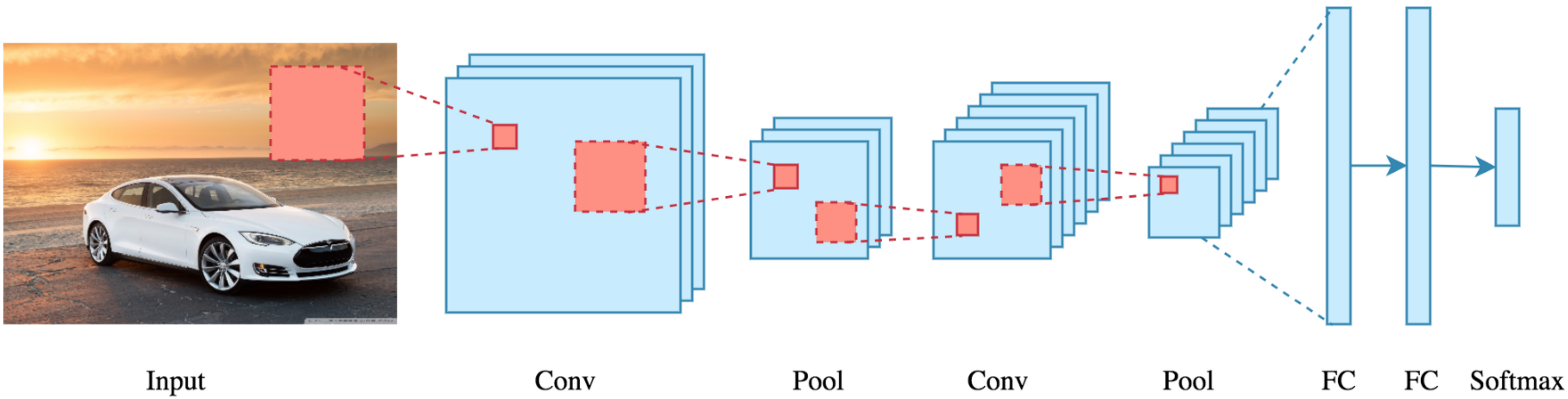
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

CNN Architecture



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Convolution is a **mathematical operation** to **merge two sets** of information

3x3 convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

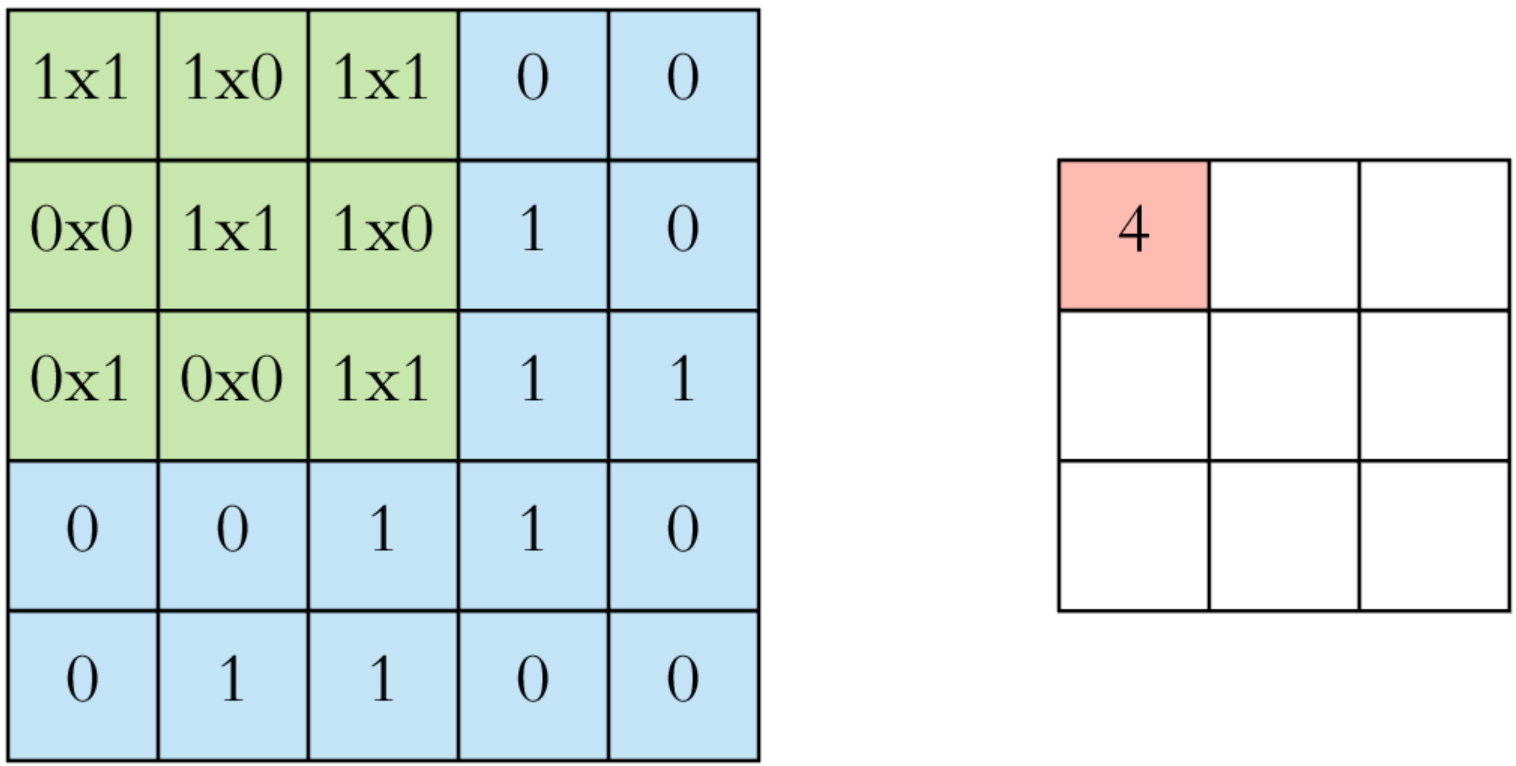
1	0	1
0	1	0
1	0	1

Filter / Kernel

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3



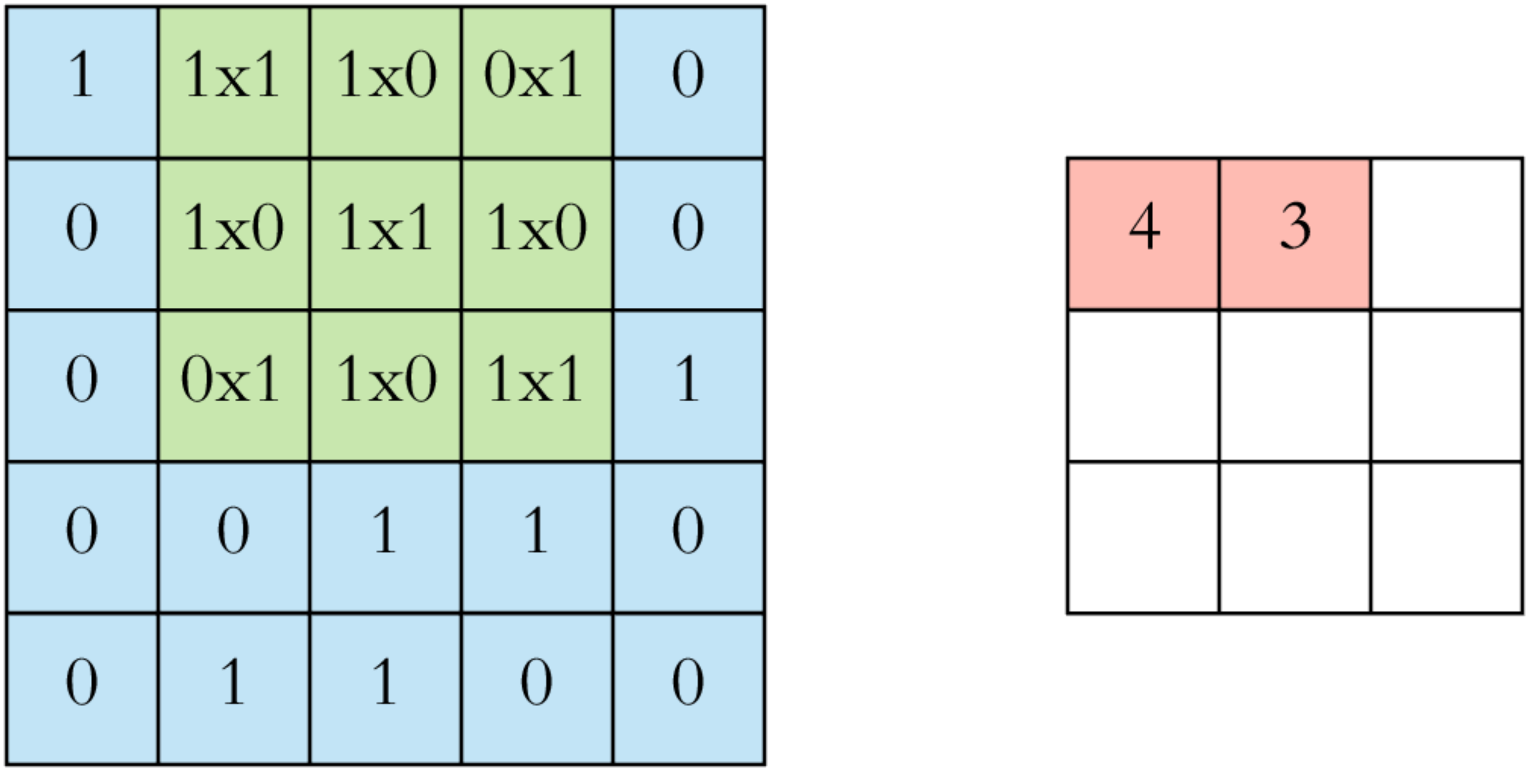
Input x Filter

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

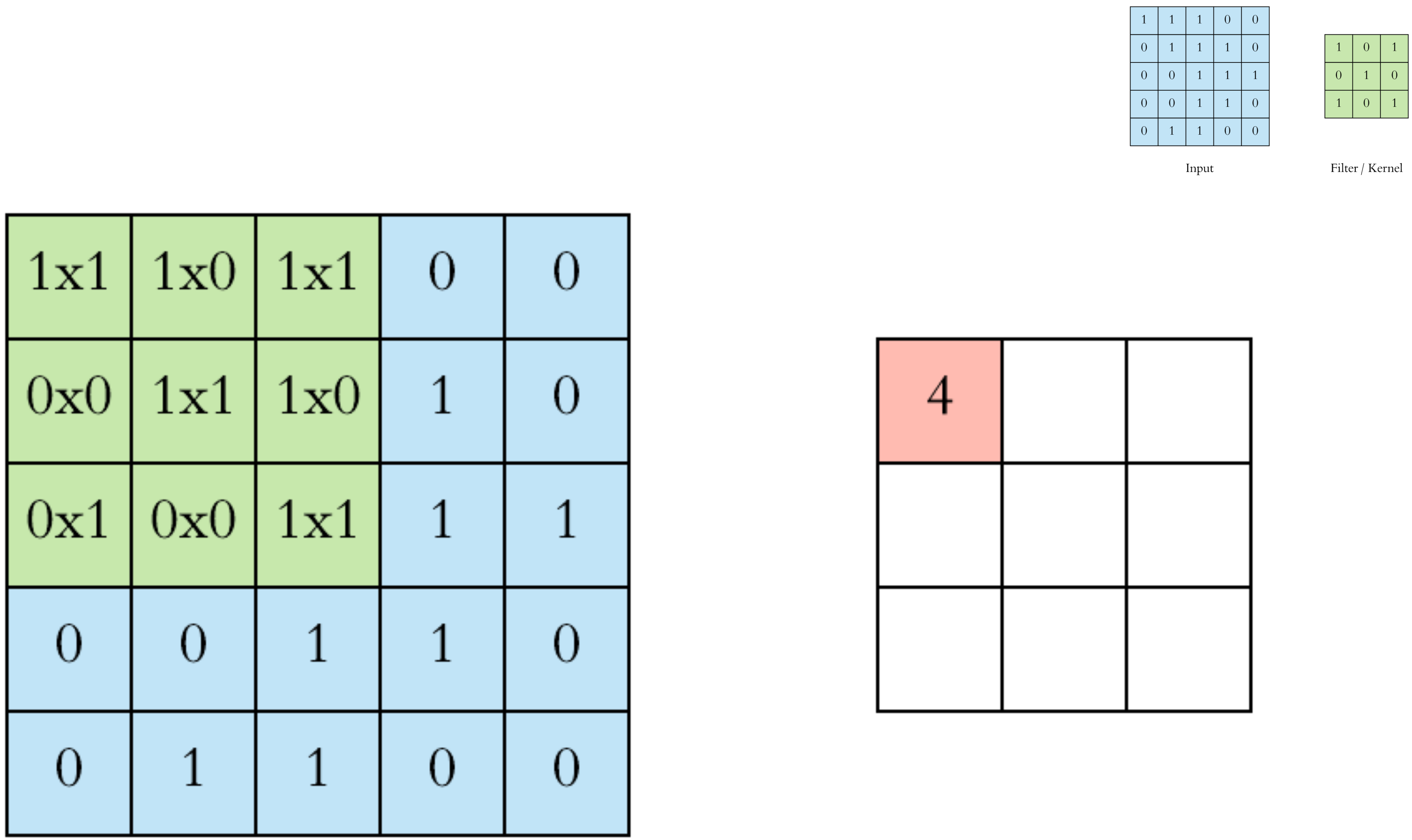


Input x Filter

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

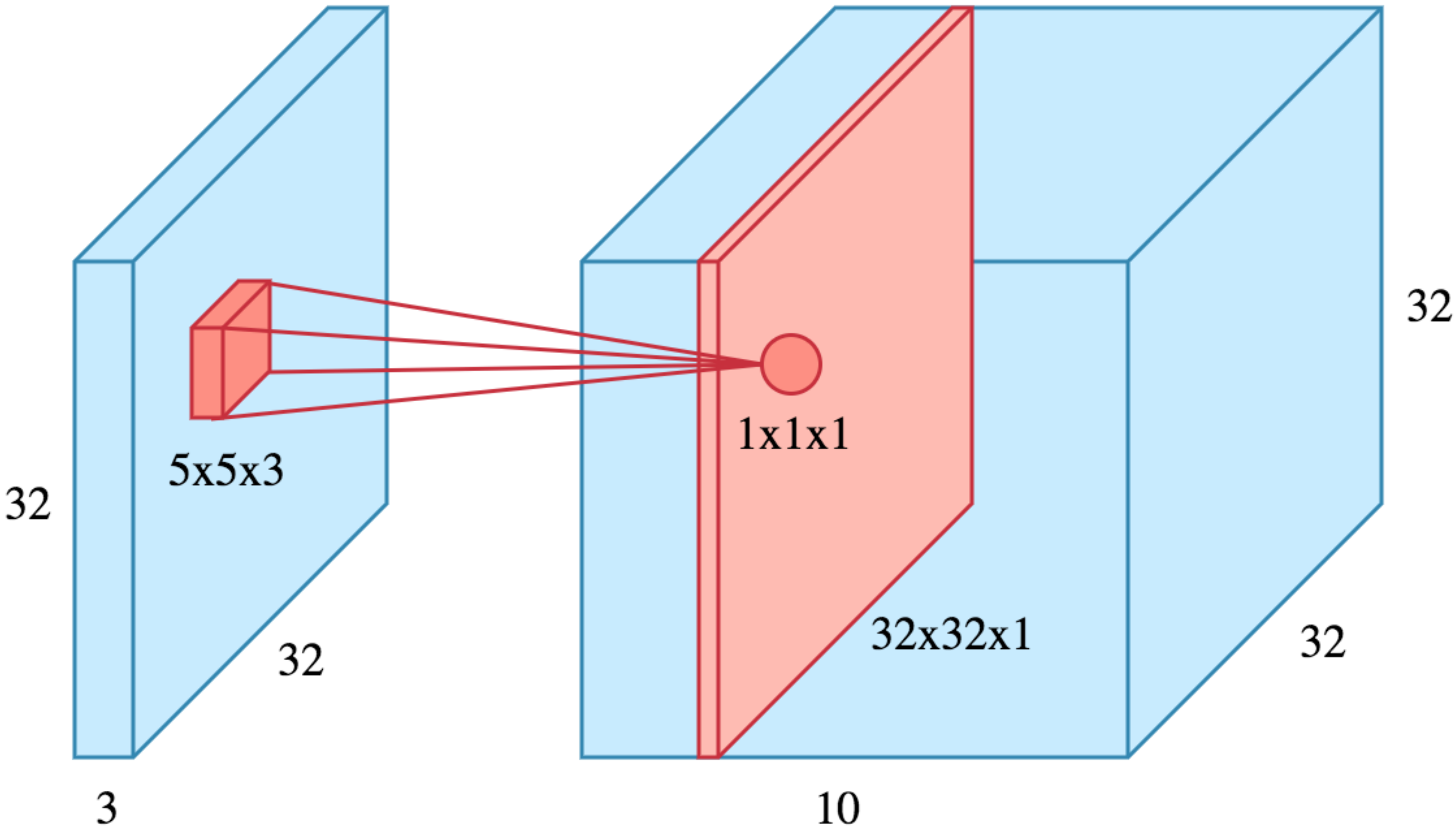


Example convolution operation shown in 2D using a 3x3 filter

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

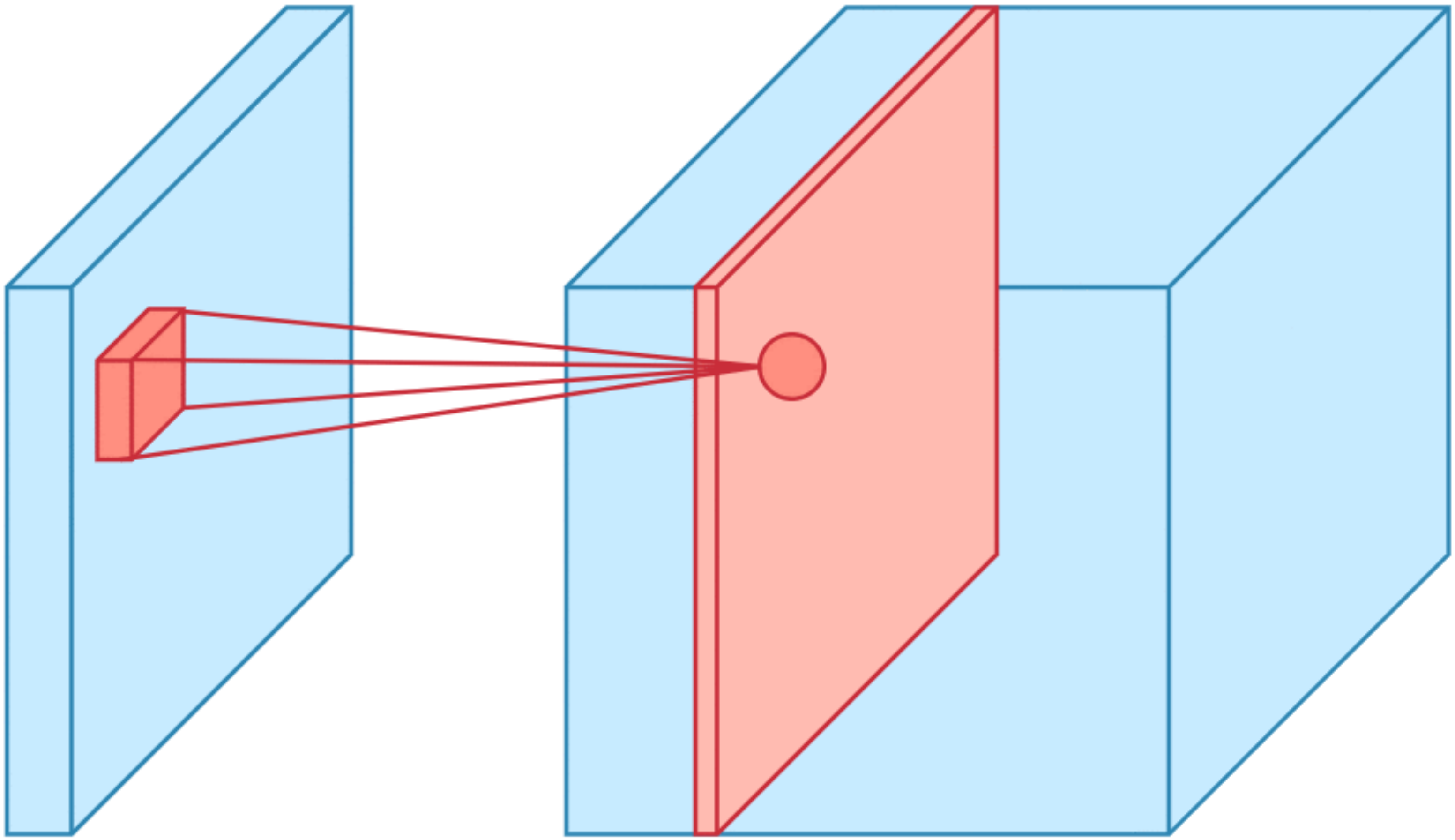
10 different filters 10 feature maps of size 32x32x1



final output of the convolution layer:
a volume of size 32x32x10

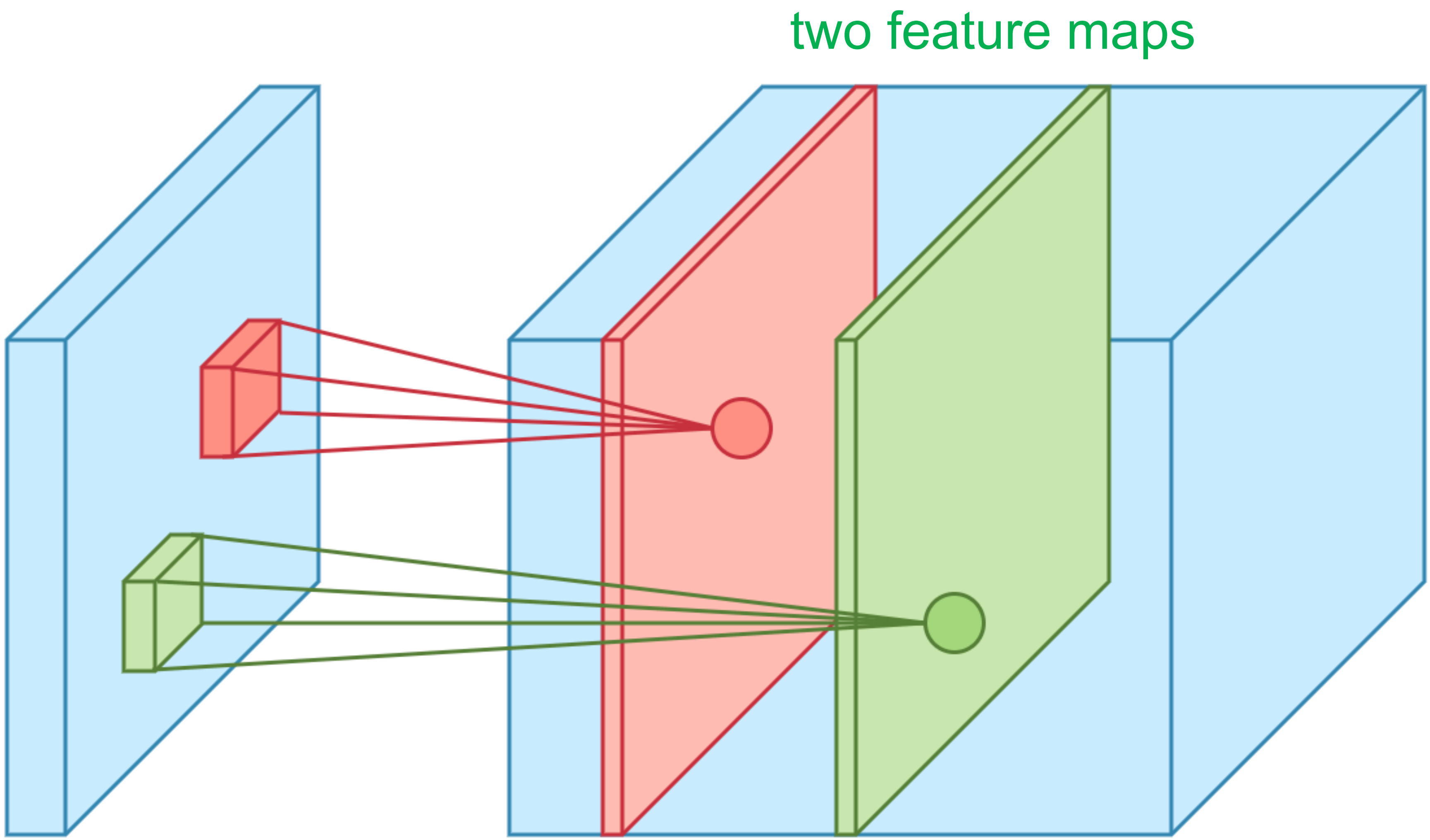
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Sliding operation at 4 locations



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

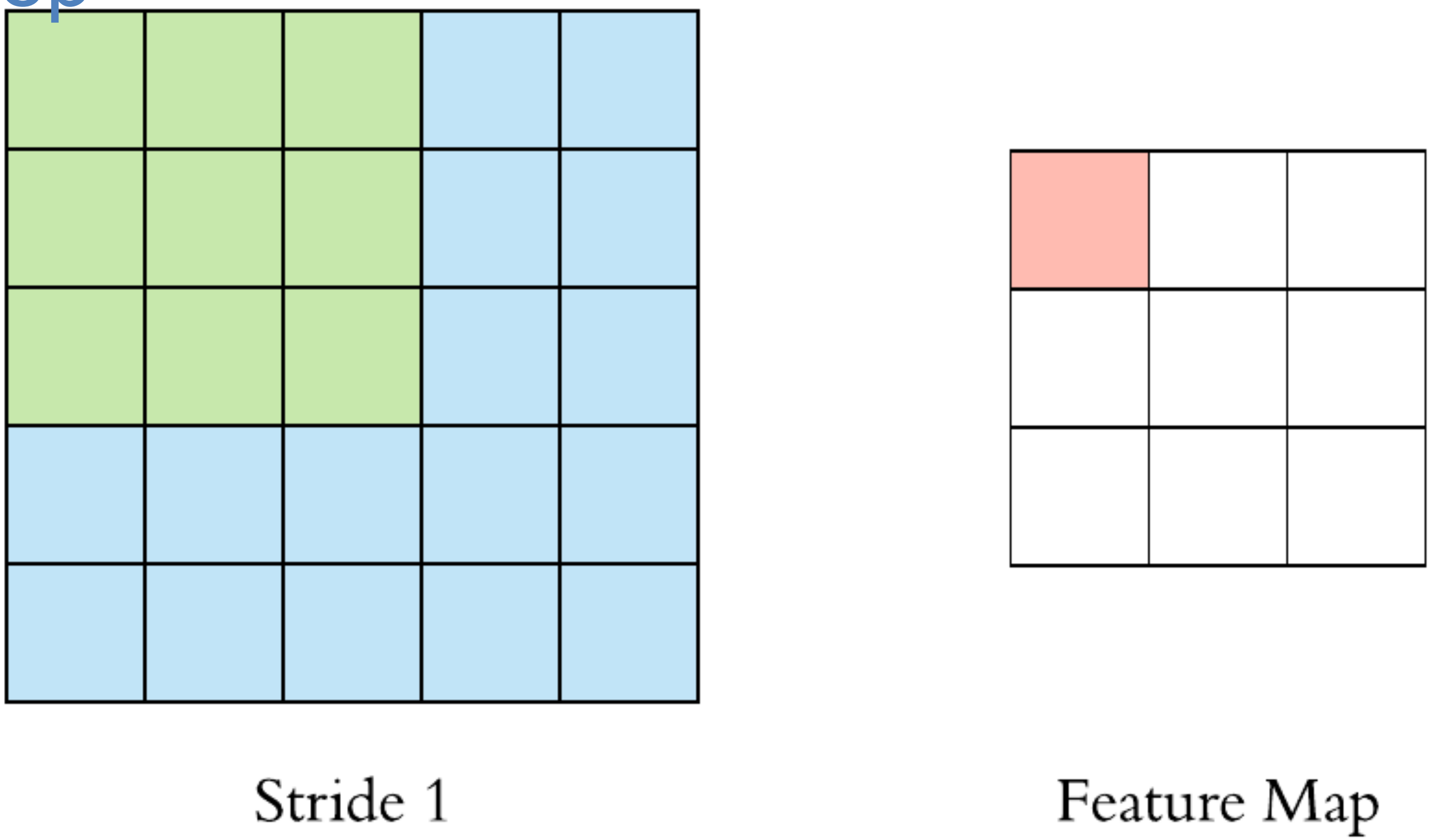
CNN Convolution Layer



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

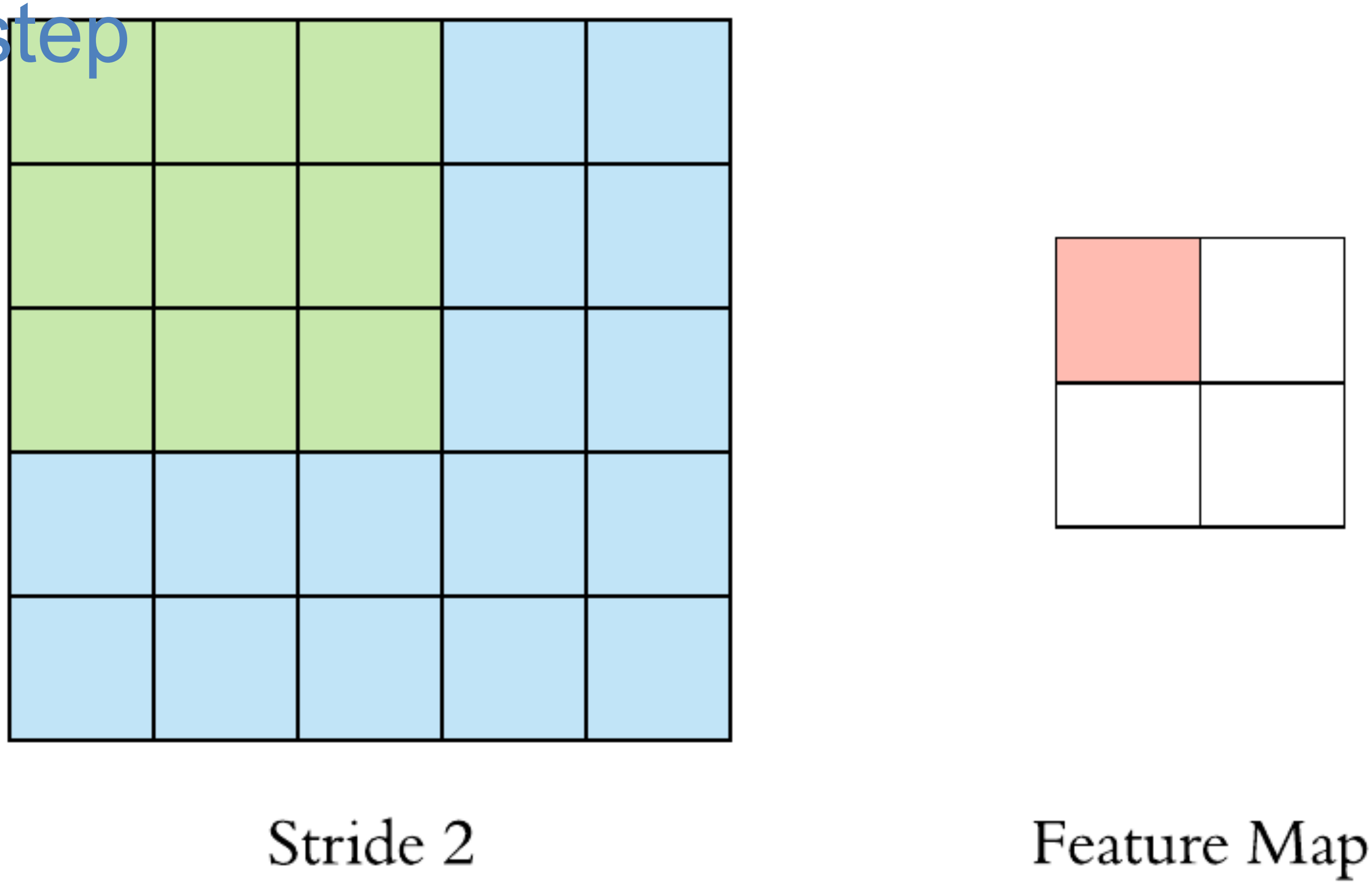
Stride specifies how much we move the convolution filter at each step



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

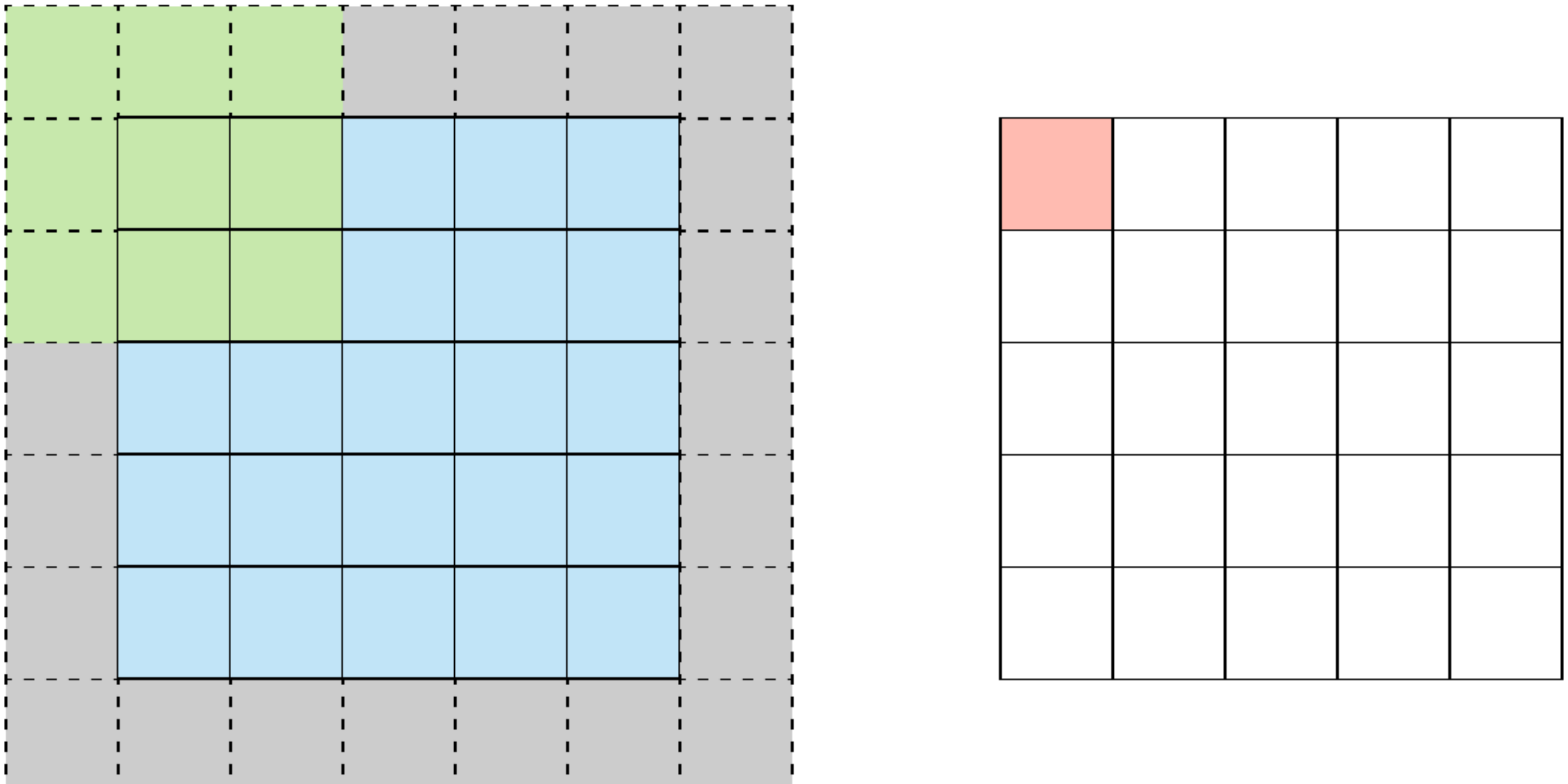
Stride specifies how much we move the convolution filter at each step



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Stride 1 with Padding



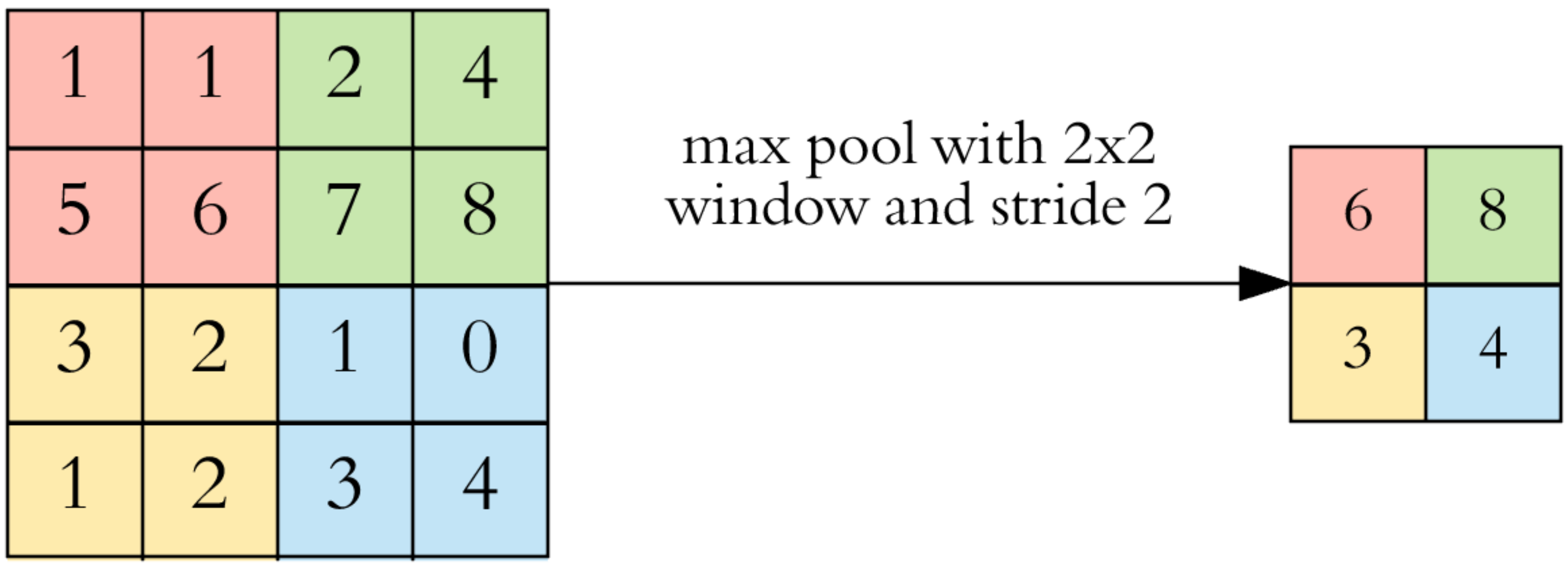
Stride 1 with Padding

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

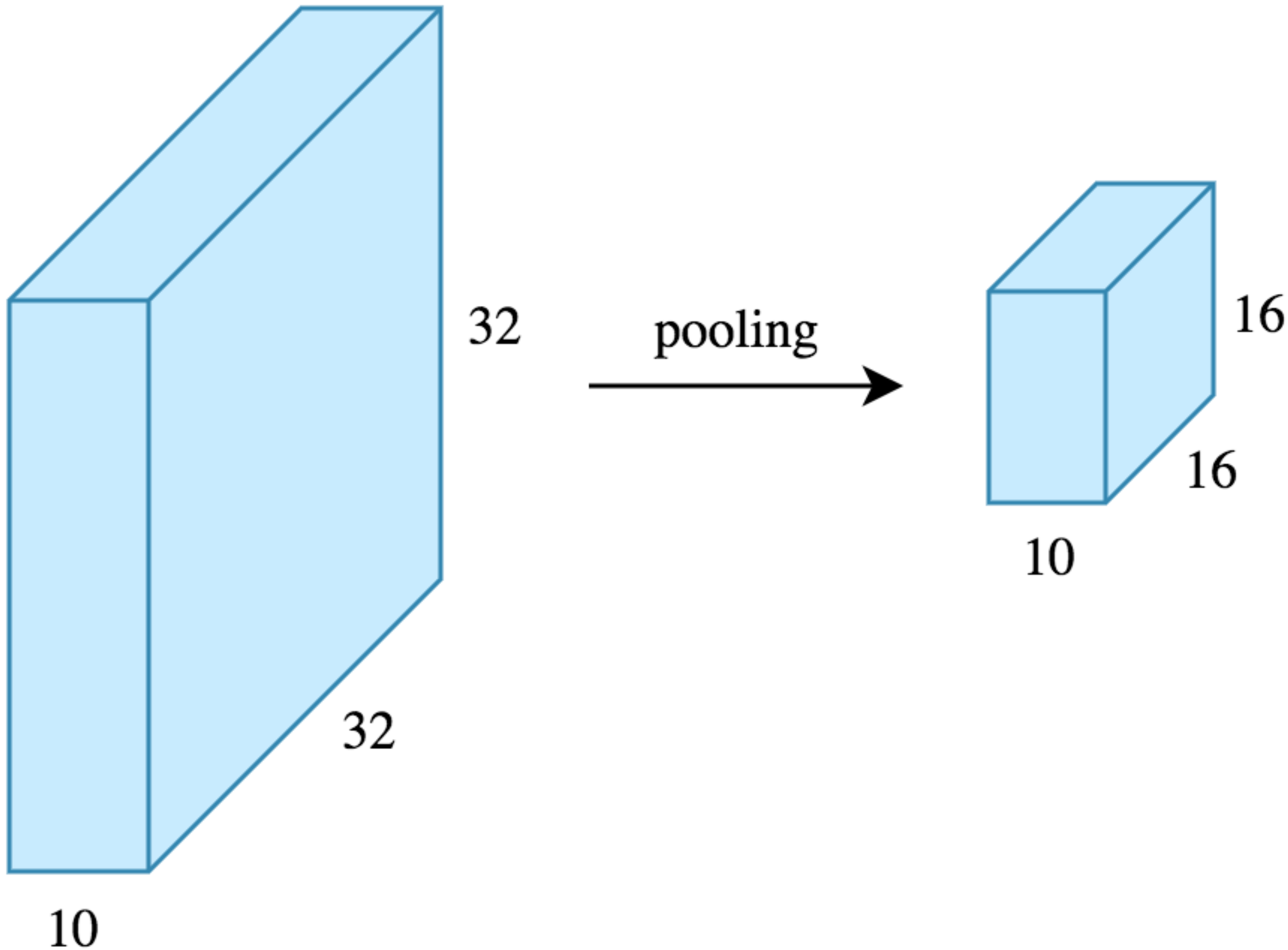
CNN Pooling Layer

Max Pooling



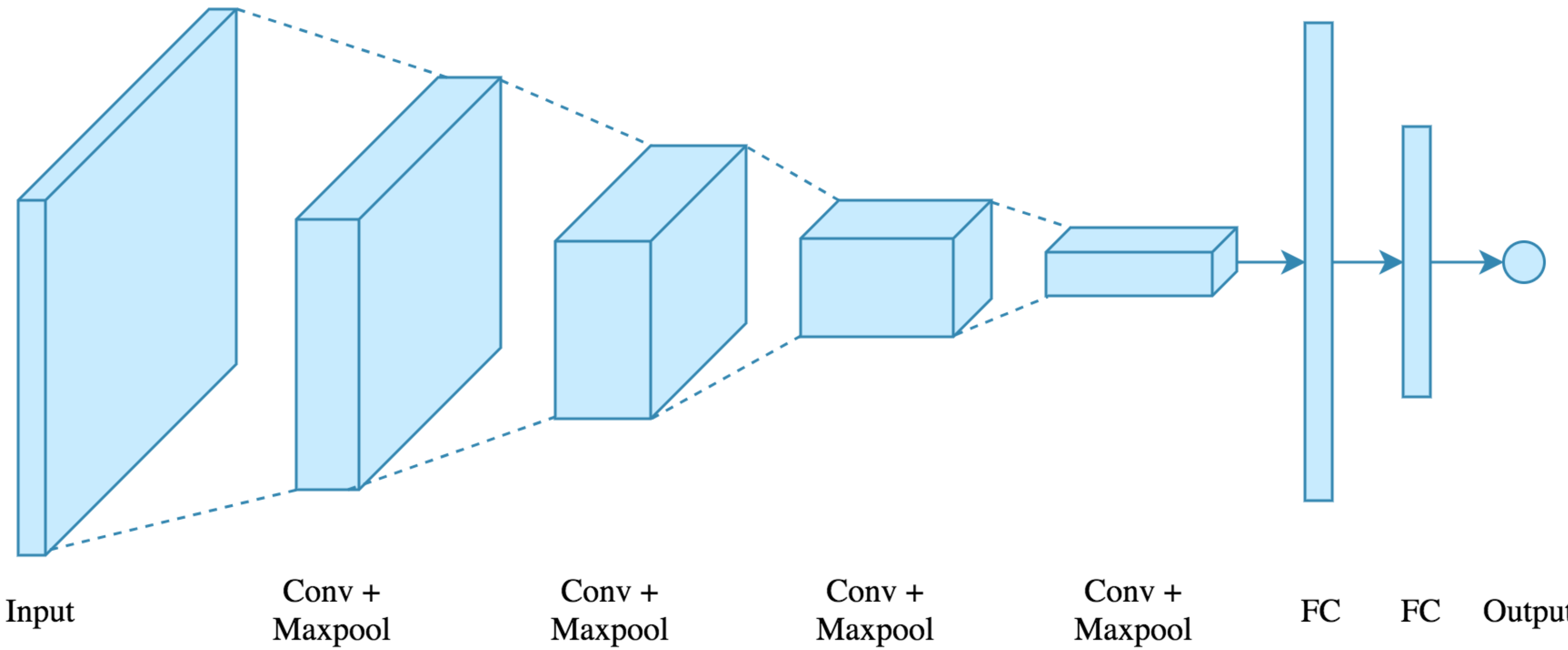
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Pooling Layer



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

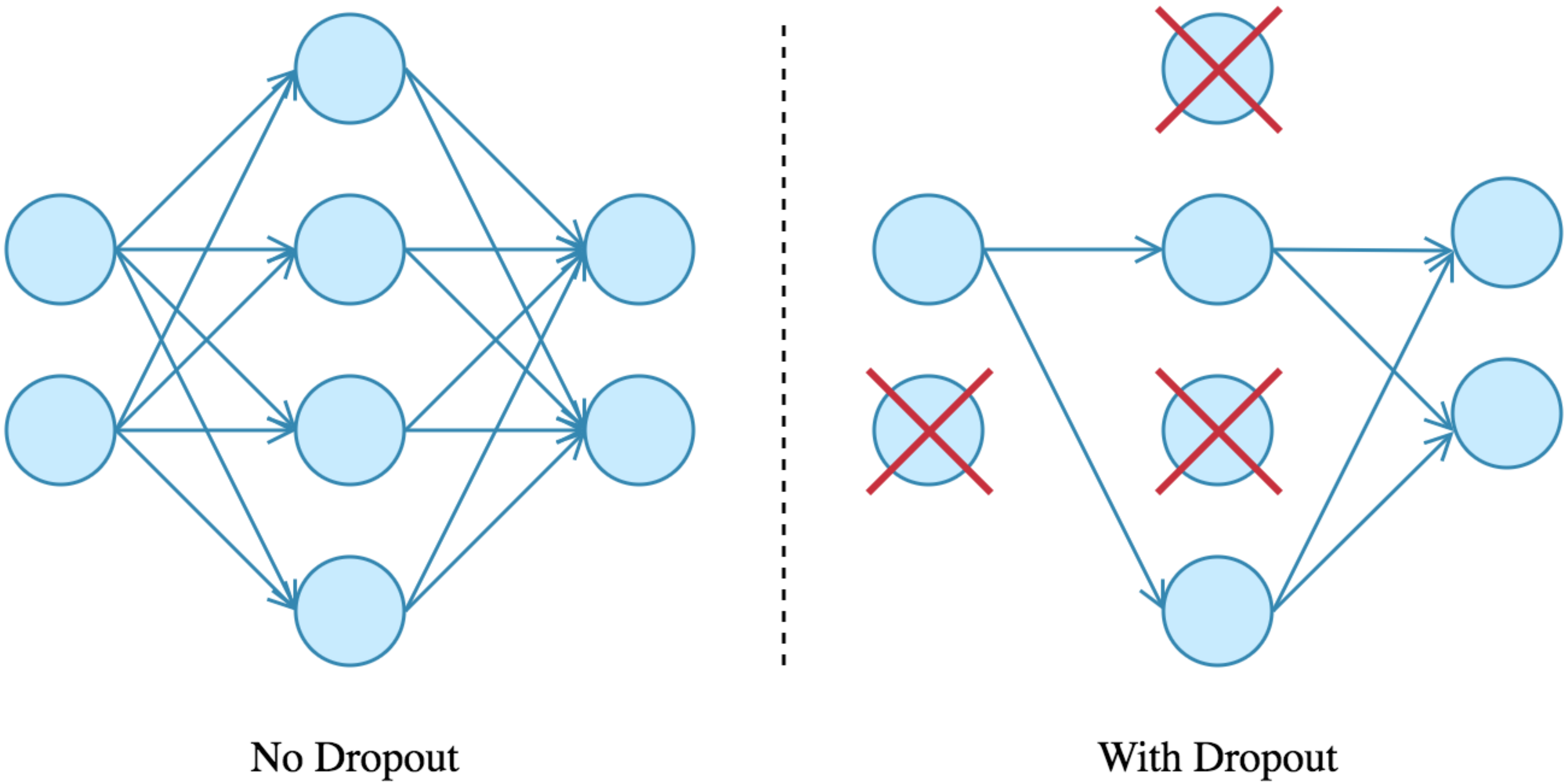
CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                 input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
```

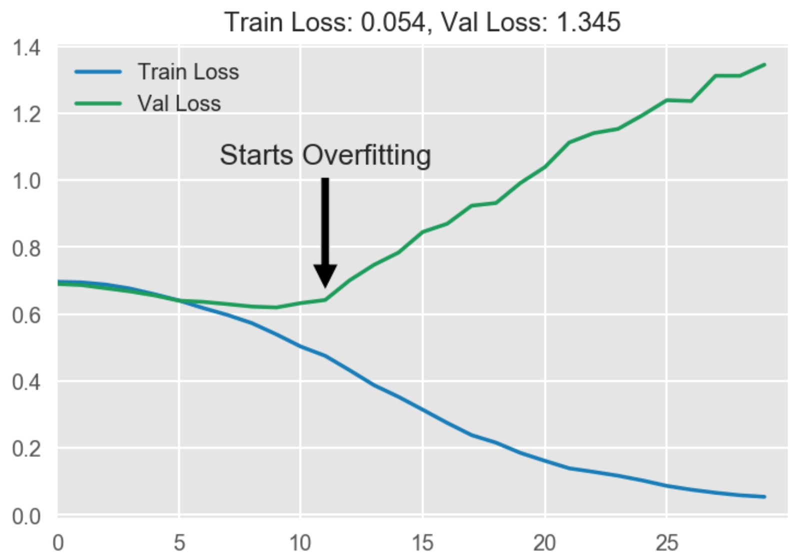
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

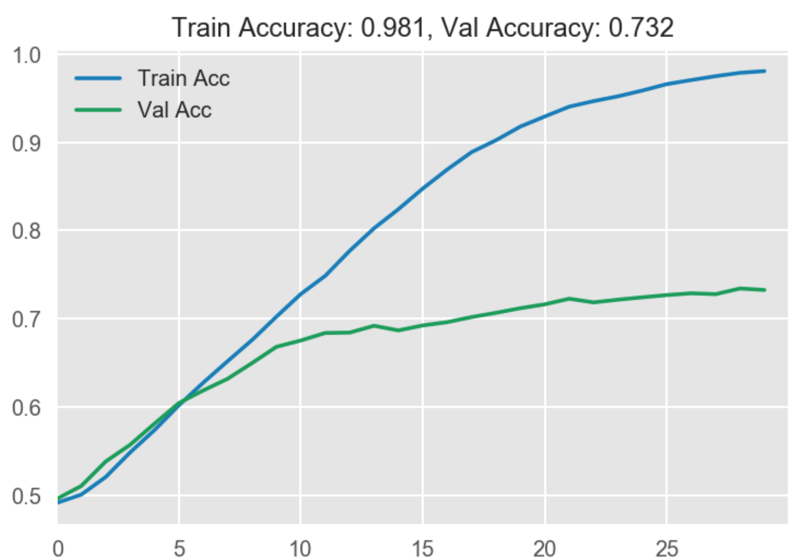
Dropout



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Model Performance

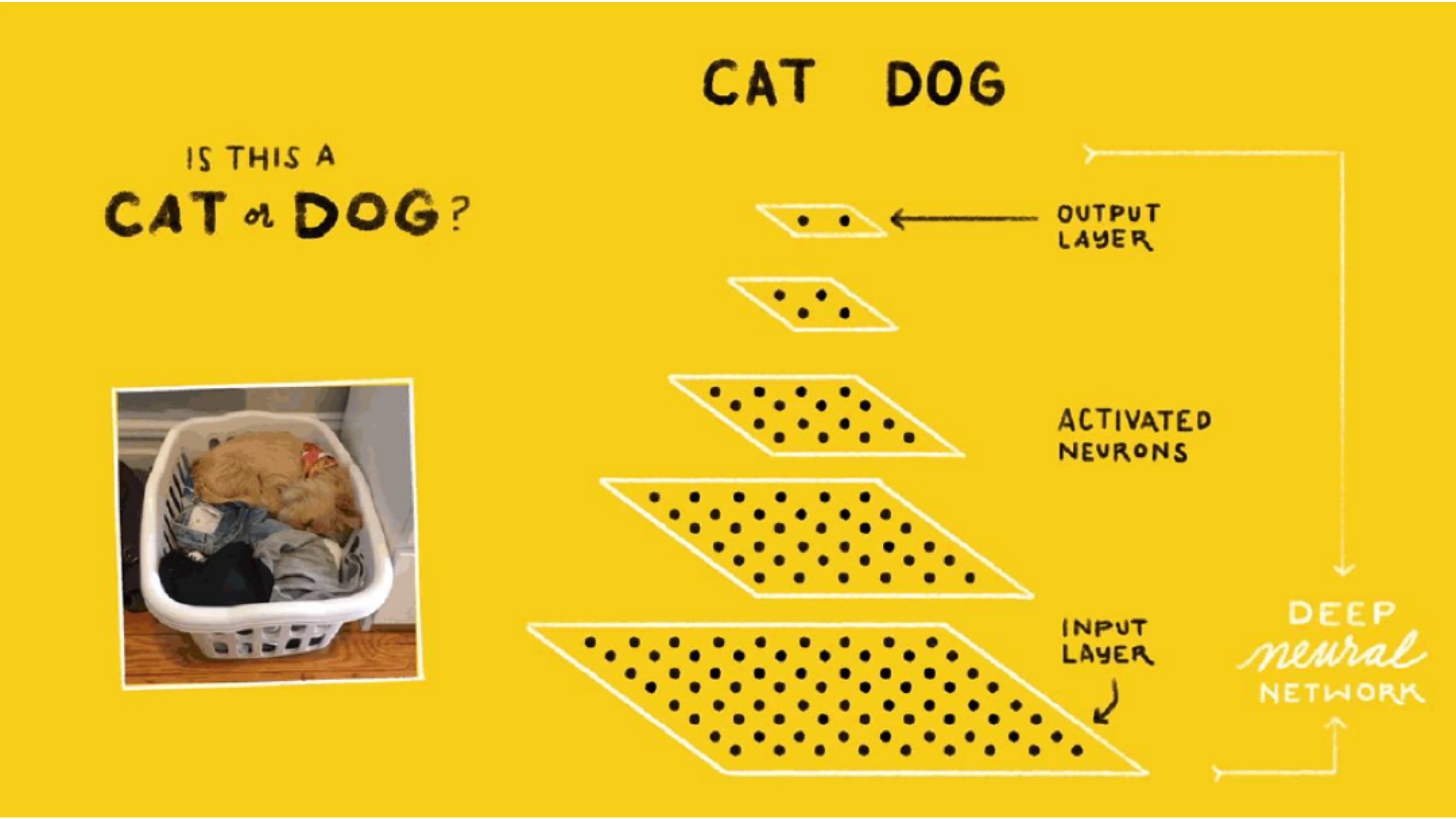




Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Visual Recognition

Image Classification



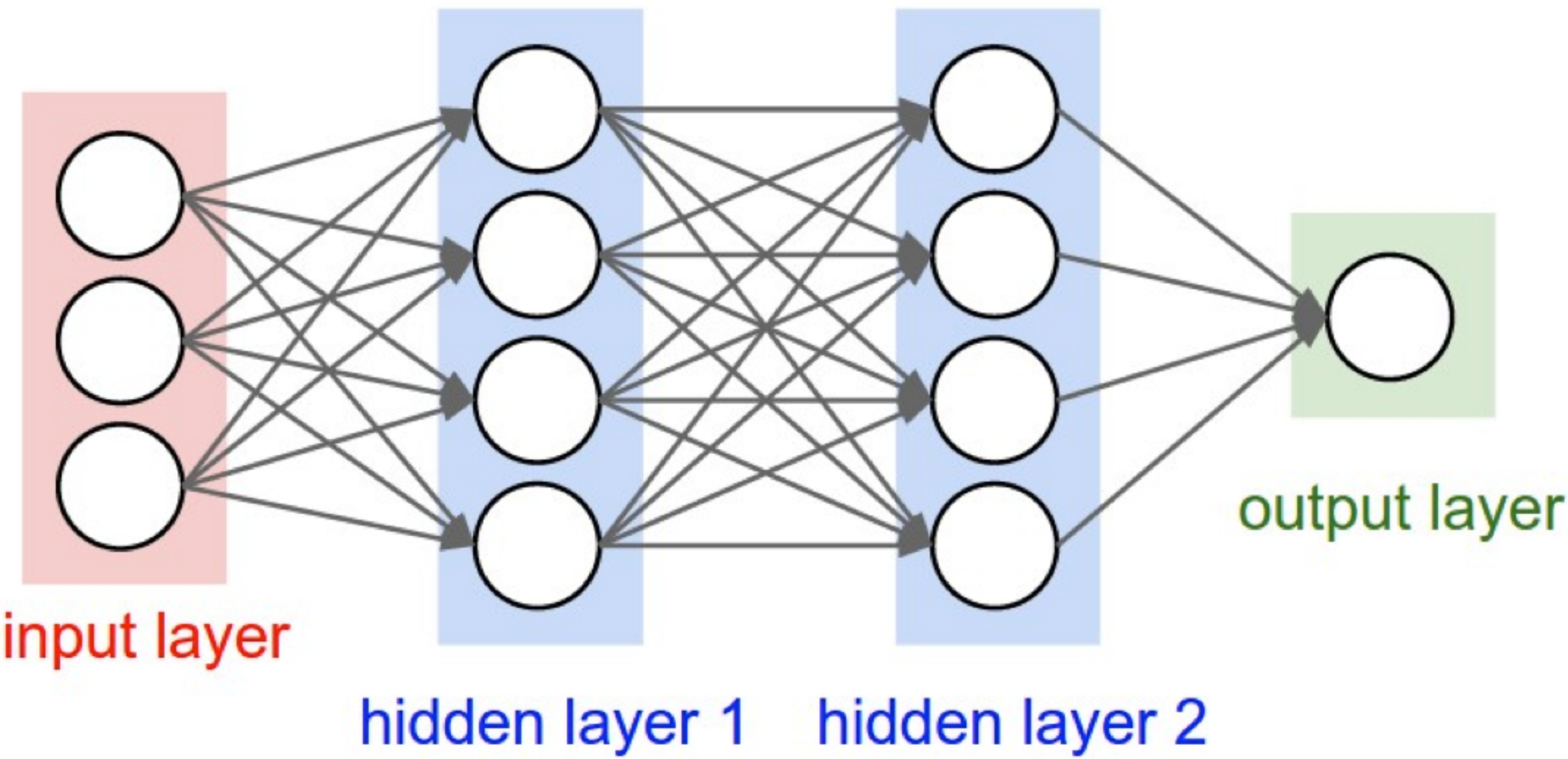
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Convolutional Neural Networks

(CNNs / ConvNets)

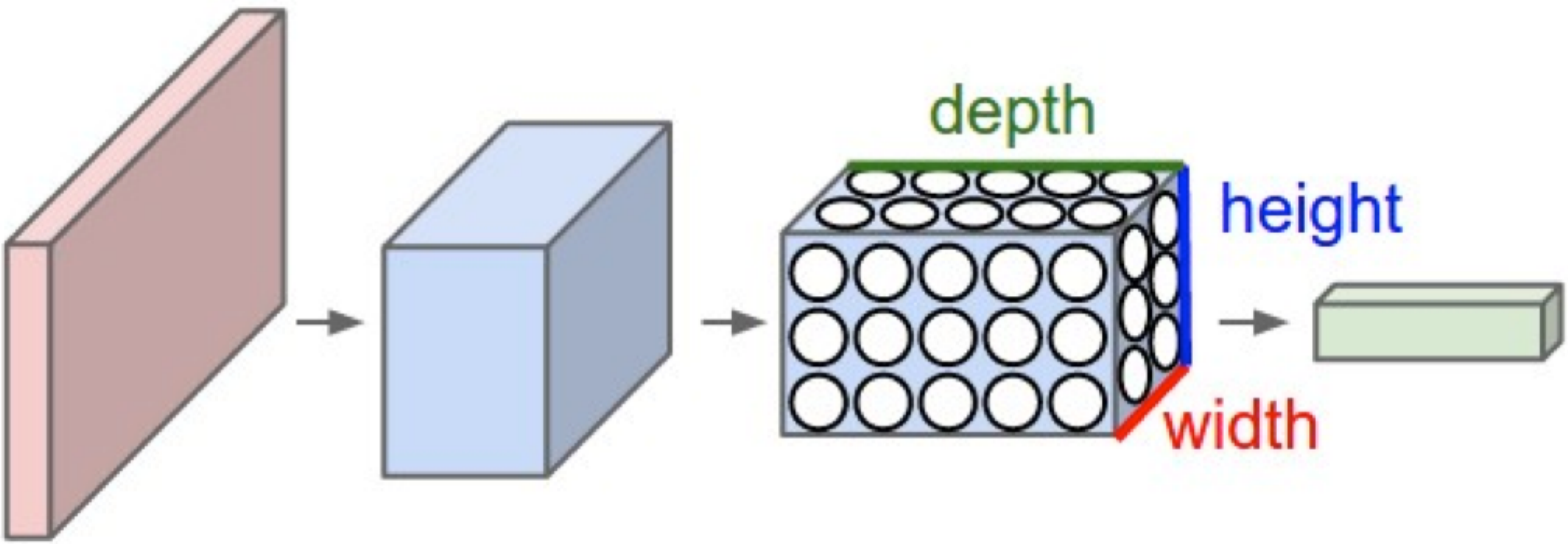
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



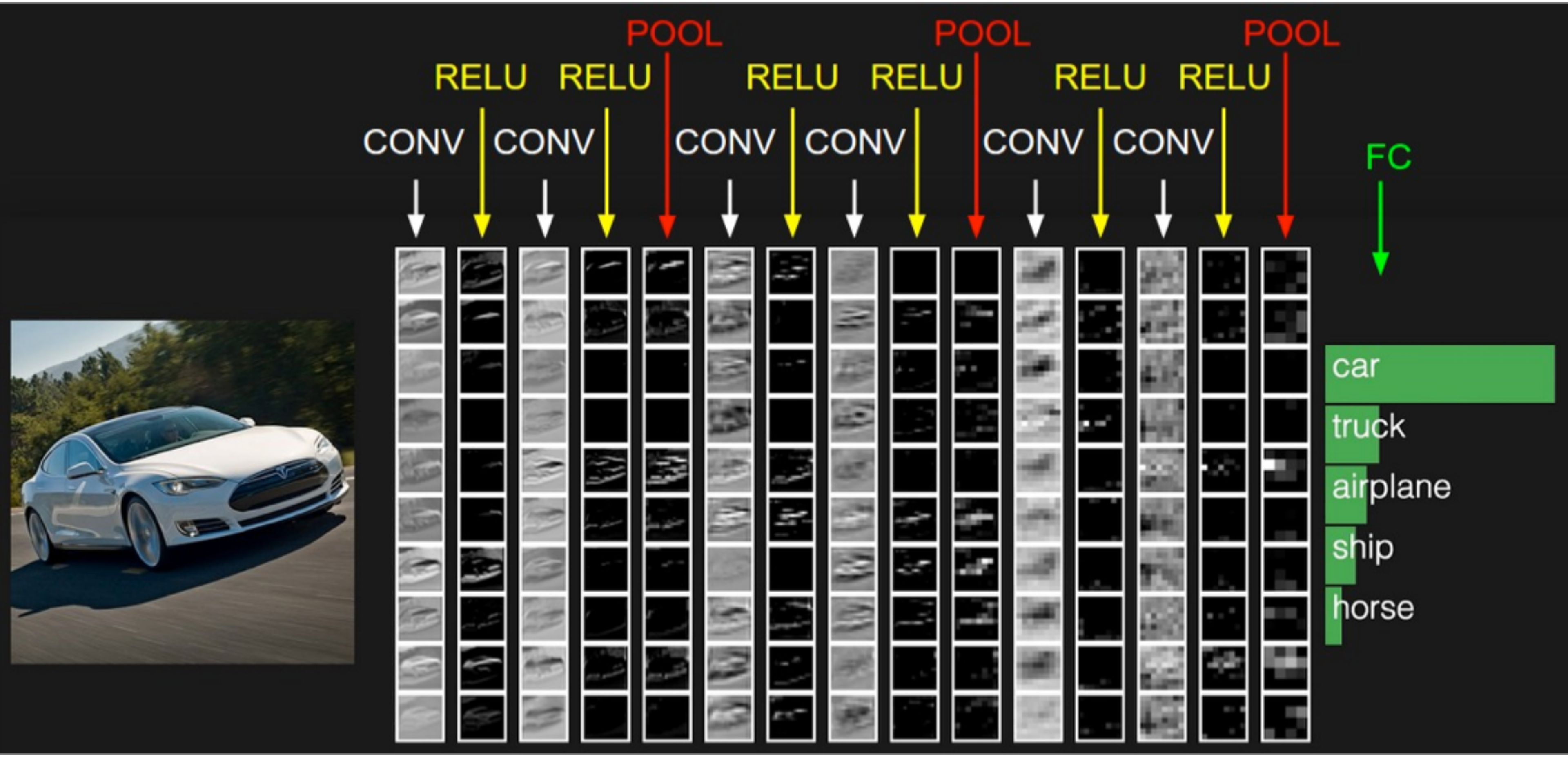
http://cs231n.github.io/convolutional-networks/

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



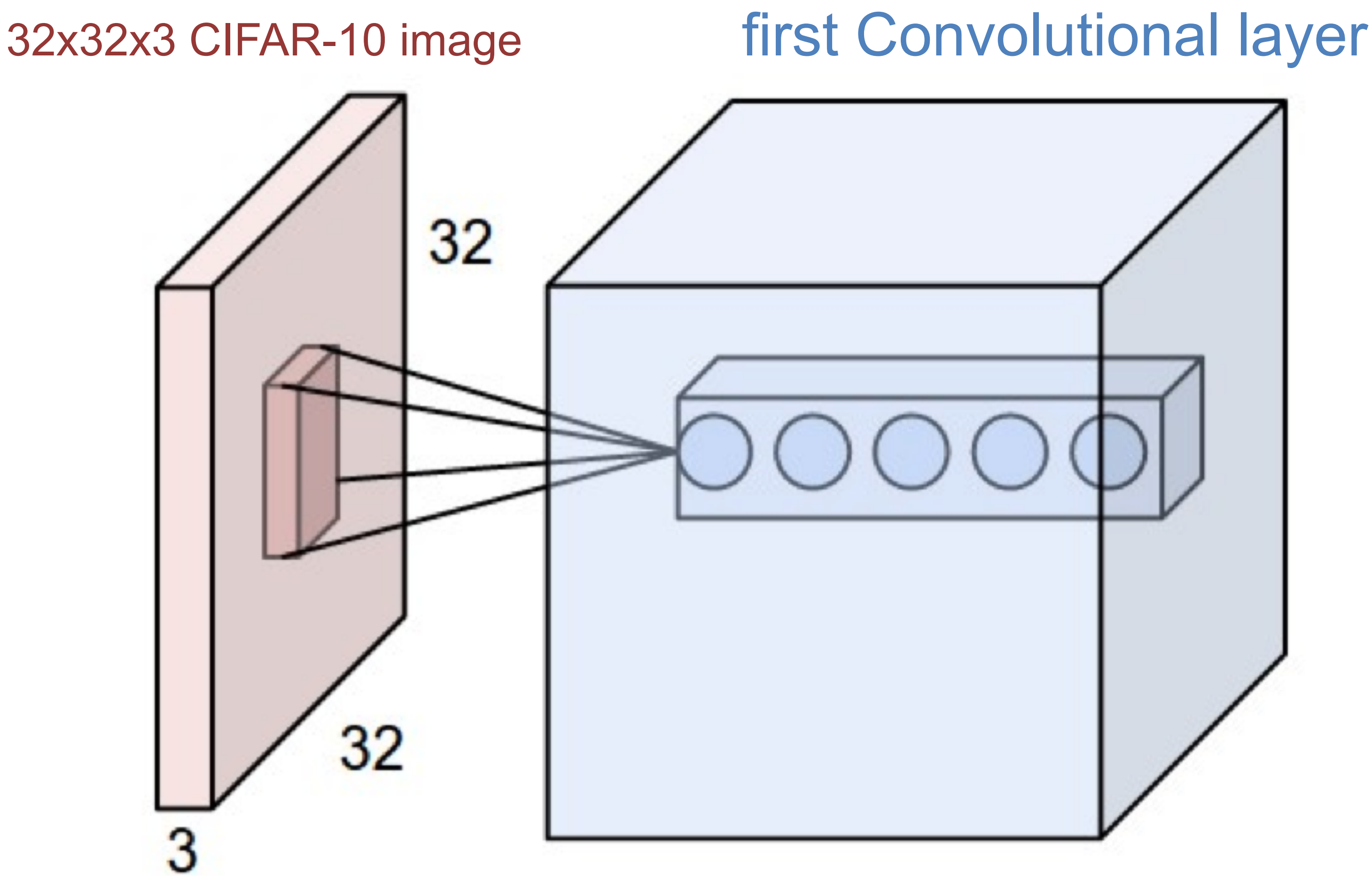
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



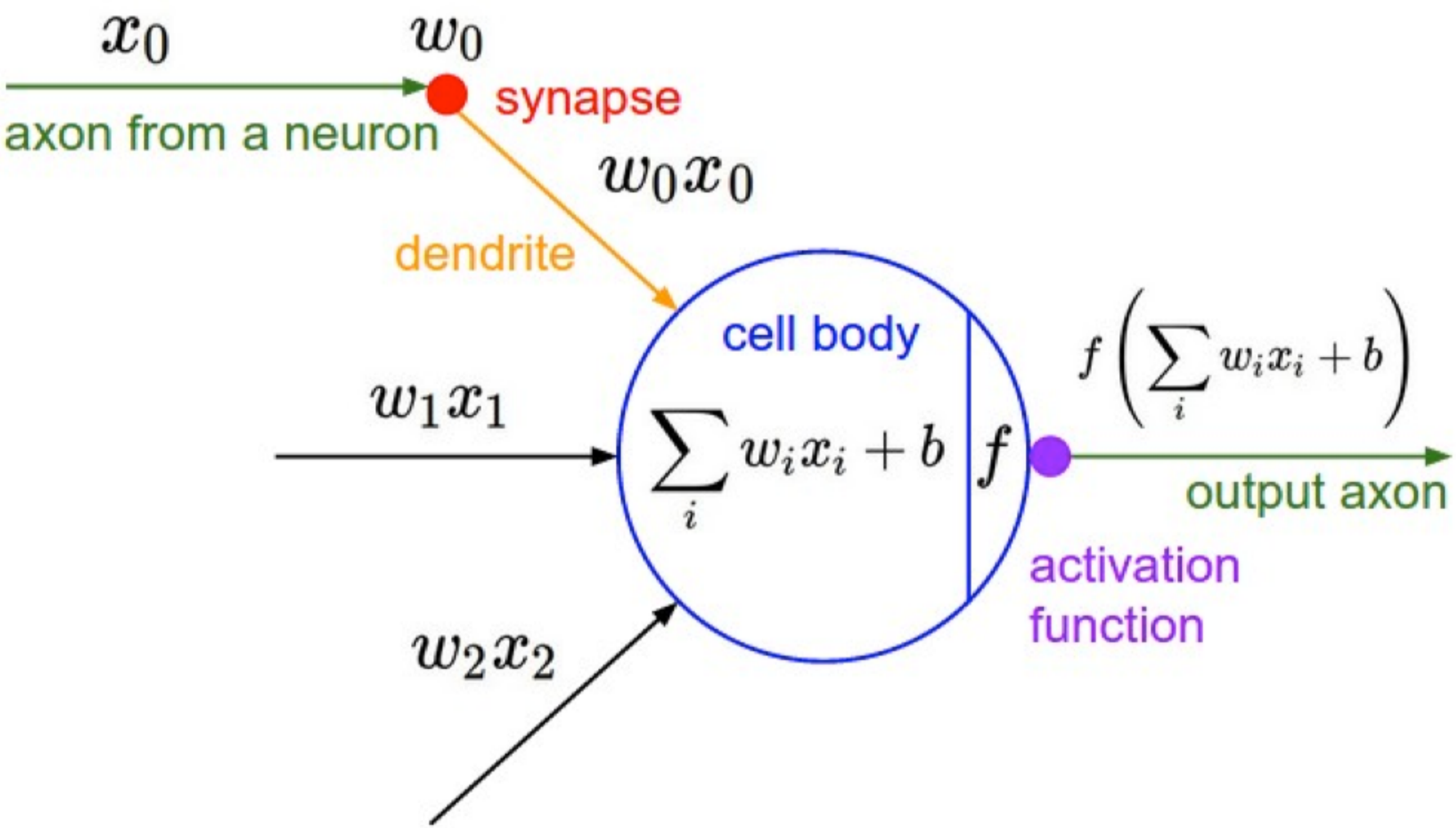
http://cs231n.github.io/convolutional-networks/

ConvNets



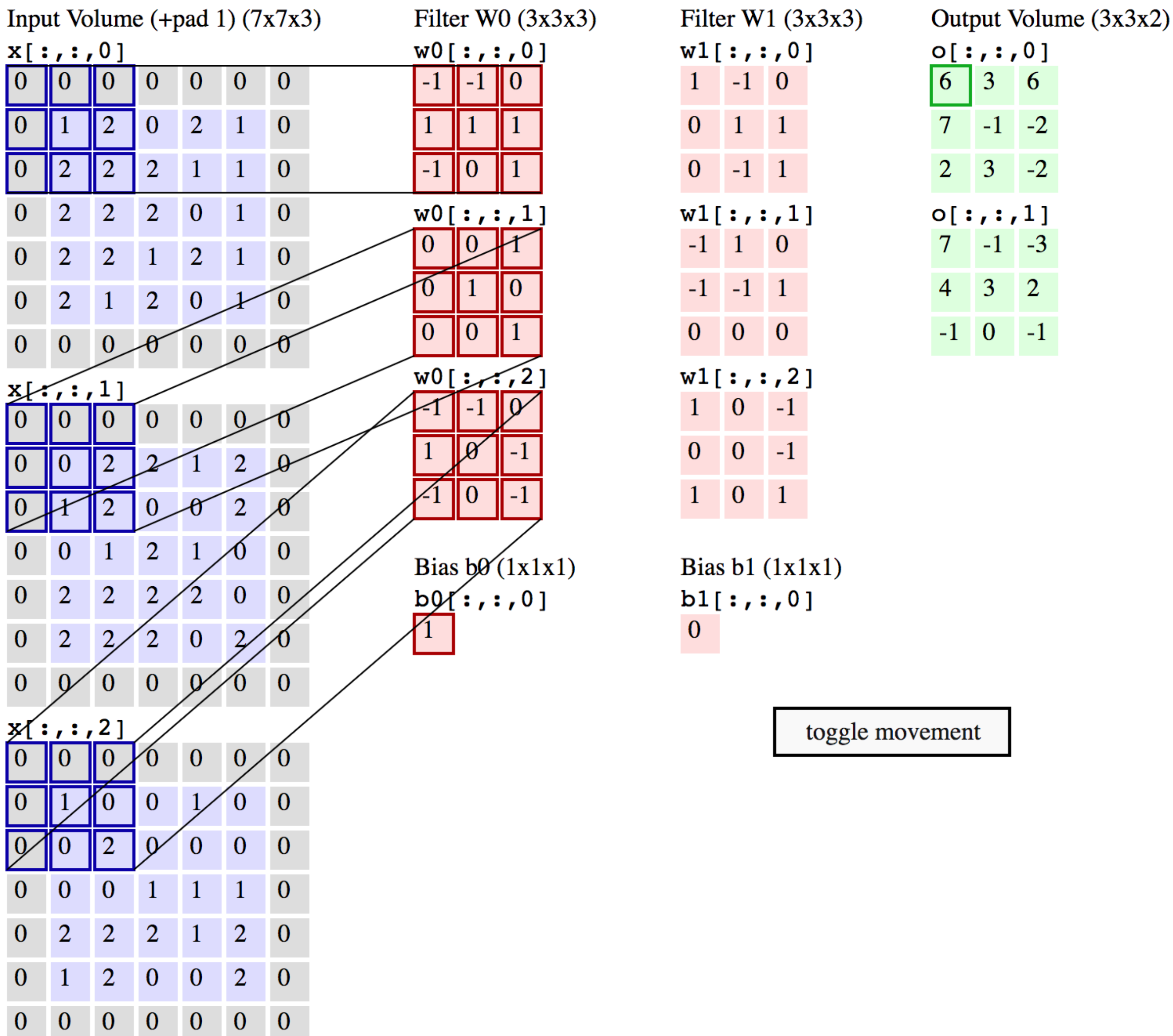
http://cs231n.github.io/convolutional-networks/

ConvNets



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

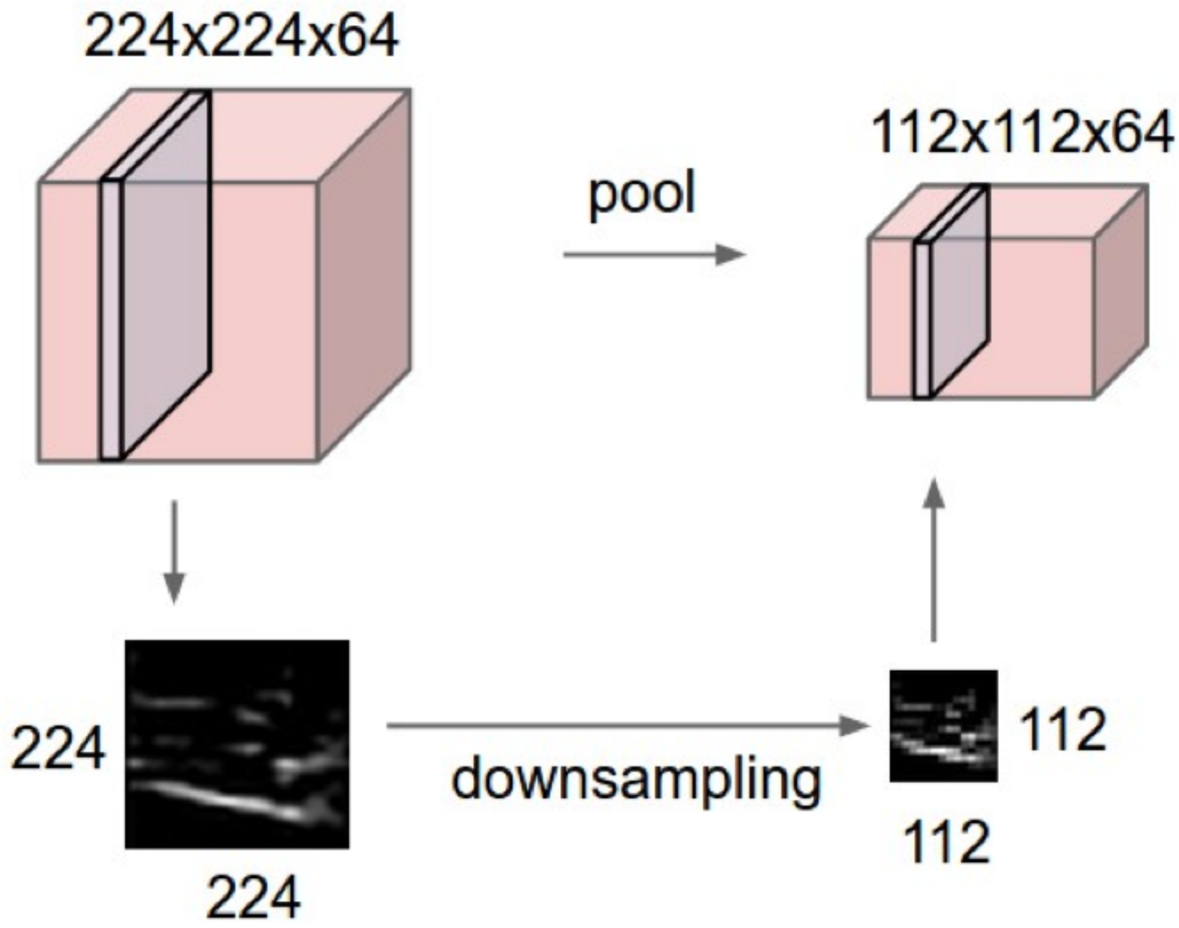
Convolution Demo



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

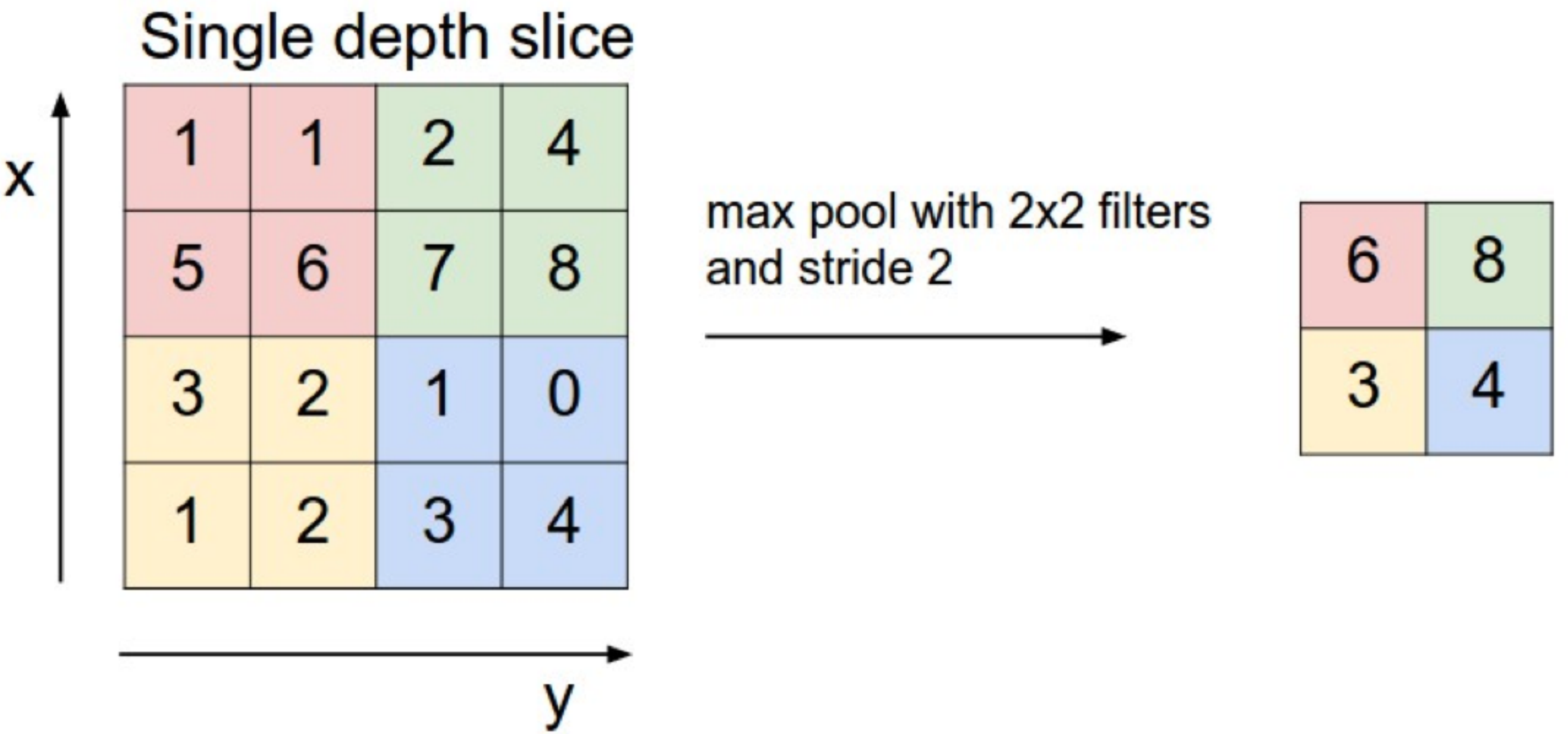
ConvNets

input volume of size [224x224x64]
is pooled with **filter** size 2, **stride** 2
into output volume of size [112x112x64]



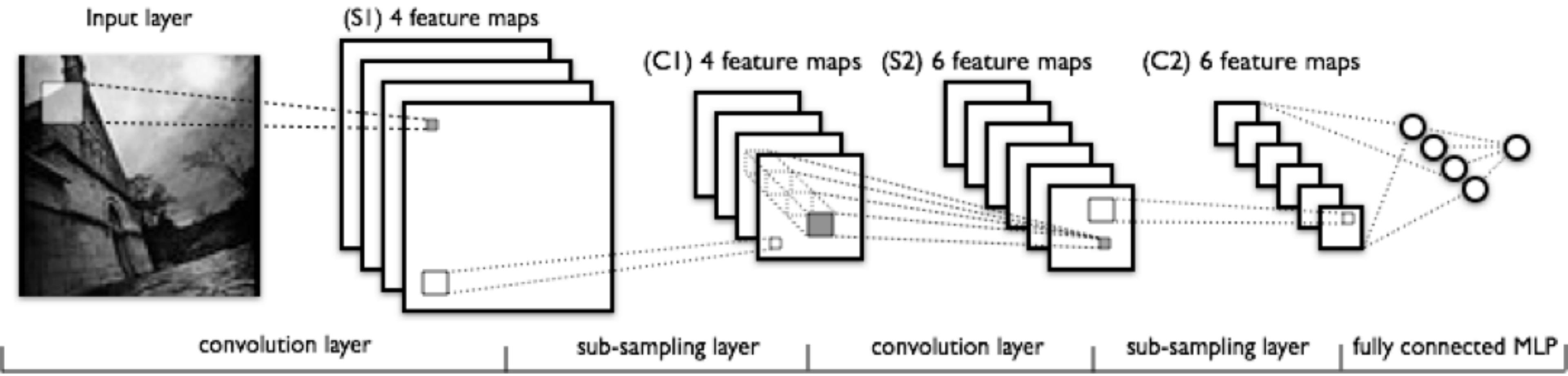
http://cs231n.github.io/convolutional-networks/

ConvNets
max pooling



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

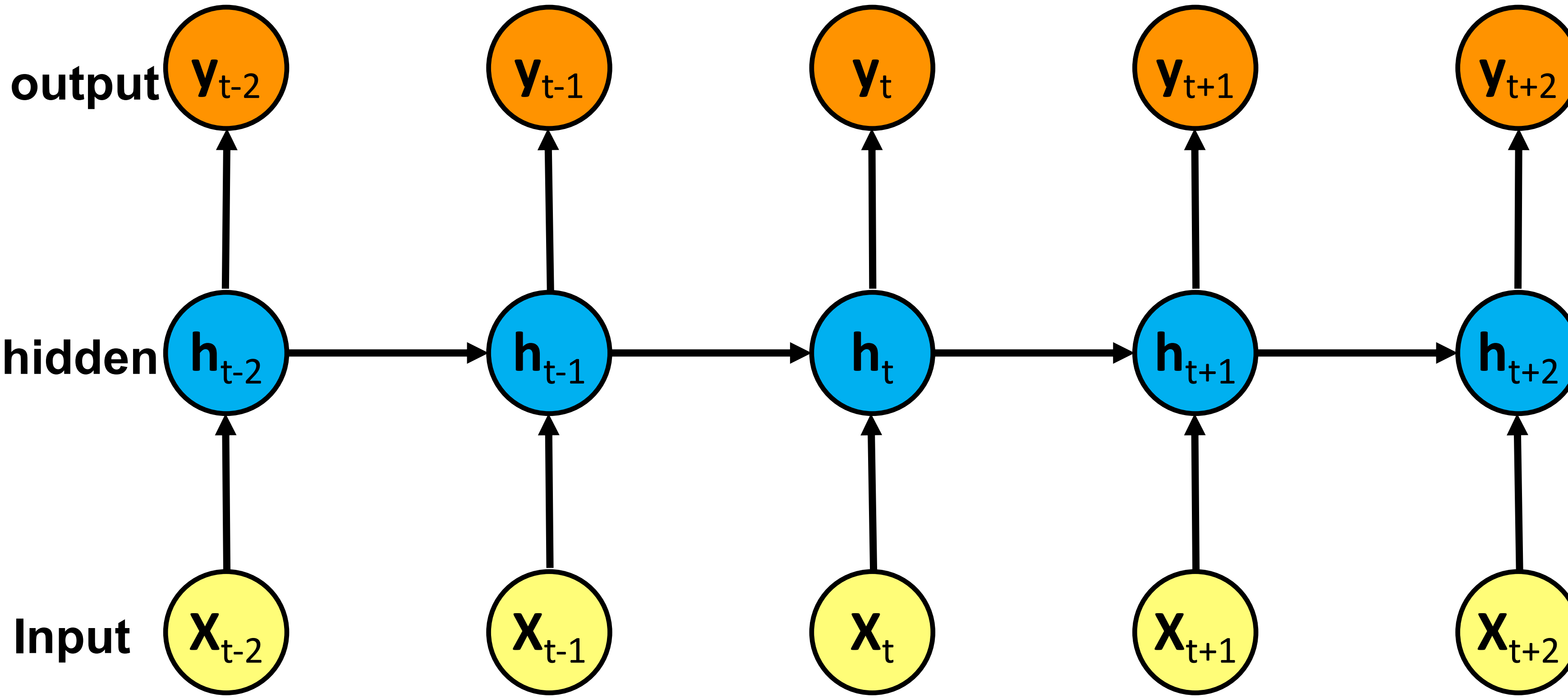
Convolutional Neural Networks (CNN) (LeNet)



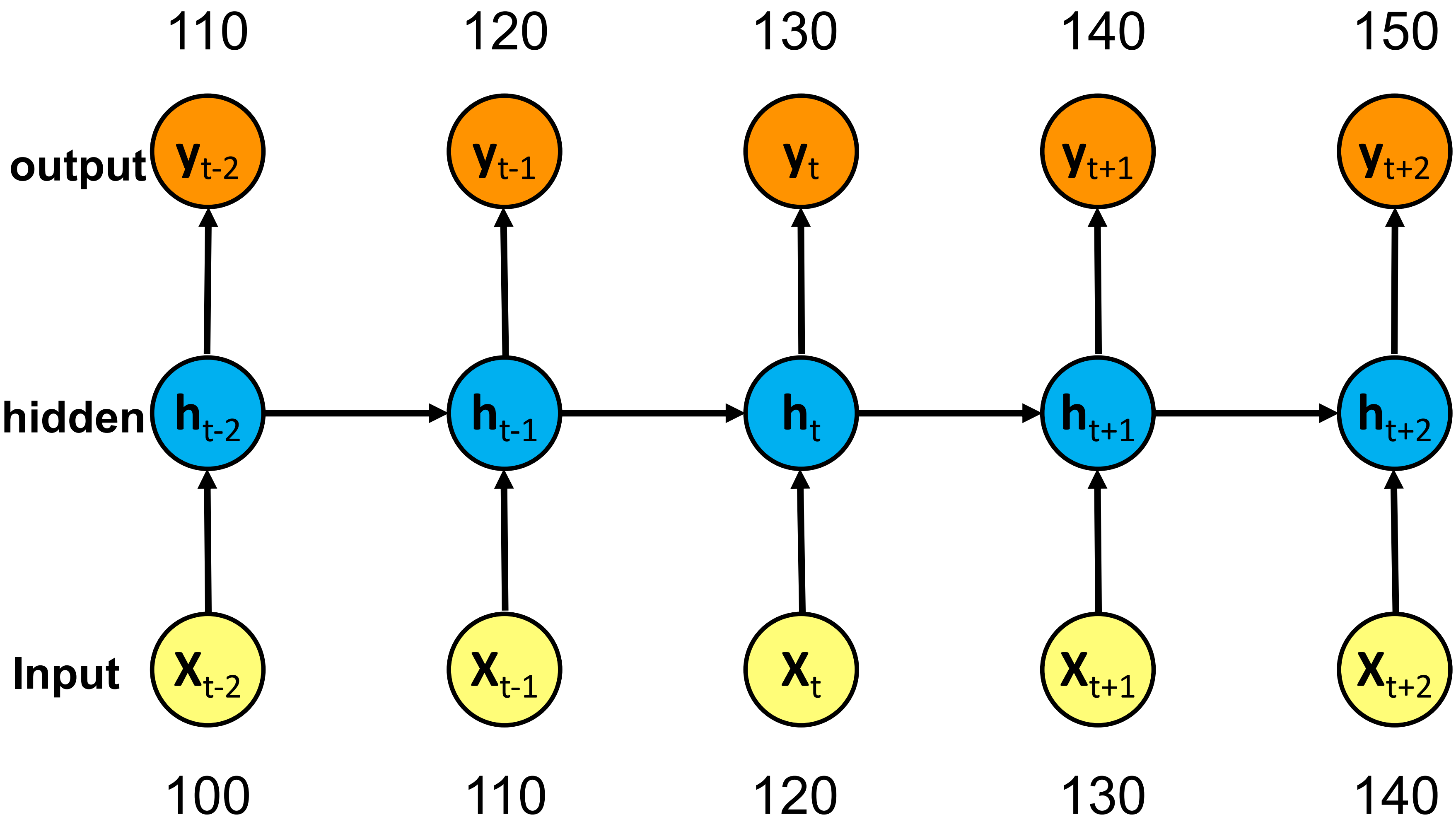
Source: http://deeplearning.net/tutorial/lenet.html

Recurrent Neural Networks (RNN)

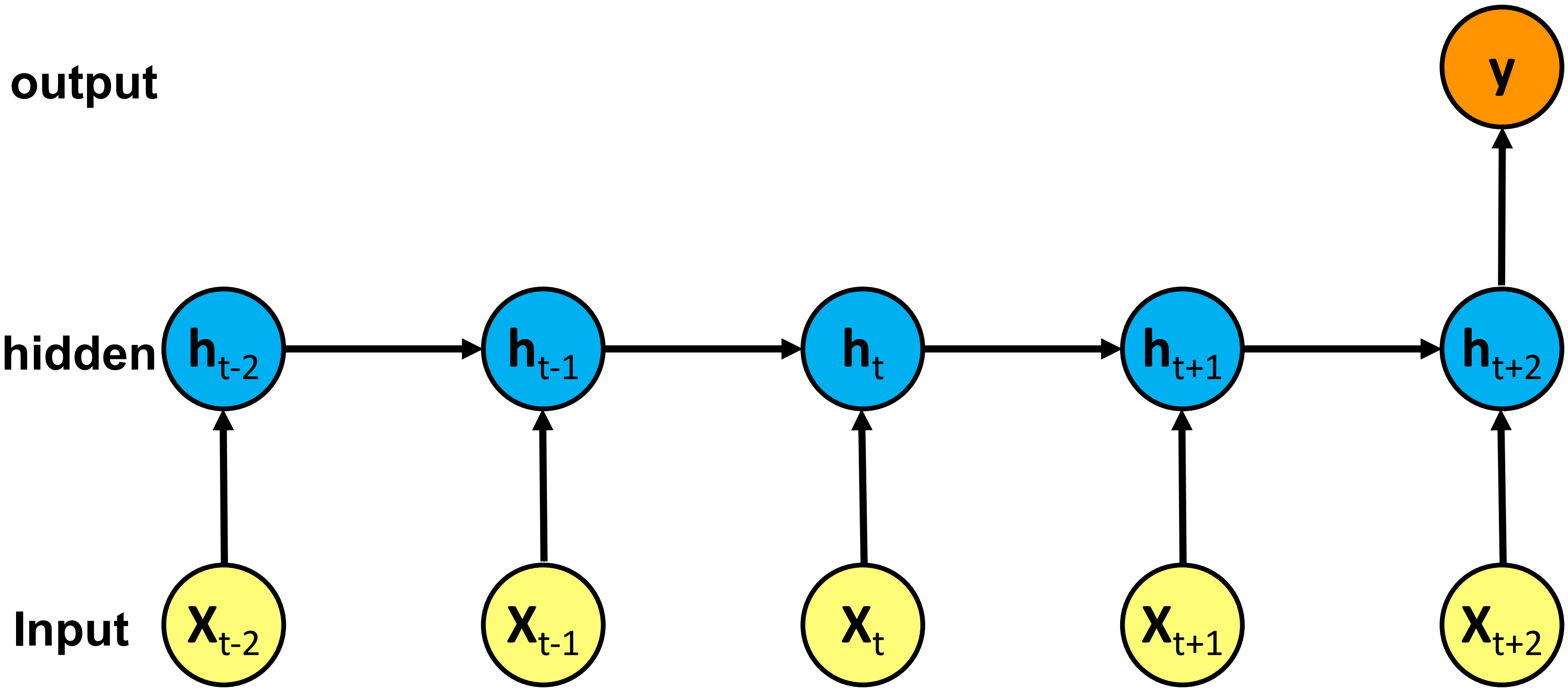
Recurrent Neural Networks (RNN)



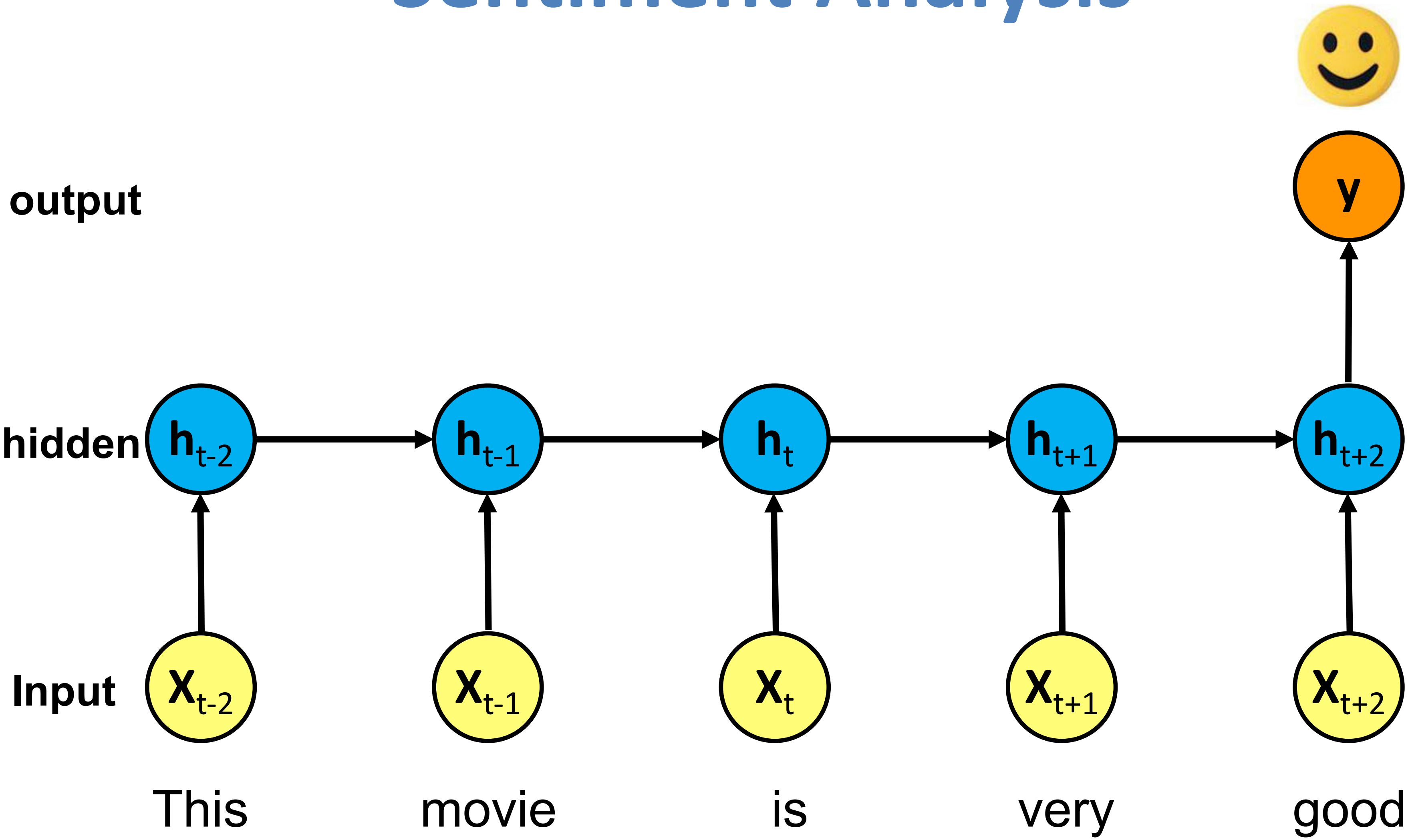
Recurrent Neural Networks (RNN)
Time Series Forecasting



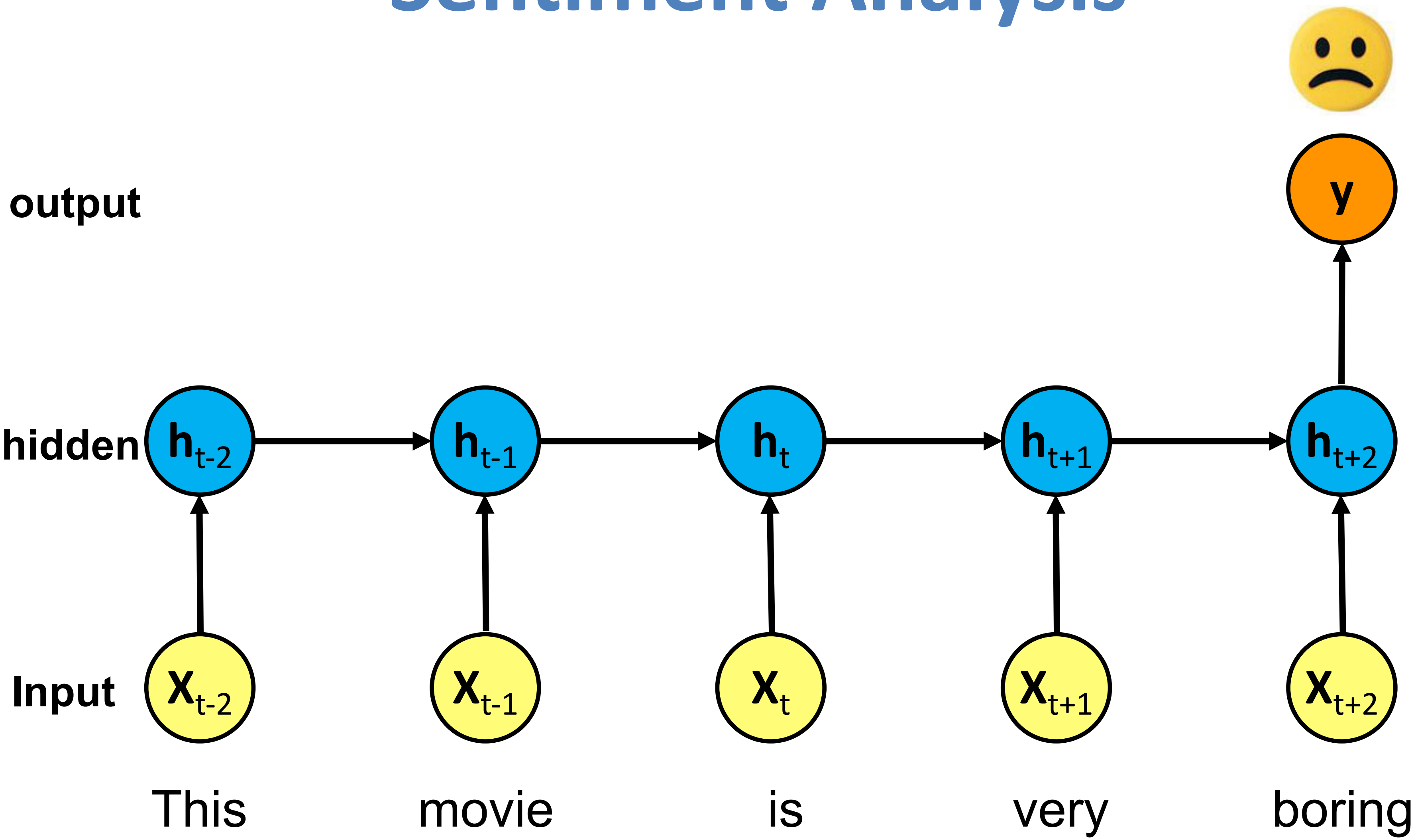
Recurrent Neural Networks (RNN)



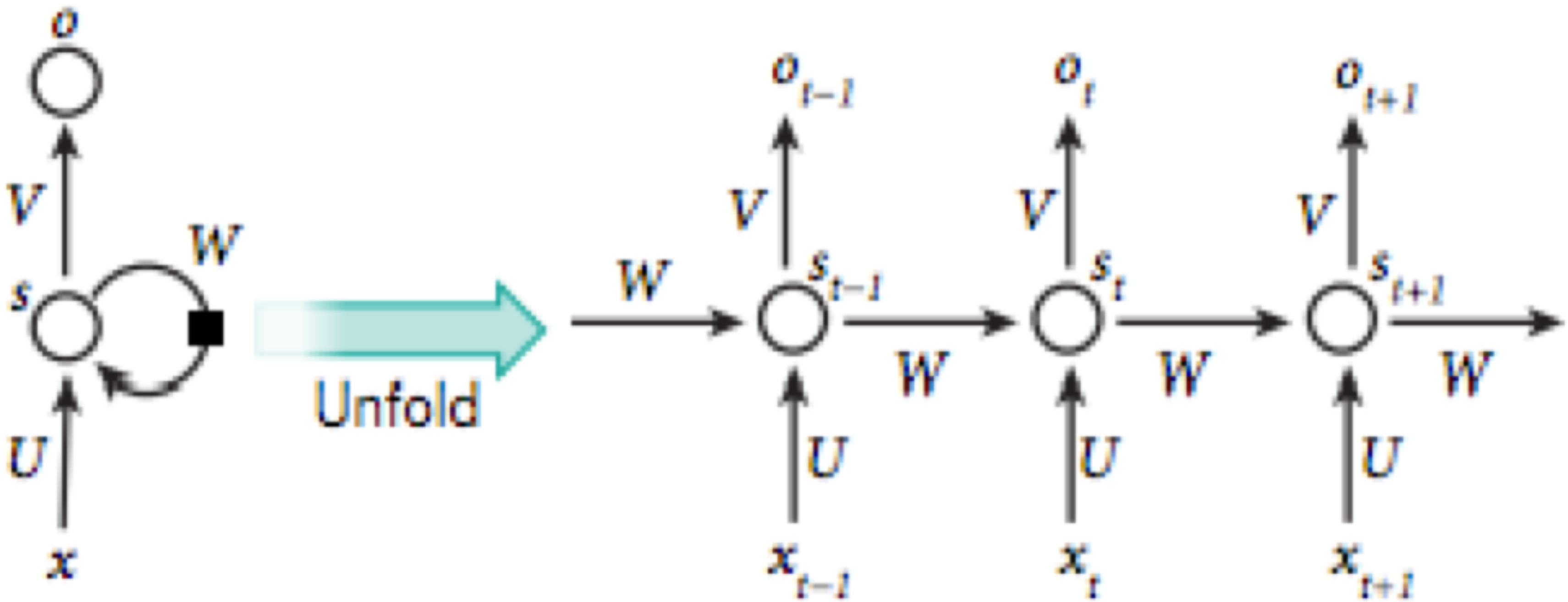
Recurrent Neural Networks (RNN) Sentiment Analysis



Recurrent Neural Networks (RNN) Sentiment Analysis



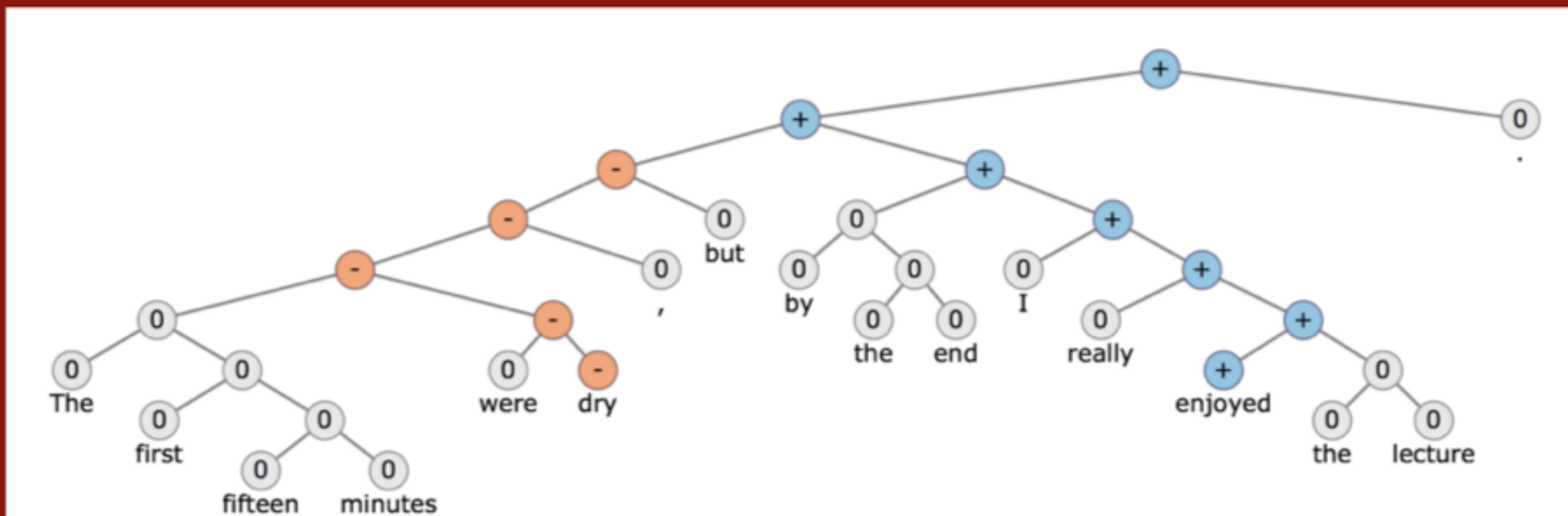
Recurrent Neural Network (RNN)



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

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

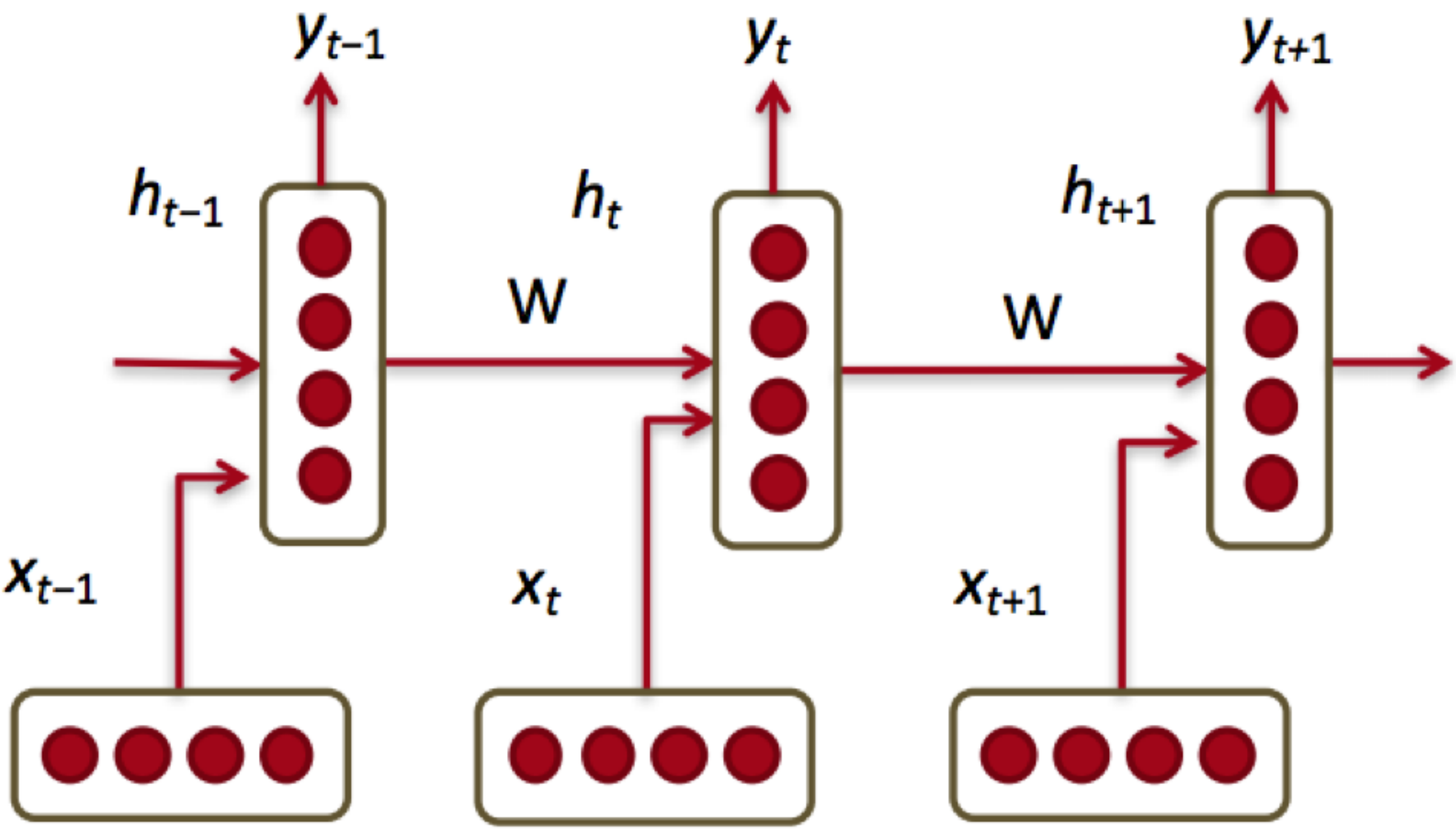


Course Description

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

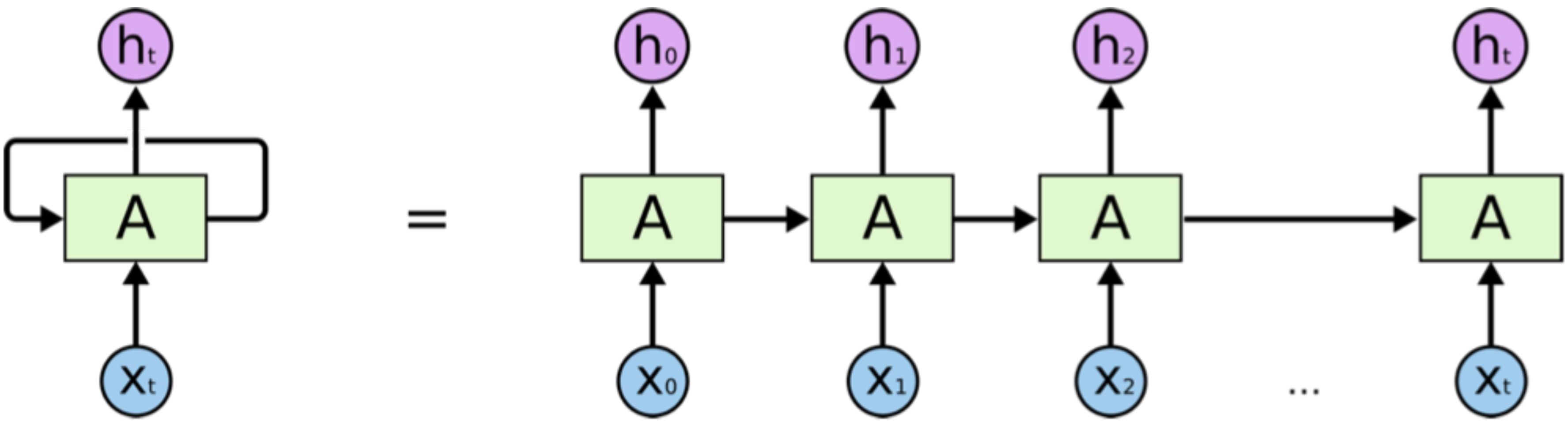
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



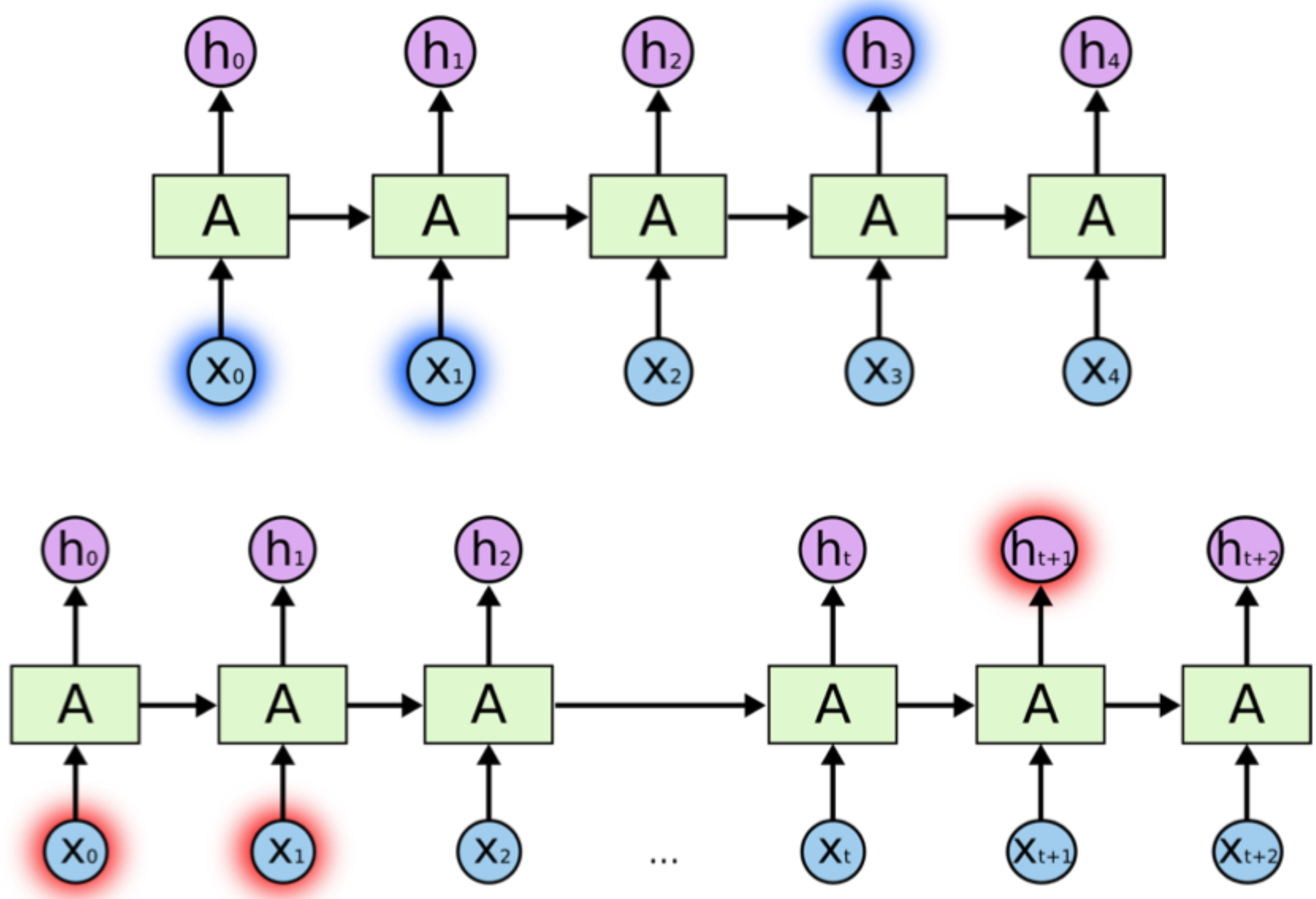
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN



Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

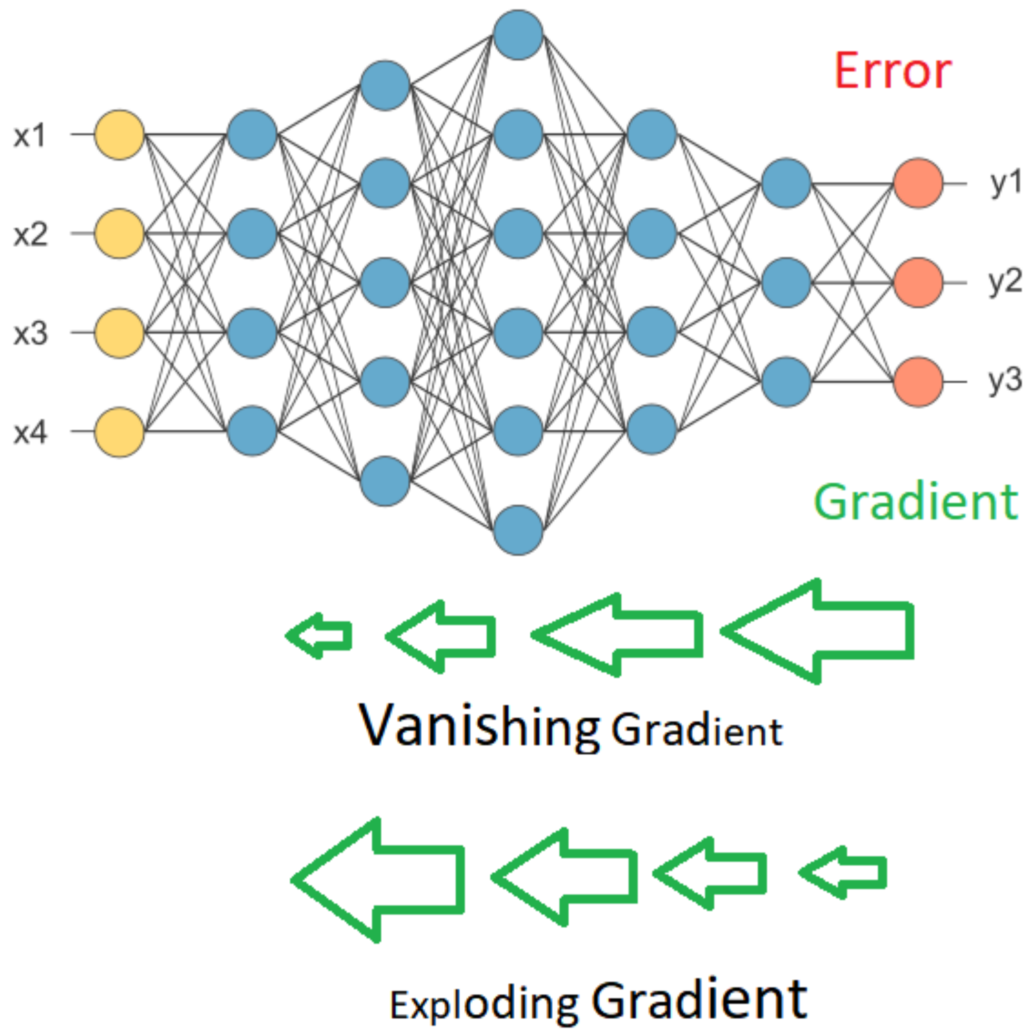
RNN long-term dependencies



I grew up in France… I speak fluent French.

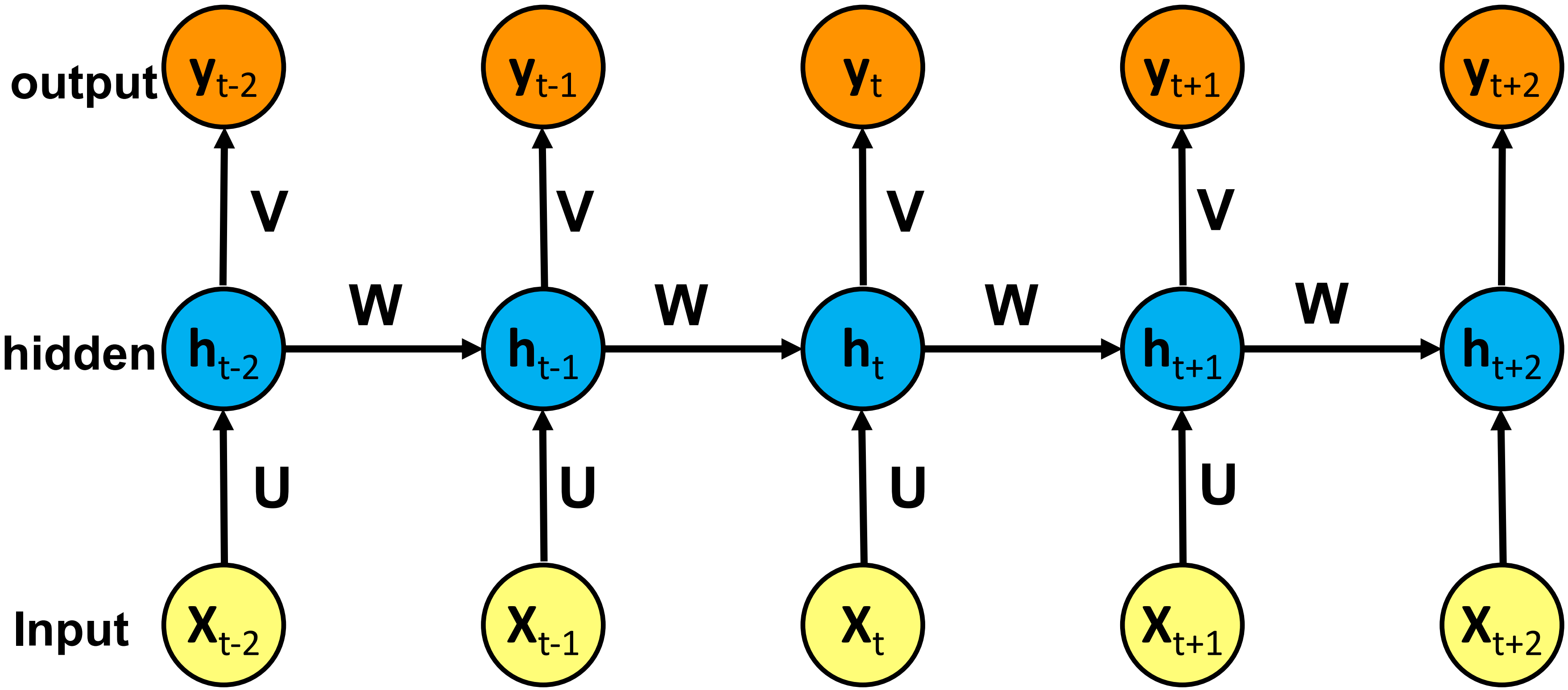
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Vanishing Gradient
Exploding Gradient

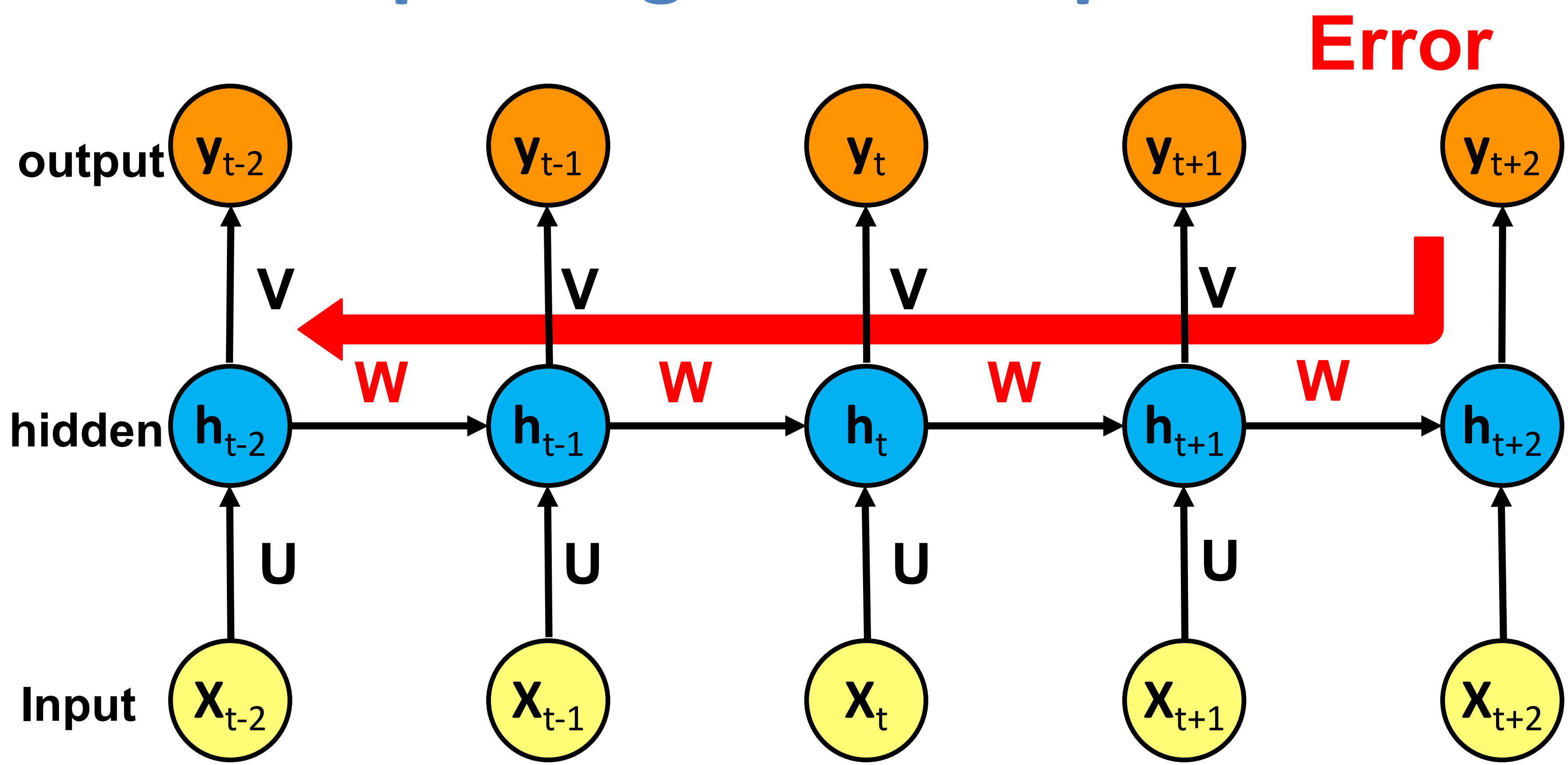


Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

Recurrent Neural Networks (RNN)



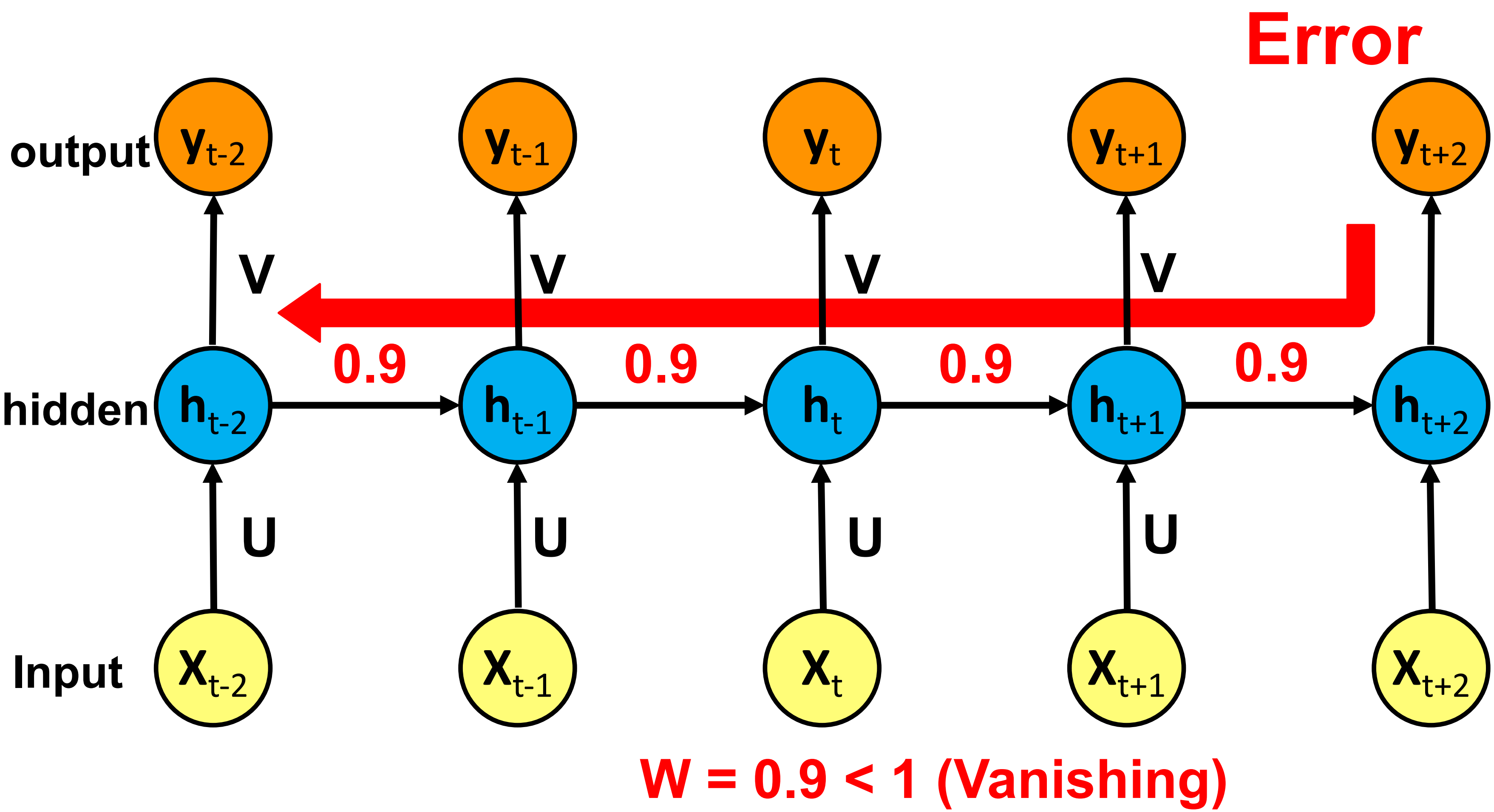
RNN
Vanishing Gradient problem
Exploding Gradient problem



if $|W| < 1$ (Vanishing)
if $|W| > 1$ (Exploding)

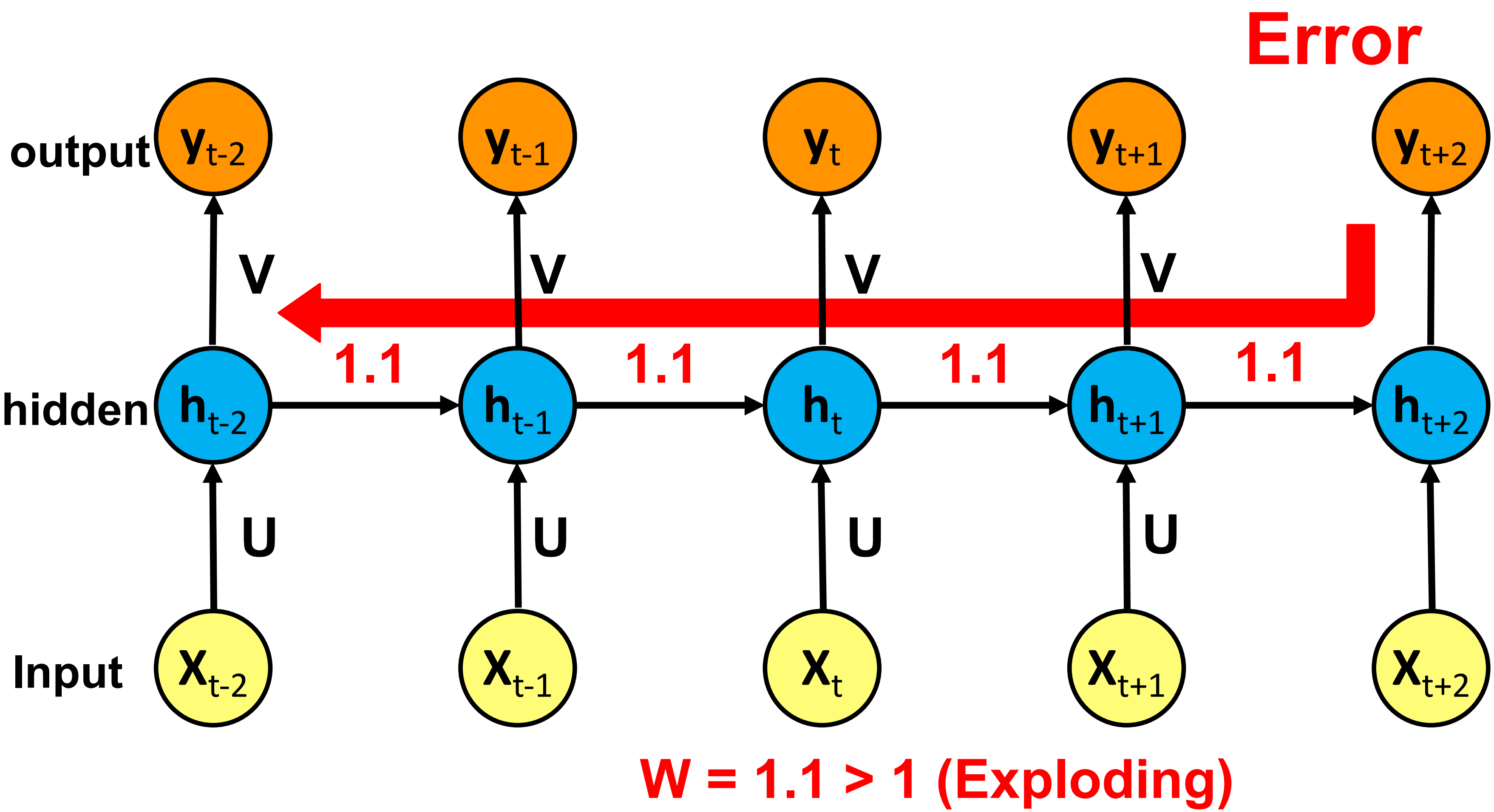
Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

RNN
Vanishing Gradient problem



Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

RNN
Exploding Gradient problem

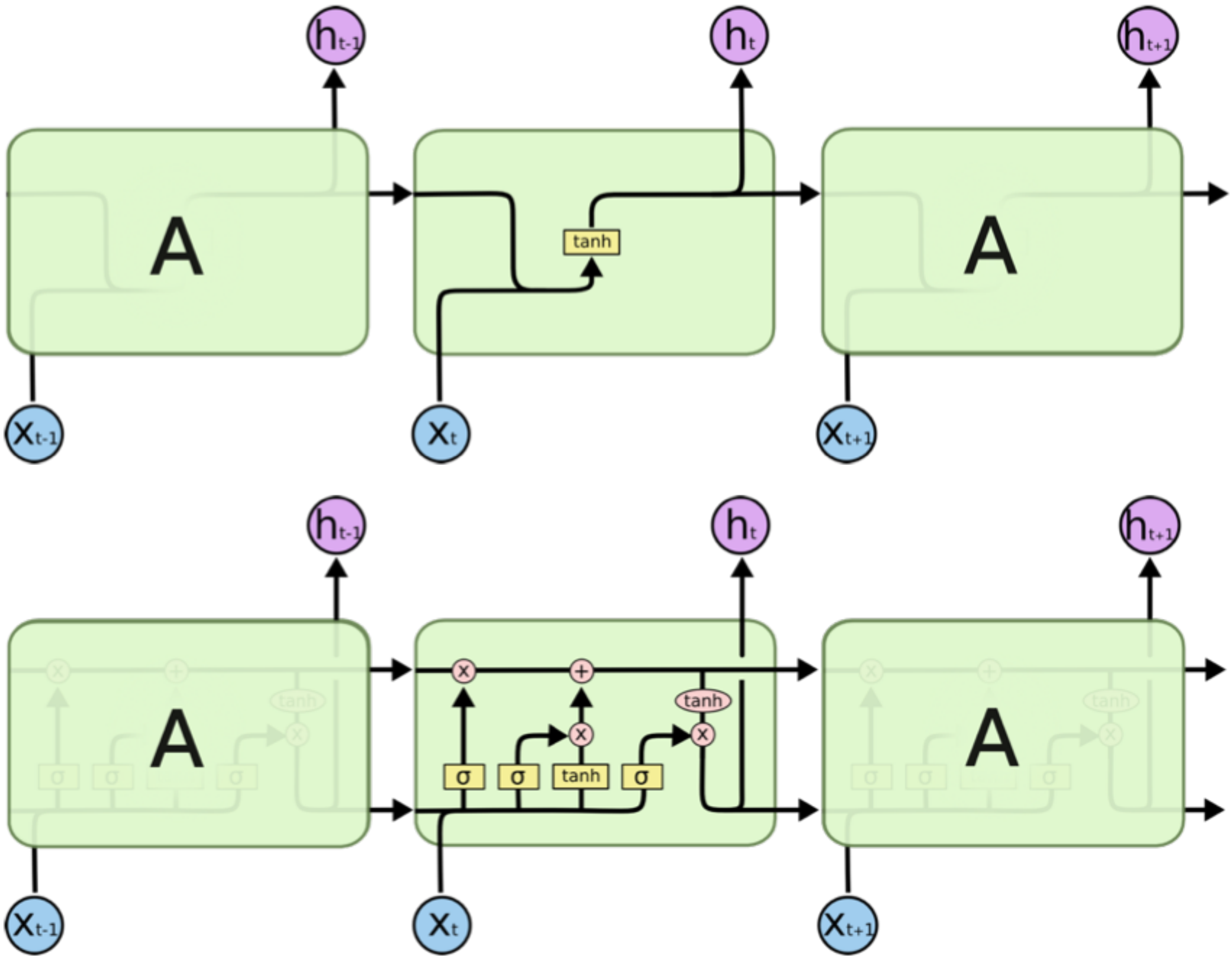


Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

RNN LSTM

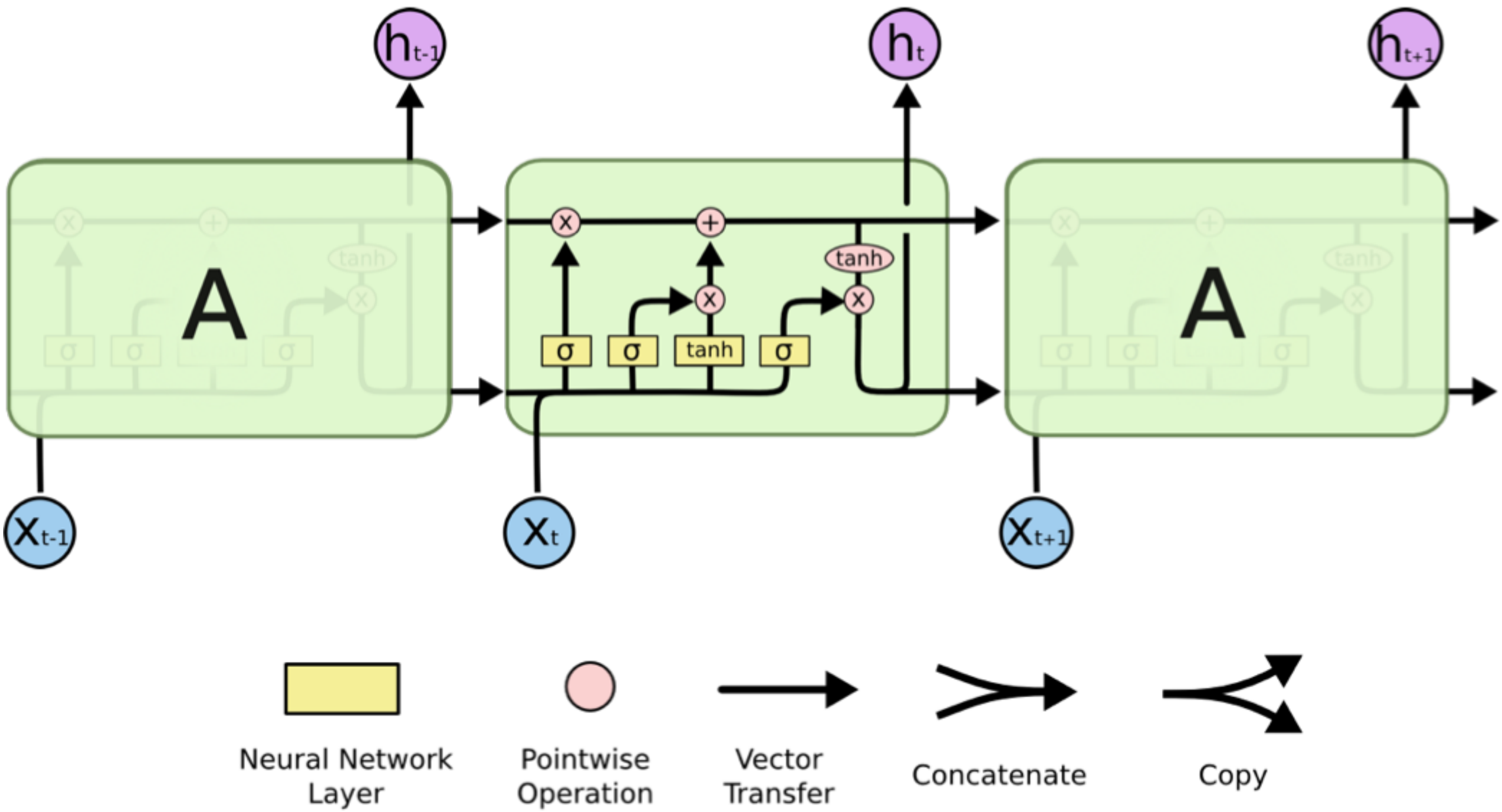
RNN

LSTM



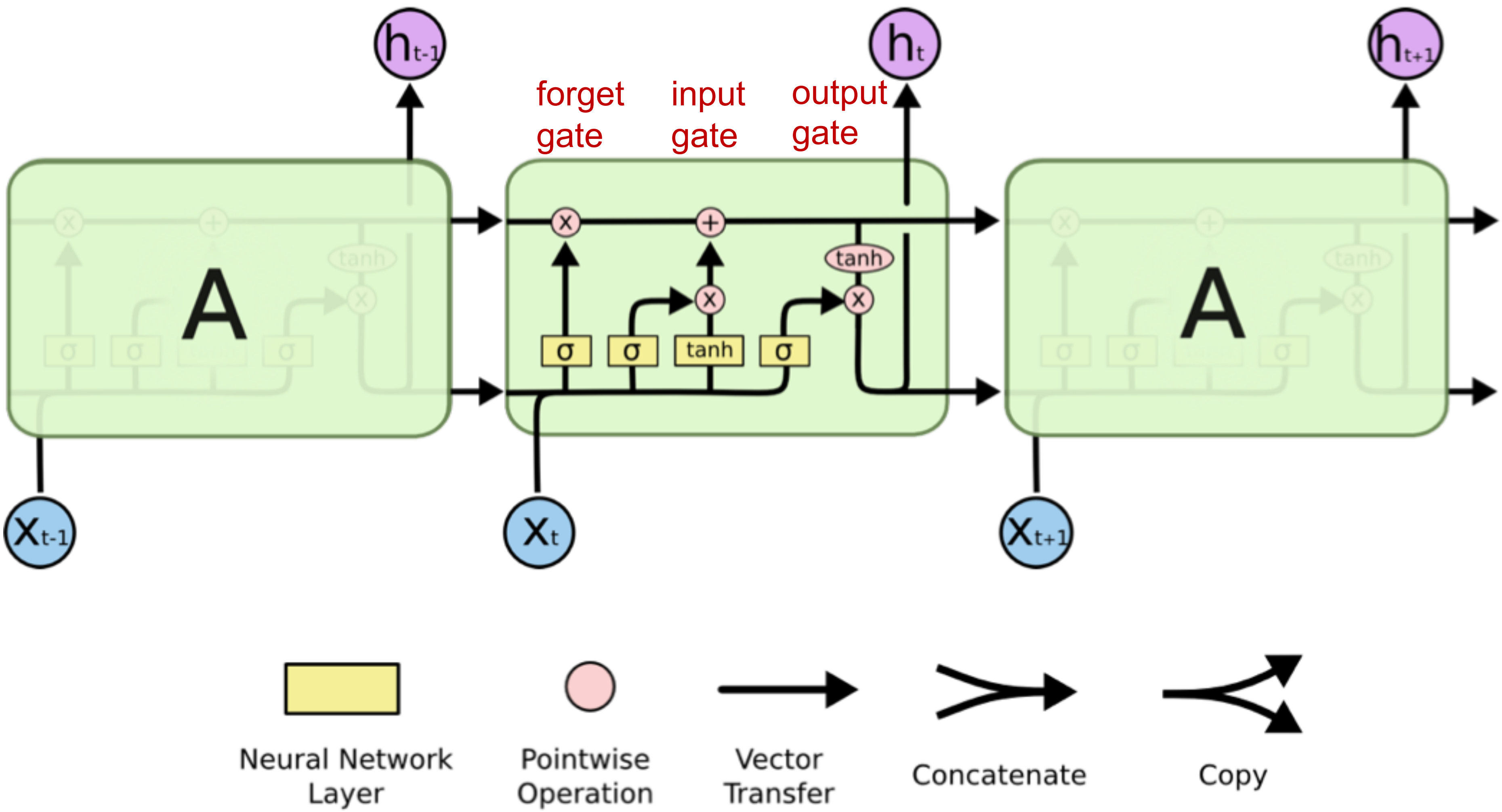
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



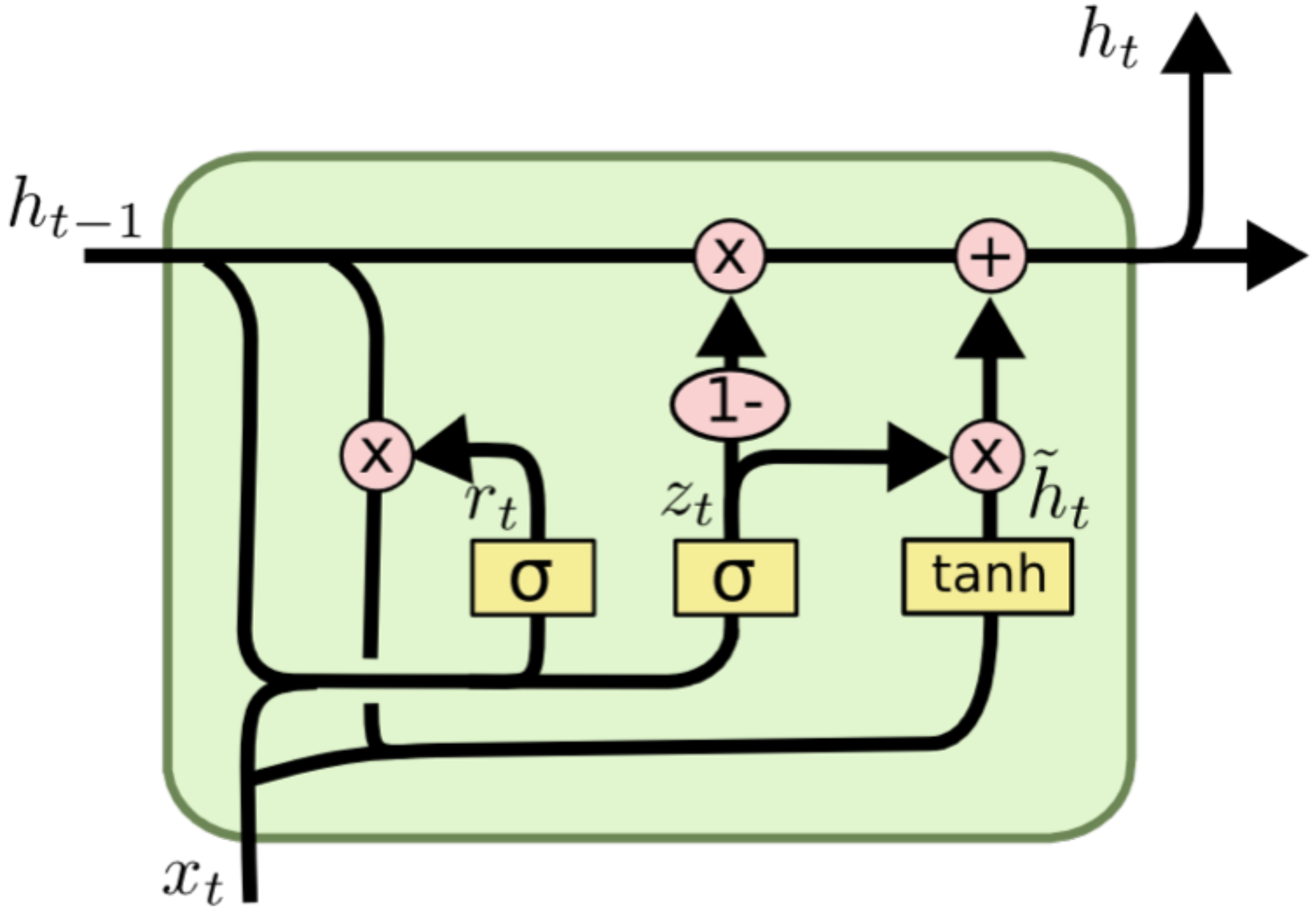
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



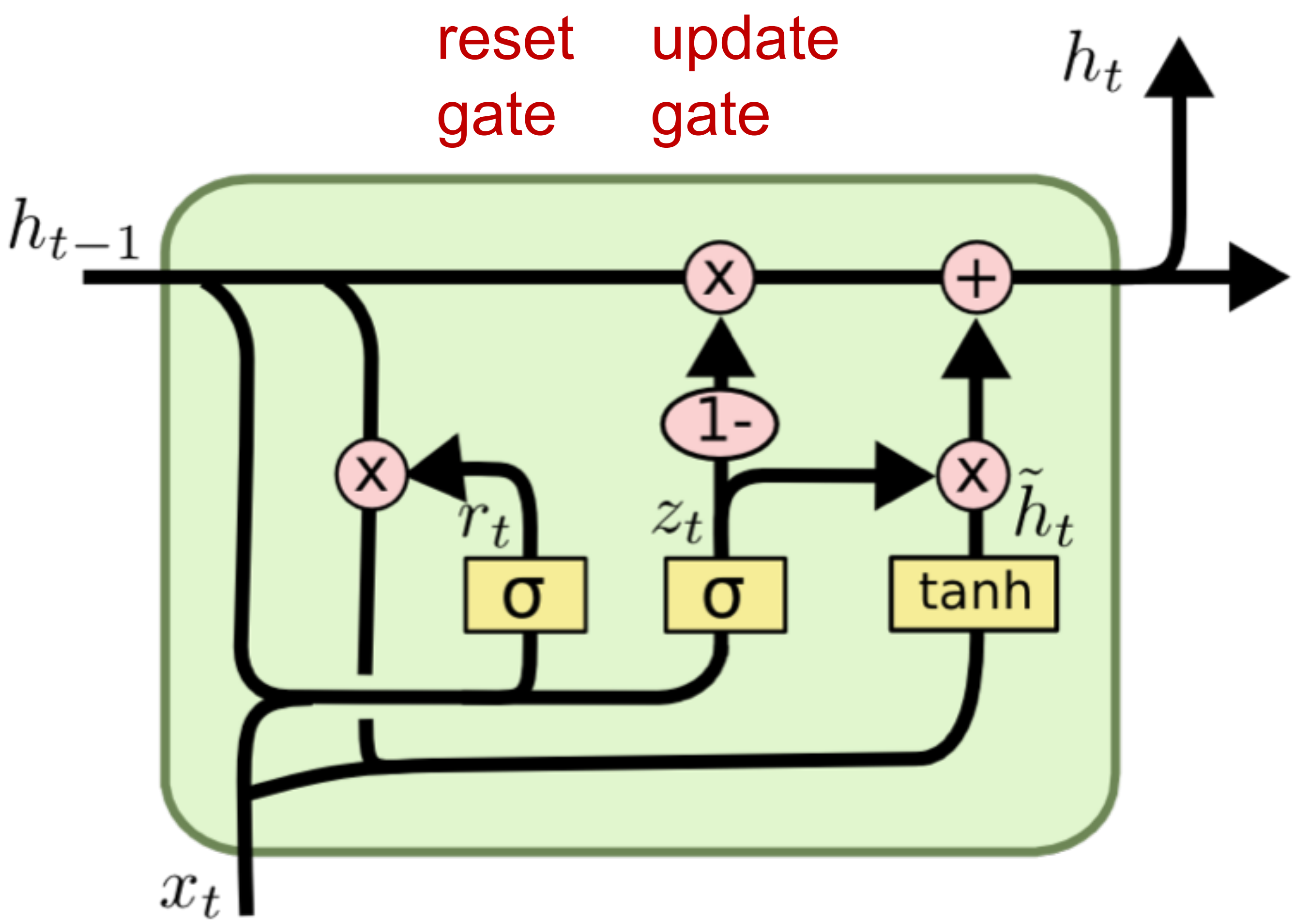
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



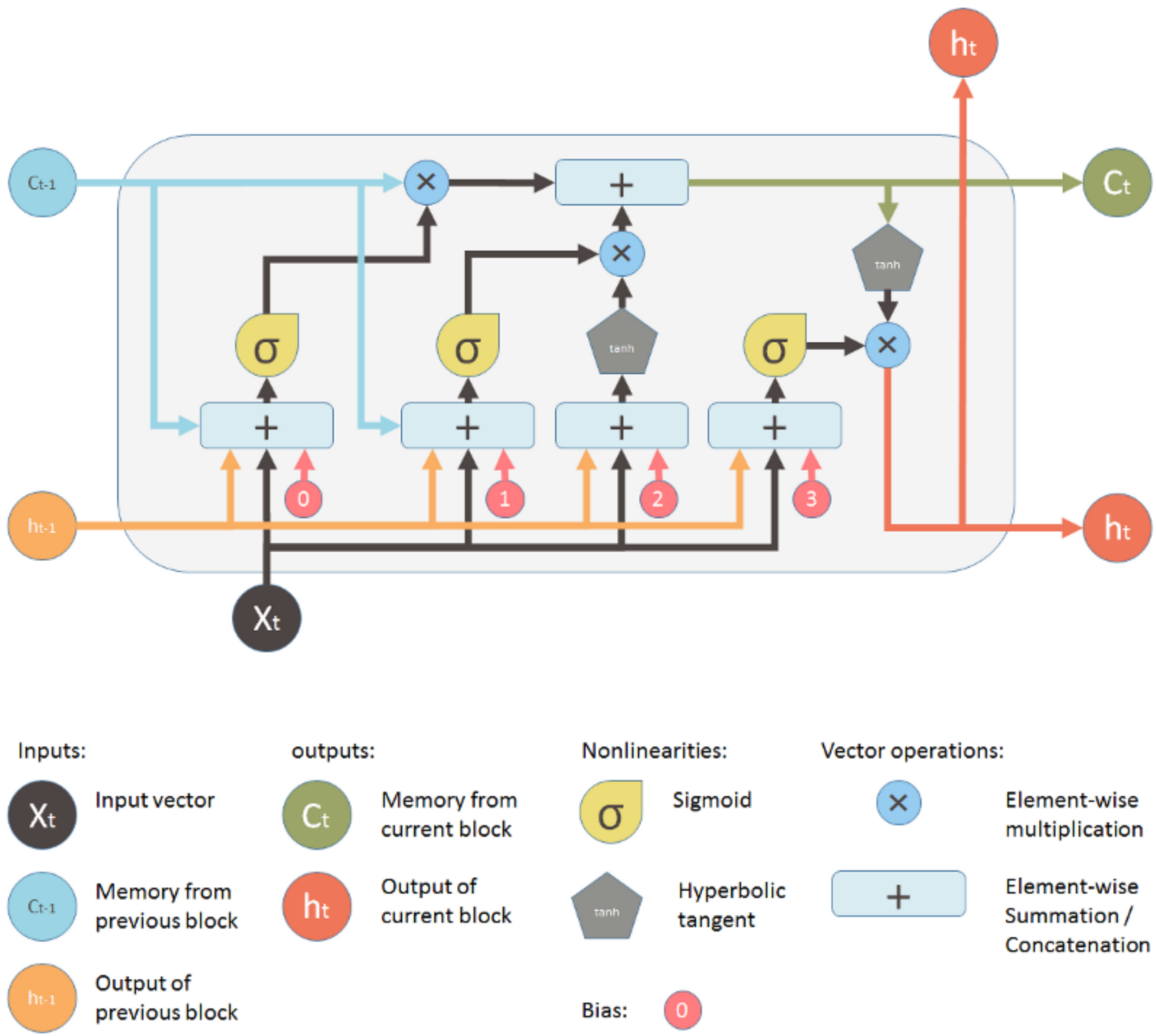
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



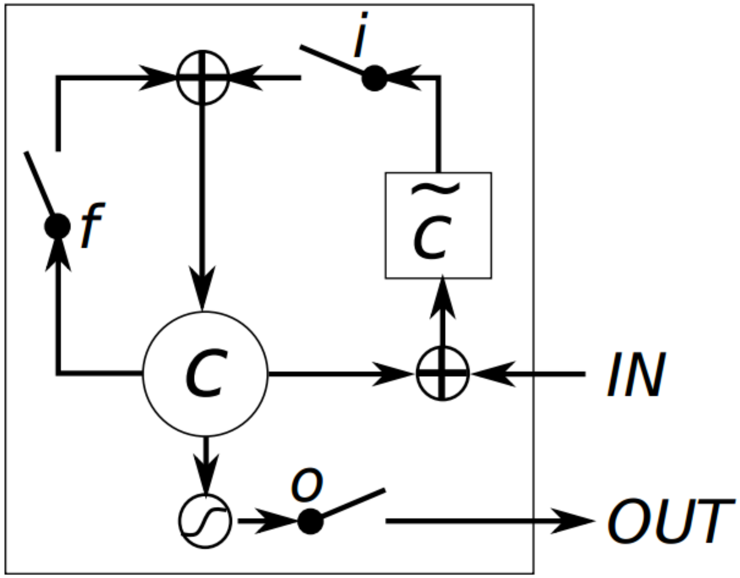
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

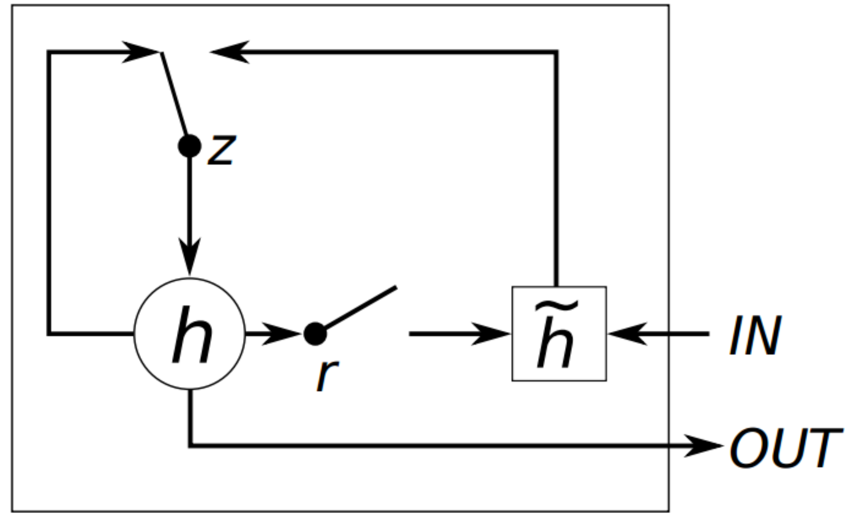
LSTM



Source: Shi Yan (2016), Understanding LSTM and its diagrams, https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

LSTM vs GRU





LSTM

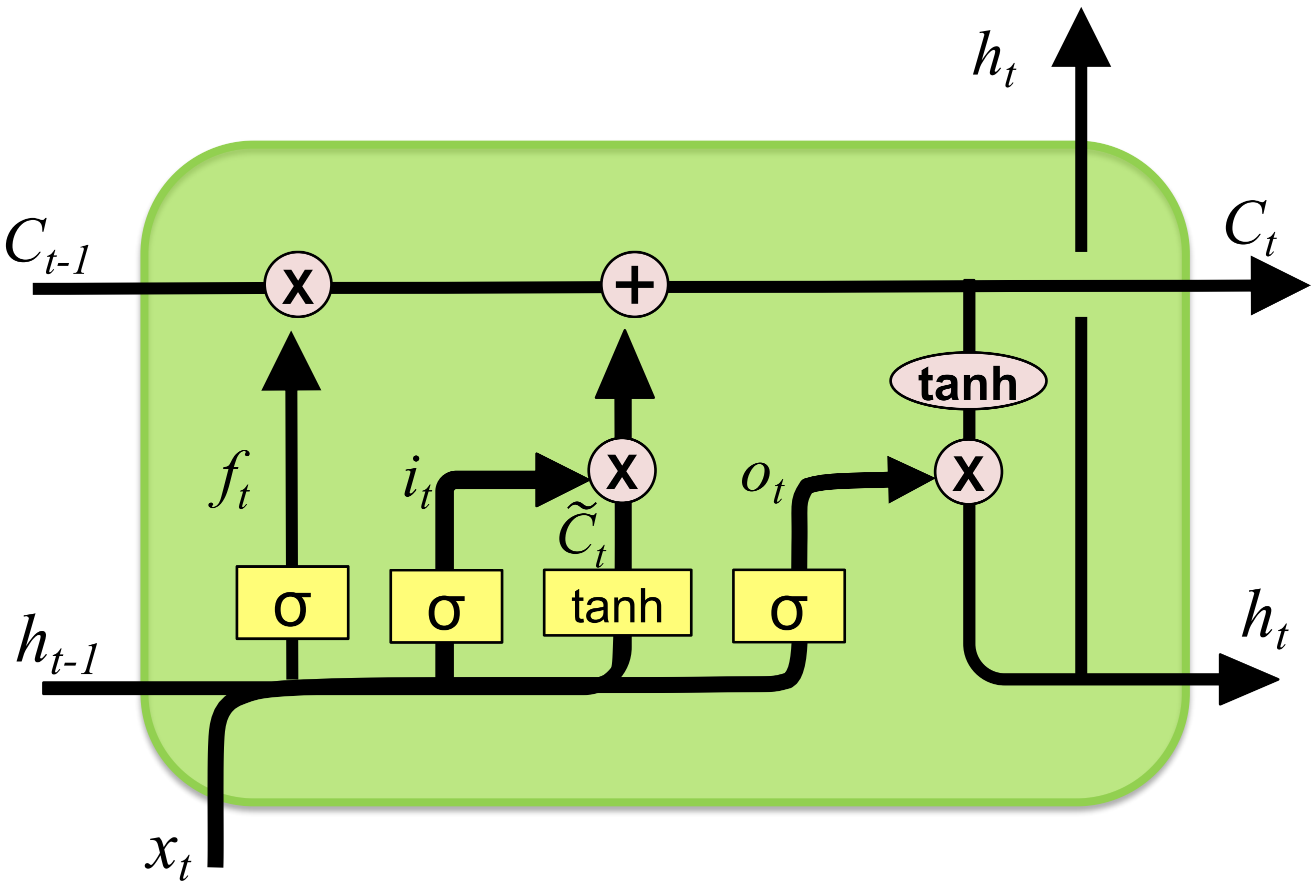
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

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

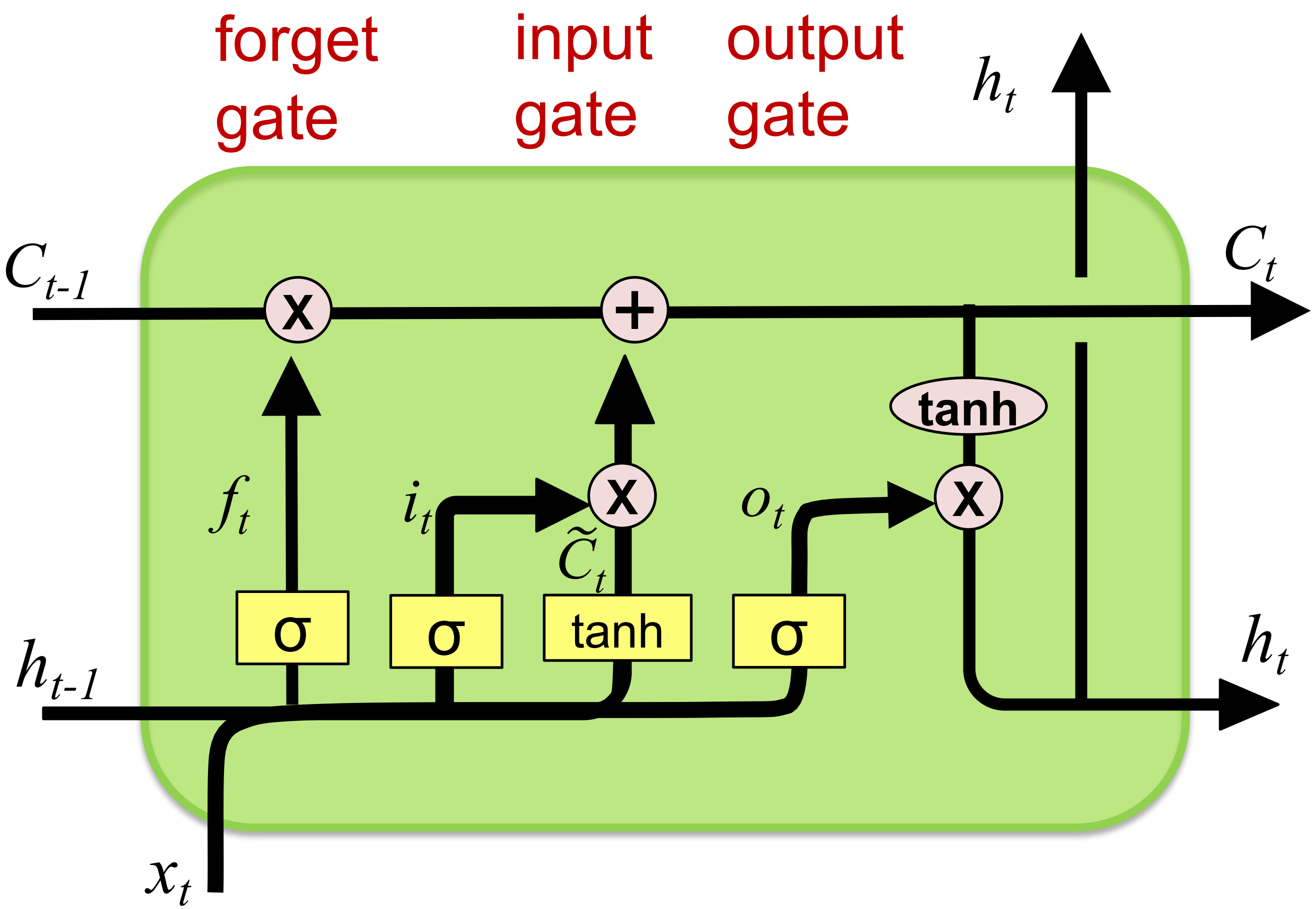
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

Long Short Term Memory (LSTM)



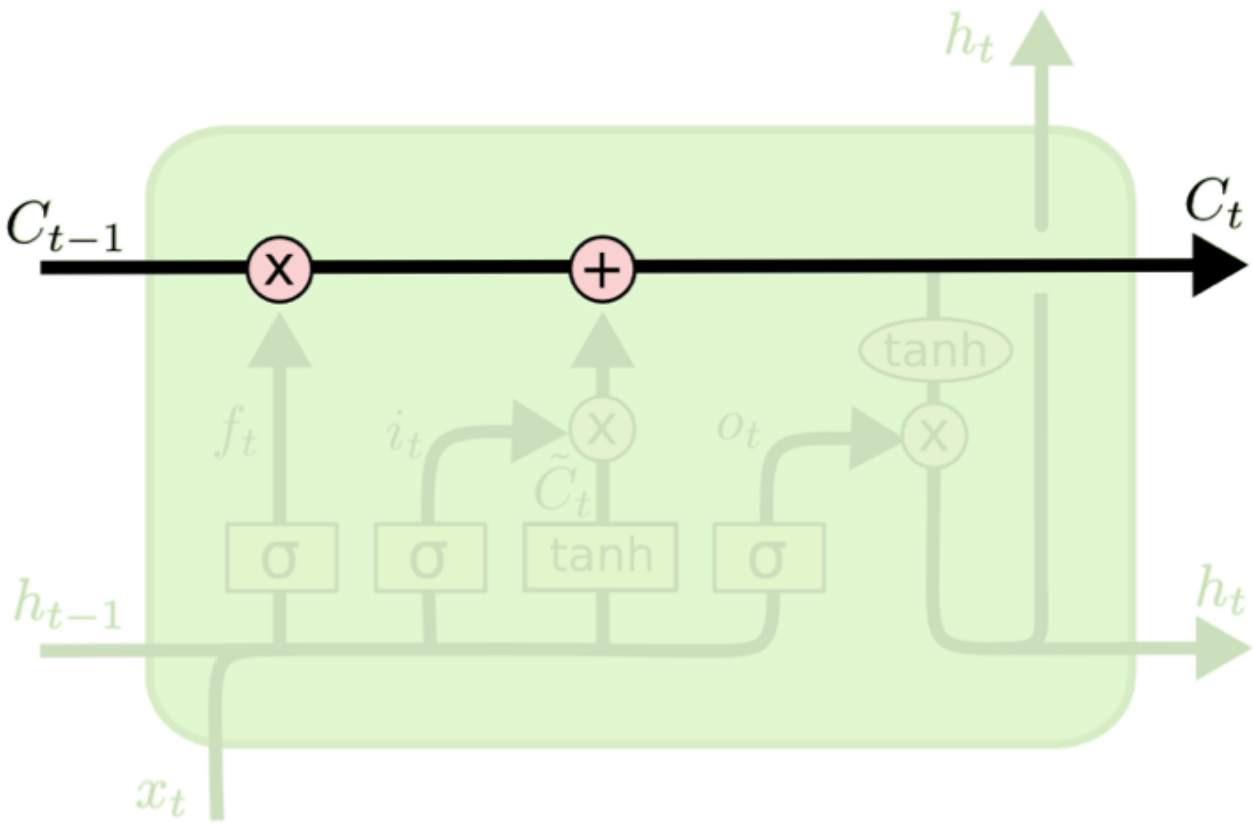
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



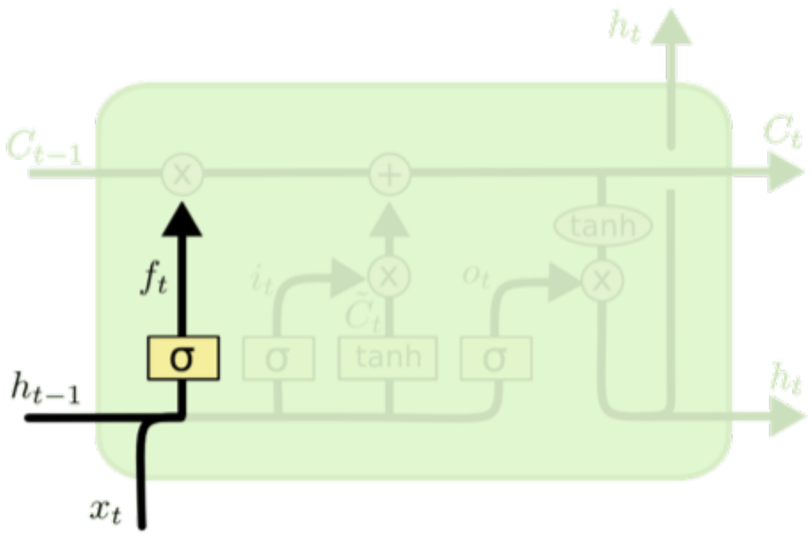
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
Memory state (C)



Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

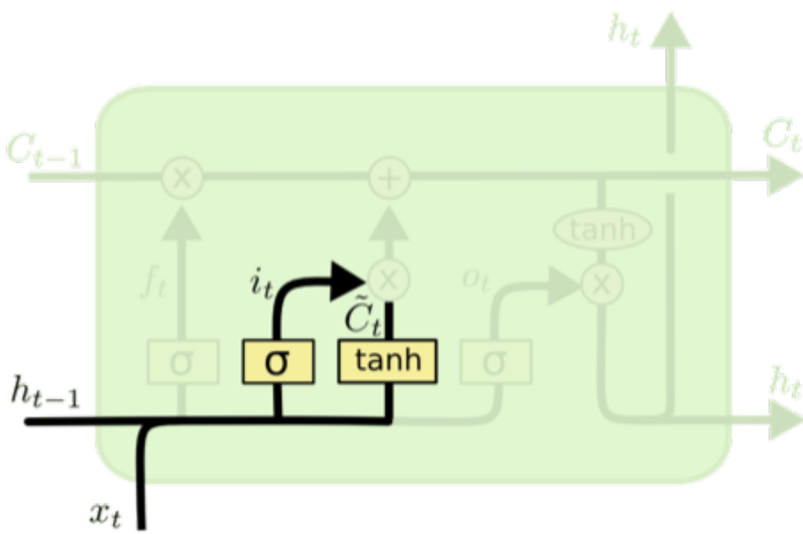
LSTM
forget gate (f)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
input gate (i)

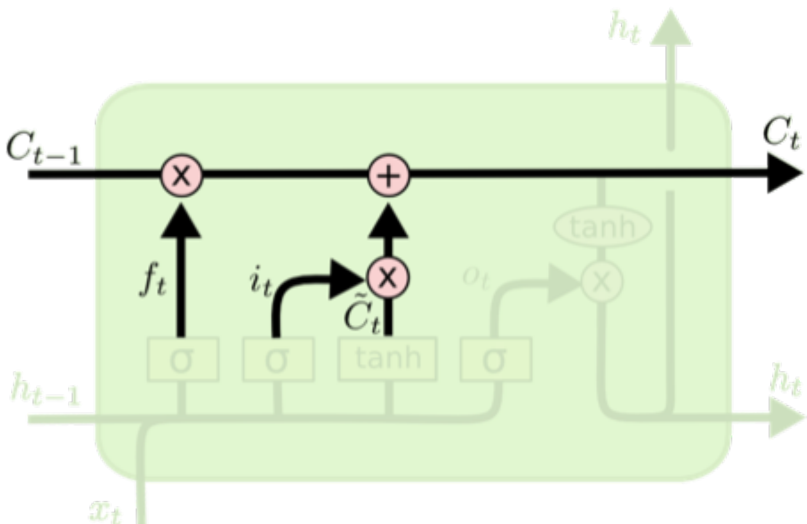


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

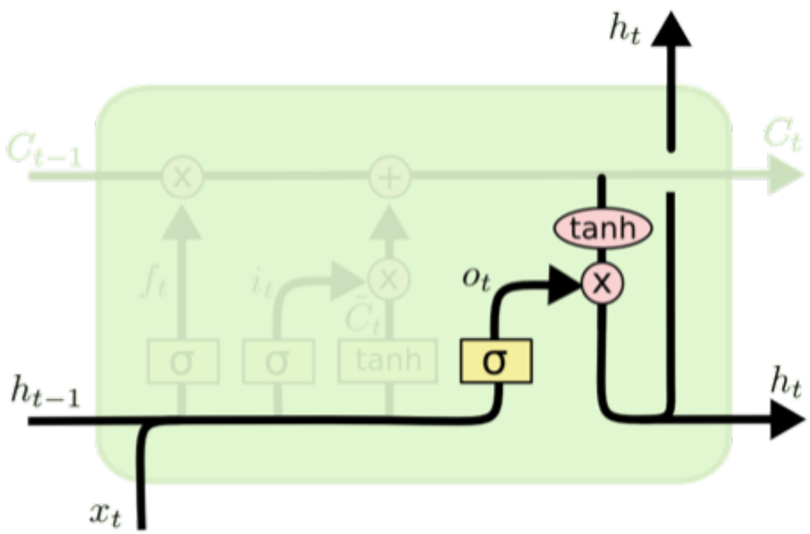
LSTM
Memory state (C)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
output gate (o)

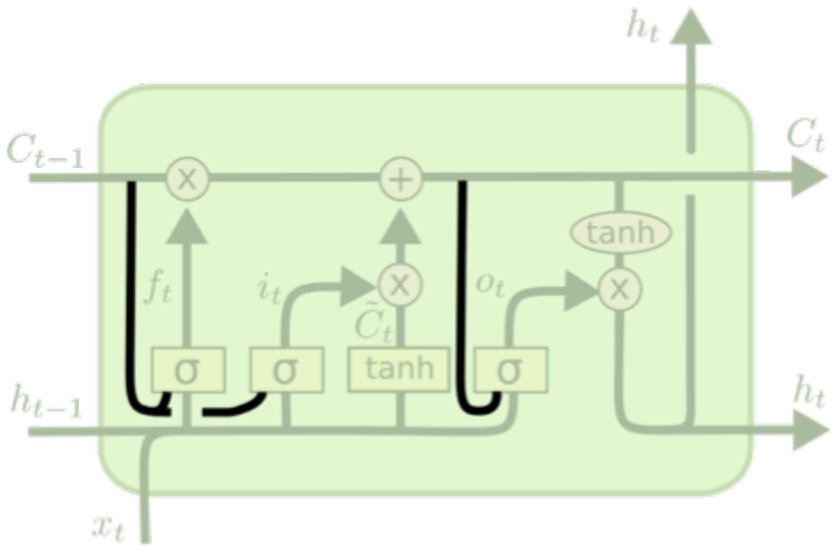


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
forget (f), input (i), output (o) gates



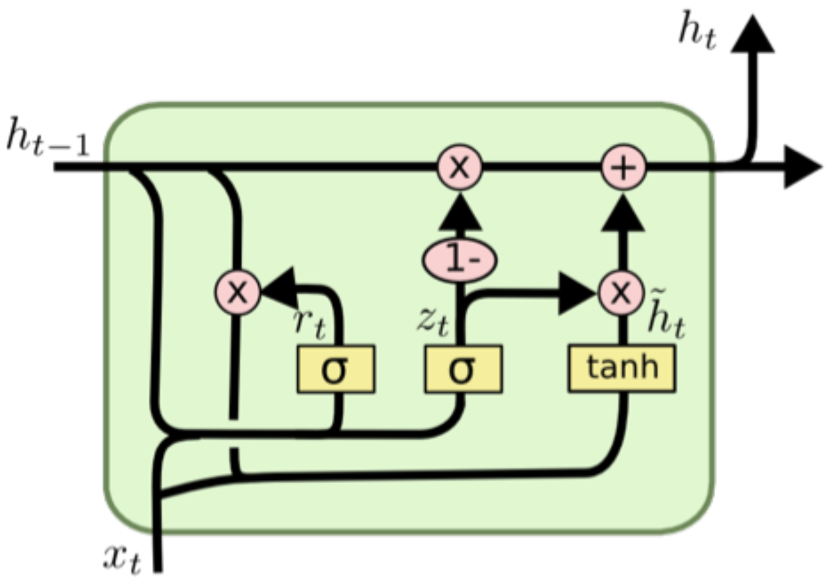
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$

$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)
update (z), reset (r) gates



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

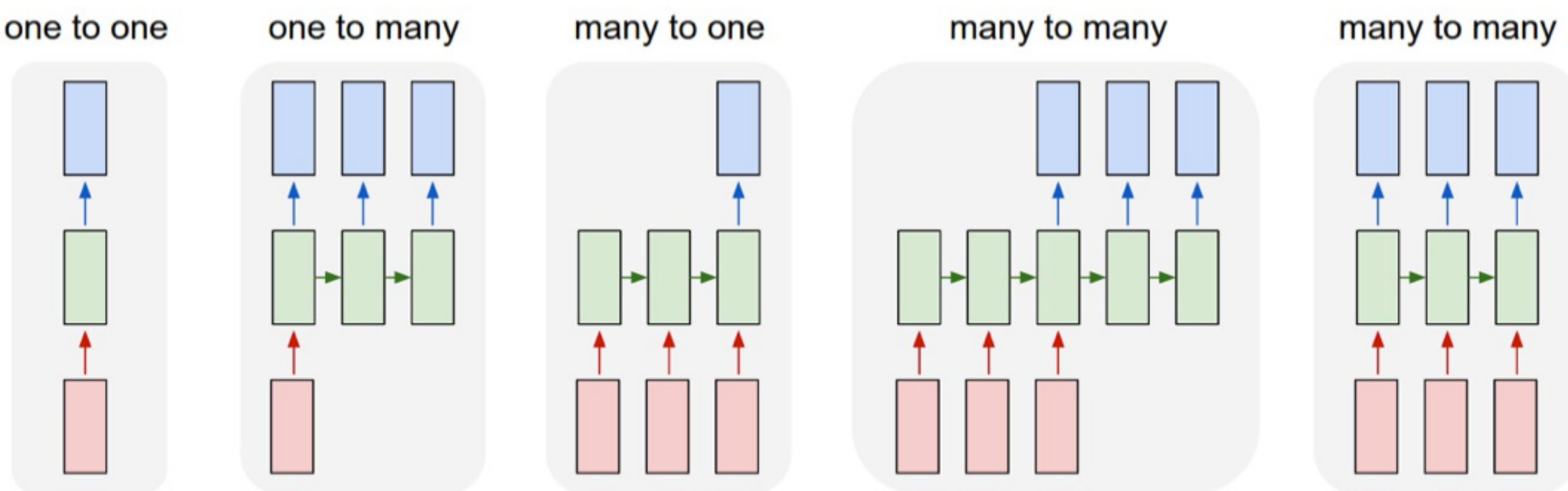
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Recurrent Neural Network



Traditional Neural Network

Music Generation

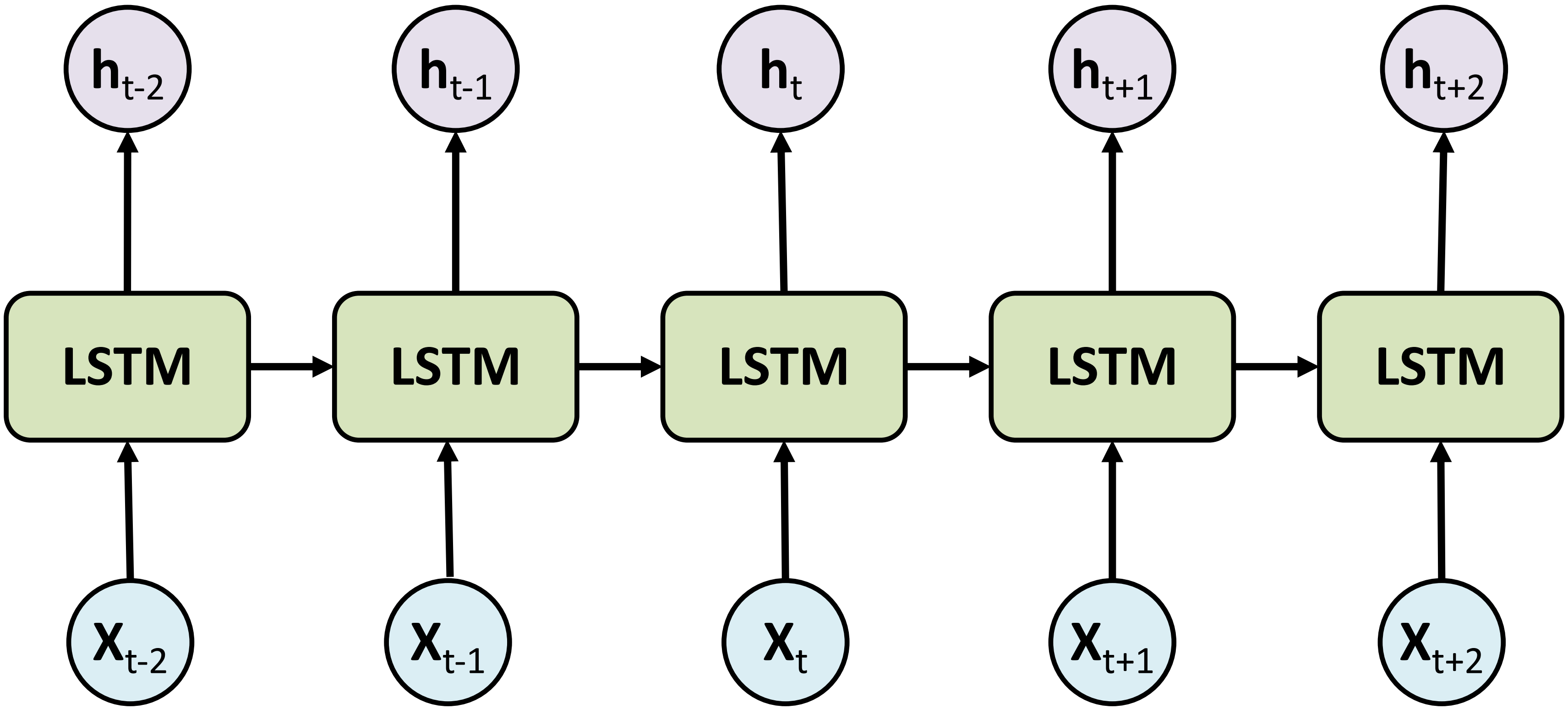
Sentiment Classification

Name Entity Recognition

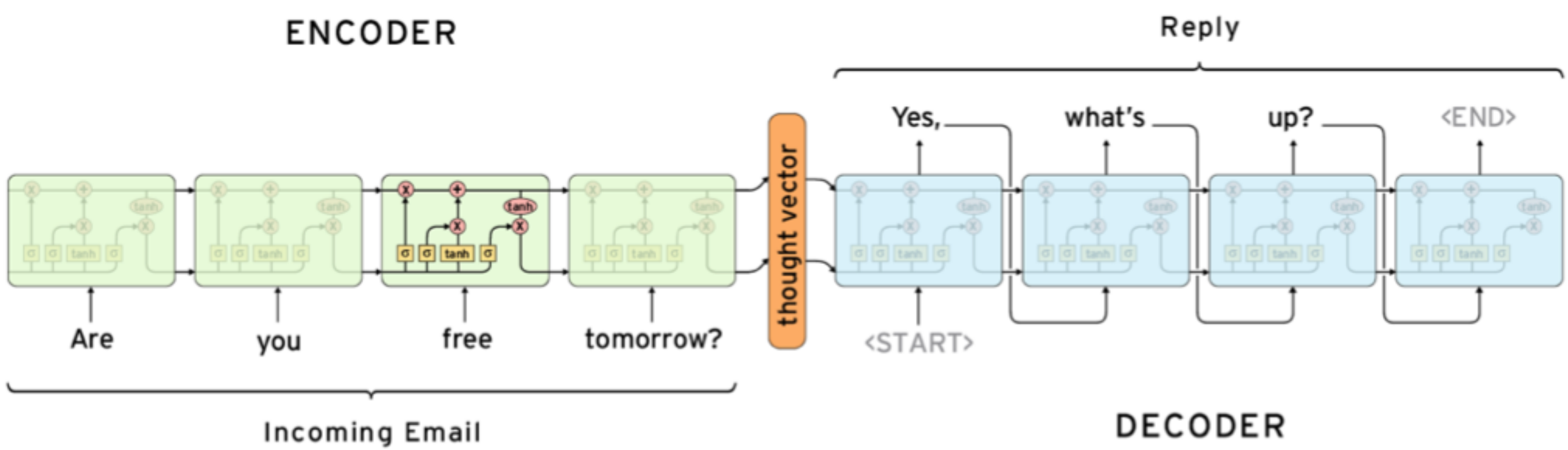
Machine Translation

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

Long Short Term Memory (LSTM) for Time Series Forecasting

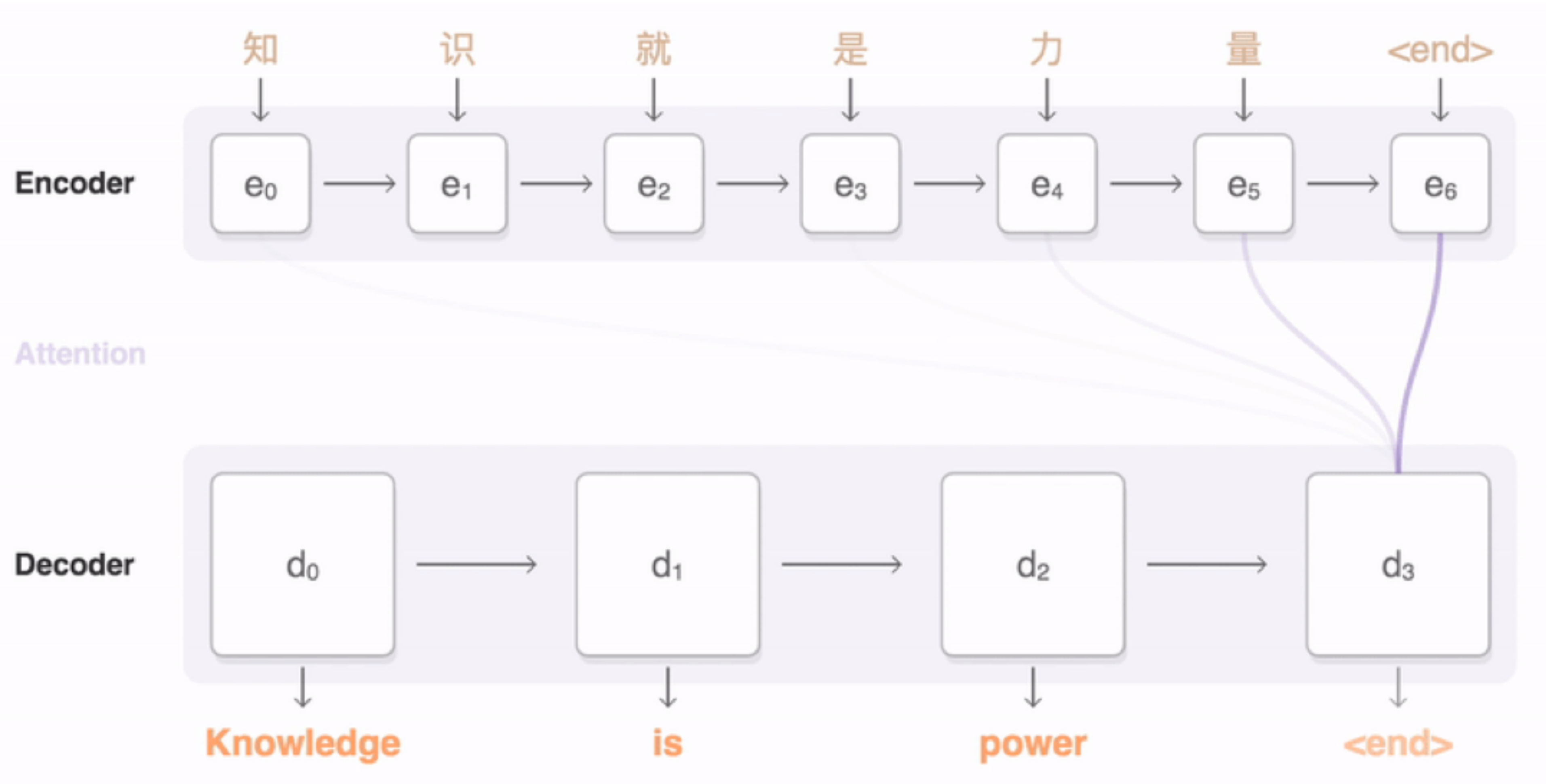


The Sequence to Sequence model (seq2seq)



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

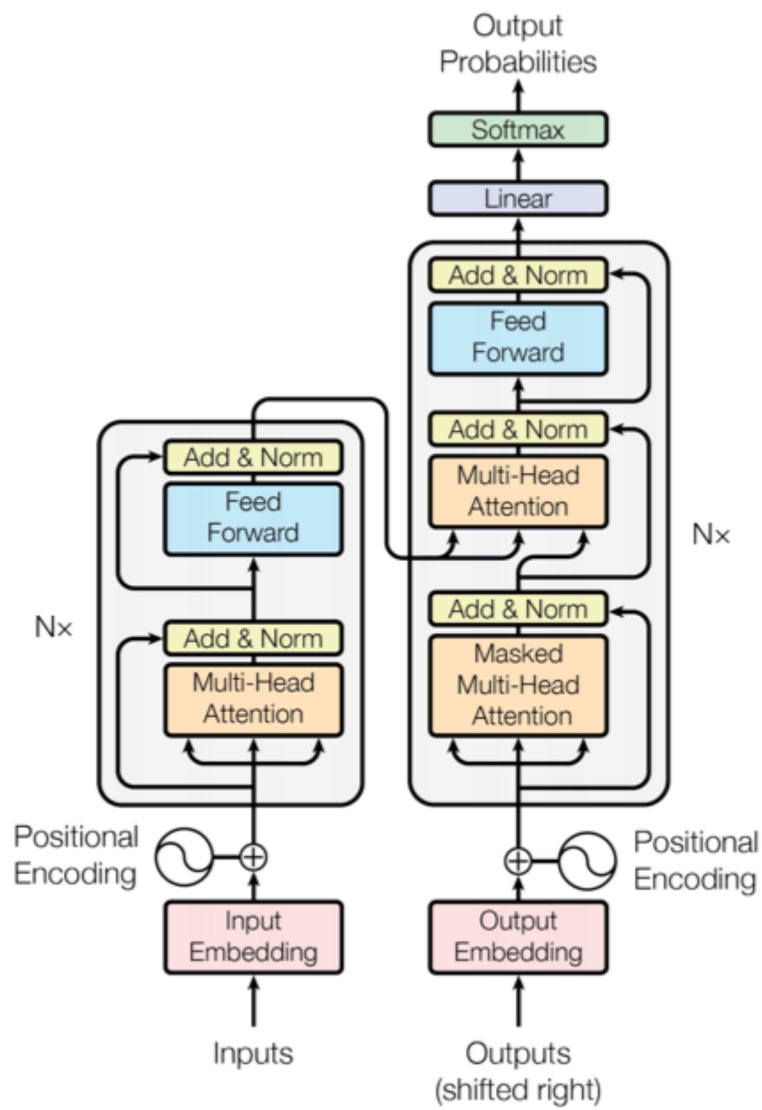
Sequence to Sequence (Seq2Seq)



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

Transformer (Attention is All You Need)

(Vaswani et al., 2017)

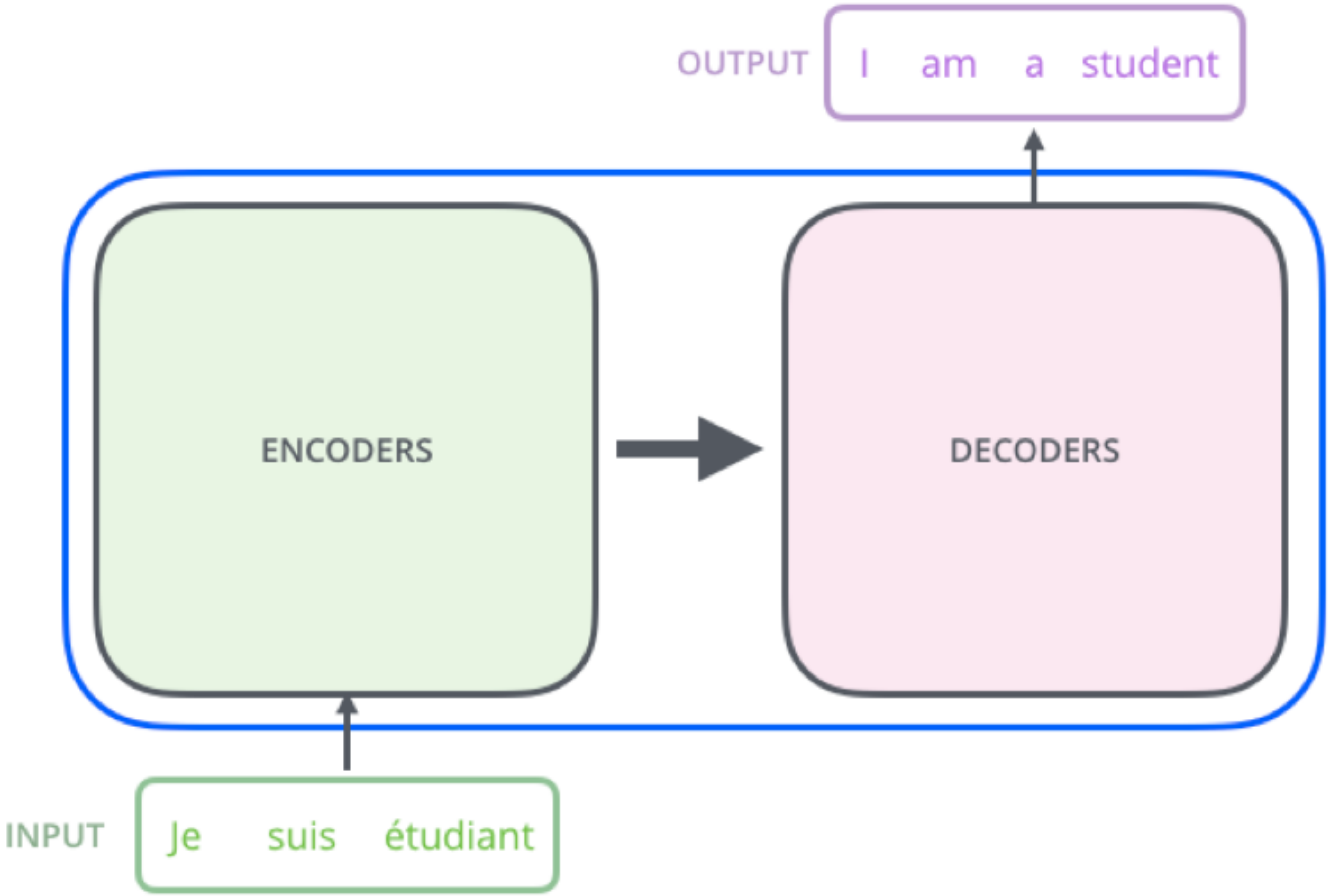


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

Transformer

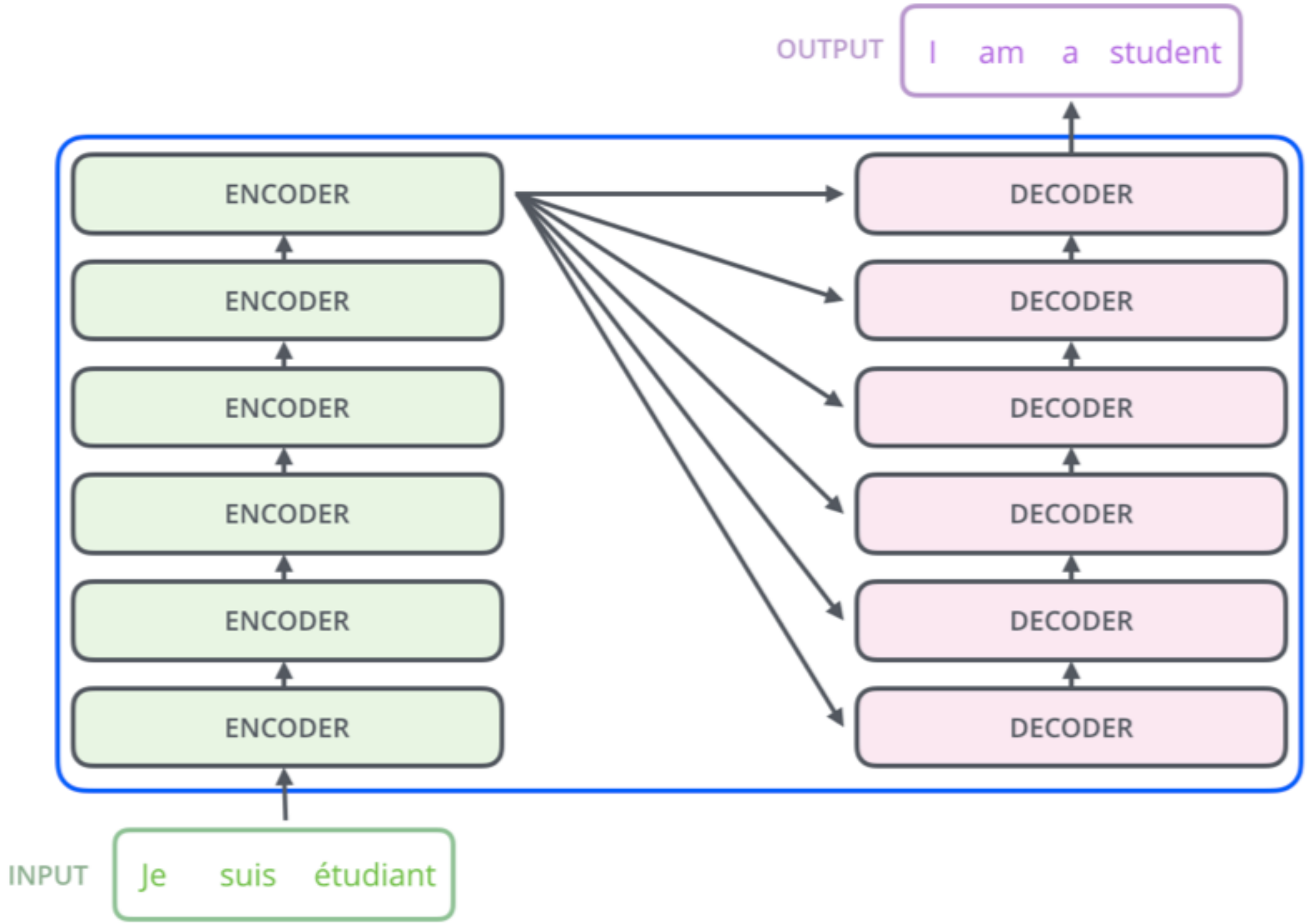
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Encoder Decoder



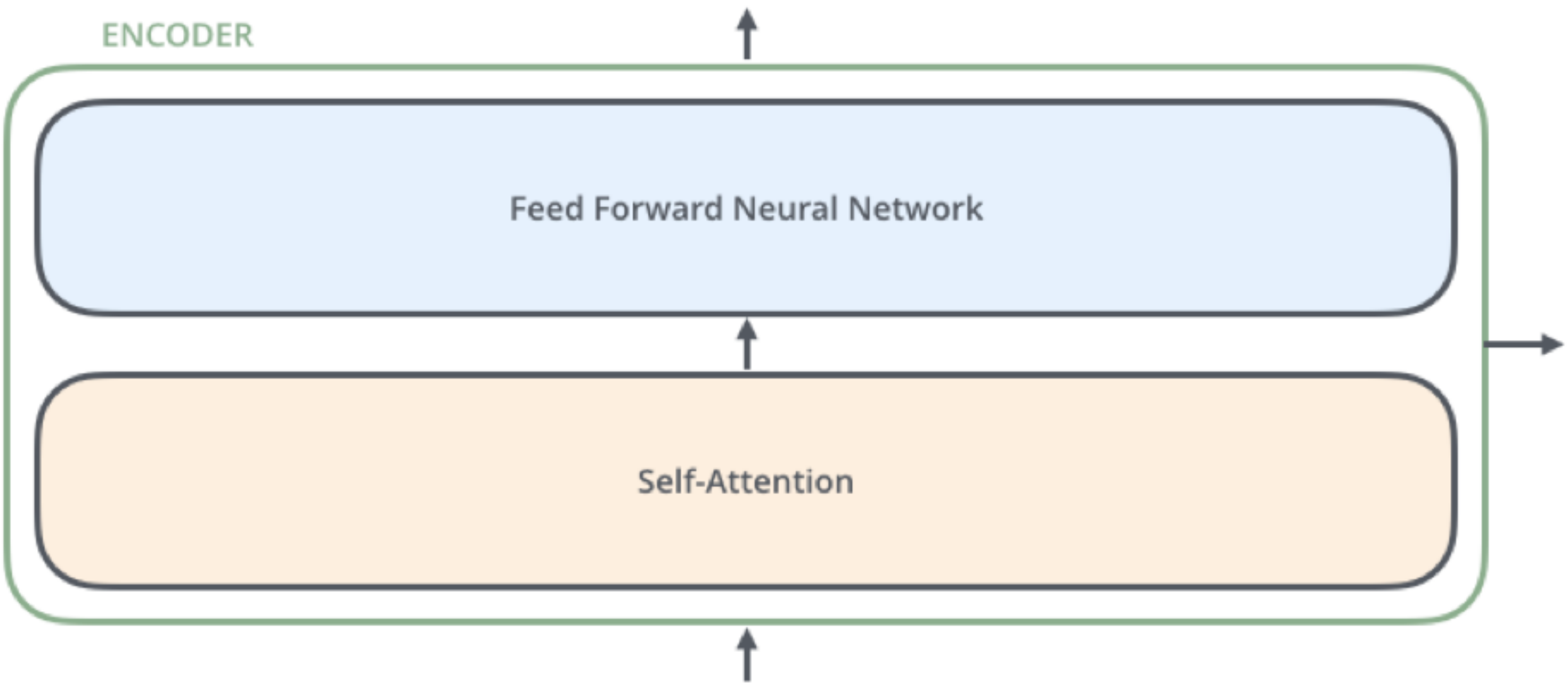
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Encoder Decoder Stack



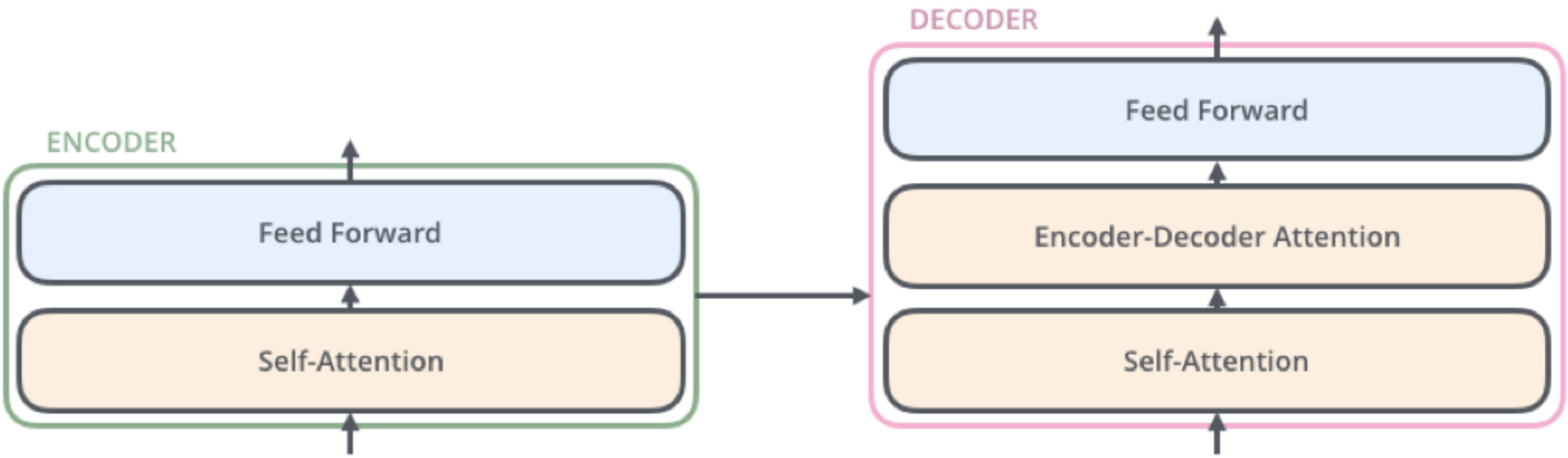
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Encoder Self-Attention



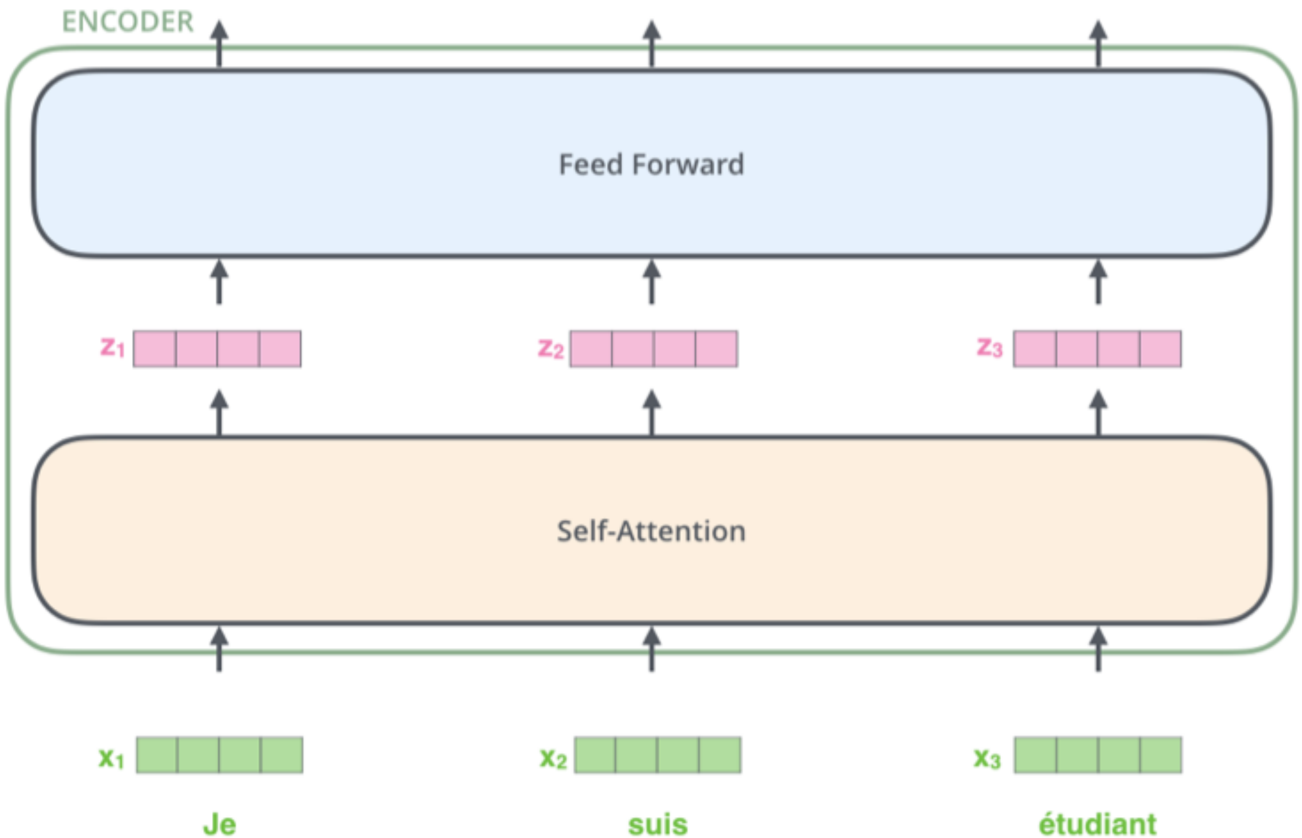
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Decoder



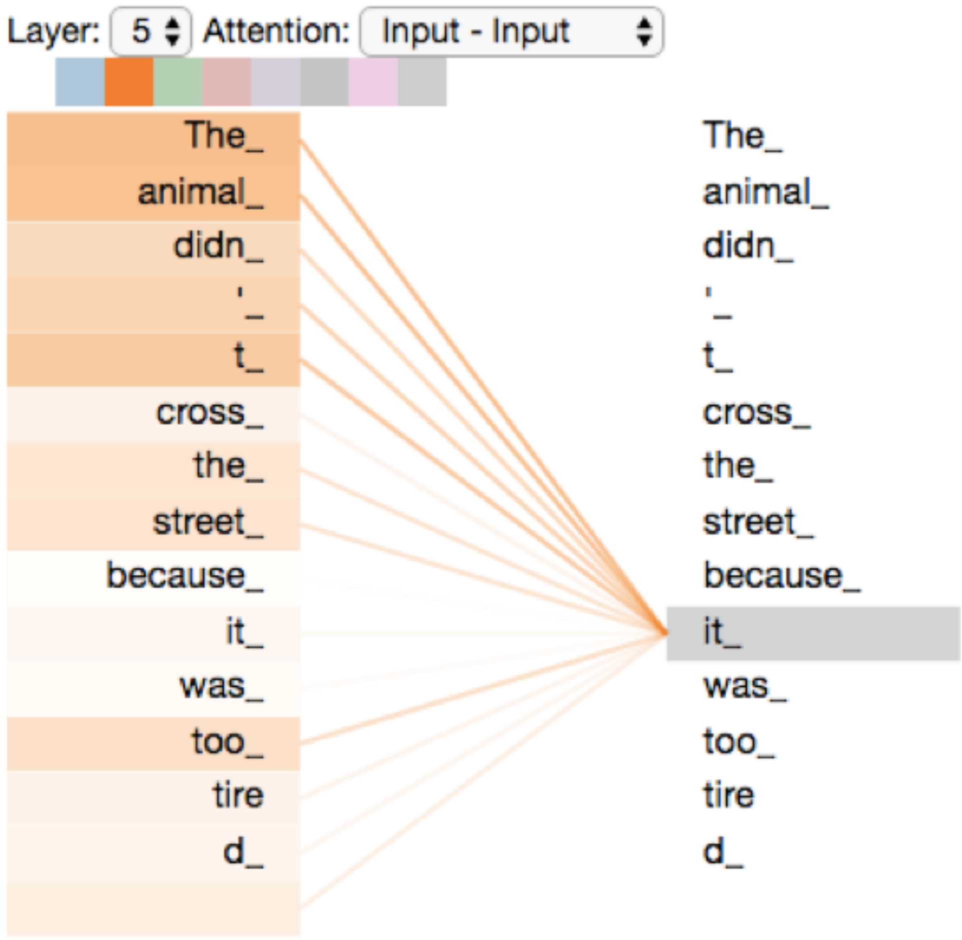
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Encoder with Tensors Word Embeddings



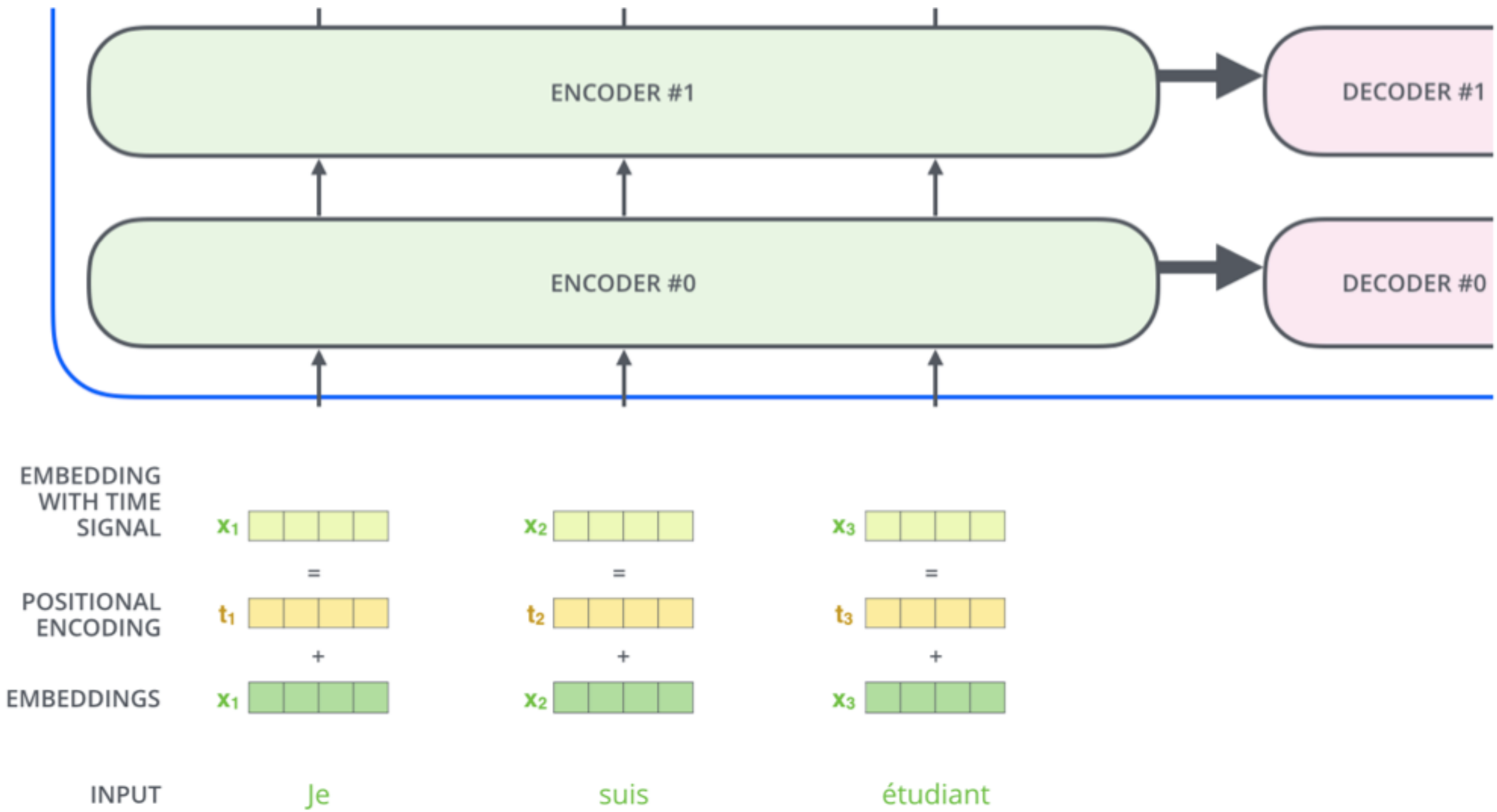
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Self-Attention Visualization



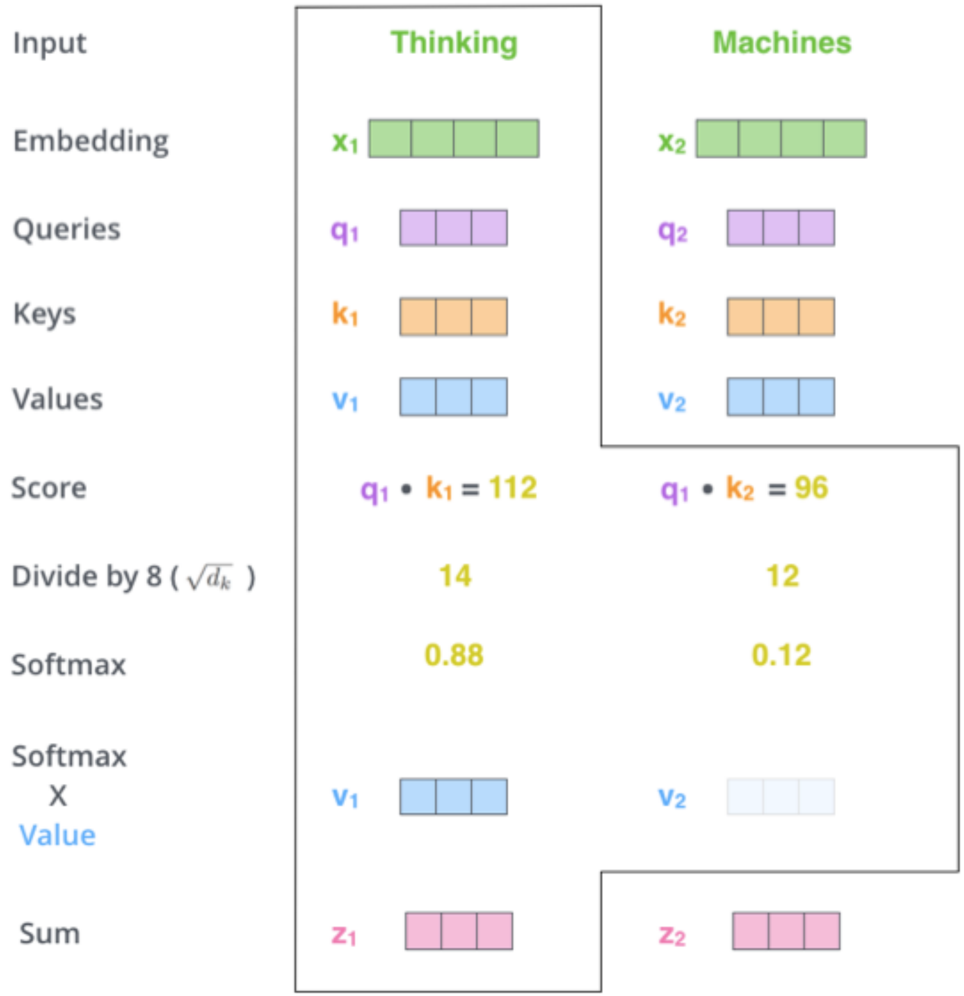
Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer Positional Encoding Vectors



Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer
Self-Attention Softmax Output

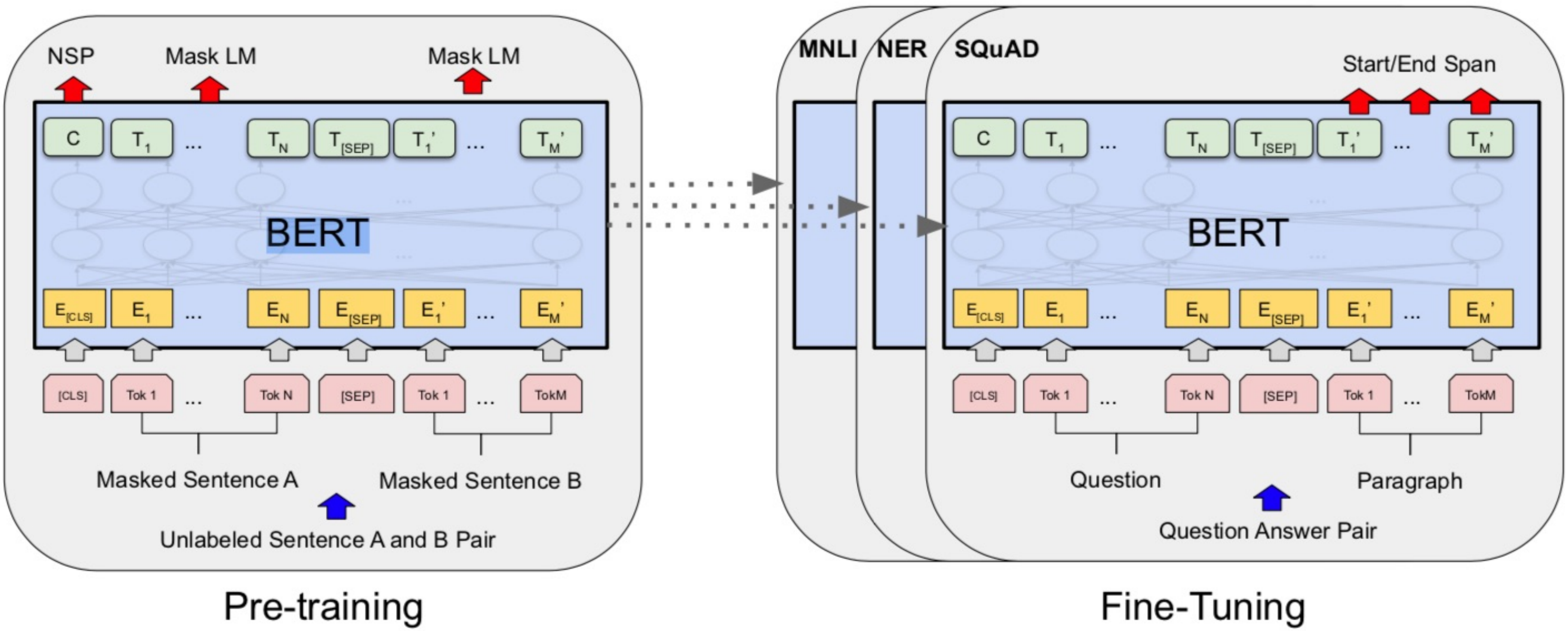


Source: Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

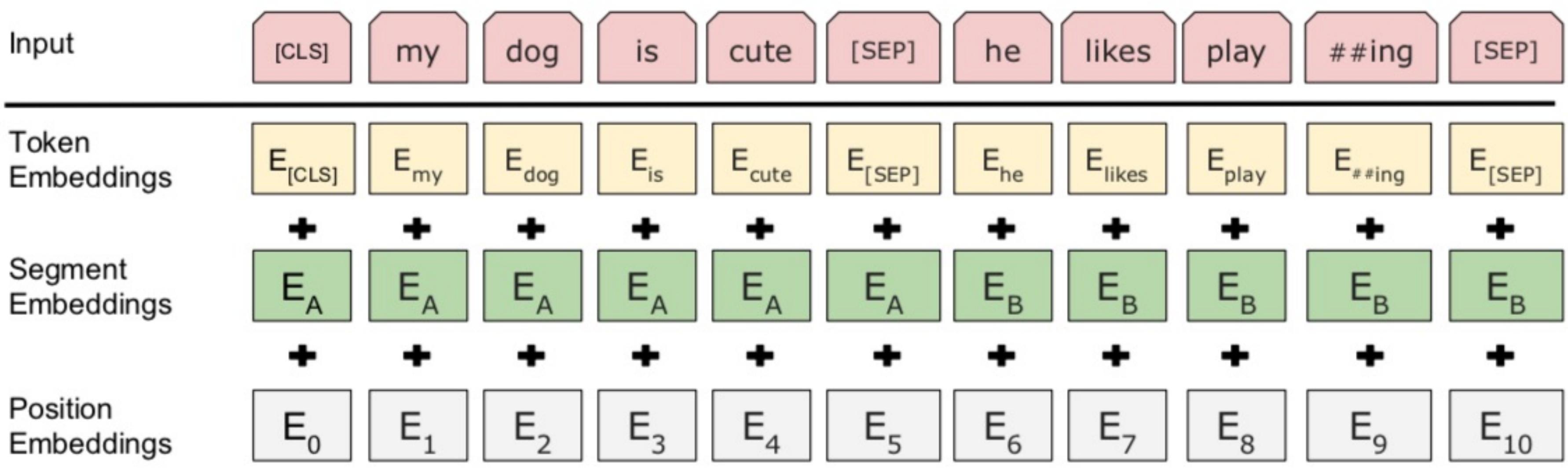


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

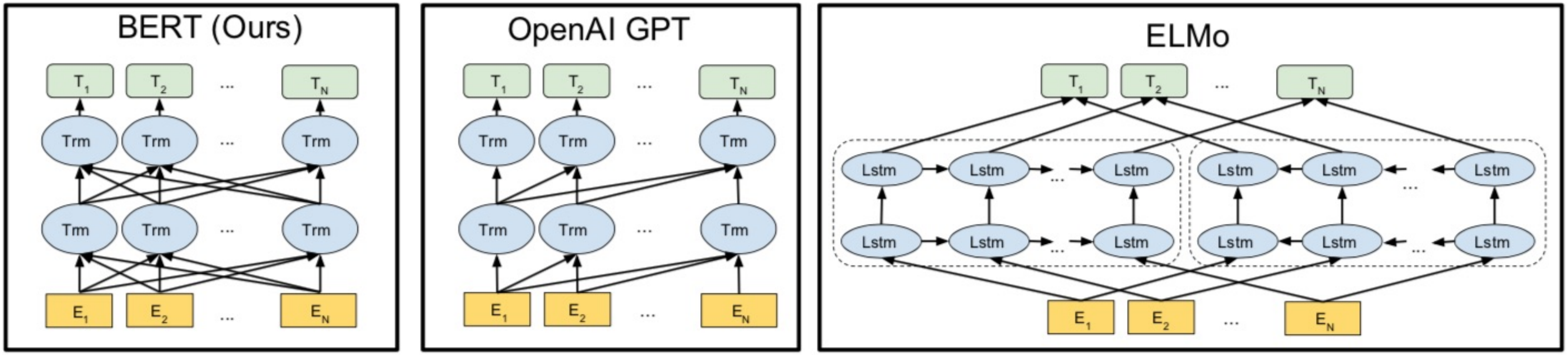
BERT (Bidirectional Encoder Representations from Transformers)

BERT input representation



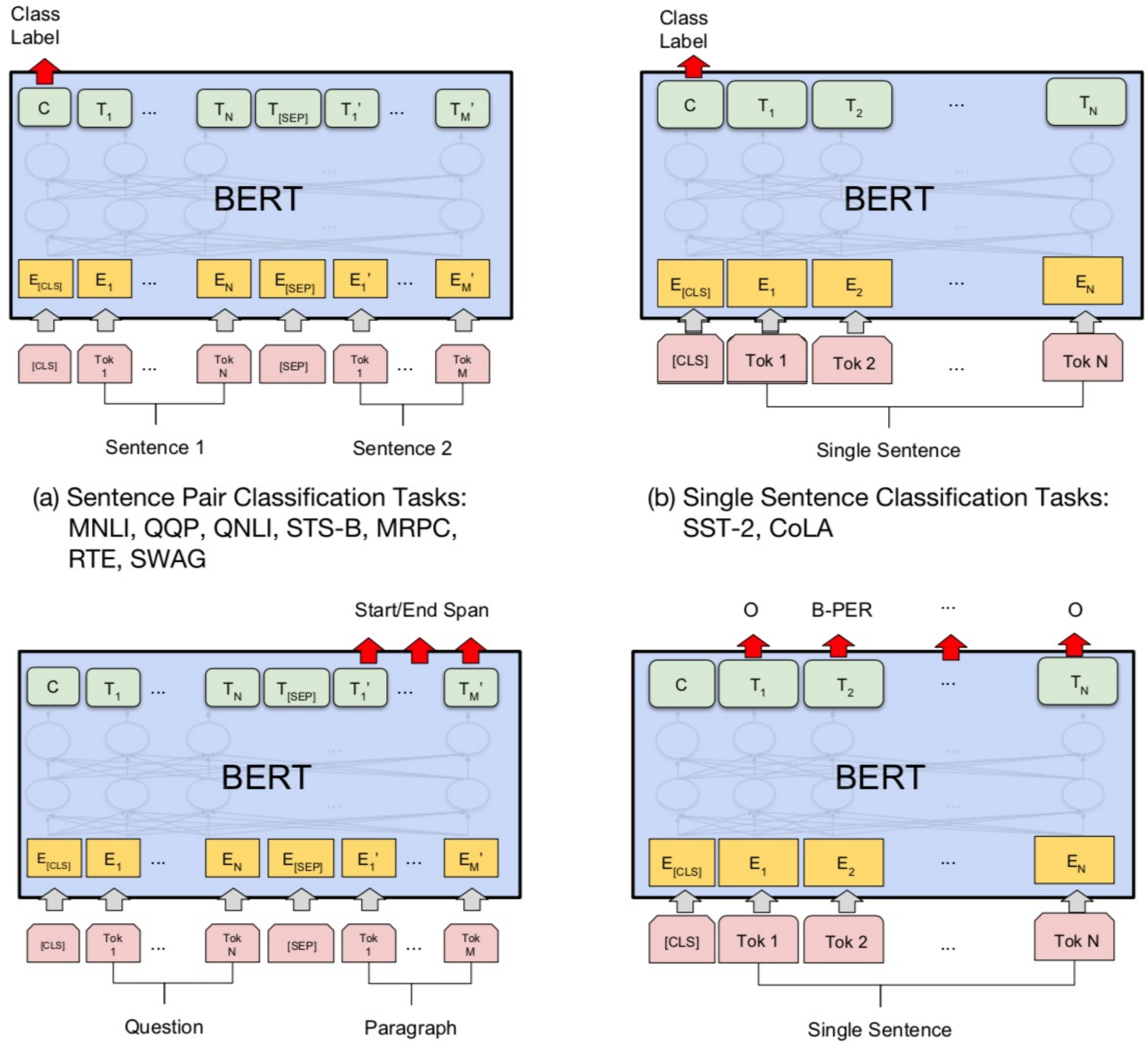
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

BERT, OpenAI GPT, ELMo



Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Different Tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

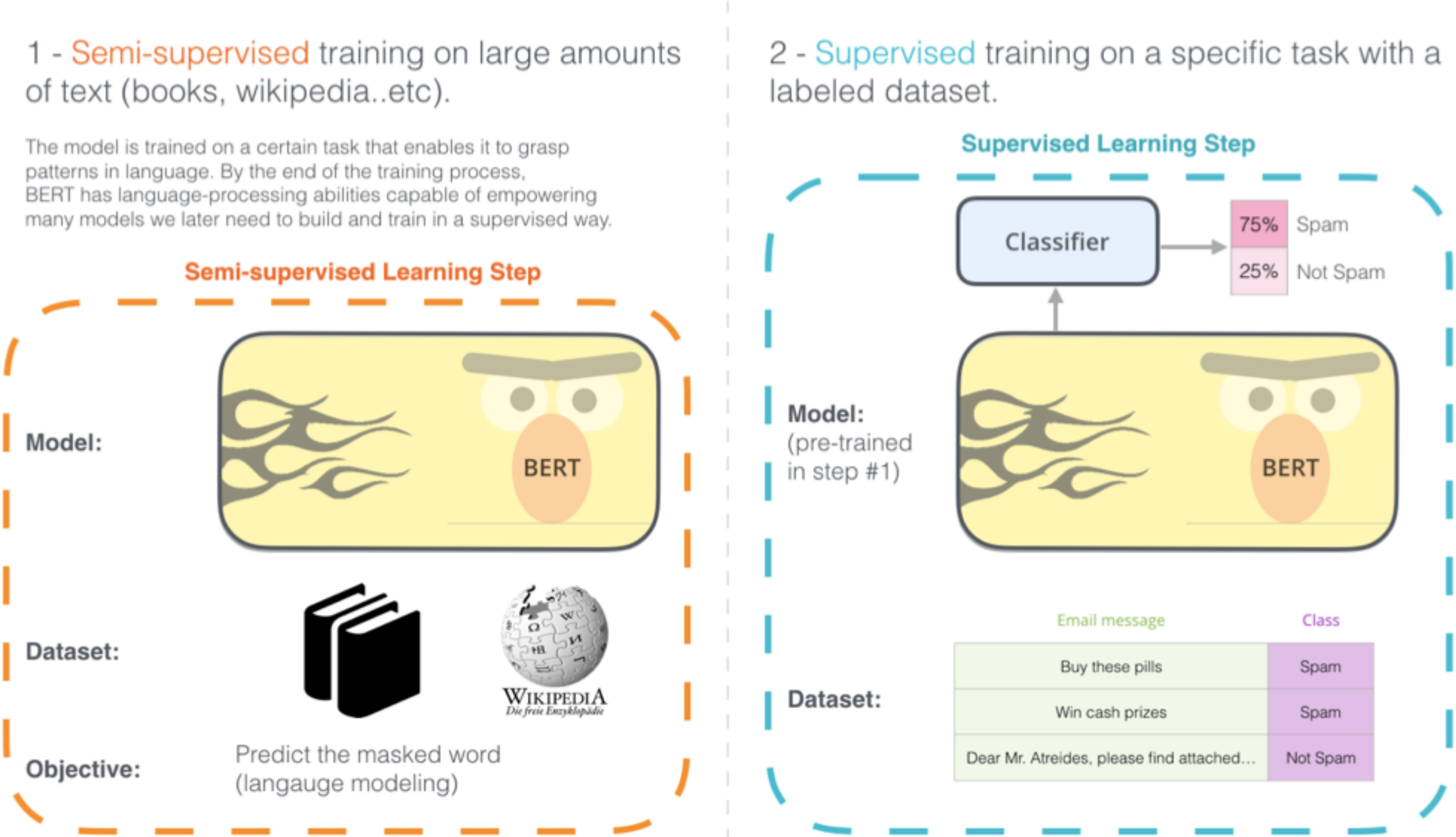
(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

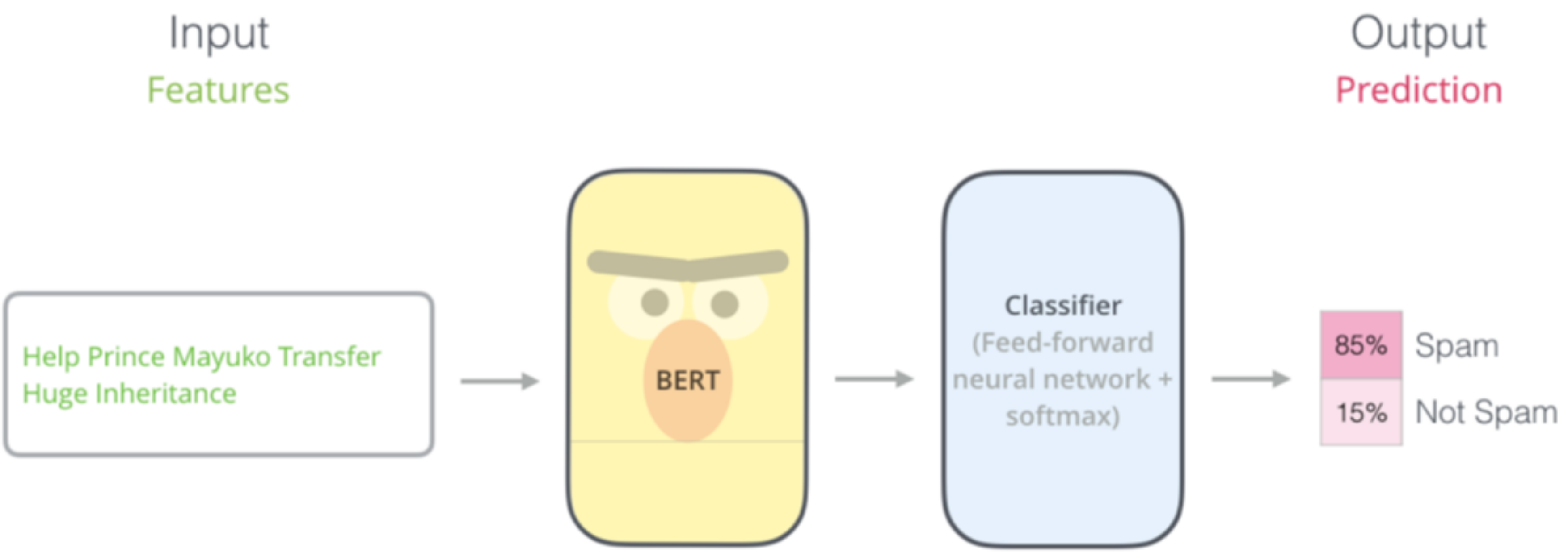
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Illustrated BERT



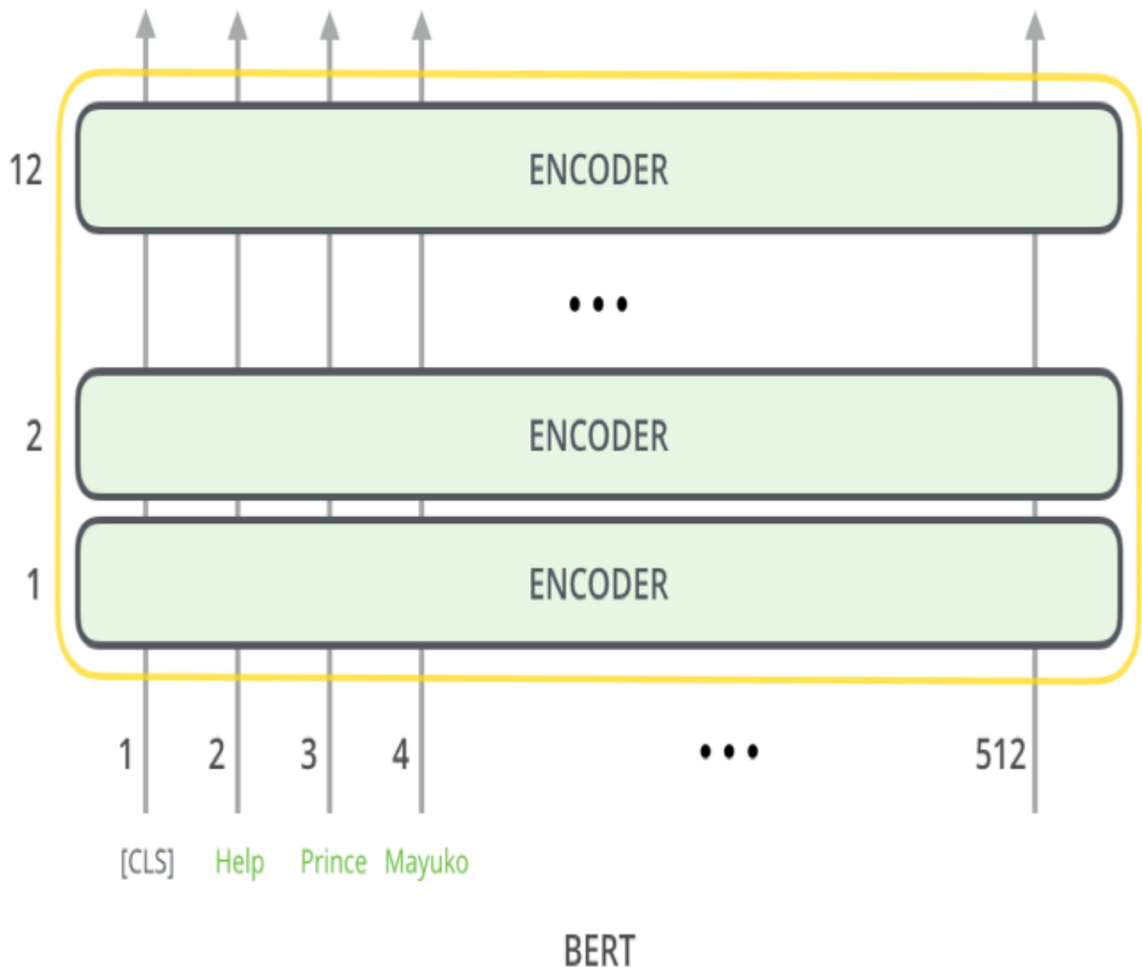
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

BERT Classification Input Output



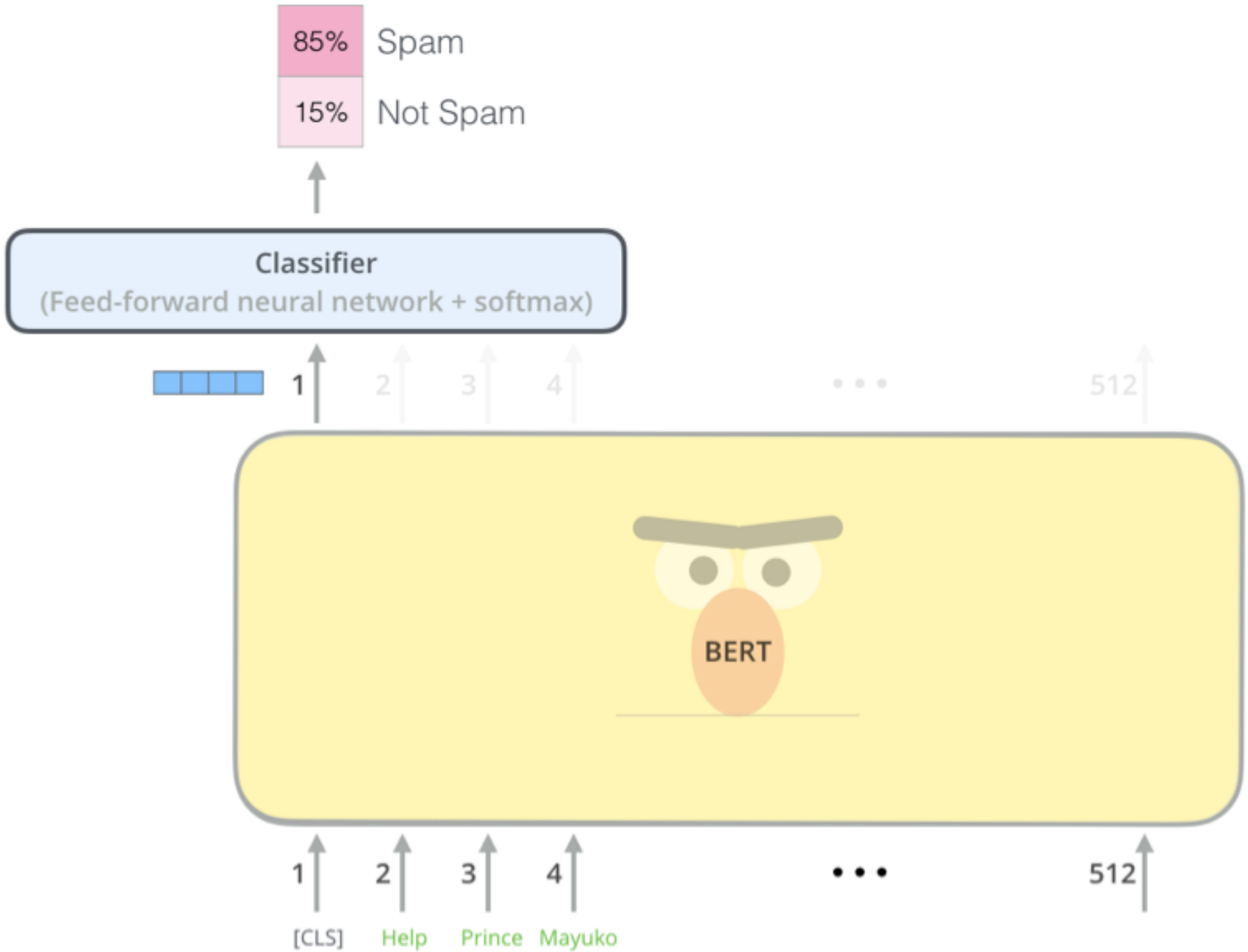
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

BERT Encoder Input



Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

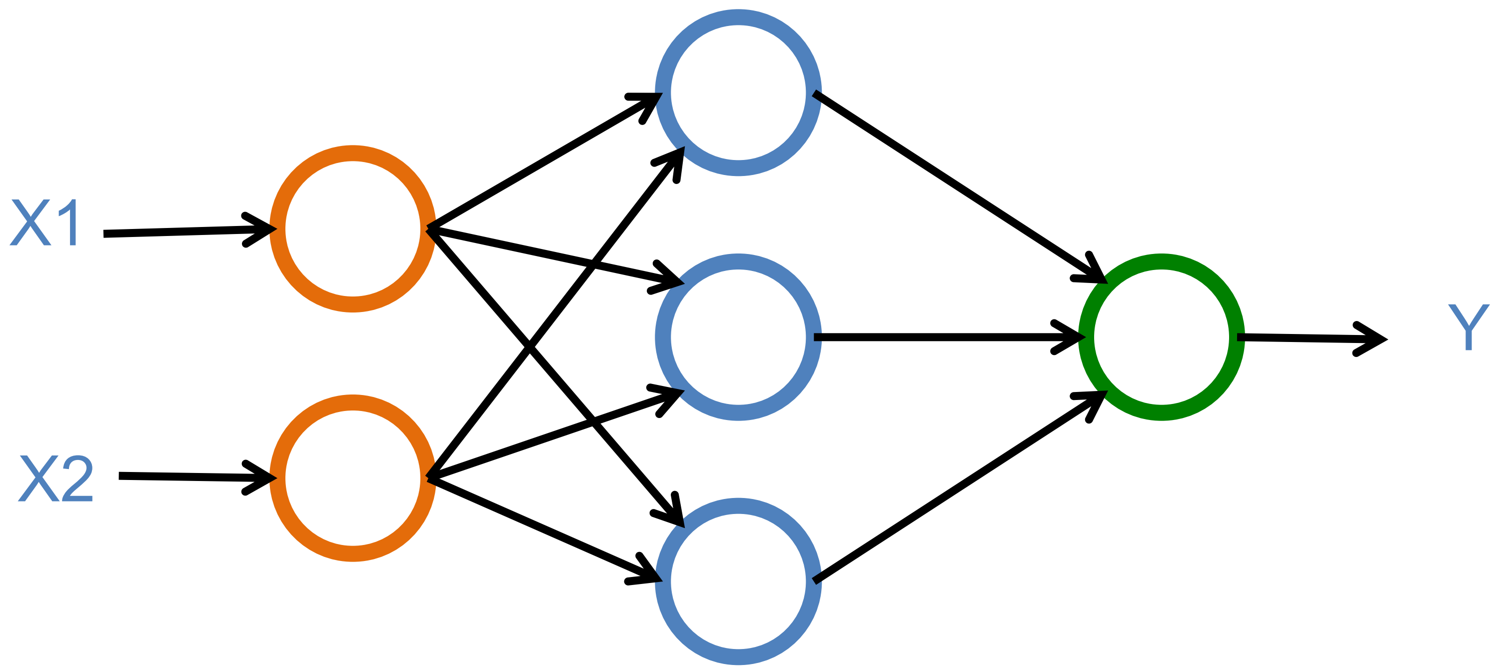
BERT Classifier



Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/

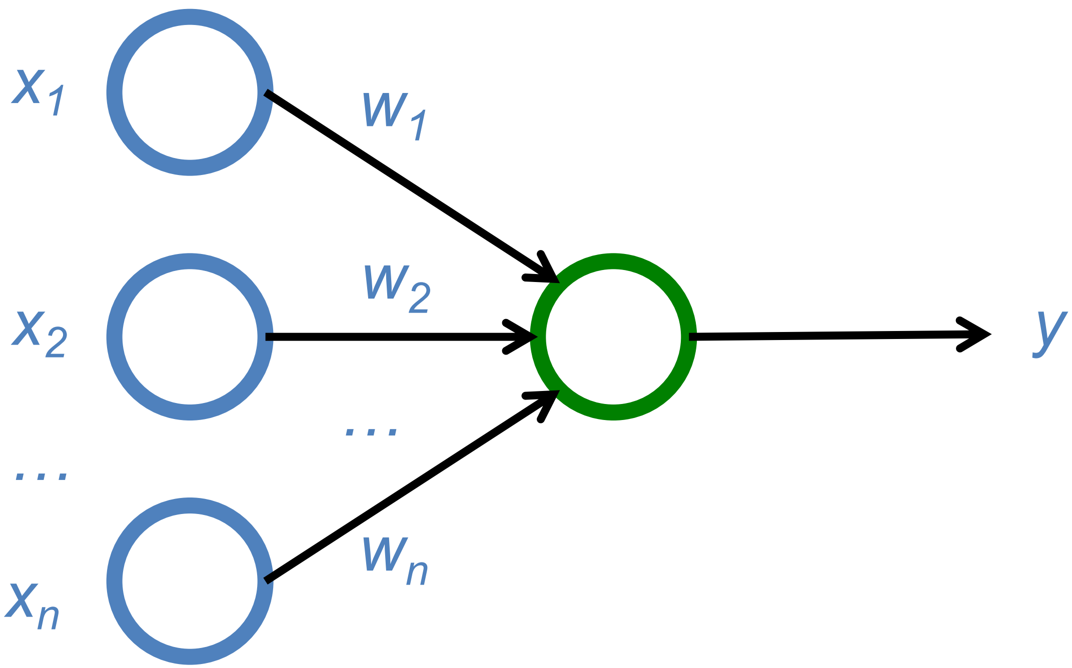
Neural Networks

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



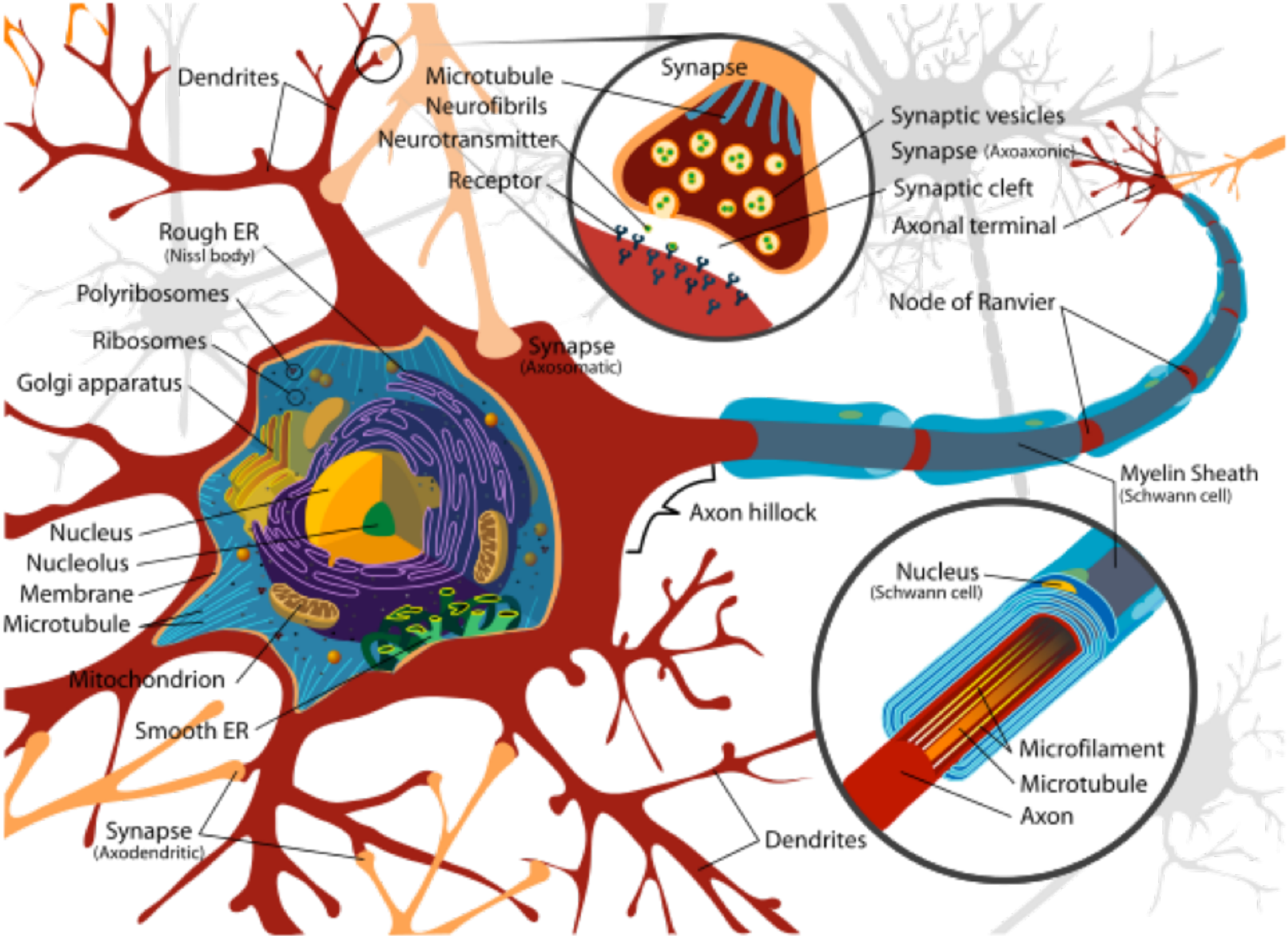
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

The Neuron



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

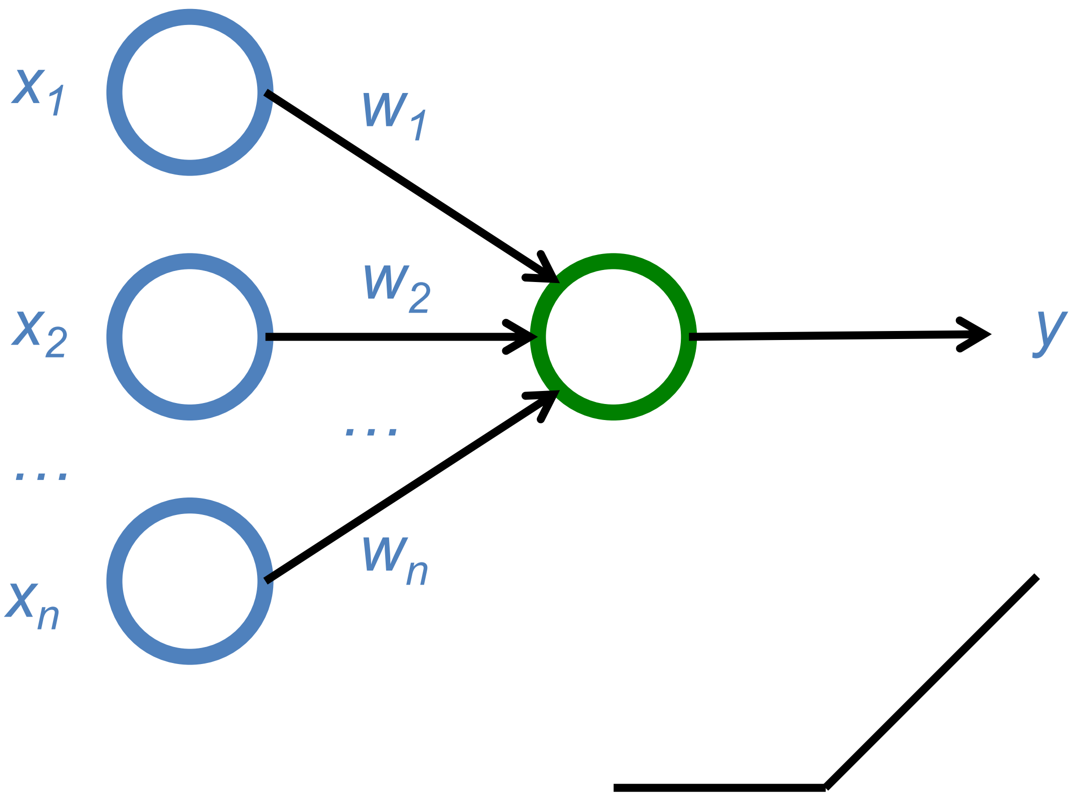
Neuron and Synapse



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

The Neuron

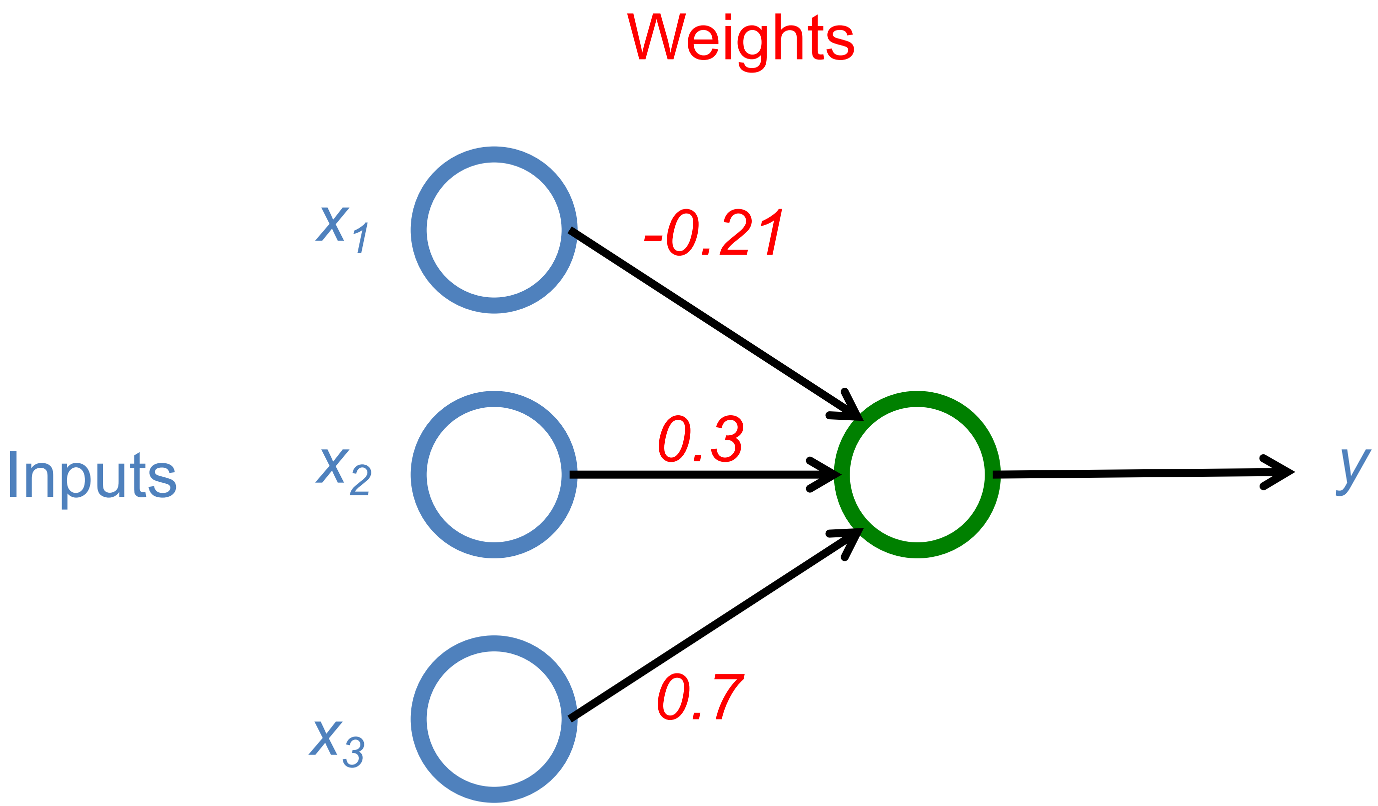
$$y = F\left(\sum_i w_i x_i\right)$$



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

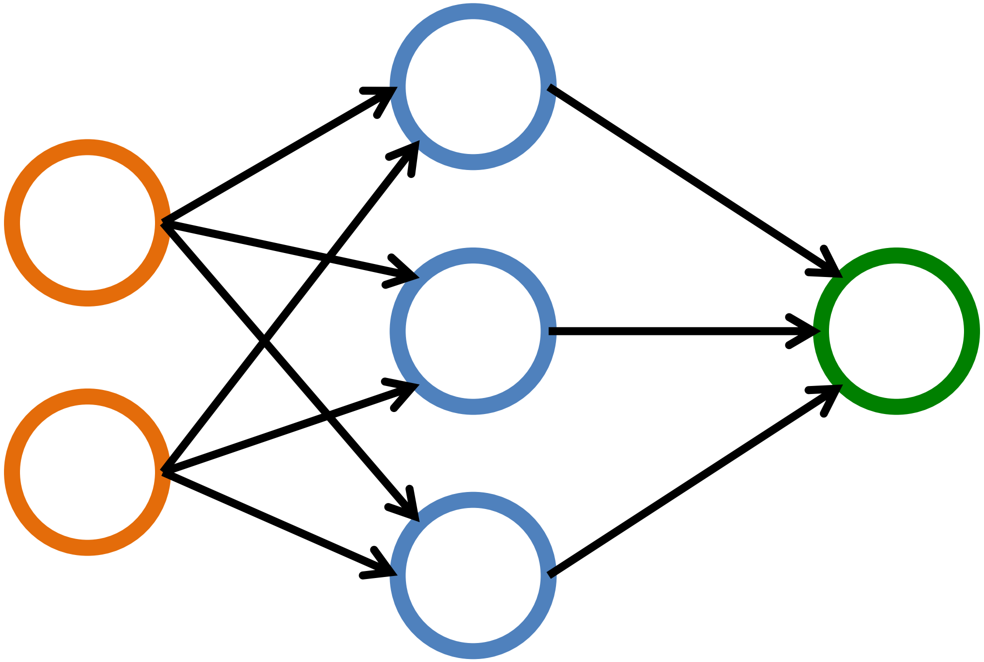
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

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

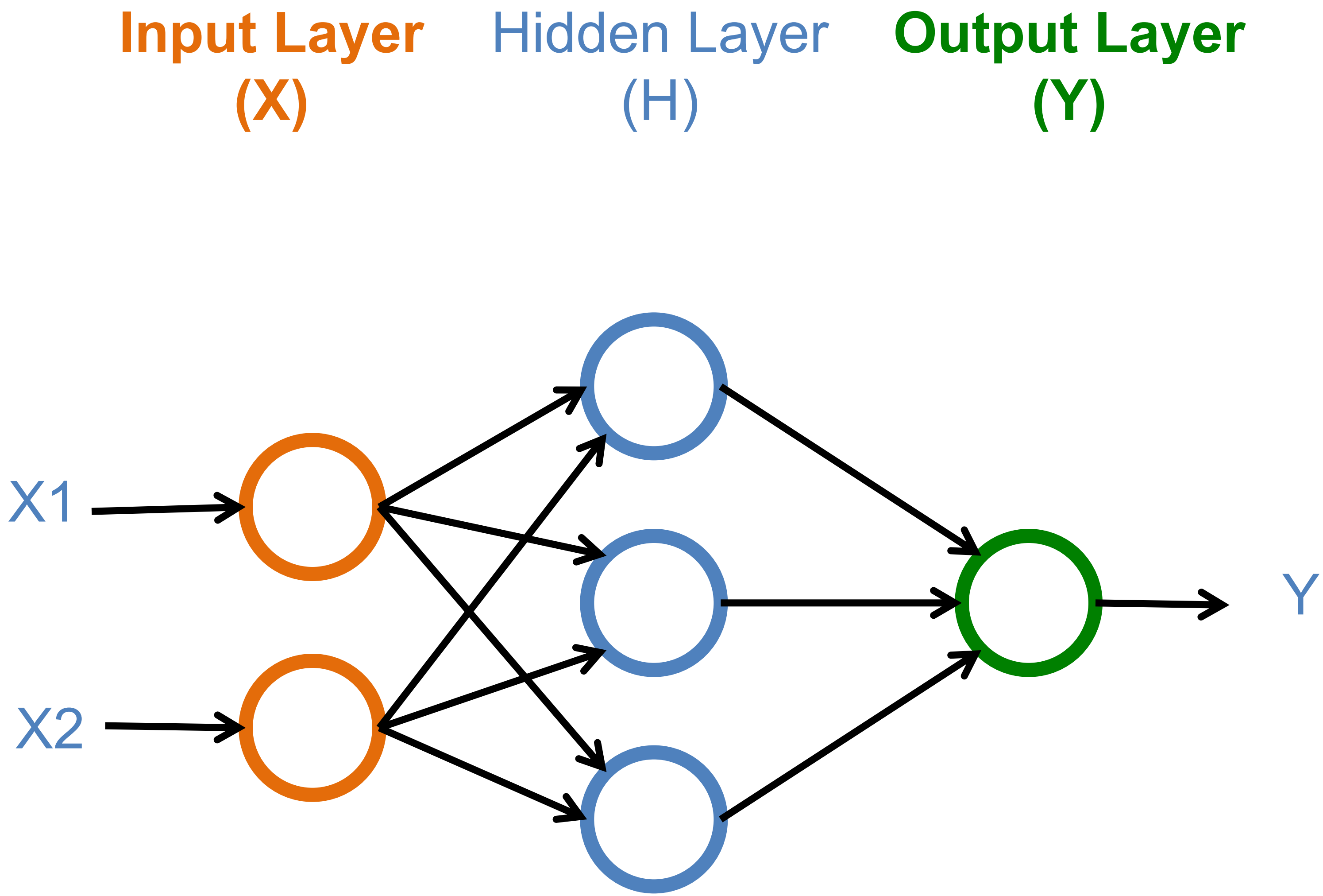


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Neural Networks



Neural Networks



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

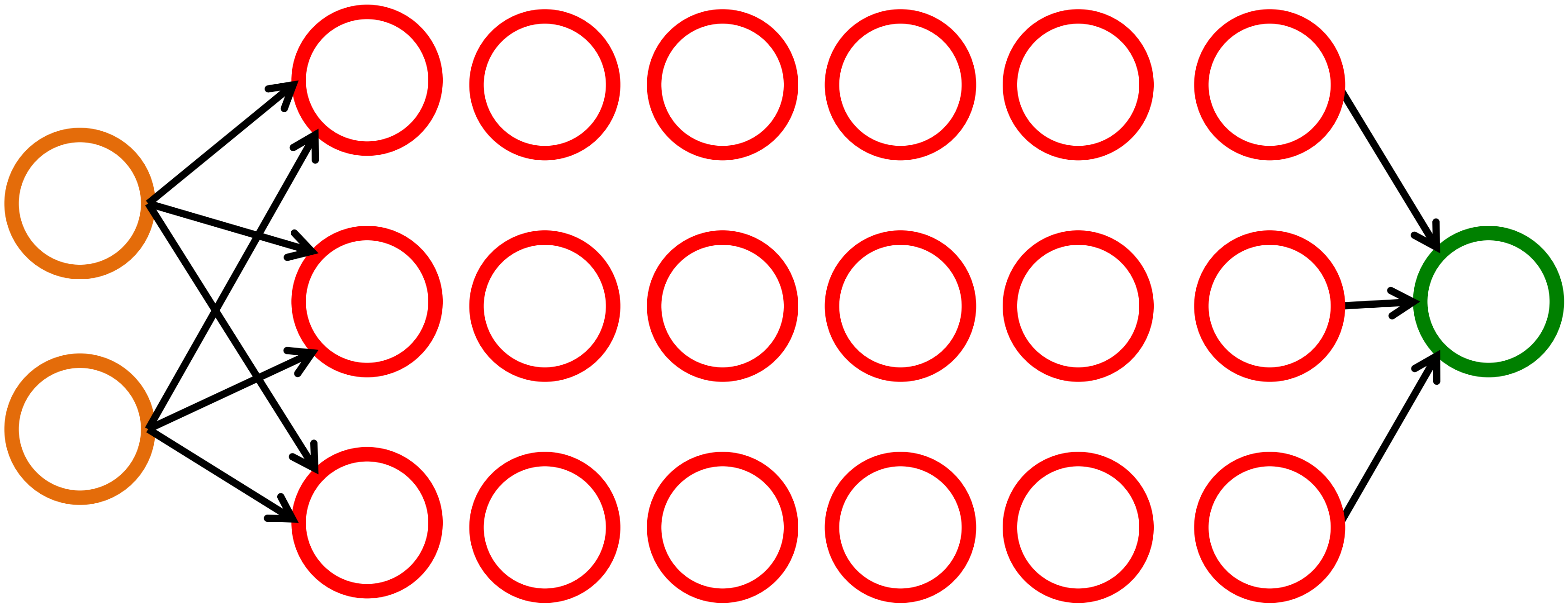
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

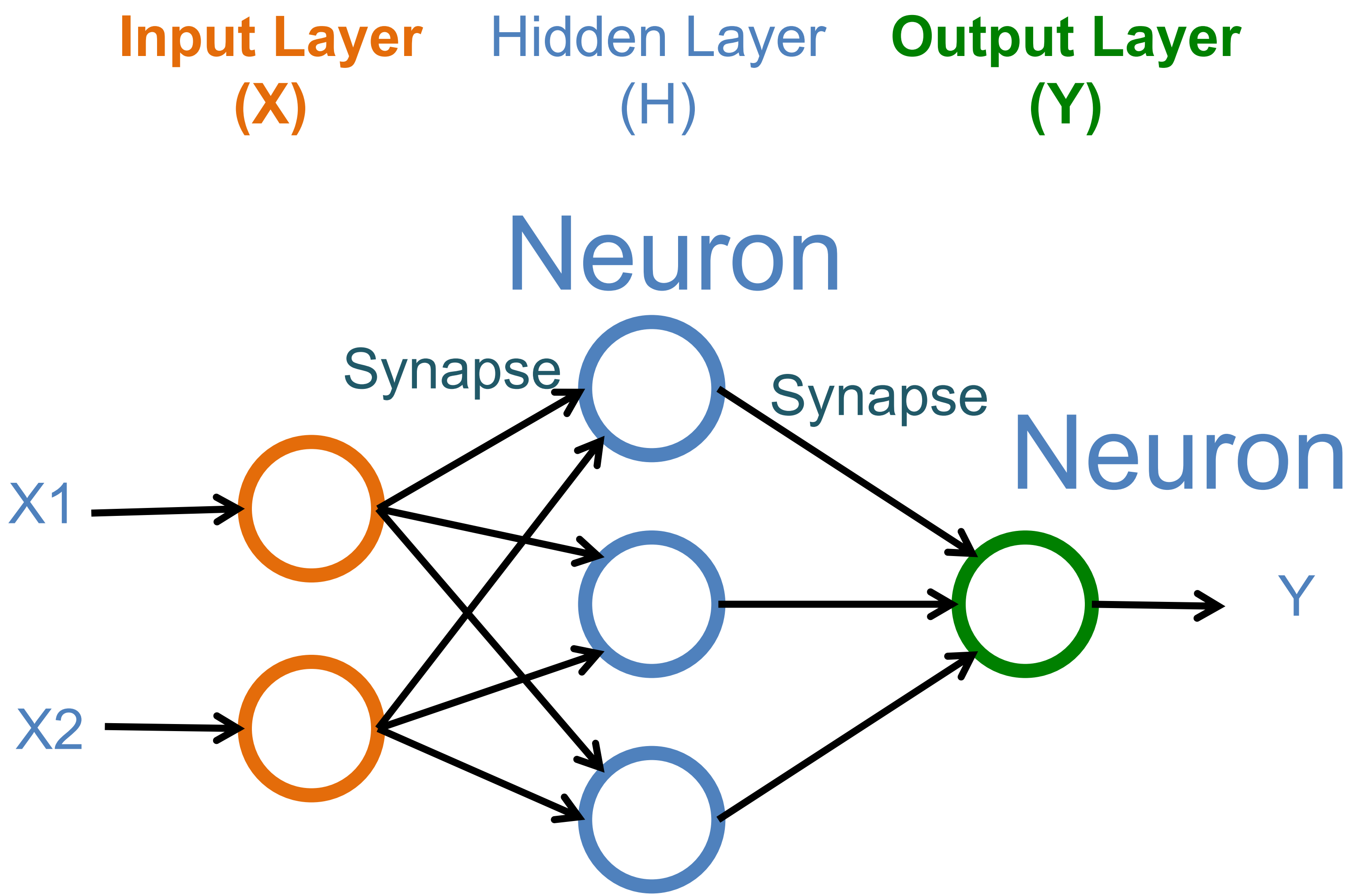
Output Layer
(Y)

Deep Neural Networks
Deep Learning



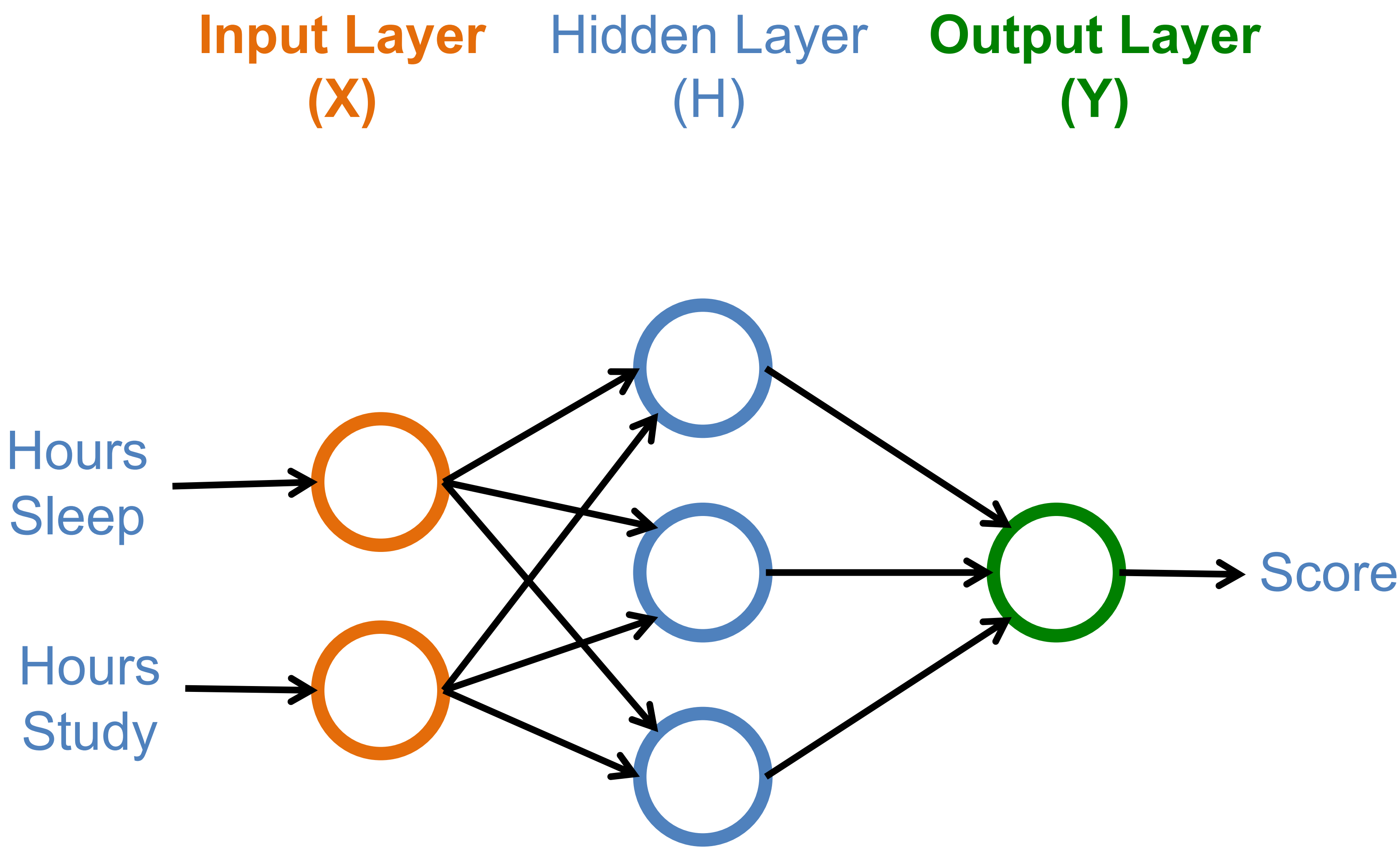
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



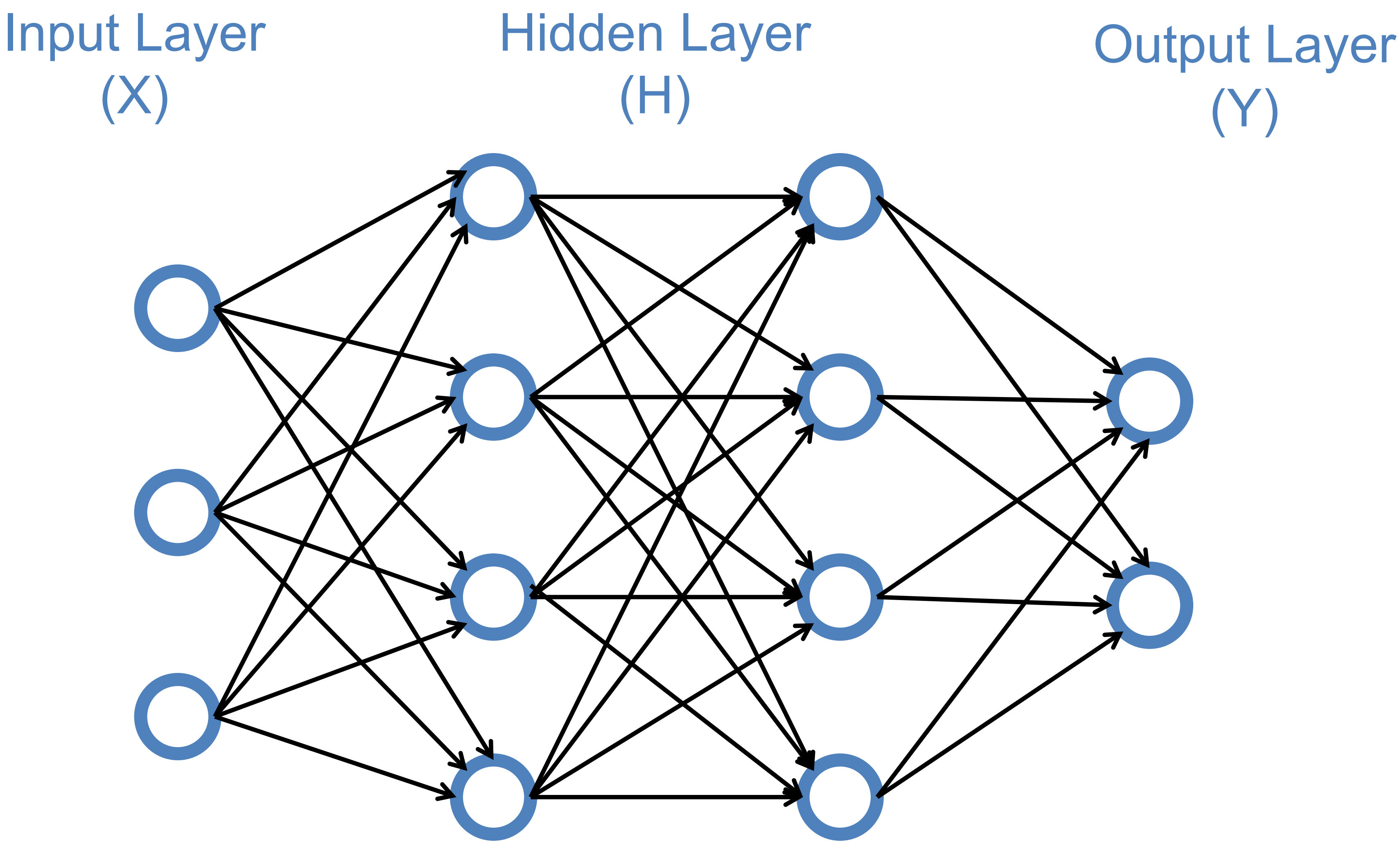
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks



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

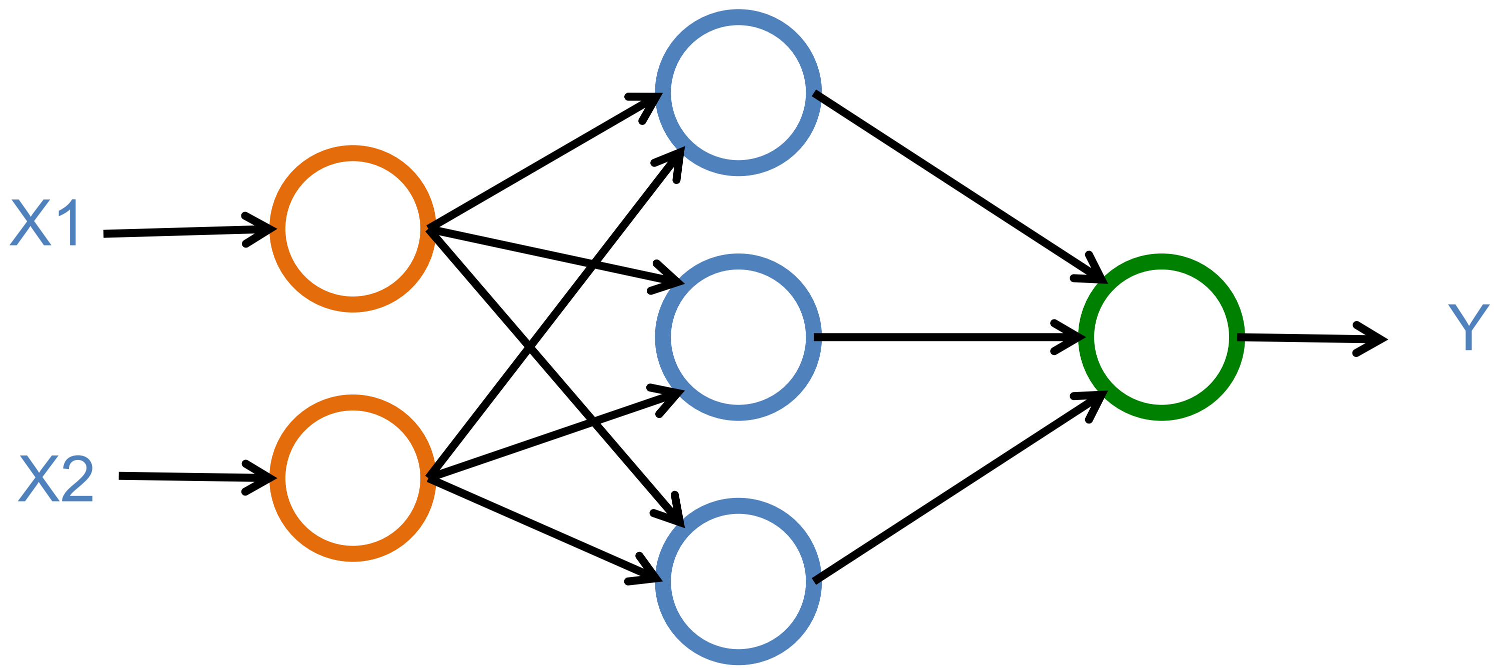
Neural Networks



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

Neural Networks

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



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

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

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

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

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

$$Y = W X + b$$

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

Output input

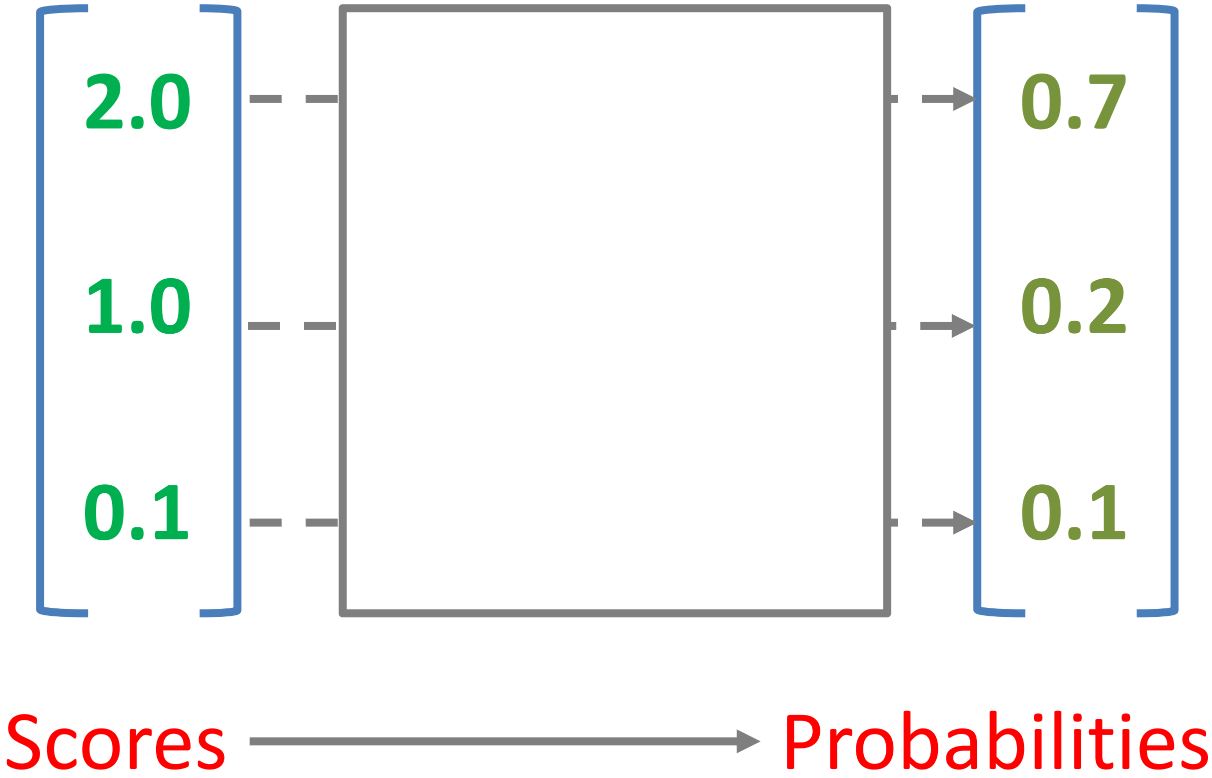
$$Y = W\,X + b$$

Weights bias

Trained

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

$$WX + b = Y$$



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

SoftMAX

$$\mathbf{W}\ \mathbf{X} + \mathbf{b} = \mathbf{Y} \quad \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \longrightarrow \boxed{S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}} \longrightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Logits

Scores $\longrightarrow$ Probabilities

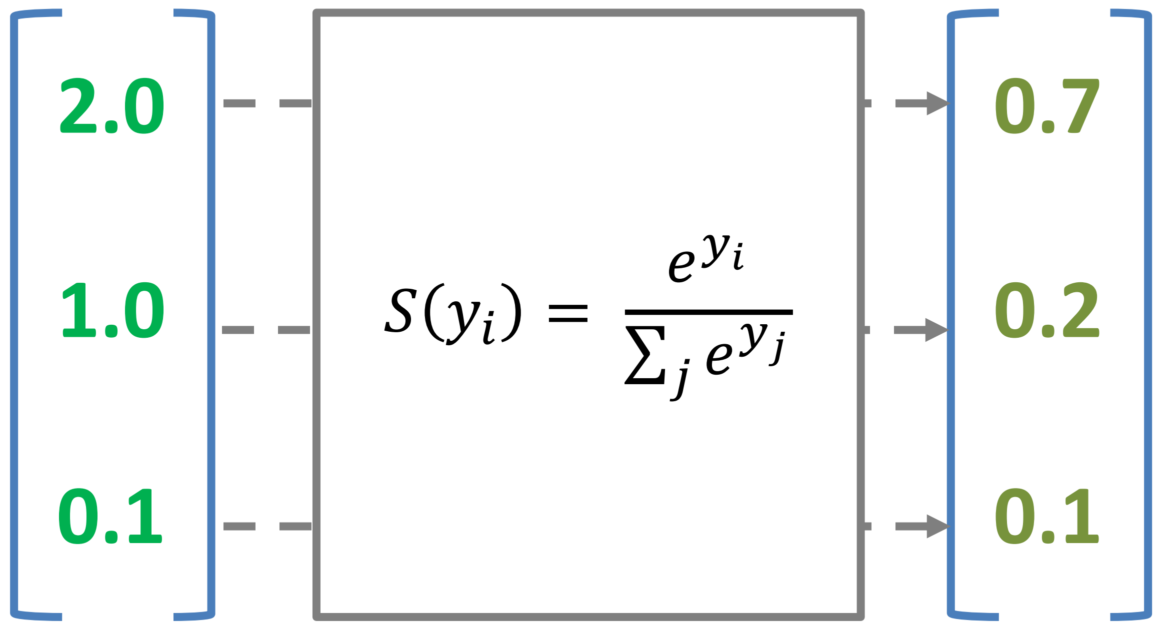
Source: https://www.youtube.com/watch?v=G8eNWzxOgqE

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

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

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

W X + b = Y



Logits

Scores → Probabilities

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

Training a Network

=

Minimize the Cost Function

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

Training a Network

=

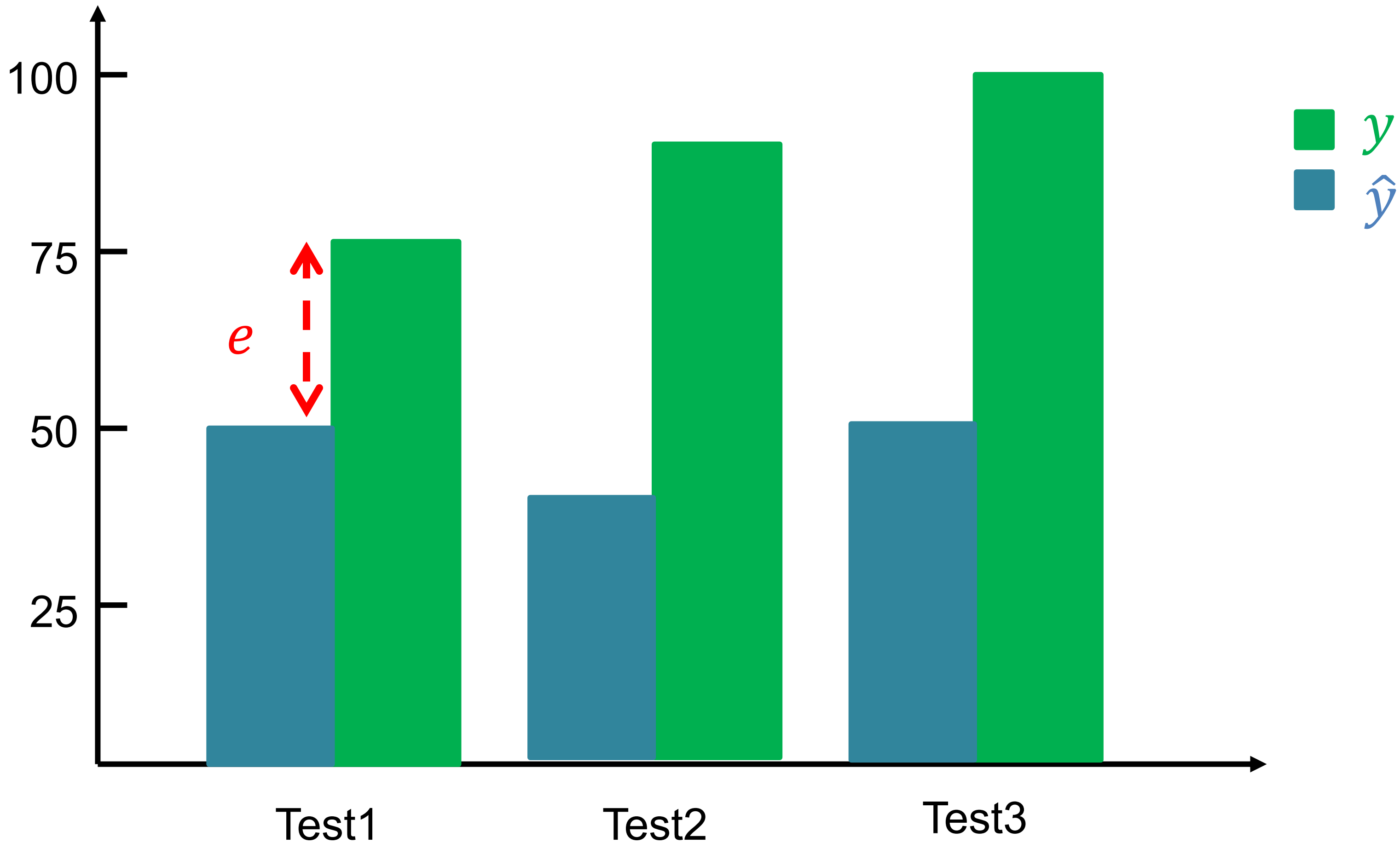
Minimize the Cost Function

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

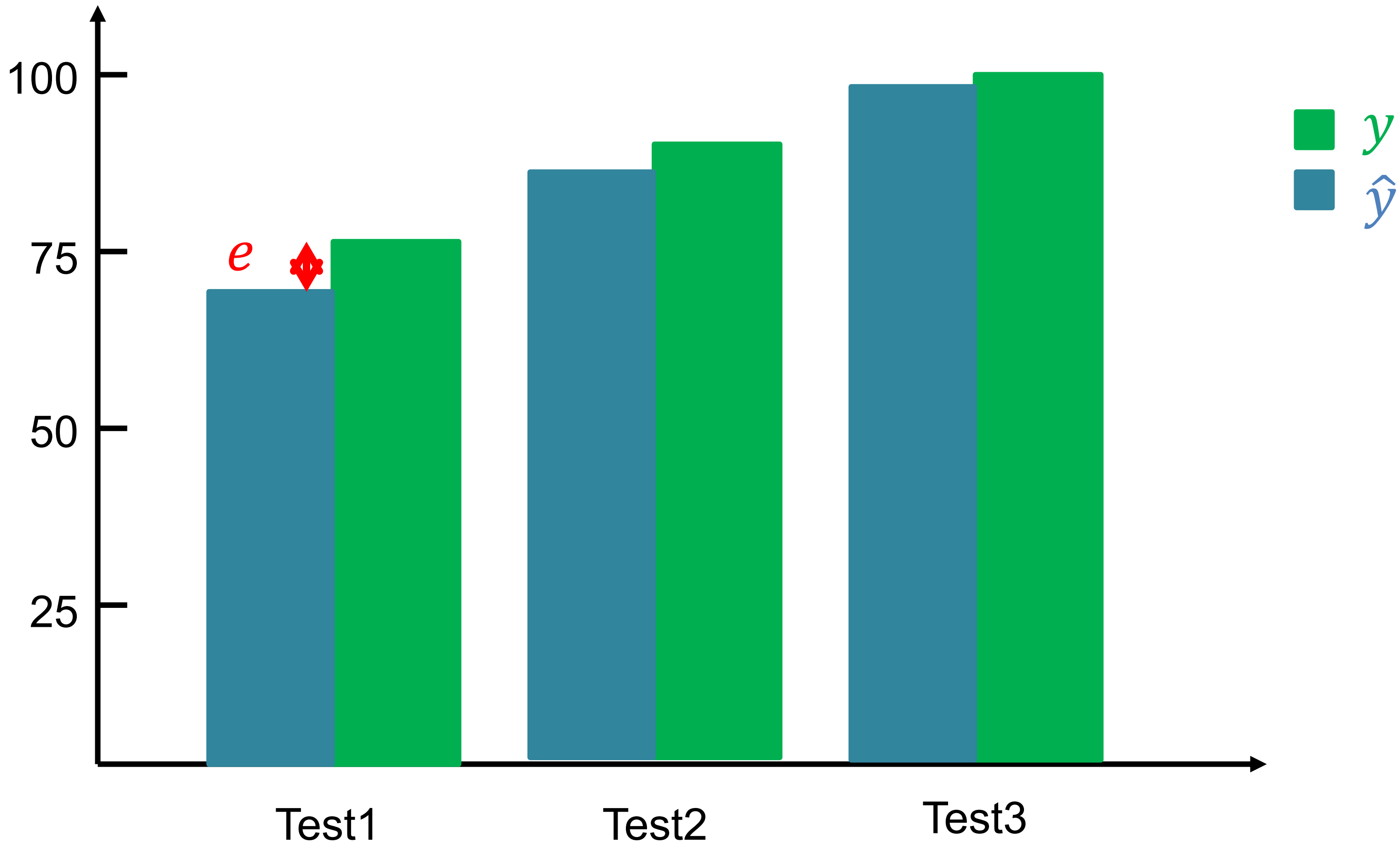
Error = Predict Y - Actual Y

Error : Cost : Loss

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

Error = Predict Y - Actual Y

Error : Cost : Loss



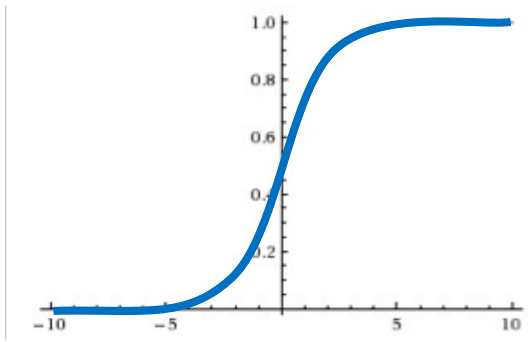
Activation Functions

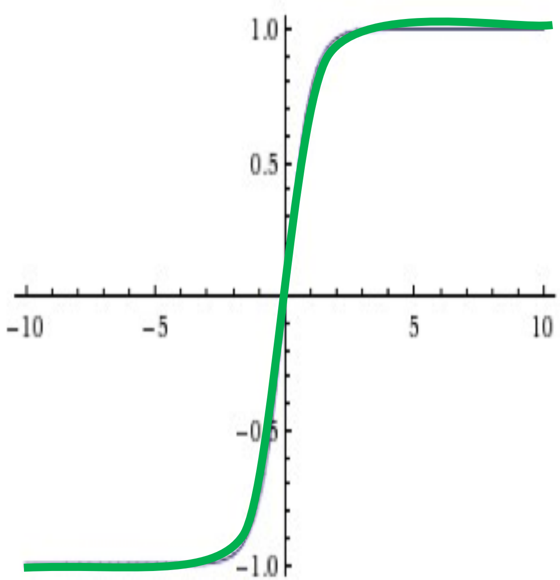
Activation Functions

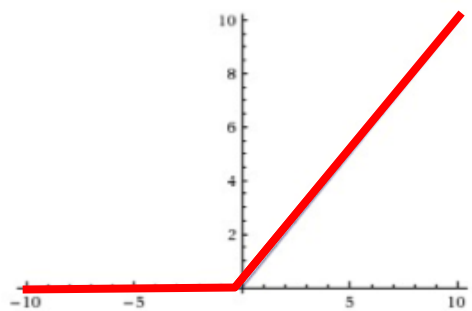
Sigmoid

TanH

ReLU
(Rectified Linear Unit)







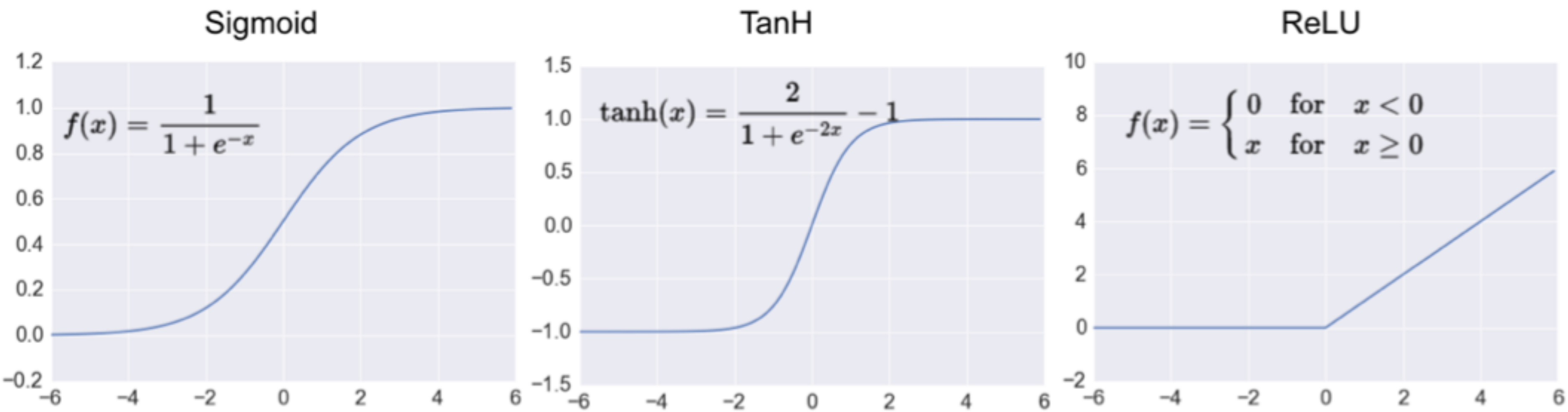
[0, 1]

[-1, 1]

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

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

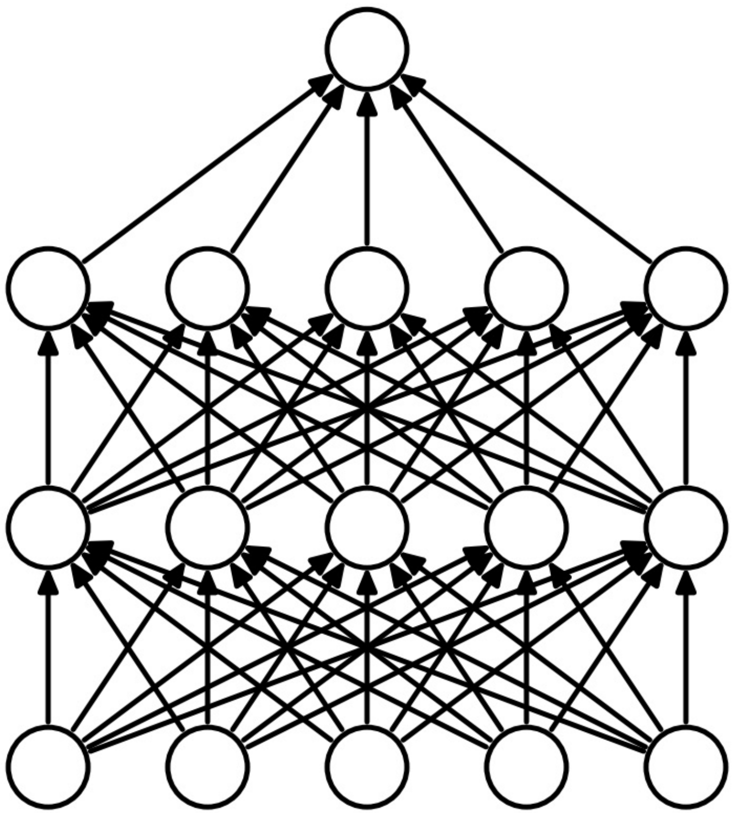
Multiple Classification: 10 Class

Activation Function:
SoftMAX

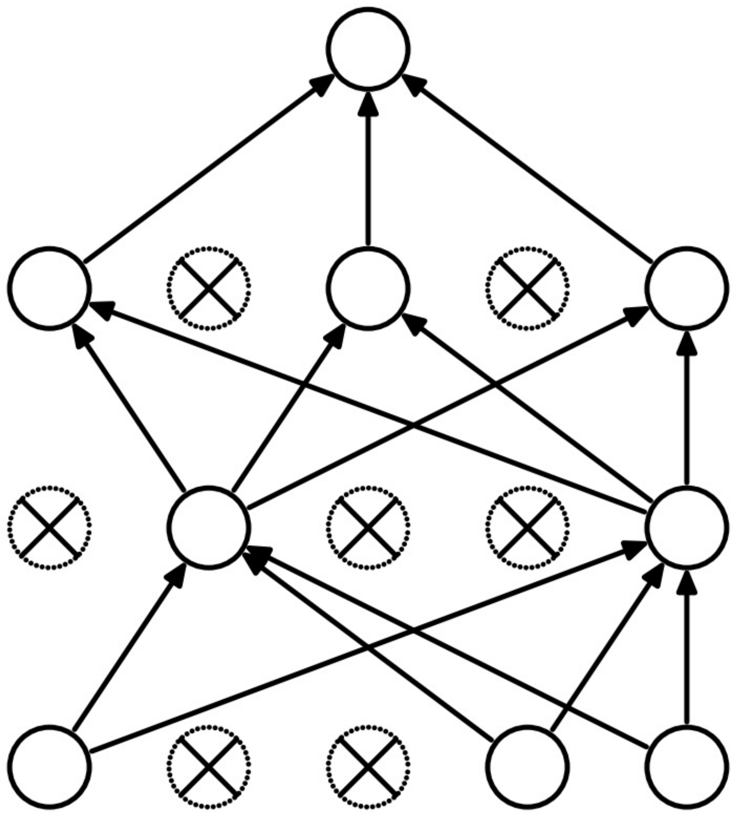
Loss Function:
Categorical Cross-Entropy

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net



(b) After applying dropout.

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

Learning Algorithm

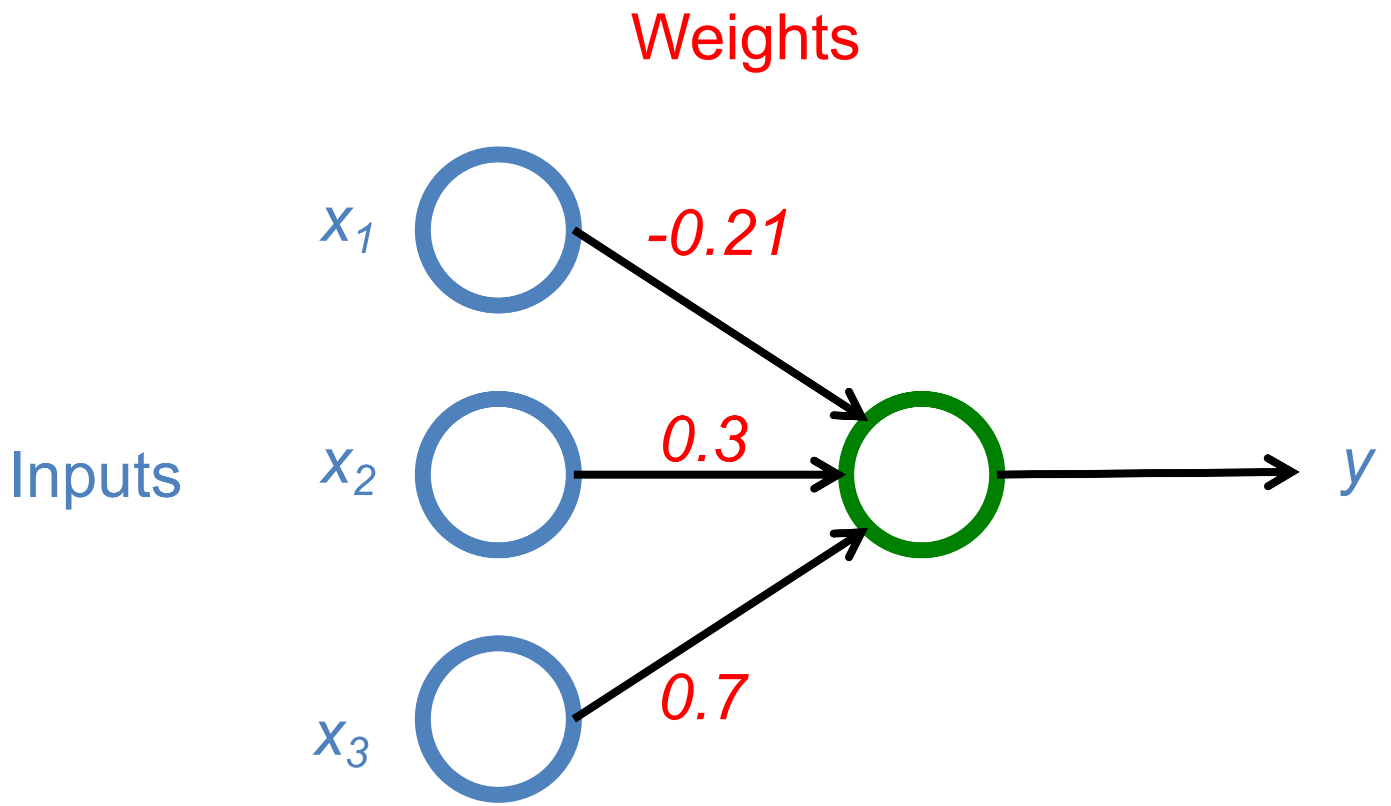
While not done:

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

Run neural network on "input"

Adjust weights on edges to make output closer to "label"

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

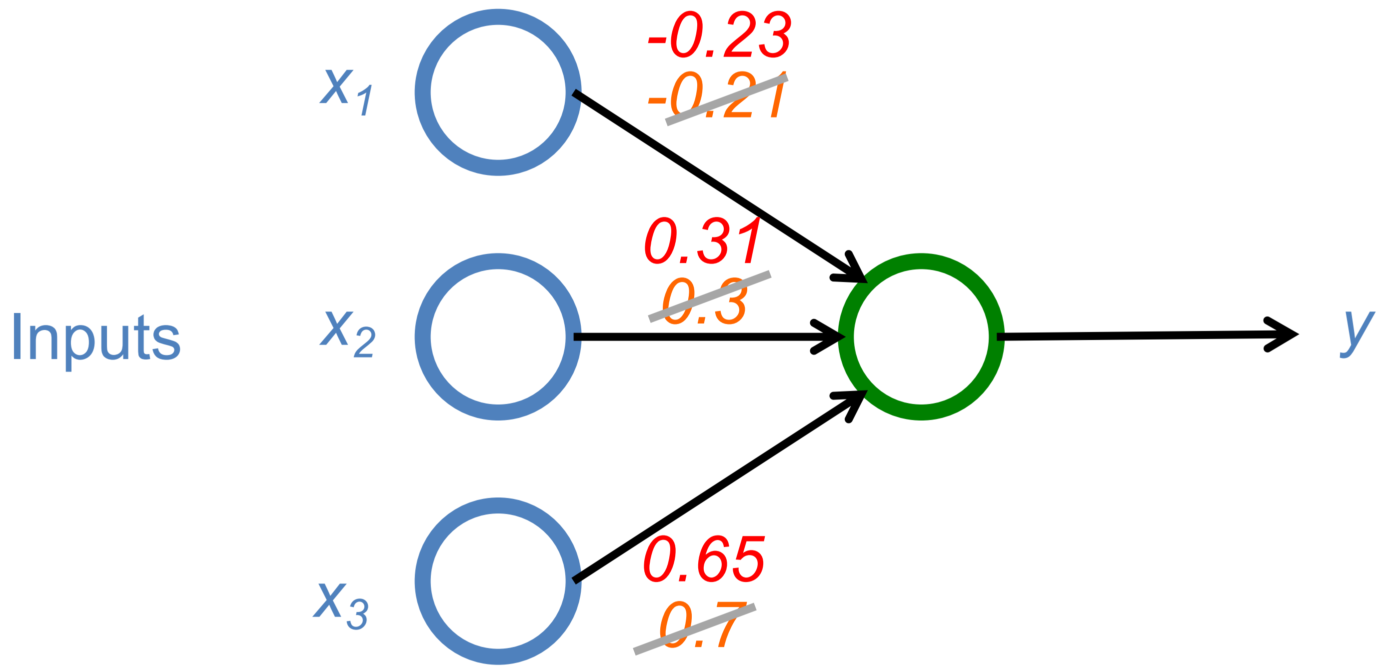


Next time:

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

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

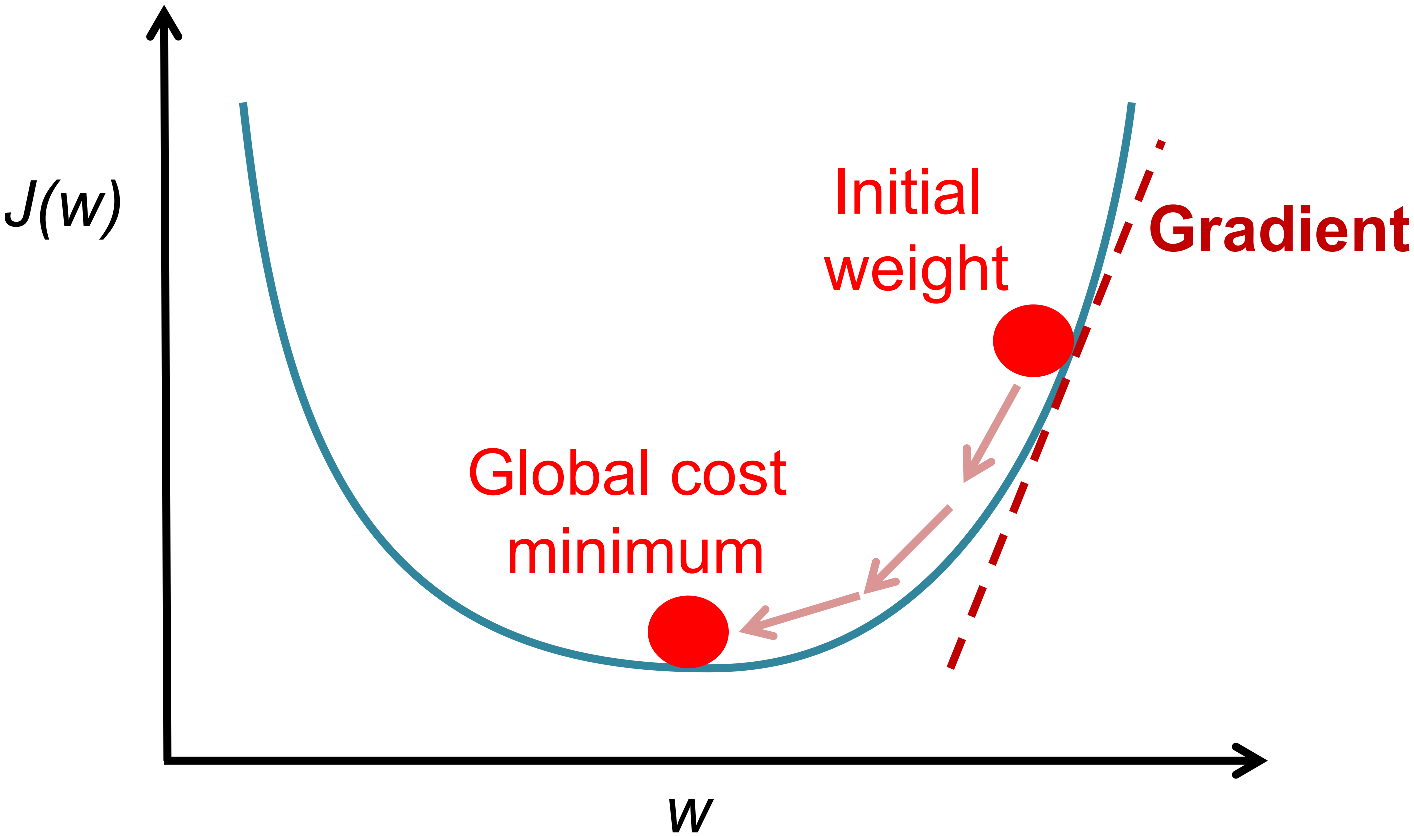
Weights

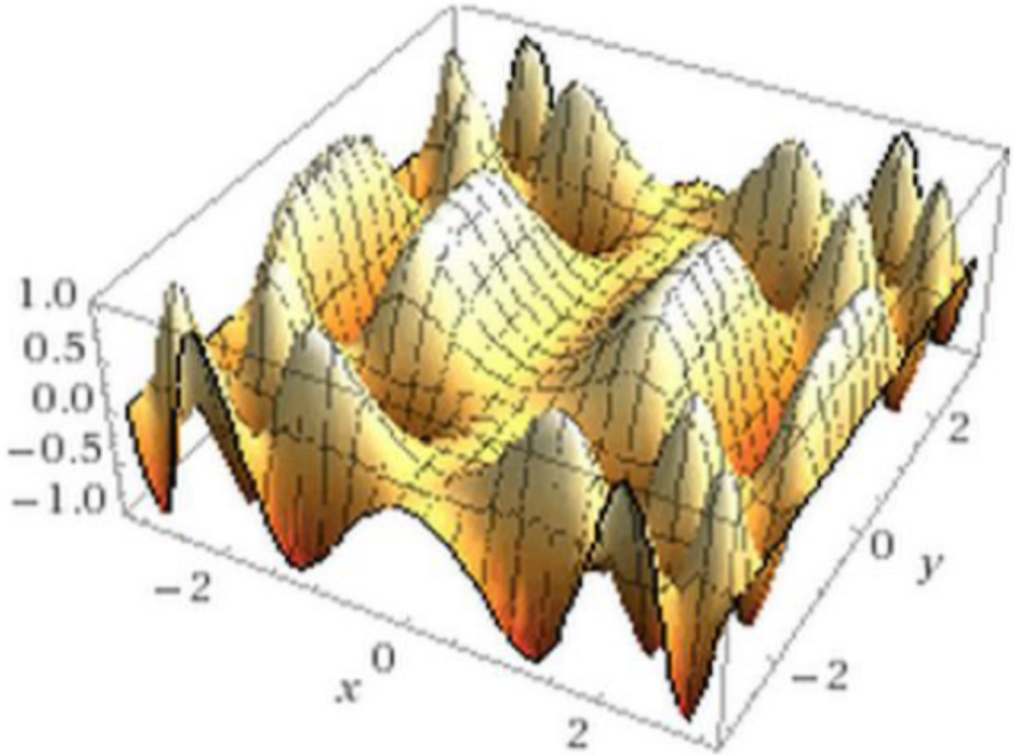


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Optimizer:

Stochastic Gradient Descent (SGD)

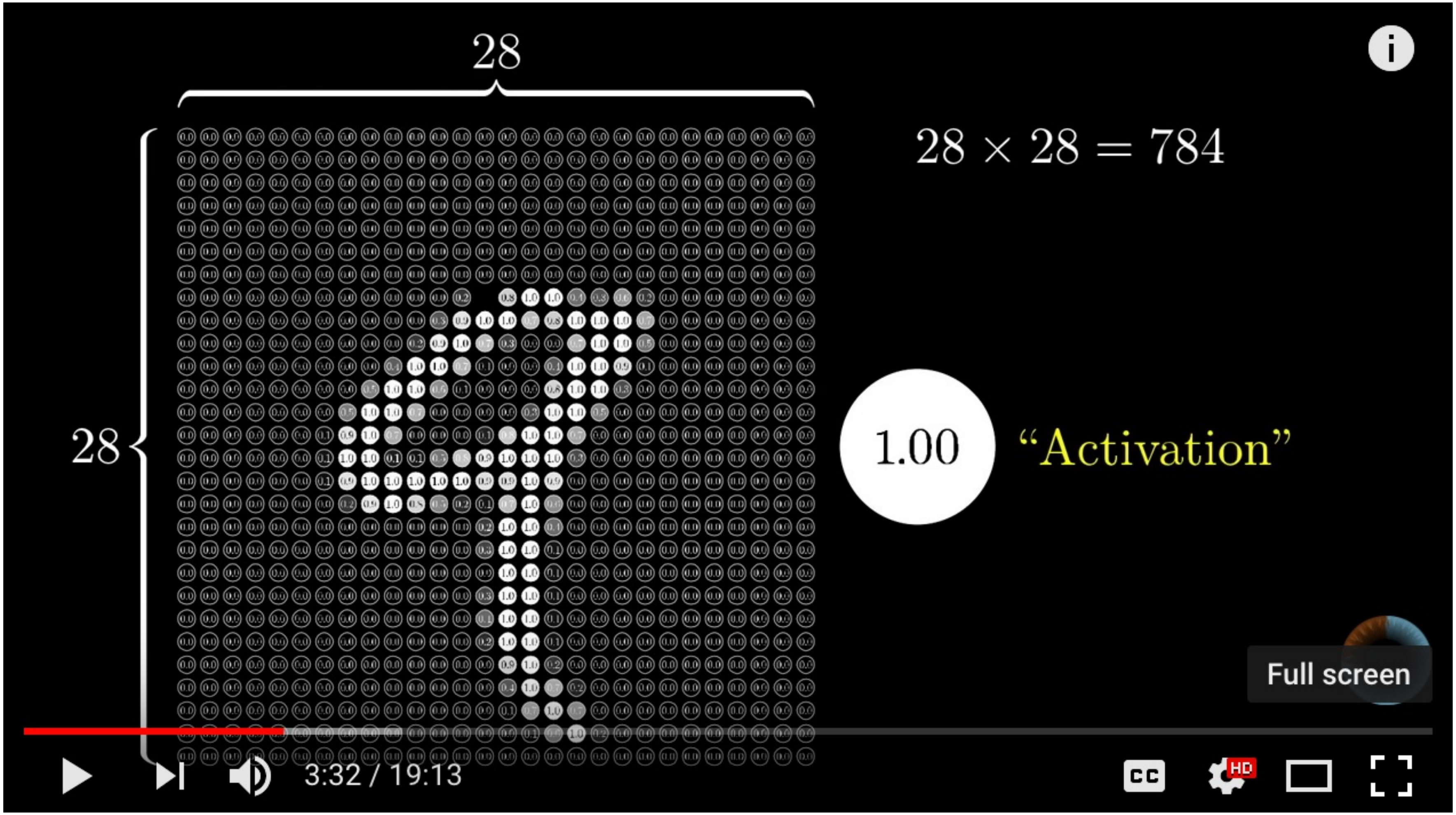




This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

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

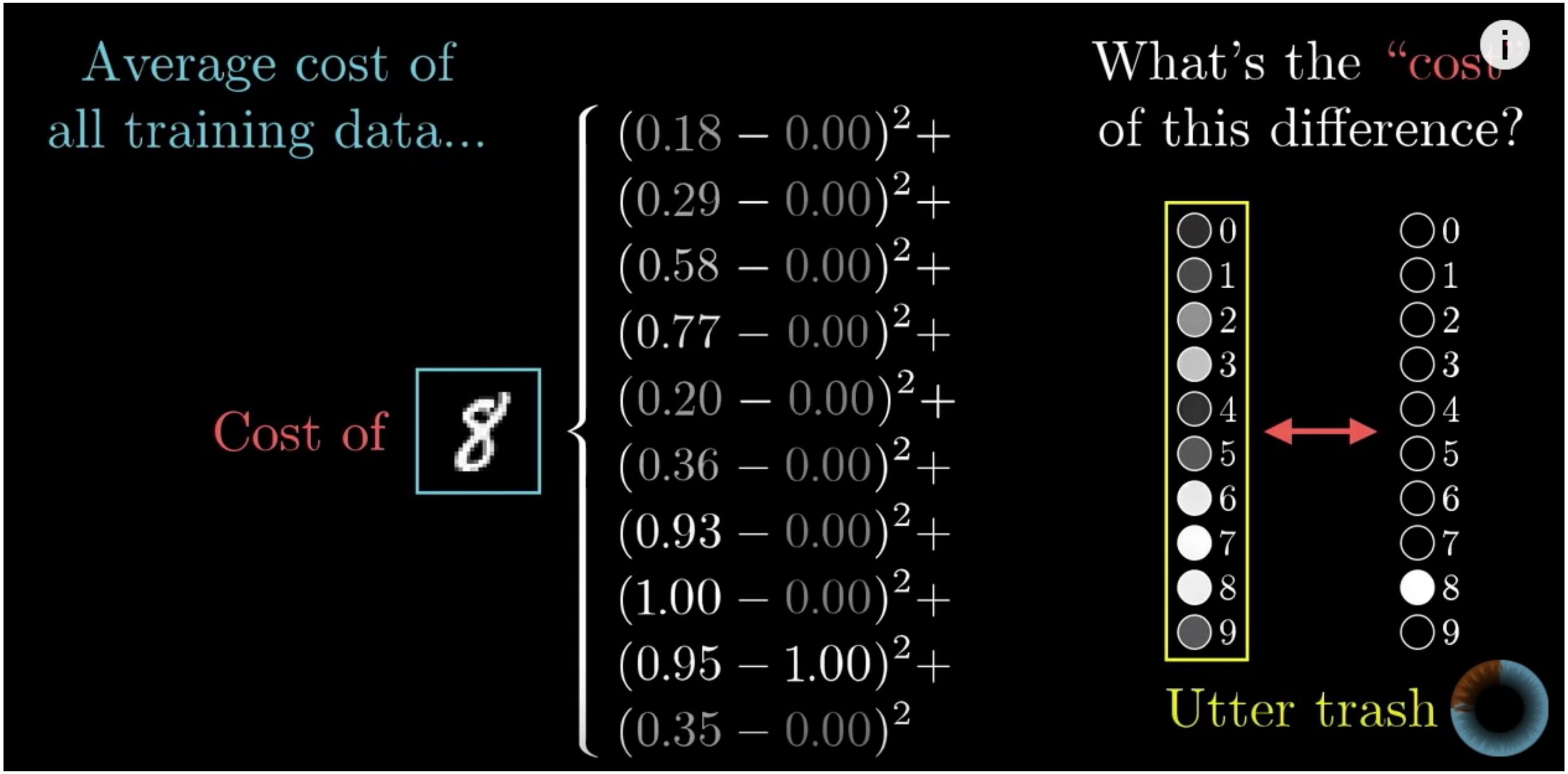
Neural Network and Deep Learning



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

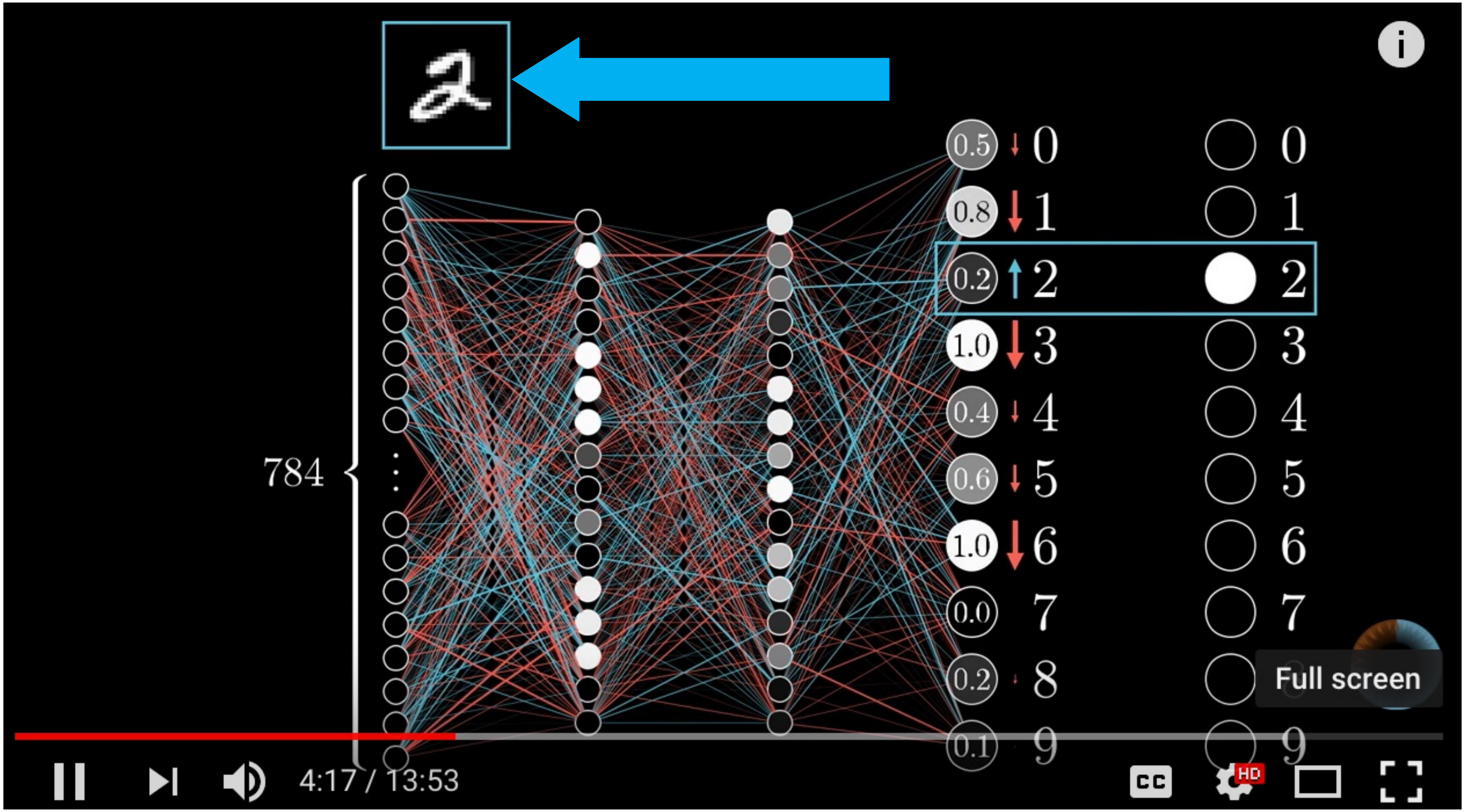
Gradient Descent

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

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"

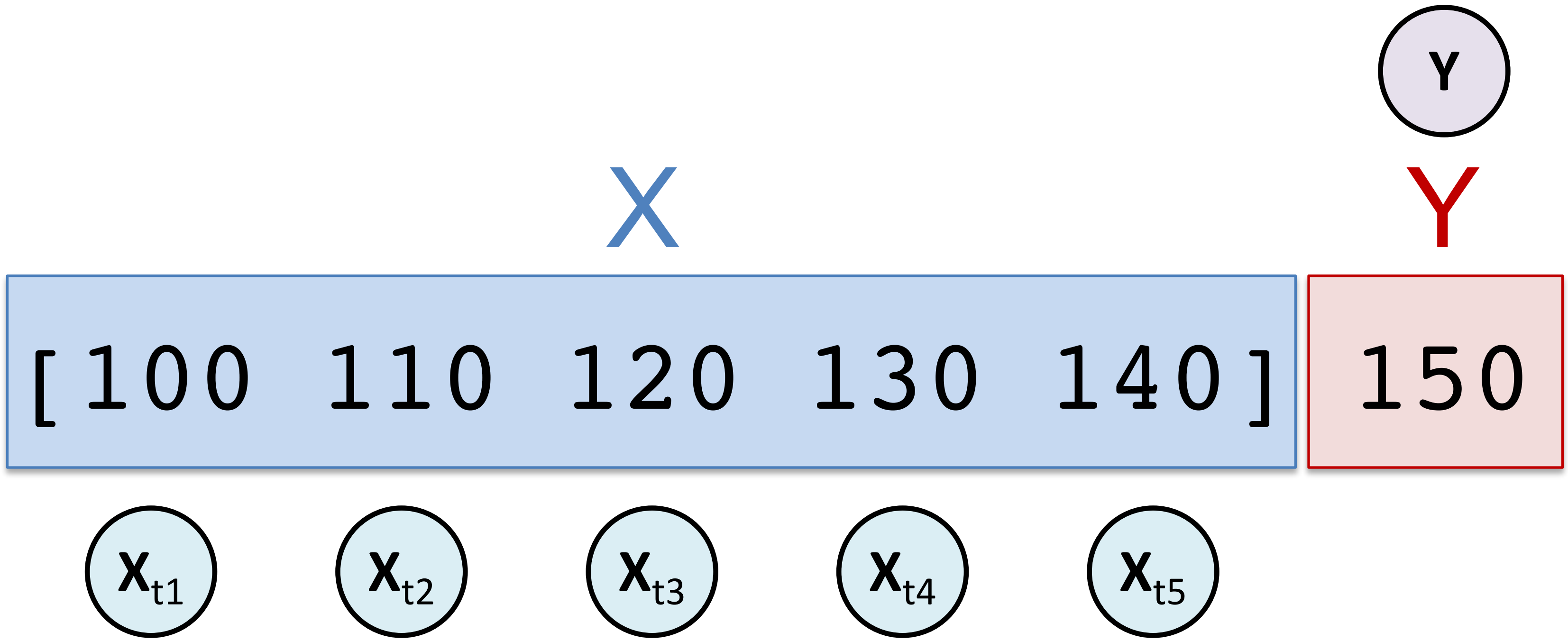
Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Financial Time Series Forecasting

Time Series Data

Time Series Data

[100, 110, 120, 130, 140, 150]



Deep Learning with TensorFlow

Deep Learning Software

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

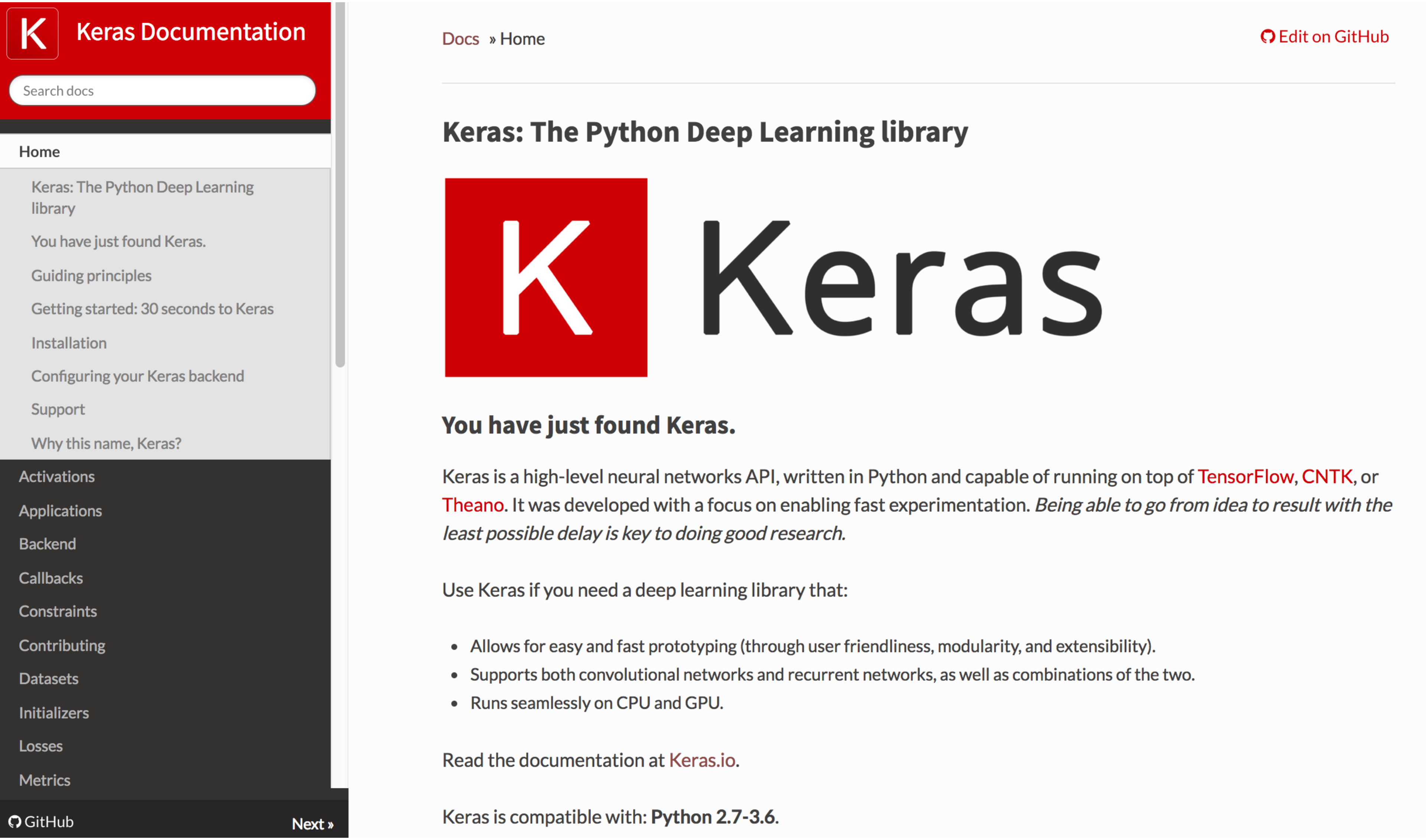
Source: http://deeplearning.net/software_links/

tf.keras

Keras:

High-level API for TensorFlow

Keras



Keras Documentation

Search docs

Home

Keras: The Python Deep Learning library

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Configuring your Keras backend

Support

Why this name, Keras?

Activations

Applications

Backend

Callbacks

Constraints

Contributing

Datasets

Initializers

Losses

Metrics

GitHub Next »

Docs » Home

Edit on GitHub

Keras: The Python Deep Learning library

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:

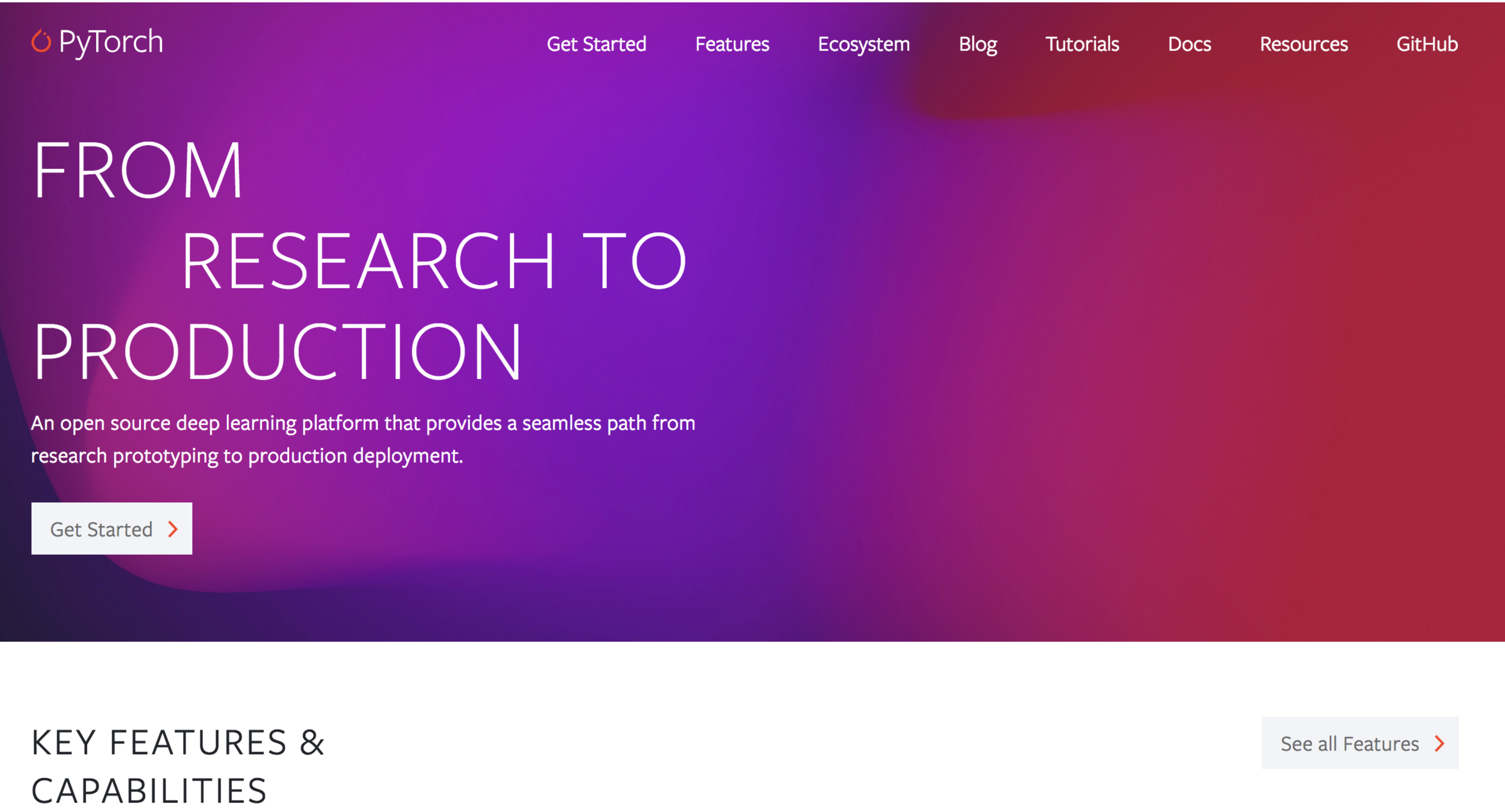
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

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

http://keras.io/

PyTorch



http://pytorch.org/

Keras

Keras

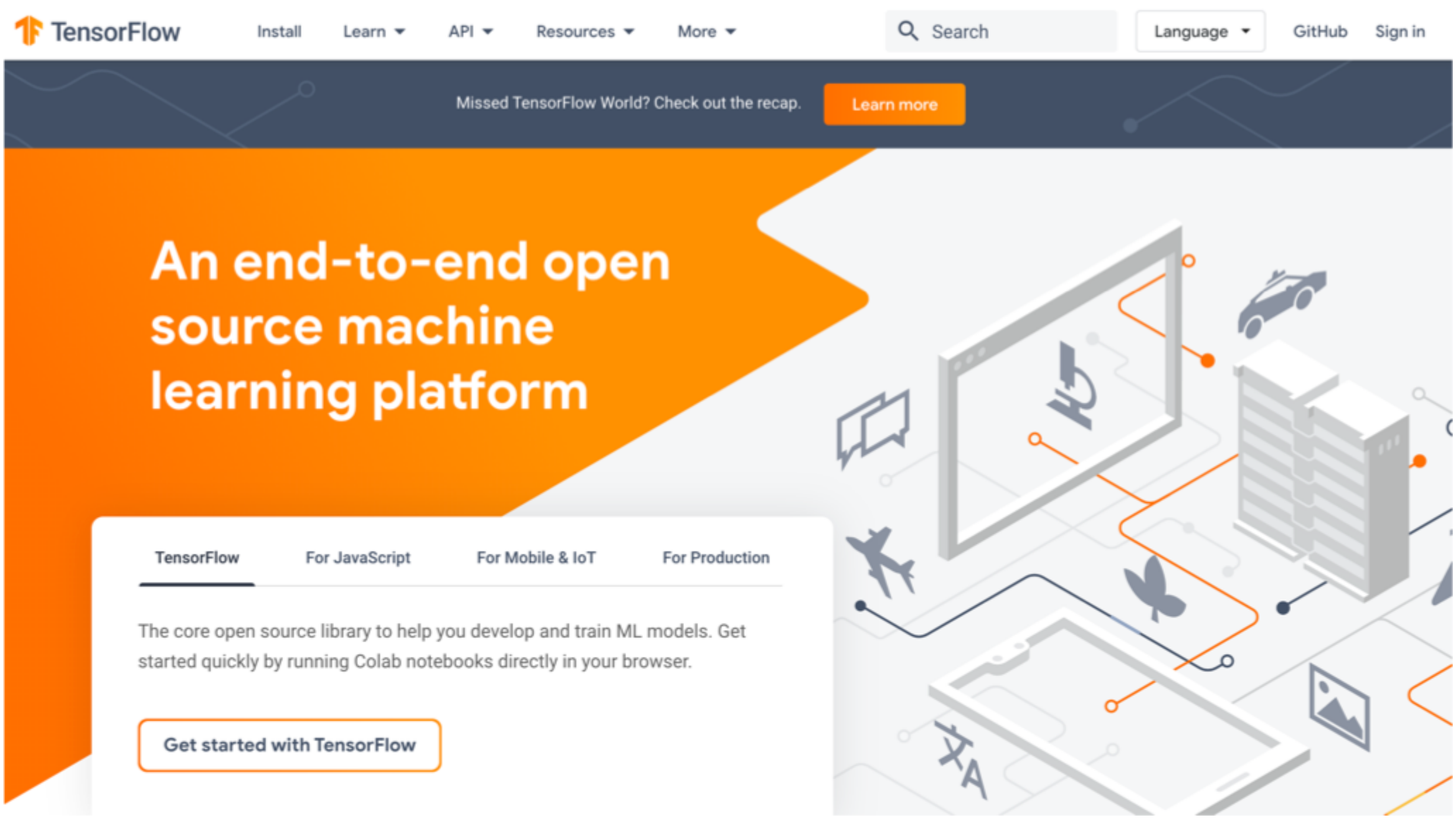
- Keras is a high-level neural networks API
- Written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- It was developed with a focus on enabling fast experimentation.
- Being able to go from idea to result with the least possible delay is key to doing good research.

Source: https://keras.io/

TensorFlow

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

TensorFlow



https://www.tensorflow.org/

TensorFlow

- An end-to-end open source machine learning platform.
- The core open source library to help you develop and train ML models.
- Get started quickly by running Colab notebooks directly in your browser.

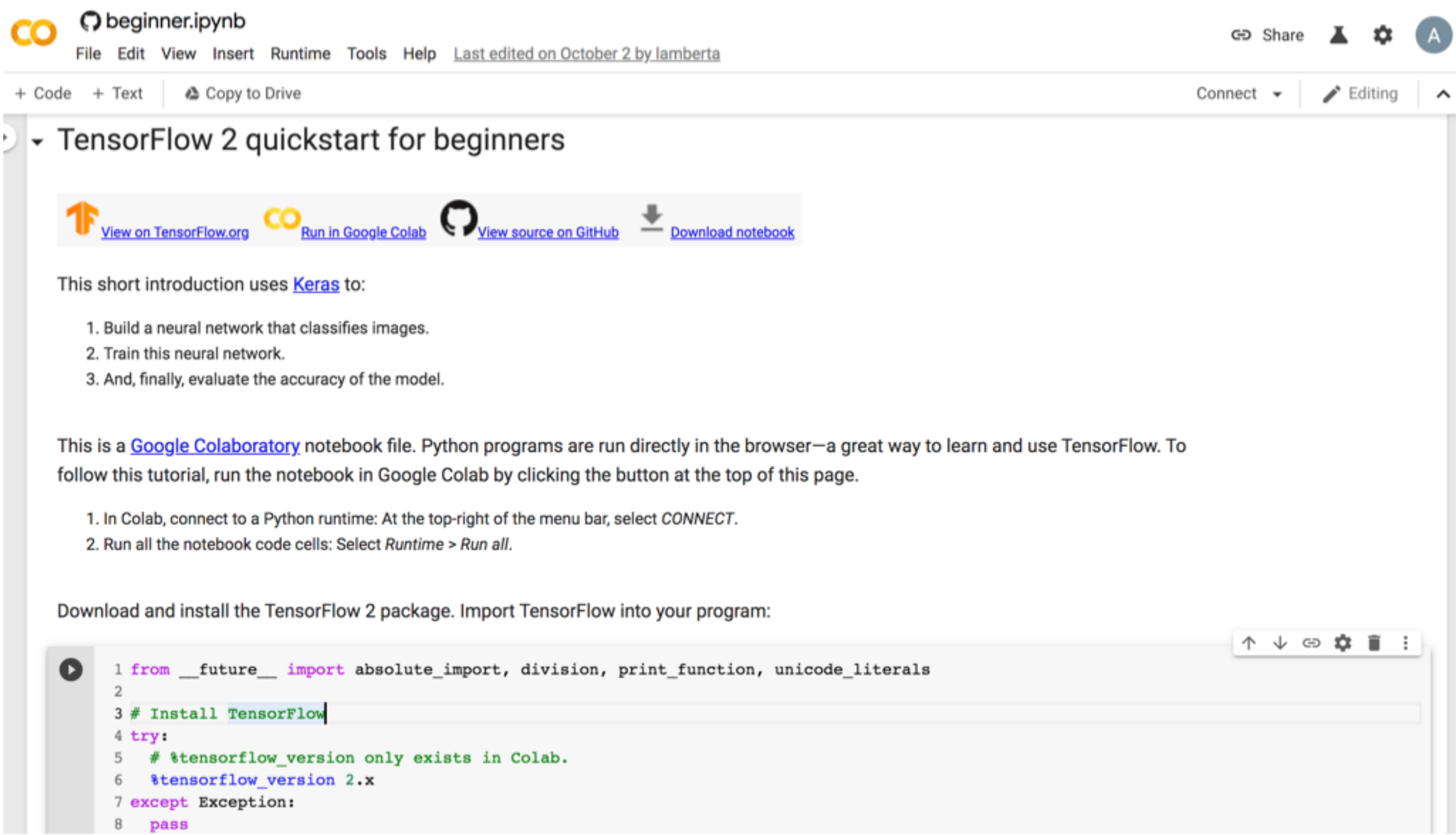
https://www.tensorflow.org/

TensorFlow 2.0

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/overview/

TensorFlow 2 Quick Start



beginner.ipynb

File Edit View Insert Runtime Tools Help Last edited on October 2 by lamberta

Share

+ Code + Text Copy to Drive

Connect Editing

TensorFlow 2 quickstart for beginners

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

This short introduction uses Keras to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

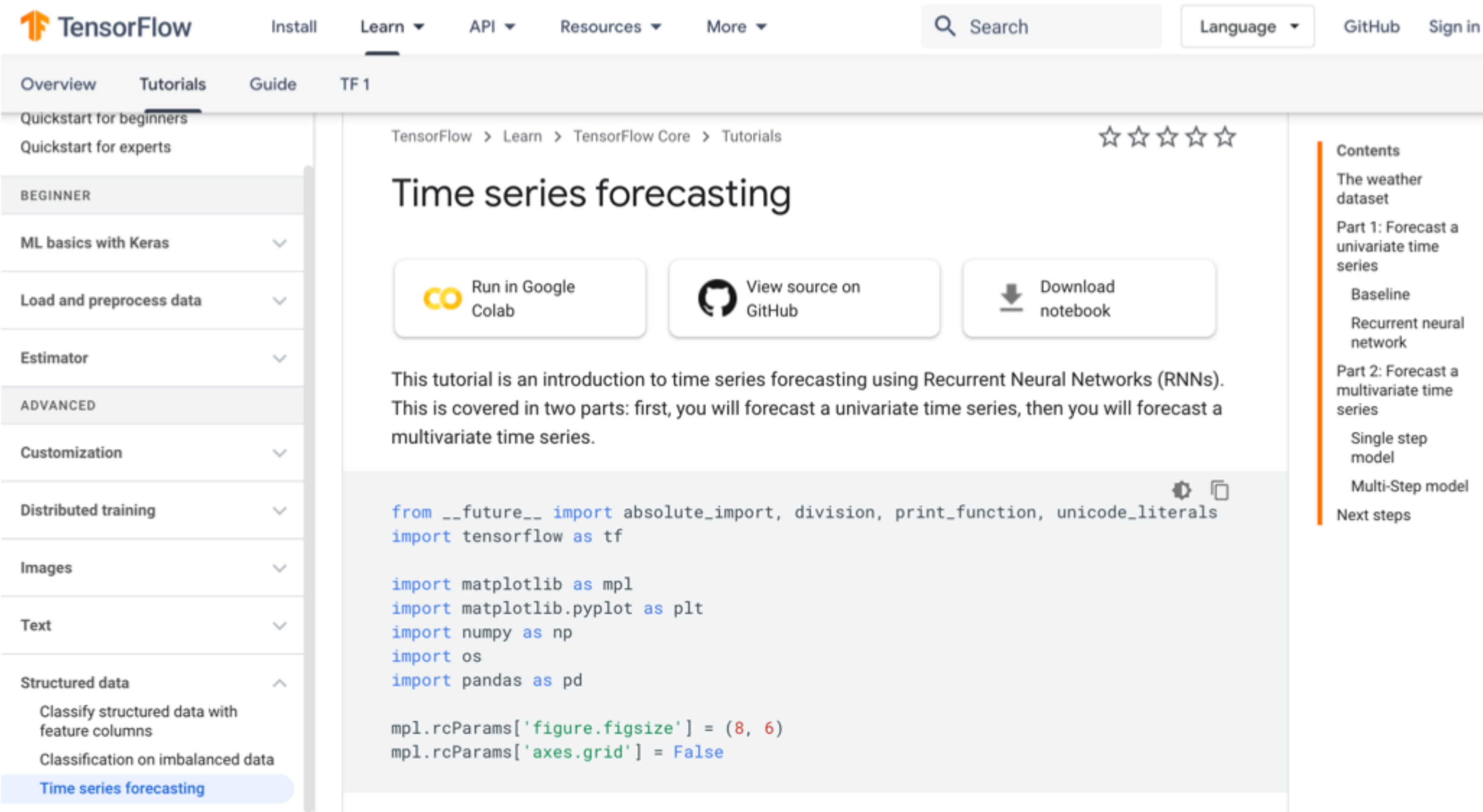
1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

Download and install the TensorFlow 2 package. Import TensorFlow into your program:

```
from __future__ import absolute_import, division, print_function, unicode_literals

# Install TensorFlow
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

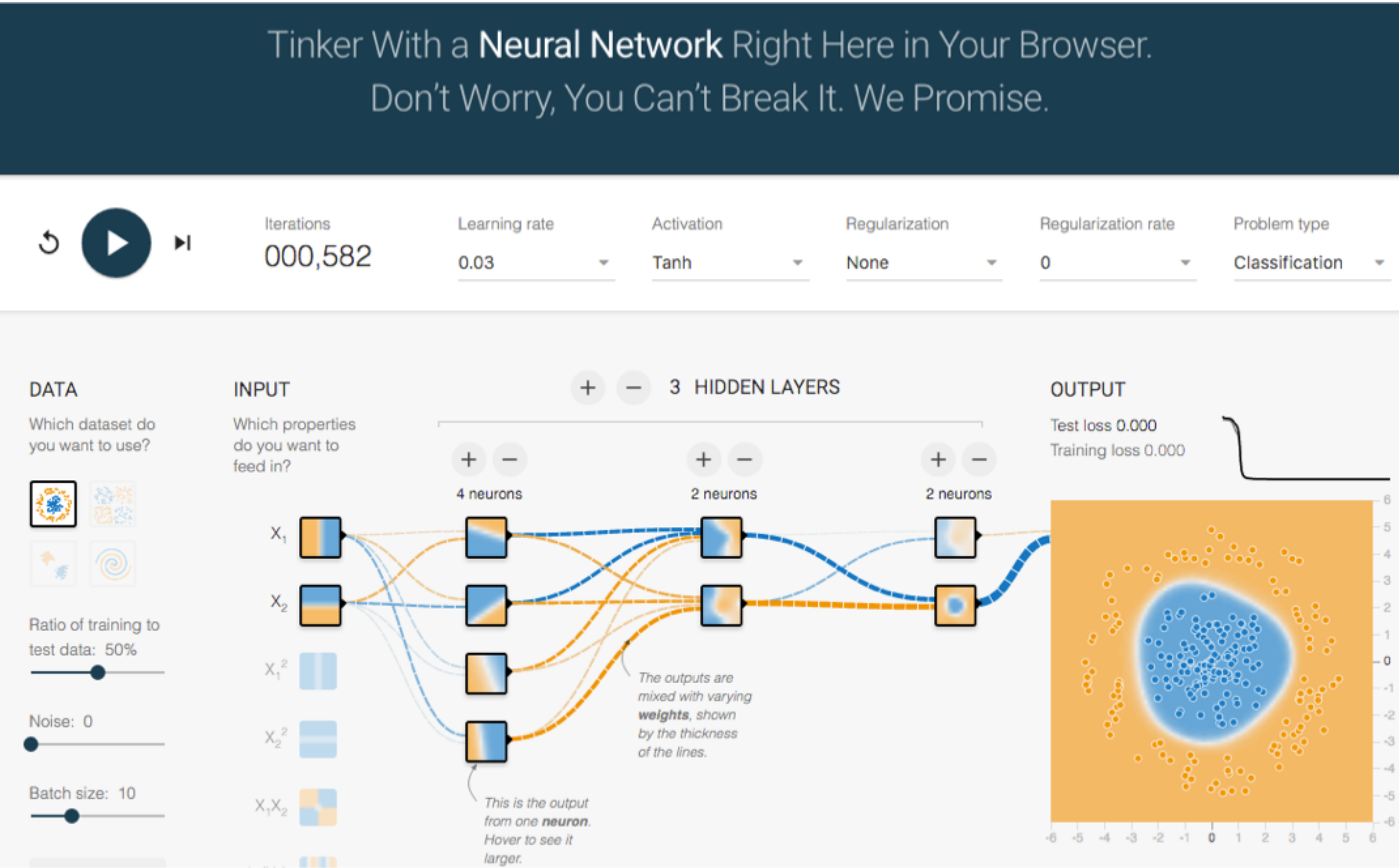
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb

TensorFlow 2
Time Series Forecasting



https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb

TensorFlow Playground



http://playground.tensorflow.org/

TensorFlow

is an

Open Source

Software Library

for

Machine Intelligence

https://www.tensorflow.org/

TensorFlow

Tensor

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

https://www.tensorflow.org/

Scalar

$$80$$

Vector

$$[50 \quad 60 \quad 70]$$

Matrix

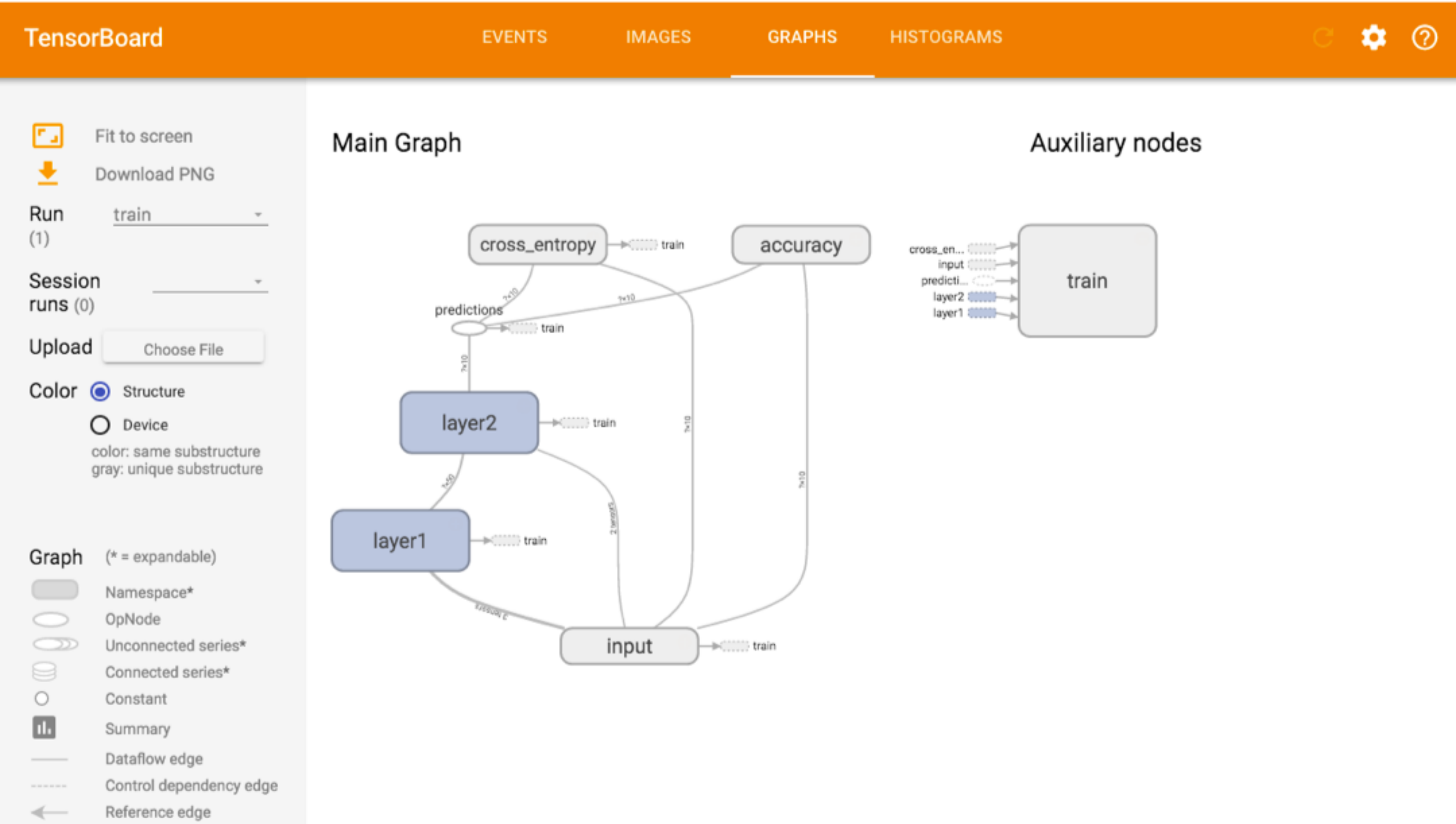
$$\begin{bmatrix} 50 & 60 & 70 \\ 55 & 65 & 75 \end{bmatrix}$$

Tensor

$$\begin{bmatrix} [50 \quad 60 \quad 70] & [70 \quad 80 \quad 90] \\ [55 \quad 65 \quad 75] & [75 \quad 85 \quad 95] \end{bmatrix}$$

TensorFlow

TensorBoard



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

Deep Learning for Financial Application Forecasting

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

Time Series Data

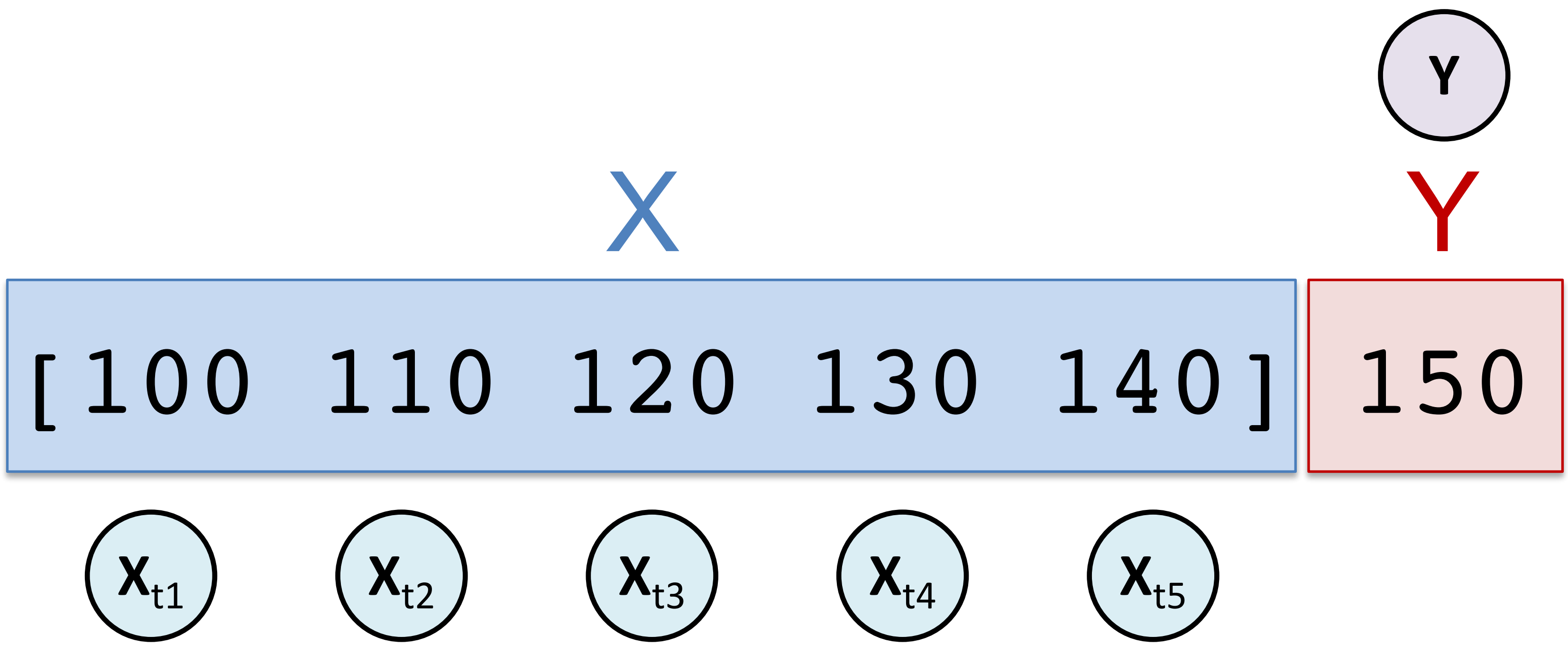
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

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

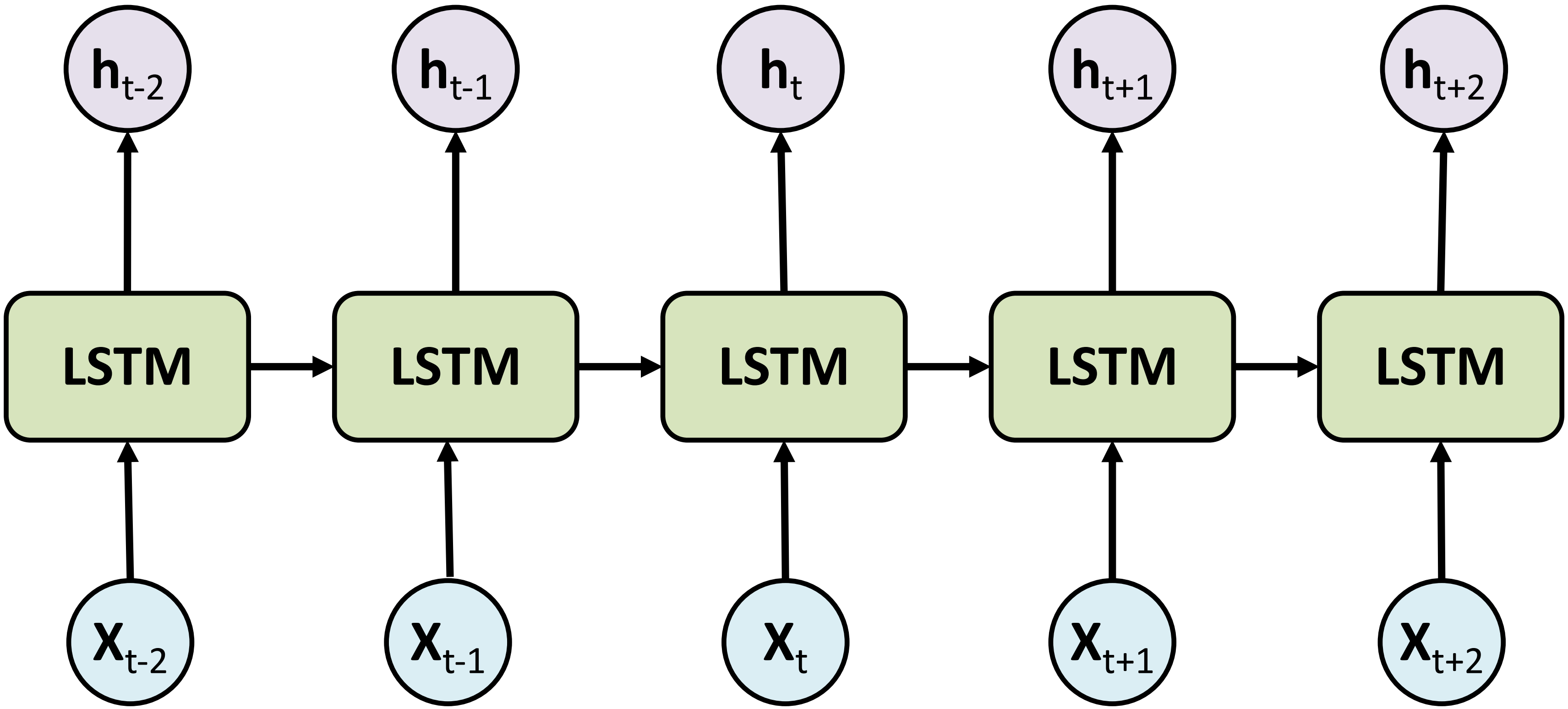
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Time Series Data

[100, 110, 120, 130, 140, 150]



Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

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

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

Deep Learning for Financial Time Series Forecasting

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

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

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

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

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

Deep Learning for Financial Time Series Forecasting

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

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

LSTM for Time Series Forecasting

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

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

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

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

Deep Learning for Financial Time Series Forecasting

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

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

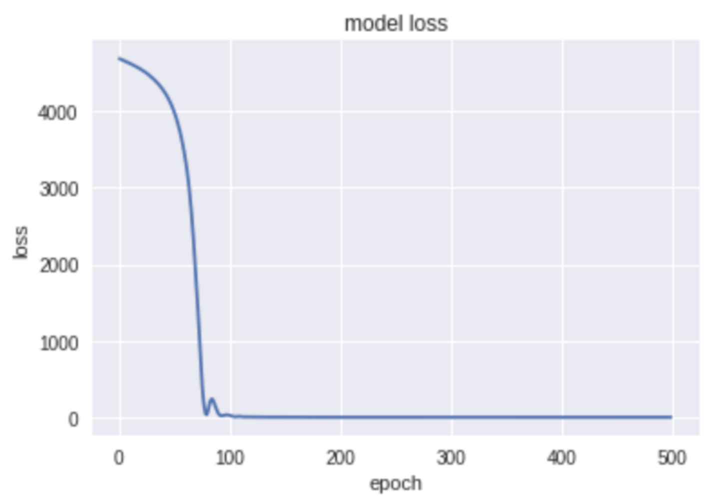
```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y =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

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

Deep Learning for Financial Time Series Forecasting

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

```
Using TensorFlow backend.
[[102.31296]]
yhat [[102.31296]]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
None
dict_keys(['loss'])
loss: 0.000000
loss: 1.2578432517784677e-07
```



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

Deep Learning for Financial Time Series Forecasting

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

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

Basic Classification
Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcutIzNHjRSLbeVl4kd5z

CO tf01_basic_classification.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents | Code snippets | Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ 2 cells hidden

Train your first neural network: basic classification

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

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

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

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

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

Text Classification

IMDB Movie Reviews

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

CO tf02_basic-text-classification.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Copyright 2018 The TensorFlow Authors.

2 cells hidden

Text classification with movie reviews

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

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

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

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

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

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

Basic Regression
Predict House Prices

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

CO tf03_basic-regression.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

Copyright 2018 The TensorFlow Authors.

Predict house prices: regression

The Boston Housing Prices dataset

Examples and features

Labels

Normalize features

Create the model

Train the model

Predict

Conclusion

SECTION

Copyright 2018 The TensorFlow Authors.

2 cells hidden

Predict house prices: regression

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

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

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

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

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

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

Python in Google Colab

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

python101.ipynb

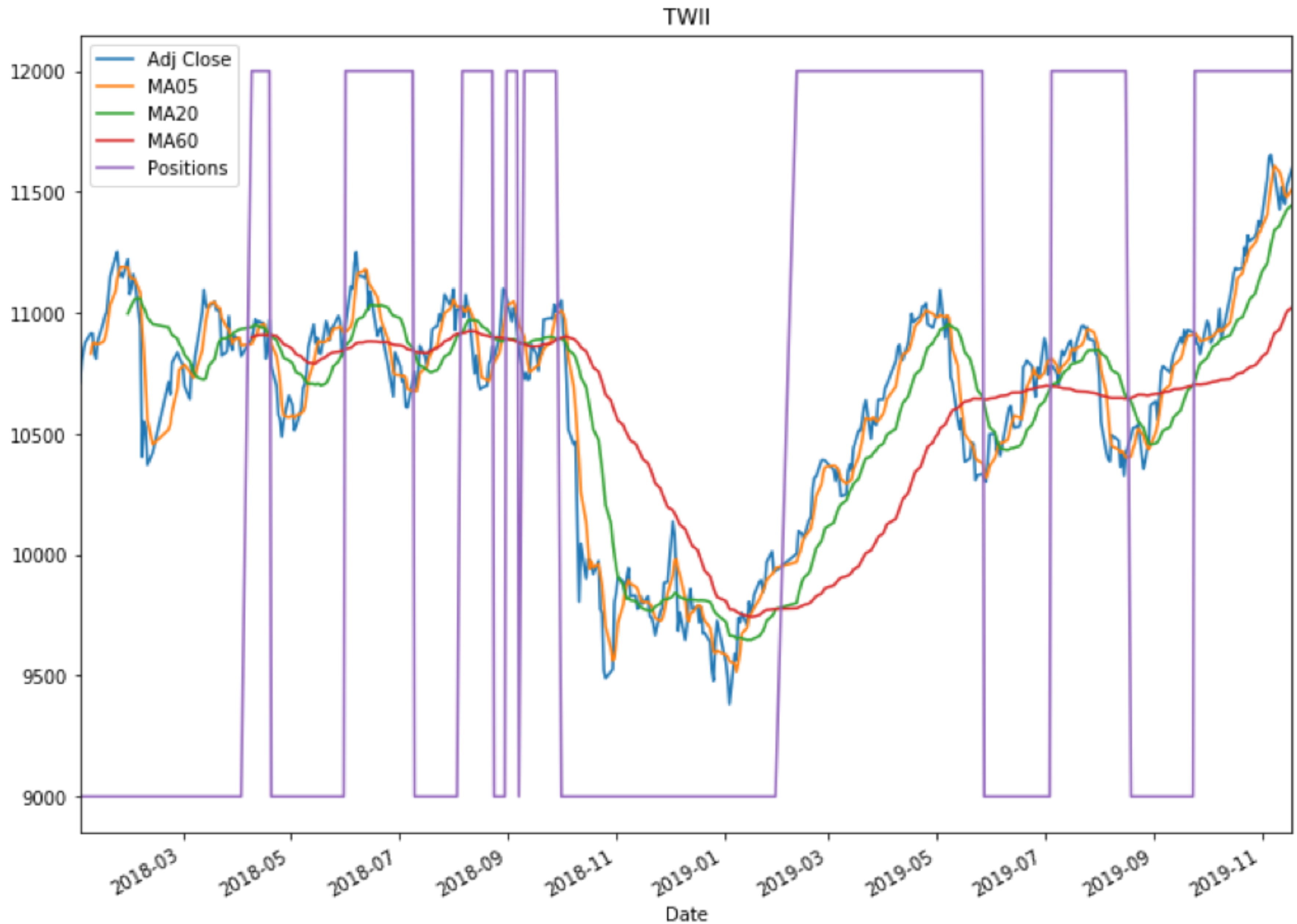
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text RAM Disk Editing

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

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2018, 1, 1)
df = web.DataReader("^TWII", 'yahoo', start, end) #^TWII #2330.TW #^DJI #AAPL
df.to_csv('TWII.csv')
print(df.head())
print(df.tail())
df2 = pd.read_csv('TWII.csv') #df.from_csv('AAPL.csv')
print(df2.tail())

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

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

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

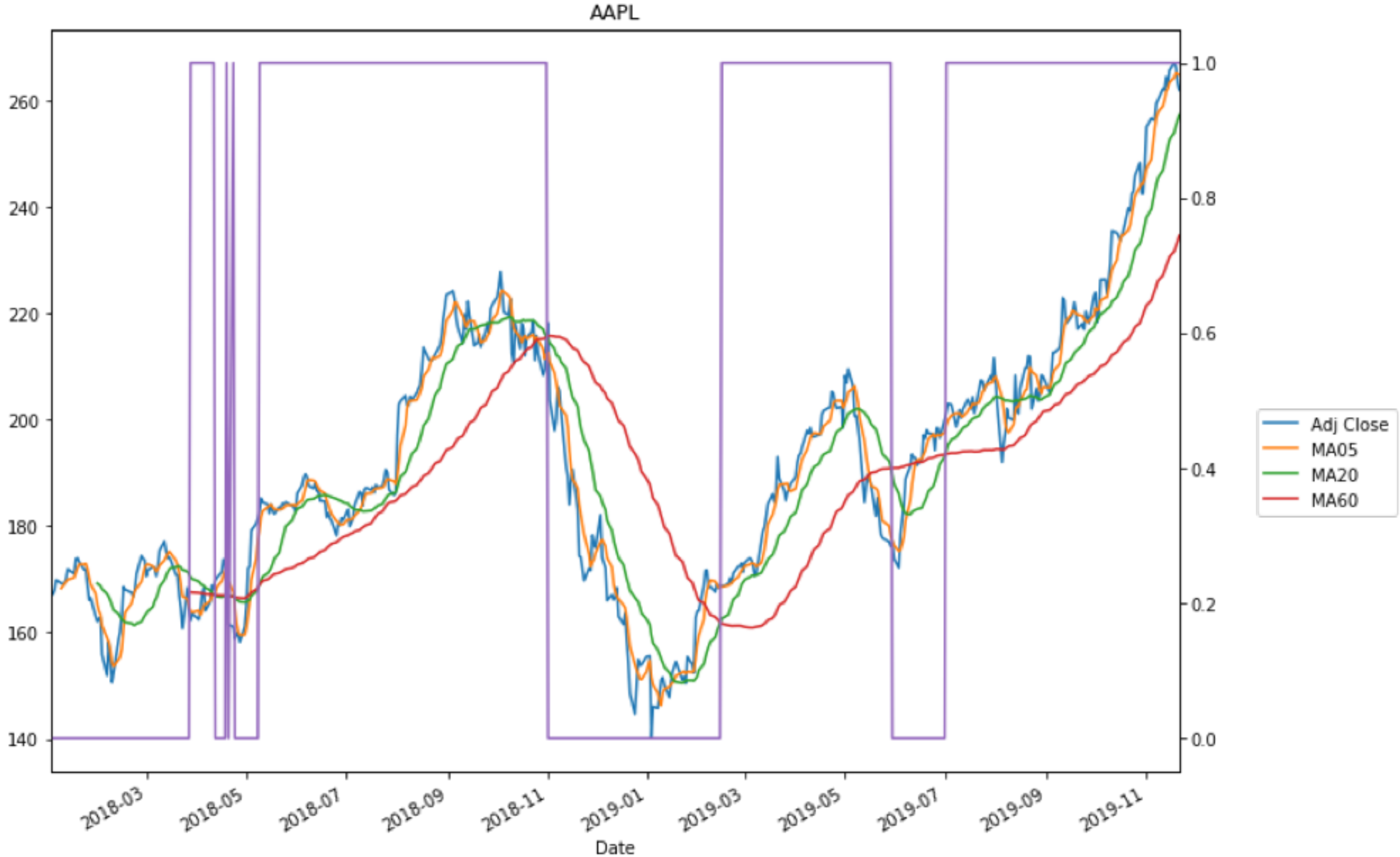


np.where (df['MA20'] > df['MA60'], 12000, 9000)

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df['Positions'] = np.where(df['MA20'] > df['MA60'], 12000,  9000)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Positions']})
```

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

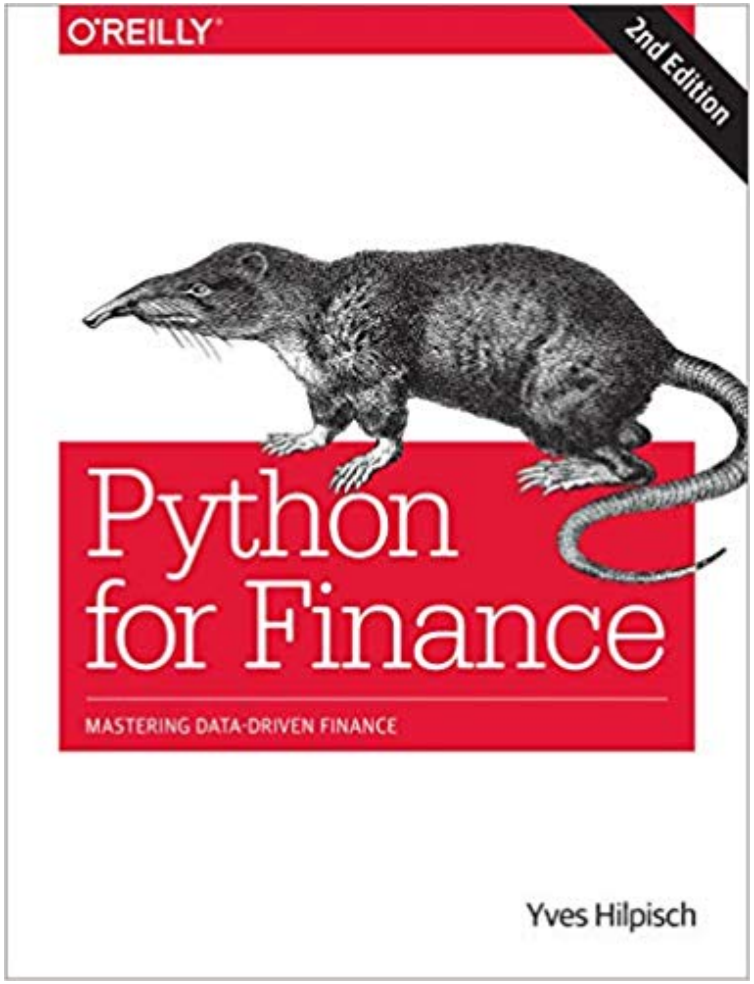
```
df2.plot(figsize=(12, 9), legend=True, title='AAPL',
secondary_y ='Positions').legend(bbox_to_anchor=(1.2, 0.5))
```



np.where (df['MA20'] > df['MA60’], 1, 0)

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df['Positions'] = np.where(df['MA20'] > df['MA60’], 1,  0)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05
'],'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Posit
ions’]})
```

Yves Hilpisch (2018),
Python for Finance: Mastering Data-Driven Finance,
O'Reilly



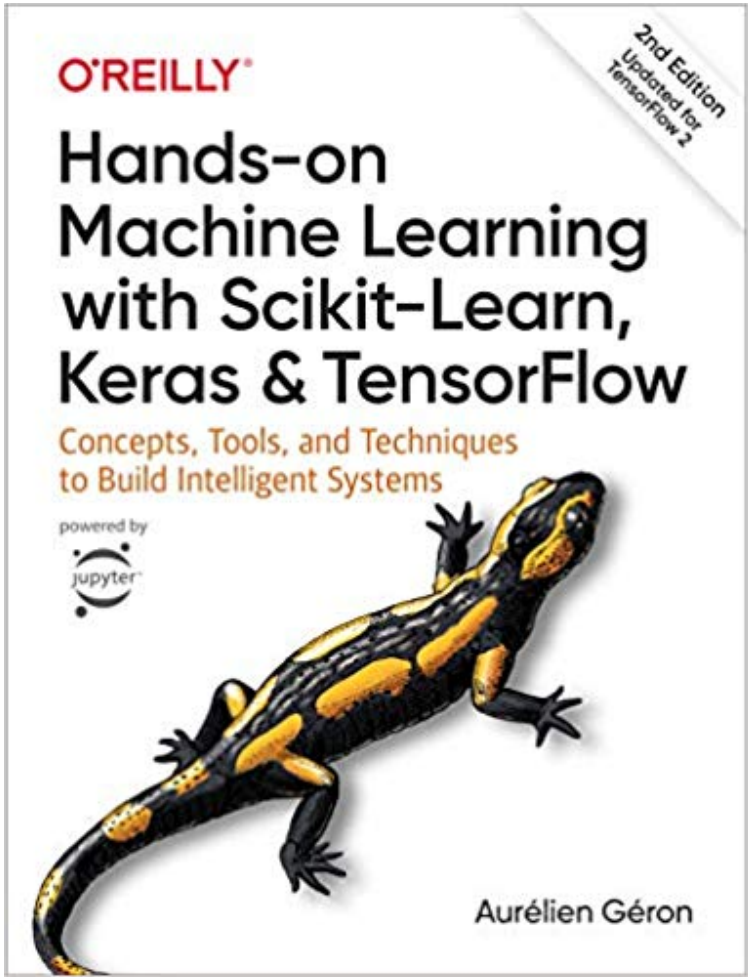
https://github.com/yhilpisch/py4fi2nd

Source: https://www.amazon.com/Python-Finance-Mastering-Data-Driven/dp/1492024333

Aurélien Géron (2019),

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition

O'Reilly Media, 2019



https://github.com/ageron/handson-ml2

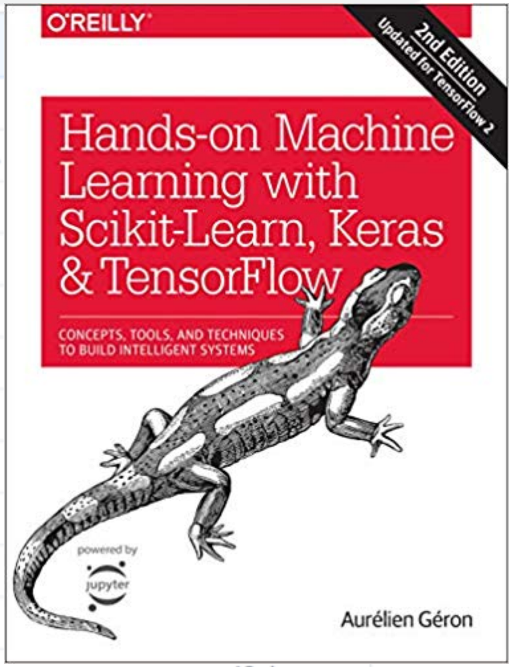
Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1492032646/

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

github.com/ageron/handson-ml2

ageron loss = metric * mean of sample weights, fixes #63

datasets	Fix vertical bars	
docker	Remove pyvirtualdisplay from environment.yml and add it to the Docker...	
images	Add breakout.gif	
work_in_progress	Remove from __future__ imports as we move away from Python 2	
.gitignore	Add jsb_chorales dataset to .gitignore	
01_the_machine_learning_landsca...	Fix typo on import urllib	
02_end_to_end_machine_learning_...	Make notebooks 1 to 9 runnable in Colab without changes	
03_classification.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	
04_training_linear_models.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	
05_support_vector_machines.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	
06_decision_trees.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	
07_ensemble_learning_and_rando...	Make notebooks 1 to 9 runnable in Colab without changes	13 days ago
08_dimensionality_reduction.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	13 days ago
09_unsupervised_learning.ipynb	Make notebooks 1 to 9 runnable in Colab without changes	13 days ago
10_neural_nets_with_keras.ipynb	Make notebooks 10 and 11 runnable in Colab without changes	13 days ago
11_training_deep_neural_networks....	Make notebooks 10 and 11 runnable in Colab without changes	13 days ago
12_custom_models_and_training_...	loss = metric * mean of sample weights, fixes #63	6 days ago
13_loading_and_preprocessing_da...	Make notebook 13 runnable in Colab without changes	13 days ago
14_deep_computer_vision_with_cn...	Make notebooks 14 to 19 runnable in Colab without changes	13 days ago
15_processing_sequences_using_r...	Make notebooks 14 to 19 runnable in Colab without changes	13 days ago

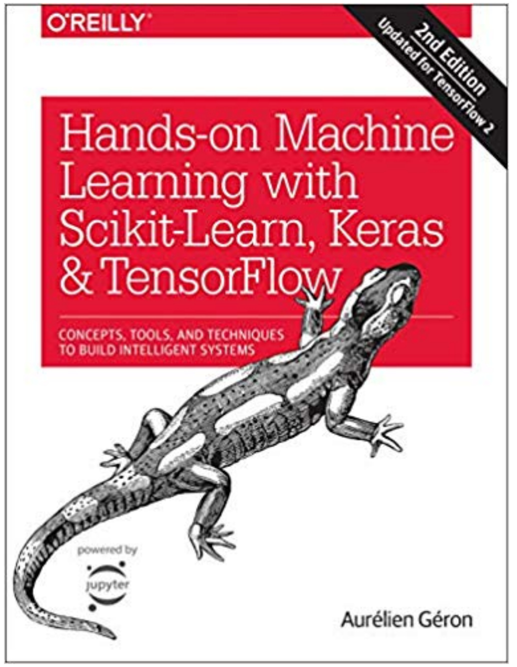


https://github.com/ageron/handson-ml2

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

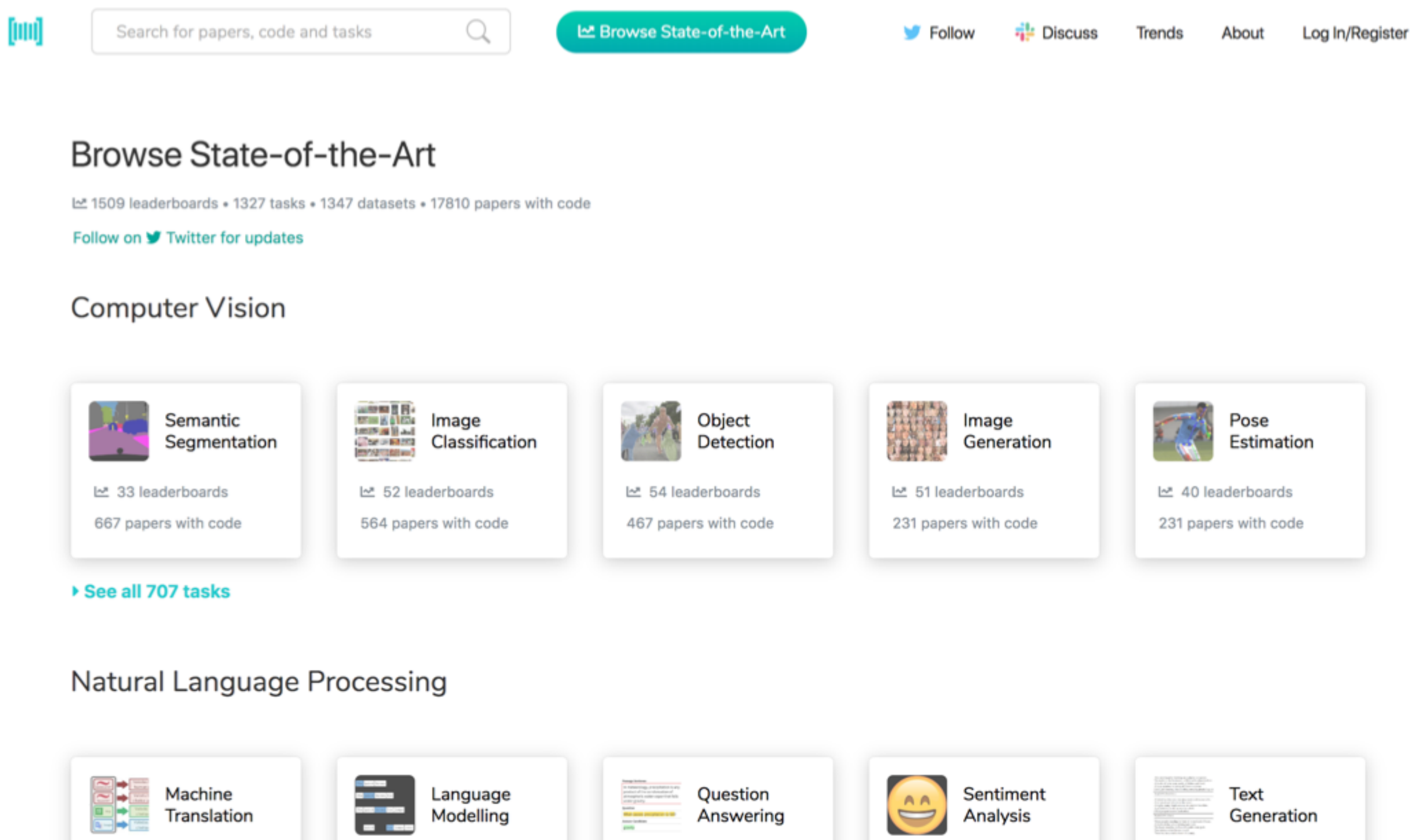
Notebooks

1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml2

Papers with Code
State-of-the-Art (SOTA)



https://paperswithcode.com/sota

Papers with Code
Stock Market Prediction

Browse > Time Series > Stock Market Prediction

Stock Market Prediction

6 papers with code · Time Series

Leaderboards

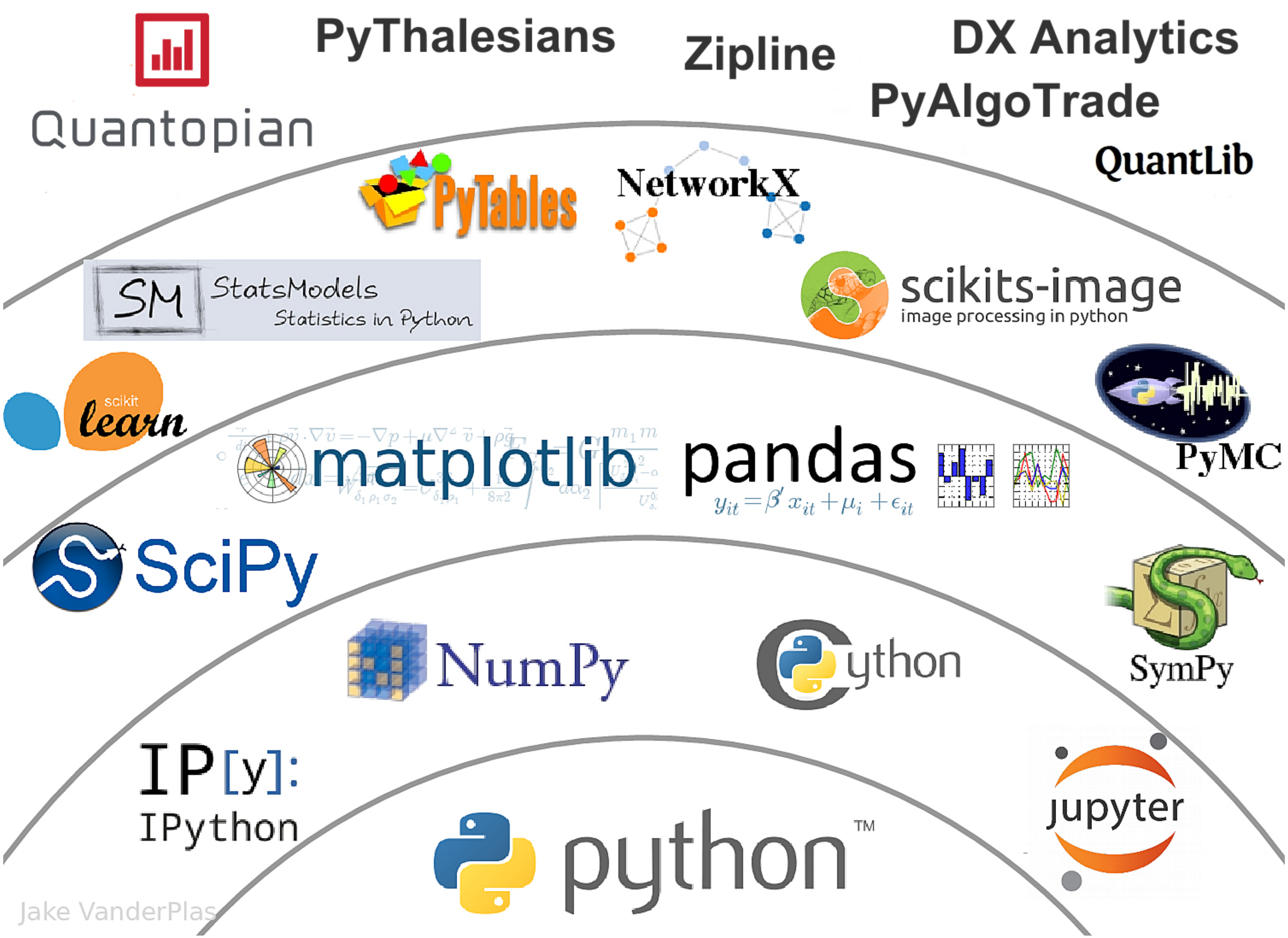
No evaluation results yet. Help compare methods by submit evaluation metrics.

Subtasks

- Stock Price Prediction — 3 papers with code
- Stock Trend Prediction — 2 papers with code
- Stock Prediction — 1 papers with code

https://paperswithcode.com/task/stock-market-prediction

The Quant Finance PyData Stack



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

Summary

- Deep Learning for Financial Application with TensorFlow
  - Deep Learning
  - Financial Application
  - TensorFlow

References

- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media, 2019, https://github.com/ageron/handson-ml2
- Yves Hilpisch (2018), "Python for Finance: Mastering Data-Driven Finance", 2nd Edition, O'Reilly Media. https://github.com/yhilpisch/py4fi2nd
- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media. https://github.com/wesm/pydata-book
- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Avinash Jain (2017), Introduction To Python Programming, Udemy, https://www.udemy.com/pythonforbeginnersintro/
- Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A
- Jason Brownlee (2016), Your First Machine Learning Project in Python Step-By-Step, https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
- Jason Brownlee (2018), How to Develop LSTM Models for Time Series Forecasting, https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.
- Jay Alammar (2019), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/
- Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow