

人工智慧投資分析



Tamkang
Universit

淡江大學

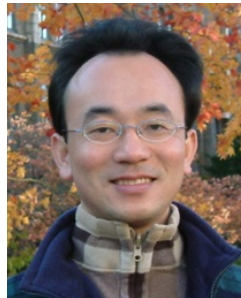
AI for Investment Analysis

TensorFlow 深度學習投資分析 (Deep Learning for Investment Analysis with TensorFlow)

1082AIIA09

MBA, IMTKU (M2399) (8409) (Spring 2020)

Wed 3, 4 (10:10-12:00) (B206)



Min-Yuh Day

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/>

2020-05-13, 05-20, 06-03



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/03/04	人工智慧投資分析課程介紹 (Course Orientation on AI for Investment Analysis)
2	2020/03/11	AI 金融科技: 金融服務創新應用 (AI in FinTech: Financial Services Innovation and Application)
3	2020/03/18	機器人理財顧問與AI交談機器人 (Robo-Advisors and AI Chatbots)
4	2020/03/25	投資心理學與行為財務學 (Investing Psychology and Behavioral Finance)
5	2020/04/01	財務金融事件研究法 (Event Studies in Finance)
6	2020/04/08	人工智慧投資分析個案研究 I (Case Study on AI for Investment Analysis I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/04/15	Python AI投資分析基礎 (Foundations of AI Investment Analysis in Python)
8	2020/04/22	Python Pandas 量化投資分析 (Quantitative Investing with Pandas in Python)
9	2020/04/29	期中報告 (Midterm Project Report)
10	2020/05/06	Python Scikit-Learn 機器學習投資分析 (Machine Learning for Investment Analysis with Scikit-Learn in Python)
11	2020/05/13	TensorFlow 深度學習投資分析 I (Deep Learning for Investment Analysis with TensorFlow I)
12	2020/05/20	TensorFlow 深度學習投資分析 II (Deep Learning for Investment Analysis with TensorFlow II)

課程大綱 (Syllabus)

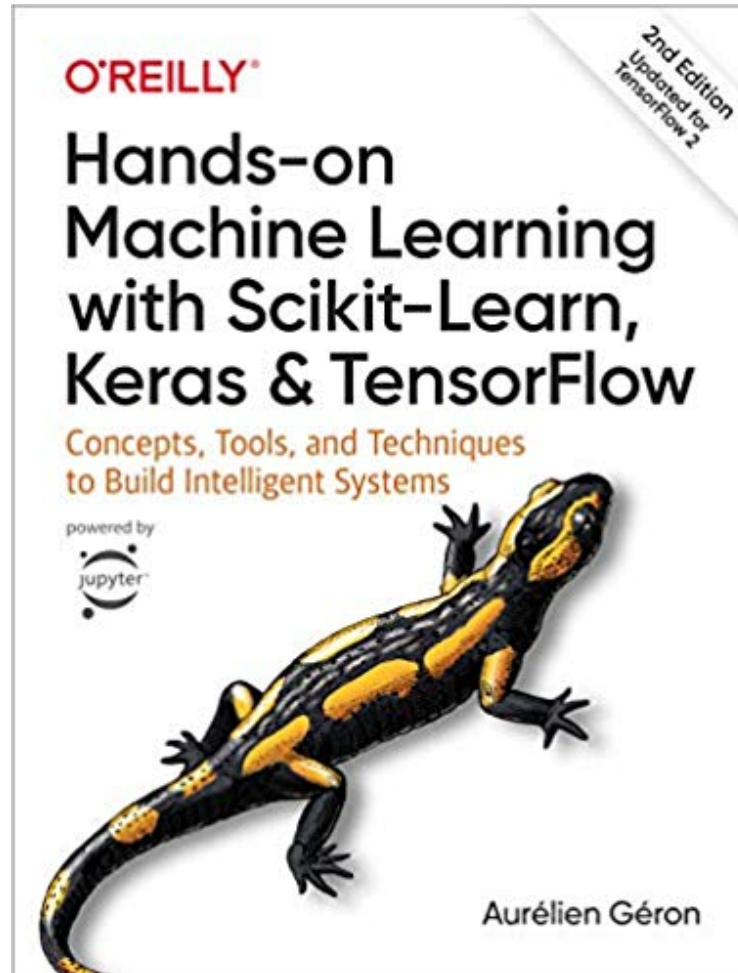
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/05/27	人工智慧投資分析個案研究 II (Case Study on Artificial Intelligence for Investment Analysis II)
14	2020/06/03	TensorFlow 深度學習投資分析 III (Deep Learning for Investment Analysis with TensorFlow III)
15	2020/06/10	投資組合最佳化與程式交易 (Portfolio Optimization and Algorithmic Trading)
16	2020/06/17	期末報告 I (Final Project Presentation I)
17	2020/06/24	期末報告 II (Final Project Presentation II)
18	2020/07/01	教師彈性補充教學

Deep Learning for Investment Analysis with TensorFlow

Outline

- **Deep Learning for Investment Analysis with TensorFlow**
 - **Deep Learning**
 - **Financial Time Series Forecasting**
 - **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**

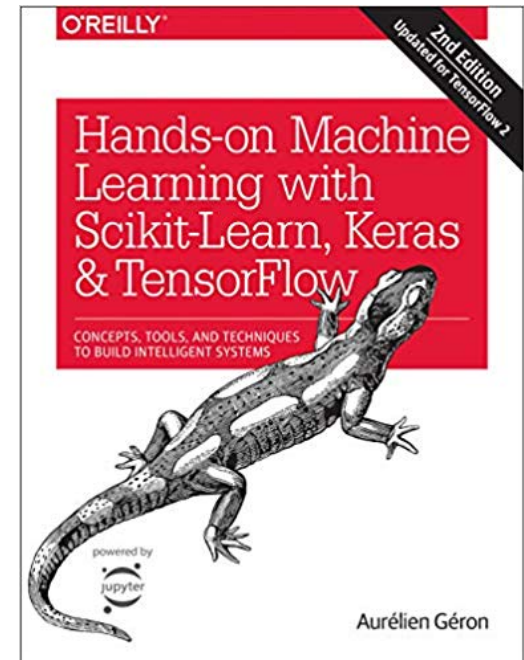


<https://github.com/ageron/handson-ml2>

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](#)



Sequences using RNNs and CNNs



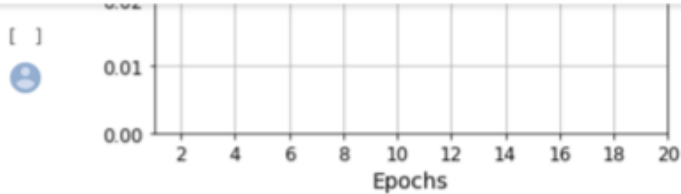
15_processing_sequences_using_rnn_and_cnns.ipynb

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

Share

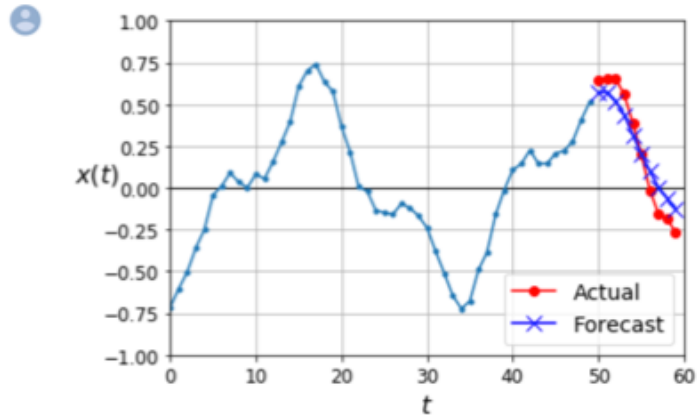
+ Code + Text Copy to Drive

Connect Editing



```
[ ] 1 np.random.seed(43)
2
3 series = generate_time_series(1, 50 + 10)
4 X_new, Y_new = series[:, :50, :], series[:, 50:, :]
5 Y_pred = model.predict(X_new[:, -1][..., np.newaxis])
```

```
[ ] 1 plot_multiple_forecasts(X_new, Y_new, Y_pred)
2 plt.show()
```



TensorFlow

Missed TensorFlow World? Check out the recap.

[Learn more](#)

An end-to-end open
source machine
learning platform

[TensorFlow](#)[For JavaScript](#)[For Mobile & IoT](#)[For Production](#)

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

[Get started with TensorFlow](#)

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.

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})
```

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)
```


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:

```

1 from __future__ import absolute_import, division, print_function, unicode_literals
2
3 # Install TensorFlow
4 try:
5     # %tensorflow_version only exists in Colab.
6     %tensorflow_version 2.x
7 except Exception:
8     pass
    
```

TensorFlow

Image Classification



Install
Learn ▾
API ▾
Resources ▾
More ▾

🔍

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 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__)
                    
```

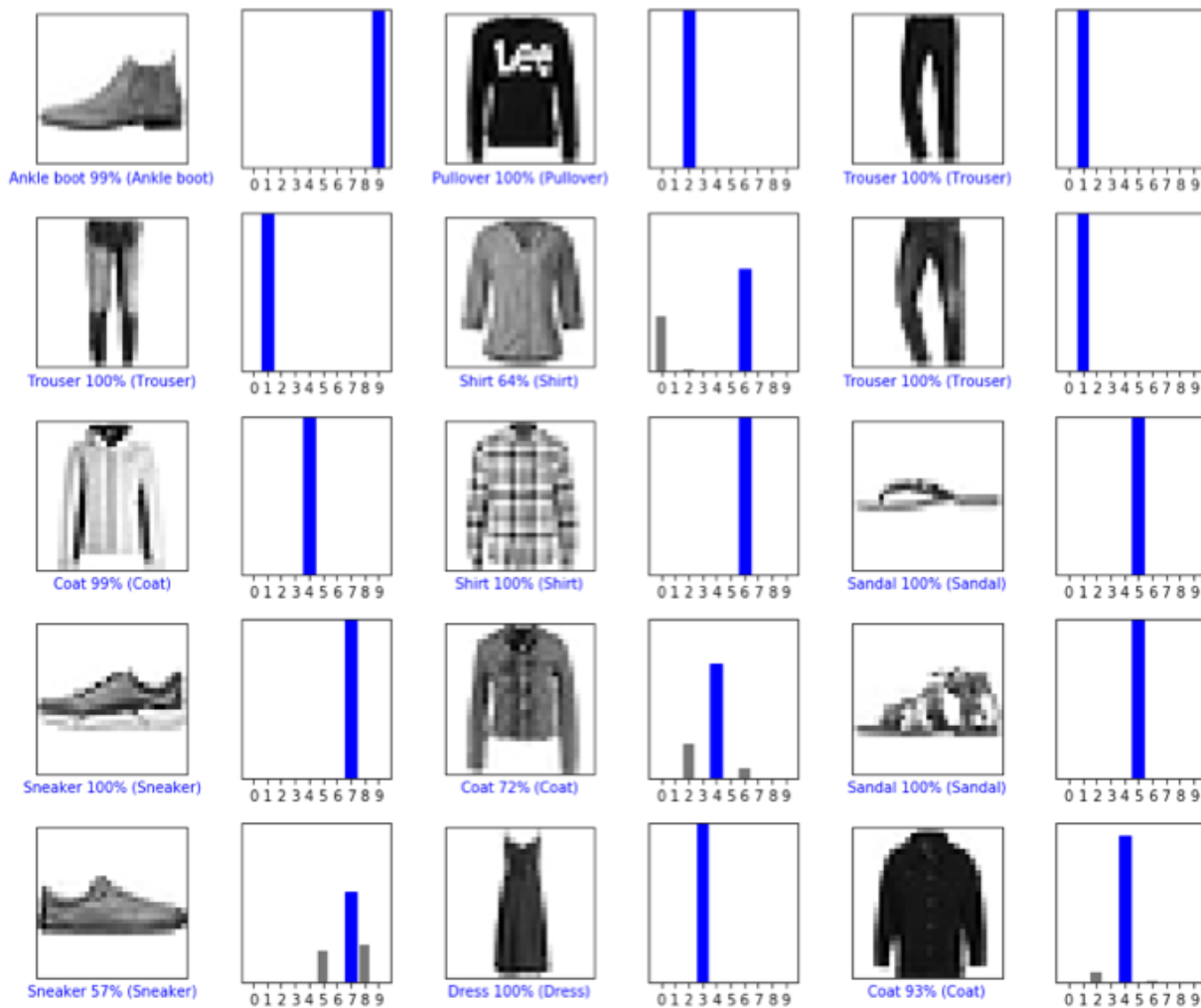
2.0.0


Contents

- Import the Fashion MNIST dataset
- Explore the data
- Preprocess the data
- Build the model
 - Set up the layers
 - Compile the model
- Train the model
- Evaluate accuracy
- Make predictions

Image Classification

Fashion MNIST dataset



Text Classification with TF Hub

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

Contents

- Download the IMDB dataset
- Explore the data
- Build the model
 - Loss function and optimizer
- Train the model
- Evaluate the model
- Further reading

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.

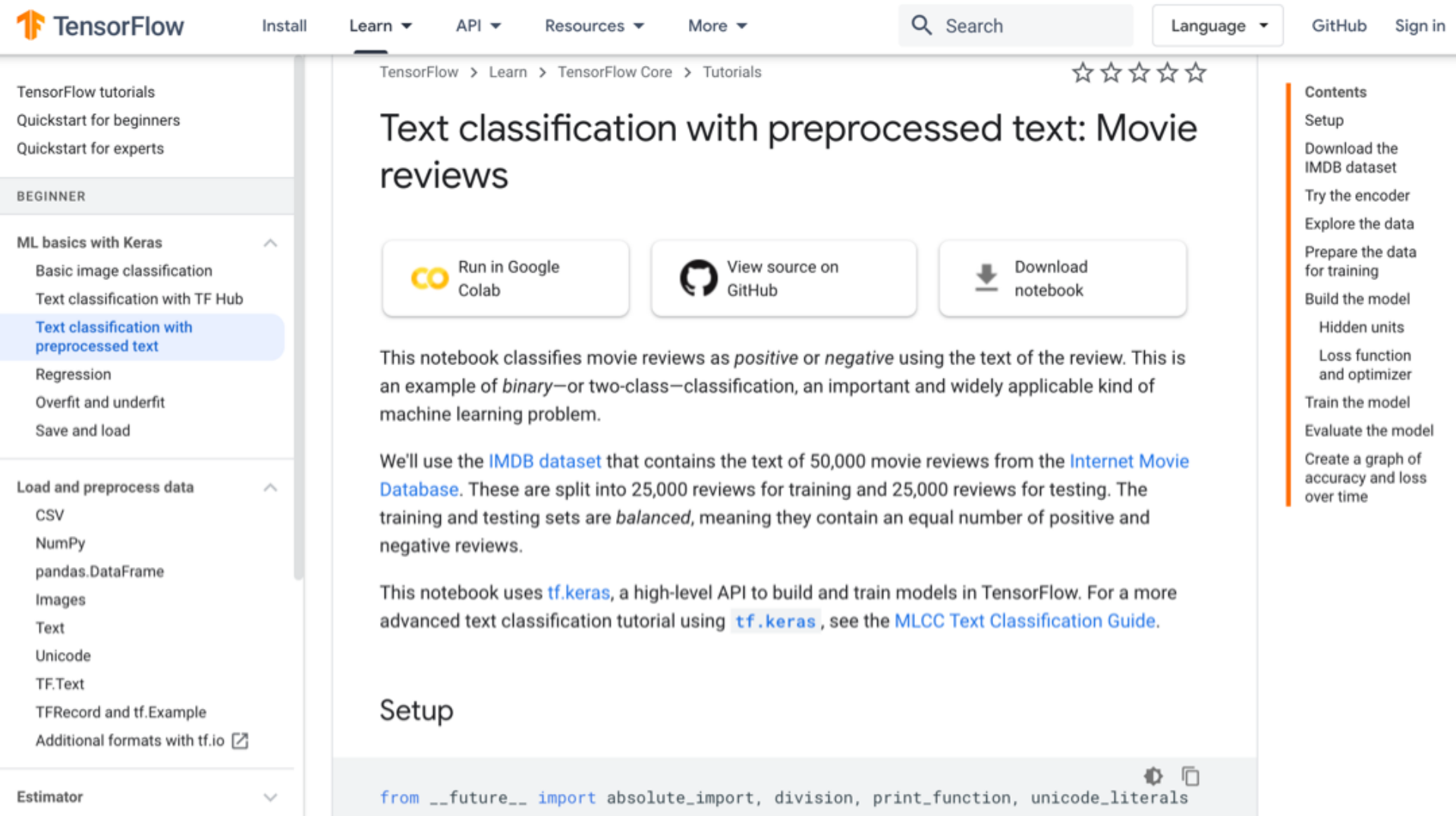
The tutorial demonstrates the basic application of transfer learning with TensorFlow Hub and Keras.

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
```

Text Classification with Pre Text



The screenshot shows the TensorFlow website interface. At the top, there is a navigation bar with 'TensorFlow', 'Install', 'Learn', 'API', 'Resources', and 'More' menus. A search bar and a 'Language' dropdown are also present. The main content area displays the title 'Text classification with preprocessed text: Movie reviews' and three action buttons: 'Run in Google Colab', 'View source on GitHub', and 'Download notebook'. The text below explains that the notebook classifies movie reviews as positive or negative using the text of the review, which is an example of binary classification. It mentions the use of the IMDB dataset and the tf.keras API. A 'Setup' section is visible at the bottom, showing the start of a code block: `from __future__ import absolute_import, division, print_function, unicode_literals`. On the right side, there is a 'Contents' sidebar with a list of sections: Setup, Download the IMDB dataset, Try the encoder, Explore the data, Prepare the data for training, Build the model (with sub-items: Hidden units, Loss function and optimizer), Train the model, Evaluate the model, and Create a graph of accuracy and loss over time.

Regression

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

The Auto MPG dataset

- Get the data
- Clean the data
- Split the data into train and test

- Inspect the data
- Split features from labels
- Normalize the data

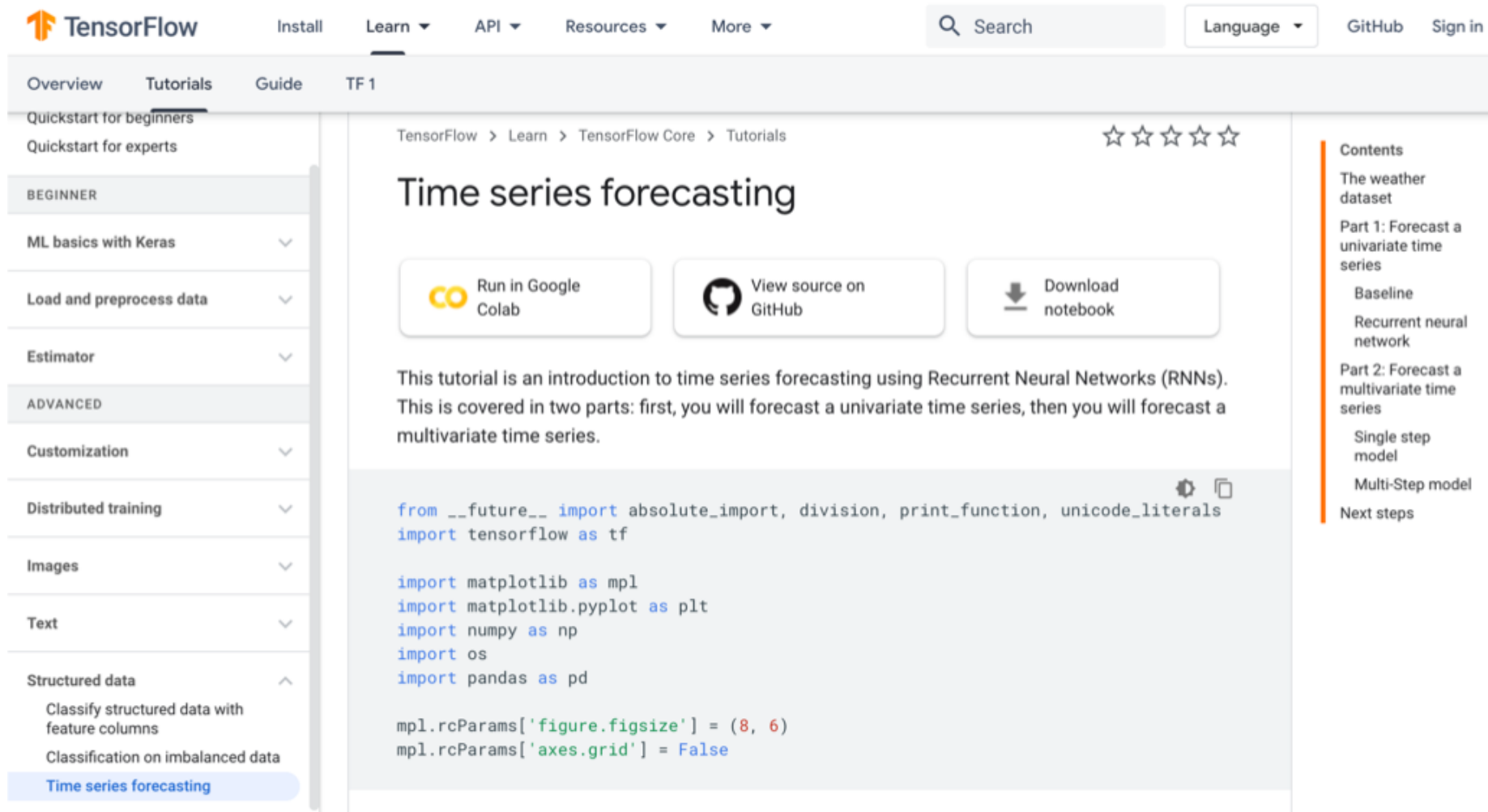
The model

- Build the model
- Inspect the model
- Train the model
- Make predictions

Conclusion

TensorFlow 2.0

Time Series Forecasting



The screenshot shows the TensorFlow 2.0 website interface for the 'Time series forecasting' tutorial. The page includes a navigation menu with 'Overview', 'Tutorials', 'Guide', and 'TF 1'. The 'Tutorials' section is expanded to show 'Time series forecasting' as the selected item. The main content area features the title 'Time series forecasting' with a breadcrumb trail 'TensorFlow > Learn > TensorFlow Core > Tutorials' and five star ratings. Below the title are three buttons: 'Run in Google Colab', 'View source on GitHub', and 'Download notebook'. The introductory text states: 'This tutorial is an introduction to time series forecasting using Recurrent Neural Networks (RNNs). This is covered in two parts: first, you will forecast a univariate time series, then you will forecast a multivariate time series.' A code block shows the initial Python imports for tensorflow, matplotlib, numpy, os, and pandas. The right sidebar contains a 'Contents' section with links to 'The weather dataset', 'Part 1: Forecast a univariate time series' (including 'Baseline' and 'Recurrent neural network'), 'Part 2: Forecast a multivariate time series' (including 'Single step model' and 'Multi-Step model'), and 'Next steps'.

Time Series Data

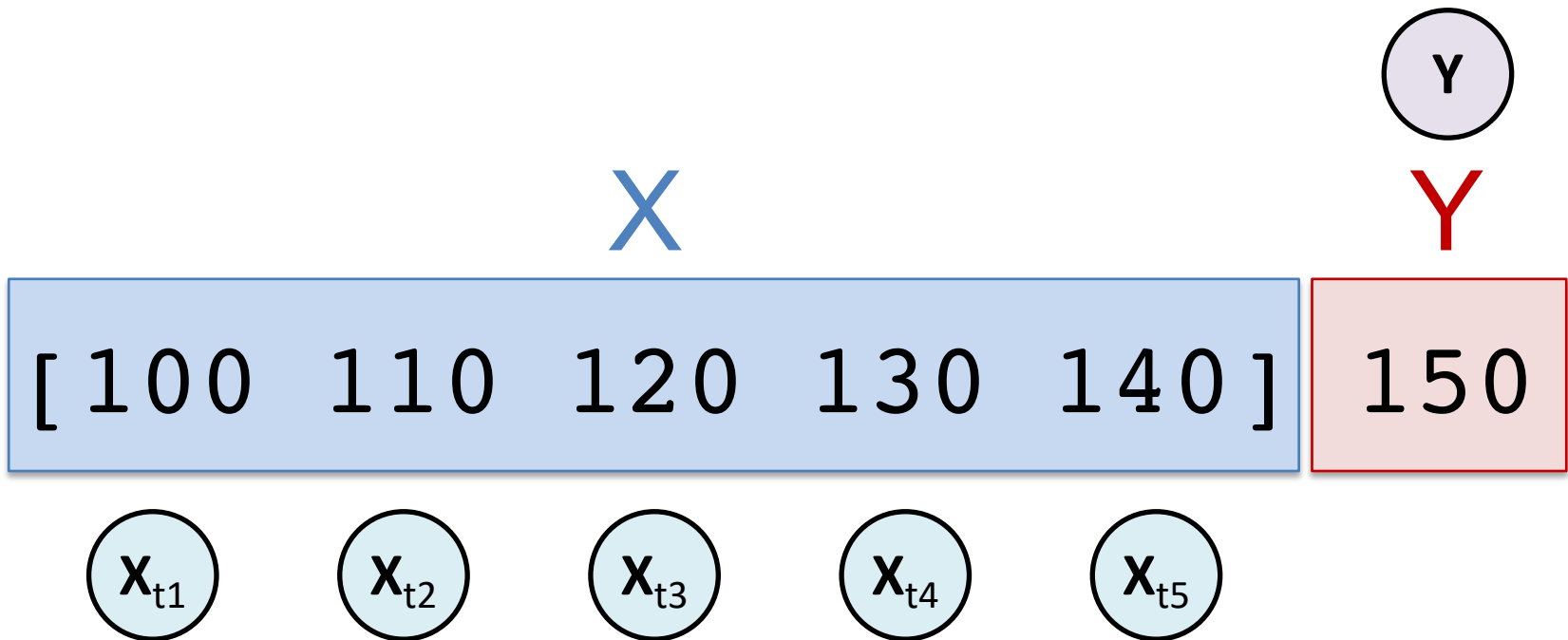
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

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

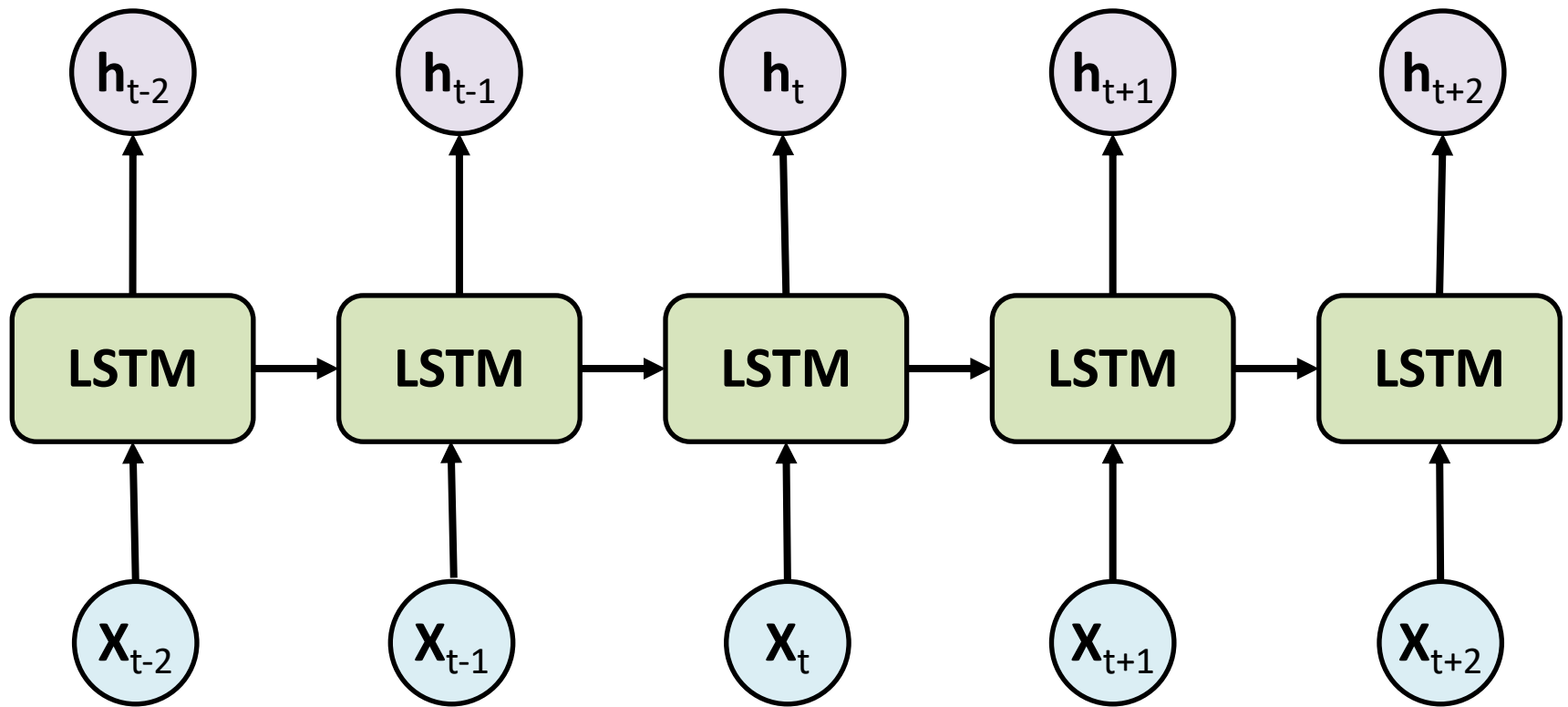


Time Series Data

[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

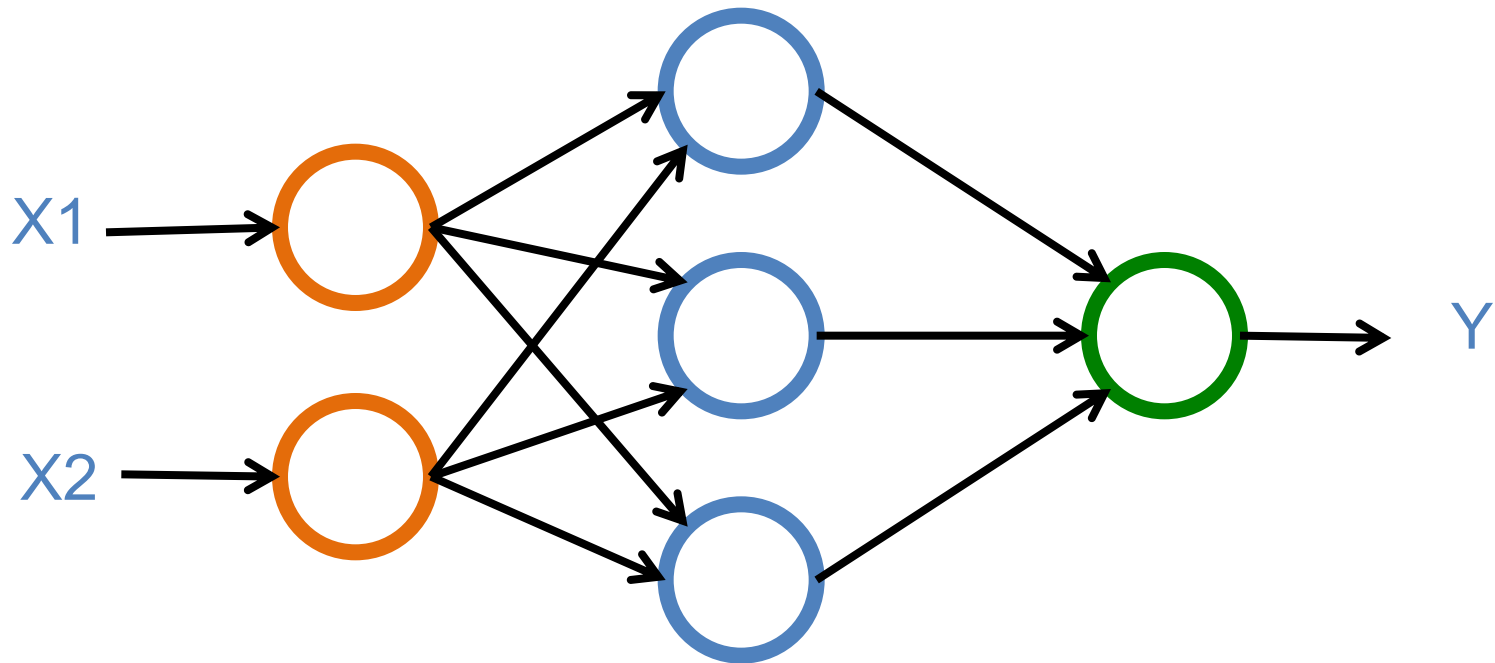
Deep Learning Foundations: Neural Networks

Deep Learning and Neural Networks

**Input Layer
(X)**

**Hidden Layer
(H)**

**Output Layer
(Y)**

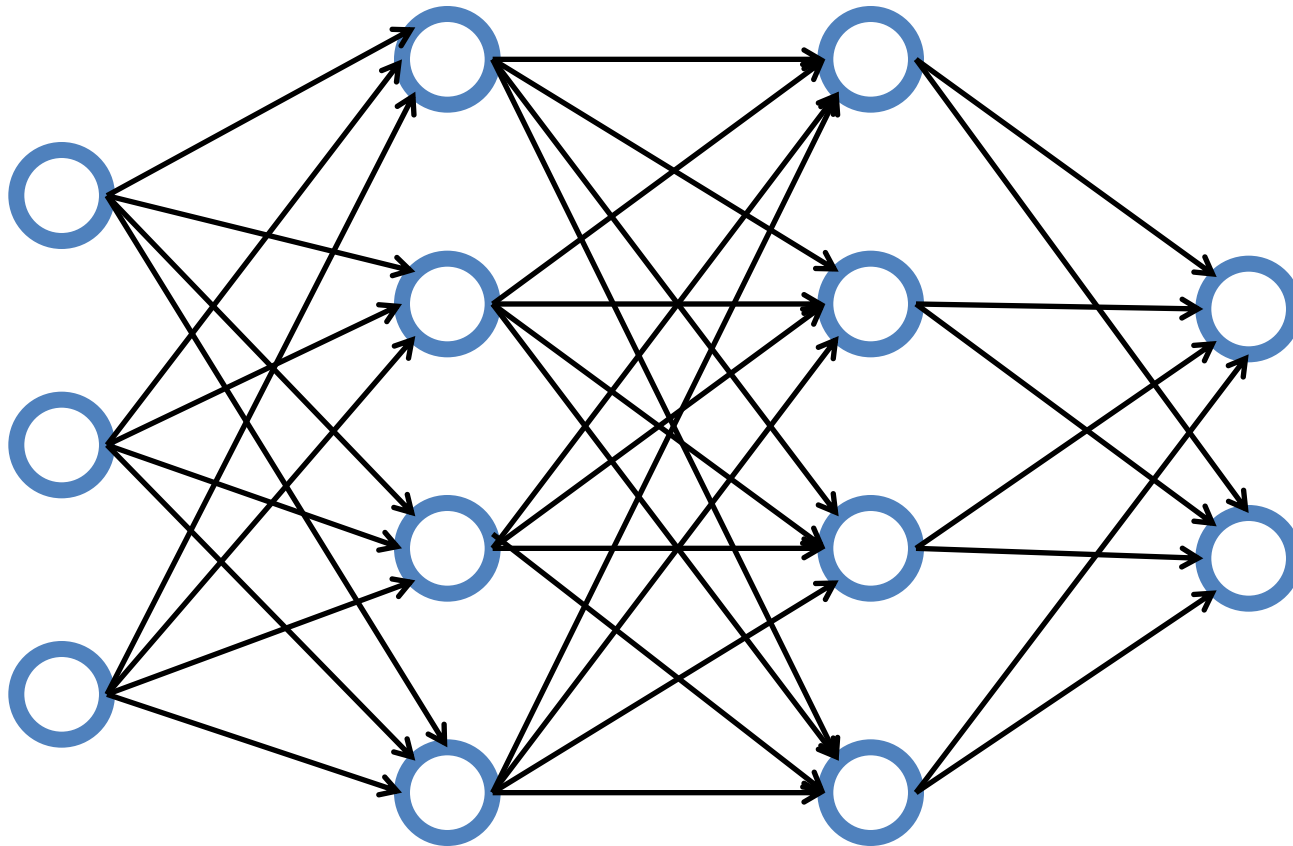


Deep Learning and Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



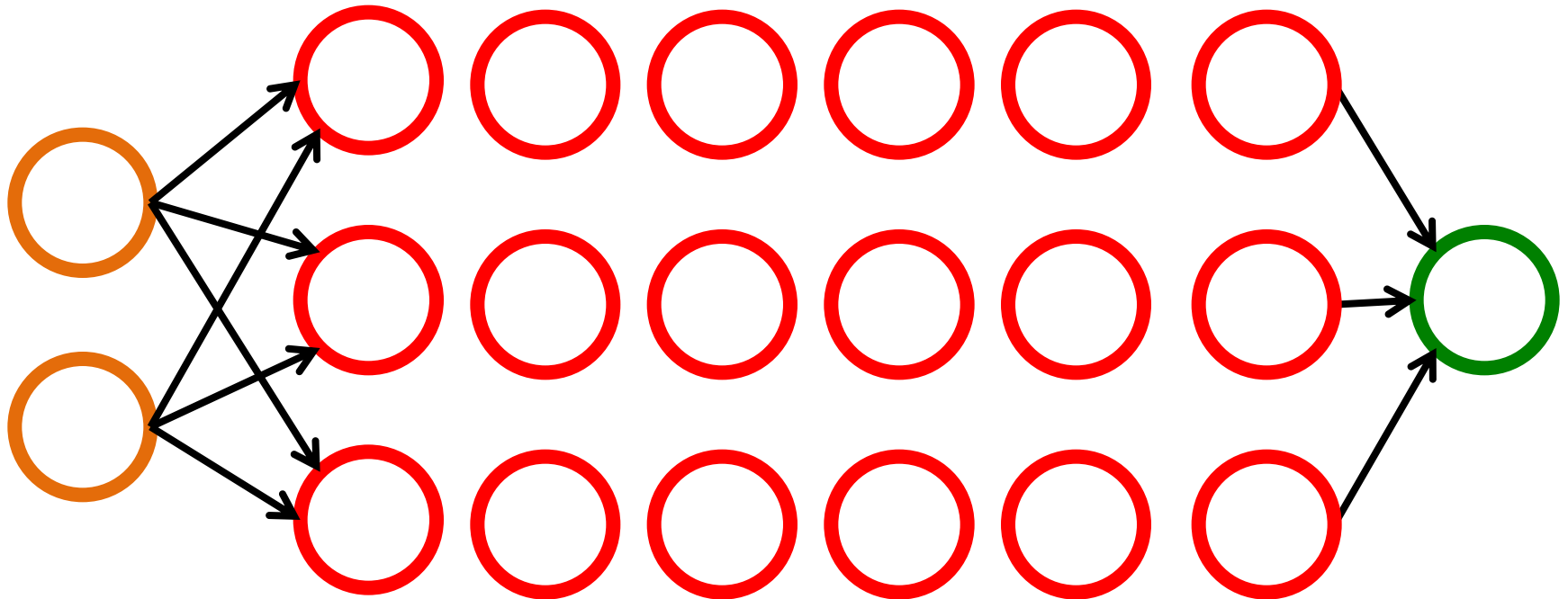
Deep Learning and Neural Networks

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.**

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & 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.

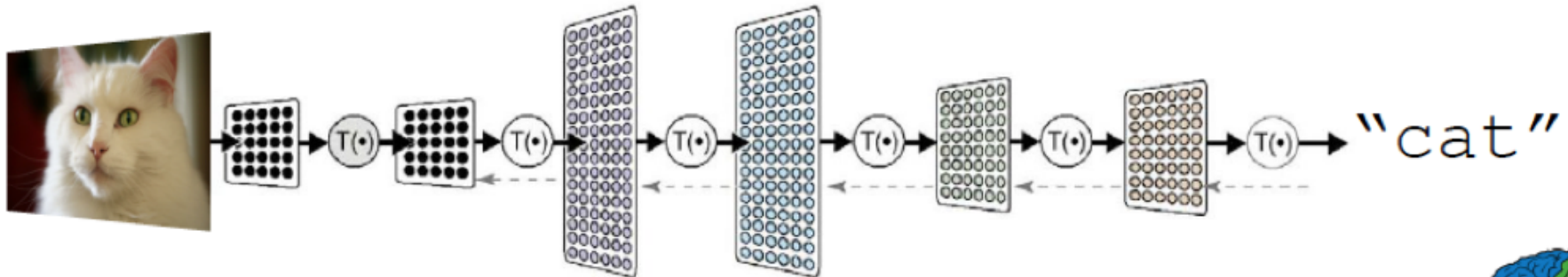
Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects 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 recognition¹⁻⁴ and speech recognition⁵⁻⁷, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules⁸, analysing particle accelerator data^{9,10}, reconstructing brain circuits¹¹, 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 understanding¹⁴, particularly topic classification, sentiment analysis, question answering¹⁵ and language translation^{16,17}.

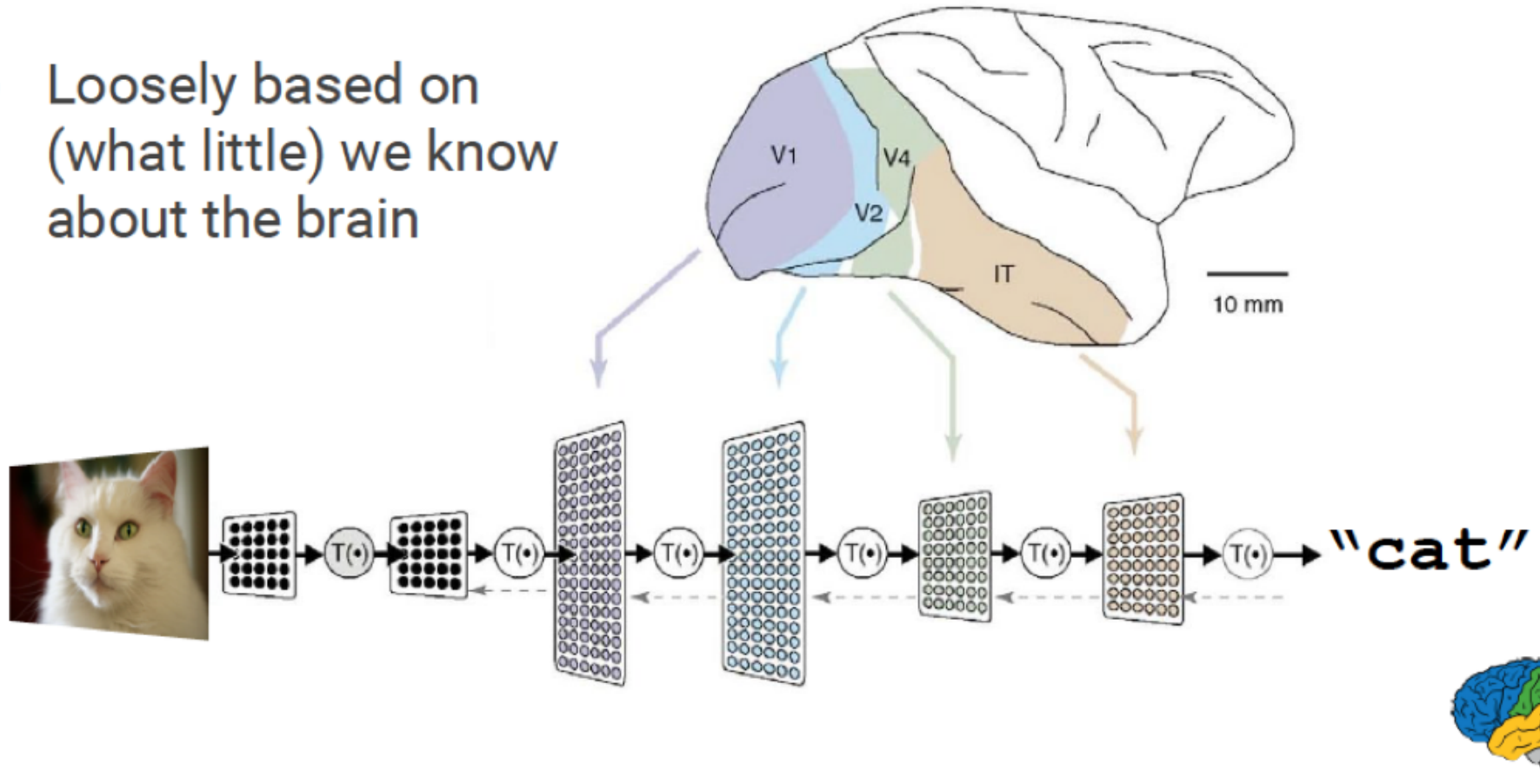
Deep Learning

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



What is Deep Learning?

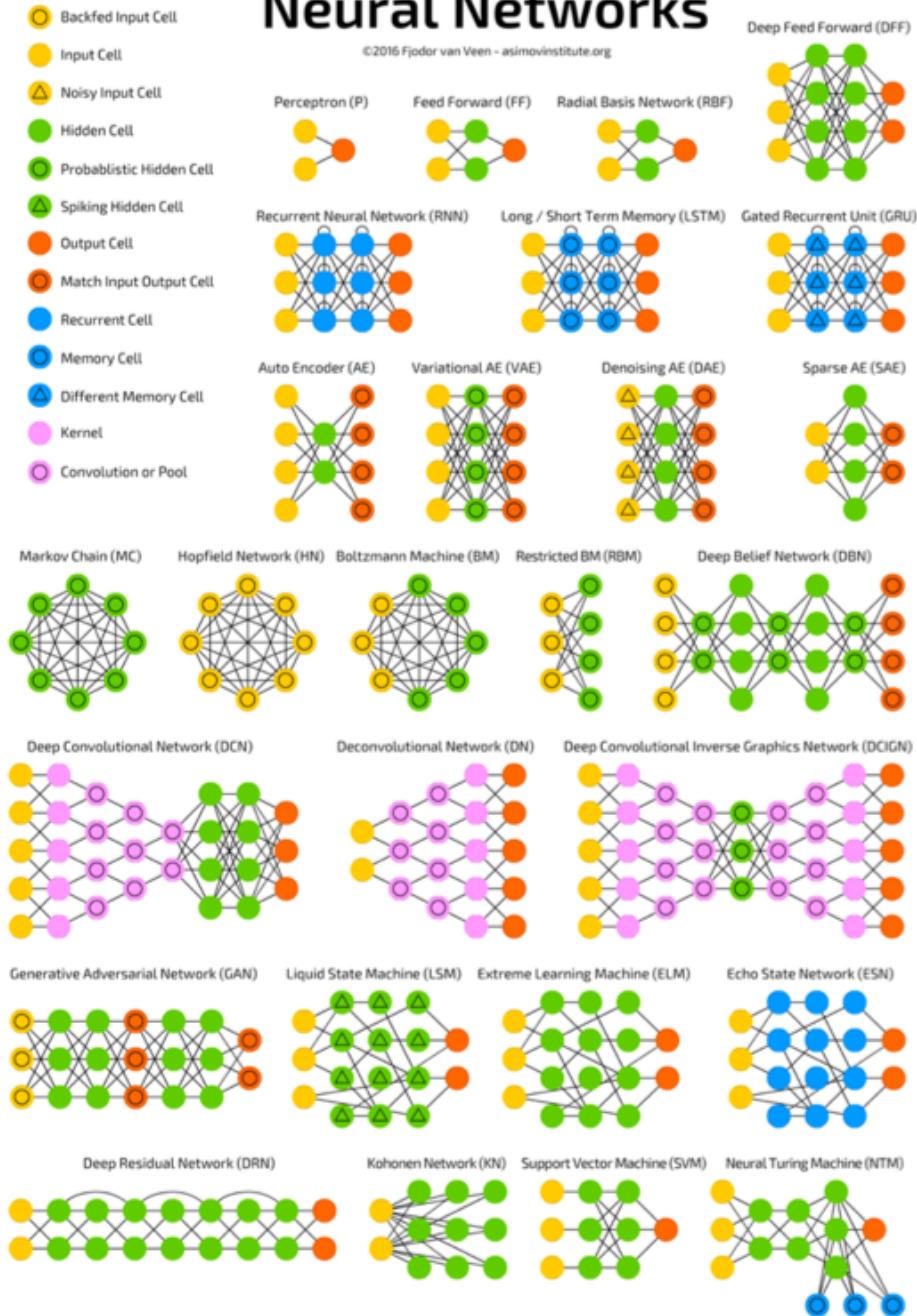
- Loosely based on (what little) we know about the brain



Neural Networks (NN)

A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org



Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

Deep Feed Forward (DFF)



Perceptron (P)



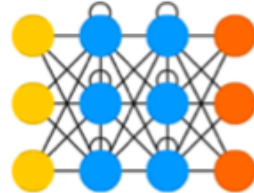
Feed Forward (FF)



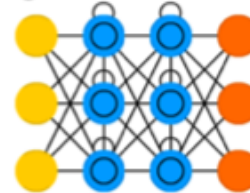
Radial Basis Network (RBF)



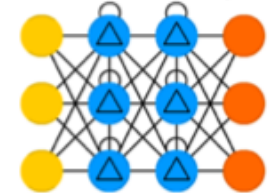
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



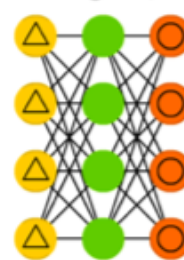
Auto Encoder (AE)



Variational AE (VAE)



Denosing AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



Hopfield Network (HN)



Boltzmann Machine (BM)



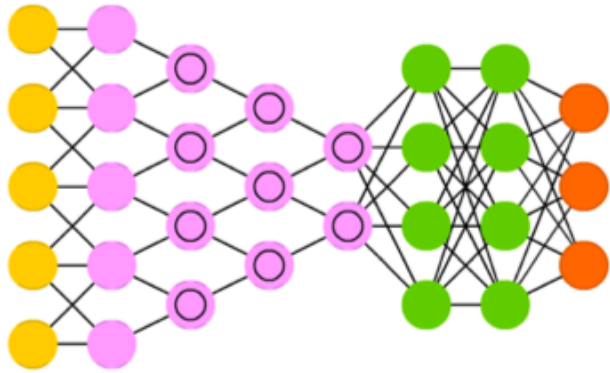
Restricted BM (RBM)



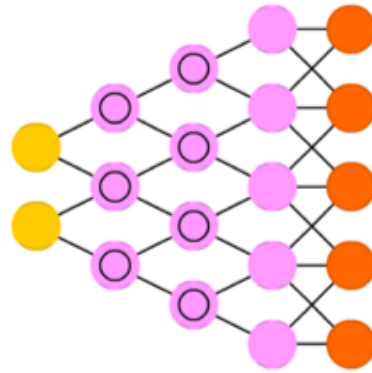
Deep Belief Network (DBN)



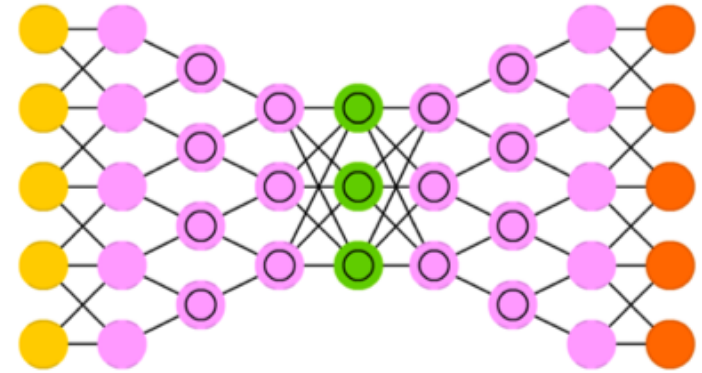
Deep Convolutional Network (DCN)



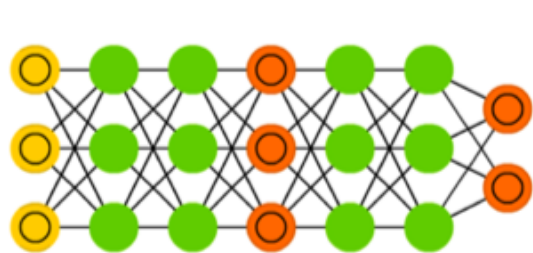
Deconvolutional Network (DN)



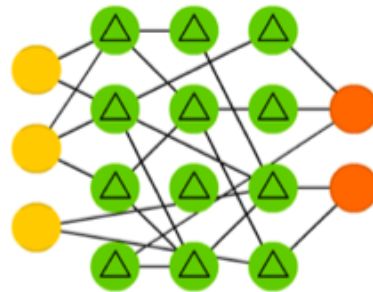
Deep Convolutional Inverse Graphics Network (DCIGN)



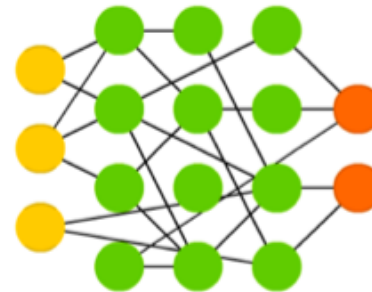
Generative Adversarial Network (GAN)



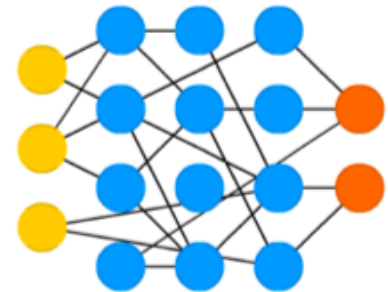
Liquid State Machine (LSM)



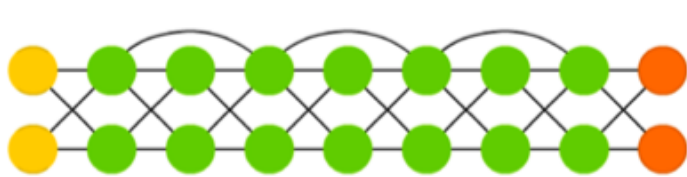
Extreme Learning Machine (ELM)



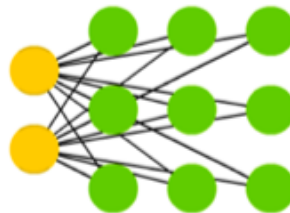
Echo State Network (ESN)



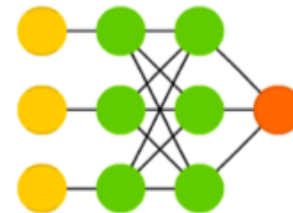
Deep Residual Network (DRN)



Kohonen Network (KN)



Support Vector Machine (SVM)

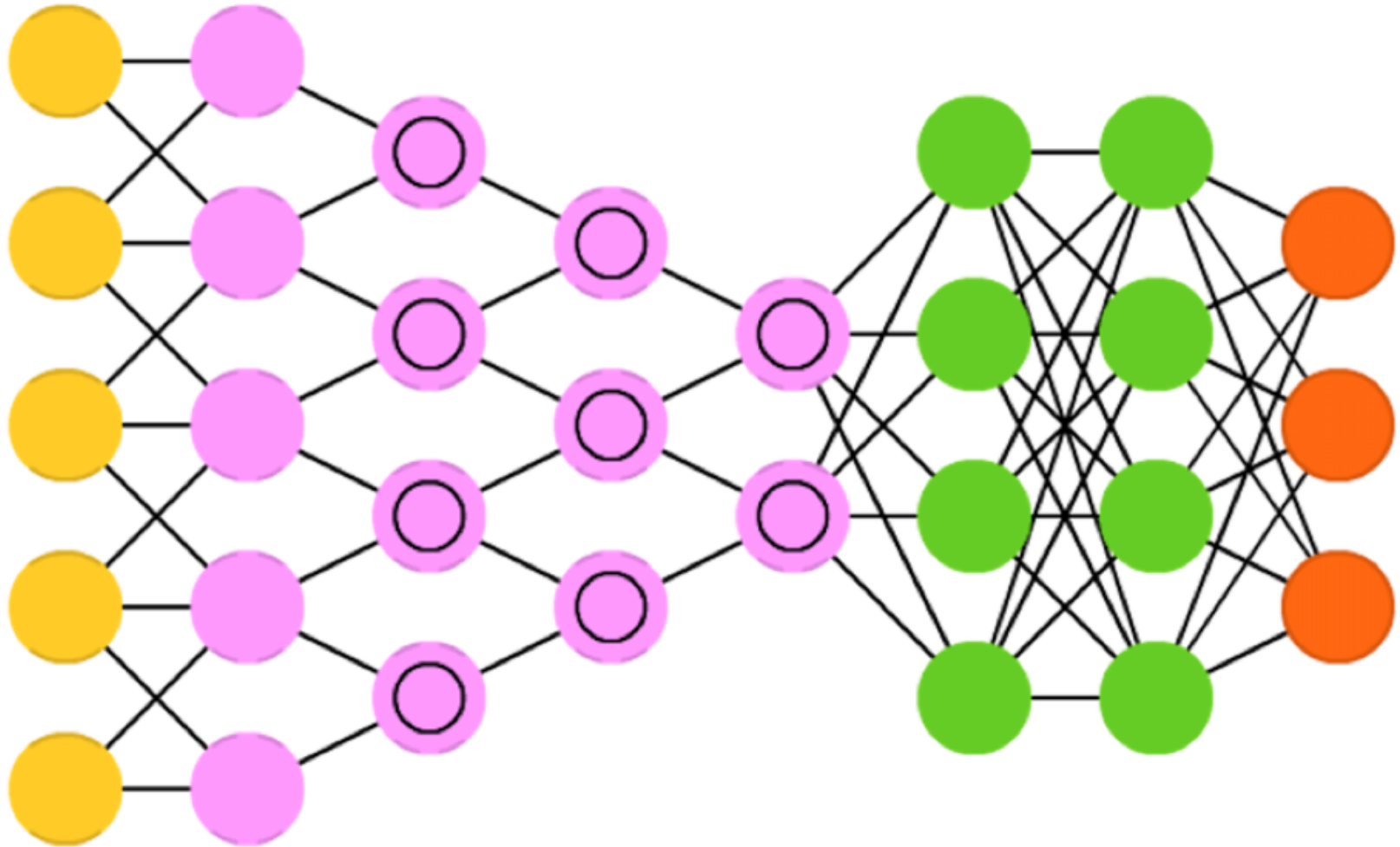


Neural Turing Machine (NTM)

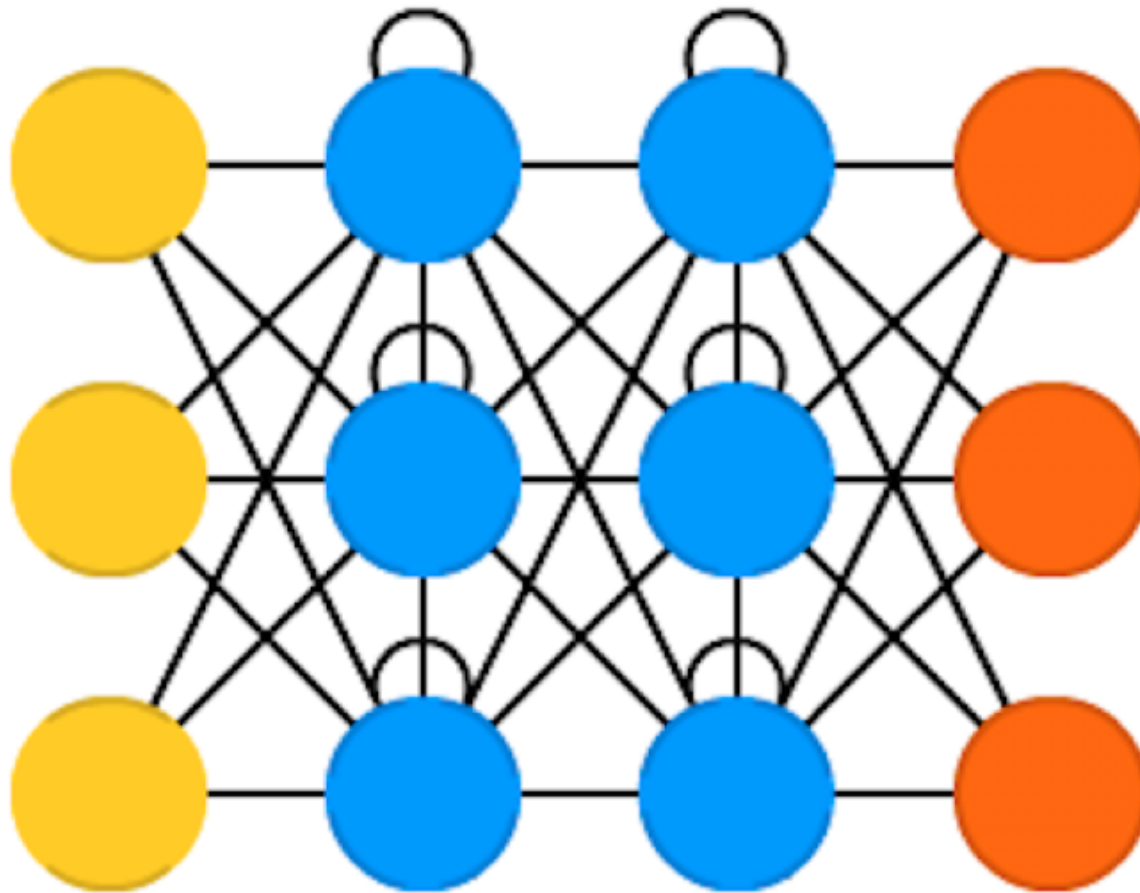


Convolutional Neural Networks

(CNN or Deep Convolutional Neural Networks, DCNN)



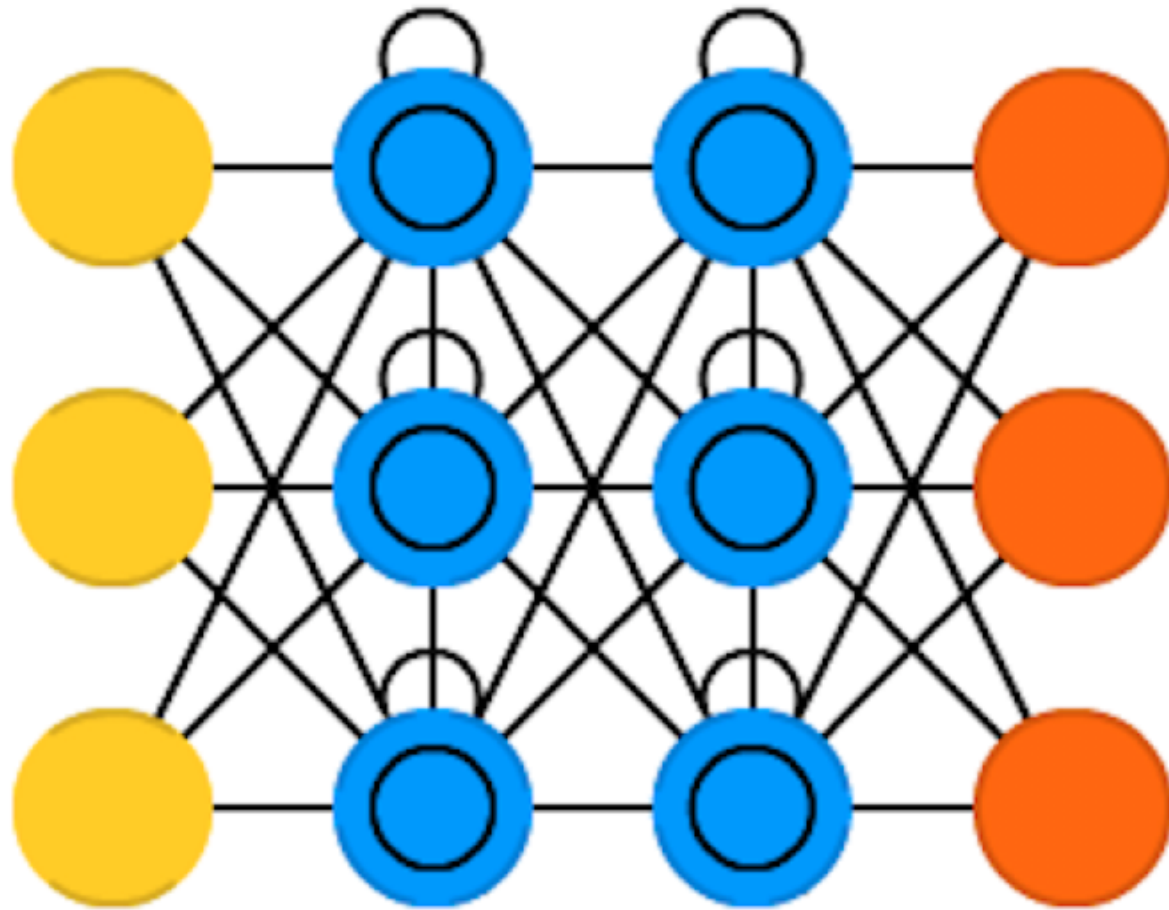
Recurrent Neural Networks (RNN)



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

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

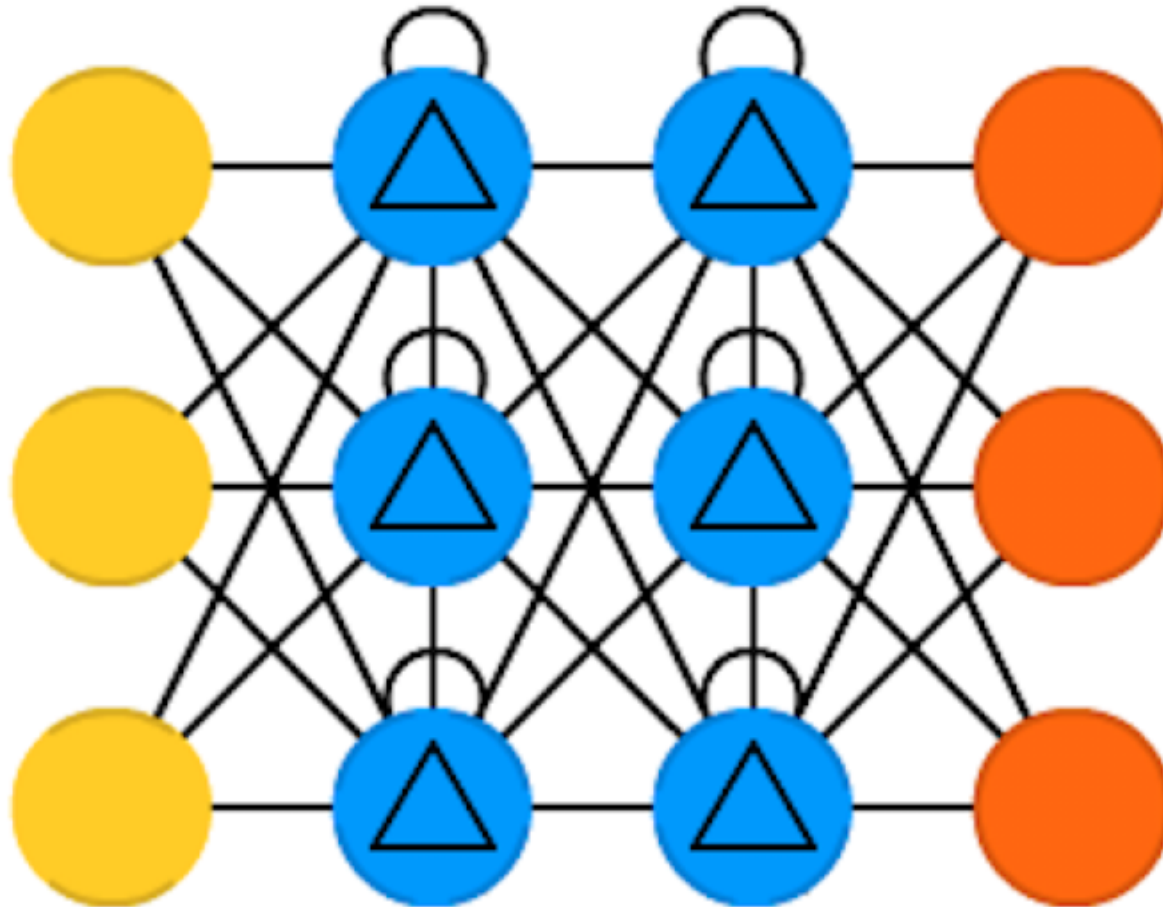
Long / Short Term Memory (LSTM)



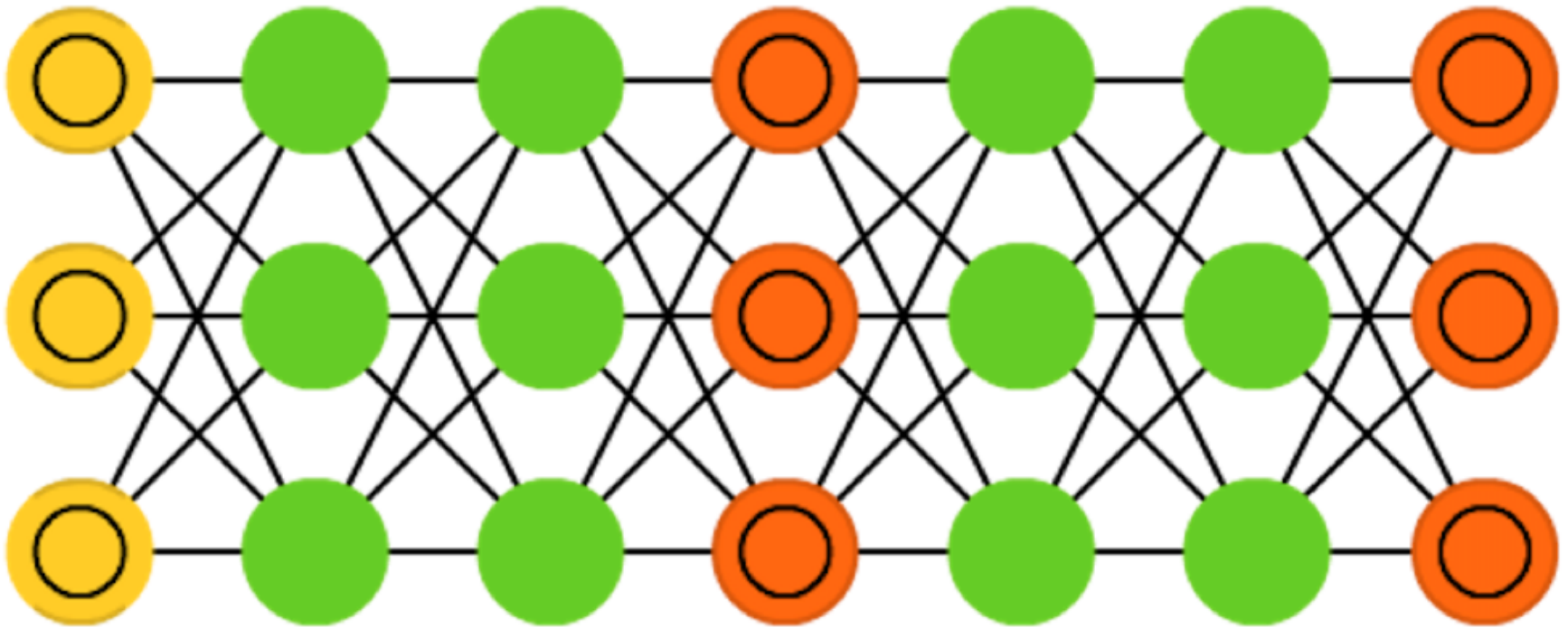
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

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

Gated Recurrent Units (GRU)



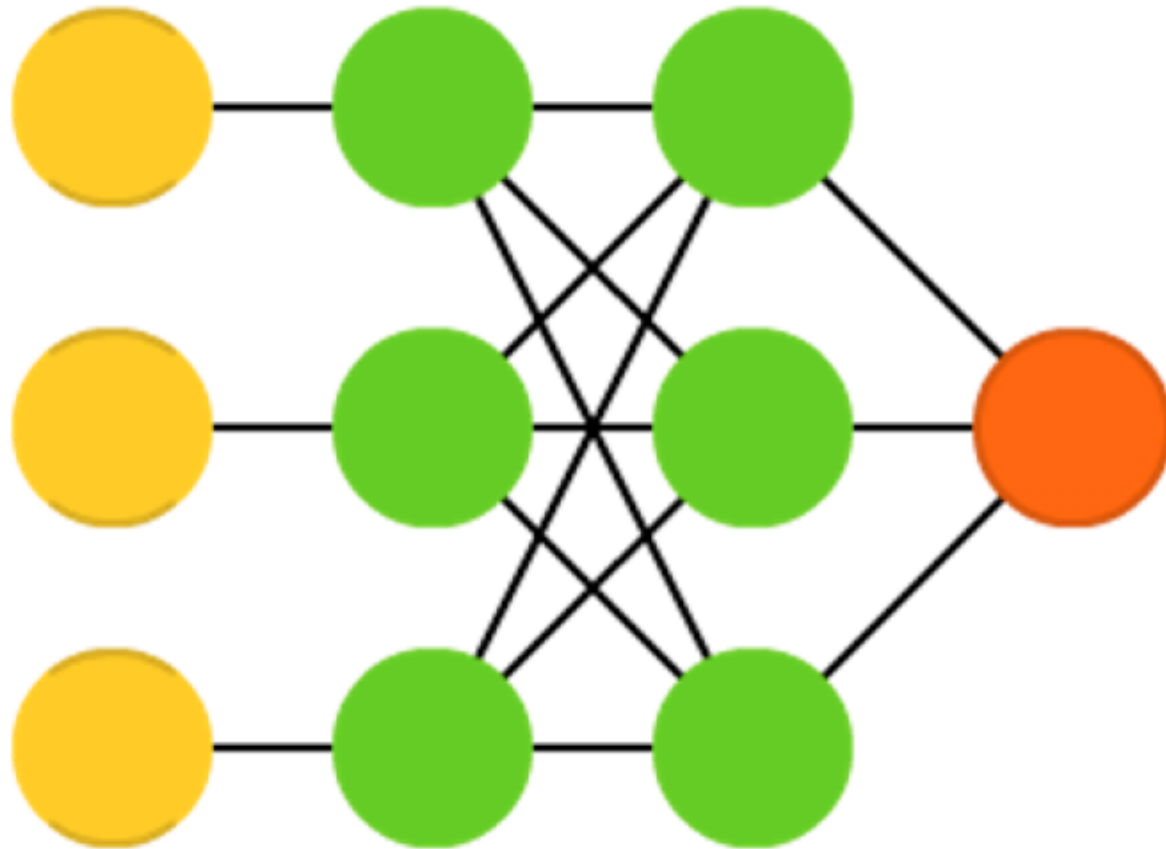
Generative Adversarial Networks (GAN)



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/>

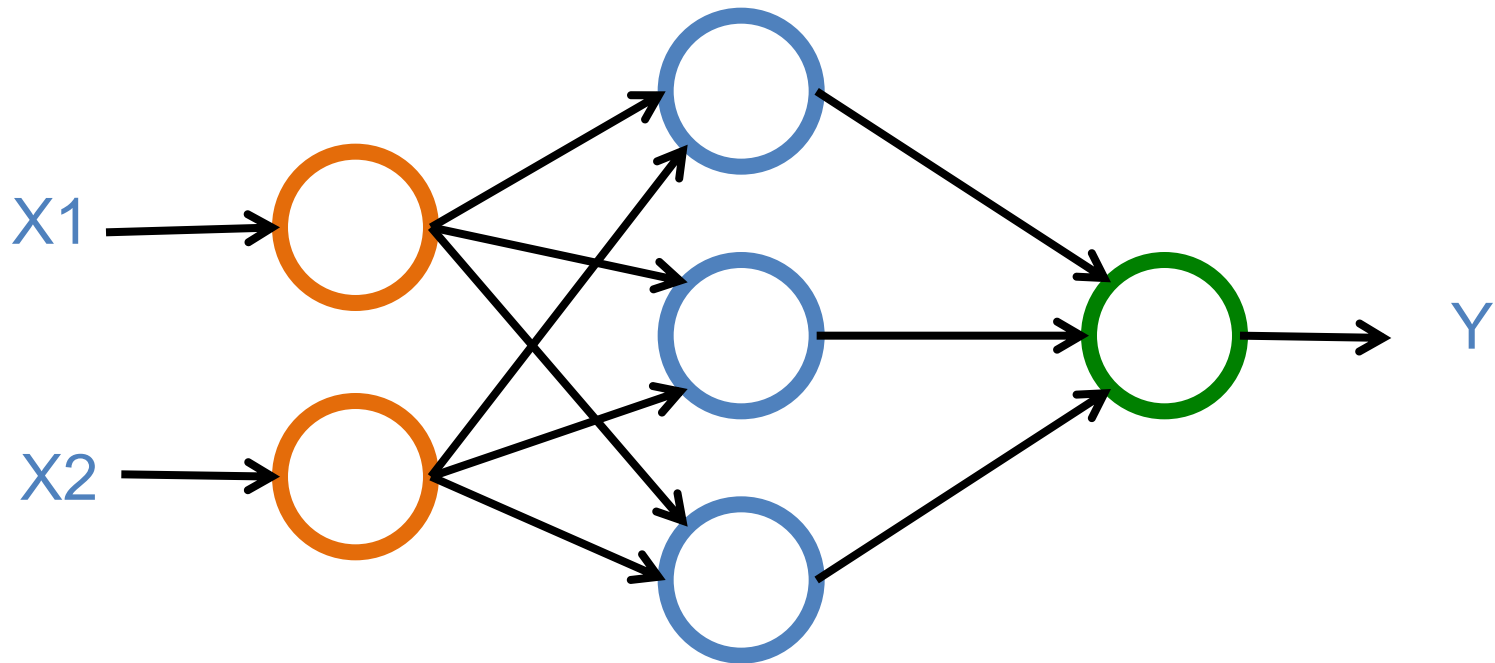
Neural networks (NN) 1960

Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



Multilayer Perceptrons (MLP) 1985

Support Vector Machine (SVM) 1995



Hinton presents the

Deep Belief Network (DBN)

**New interests in deep learning
and RBM**

State of the art MNIST

2005

Deep Recurrent Neural Network (RNN) 2009

Convolutional DBN 2010

Max-Pooling CDBN 2011

Deep Learning

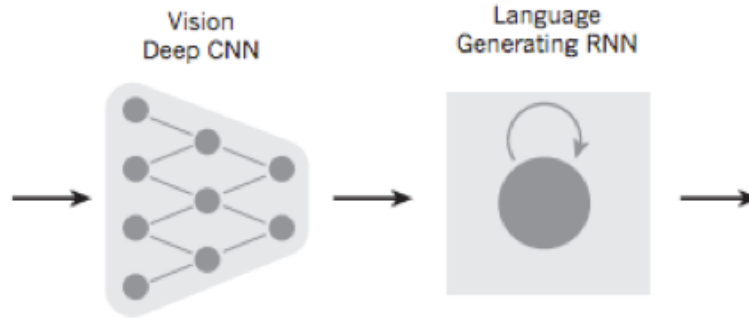
Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng

From image to text



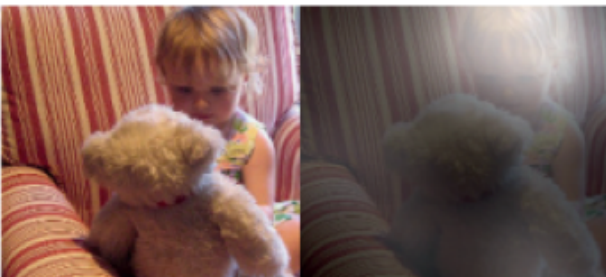
A woman is throwing a **frisbee** in a park.



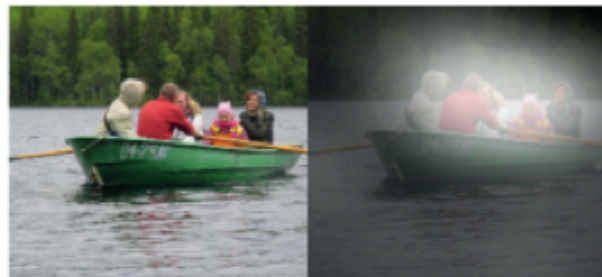
A **dog** is standing on a hardwood floor.



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



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



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



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

From image to text

Image: deep convolution neural network (CNN)

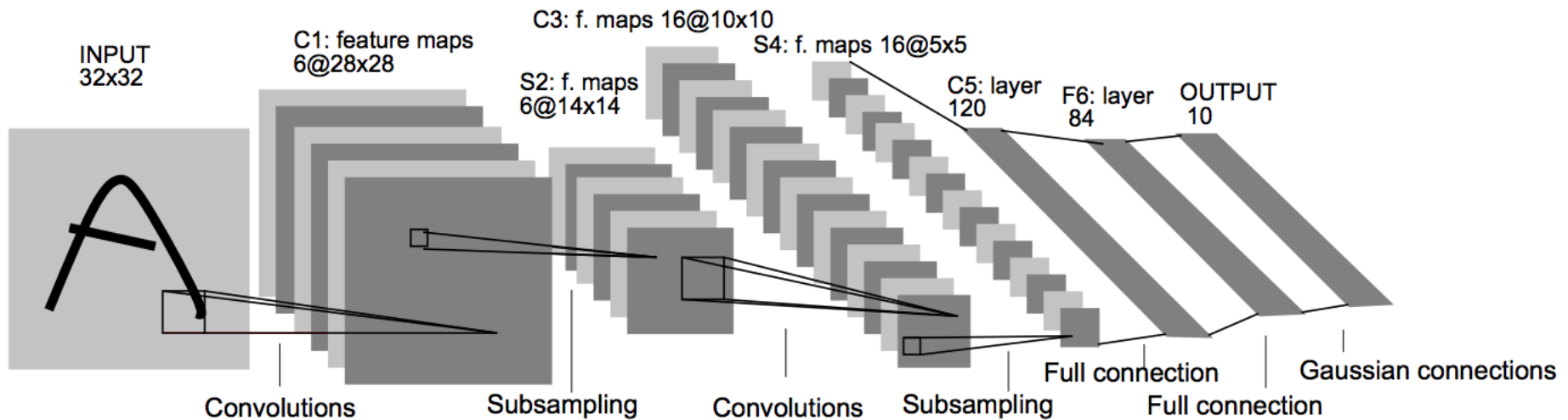
Text: recurrent neural network (RNN)



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

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>

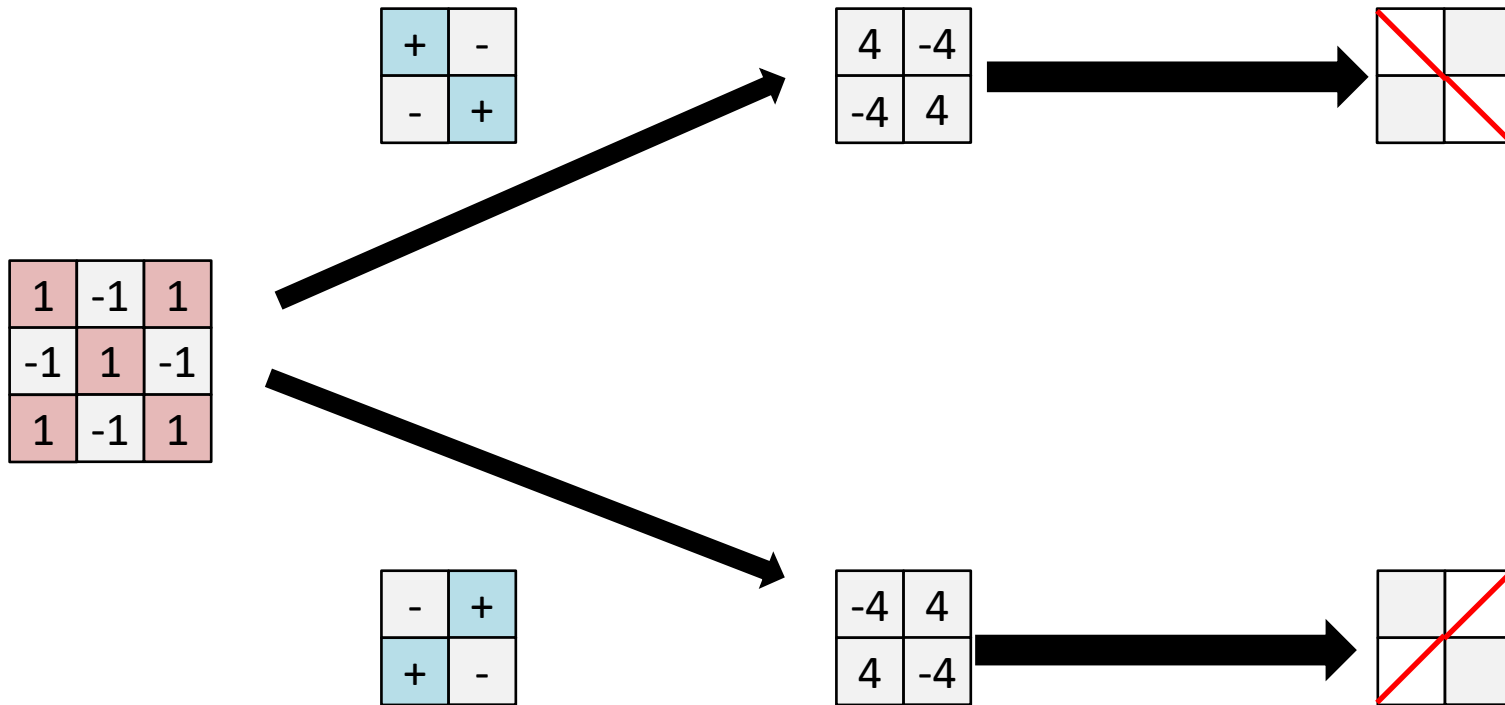
Source: LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner.

"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)

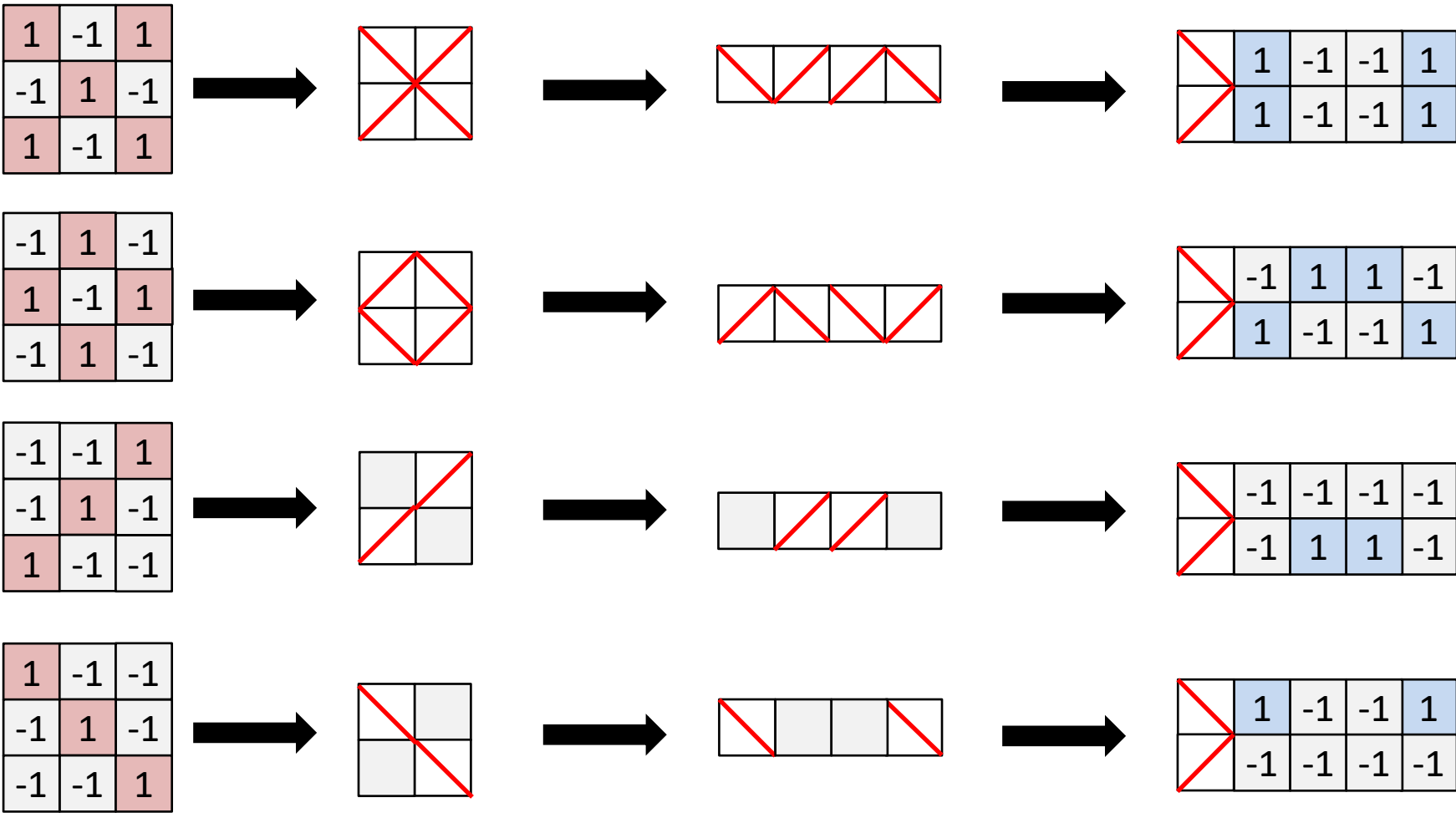
A friendly introduction to Convolutional Neural Networks and Image Recognition



Convolution Layer

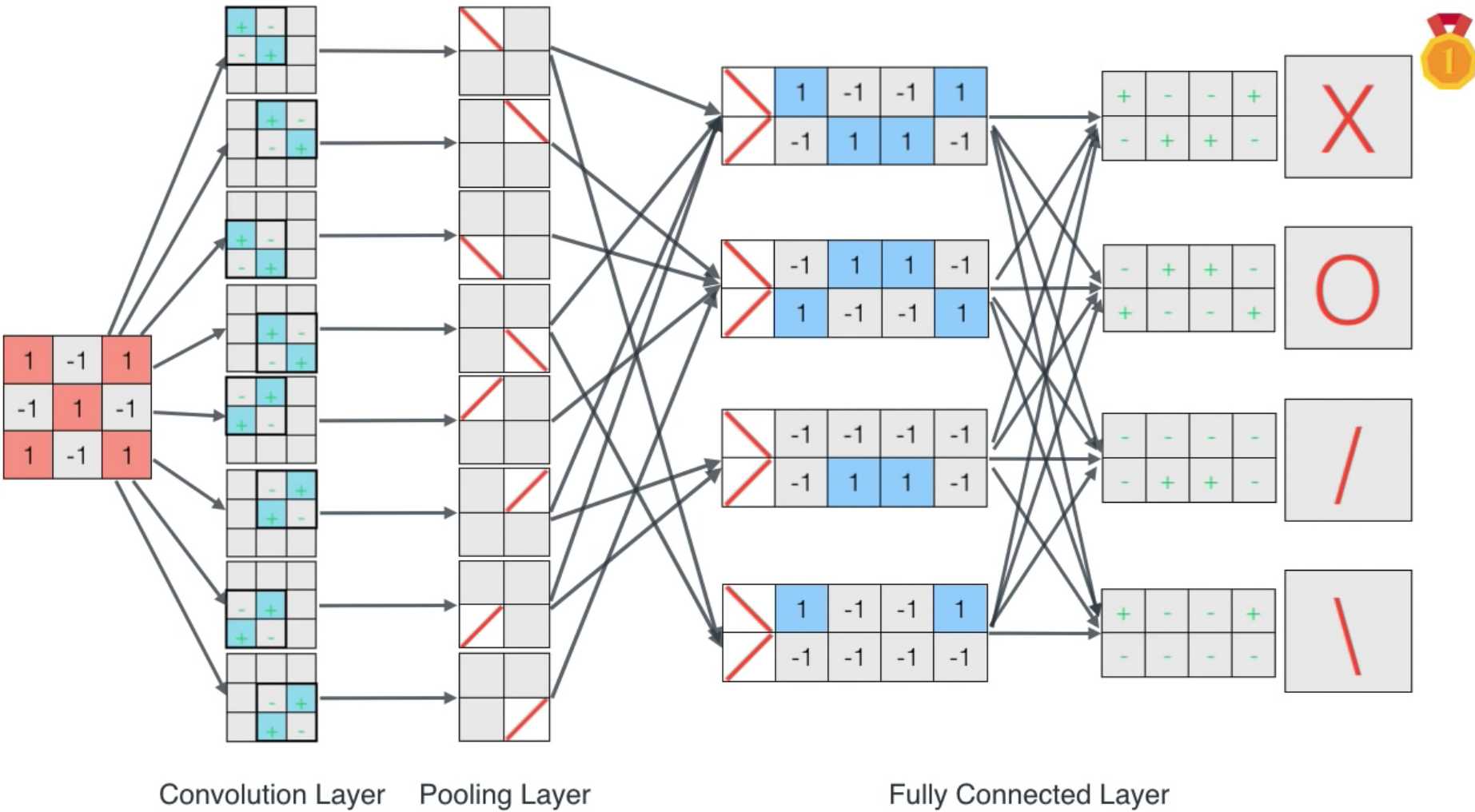
Pooling Layer

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-OI7ZB0MmU>

A friendly introduction to Convolutional Neural Networks and Image Recognition



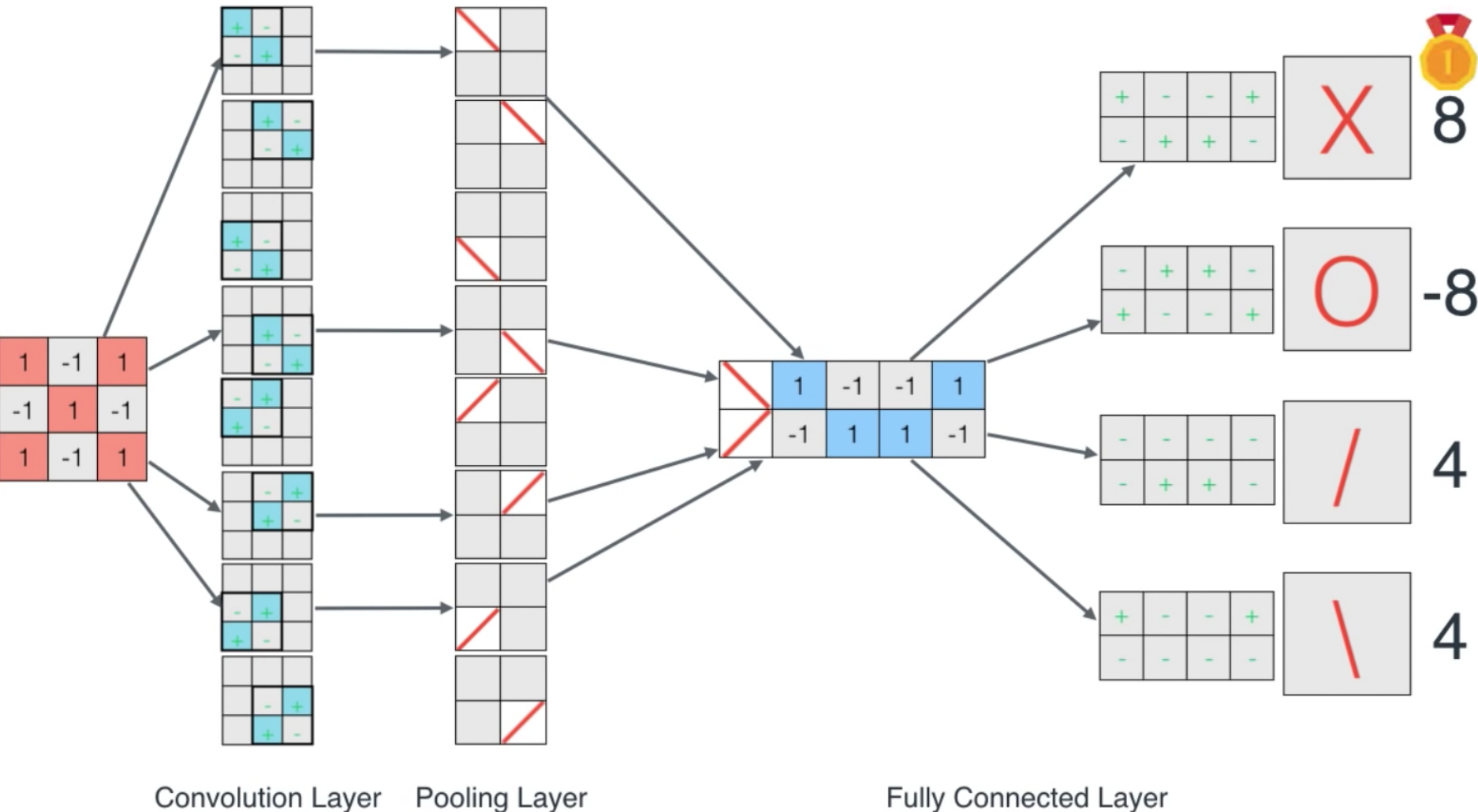
Convolution Layer

Pooling Layer

Fully Connected Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, <https://www.youtube.com/watch?v=2-OI7ZB0MmU>

A friendly introduction to Convolutional Neural Networks and Image Recognition



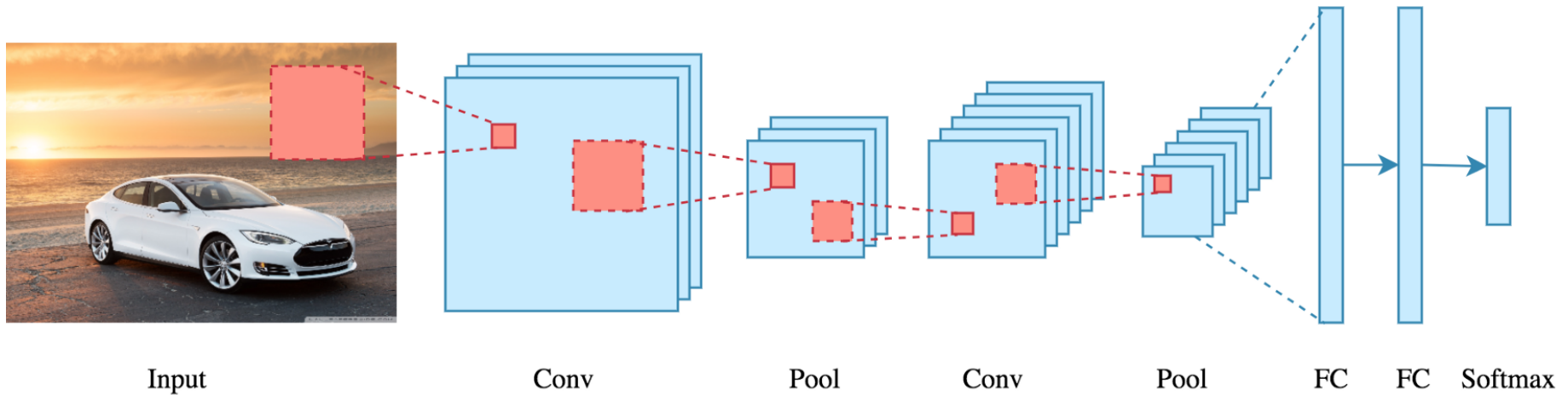
Convolution Layer

Pooling Layer

Fully Connected Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, <https://www.youtube.com/watch?v=2-OI7ZB0MmU>

CNN Architecture



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

CNN Convolution Layer

Input x Filter --> Feature Map

receptive field: 3x3

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4		

Feature Map

CNN Convolution Layer

Input x Filter --> Feature Map

receptive field: 3x3

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4	3	

Feature Map

CNN Convolution Layer

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

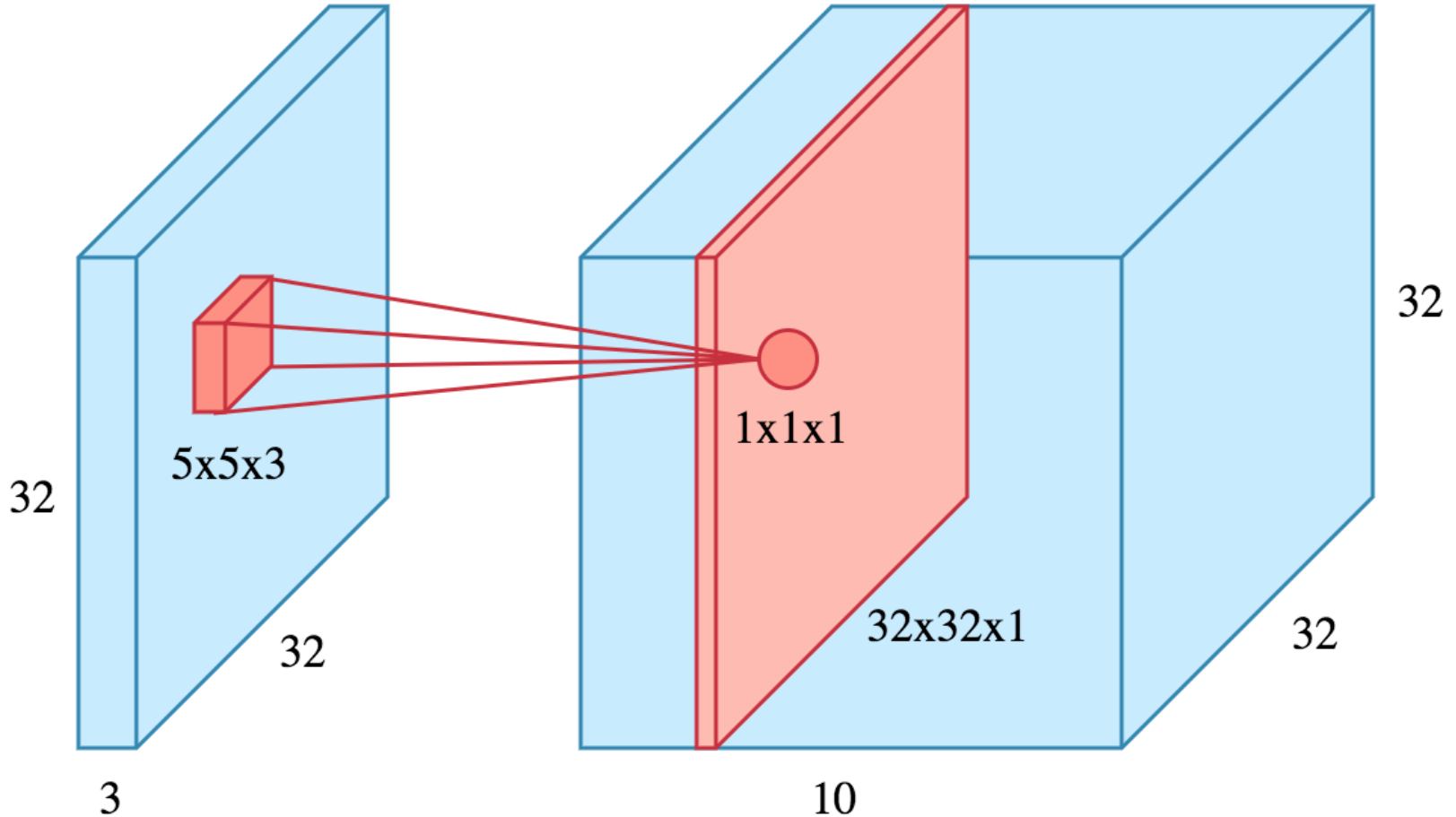
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

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

CNN Convolution Layer

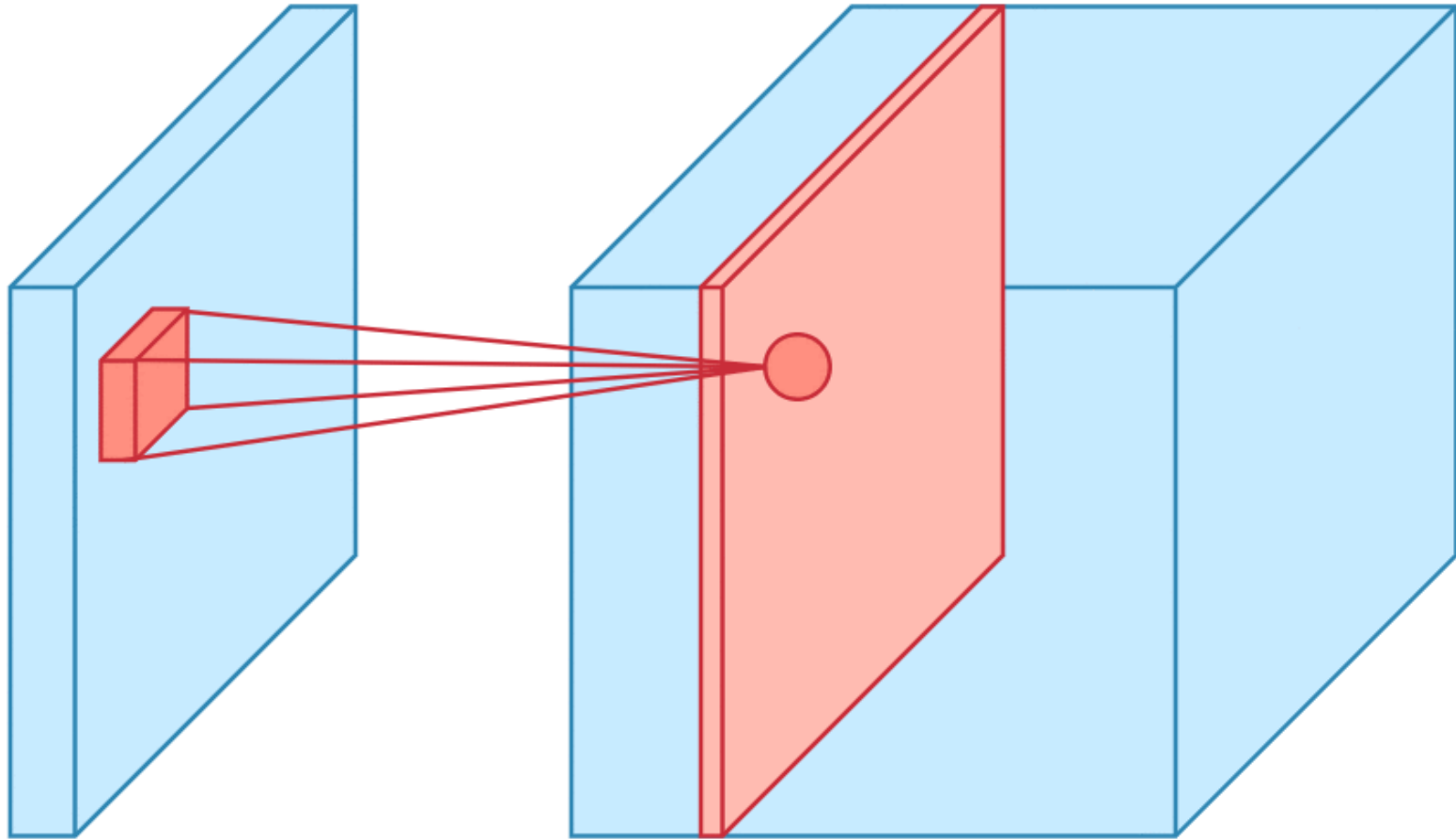
10 different filters 10 feature maps of size 32x32x1



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

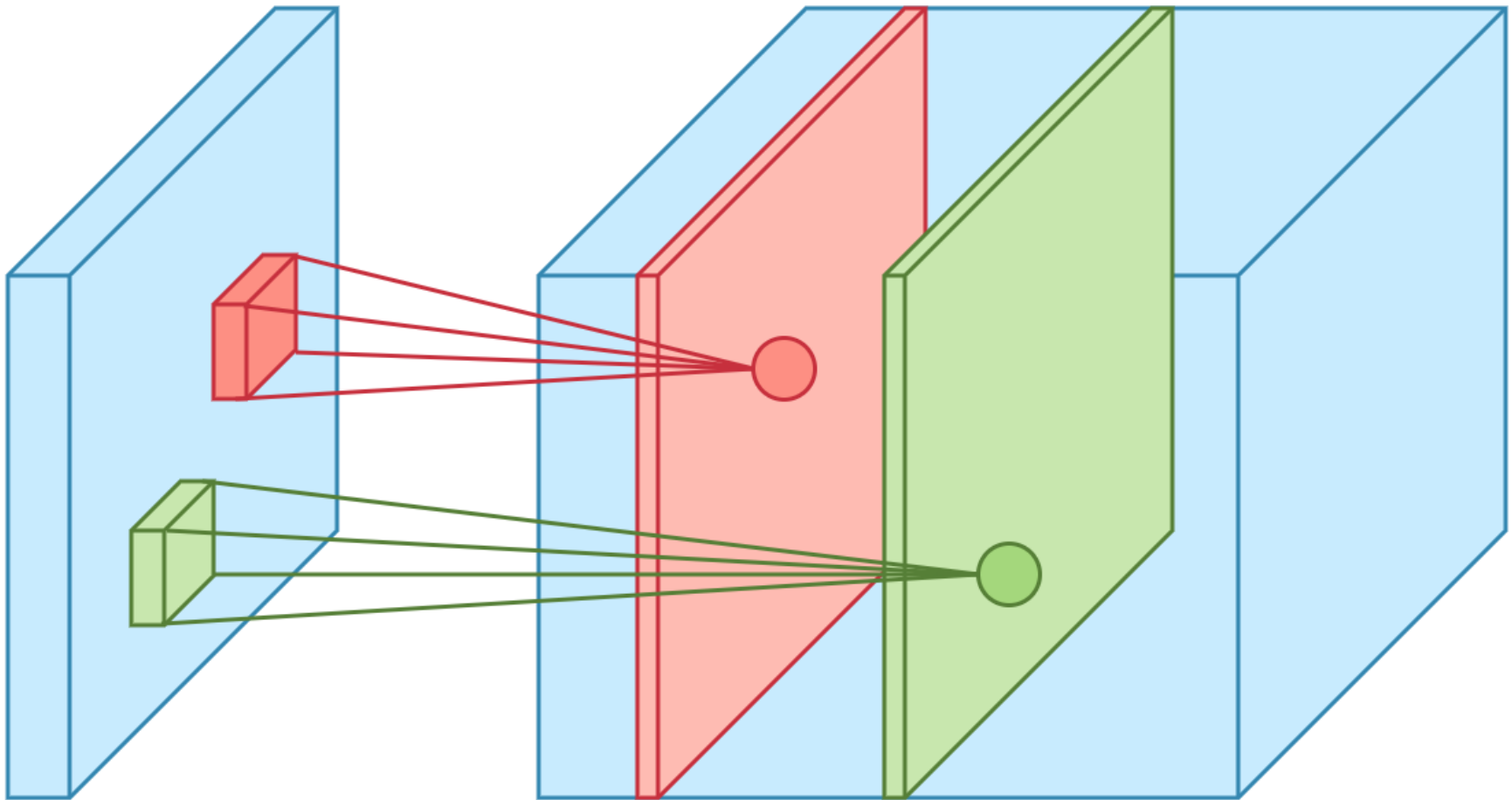
CNN Convolution Layer

Sliding operation at 4 locations



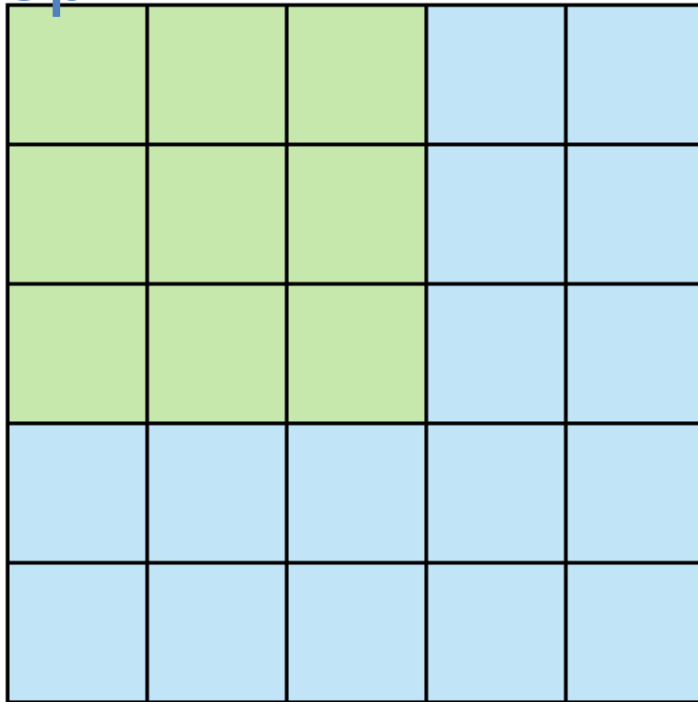
CNN Convolution Layer

two feature maps

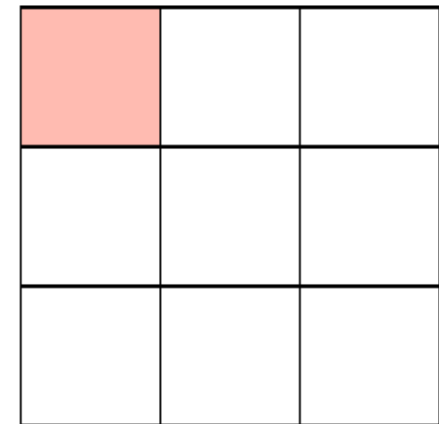


CNN Convolution Layer

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



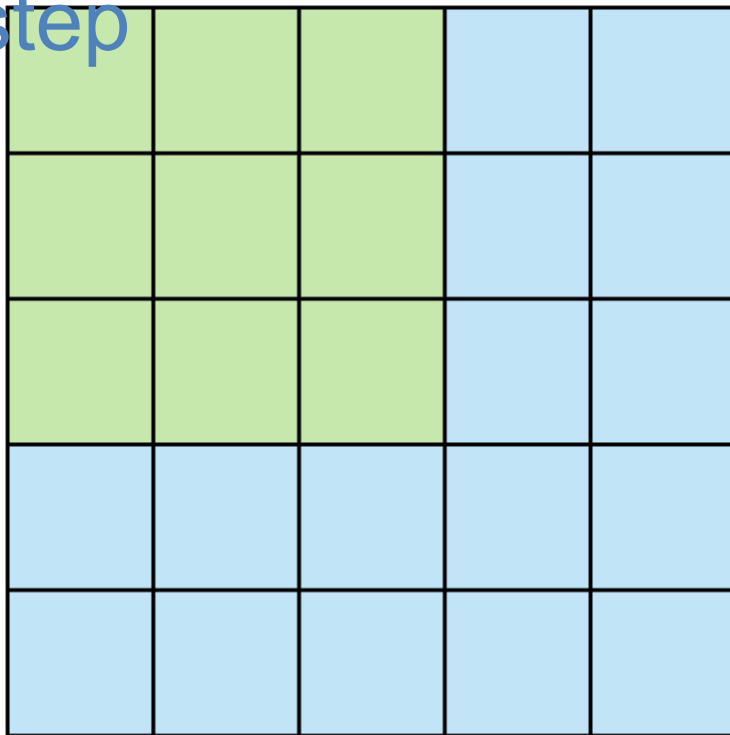
Stride 1



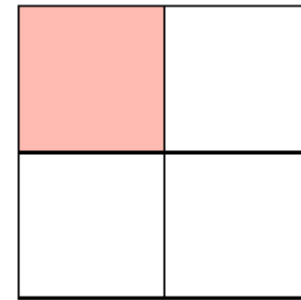
Feature Map

CNN Convolution Layer

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



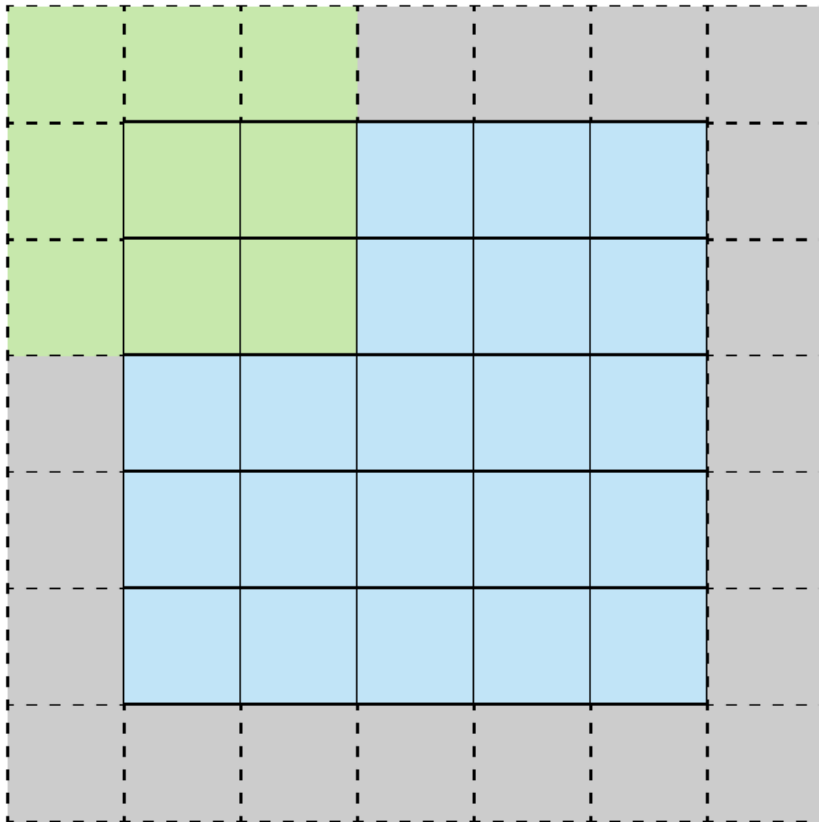
Stride 2



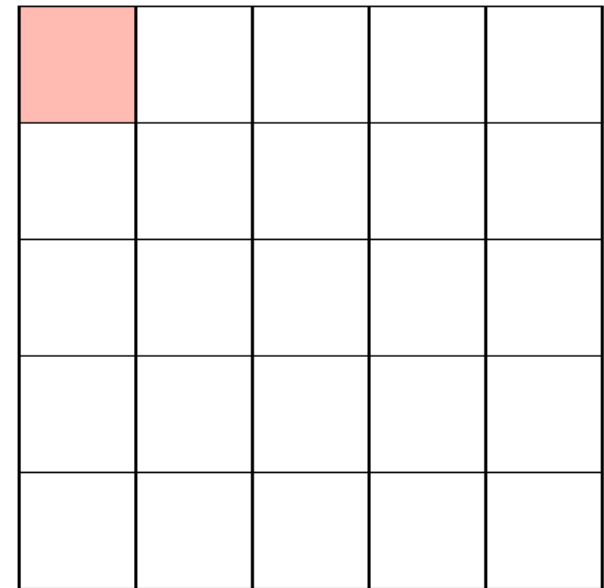
Feature Map

CNN Convolution Layer

Stride 1 with Padding



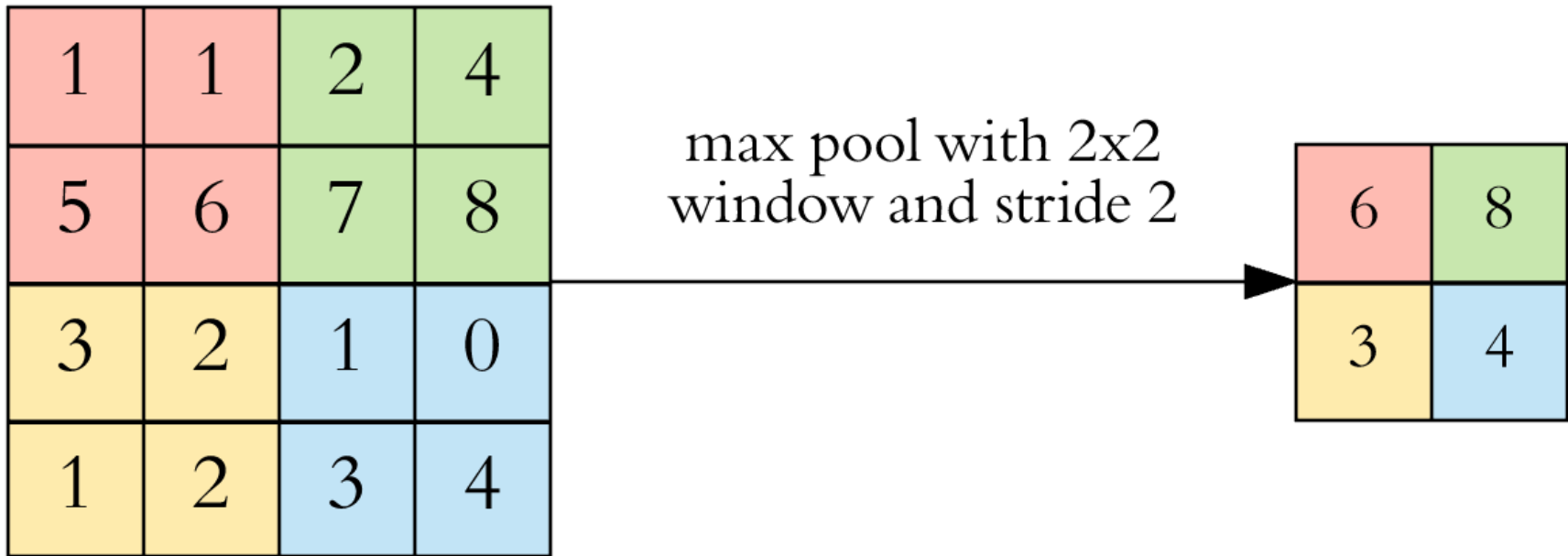
Stride 1 with Padding



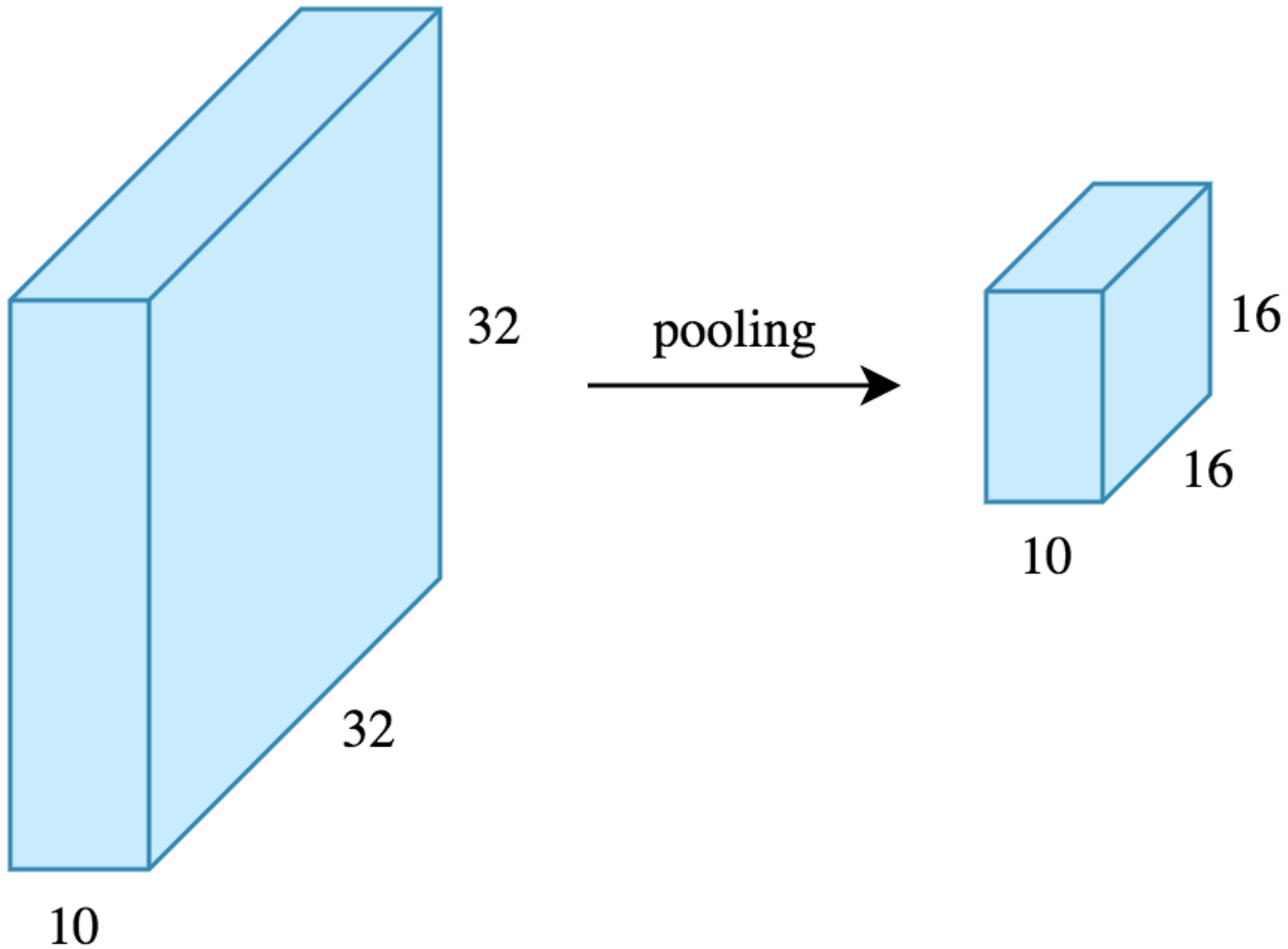
Feature Map

CNN Pooling Layer

Max Pooling

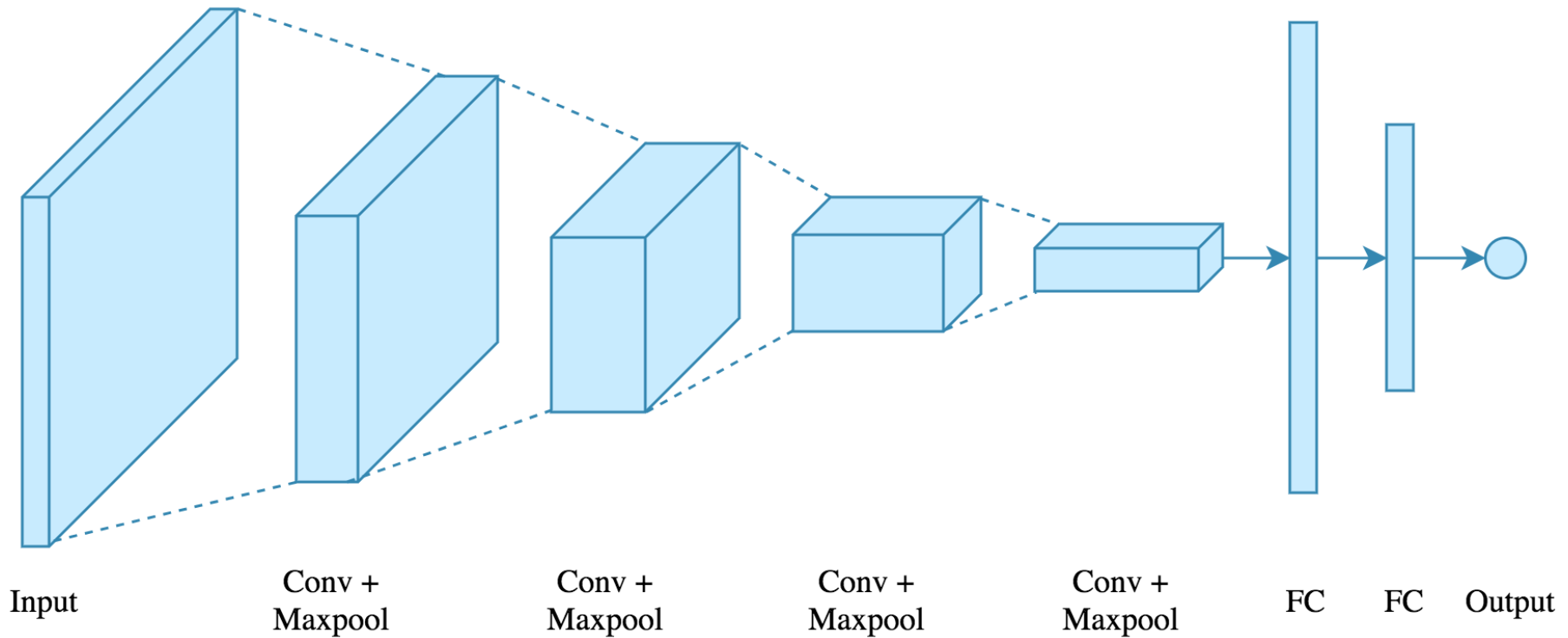


CNN Pooling Layer



CNN Architecture

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



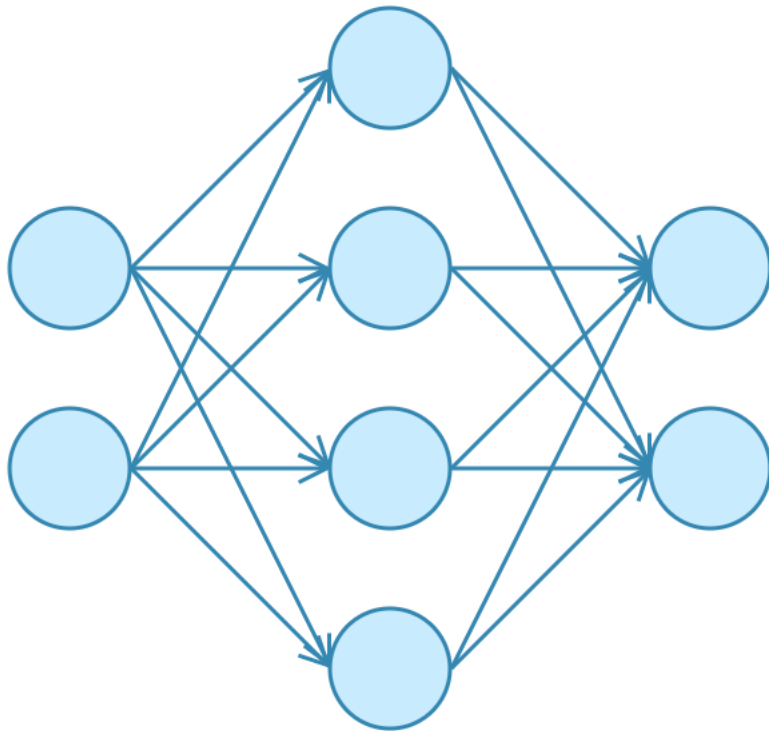
CNN Architecture

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

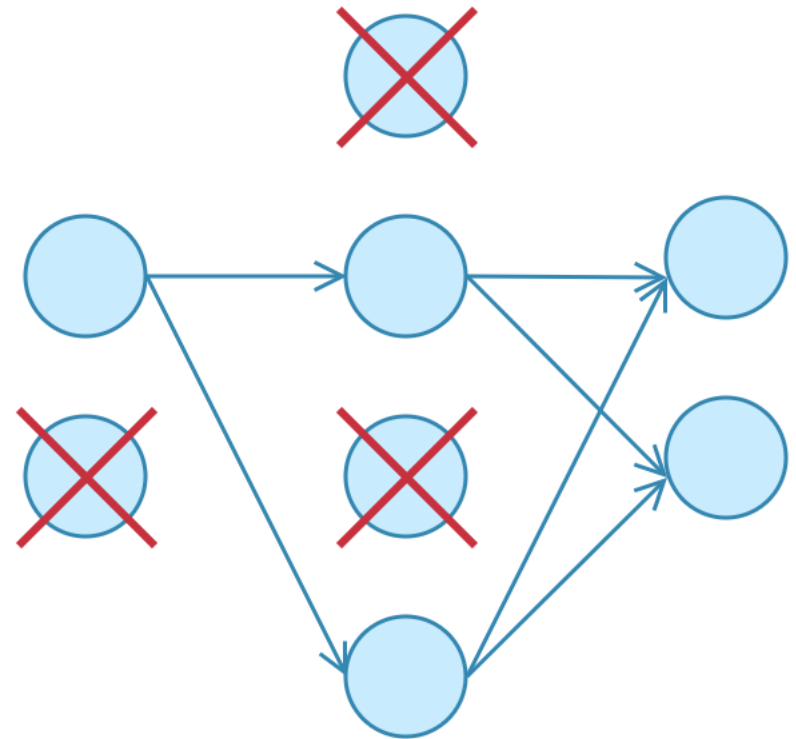
<https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3>

```
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'))
```

Dropout

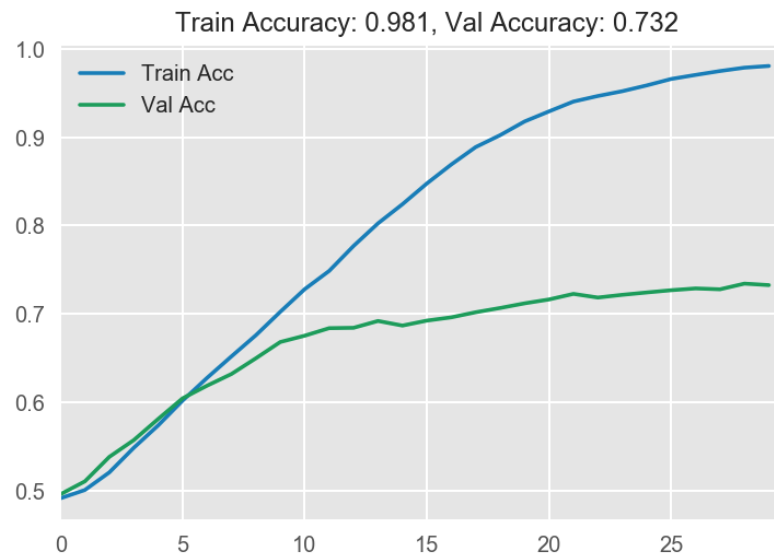
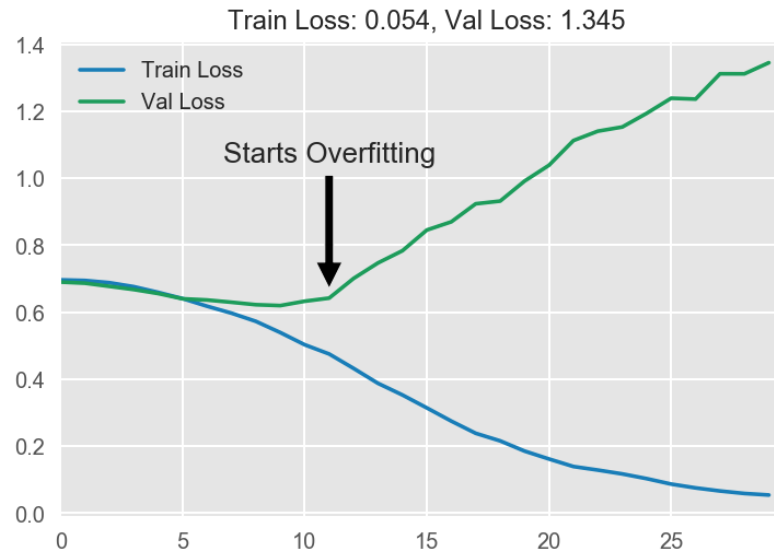


No Dropout



With Dropout

Model Performance



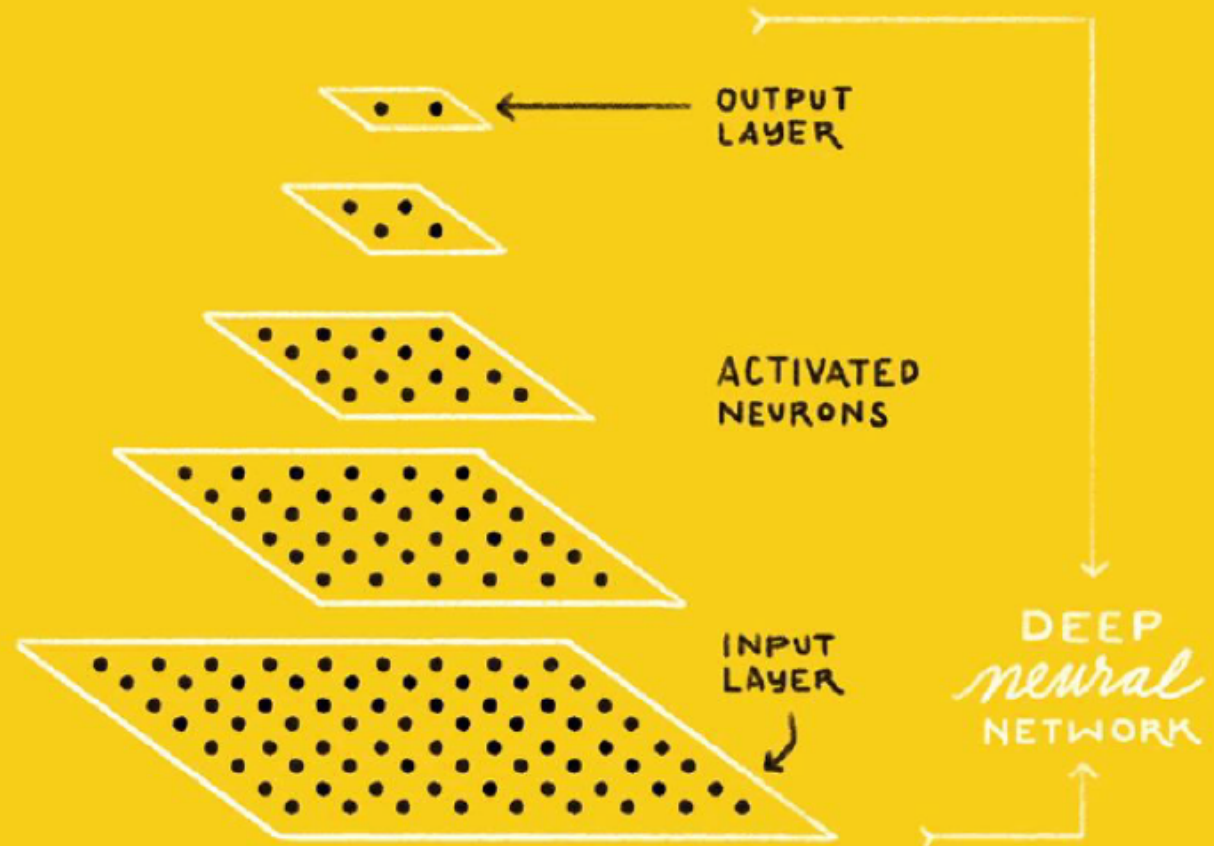
Visual Recognition

Image Classification

IS THIS A
CAT or **DOG**?

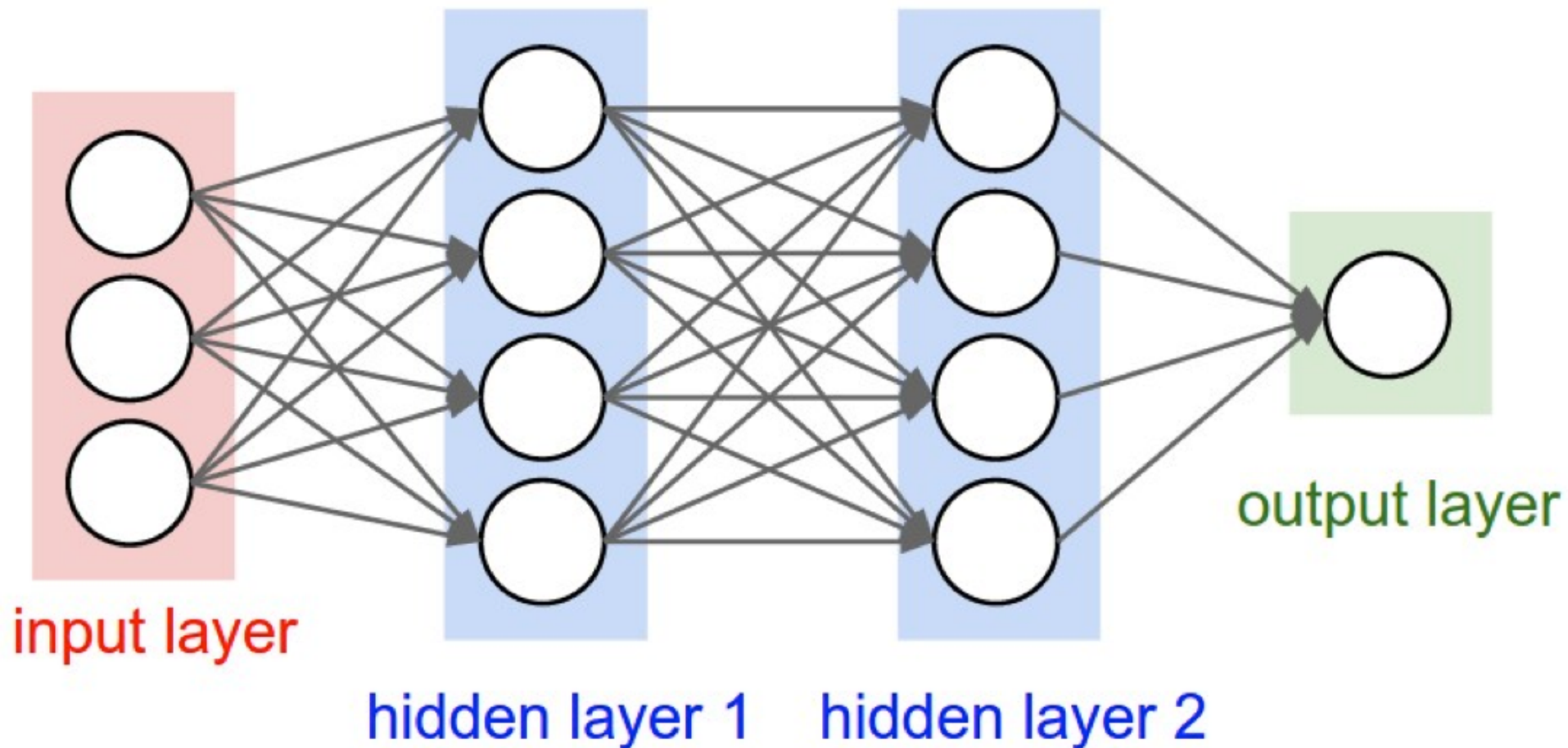


CAT **DOG**

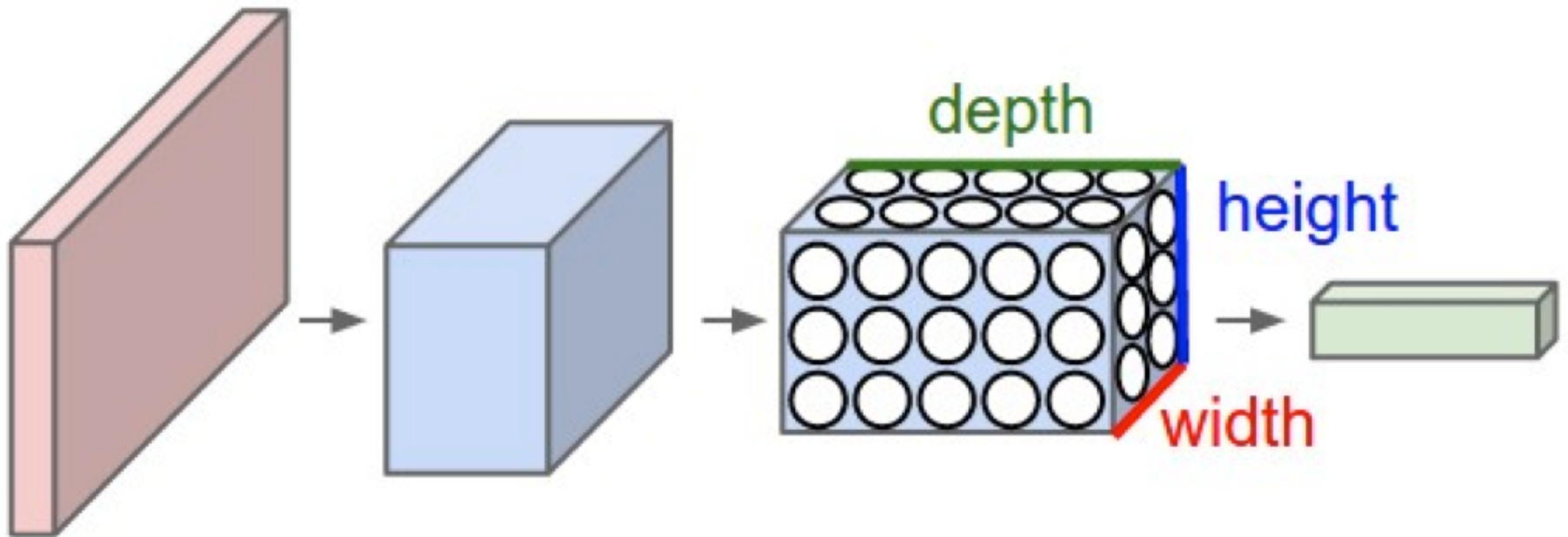


Convolutional Neural Networks (CNNs / ConvNets)

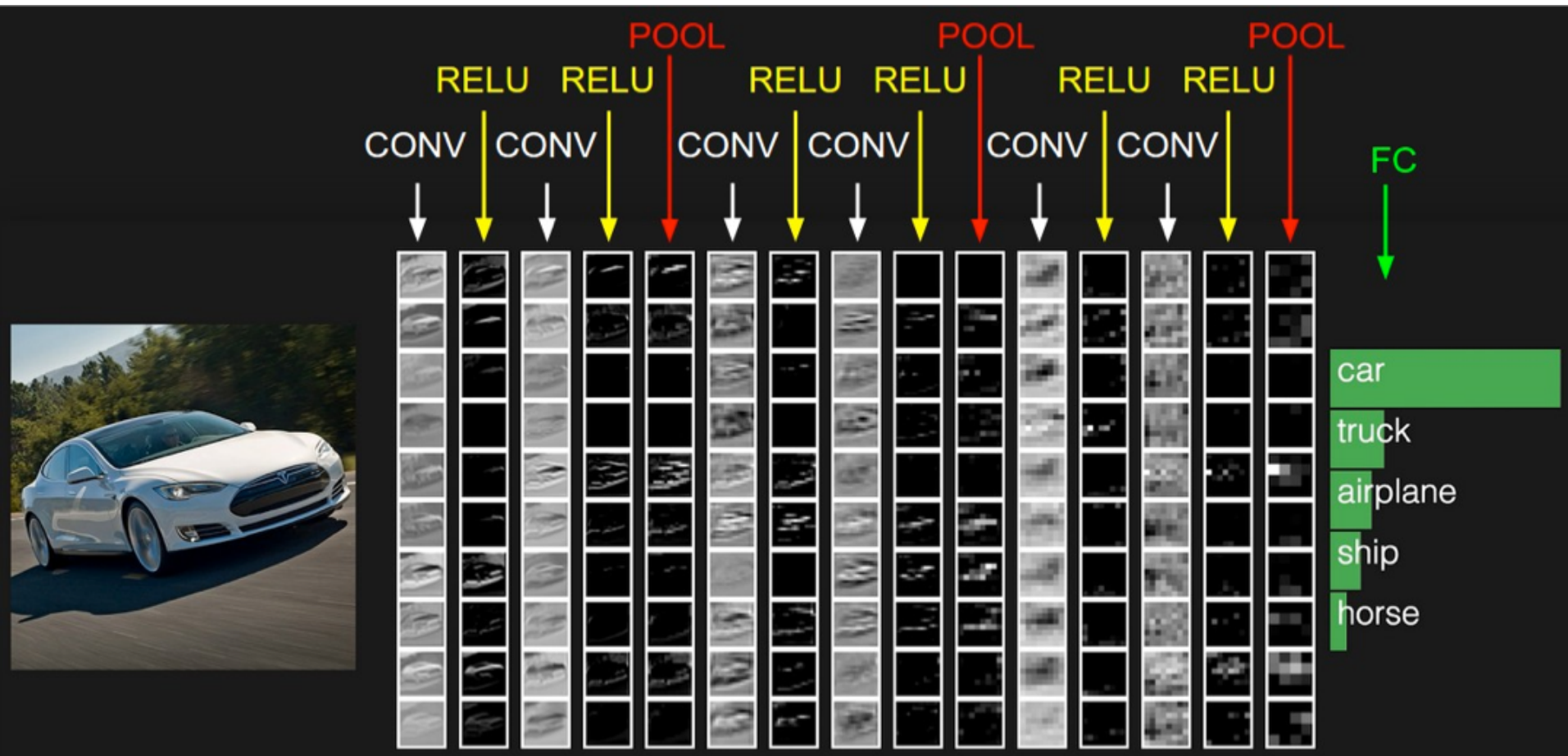
A regular 3-layer Neural Network



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



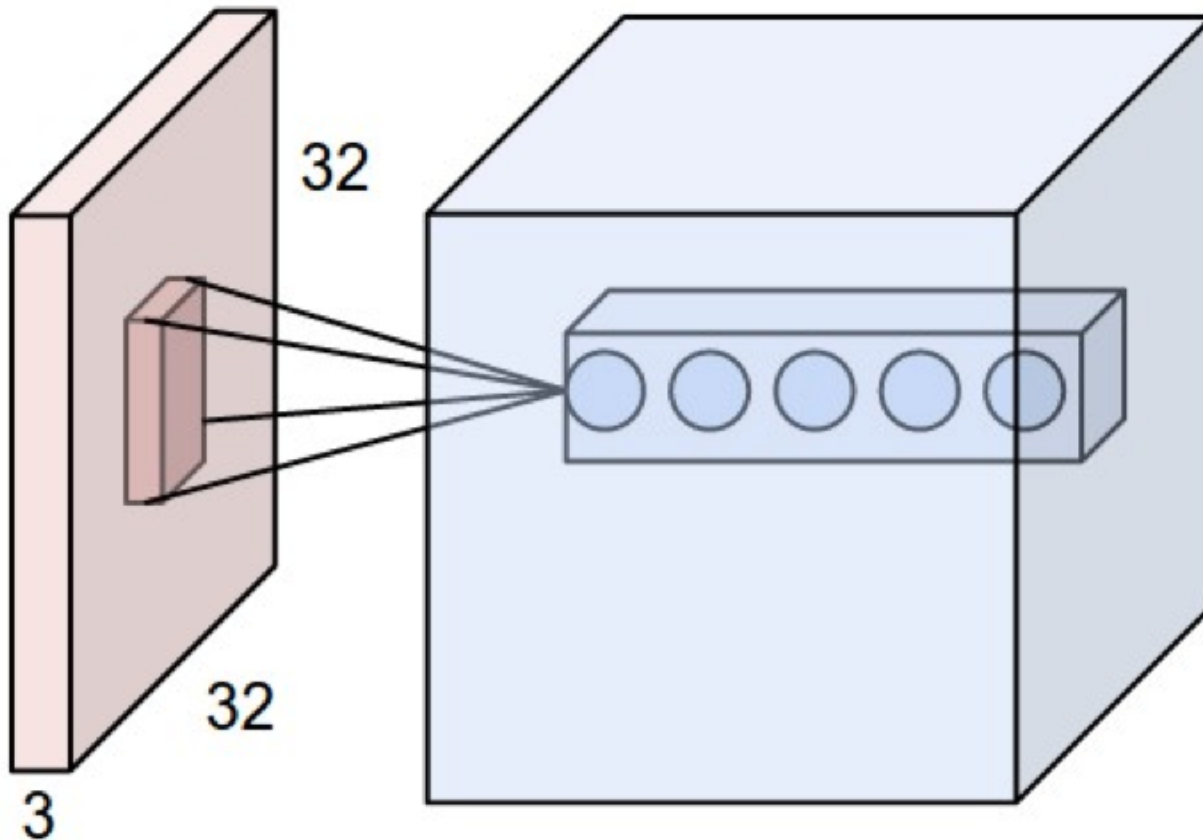
The activations of an example ConvNet architecture.



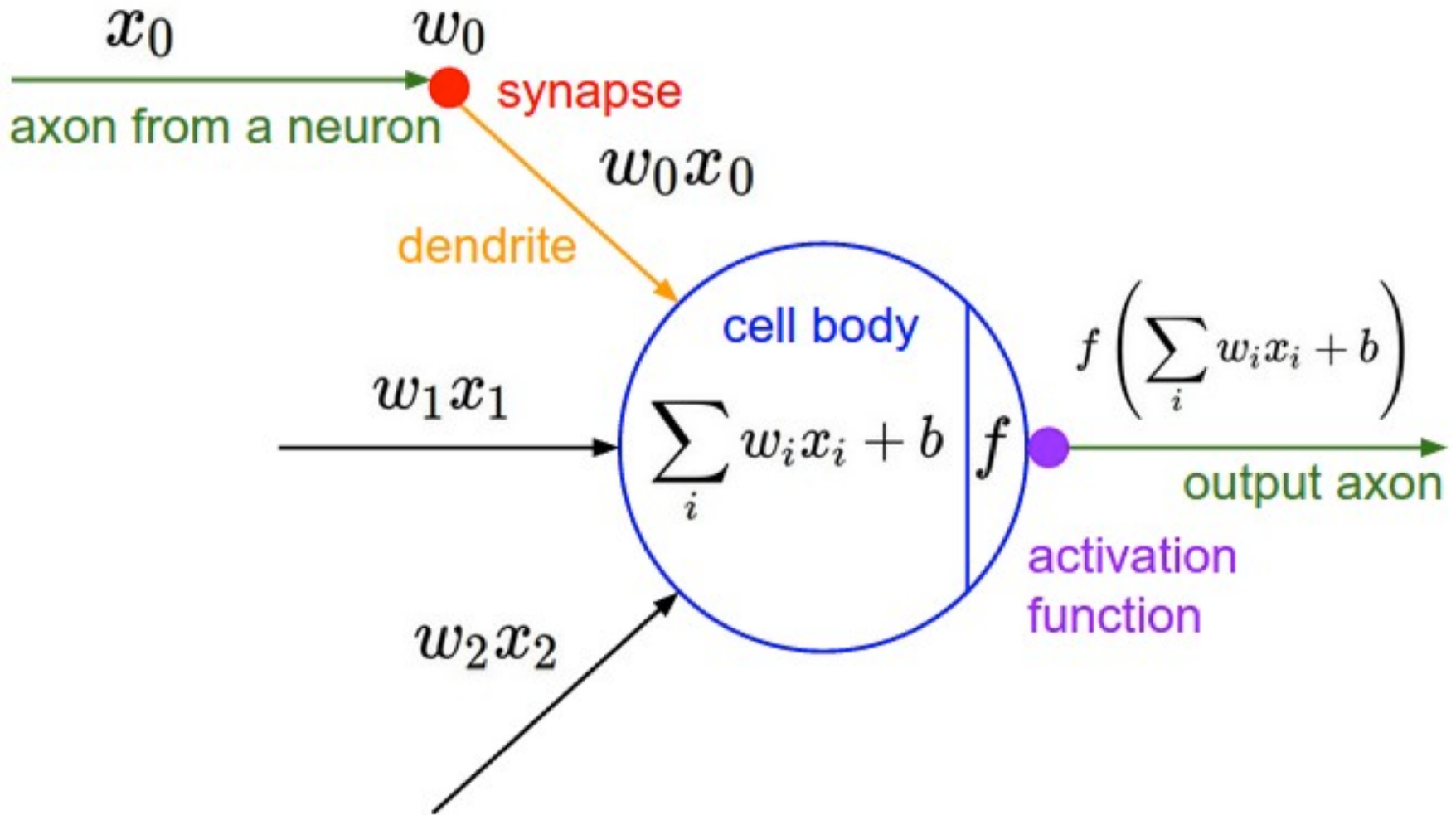
ConvNets

32x32x3 CIFAR-10 image

first Convolutional layer



ConvNets



Convolution Demo

Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$

0	0	0	0	0	0	0
0	1	2	0	2	1	0
0	2	2	2	1	1	0
0	2	2	2	0	1	0
0	2	2	1	2	1	0
0	2	1	2	0	1	0
0	0	0	0	0	0	0

$x[:, :, 1]$

0	0	0	0	0	0	0
0	0	2	2	1	2	0
0	1	2	0	0	2	0
0	0	1	2	1	0	0
0	2	2	2	2	0	0
0	2	2	2	0	2	0
0	0	0	0	0	0	0

$x[:, :, 2]$

0	0	0	0	0	0	0
0	1	0	0	1	0	0
0	0	2	0	0	0	0
0	0	0	1	1	1	0
0	2	2	2	1	2	0
0	1	2	0	0	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$

-1	-1	0
1	1	1
-1	0	1

$w0[:, :, 1]$

0	0	1
0	1	0
0	0	1

$w0[:, :, 2]$

-1	-1	0
1	0	-1
-1	0	-1

Bias $b0$ (1x1x1)

$b0[:, :, 0]$

1

Filter W1 (3x3x3)

$w1[:, :, 0]$

1	-1	0
0	1	1
0	-1	1

$w1[:, :, 1]$

-1	1	0
-1	-1	1
0	0	0

$w1[:, :, 2]$

1	0	-1
0	0	-1
1	0	1

Bias $b1$ (1x1x1)

$b1[:, :, 0]$

0

Output Volume (3x3x2)

$o[:, :, 0]$

6	3	6
7	-1	-2
2	3	-2

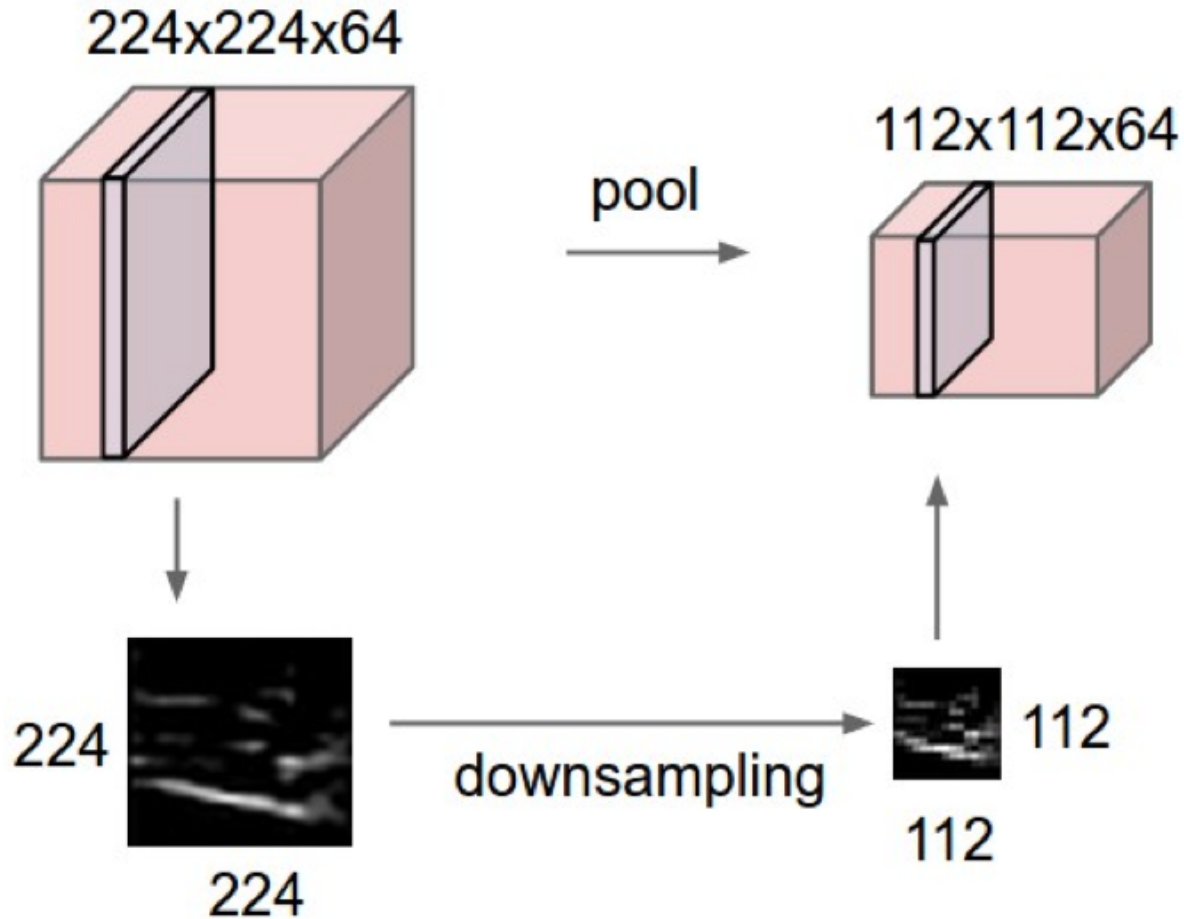
$o[:, :, 1]$

7	-1	-3
4	3	2
-1	0	-1

toggle movement

ConvNets

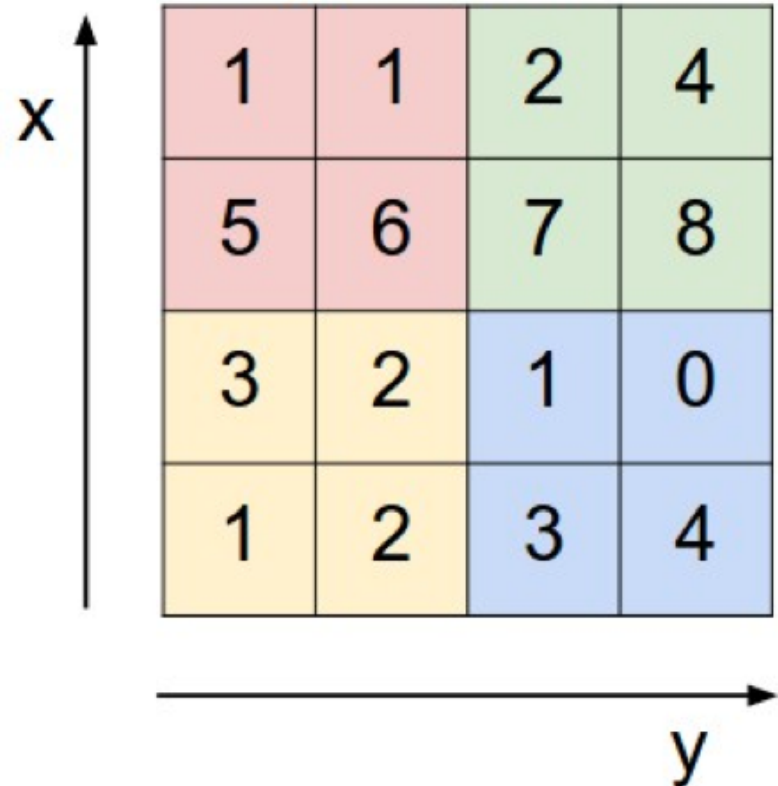
input volume of size [224x224x64]
is pooled with **filter** size 2, **stride** 2
into output volume of size [112x112x64]



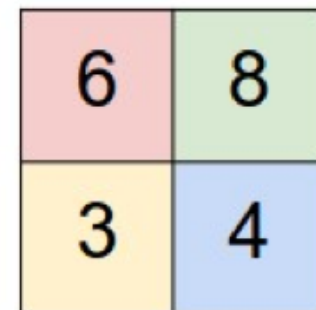
ConvNets

max pooling

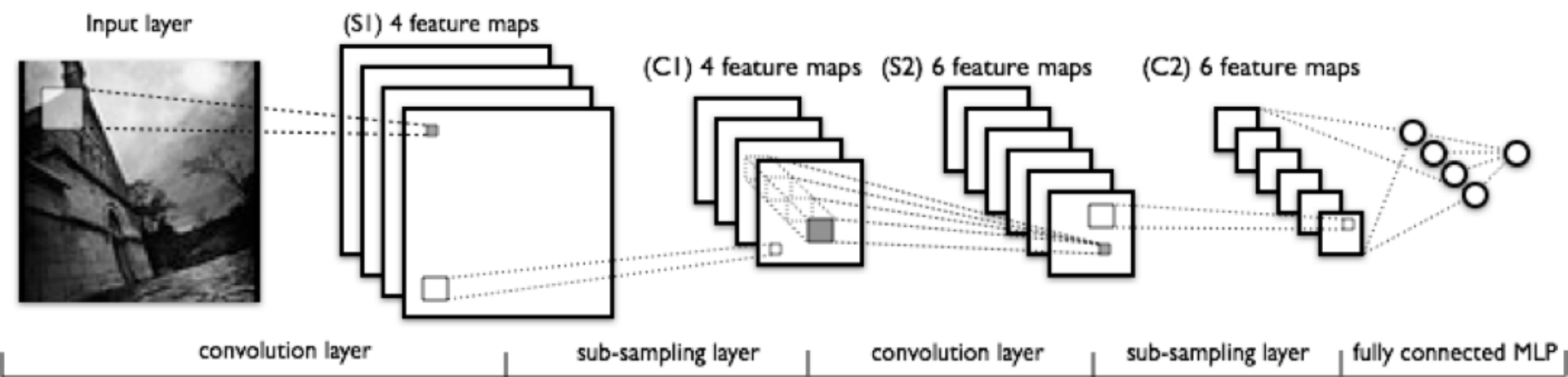
Single depth slice



max pool with 2x2 filters
and stride 2

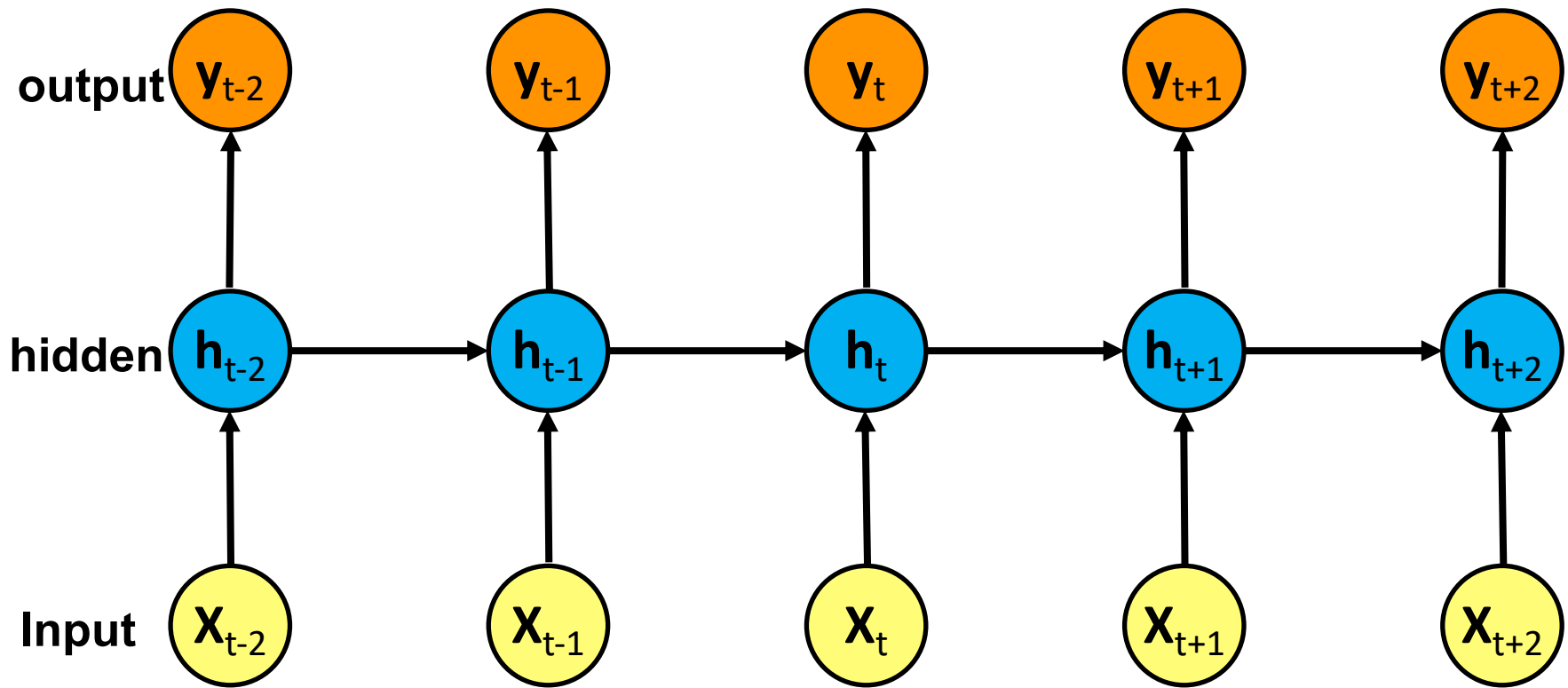


Convolutional Neural Networks (CNN) (LeNet)



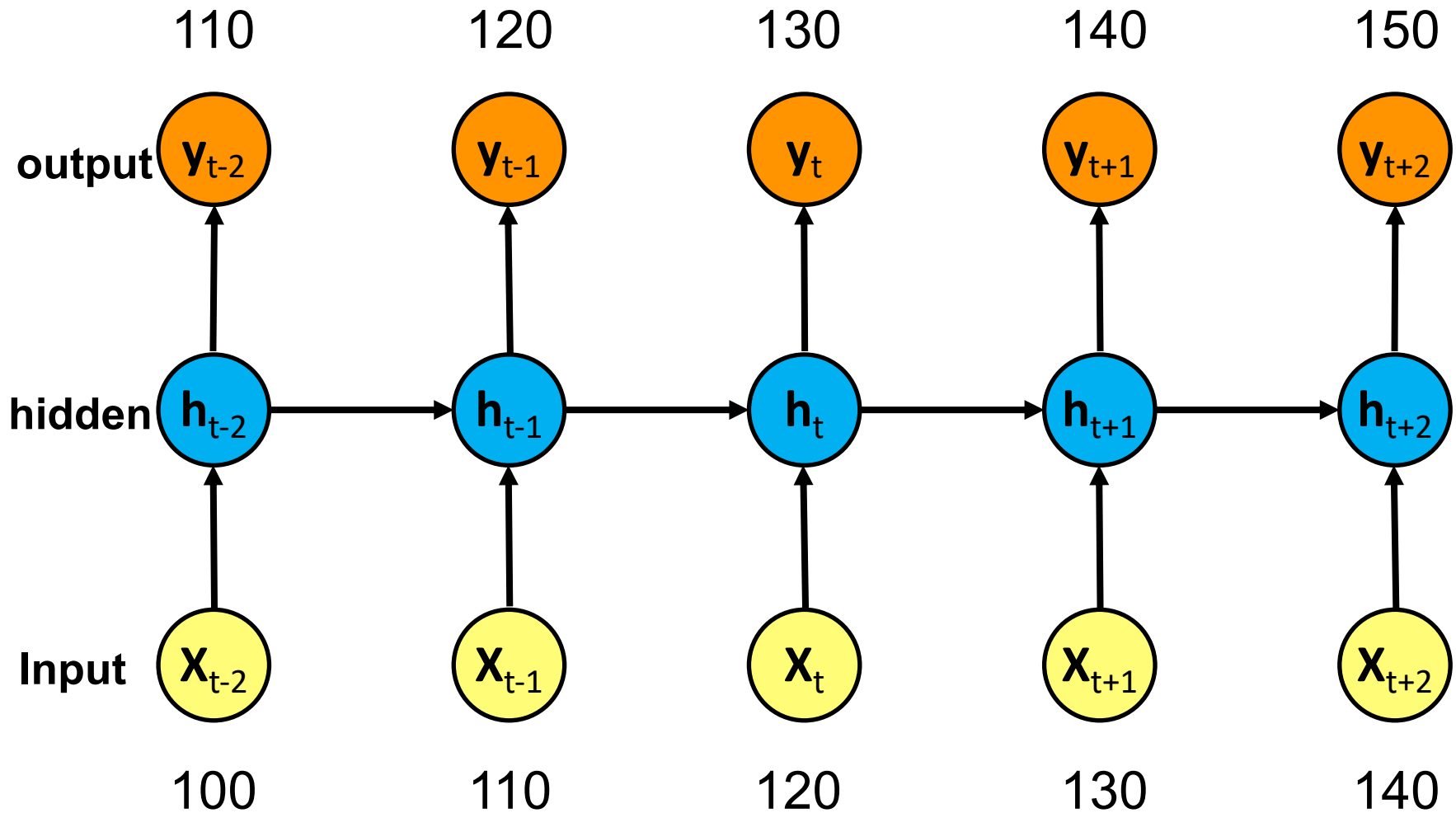
Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN)

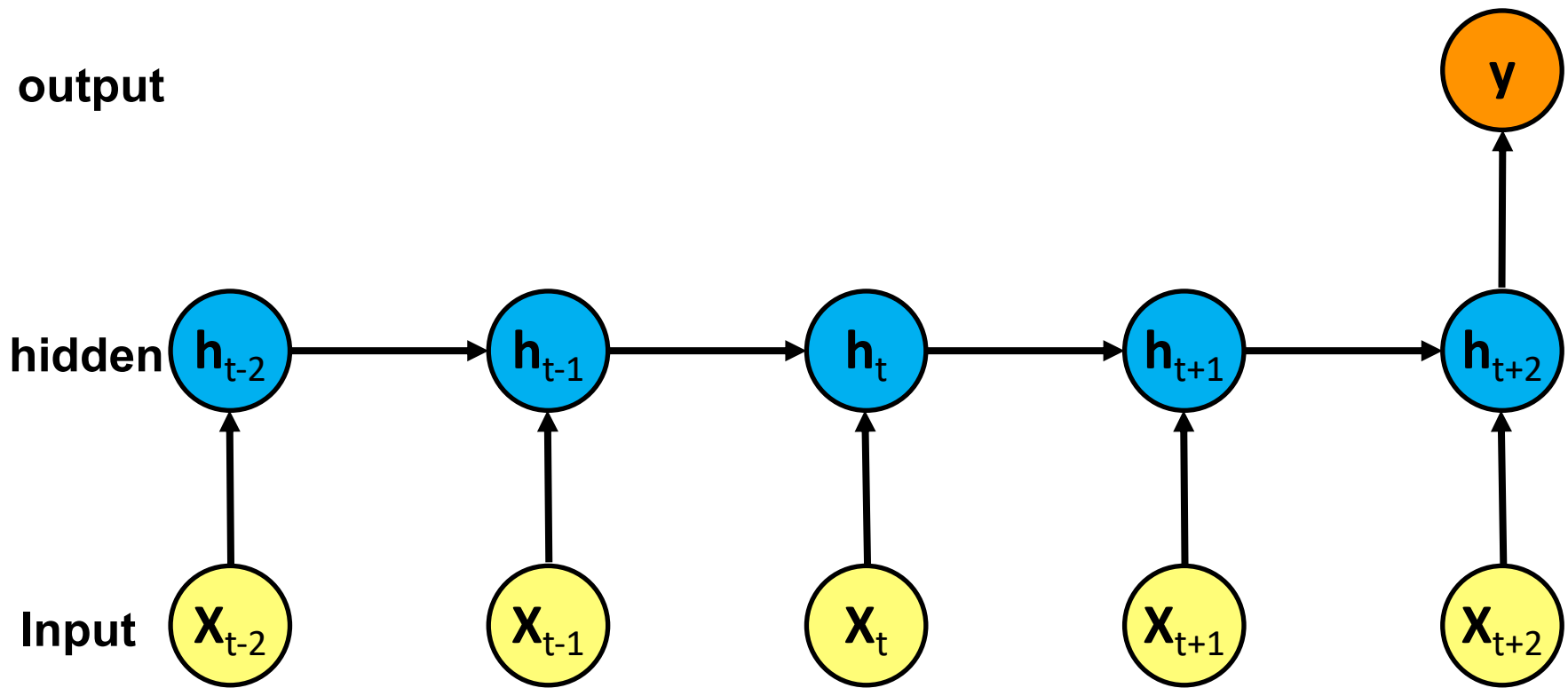


Recurrent Neural Networks (RNN)

Time Series Forecasting

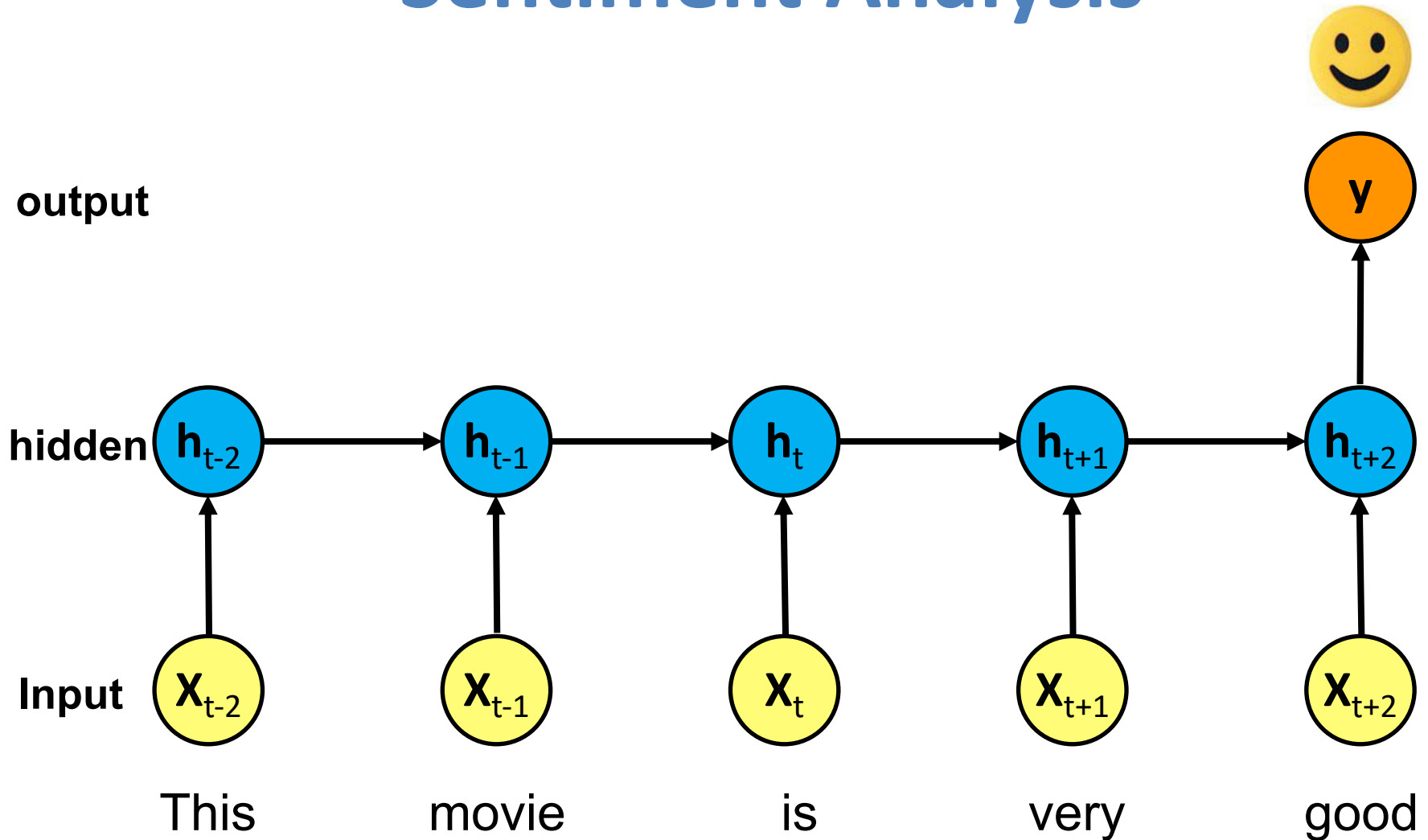


Recurrent Neural Networks (RNN)



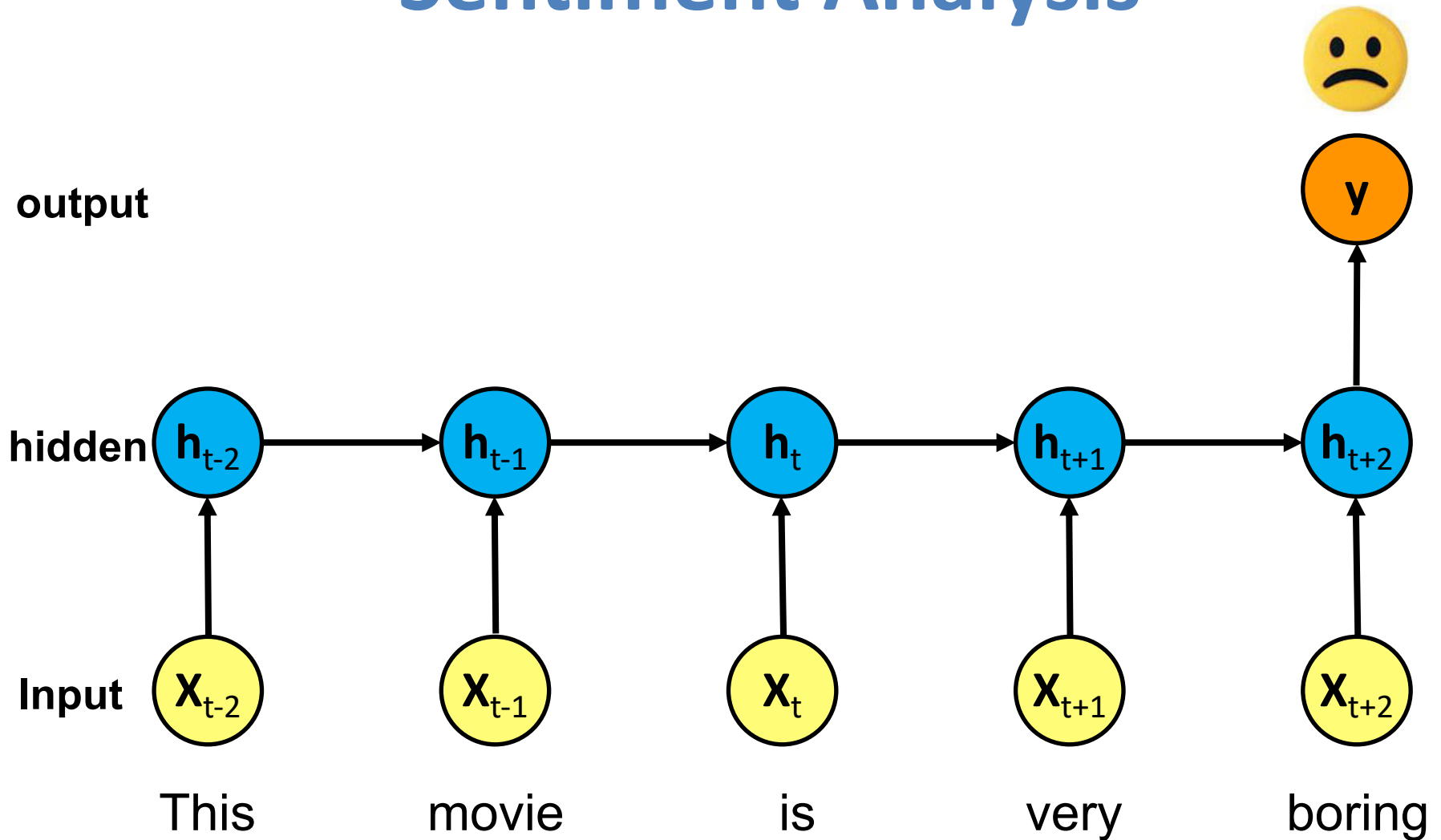
Recurrent Neural Networks (RNN)

Sentiment Analysis

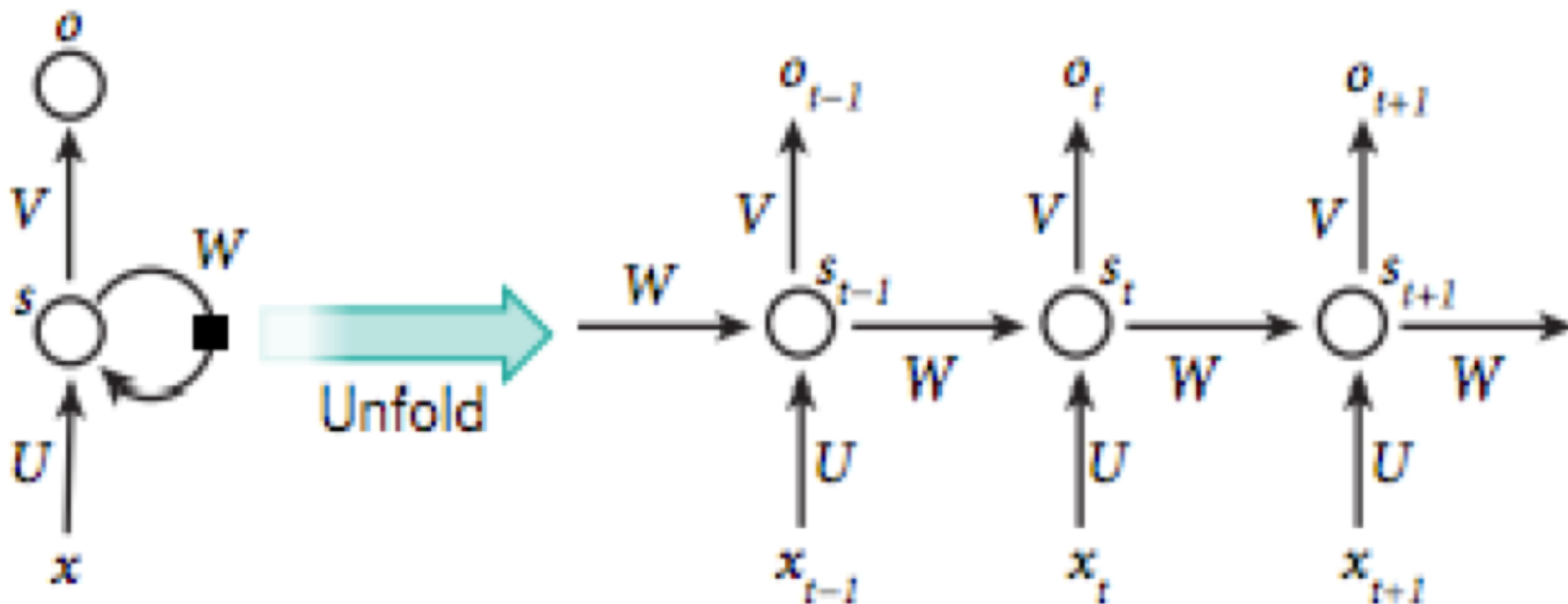


Recurrent Neural Networks (RNN)

Sentiment Analysis

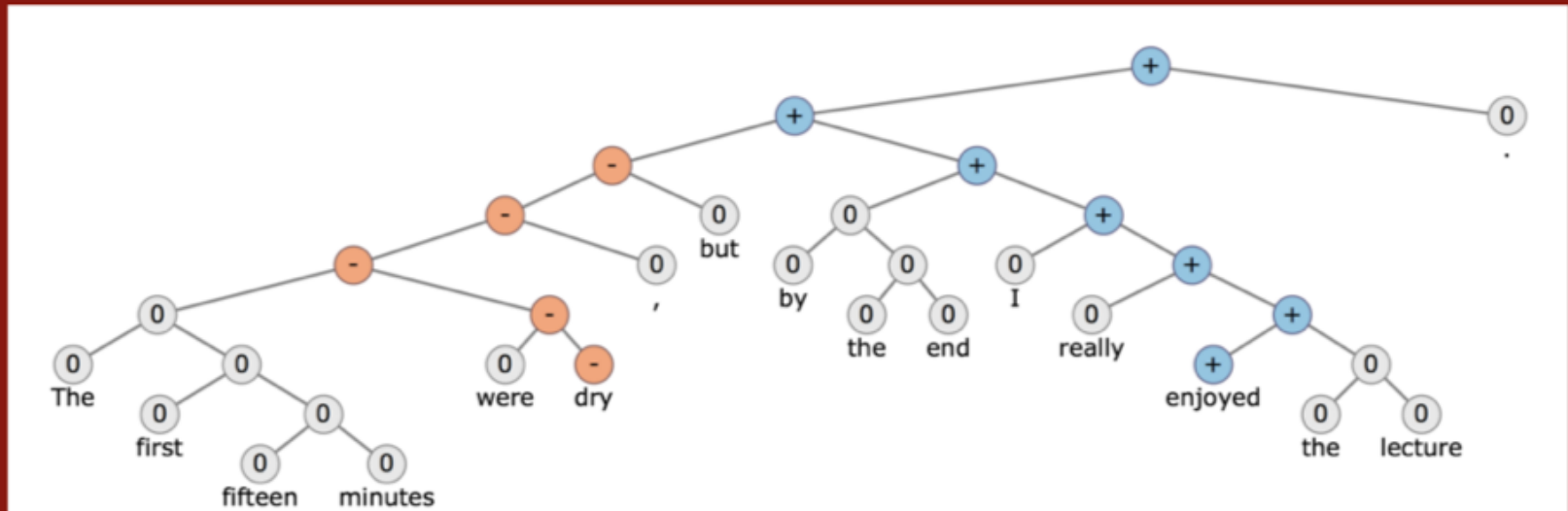


Recurrent Neural Network (RNN)



CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

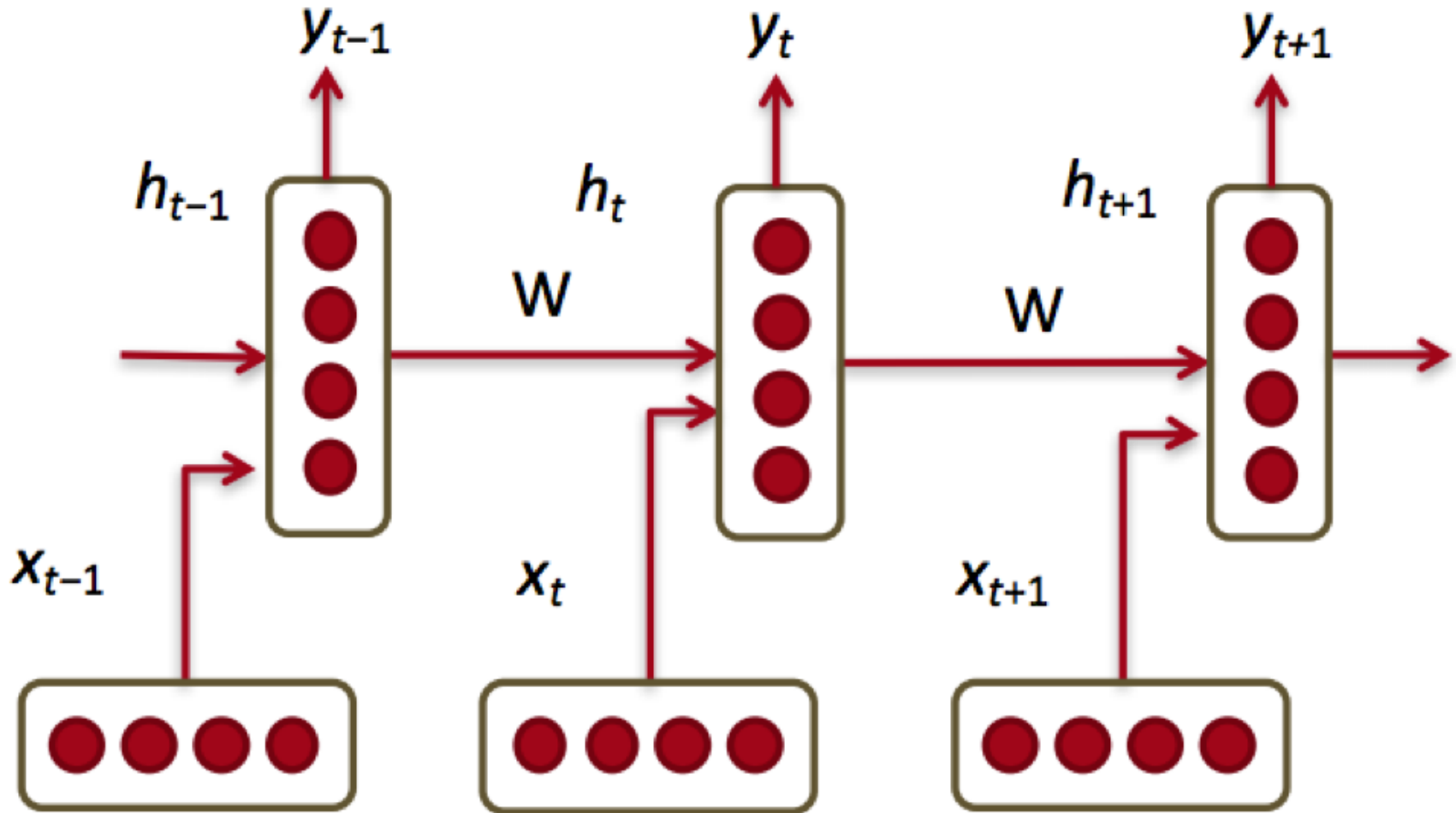


Course Description

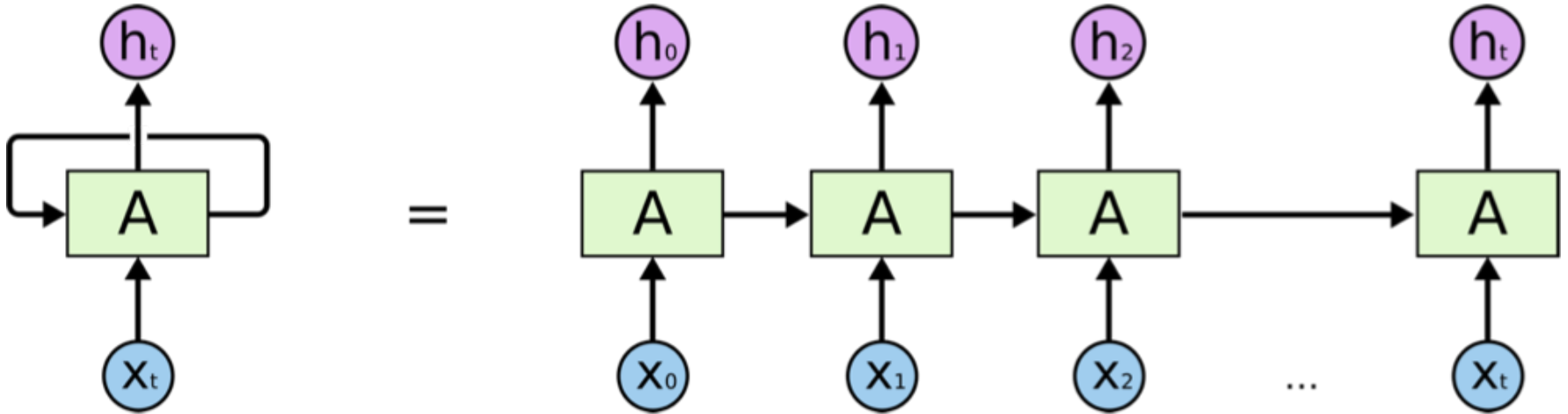
Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, 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/>

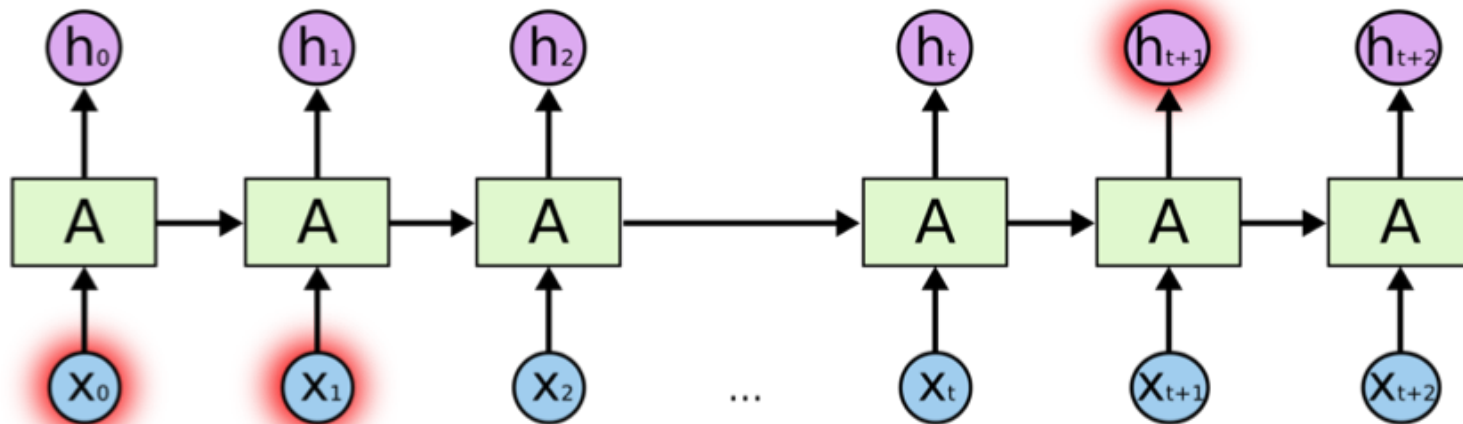
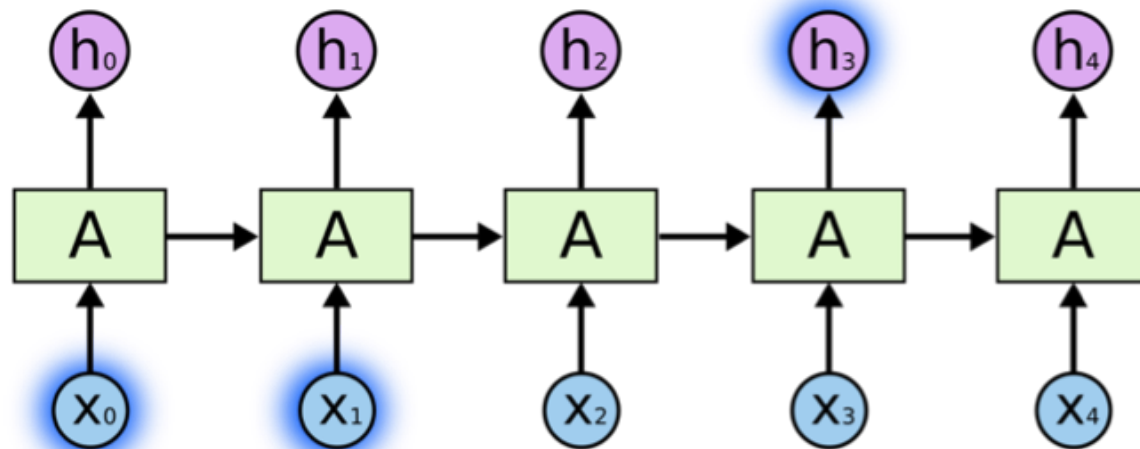
Recurrent Neural Networks (RNNs)



RNN



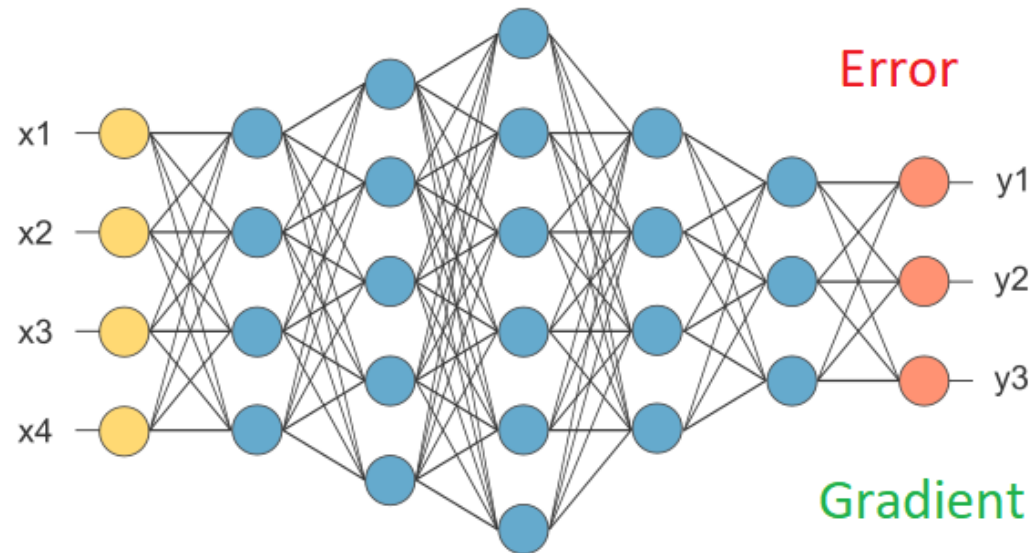
RNN long-term dependencies



I grew up in France... I speak fluent French.

Vanishing Gradient

Exploding Gradient

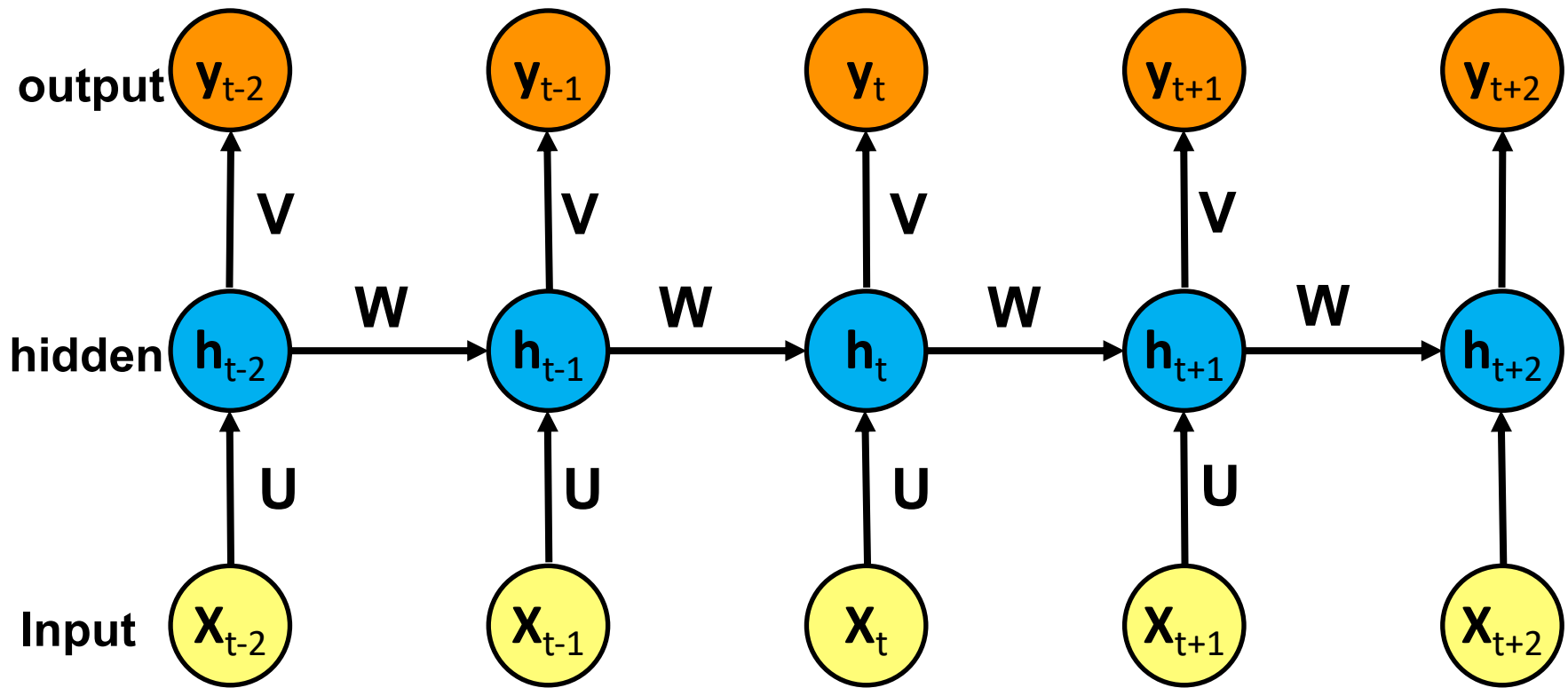


Vanishing Gradient



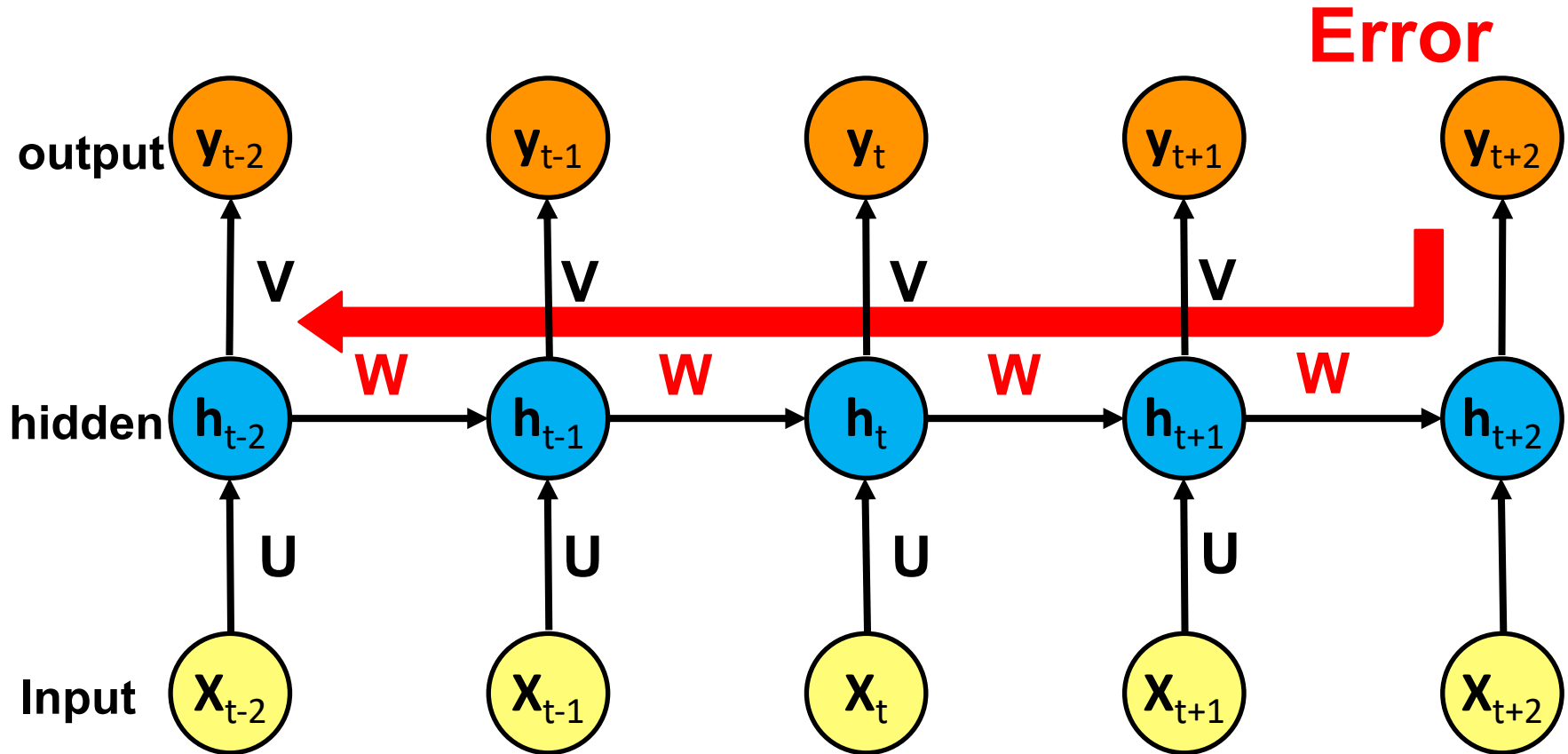
Exploding Gradient

Recurrent Neural Networks (RNN)



RNN

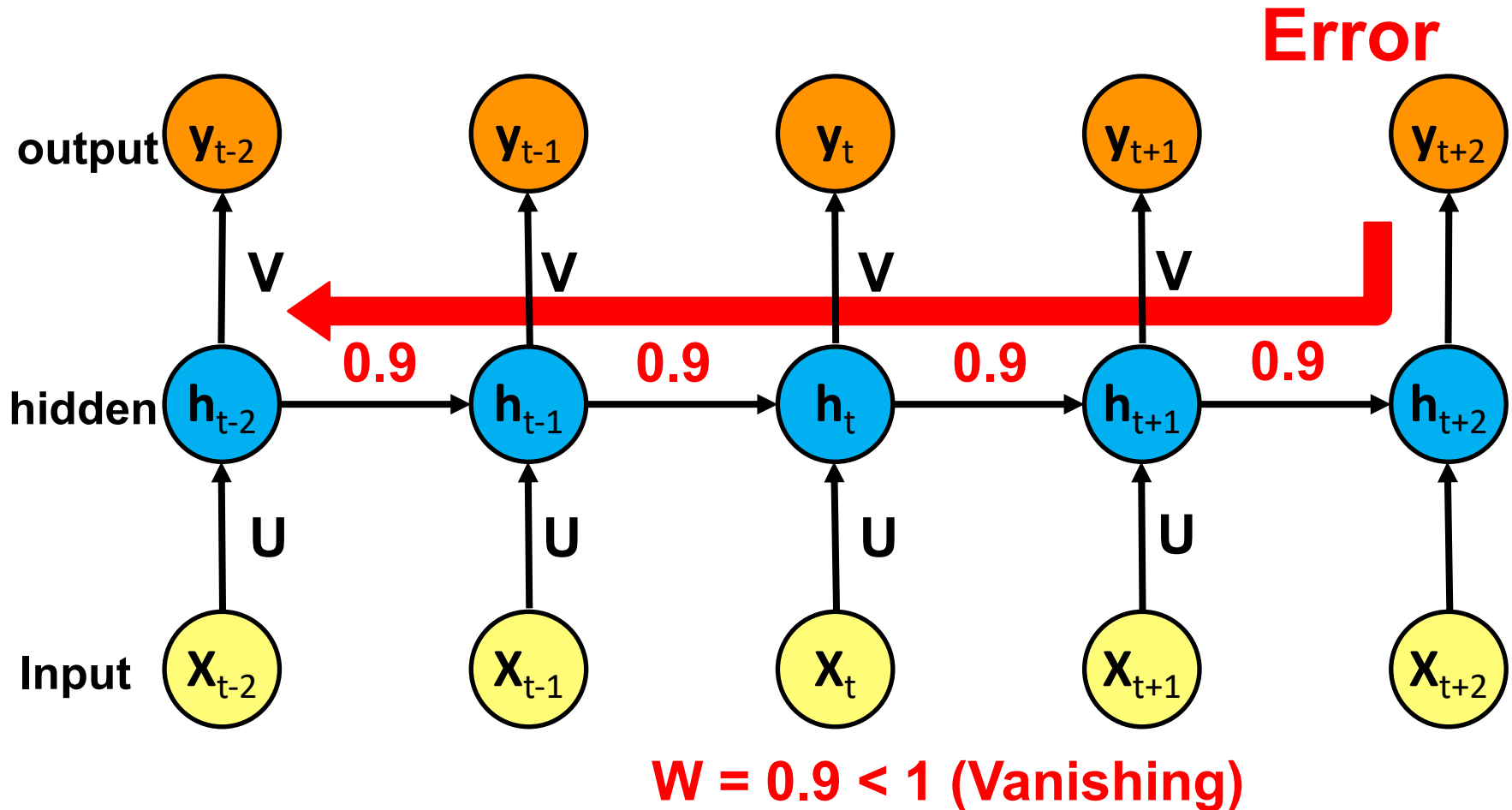
Vanishing Gradient problem Exploding Gradient problem



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

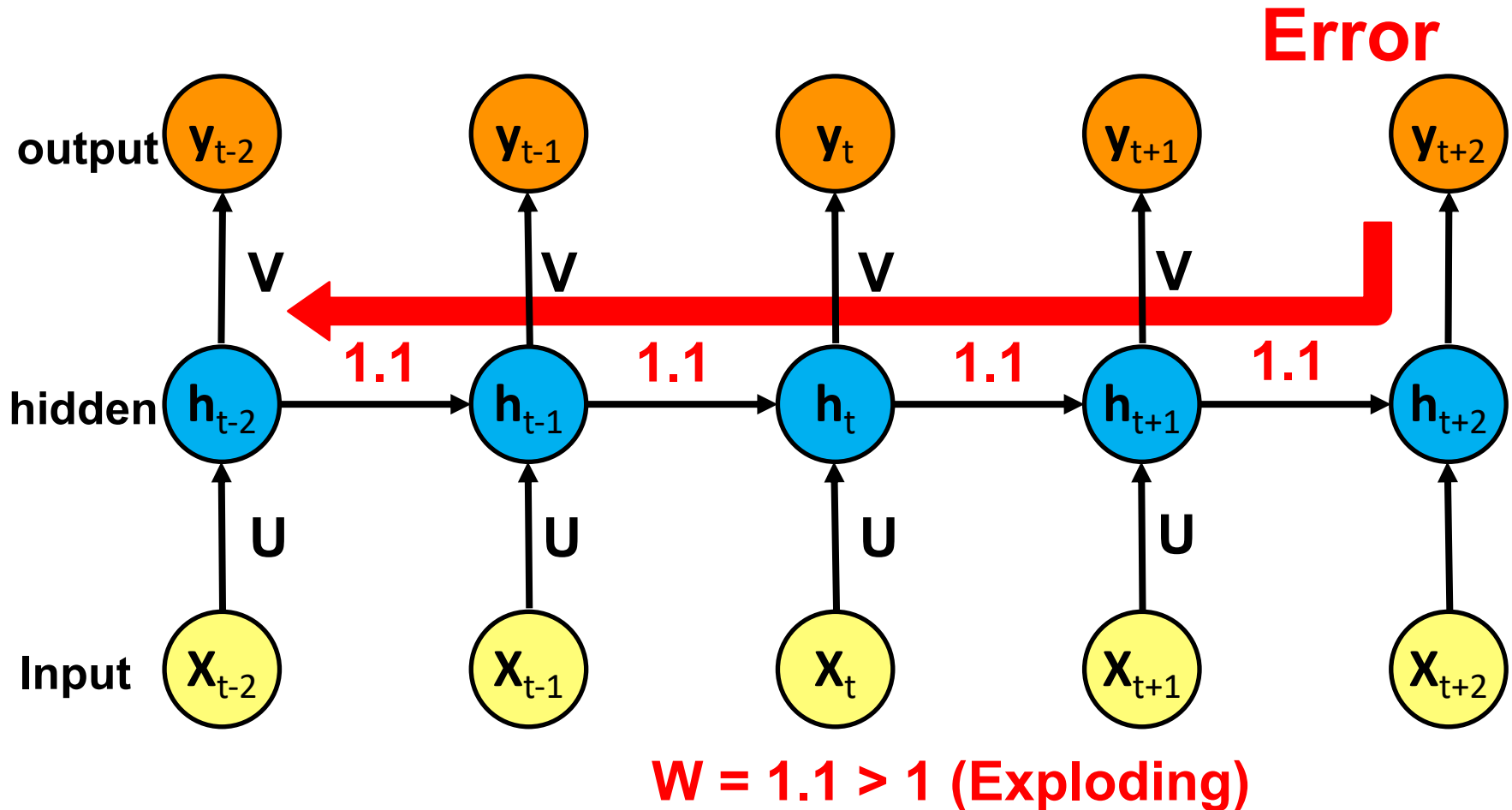
RNN

Vanishing Gradient problem



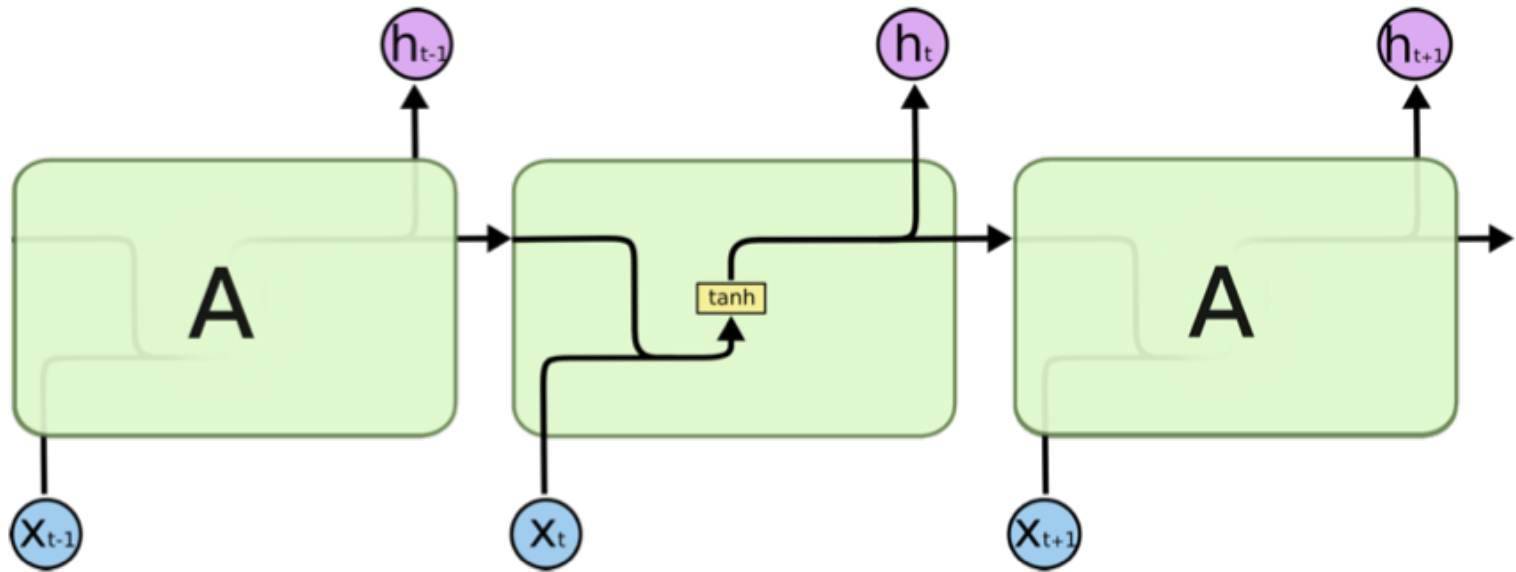
RNN

Exploding Gradient problem

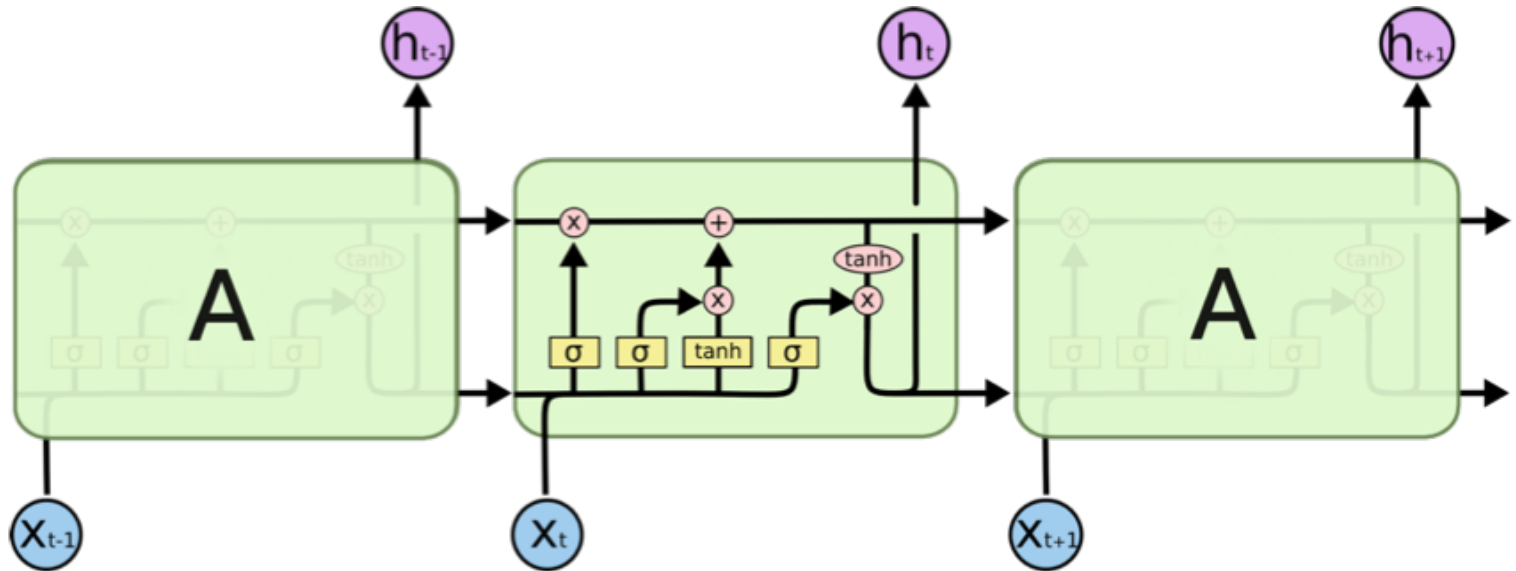


RNN LSTM

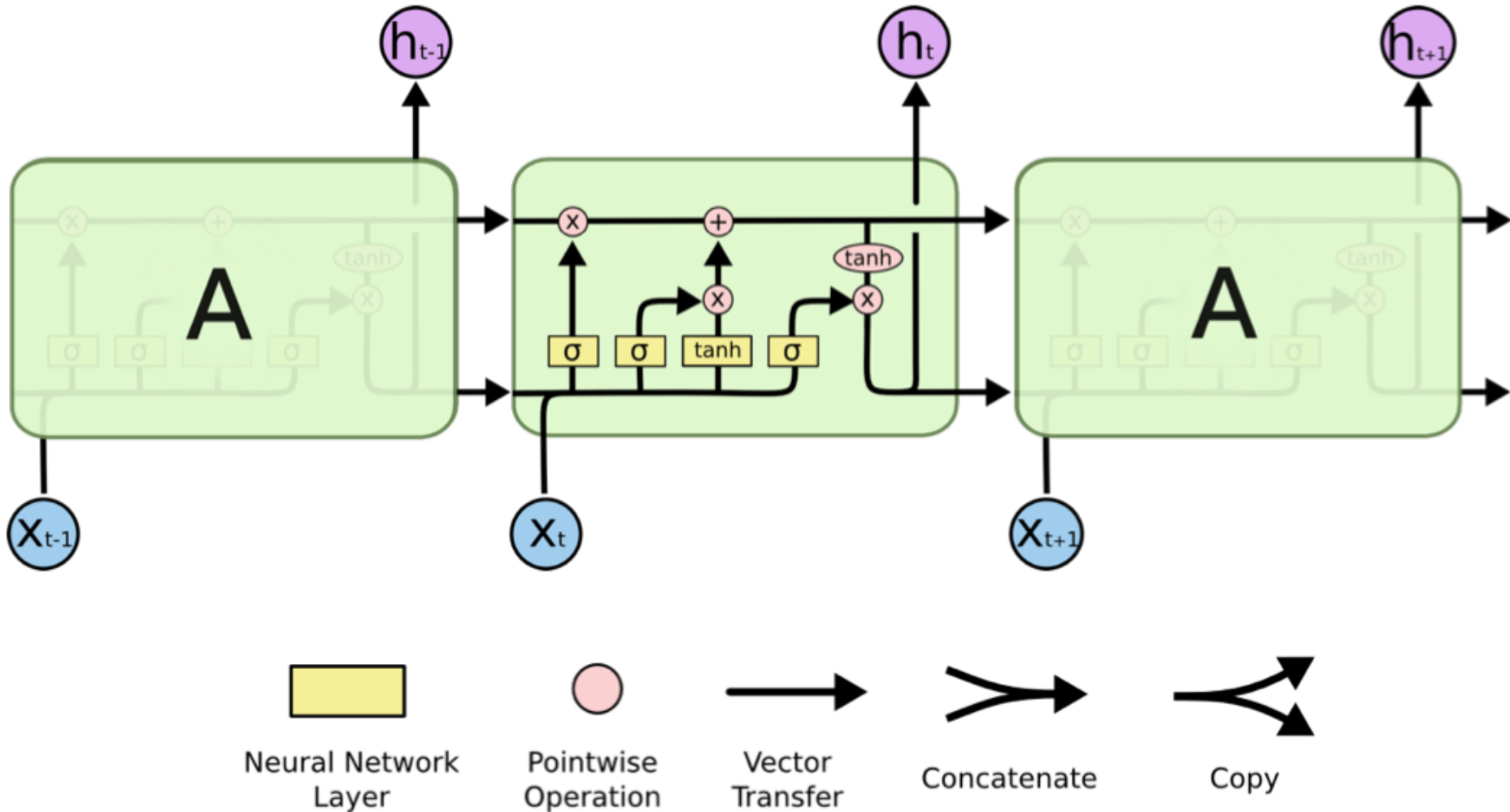
RNN



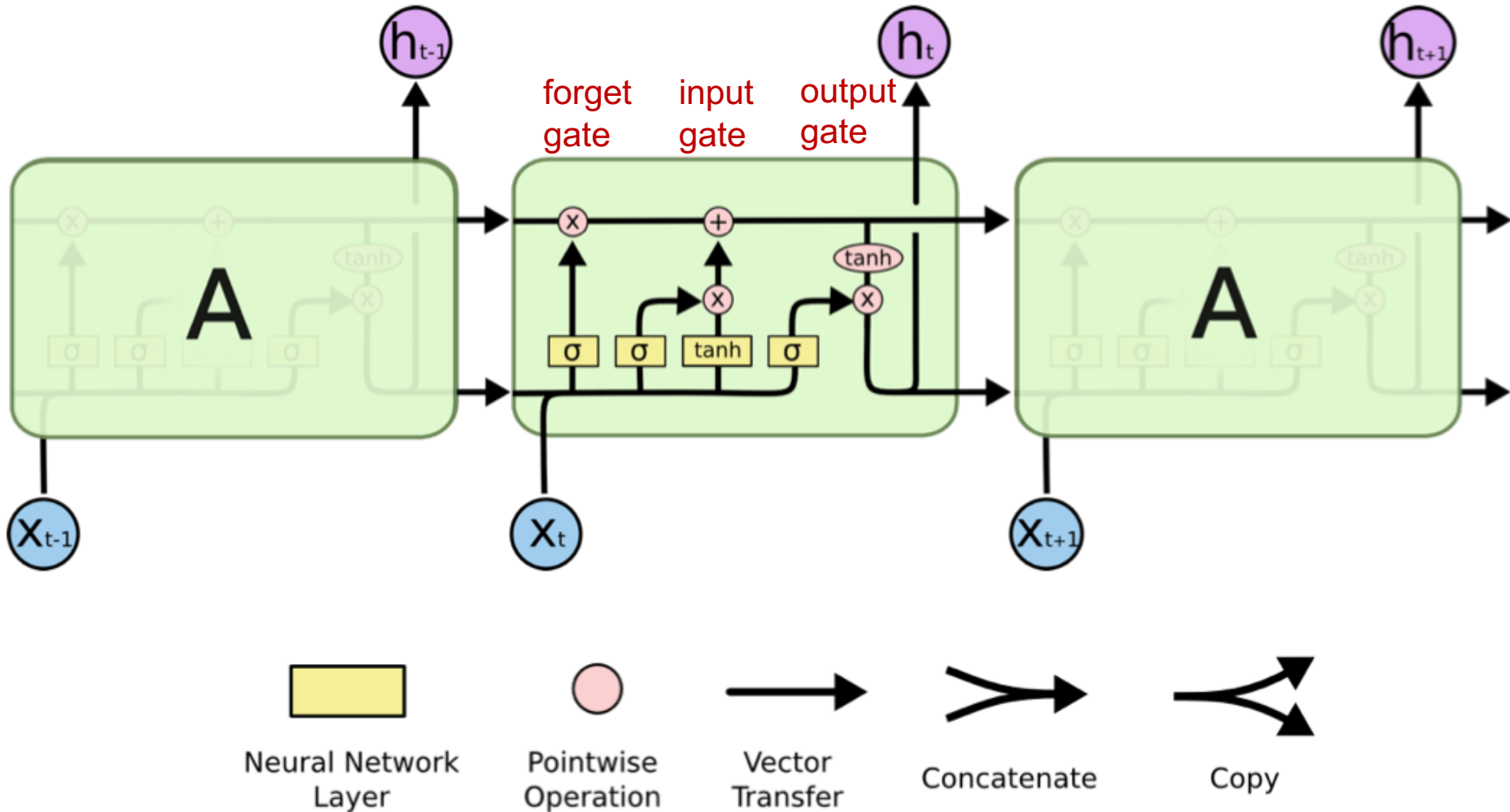
LSTM



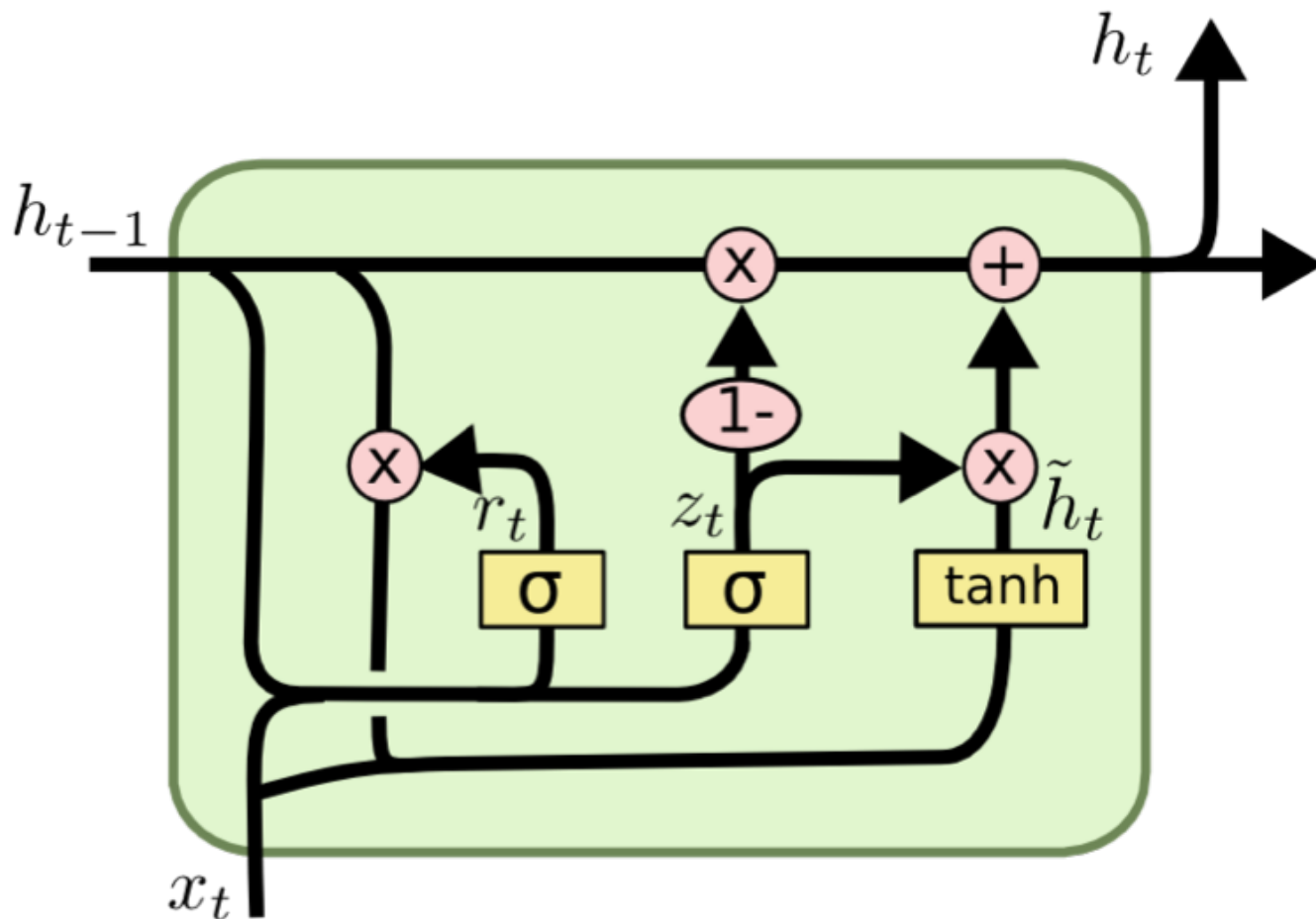
Long Short Term Memory (LSTM)



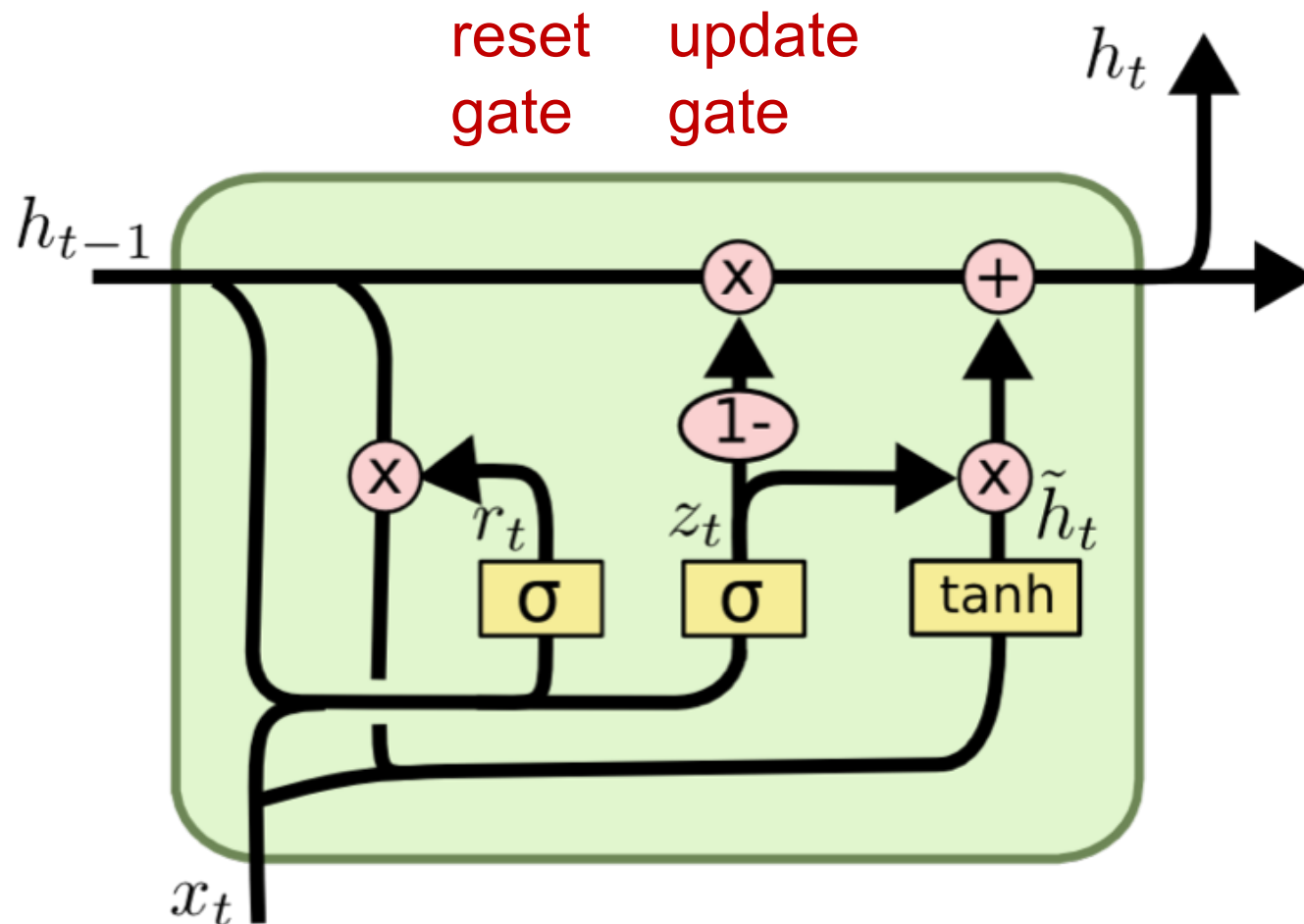
Long Short Term Memory (LSTM)



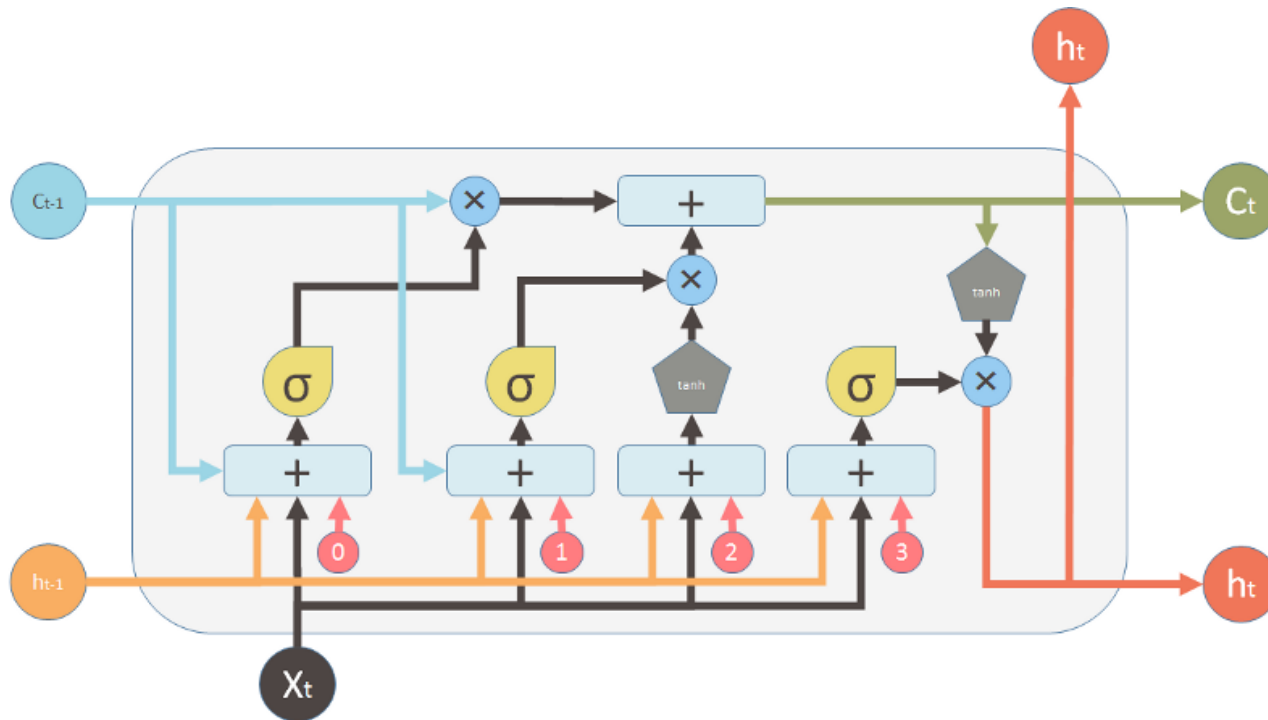
Gated Recurrent Unit (GRU)






Gated Recurrent Unit (GRU)





LSTM





Inputs:


-  Input vector
-  Memory from previous block
-  Output of previous block

outputs:



-  Memory from current block
-  Output of current block

Nonlinearities:

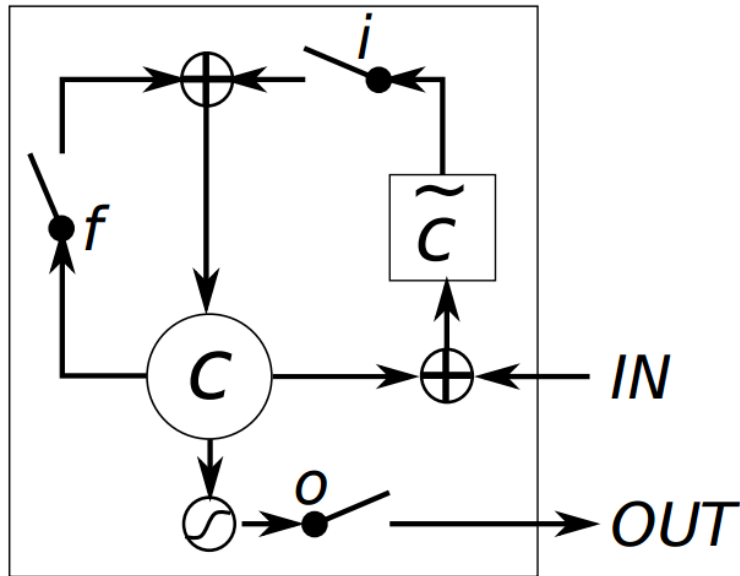
-  Sigmoid
-  Hyperbolic tangent

Bias: 

Vector operations:

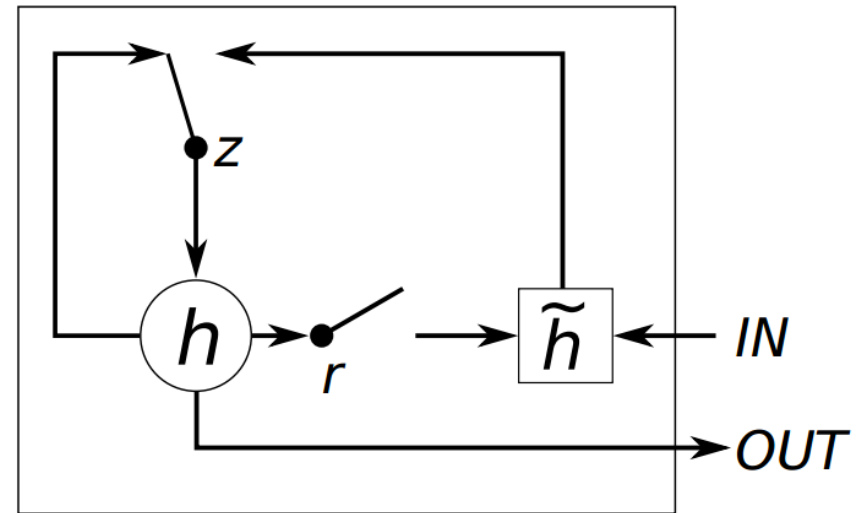
-  Element-wise multiplication
-  Element-wise Summation / Concatenation

LSTM vs GRU



LSTM

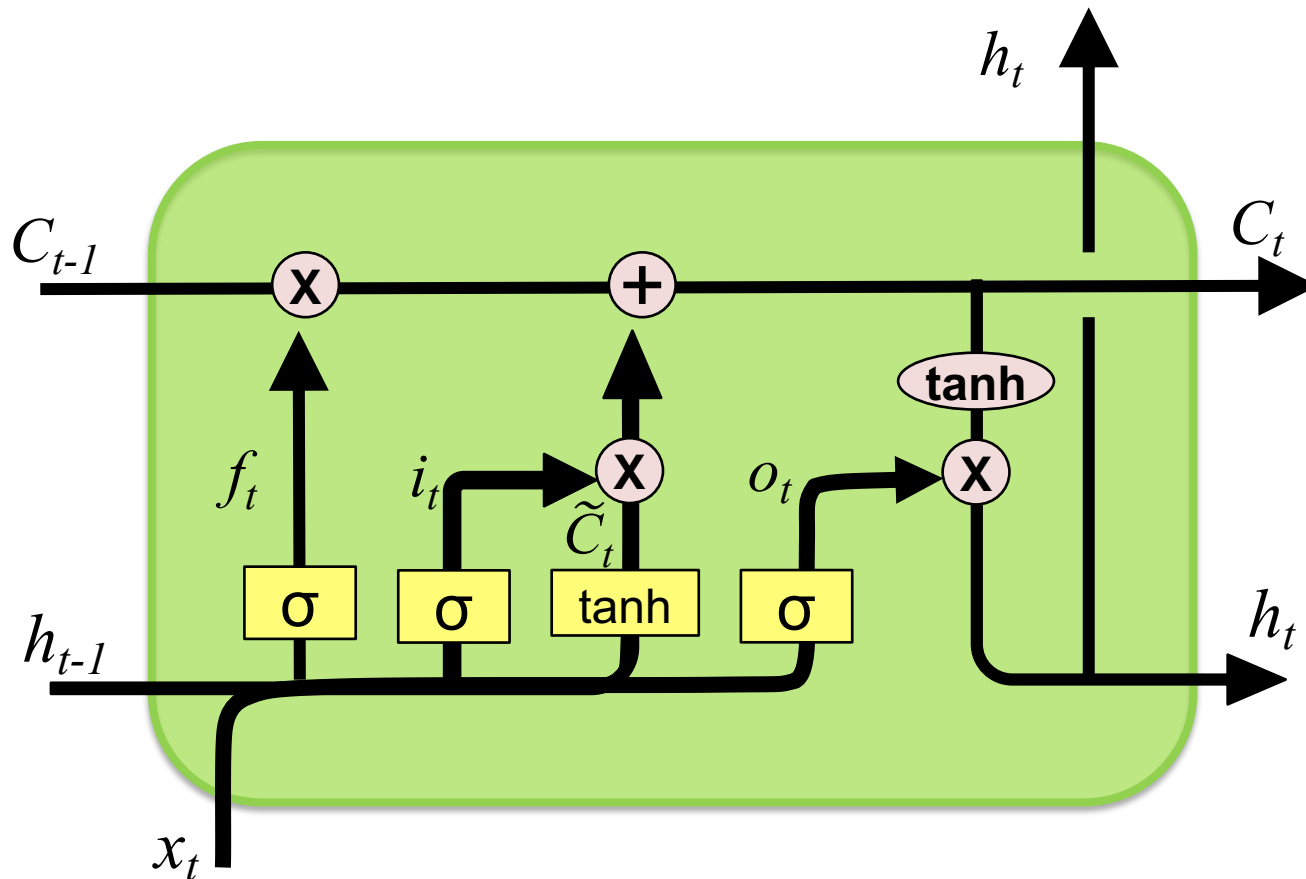
i , f and o are the **input**, **forget** and **output** gates, respectively.
 c and \tilde{c} denote the memory cell and the new memory cell content.



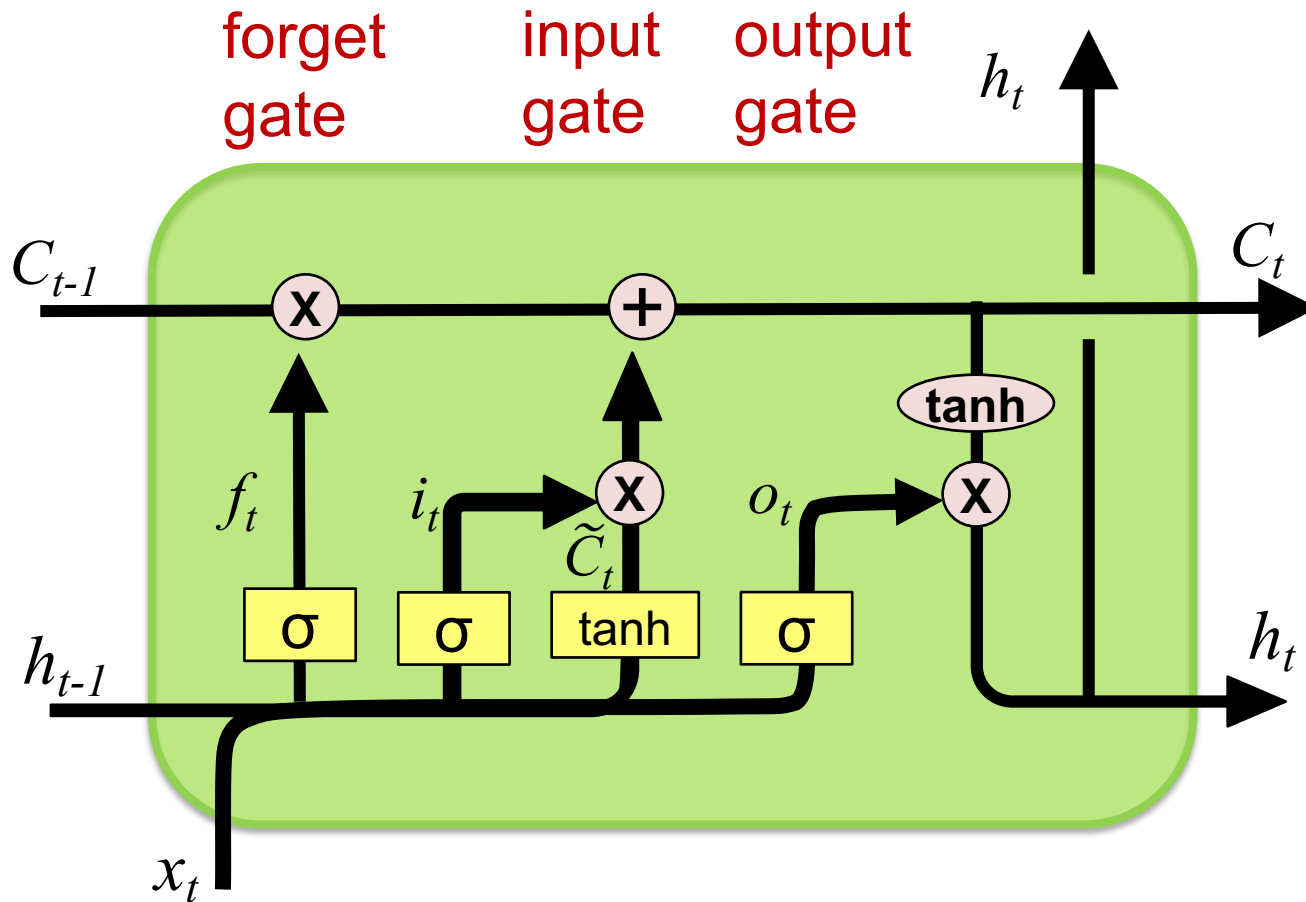
GRU

r and z are the **reset** and **update** gates, and h and \tilde{h} are the activation and the candidate activation.

Long Short Term Memory (LSTM)

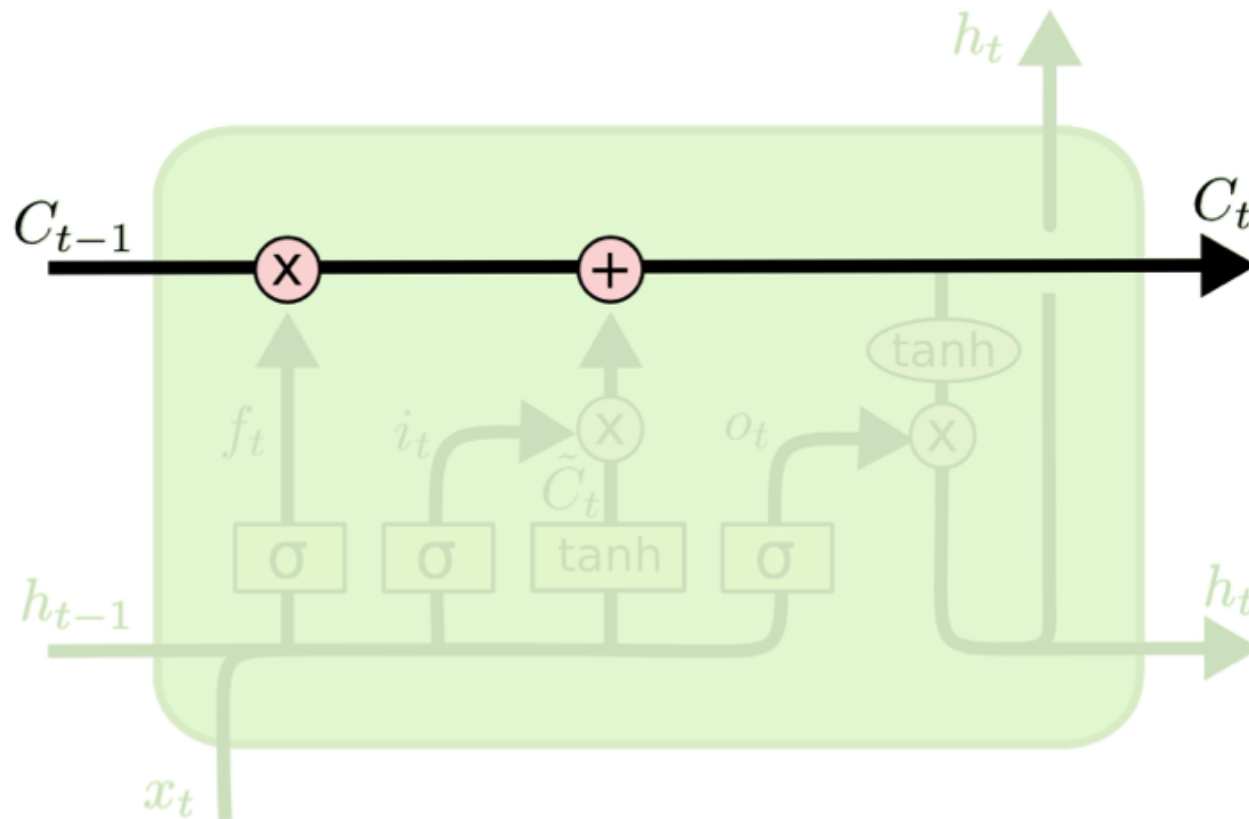


Long Short Term Memory (LSTM)



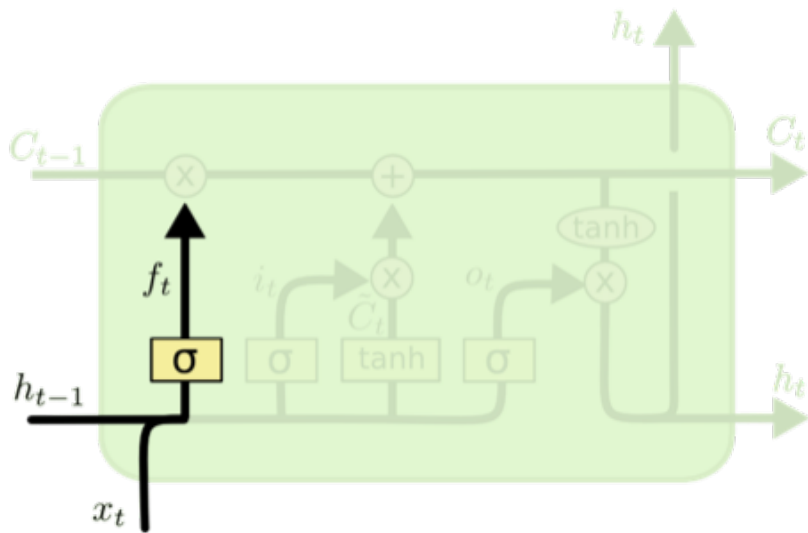
LSTM

Memory state (C)



LSTM

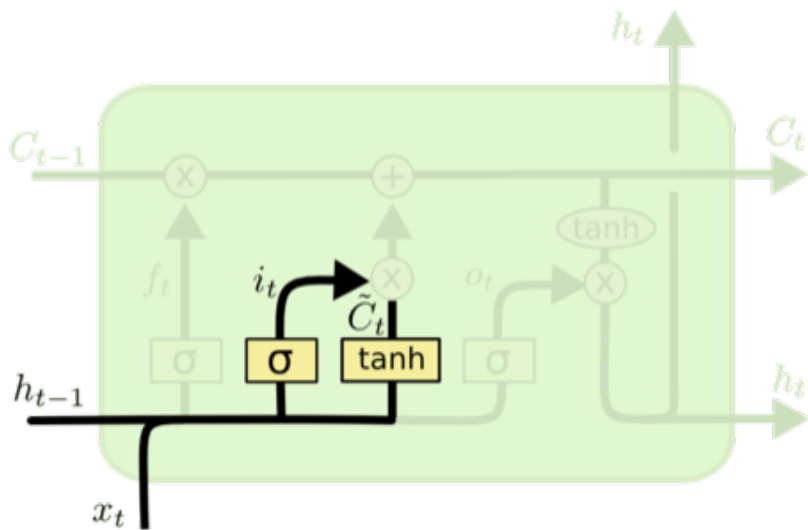
forget gate (f)



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

LSTM

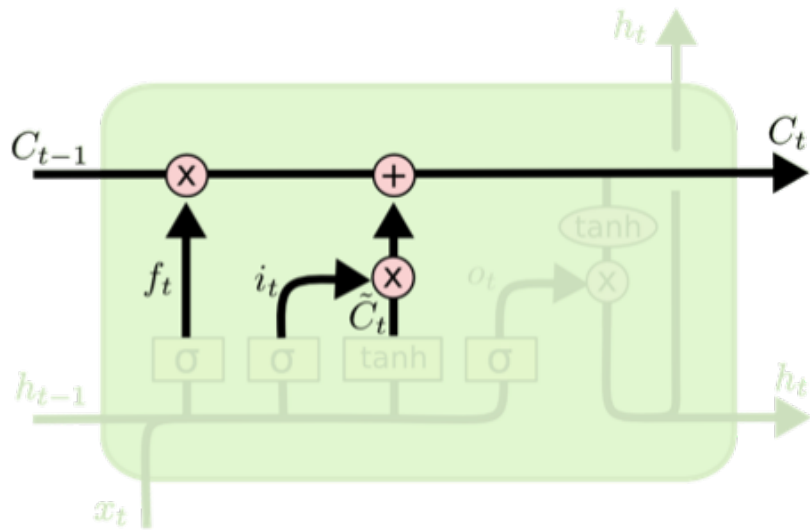
input gate (i)



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

LSTM

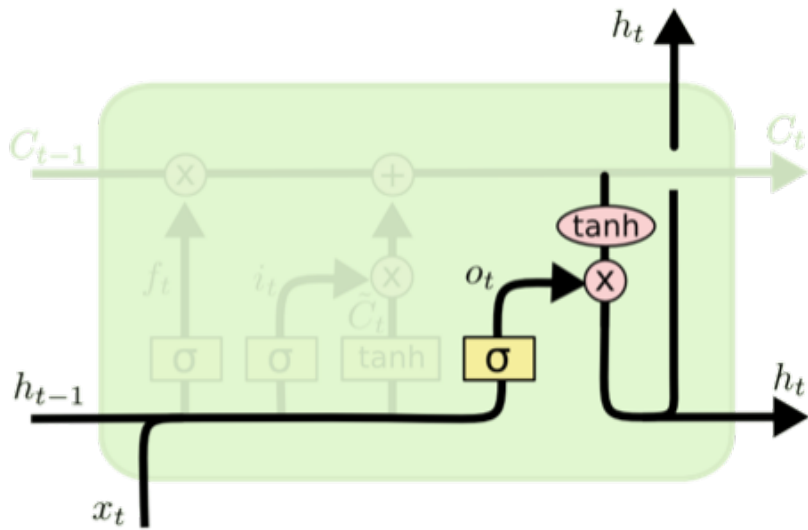
Memory state (C)



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

LSTM

output gate (o)

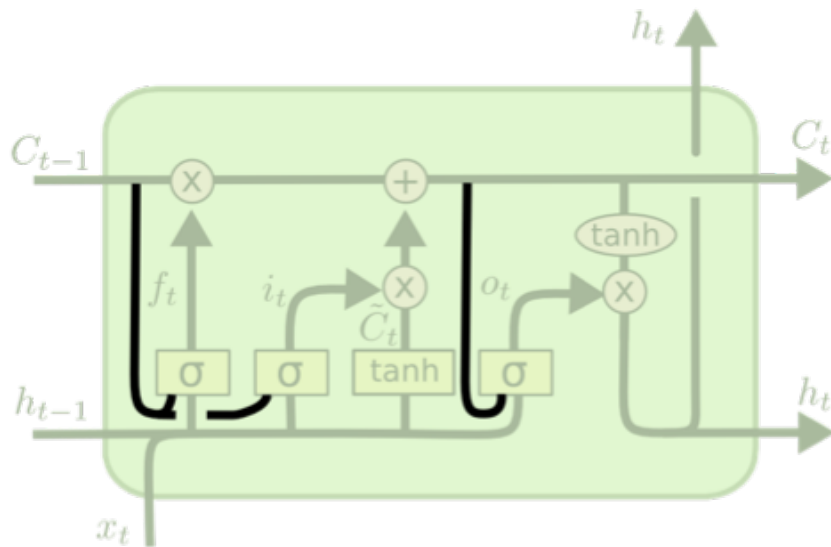


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

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

LSTM

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



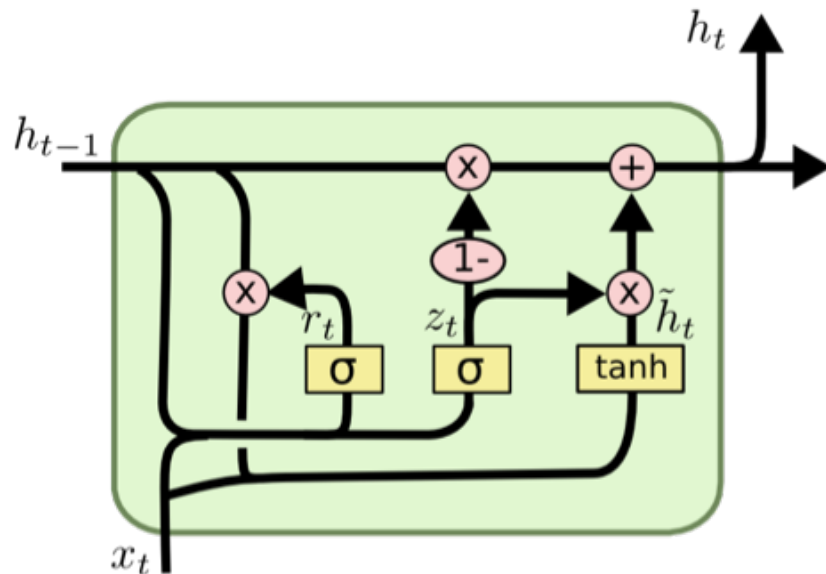
$$f_t = \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

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

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

Gated Recurrent Unit (GRU)

update (z), reset (r) gates



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

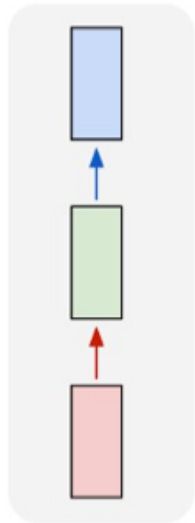
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

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

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

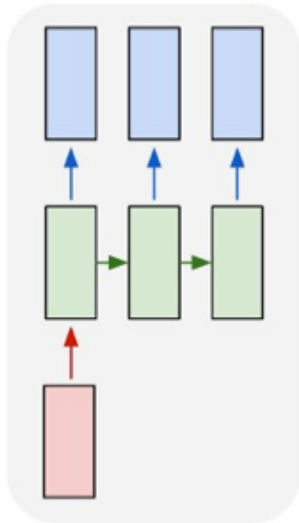
LSTM Recurrent Neural Network

one to one



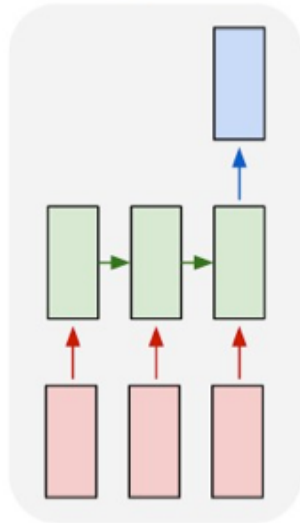
**Traditional
Neural
Network**

one to many



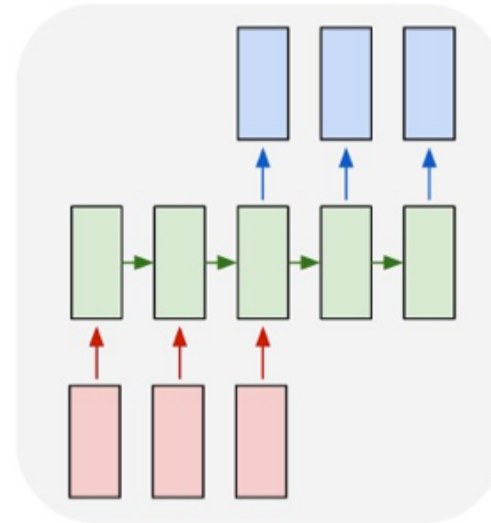
**Music
Generation**

many to one



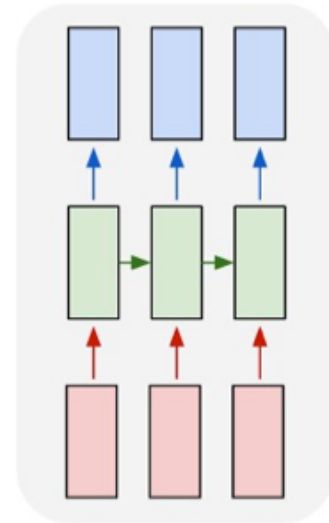
**Sentiment
Classification**

many to many



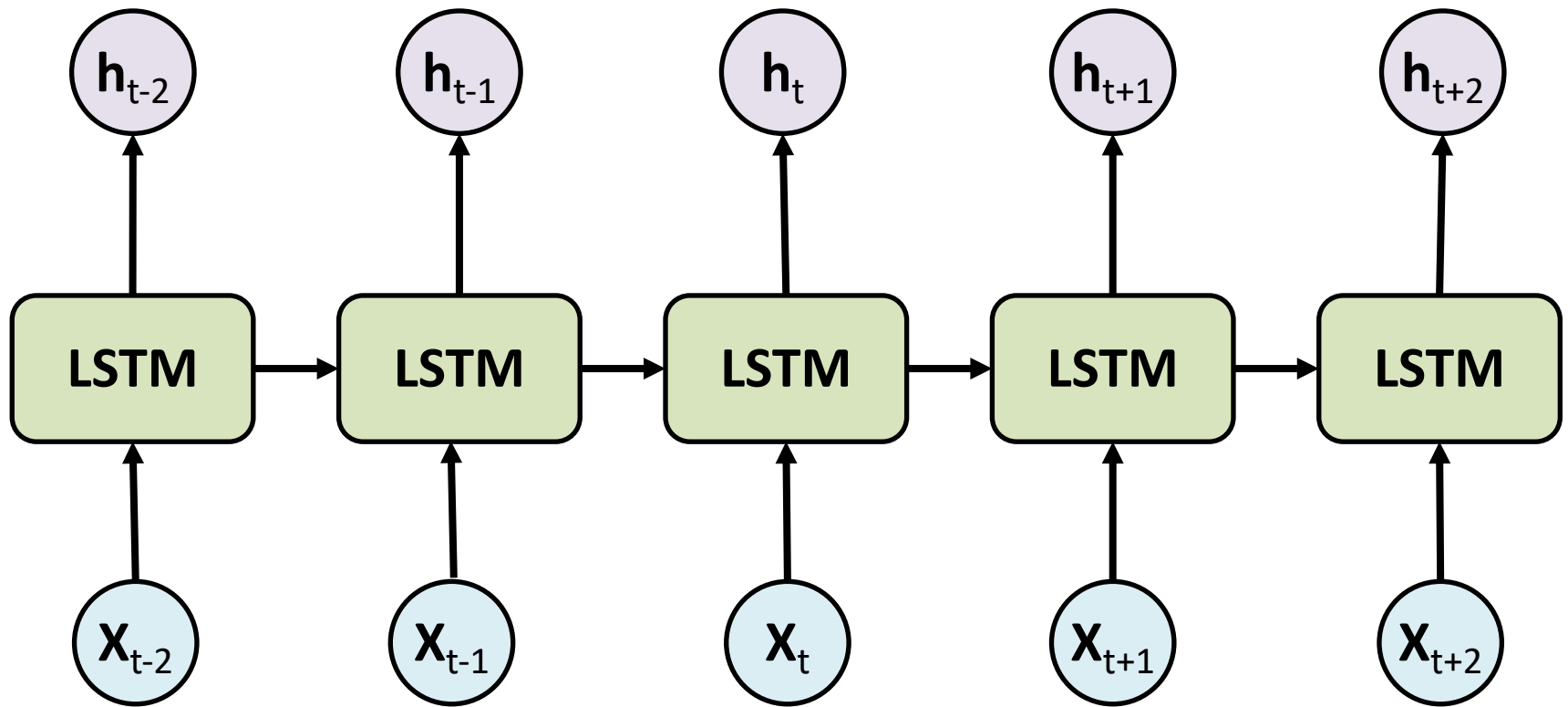
**Name
Entity
Recognition**

many to many

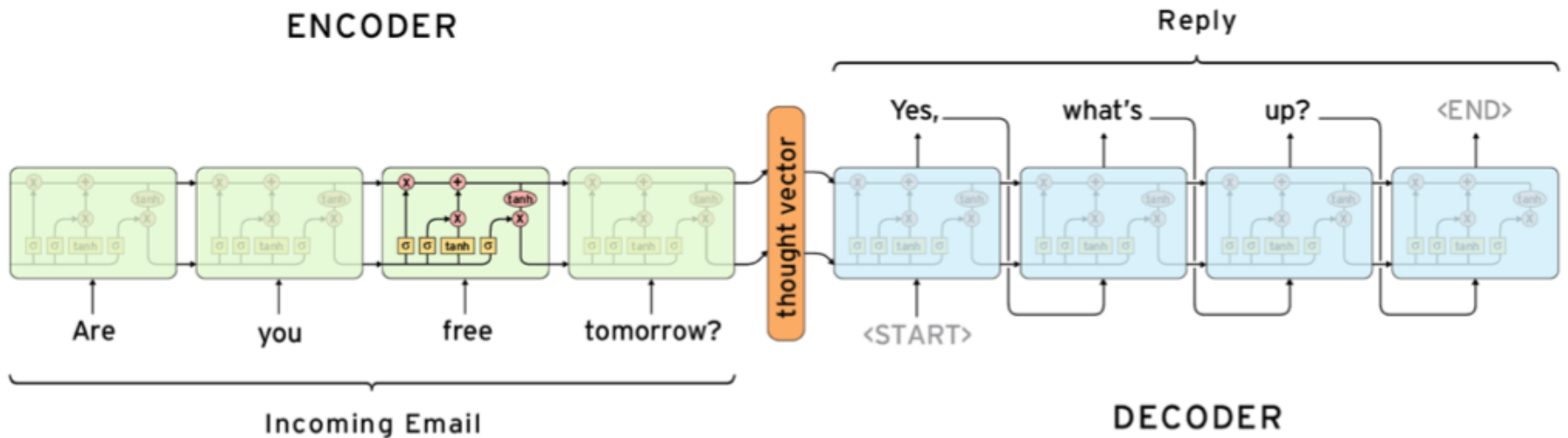


**Machine
Translation**

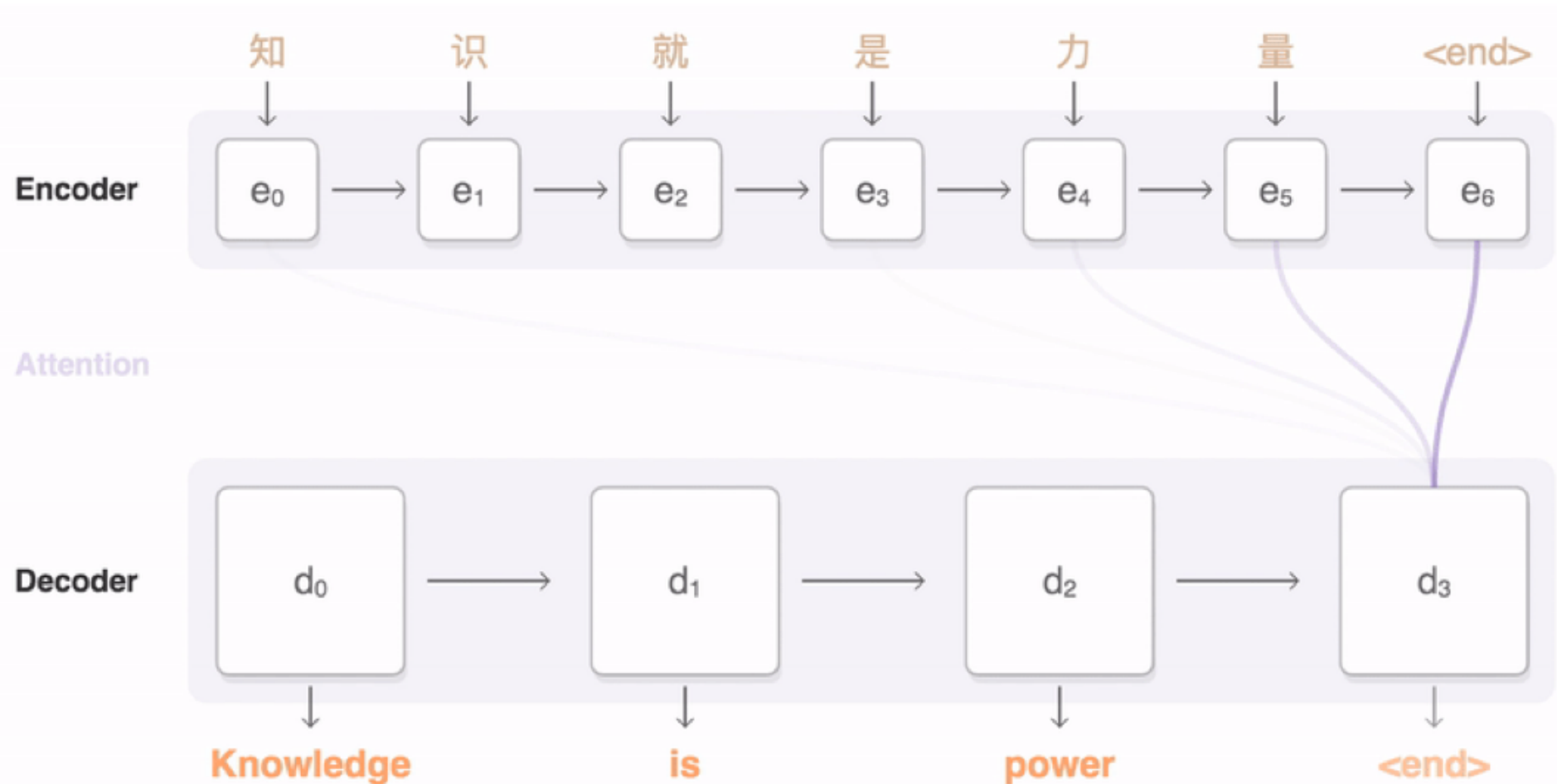
Long Short Term Memory (LSTM) for Time Series Forecasting



The Sequence to Sequence model (seq2seq)

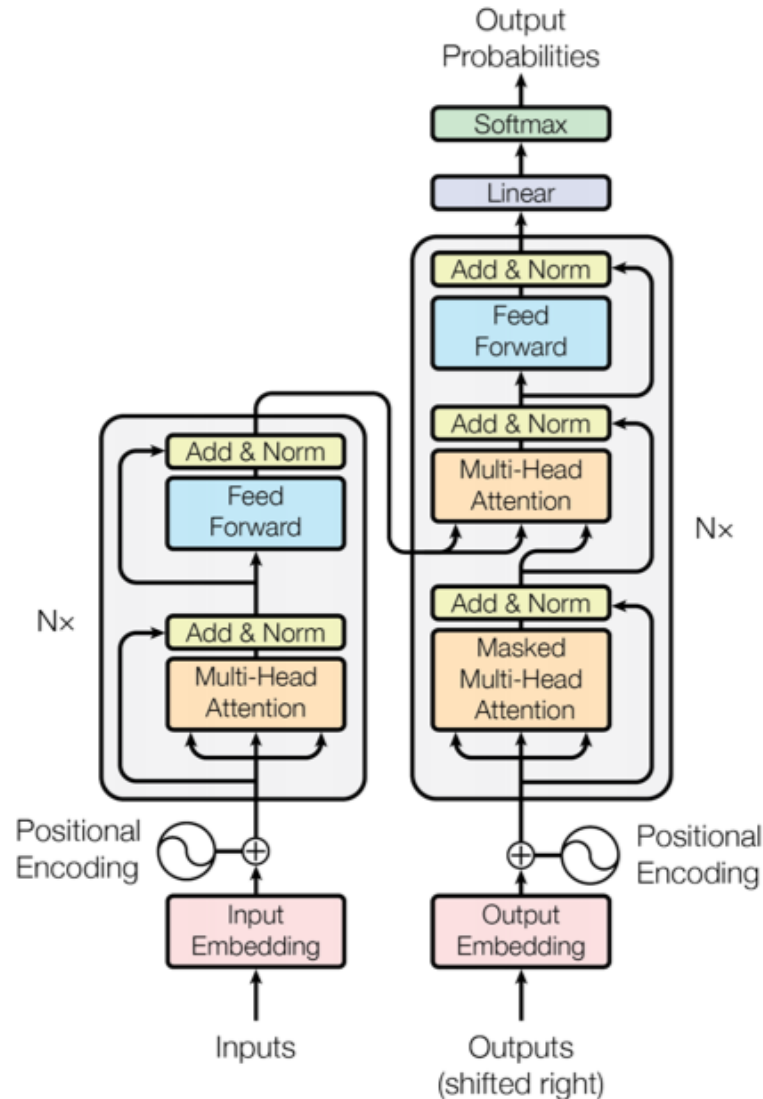


Sequence to Sequence (Seq2Seq)



Transformer (Attention is All You Need)

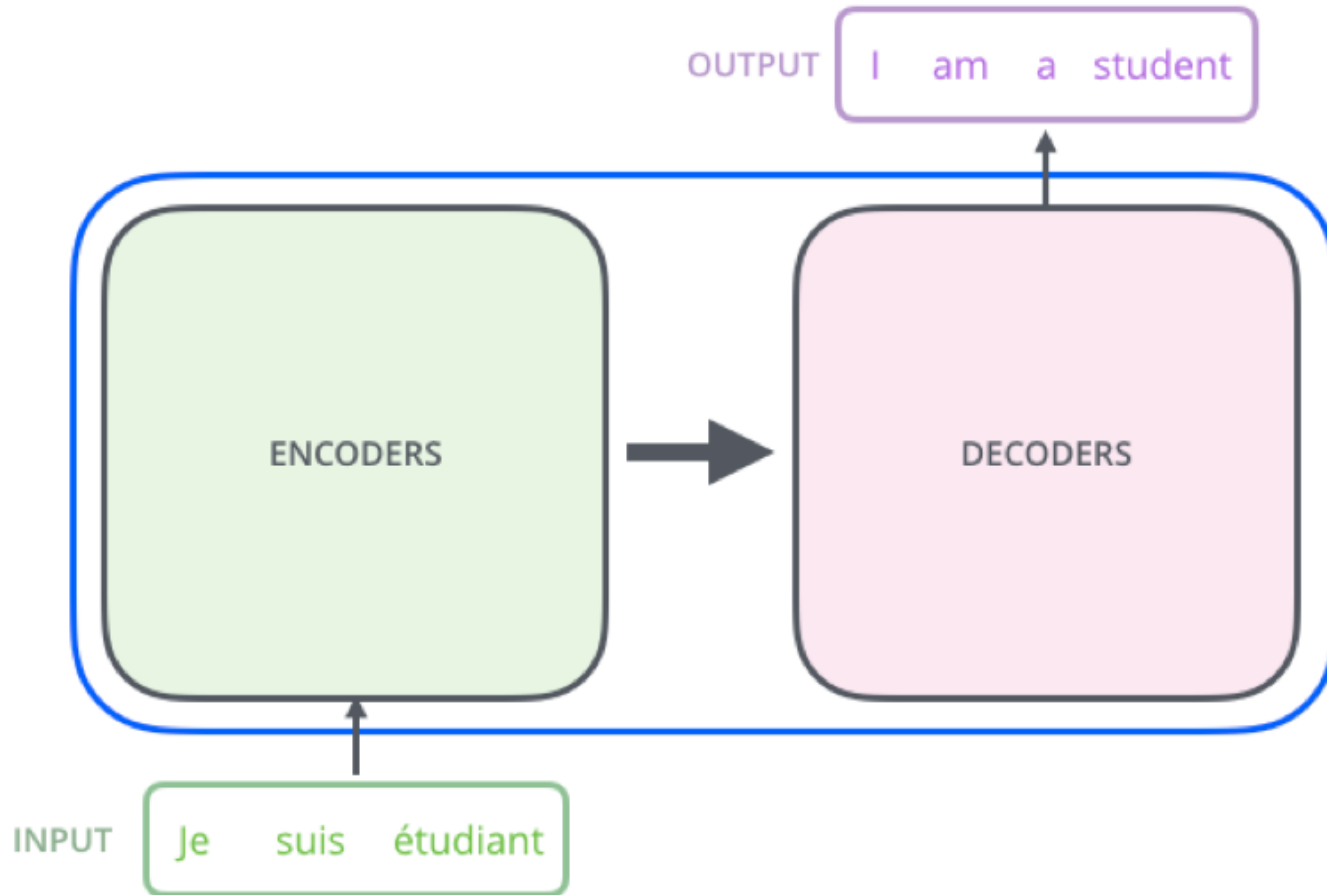
(Vaswani et al., 2017)



Transformer

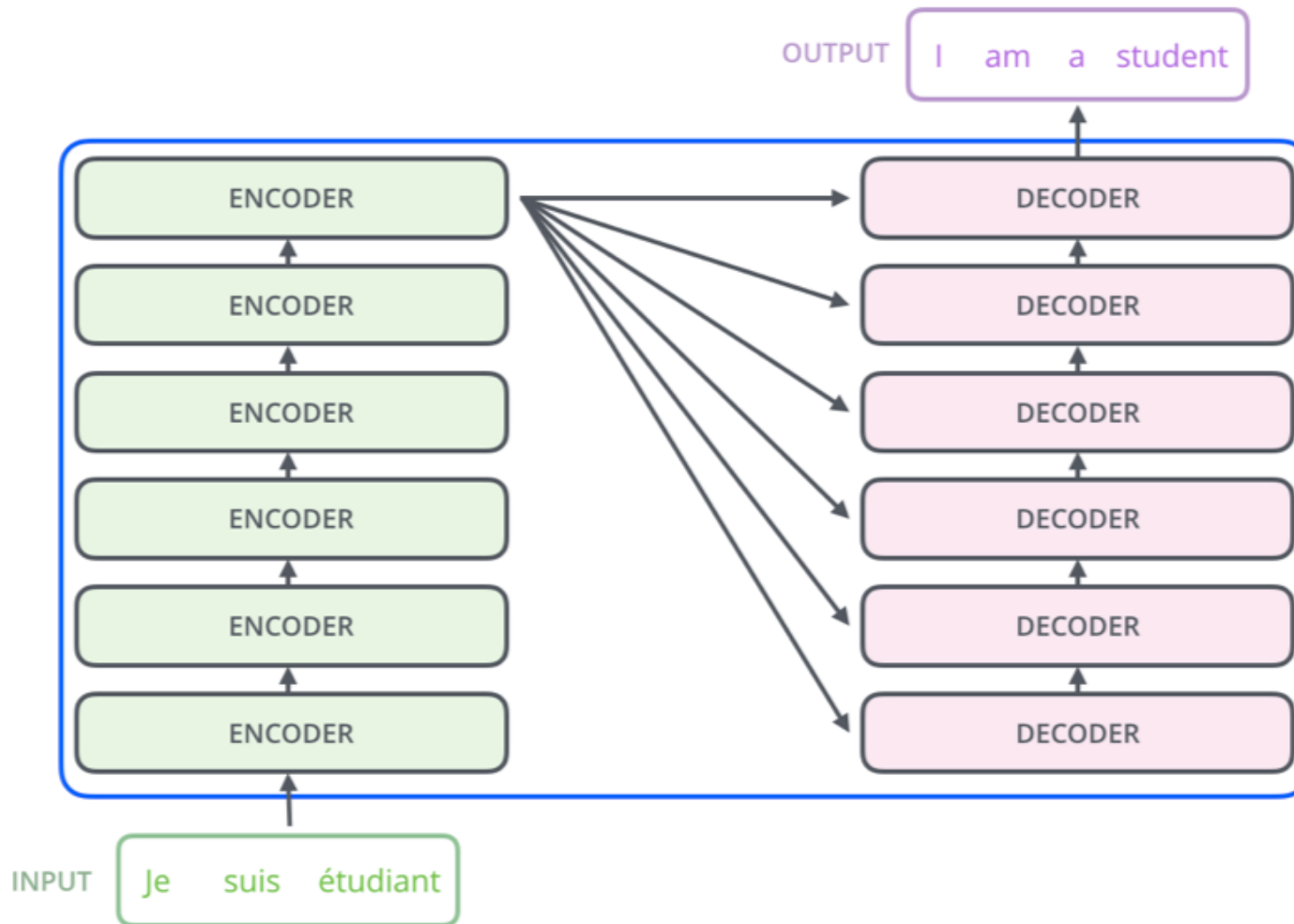


Transformer Encoder Decoder



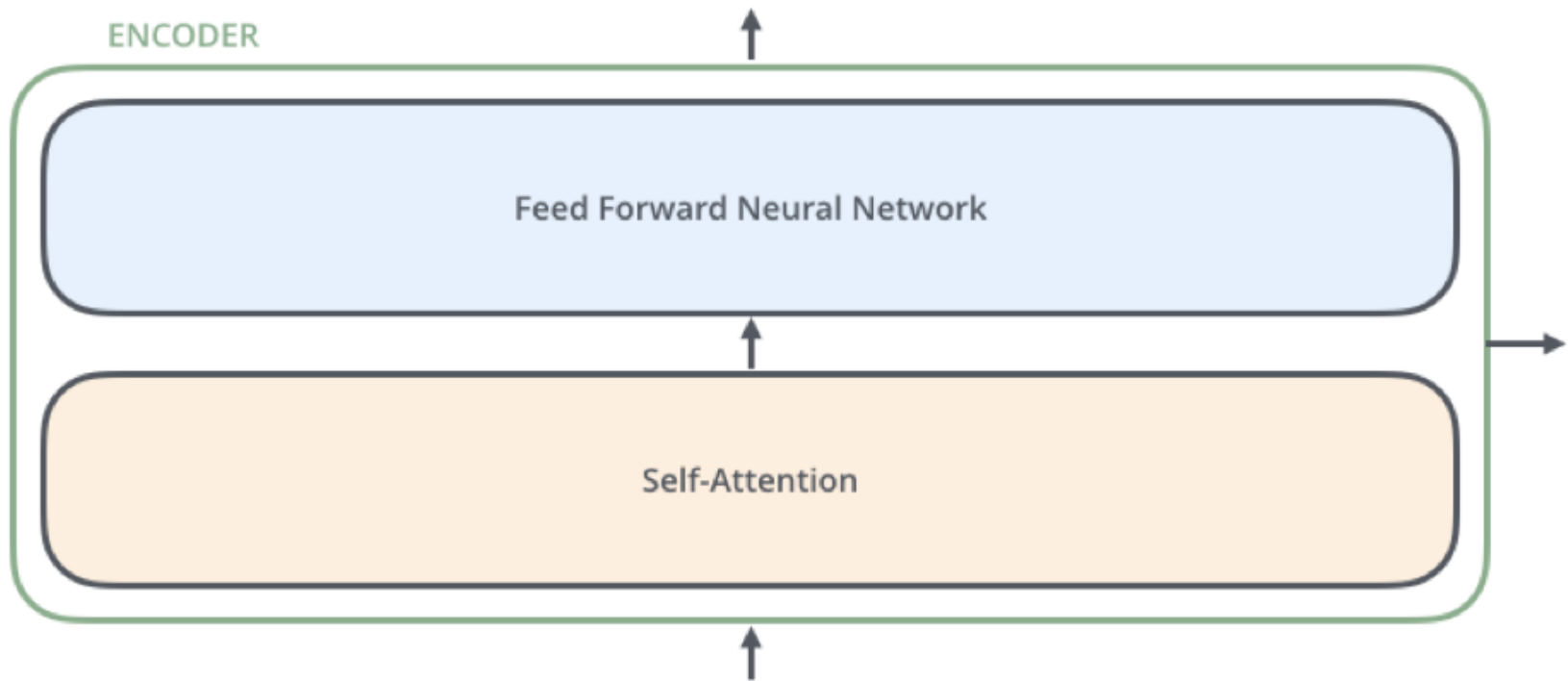
Transformer

Encoder Decoder Stack

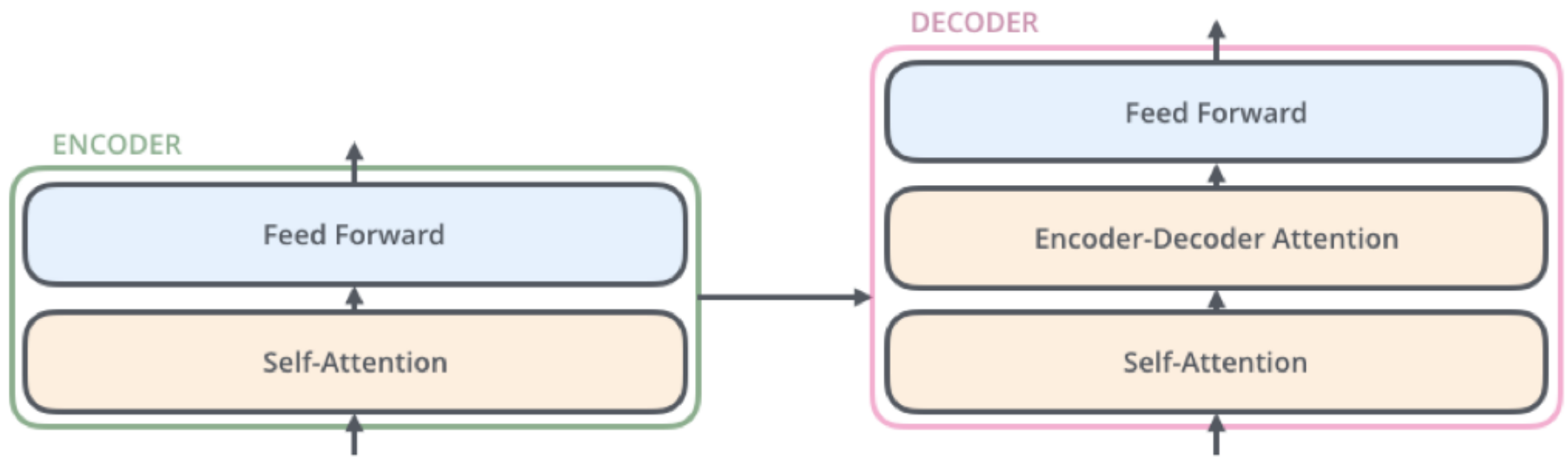


Transformer

Encoder Self-Attention



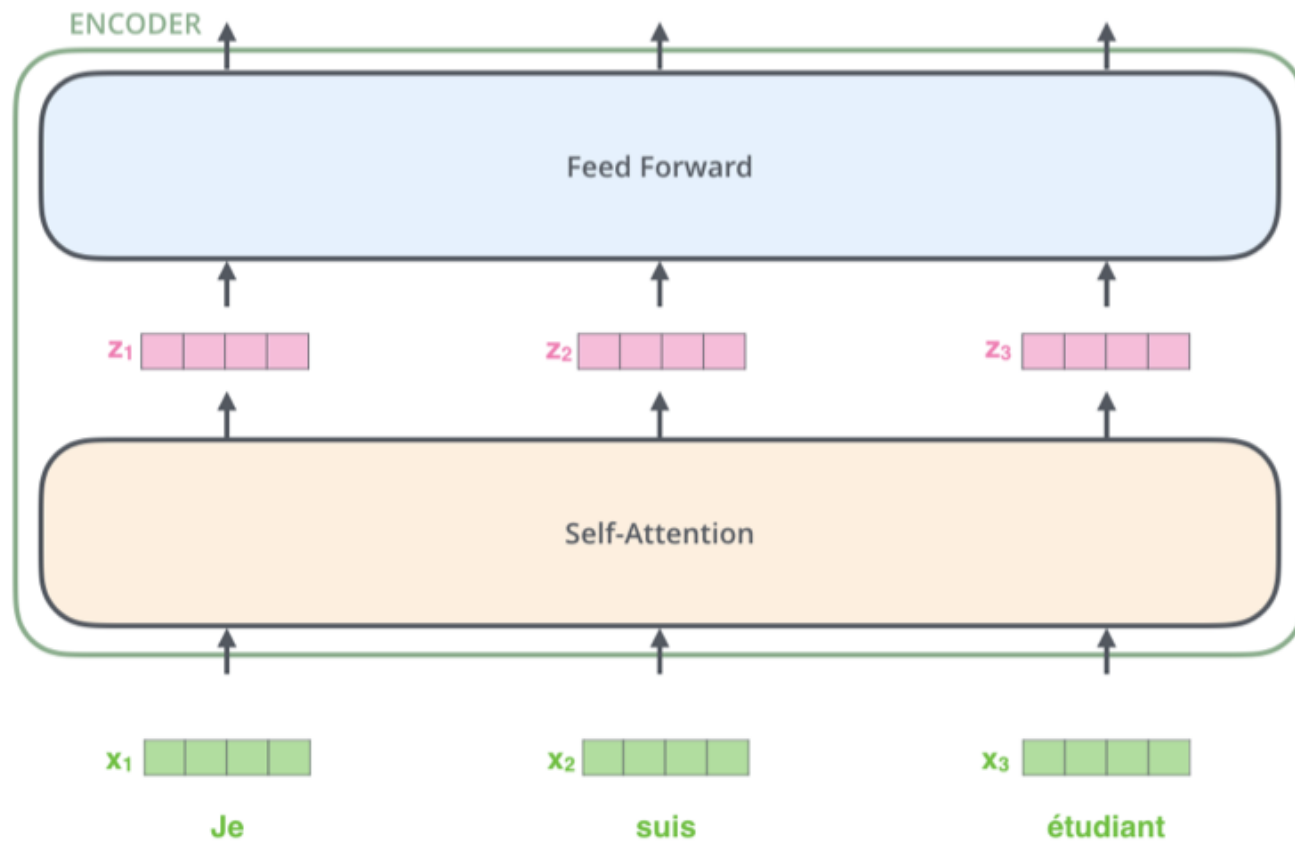
Transformer Decoder



Transformer

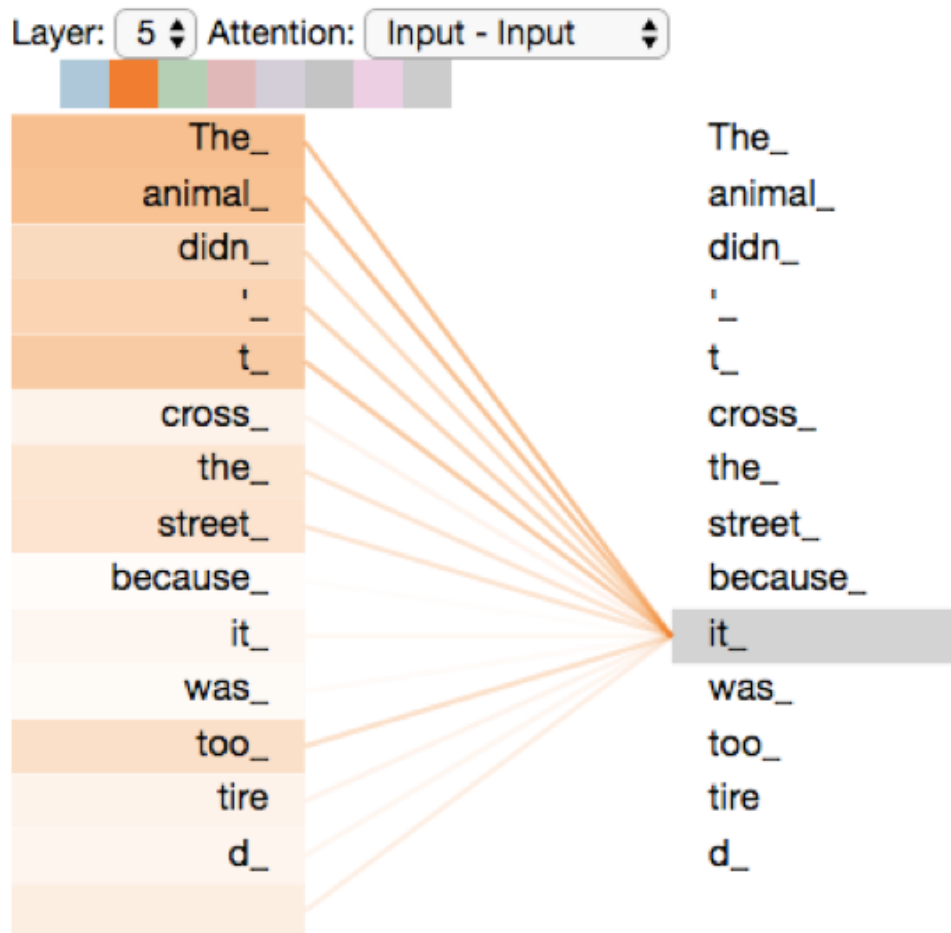
Encoder with Tensors

Word Embeddings



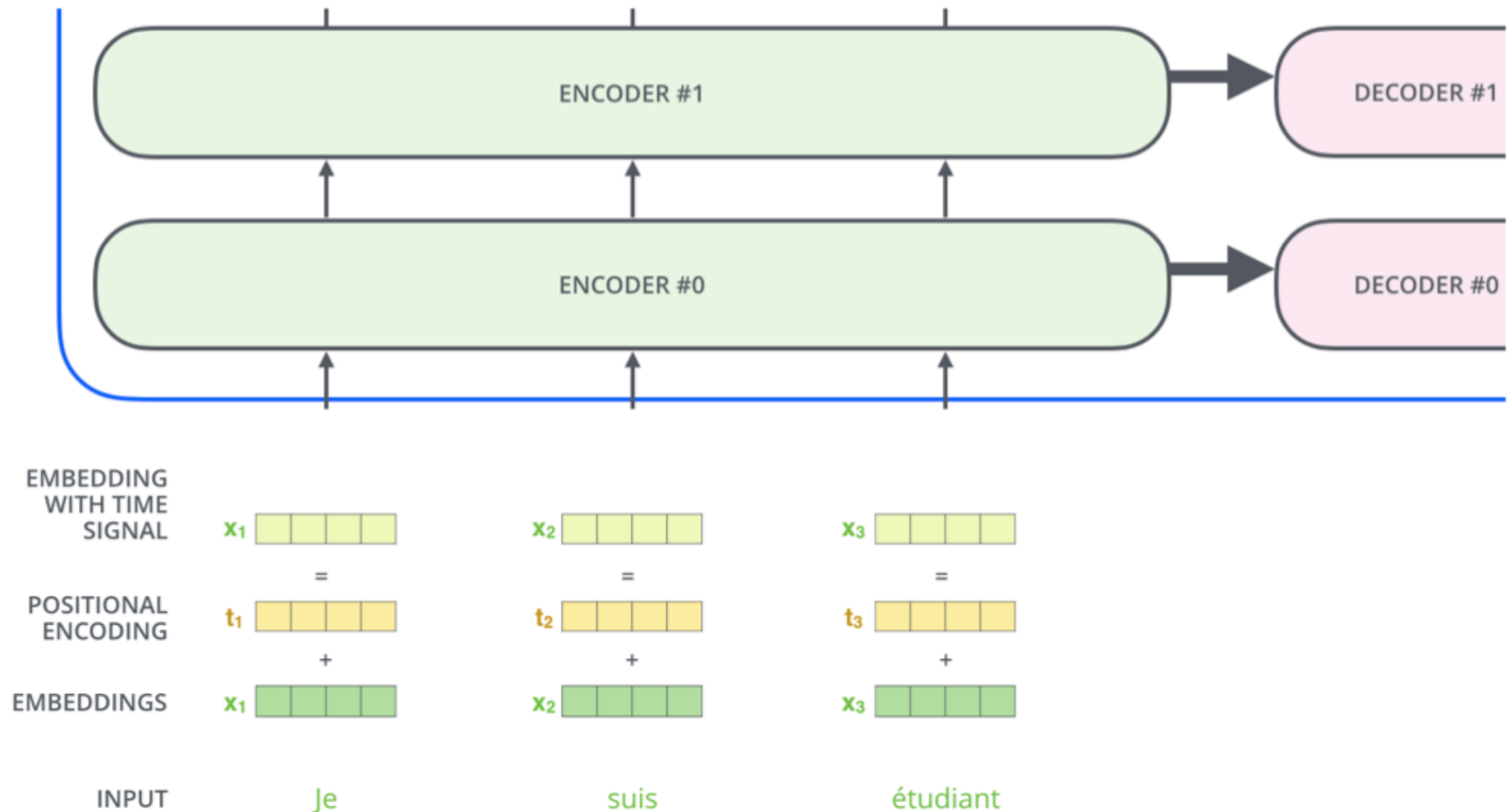
Transformer

Self-Attention Visualization



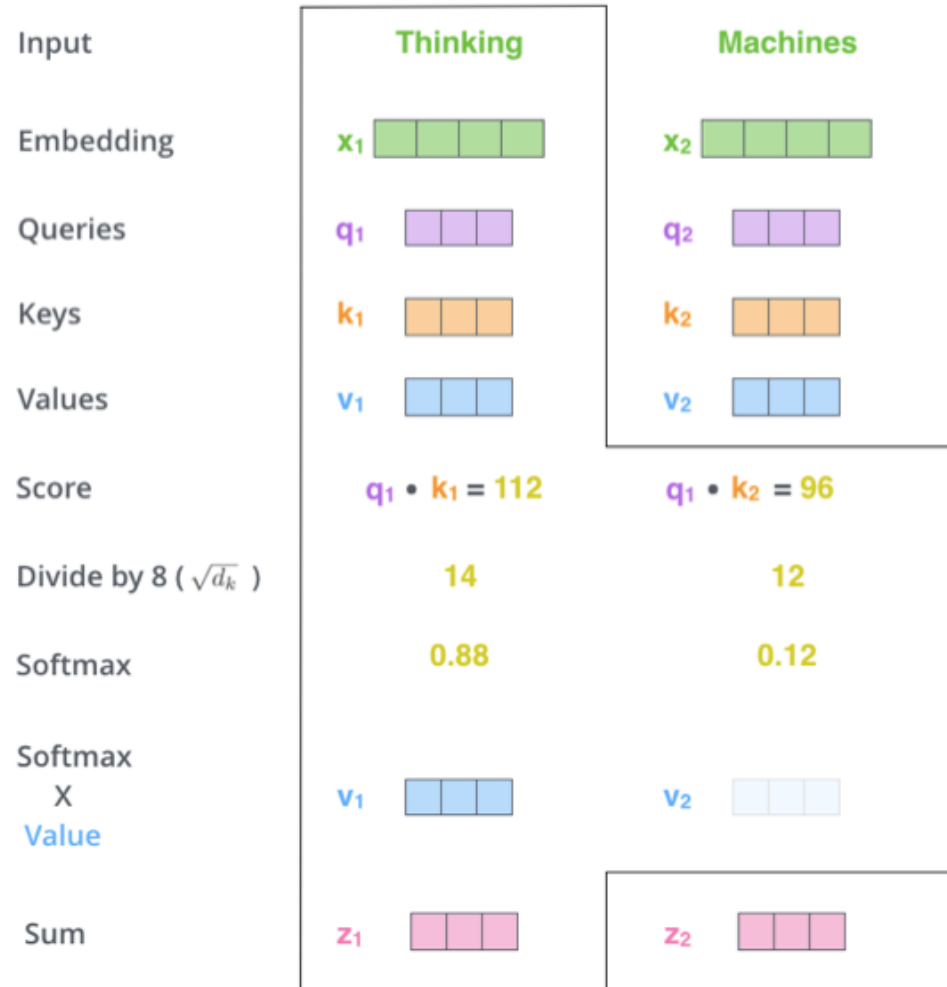
Transformer

Positional Encoding Vectors



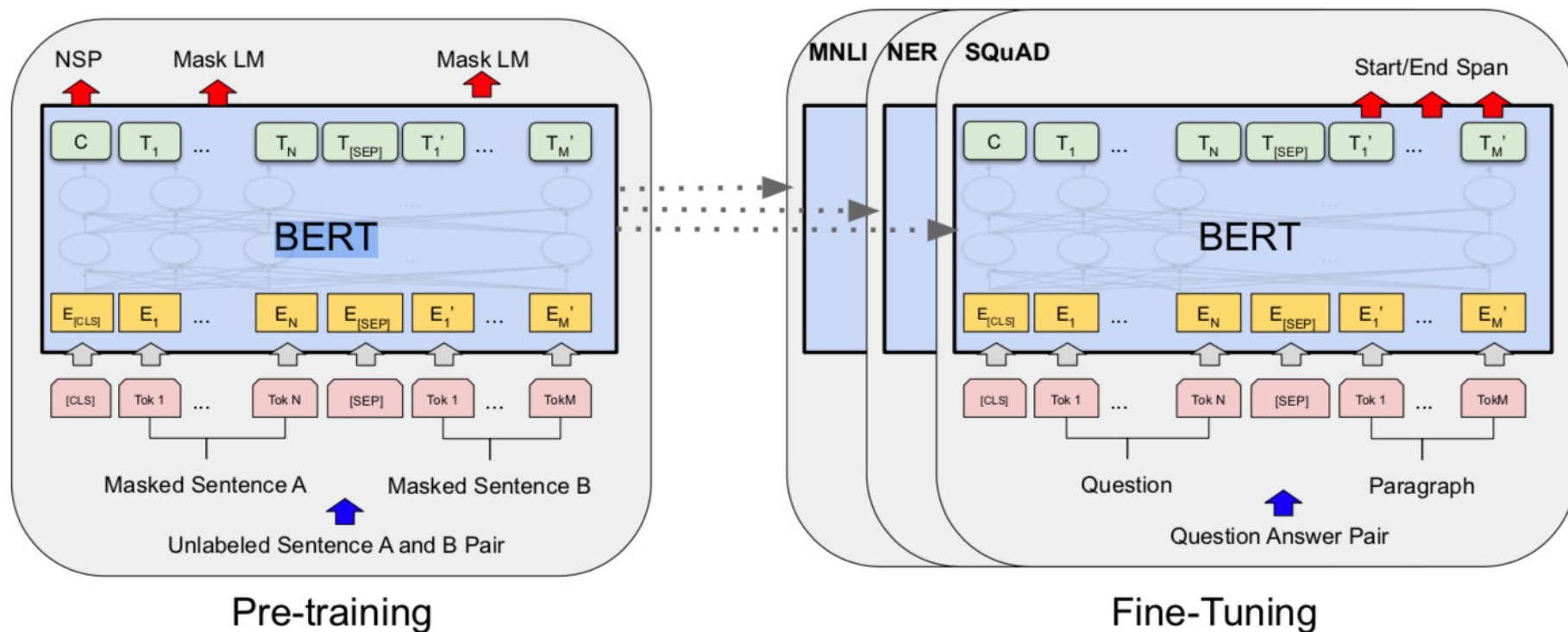
Transformer

Self-Attention Softmax Output



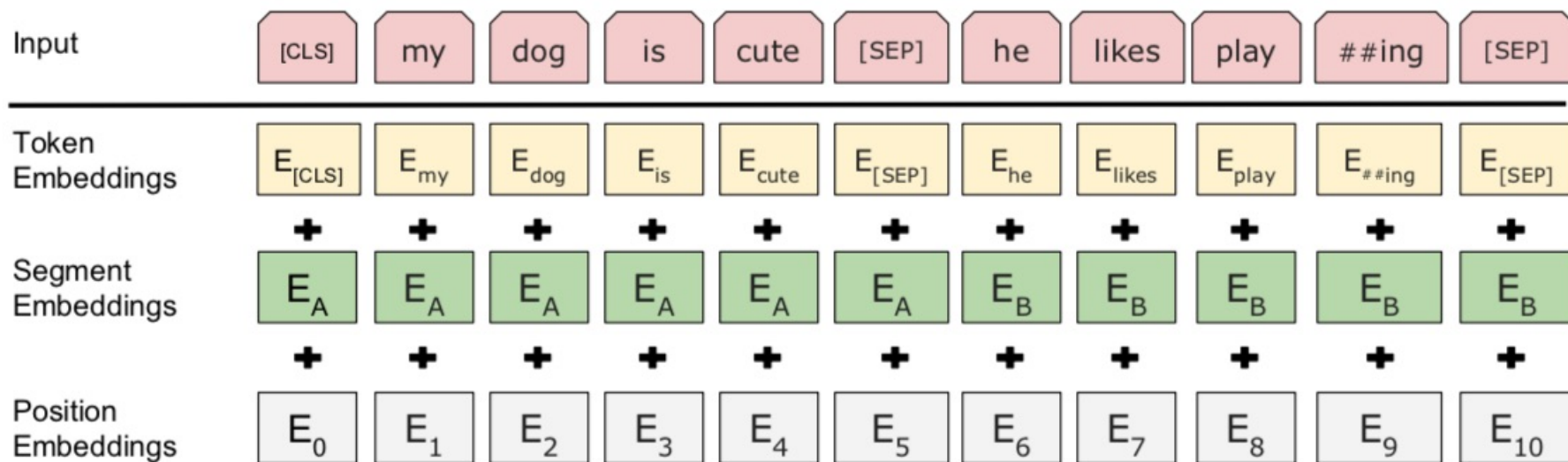
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)
Overall pre-training and fine-tuning procedures for BERT

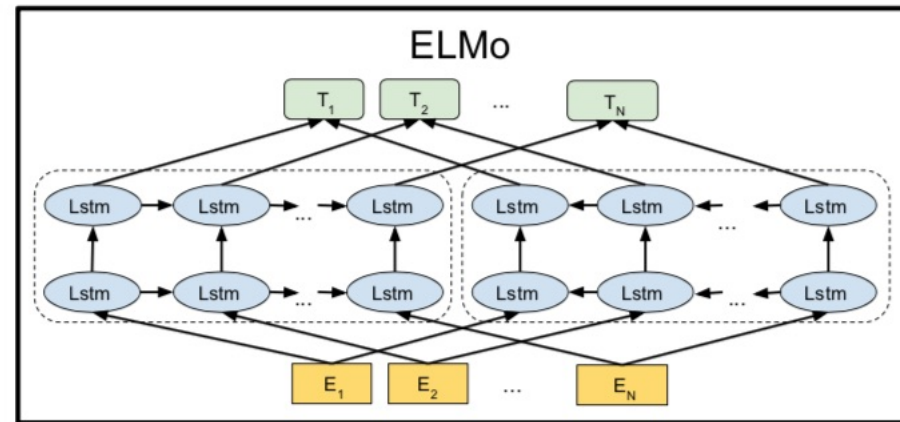
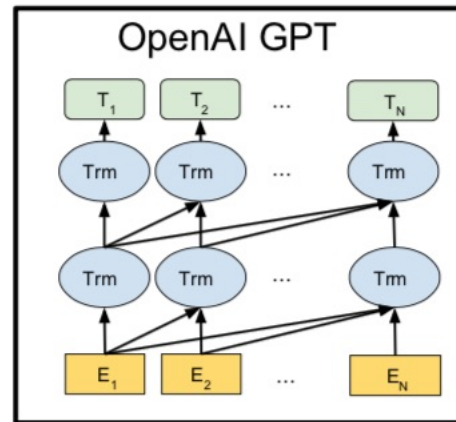
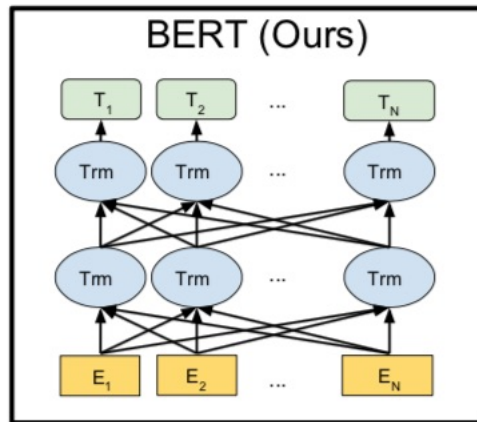


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

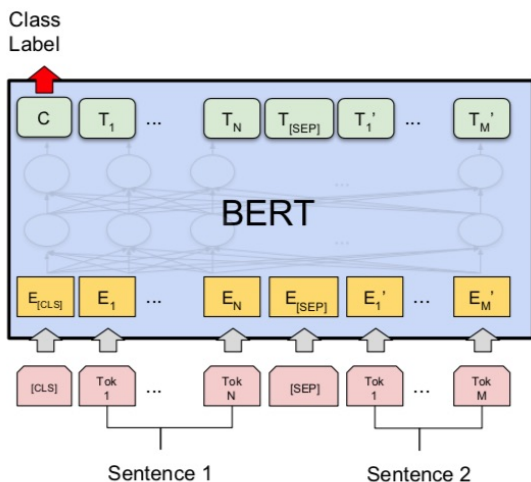
BERT (Bidirectional Encoder Representations from Transformers)
BERT input representation



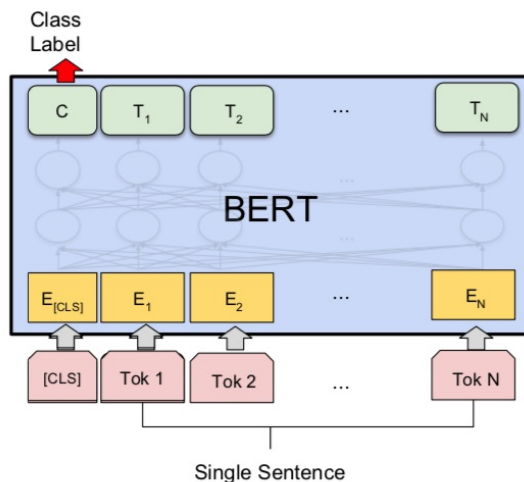
BERT, OpenAI GPT, ELMo



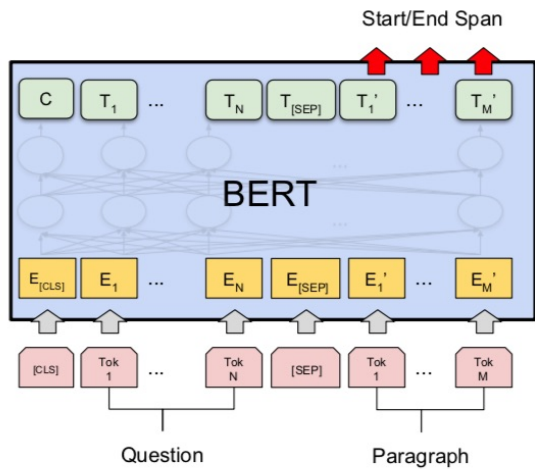
Fine-tuning BERT on Different Tasks



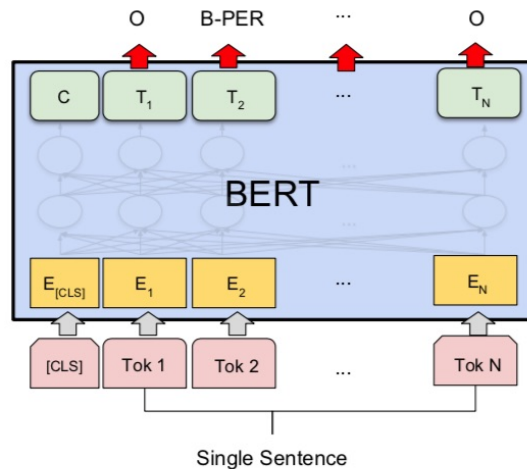
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1



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

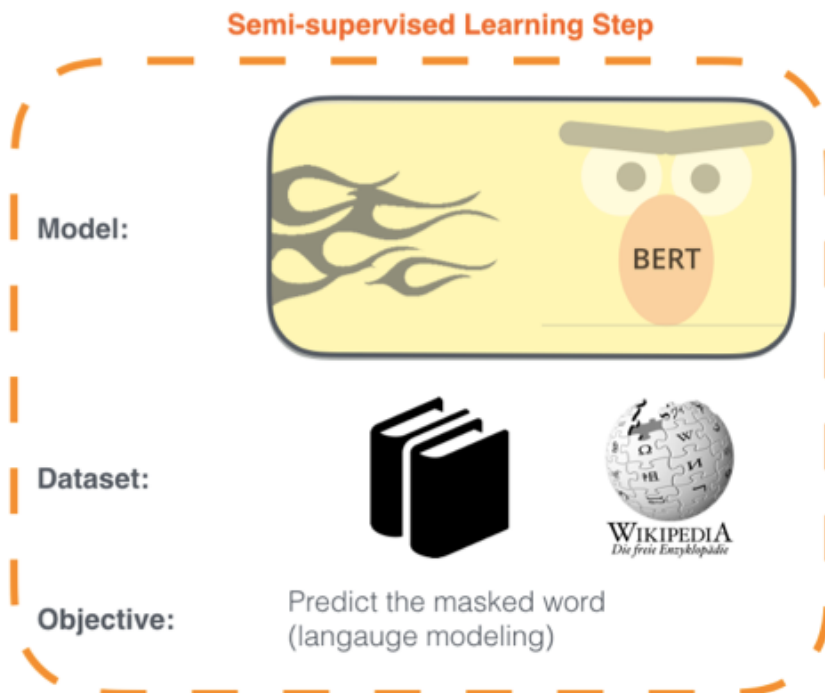
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

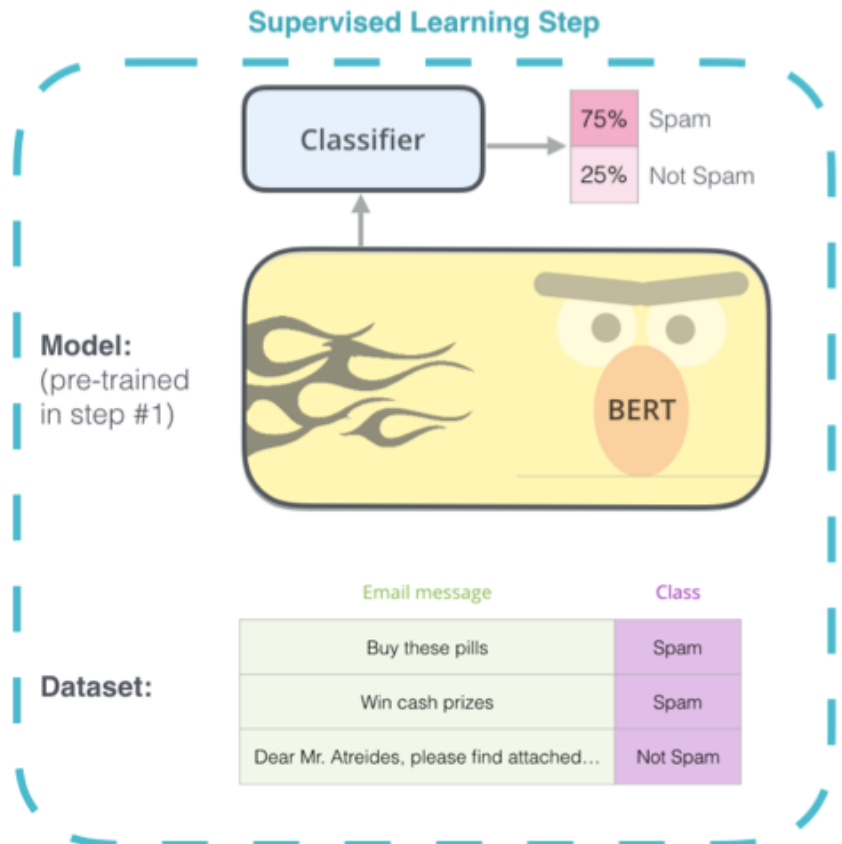
Illustrated BERT

1 - **Semi-supervised** training on large amounts of text (books, wikipedia..etc).

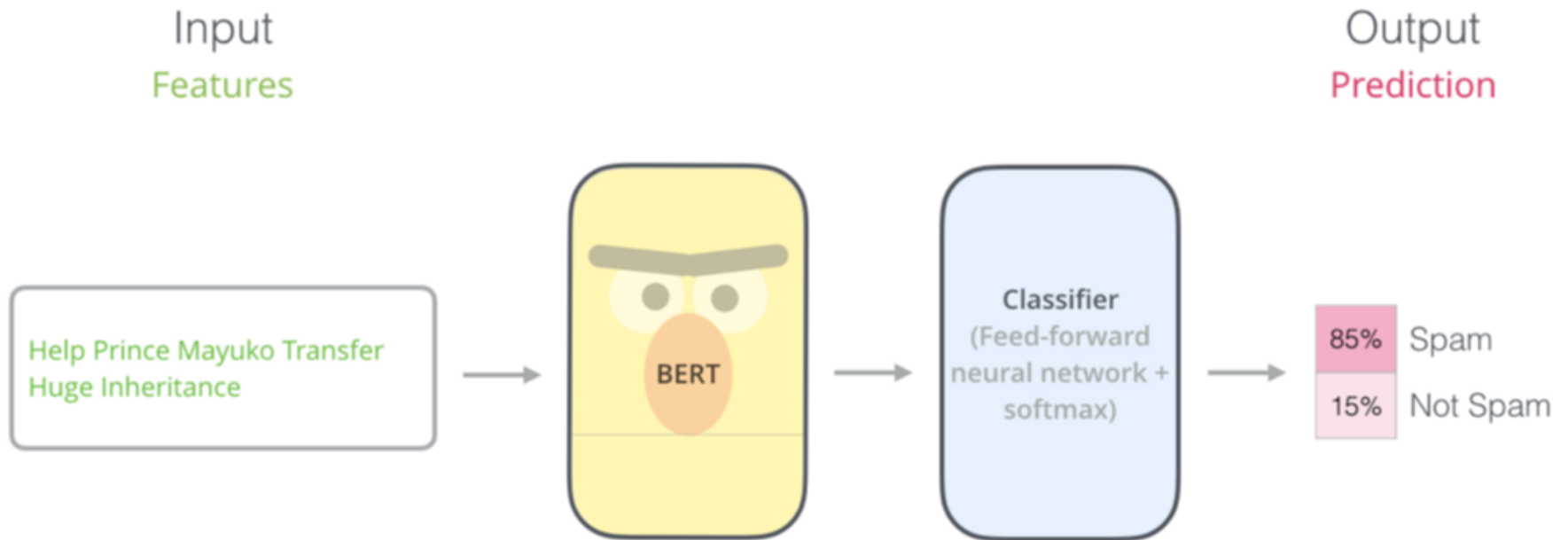
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



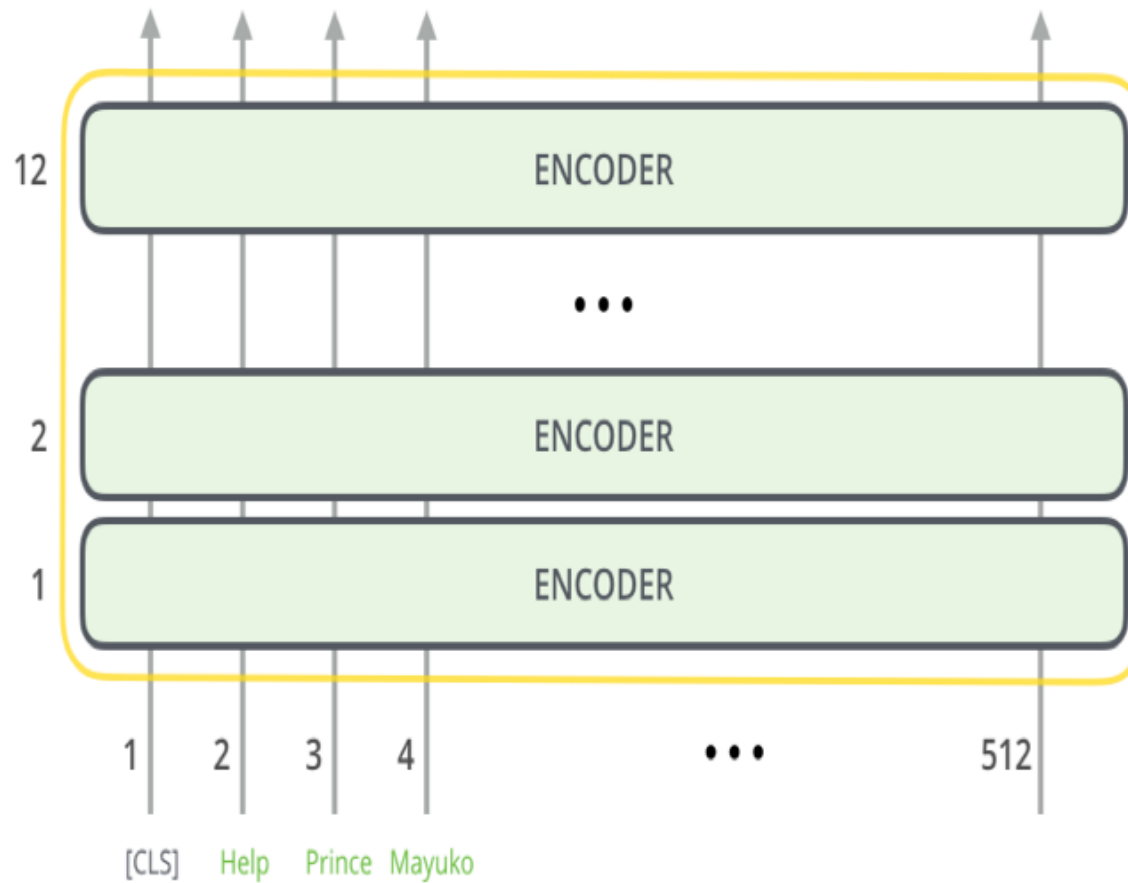
2 - **Supervised** training on a specific task with a labeled dataset.



BERT Classification Input Output

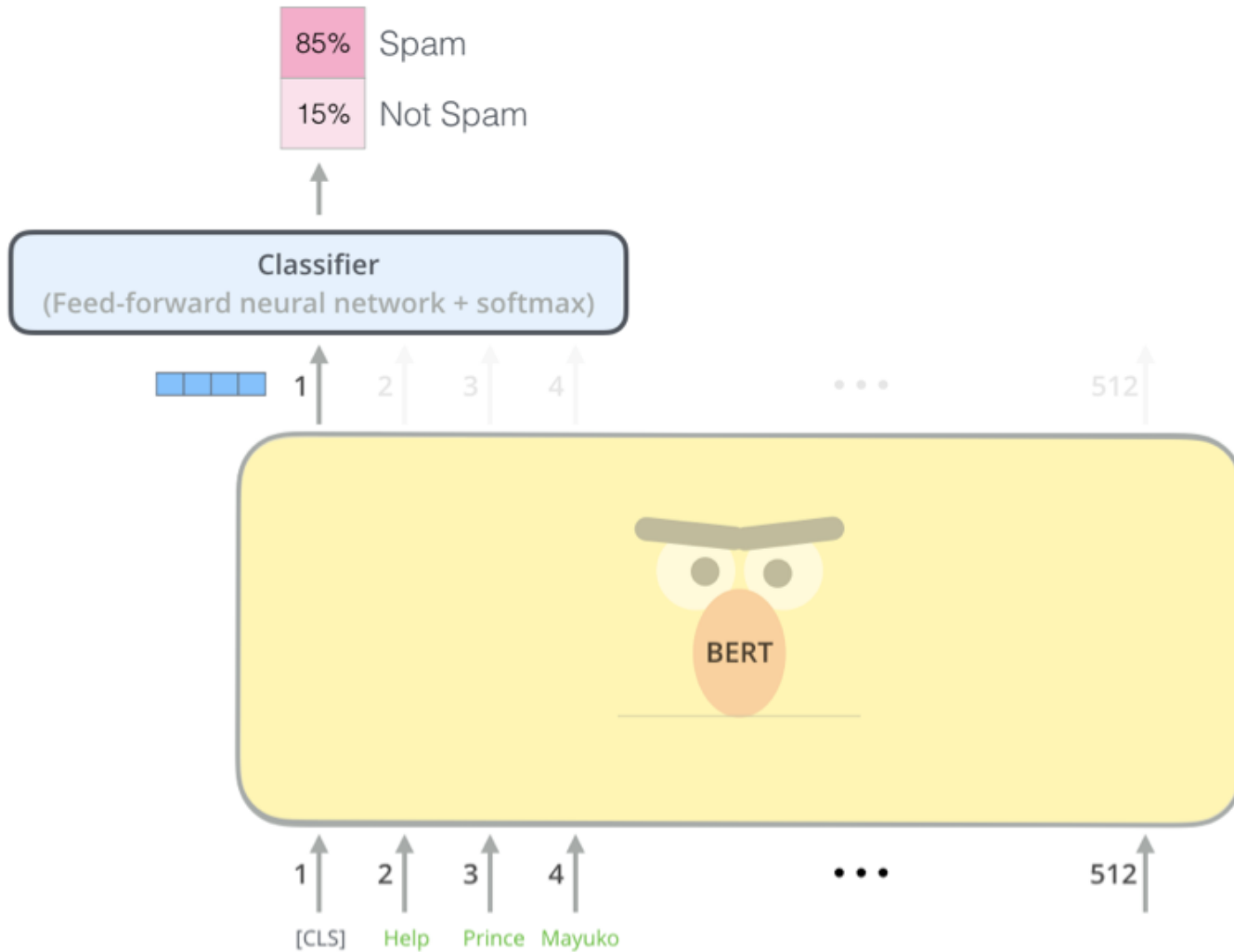


BERT Encoder Input



BERT

BERT Classifier



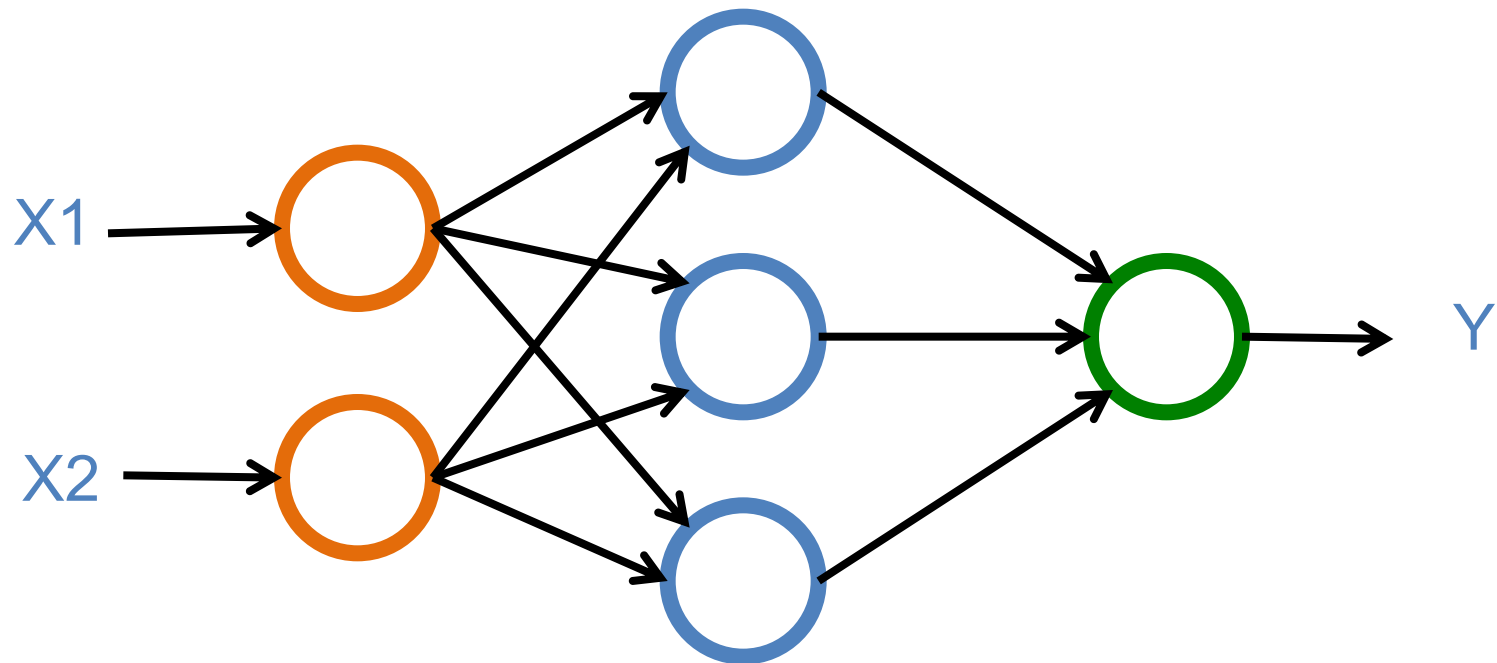
Source: Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), <http://jalamar.github.io/illustrated-bert/>

Neural Networks

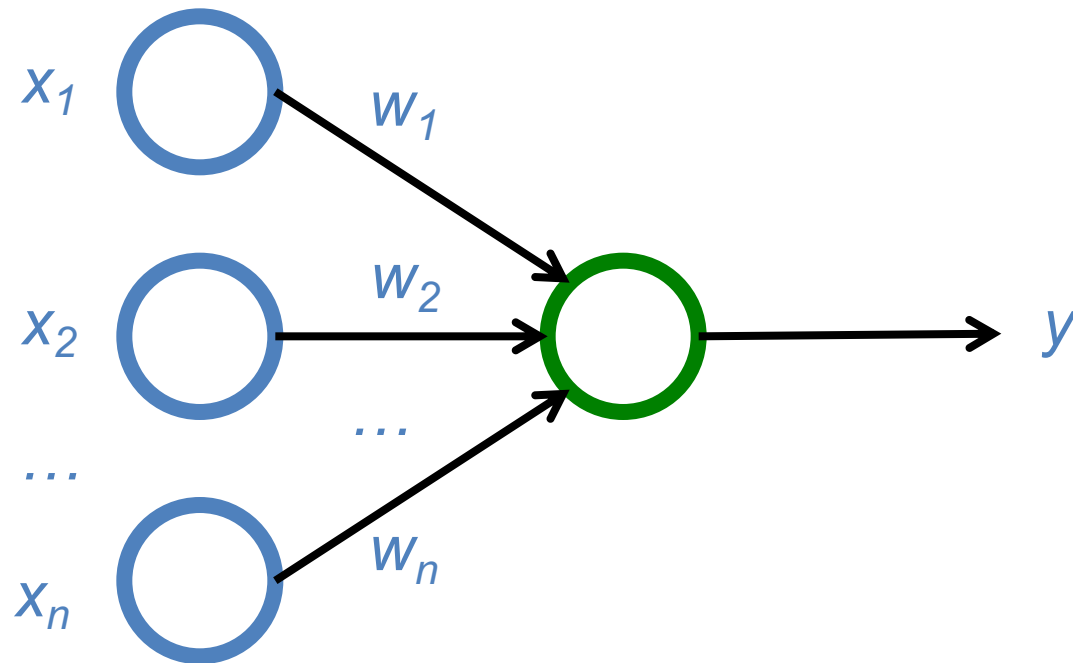
Input Layer
(X)

Hidden Layer
(H)

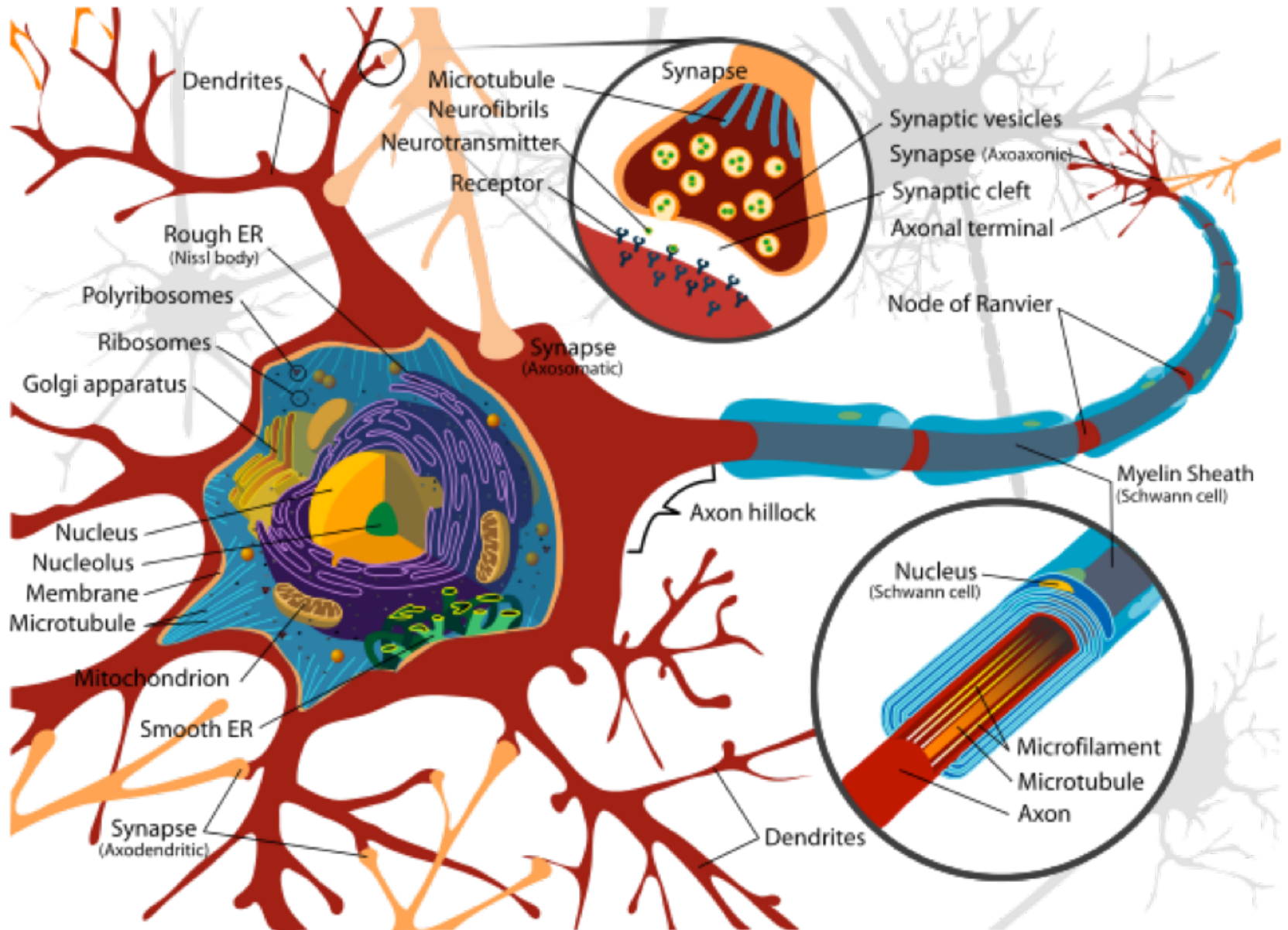
Output Layer
(Y)



The Neuron

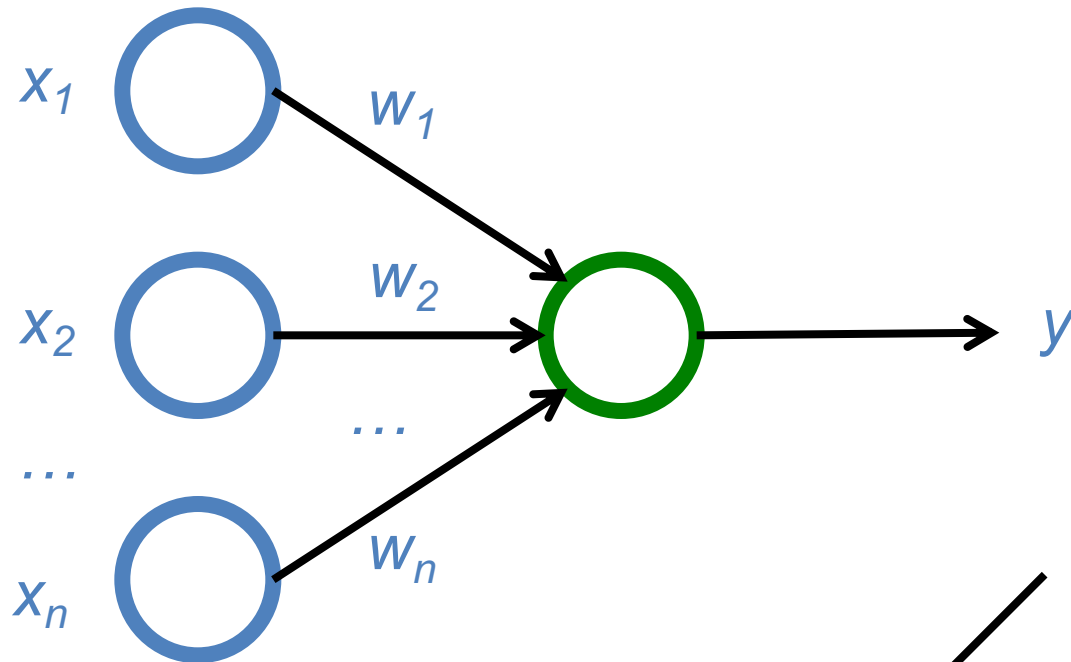


Neuron and Synapse



The Neuron

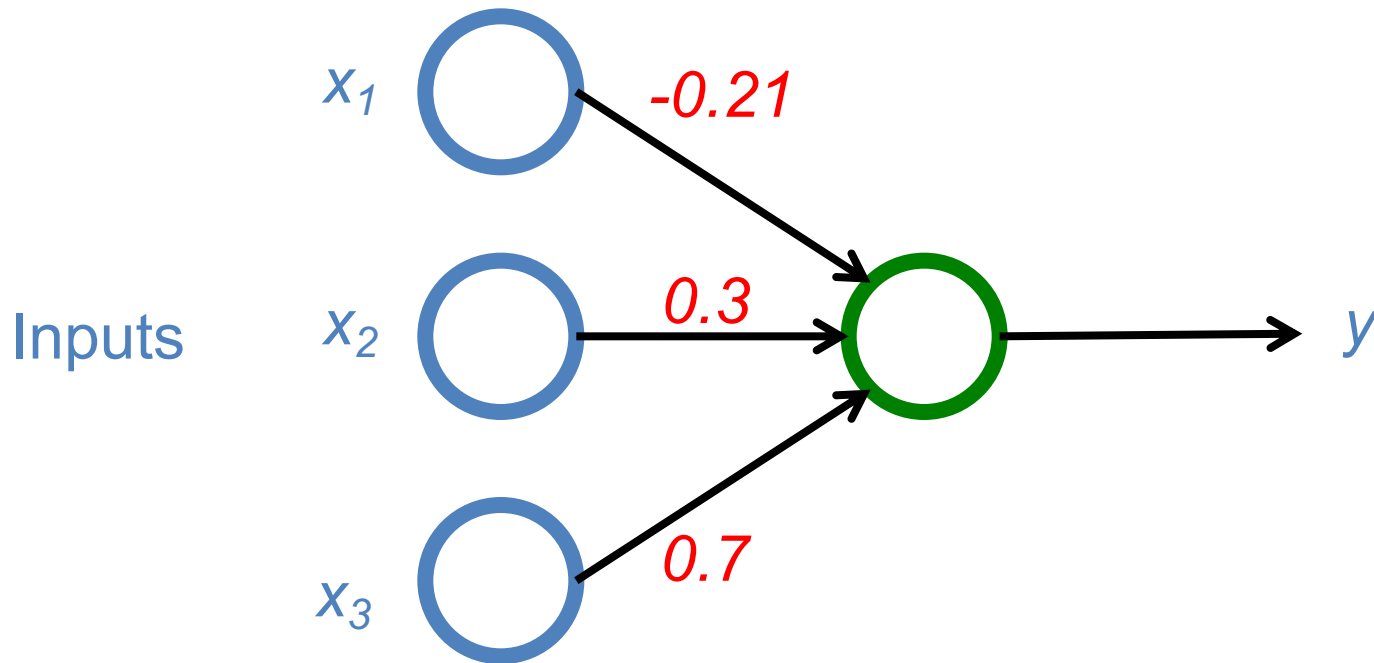
$$y = F\left(\sum_i w_i x_i\right)$$



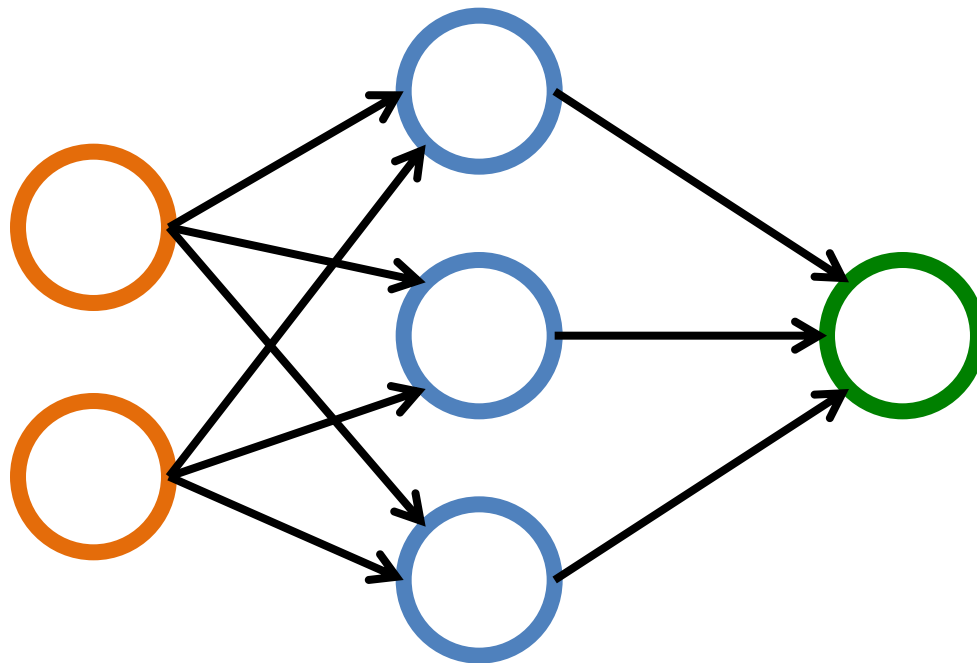
$$F(x) = \max(0, x)$$

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

Weights



Neural Networks

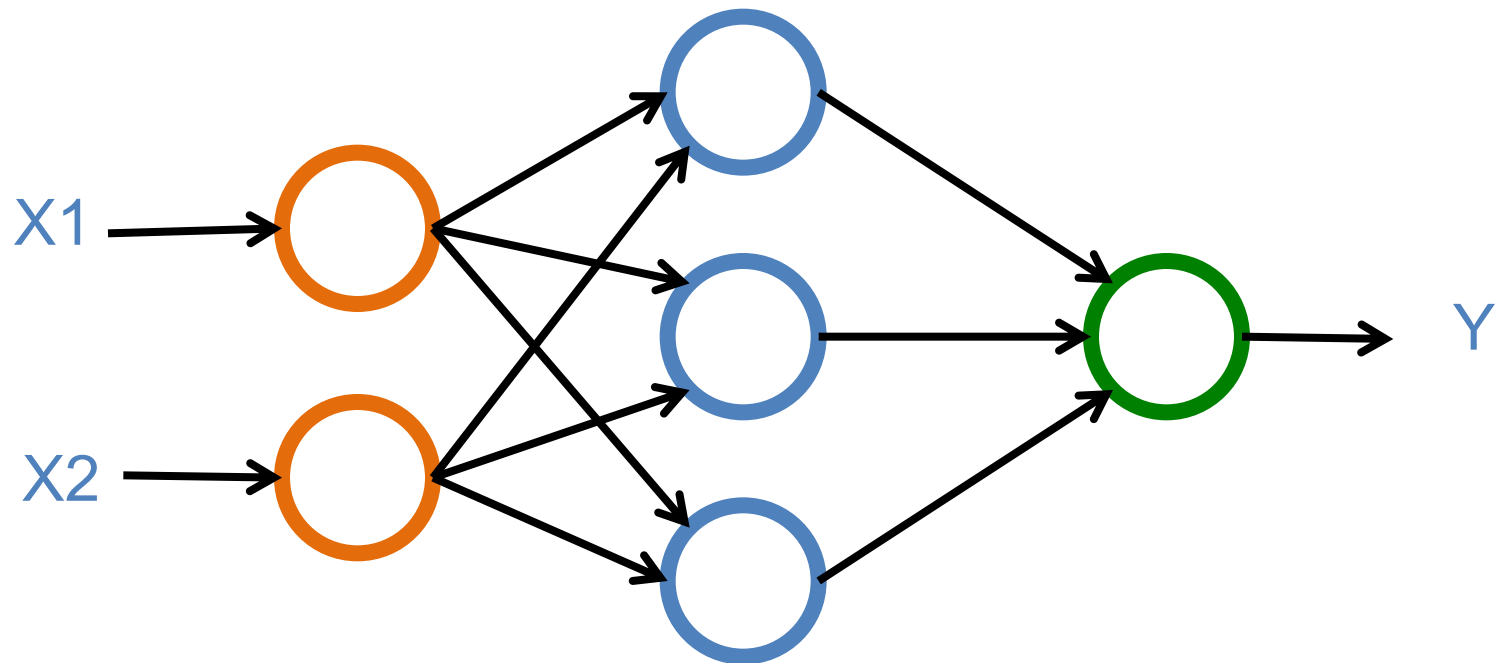


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



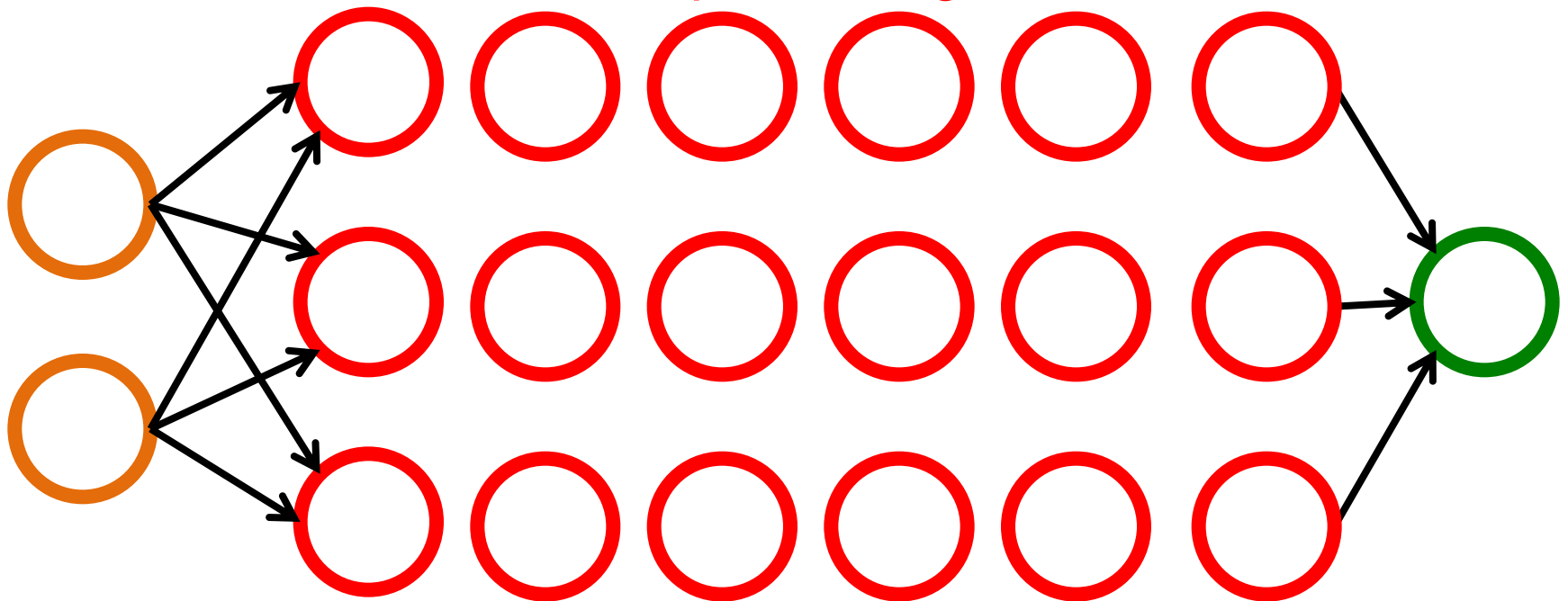
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

Output Layer
(Y)

Deep Neural Networks
Deep Learning

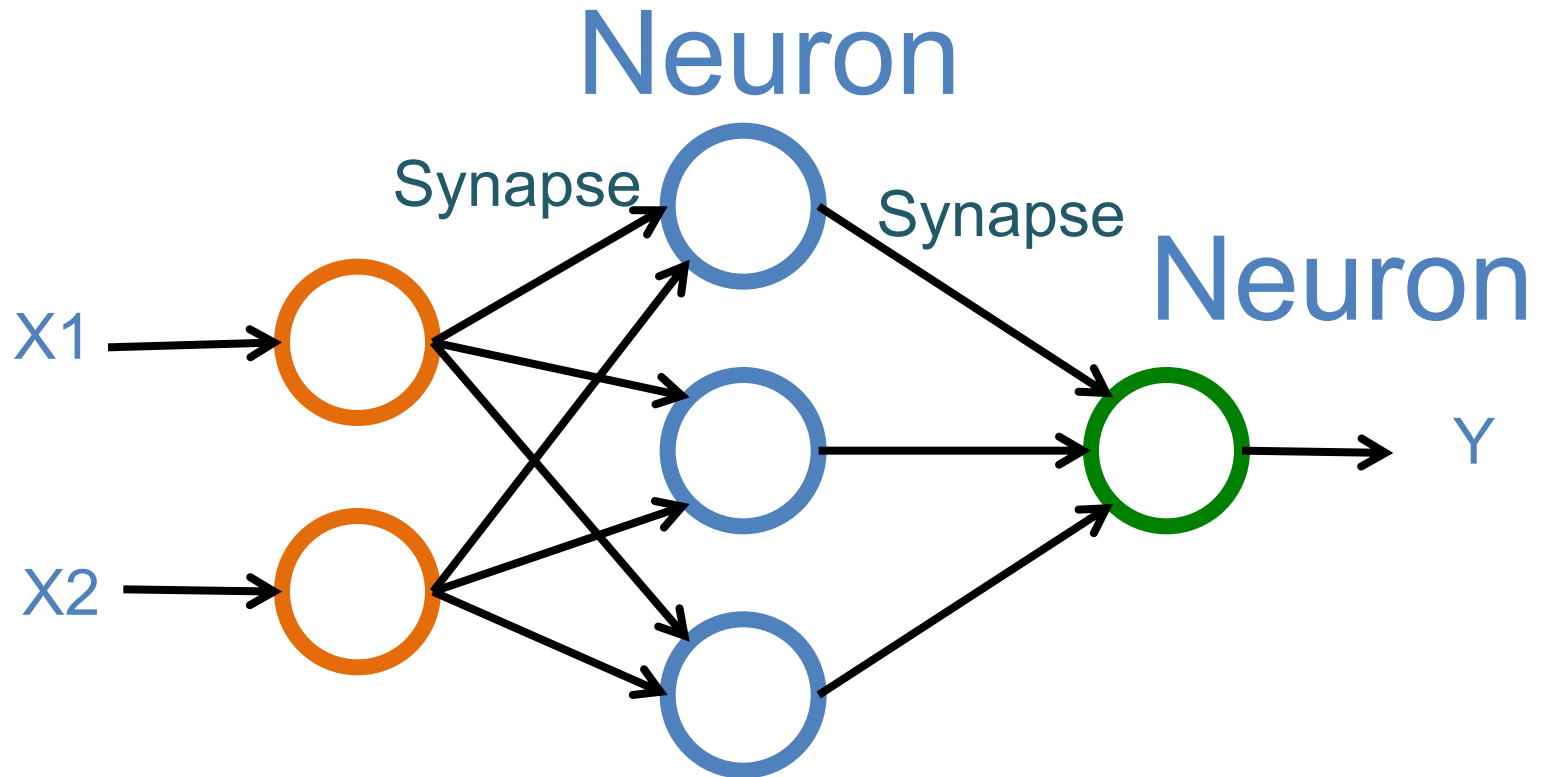


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

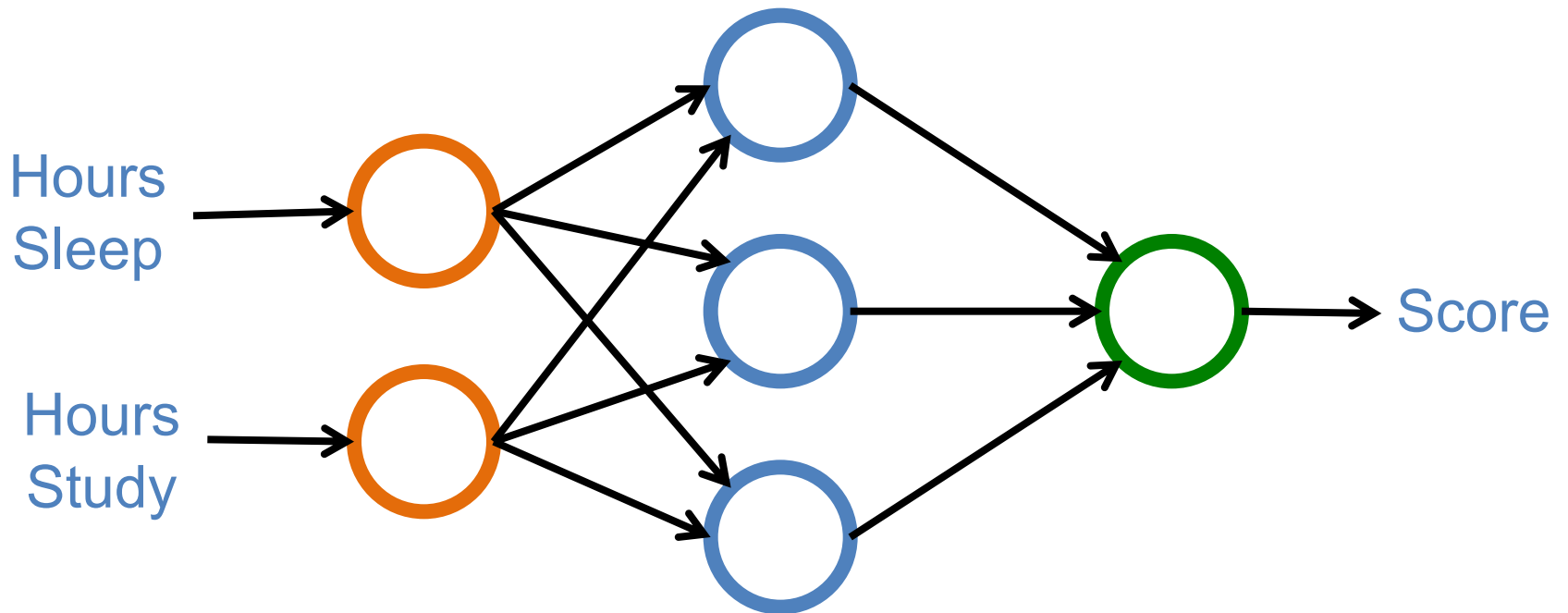


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

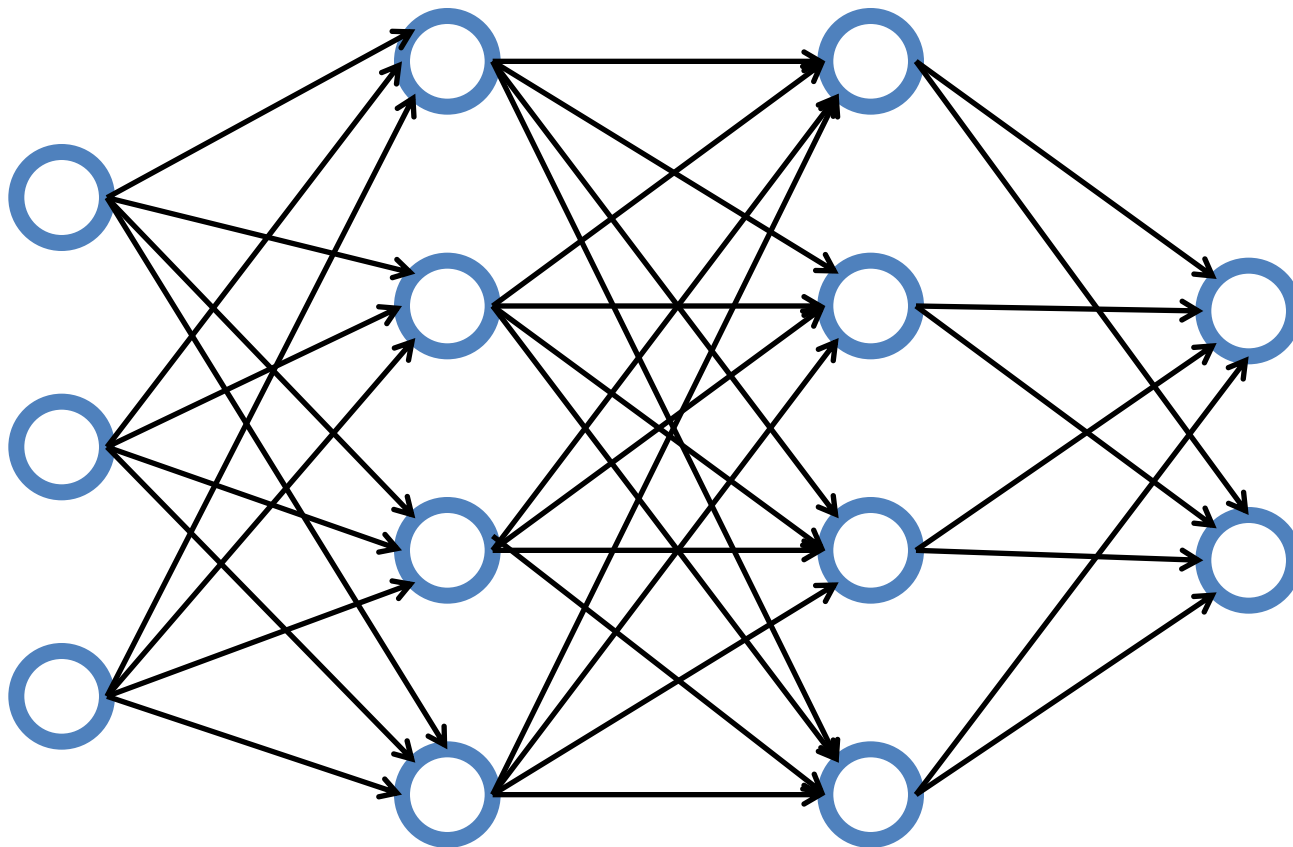


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

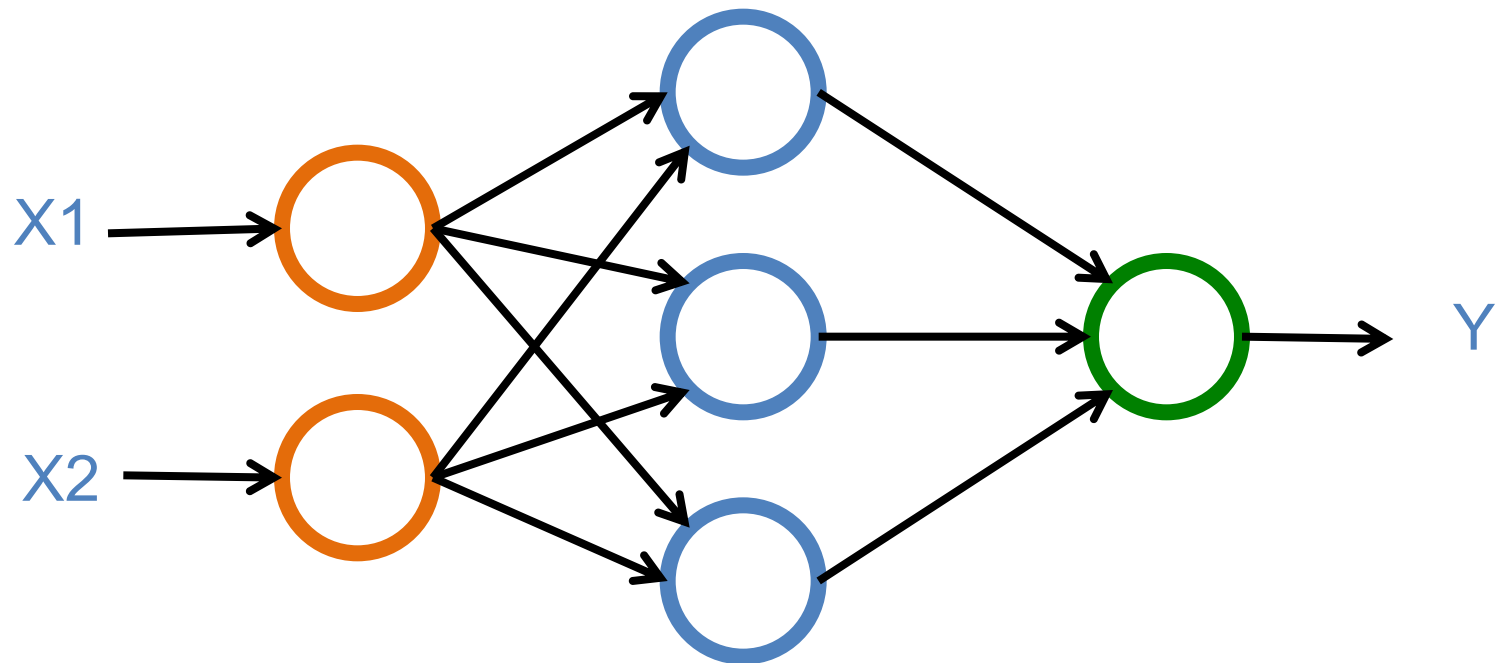


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

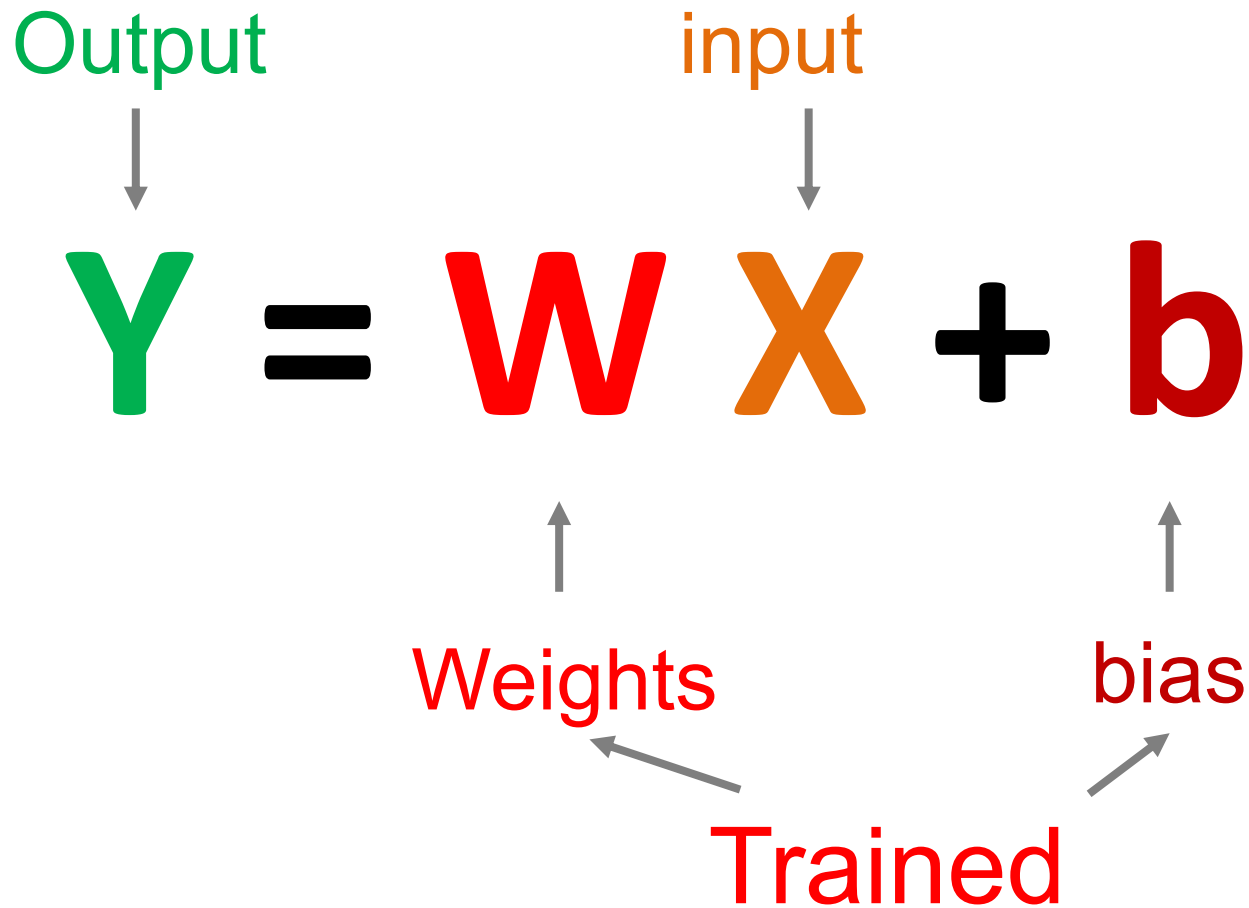
Output Layer
(Y)



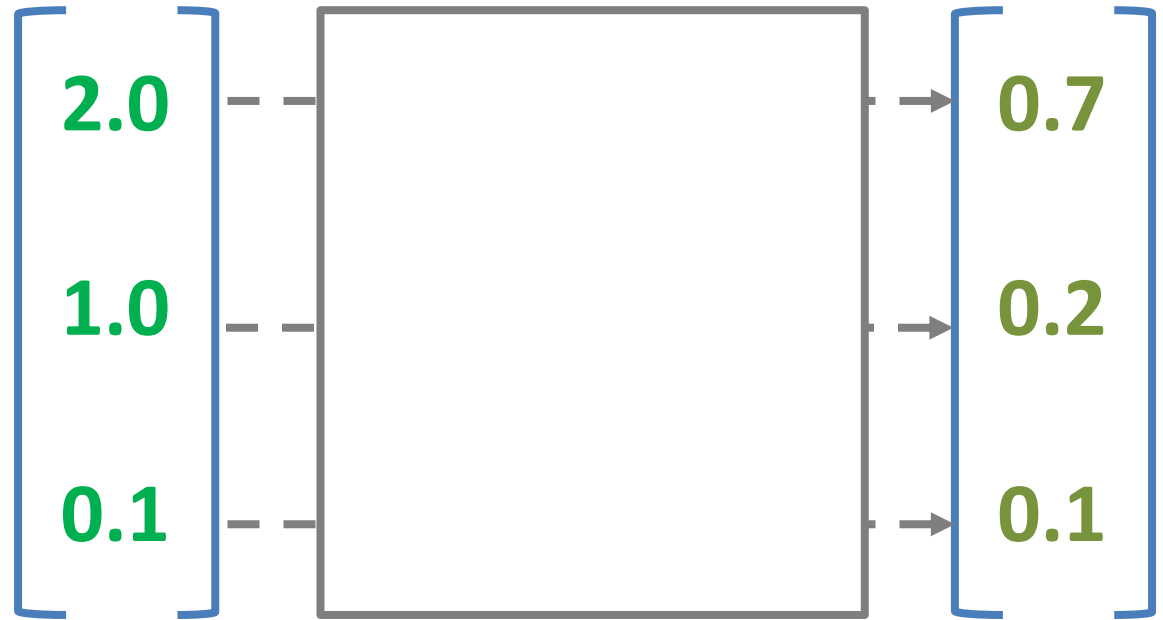
X		Y
Hours Sleep	Hours Study	Score
3	5	75
5	1	82
10	2	93
8	3	?

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

$$Y = WX + b$$



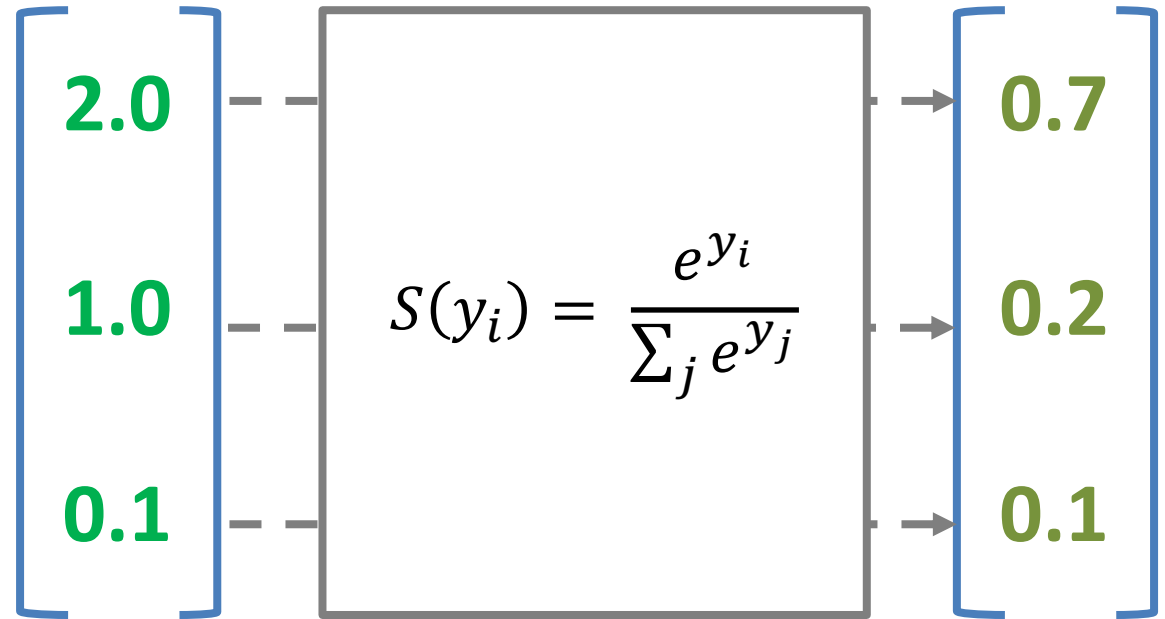
$$W X + b = Y$$



Scores \longrightarrow Probabilities

SoftMAX

$$W X + b = Y$$



Logits

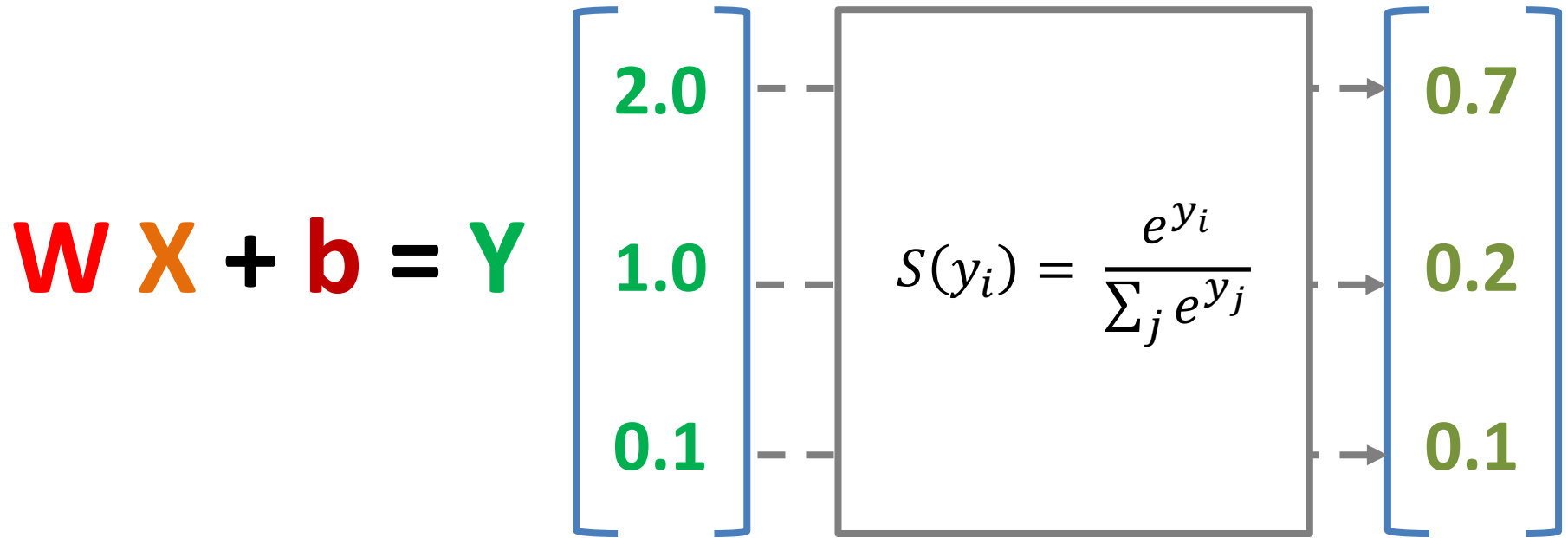
Scores

Probabilities

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

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

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



Logits

Scores

Probabilities

Training a Network
=
Minimize the Cost Function

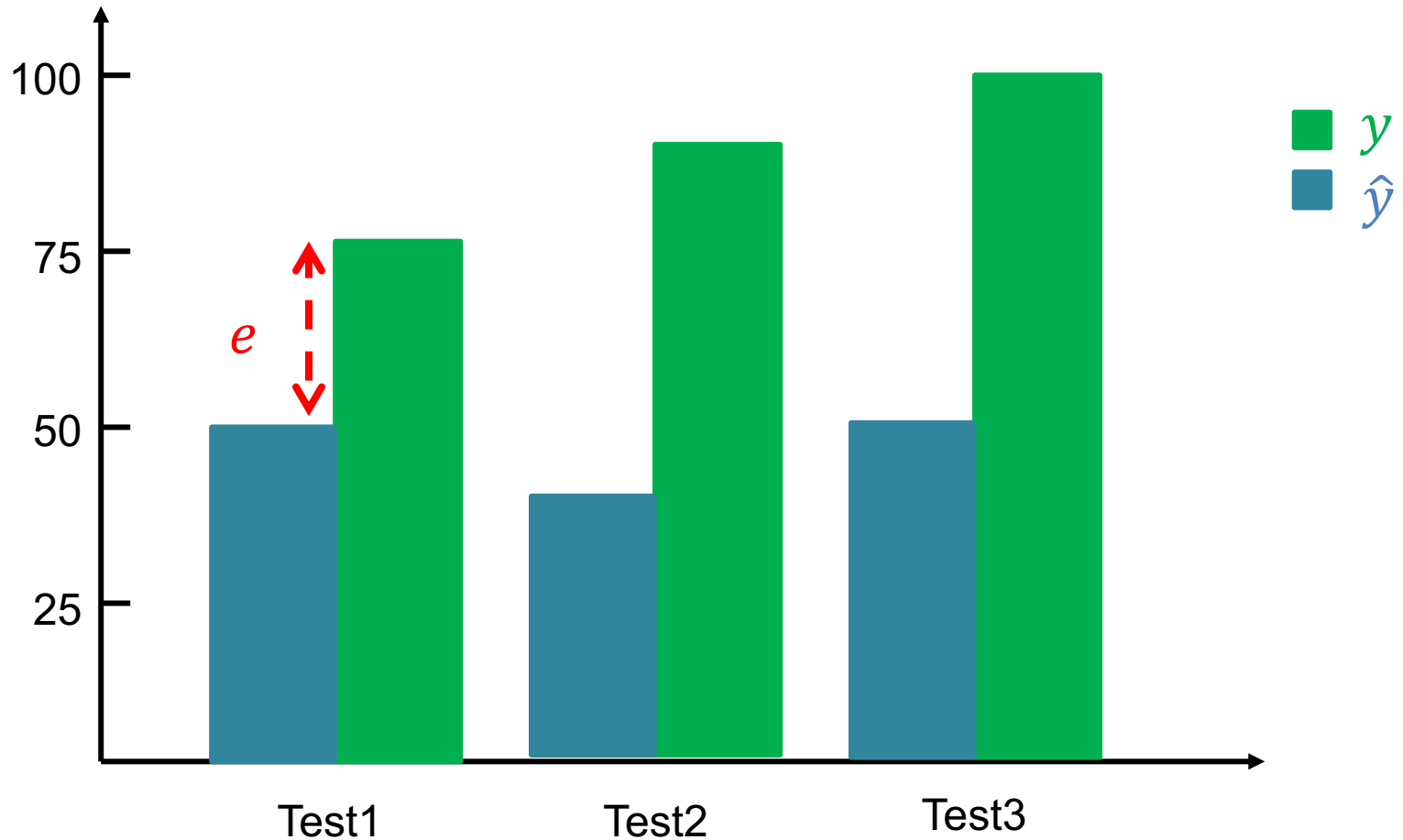
Training a Network

=

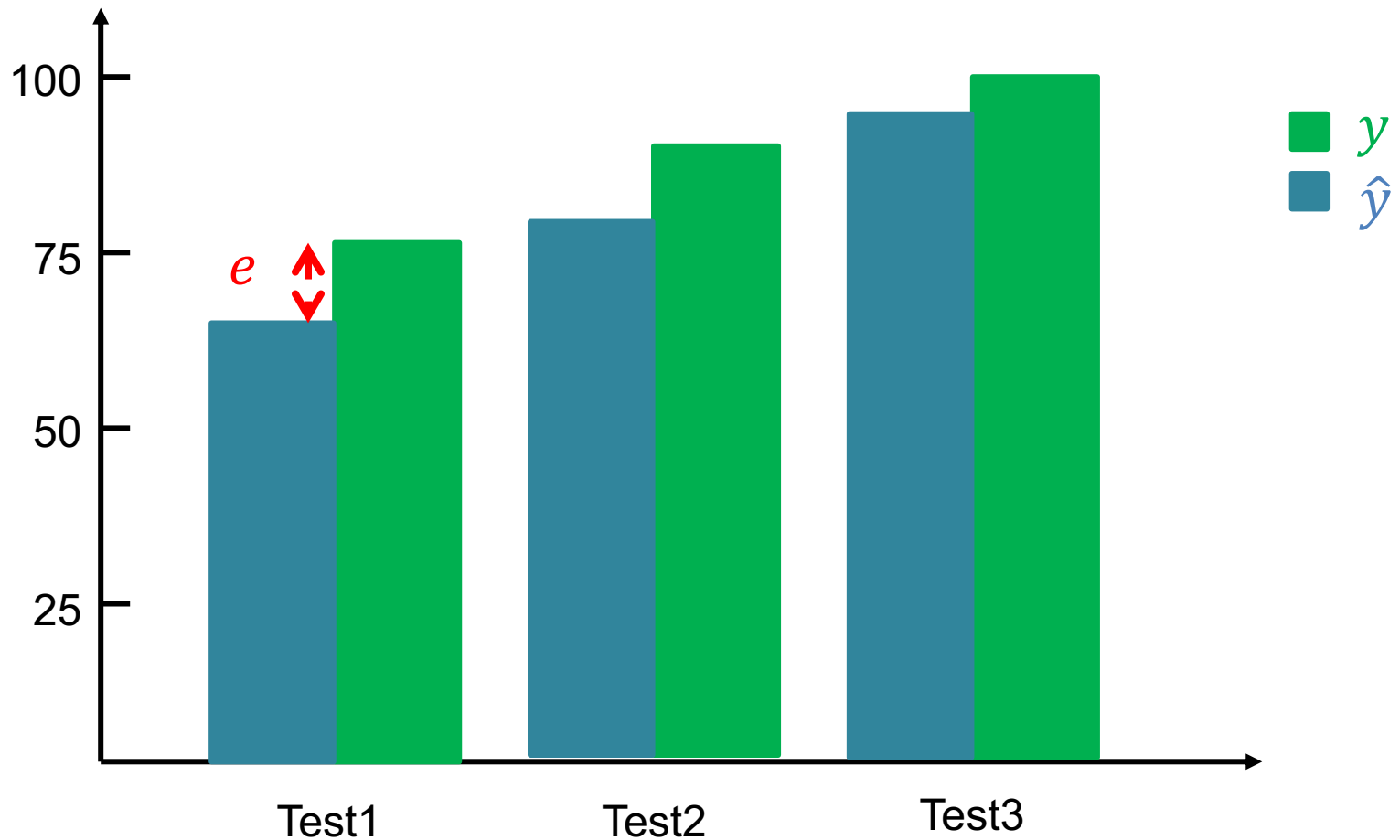
Minimize the **Cost** Function

Minimize the **Loss** Function

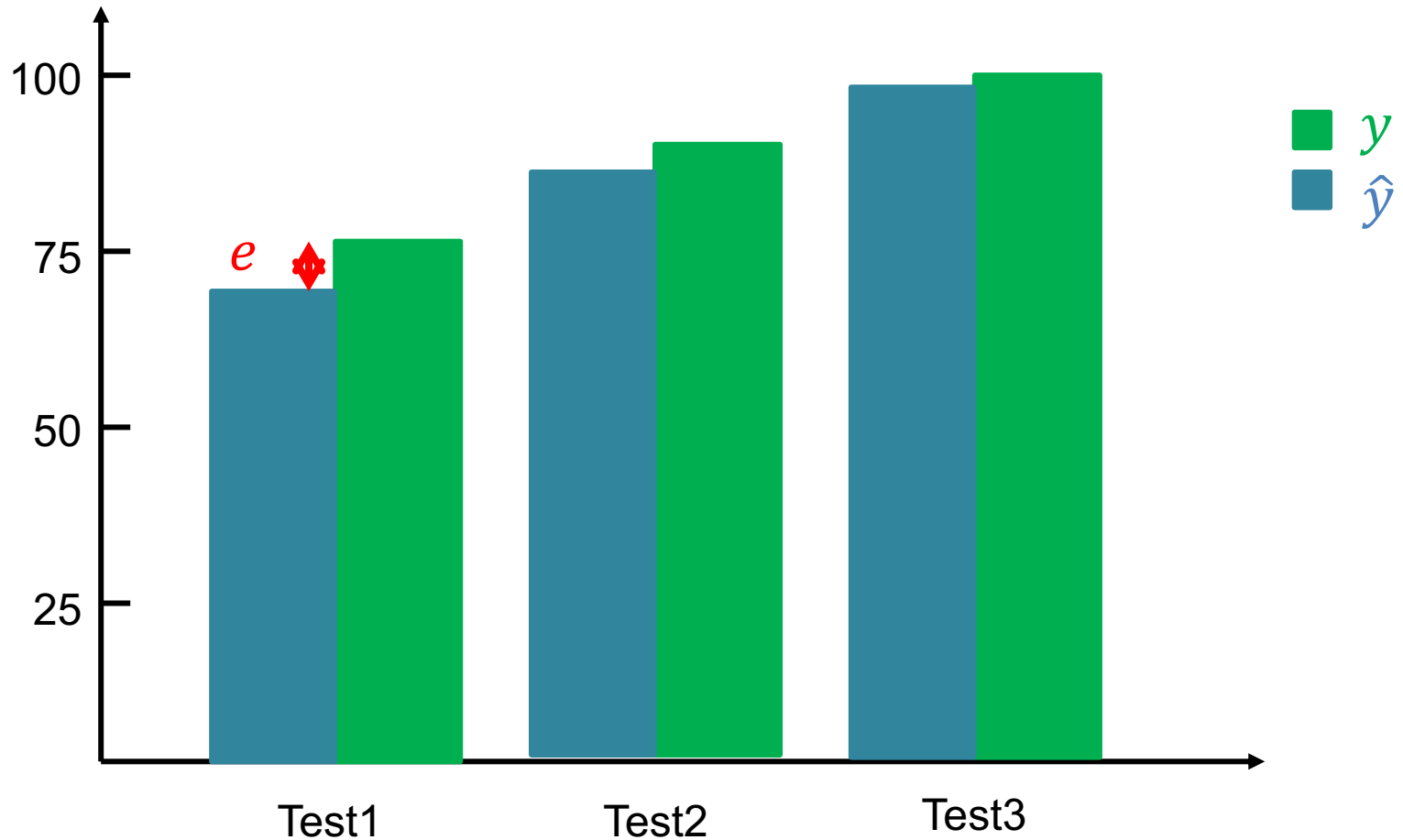
Error = Predict Y - Actual Y
Error : Cost : Loss



Error = Predict Y - Actual Y
Error : Cost : Loss



Error = Predict Y - Actual Y
Error : Cost : Loss



Activation Functions

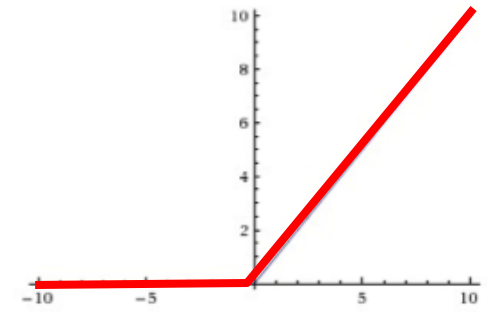
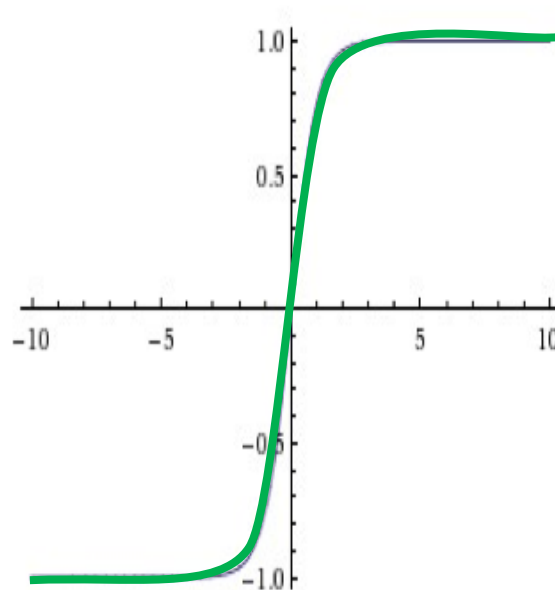
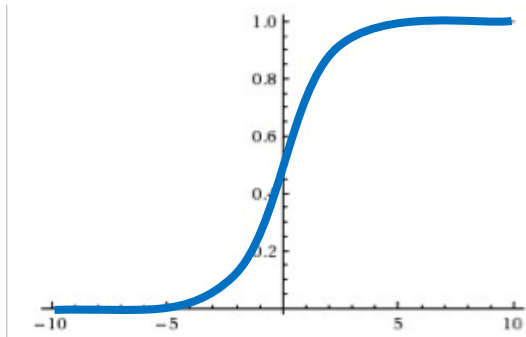
Activation Functions

Sigmoid

TanH

ReLU

(Rectified Linear Unit)



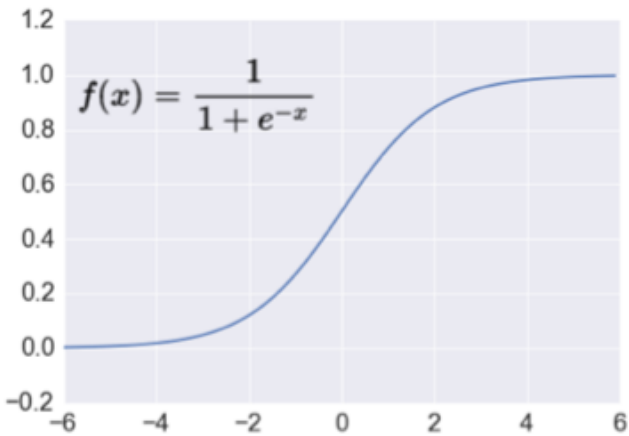
[0, 1]

[-1, 1]

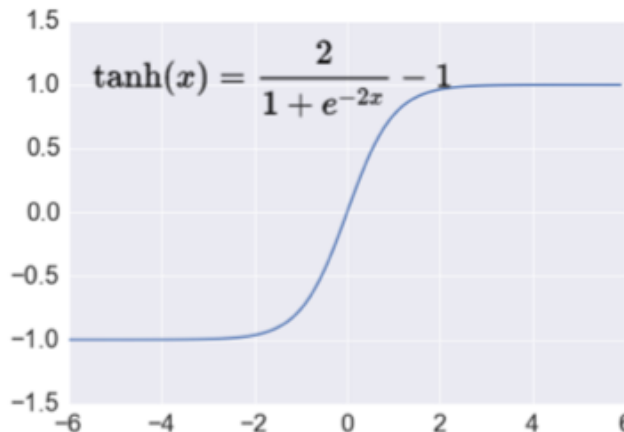
$f(x) = \max(0, x)$

Activation Functions

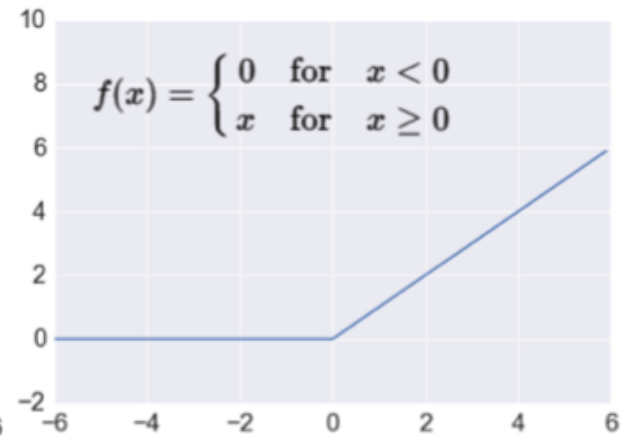
Sigmoid



TanH



ReLU



Loss Function

Binary Classification: 2 Class

**Activation Function:
Sigmoid**

**Loss Function:
Binary Cross-Entropy**

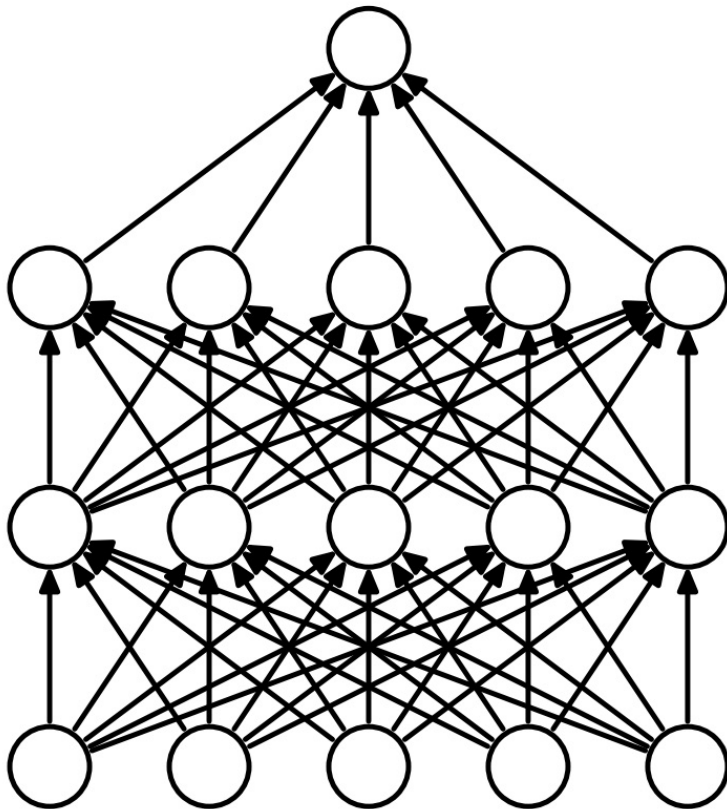
Multiple Classification: 10 Class

**Activation Function:
SoftMAX**

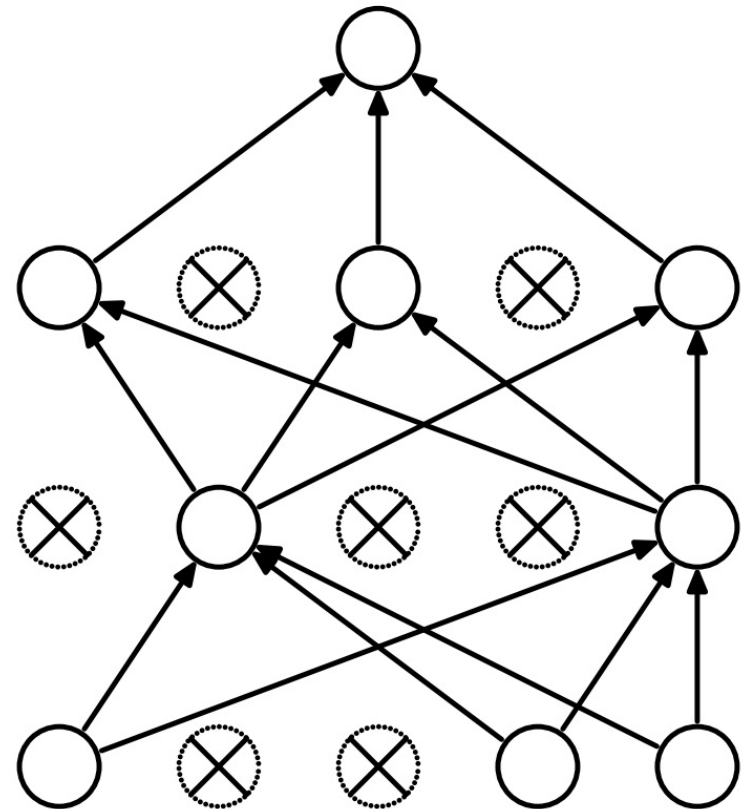
**Loss Function:
Categorical Cross-Entropy**

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:

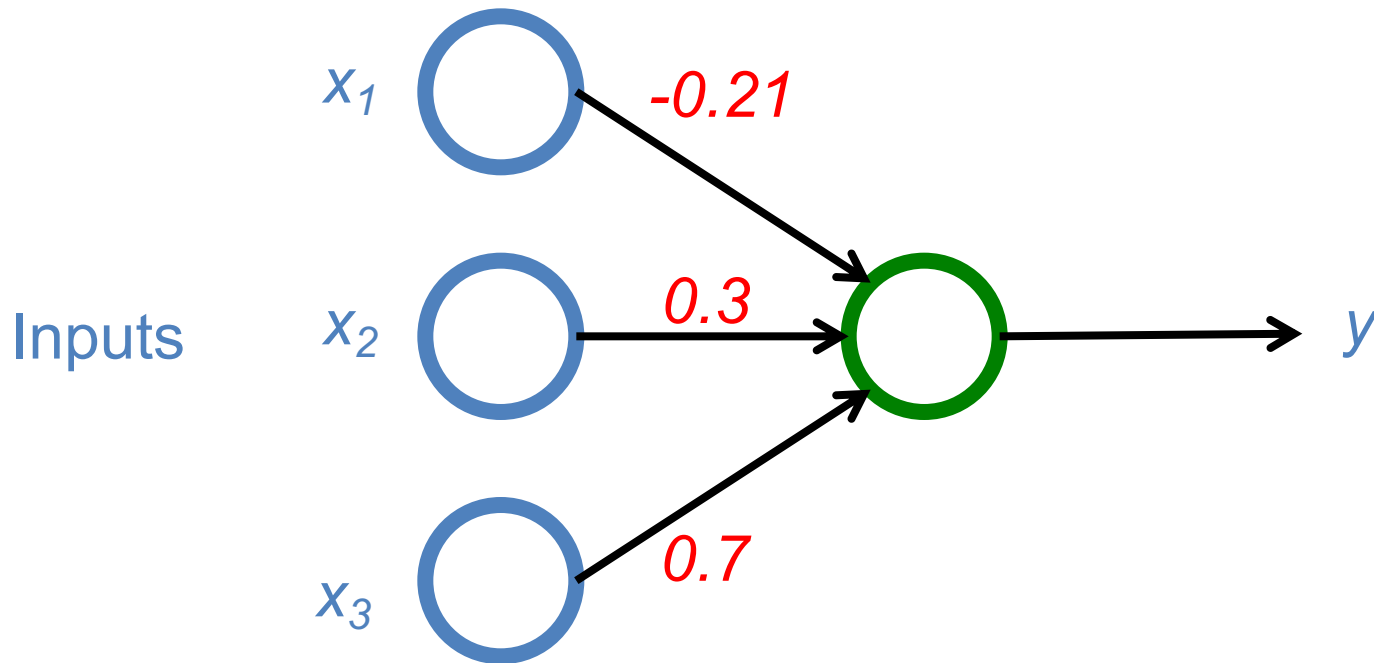
Pick a random training example “(input, label)”

Run neural network on “input”

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

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

Weights

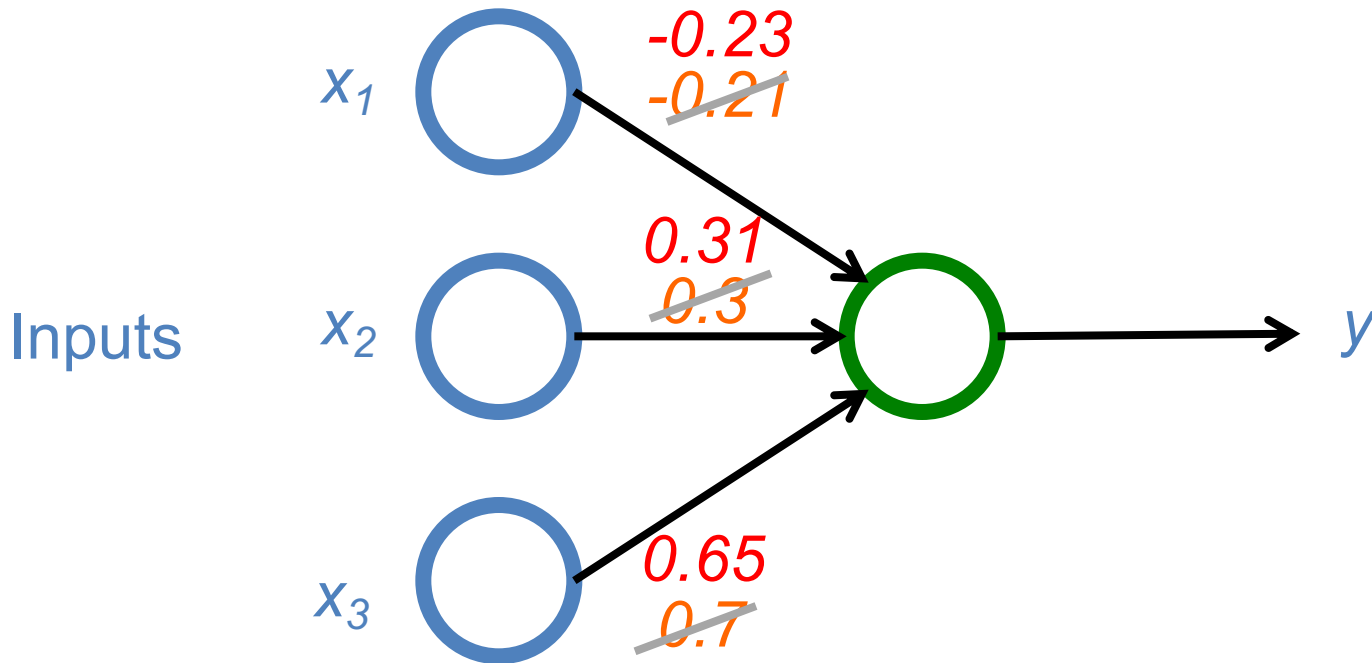


Next time:

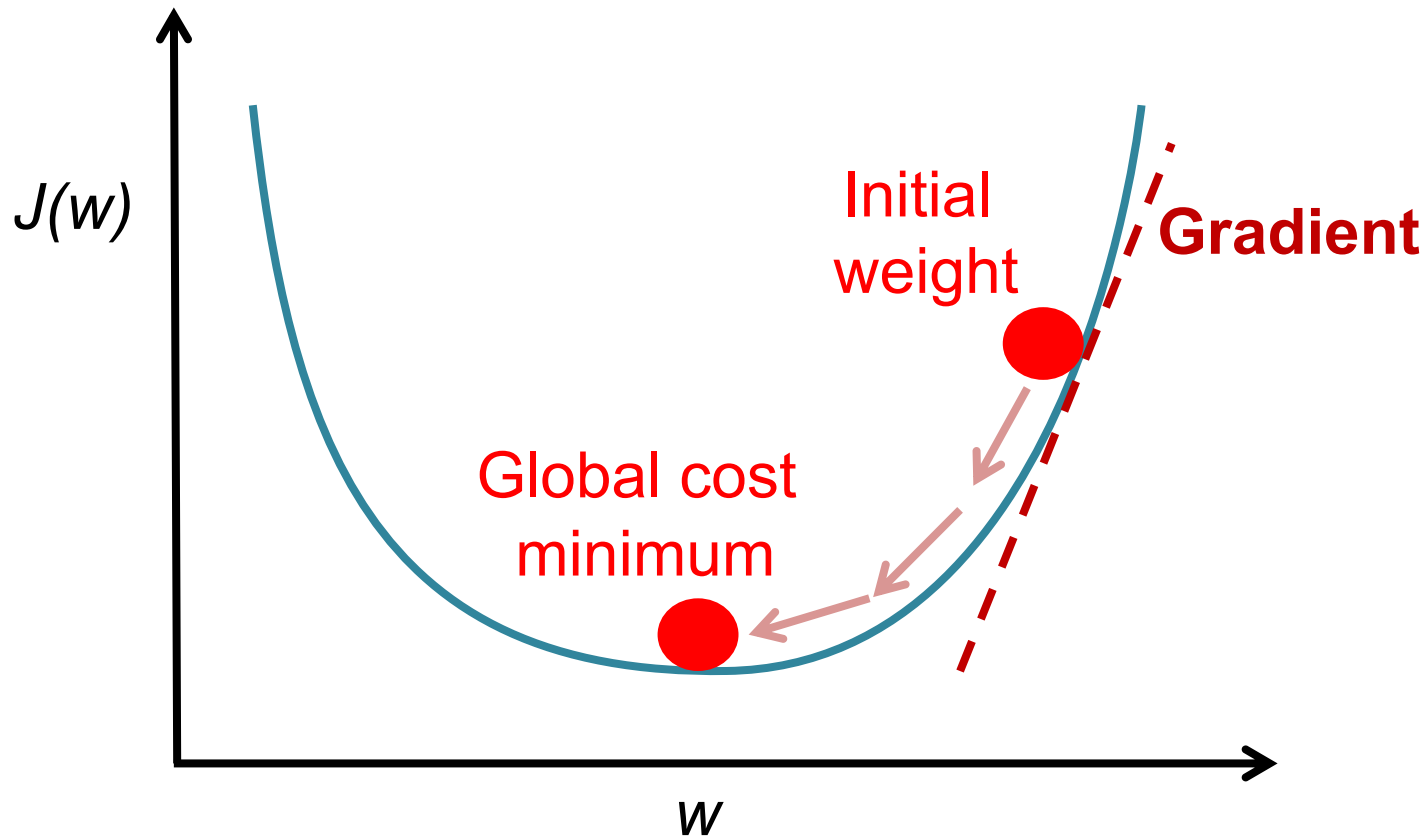
$$y = \max(0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3)$$

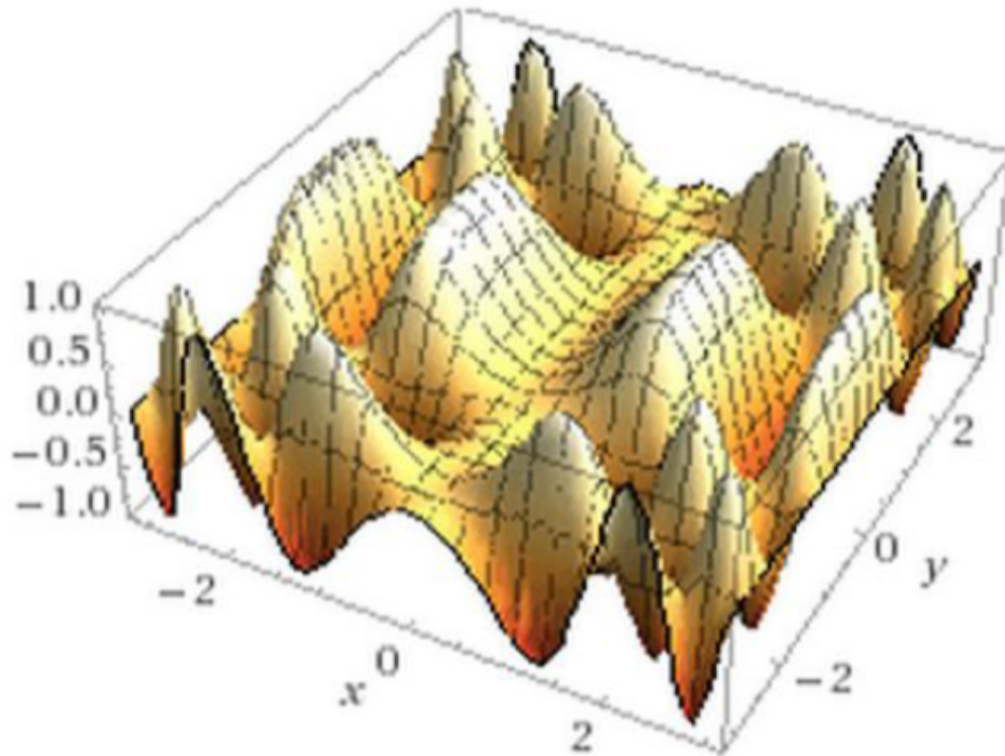
~~$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$~~

Weights



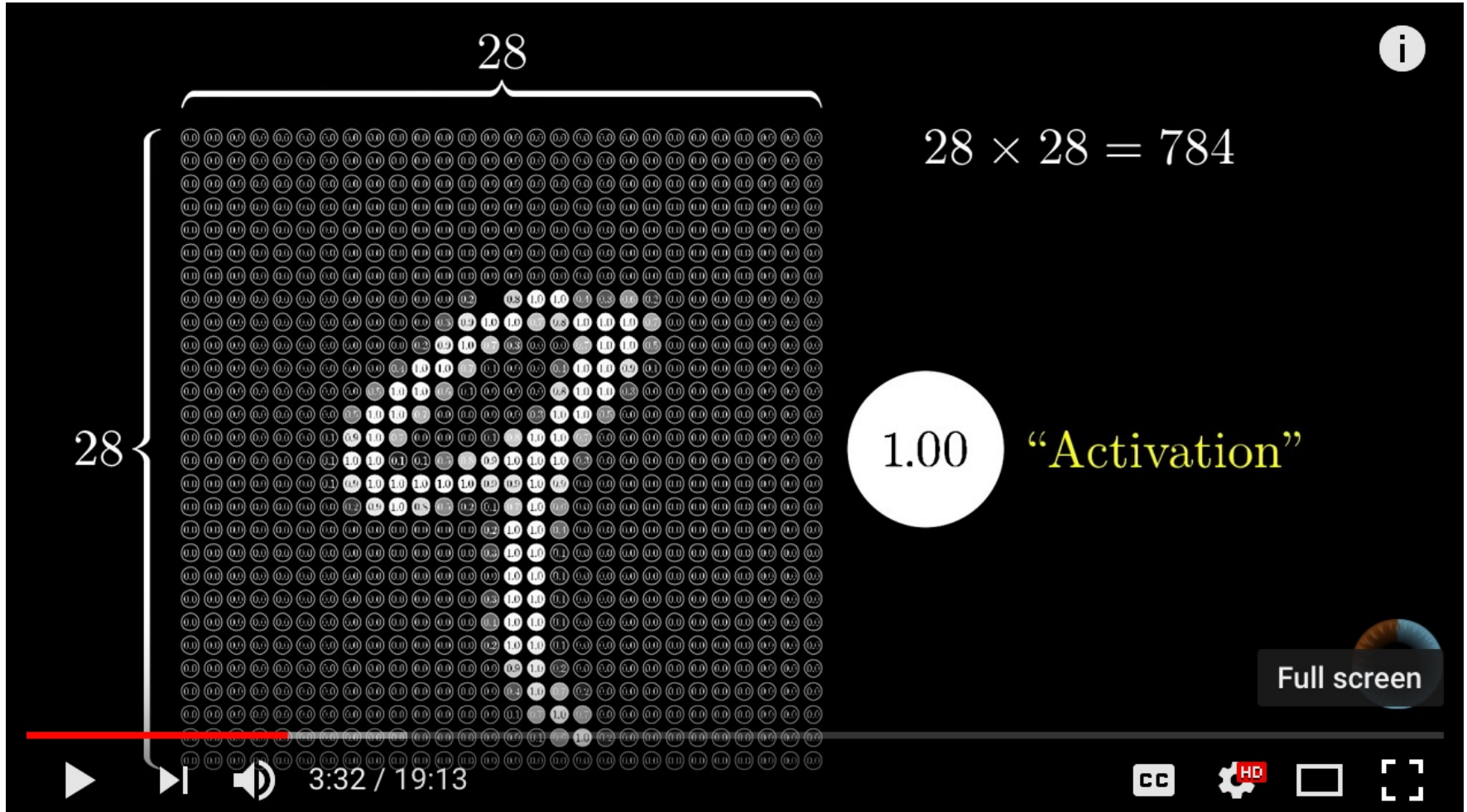
Optimizer: Stochastic Gradient Descent (SGD)





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

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

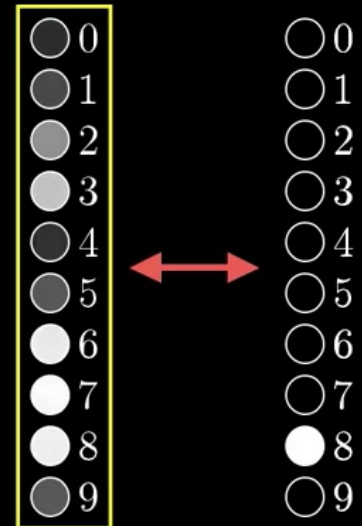
Average cost of
all training data...

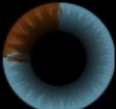
Cost of



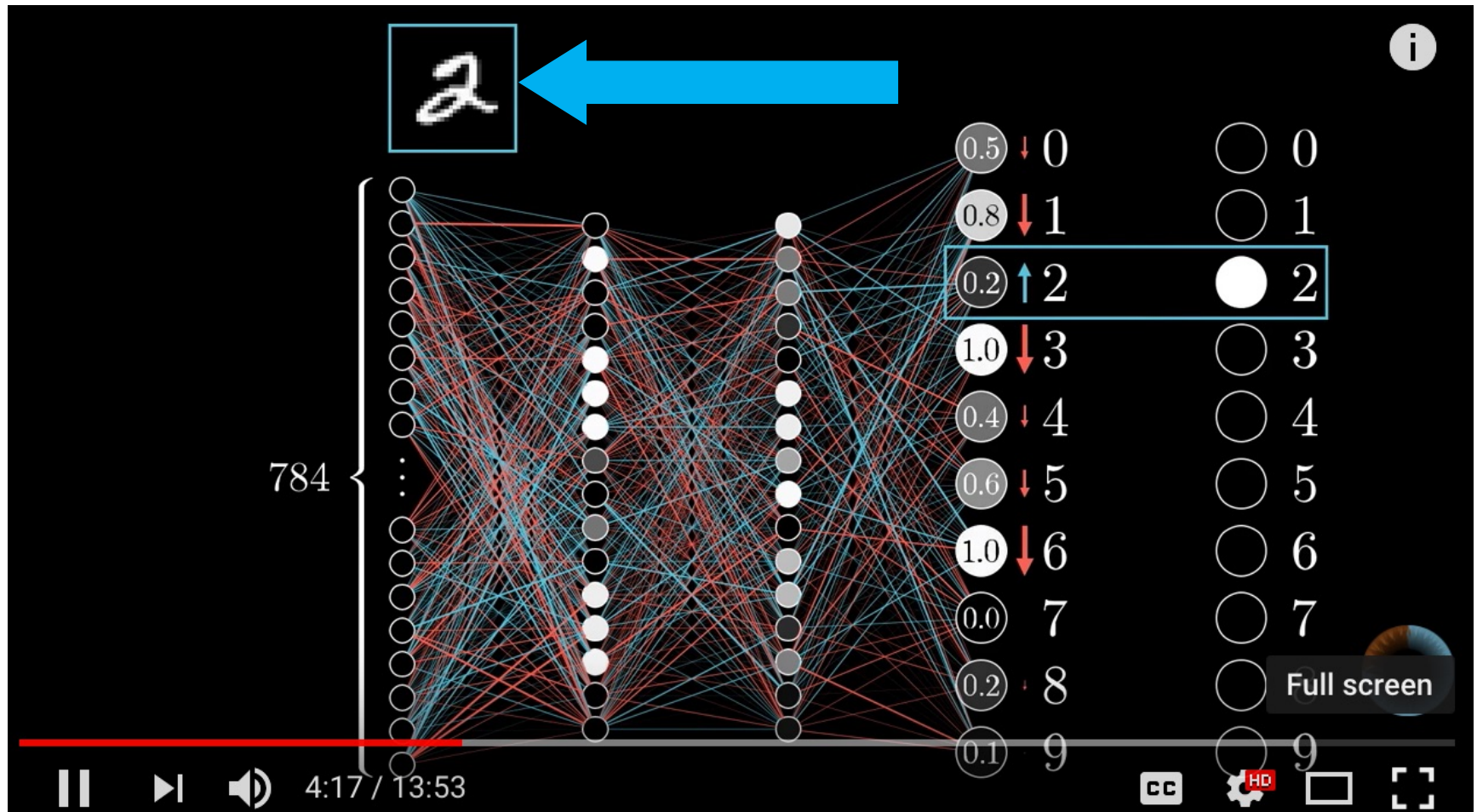
$$\left\{ \begin{array}{l} (0.18 - 0.00)^2 + \\ (0.29 - 0.00)^2 + \\ (0.58 - 0.00)^2 + \\ (0.77 - 0.00)^2 + \\ (0.20 - 0.00)^2 + \\ (0.36 - 0.00)^2 + \\ (0.93 - 0.00)^2 + \\ (1.00 - 0.00)^2 + \\ (0.95 - 1.00)^2 + \\ (0.35 - 0.00)^2 \end{array} \right.$$

What's the "cost" ⁱ
of this difference?



Utter trash 

Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=llg3gGewQ5U>

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”

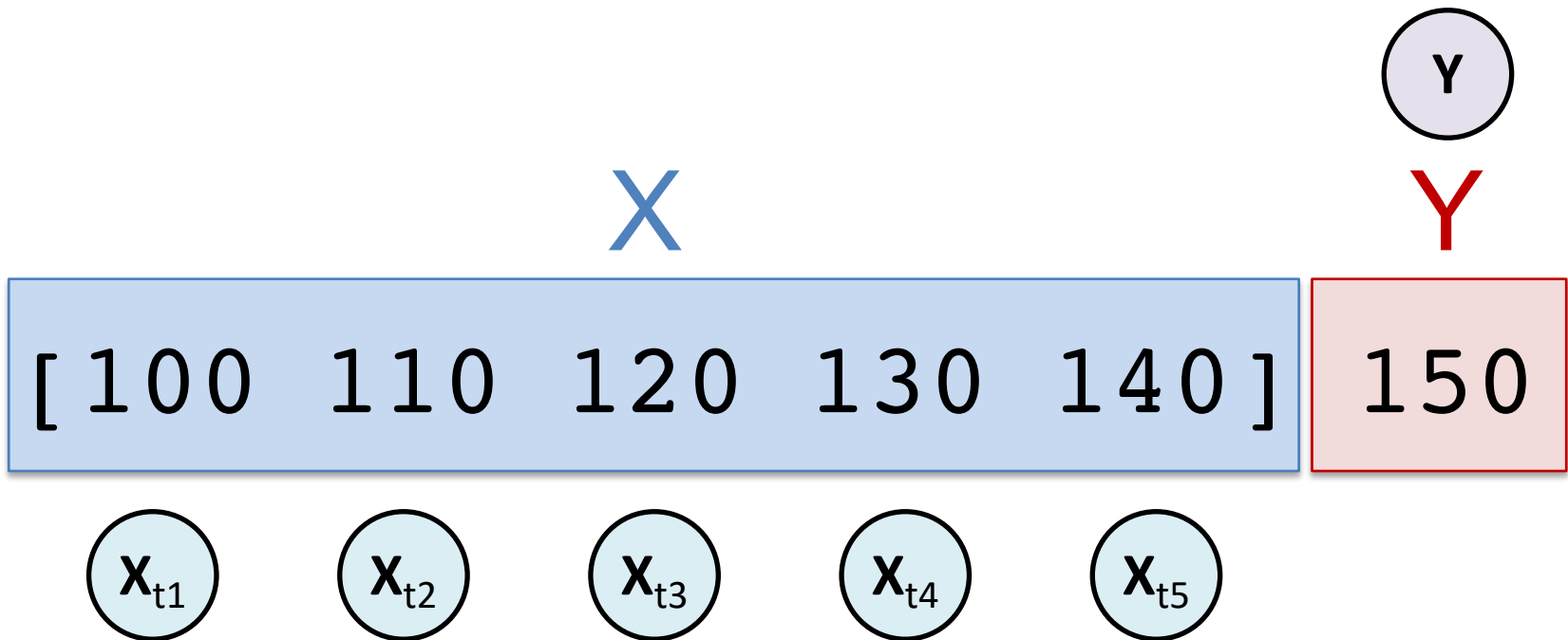
Financial Time Series Forecasting

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

tf.keras

Keras:

High-level API

for TensorFlow

Keras

K Keras Documentation

Home

- Keras: The Python Deep Learning library
- You have just found Keras.
- Guiding principles
- Getting started: 30 seconds to Keras
- Installation
- Configuring your Keras backend
- Support
- Why this name, Keras?

- Activations
- Applications
- Backend
- Callbacks
- Constraints
- Contributing
- Datasets
- Initializers
- Losses
- Metrics

GitHub Next »

Docs » Home

[Edit on GitHub](#)

Keras: The Python Deep Learning library



You have just found Keras.

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

Use Keras if you need a deep learning library that:

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

Read the documentation at [Keras.io](#).

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

<http://keras.io/>

PyTorch

[Get Started](#)[Features](#)[Ecosystem](#)[Blog](#)[Tutorials](#)[Docs](#)[Resources](#)[GitHub](#)

FROM RESEARCH TO PRODUCTION

An open source deep learning platform that provides a seamless path from research prototyping to production deployment.

[Get Started >](#)

KEY FEATURES & CAPABILITIES

[See all Features >](#)

<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.



TensorFlow

TensorFlow



Install

Learn ▾

API ▾

Resources ▾

More ▾

🔍 Search

Language ▾

GitHub

Sign in

Missed TensorFlow World? Check out the recap.

Learn more

An end-to-end open
source machine
learning platform

TensorFlow

For JavaScript

For Mobile & IoT

For Production

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

Get started with TensorFlow



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.

TensorFlow 2.0

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

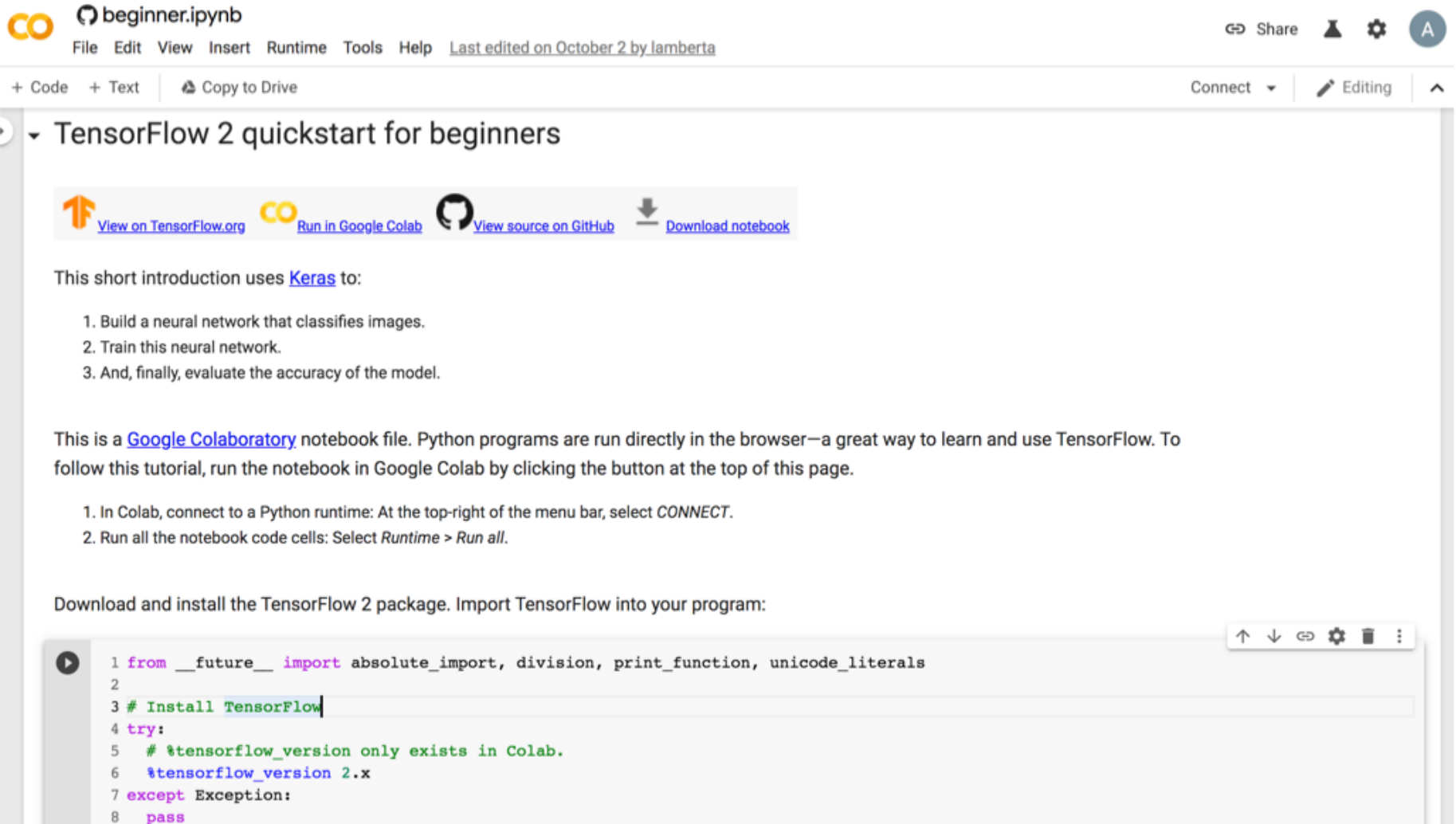
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

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)
```

TensorFlow 2 Quick Start



The screenshot shows a Google Colab notebook interface. At the top left, the notebook is titled "beginner.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", with a note "Last edited on October 2 by lamberta". On the right, there are icons for "Share", a warning icon, a settings gear, and a user profile icon labeled "A". Below the menu bar, there are options for "+ Code", "+ Text", and "Copy to Drive". The main content area has a title "TensorFlow 2 quickstart for beginners" and a toolbar with links: "View on TensorFlow.org", "Run in Google Colab", "View source on GitHub", and "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:

```
1 from __future__ import absolute_import, division, print_function, unicode_literals
2
3 # Install TensorFlow
4 try:
5     # %tensorflow_version only exists in Colab.
6     %tensorflow_version 2.x
7 except Exception:
8     pass
```


TensorFlow 2

Time Series Forecasting

TensorFlow

Install Learn API Resources More

Search Language GitHub Sign in

Overview **Tutorials** Guide TF 1

Quickstart for beginners
Quickstart for experts

BEGINNER

- ML basics with Keras
- Load and preprocess data
- Estimator

ADVANCED

- Customization
- Distributed training
- Images
- Text
- Structured data
 - Classify structured data with feature columns
 - Classification on imbalanced data
 - Time series forecasting**

TensorFlow > Learn > TensorFlow Core > Tutorials

Time series forecasting

☆☆☆☆☆

Run in Google Colab View source on GitHub Download notebook

This tutorial is an introduction to time series forecasting using Recurrent Neural Networks (RNNs). This is covered in two parts: first, you will forecast a univariate time series, then you will forecast a multivariate time series.

```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False
```

Contents

- The weather dataset
- Part 1: Forecast a univariate time series
 - Baseline
 - Recurrent neural network
- Part 2: Forecast a multivariate time series
 - Single step model
 - Multi-Step model
- Next steps

TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Iterations
000,582

Learning rate
0.03

Activation
Tanh

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



INPUT

Which properties do you want to feed in?

X_1



X_2



X_1^2



X_2^2



$X_1 X_2$



+ - 3 HIDDEN LAYERS

+ -
4 neurons



+ -
2 neurons



+ -
2 neurons

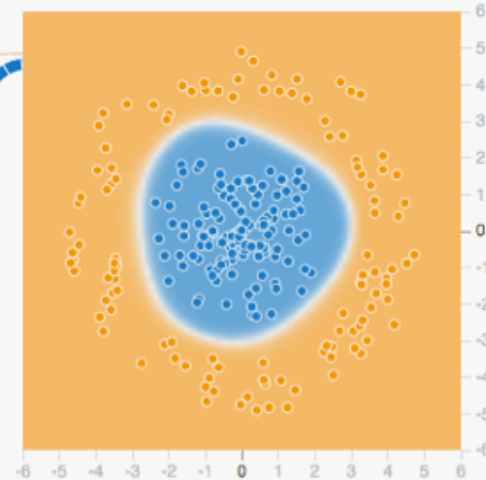


The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

OUTPUT

Test loss 0.000
Training loss 0.000





TensorFlow
is an
Open Source
Software Library
for
Machine Intelligence

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]

Scalar

80

Vector

[50 60 70]

Matrix

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

Tensor

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

TensorFlow

TensorBoard

TensorBoard

EVENTS

IMAGES

GRAPHS

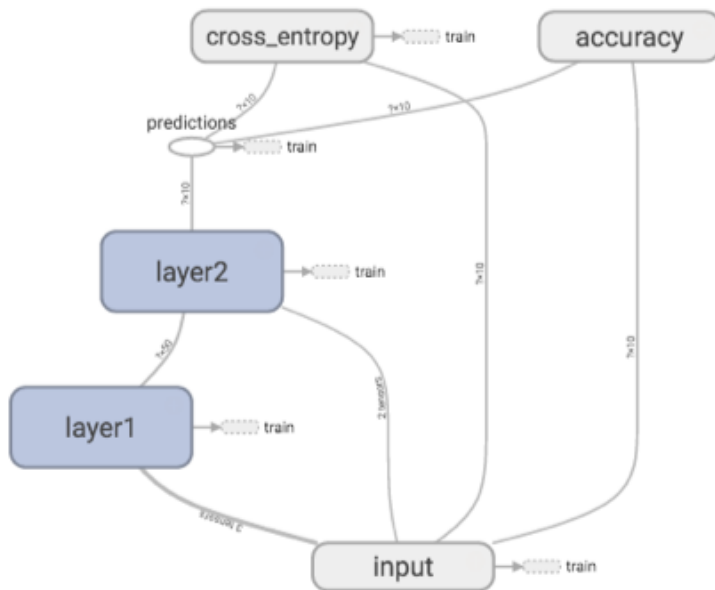
HISTOGRAMS



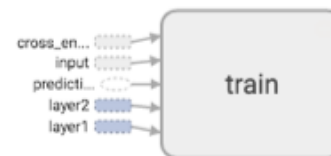
Fit to screen
 Download PNG
 Run train
 (1)
 Session runs (0)
 Upload
 Color Structure
 Device
 color: same substructure
 gray: unique substructure

Graph (* = expandable)
 Namespace*
 OpNode
 Unconnected series*
 Connected series*
 Constant
 Summary
 Dataflow edge
 Control dependency edge
 Reference edge

Main Graph



Auxiliary nodes



Deep Learning for Financial Time Series Forecasting

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

Time Series Data

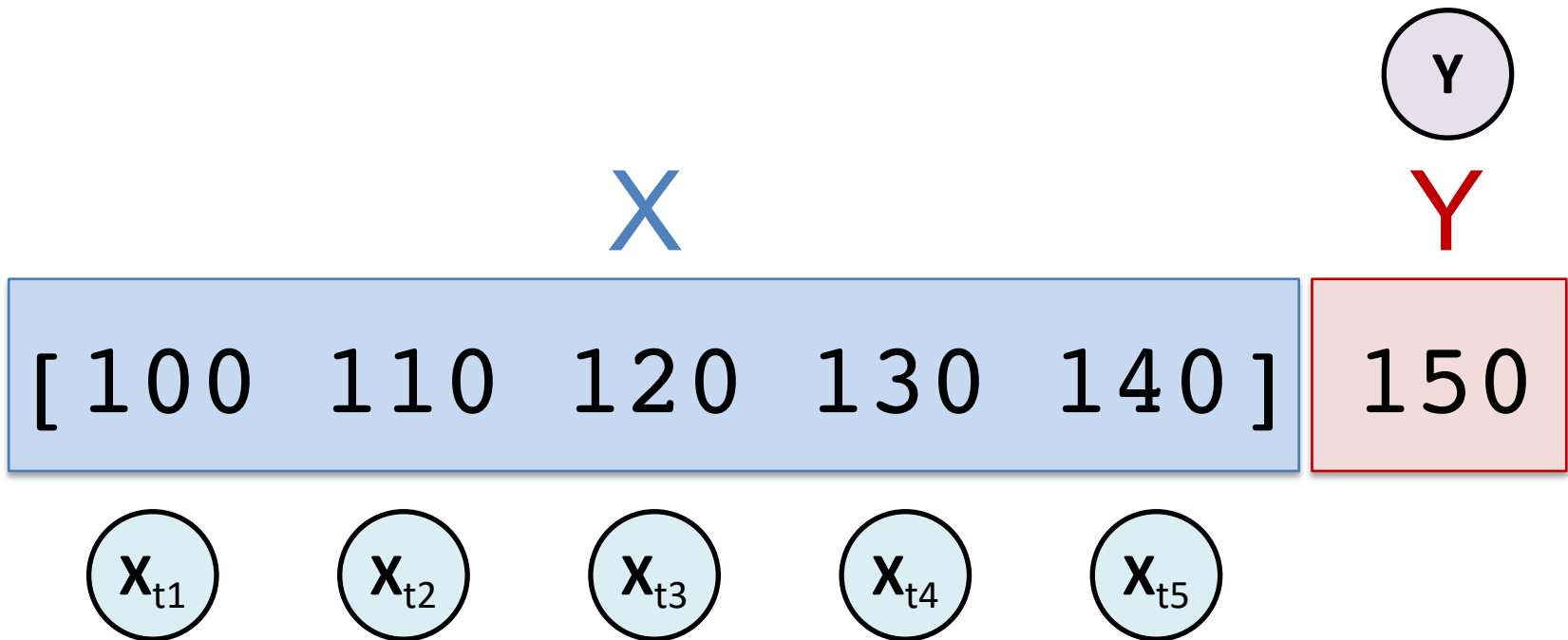
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

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

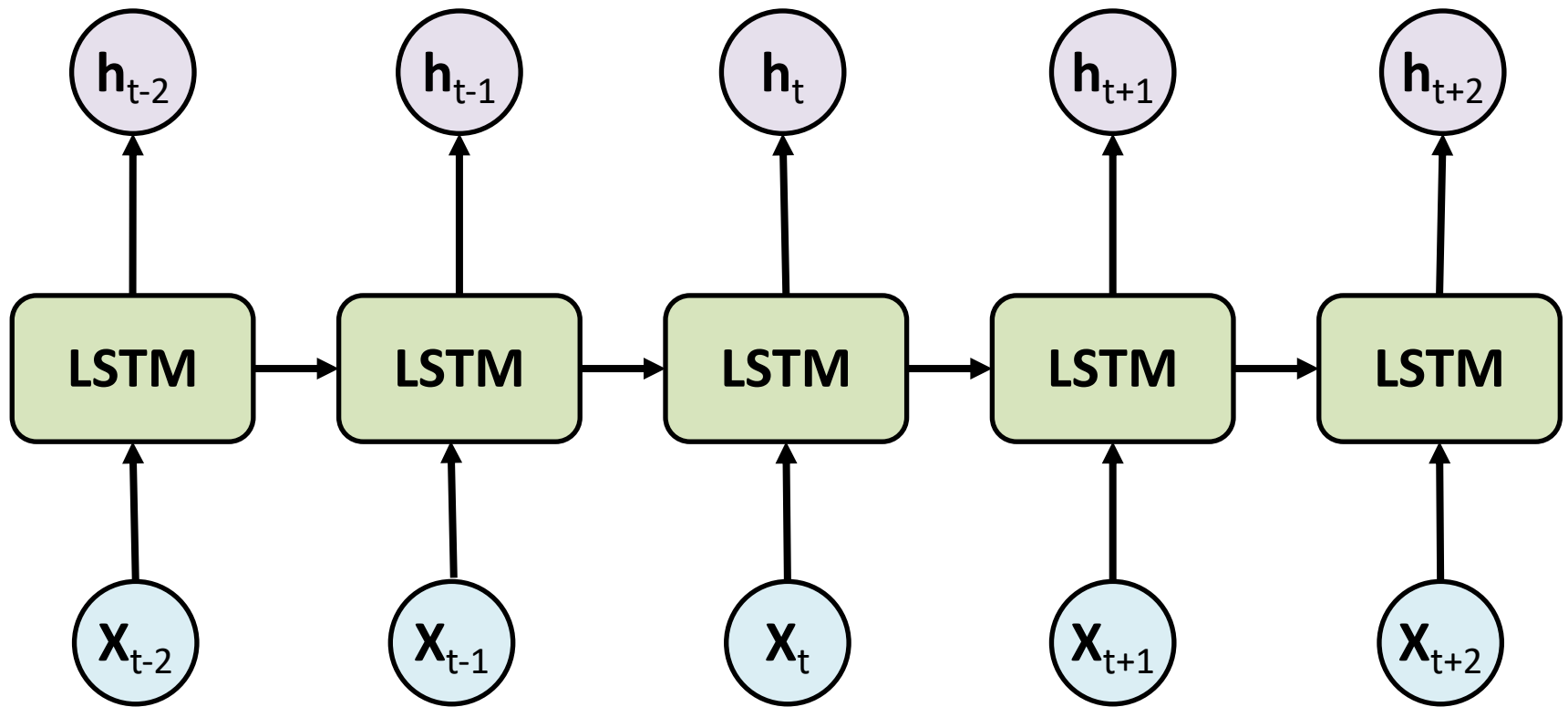


Time Series Data

[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]

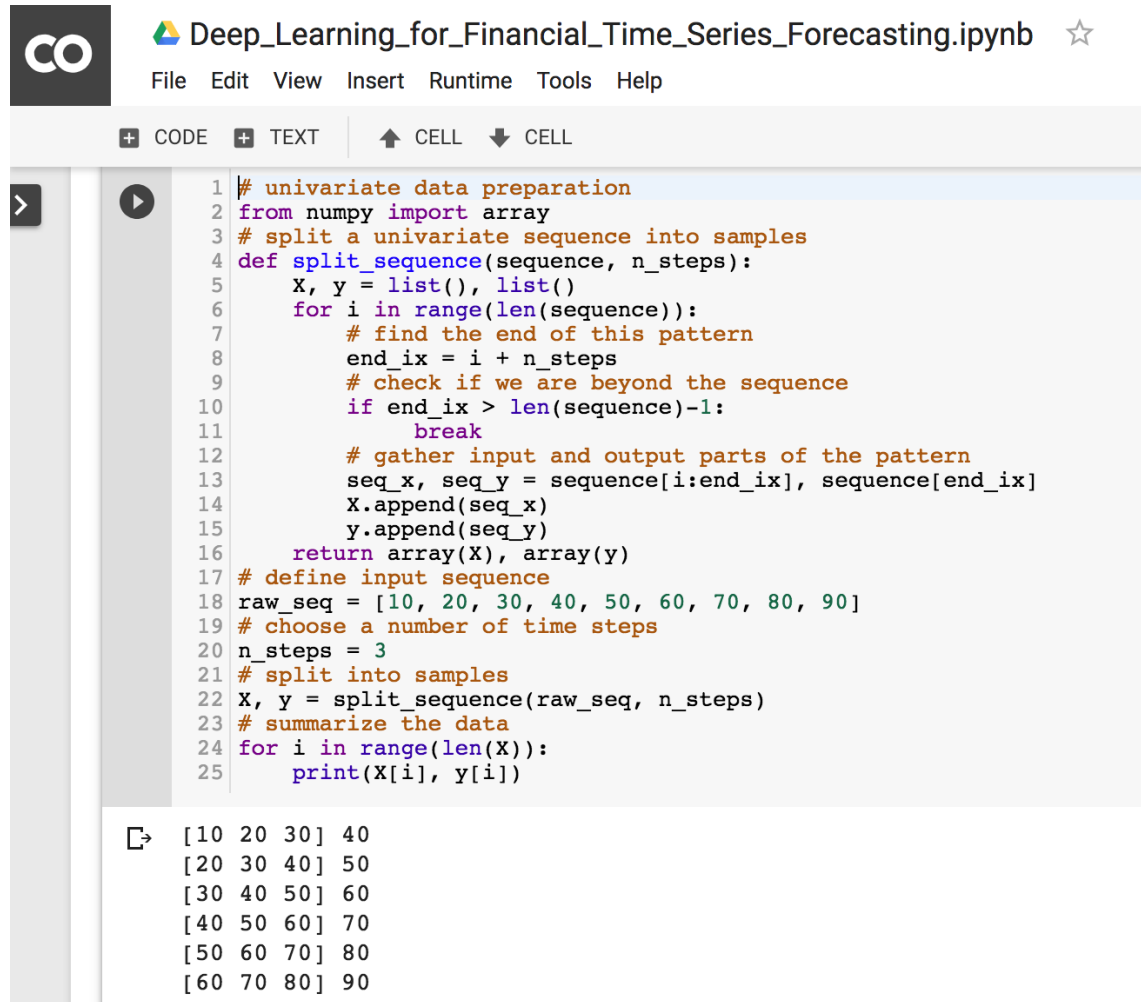
X

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>



The image shows a Google Colab notebook interface. At the top, there is a logo and the title "Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb" with a star icon. Below the title is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The main area is divided into tabs for "+ CODE", "+ TEXT", and "CELL". The code cell contains the following Python code:

```
1 # univariate data preparation
2 from numpy import array
3 # split a univariate sequence into samples
4 def split_sequence(sequence, n_steps):
5     X, y = list(), list()
6     for i in range(len(sequence)):
7         # find the end of this pattern
8         end_ix = i + n_steps
9         # check if we are beyond the sequence
10        if end_ix > len(sequence)-1:
11            break
12        # gather input and output parts of the pattern
13        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
14        X.append(seq_x)
15        y.append(seq_y)
16    return array(X), array(y)
17 # define input sequence
18 raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
19 # choose a number of time steps
20 n_steps = 3
21 # split into samples
22 X, y = split_sequence(raw_seq, n_steps)
23 # summarize the data
24 for i in range(len(X)):
25     print(X[i], y[i])
```

The output of the code is displayed below the code cell:

```
[ 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 ☆

COMMENT

SHARE

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

CONNECTED

EDITING

LSTM for Time Series Forecasting

```
1 # univariate lstm example
2 from numpy import array
3 from keras.models import Sequential
4 from keras.layers import LSTM
5 from keras.layers import Dense
6 import matplotlib.pyplot as plt
7 %matplotlib inline
8
9 # define dataset
10 x = array([[100, 110, 120], [110, 120, 130], [120, 130, 140], [130, 140, 150], [140, 150, 160]])
11 y = array([130, 140, 150, 160, 170])
12 # reshape from [samples, timesteps] into [samples, timesteps, features]
13 x = x.reshape((x.shape[0], x.shape[1], 1))
14 # define model
15 model = Sequential()
16 model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
17 model.add(Dense(1))
18 model.compile(optimizer='adam', loss='mse')
19 # fit model
20 history = model.fit(X, y, epochs=2000, verbose=0)
21 # demonstrate prediction
22 x_input = array([150, 160, 170])
23 x_input = x_input.reshape((1, 3, 1))
24 yhat = model.predict(x_input, verbose=0)
25 print('yhat', yhat)
26 print(model.summary())
27 # list all data in history
28 print(history.history.keys())
29 # summarize history for loss
30 print('loss:', '%f'%history.history['loss'][-1])
31 print('loss:', history.history['loss'][-1])
32 plt.plot(history.history['loss'])
33 plt.title('model loss')
34 plt.ylabel('loss')
35 plt.xlabel('epoch')
36 plt.show()
```

yhat [[181.34615]]

Deep Learning for Financial Time Series Forecasting

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



Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb ☆

COMMENT

SHARE

A

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

CONNECTED

EDITING

```
1 # univariate lstm example
2 from numpy import array
3 from keras.models import Sequential
4 from keras.layers import LSTM
5 from keras.layers import Dense
6 import matplotlib.pyplot as plt
7 %matplotlib inline
8 # split a univariate sequence into samples
9 def split_sequence(sequence, n_steps):
10     X, y = list(), list()
11     for i in range(len(sequence)):
12         # find the end of this pattern
13         end_ix = i + n_steps
14         # check if we are beyond the sequence
15         if end_ix > len(sequence)-1:
16             break
17         # gather input and output parts of the pattern
18         seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
19         X.append(seq_x)
20         y.append(seq_y)
21     return array(X), array(y)
22 # define input sequence
23 raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
24 # choose a number of time steps
25 n_steps = 3
26 # split into samples
27 X, y = split_sequence(raw_seq, n_steps)
28 # reshape from [samples, timesteps] into [samples, timesteps, features]
29 n_features = 1
30 X = X.reshape((X.shape[0], X.shape[1], n_features))
31 # define model
32 model = Sequential()
33 model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
34 model.add(Dense(1))
35 model.compile(optimizer='adam', loss='mse')
36 # fit model
37 history = model.fit(X, y, epochs=500, verbose=0)
38 # demonstrate prediction
39 x_input = array([70, 80, 90])
40 x_input = x_input.reshape((1, n_steps, n_features))
41 yhat = model.predict(x_input, verbose=0)
42 print(yhat)
43 print('yhat', yhat)
44 print(model.summary())
```


Deep Learning for Financial Time Series Forecasting

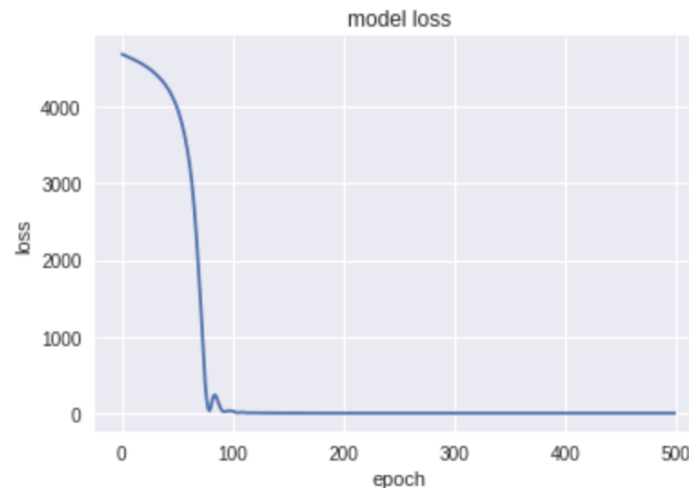
<https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM>

```
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

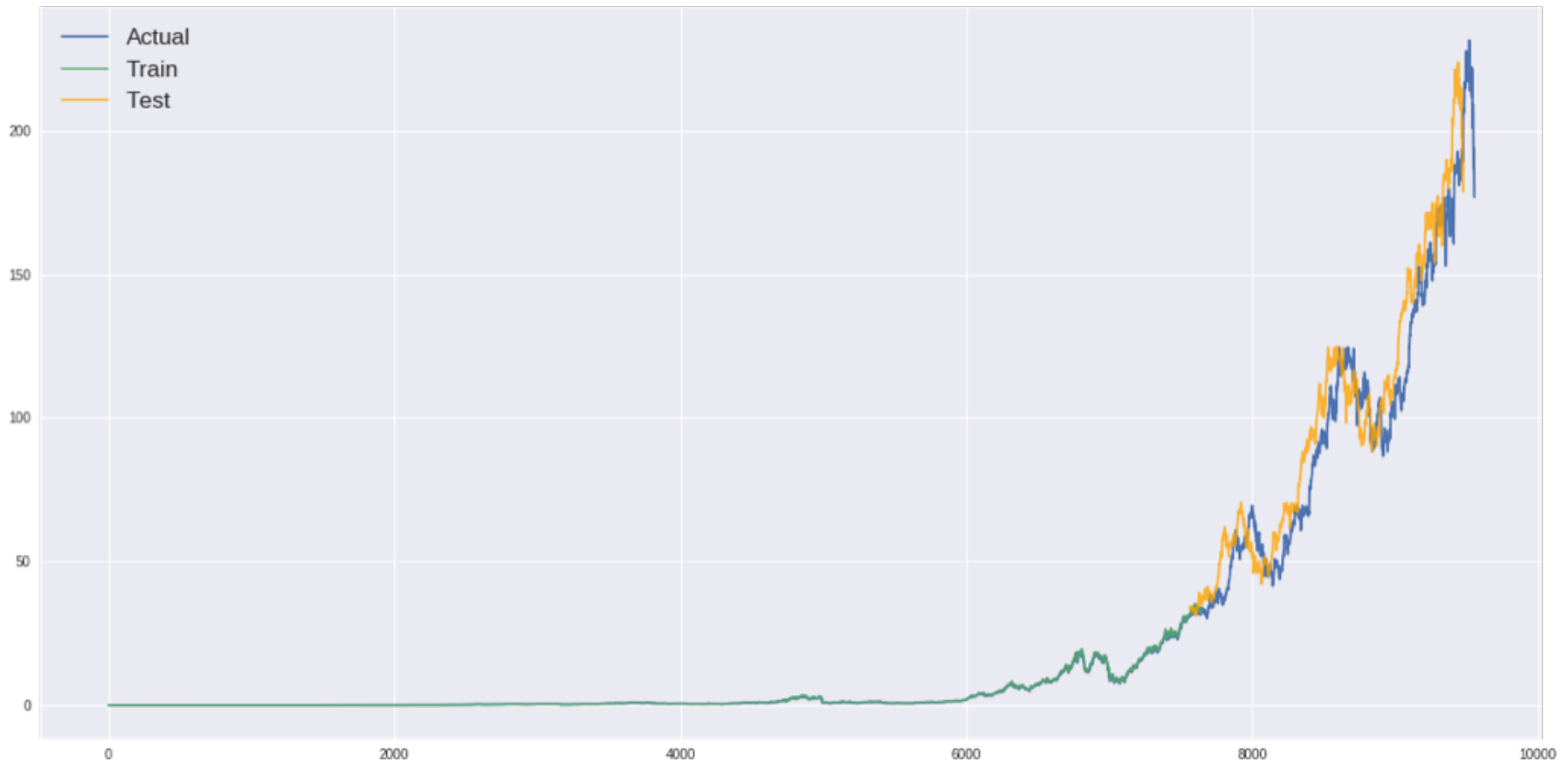
```
Total params: 10,451  
Trainable params: 10,451  
Non-trainable params: 0
```

```
None  
dict_keys(['loss'])  
loss: 0.000000  
loss: 1.2578432517784677e-07
```



Deep Learning for Financial Time Series Forecasting

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



Basic Classification

Fashion MNIST Image Classification

<https://colab.research.google.com/drive/19PJOJi1vn1kjcultzNHjRSLbeVI4kd5z>

tf01_basic_classification.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

Table of contents Code snippets Files

Copyright 2018 The TensorFlow Authors.

Licensed under the Apache License, Version 2.0 (the "License");

MIT License

Train your first neural network: basic classification

Import the Fashion MNIST dataset

Explore the data

Preprocess the data

Build the model

Setup the layers

Compile the model

Train the model

Evaluate accuracy

Make predictions

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.

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

Text Classification

IMDB Movie Reviews

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

The screenshot shows a Google Colab notebook titled "tf02_basic-text-classification.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. On the right, there are "COMMENT" and "SHARE" buttons. Below the navigation bar, there are buttons for "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". A "CONNECT" dropdown and an "EDITING" button are also visible.

The left sidebar contains a "Table of contents" with the following items:

- Copyright 2018 The TensorFlow Authors.
- Licensed under the Apache License, Version 2.0 (the "License");
- MIT License
- Text classification with movie reviews**
- Download the IMDB dataset
- Explore the data
 - Convert the integers back to words
- Prepare the data
- Build the model
 - Hidden units
 - Loss function and optimizer
- Create a validation set
- Train the model
- Evaluate the model

The main content area shows a copyright notice for 2018 The TensorFlow Authors, followed by a section titled "Text classification with movie reviews". This section includes three links: "View on TensorFlow.org", "Run in Google Colab", and "View source on GitHub".

The text below the links reads: "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](#)."

Below the text is a code cell with the following code:

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     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_AURjR6SCVBRdj

tf03_basic-regression.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

Table of contents Code snippets Files X

▶ 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.

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
14     print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
15     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 (Python101)

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



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

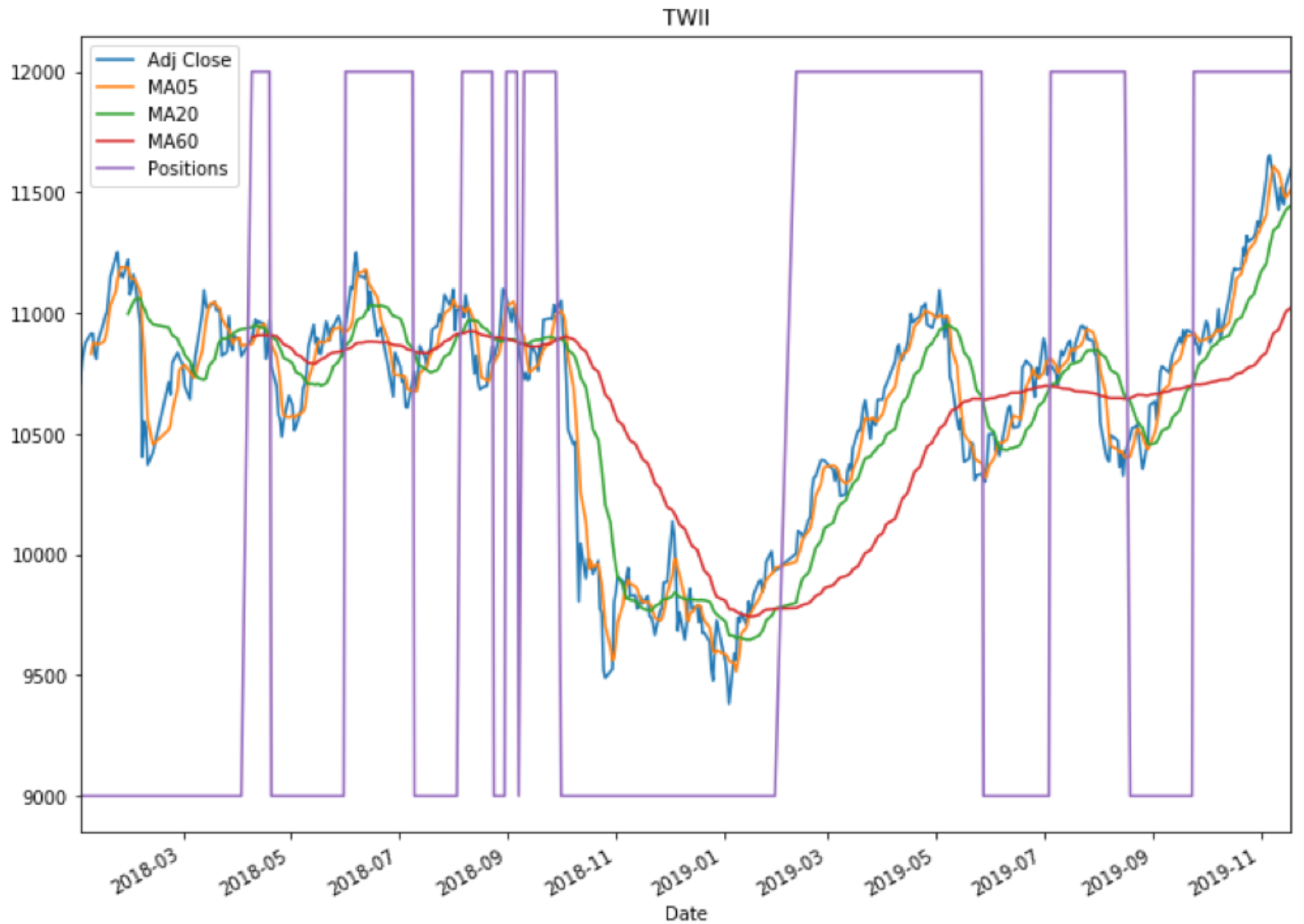
Comment Share Settings

+ Code + Text

RAM Disk Editing

```
1 import pandas as pd
2 import pandas_datareader.data as web
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import datetime as dt
6 %matplotlib inline
7
8 #Read Stock Data from Yahoo Finance
9 end = dt.datetime.now()
10 #start = dt.datetime(end.year-2, end.month, end.day)
11 start = dt.datetime(2017, 1, 1)
12 df = web.DataReader("AAPL", 'yahoo', start, end)
13 df.to_csv('AAPL.csv')
14 print(df.tail())
15 df2 = pd.read_csv('AAPL.csv') #df.from_csv('AAPL.csv')
16 print(df2.tail())
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30 plt.figure(figsize=(12,9))
31 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
32
33 # simple moving averages
34 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
35 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
36 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
37 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
38 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
39 df2.to_csv('AAPL_MA.csv')
40 fig = plt.gcf()
41 fig.set_size_inches(12, 9)
42 fig.savefig('AAPL_plot.png', dpi=300)
```

<https://tinyurl.com/imtkupython101>



`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
```

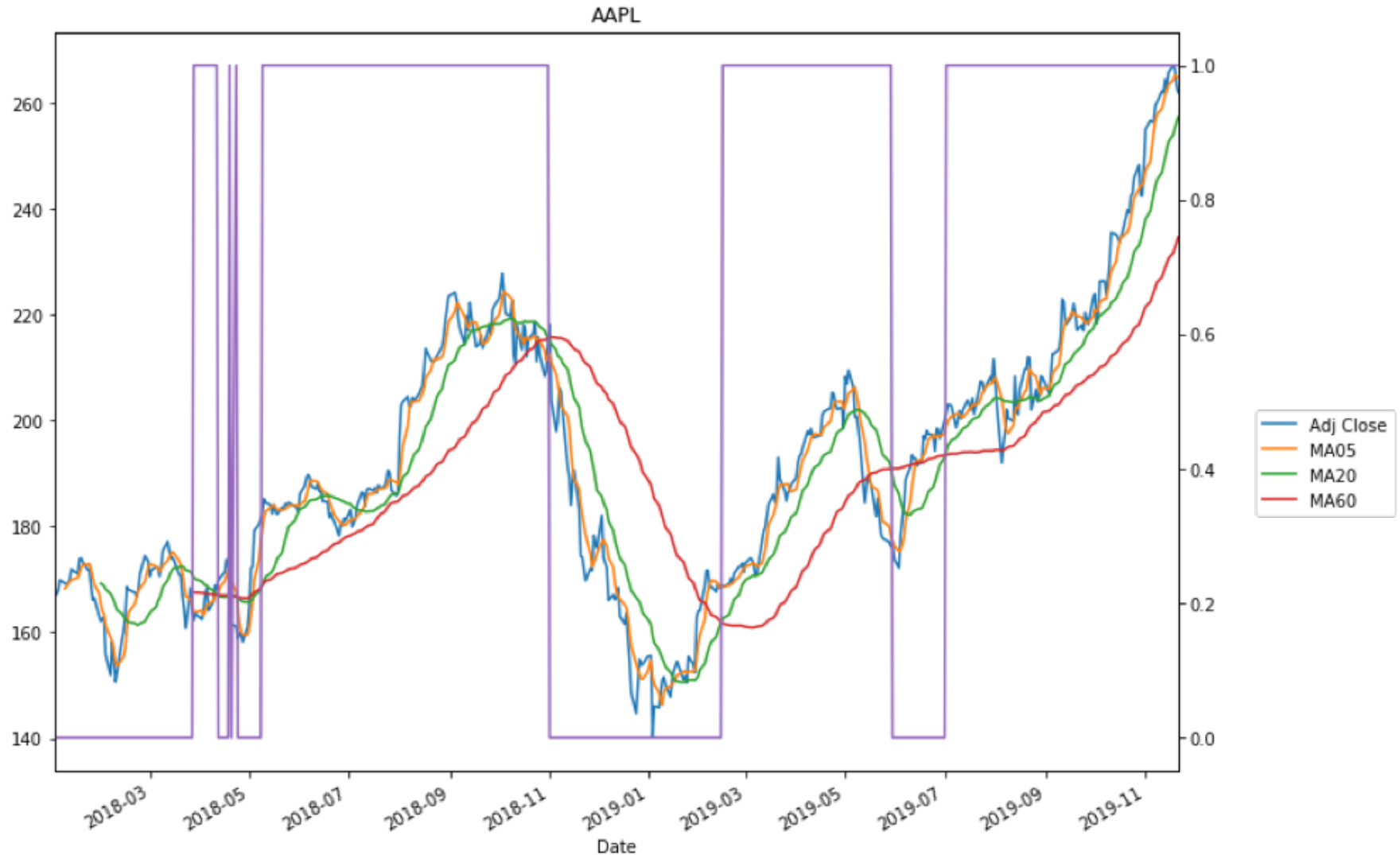
```
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
```

```
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']})
```



```
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
```

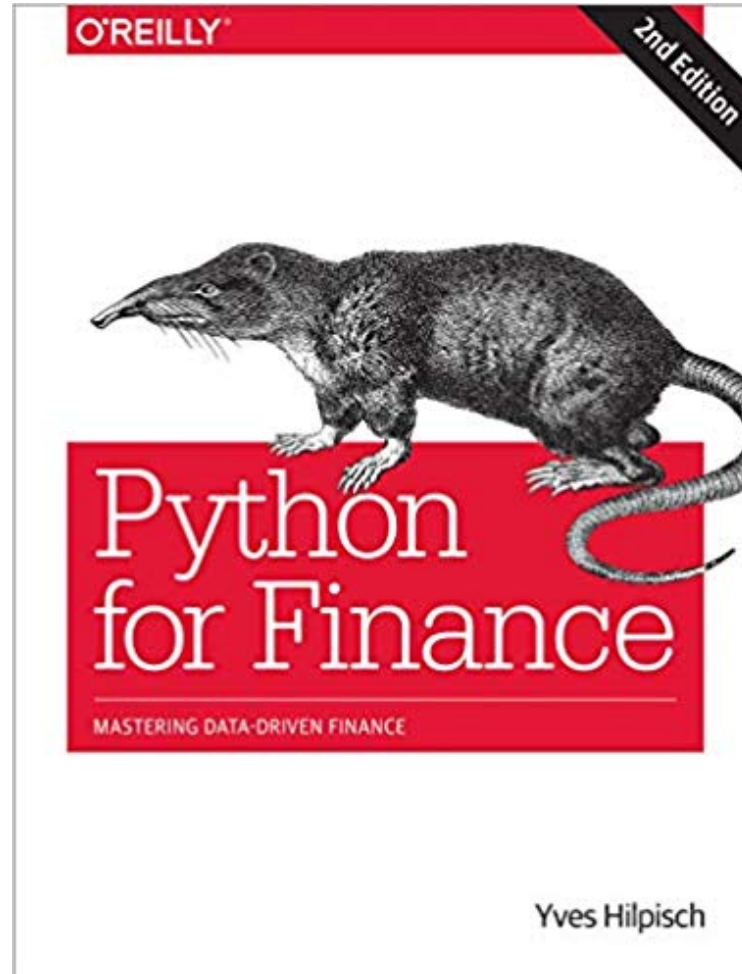
```
df['Positions'] = np.where(df['MA20'] > df['MA60'], 1, 0)
```

```
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'],  
'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Positions']})
```

Yves Hilpisch (2018),

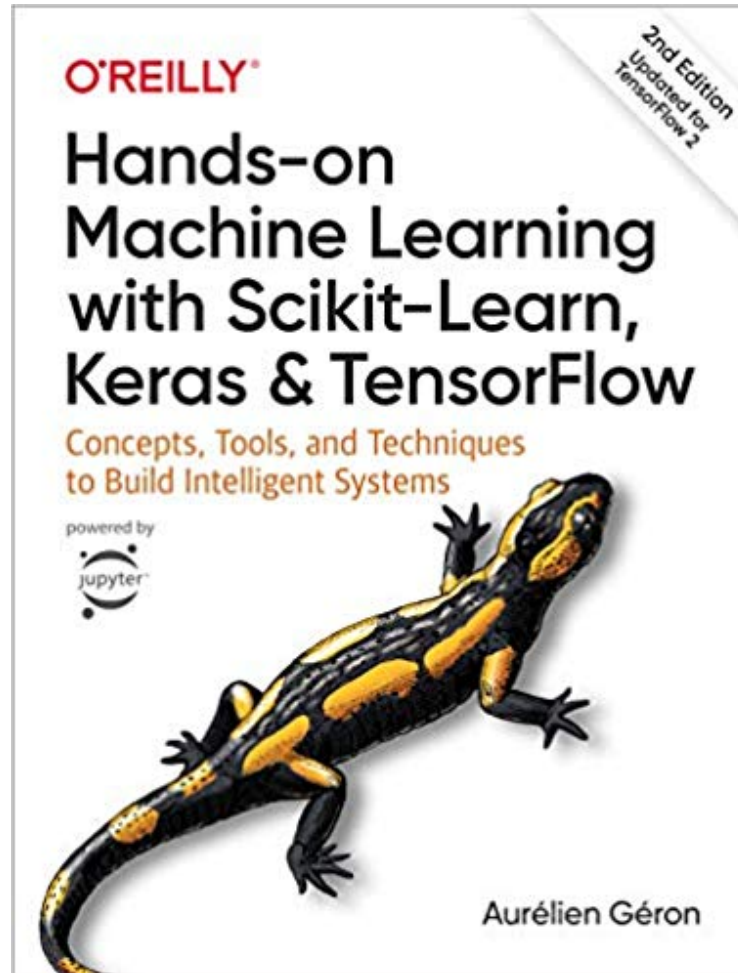
Python for Finance: Mastering Data-Driven Finance,

O'Reilly



<https://github.com/yhilpisch/py4fi2nd>

**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**



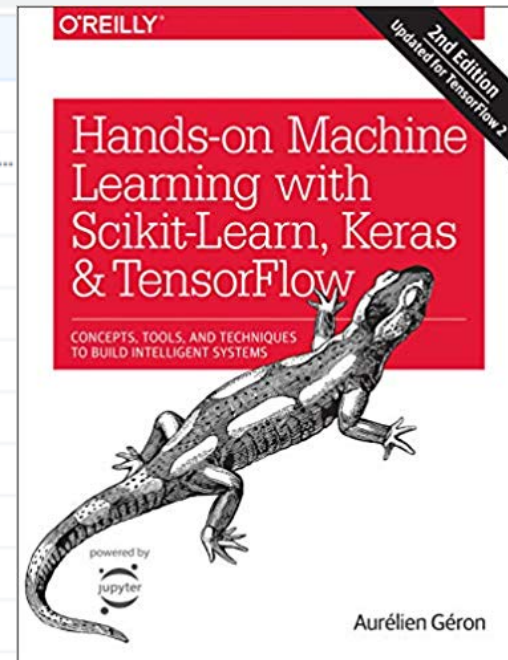
<https://github.com/ageron/handson-ml2>

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_landsc...	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
08_dimensionality_reduction.ipynb	Make notebooks 1 to 9 runnable in Colab without changes
09_unsupervised_learning.ipynb	Make notebooks 1 to 9 runnable in Colab without changes
10_neural_nets_with_keras.ipynb	Make notebooks 10 and 11 runnable in Colab without changes
11_training_deep_neural_networks....	Make notebooks 10 and 11 runnable in Colab without changes
12_custom_models_and_training_...	loss = metric * mean of sample weights, fixes #63
13_loading_and_preprocessing_da...	Make notebook 13 runnable in Colab without changes
14_deep_computer_vision_with_cn...	Make notebooks 14 to 19 runnable in Colab without changes
15_processing_sequences_using_r...	Make notebooks 14 to 19 runnable in Colab without changes



13 days ago

13 days ago

13 days ago

13 days ago

13 days ago

6 days ago

13 days ago

13 days ago

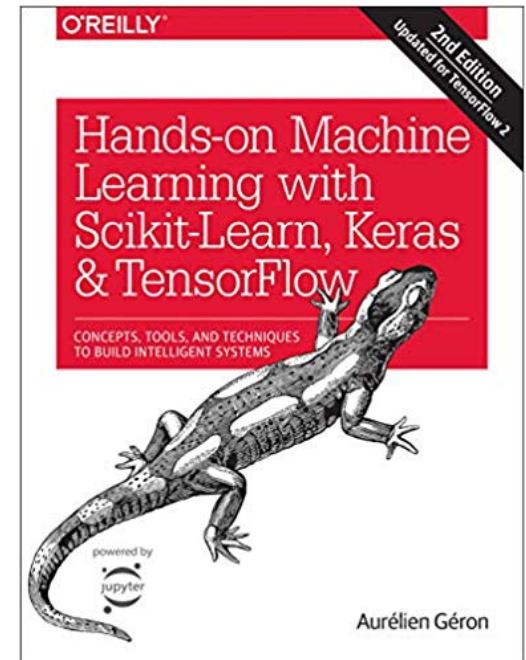
13 days ago

<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](#)



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



Search for papers, code and tasks



Browse State-of-the-Art

Follow

Discuss

Trends

About

Log In/Register

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on Twitter for updates

Computer Vision



Semantic Segmentation

33 leaderboards
667 papers with code



Image Classification

52 leaderboards
564 papers with code



Object Detection

54 leaderboards
467 papers with code



Image Generation

51 leaderboards
231 papers with code



Pose Estimation

40 leaderboards
231 papers with code

[▶ See all 707 tasks](#)

Natural Language Processing



Machine Translation



Language Modelling



Question Answering



Sentiment Analysis



Text Generation

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

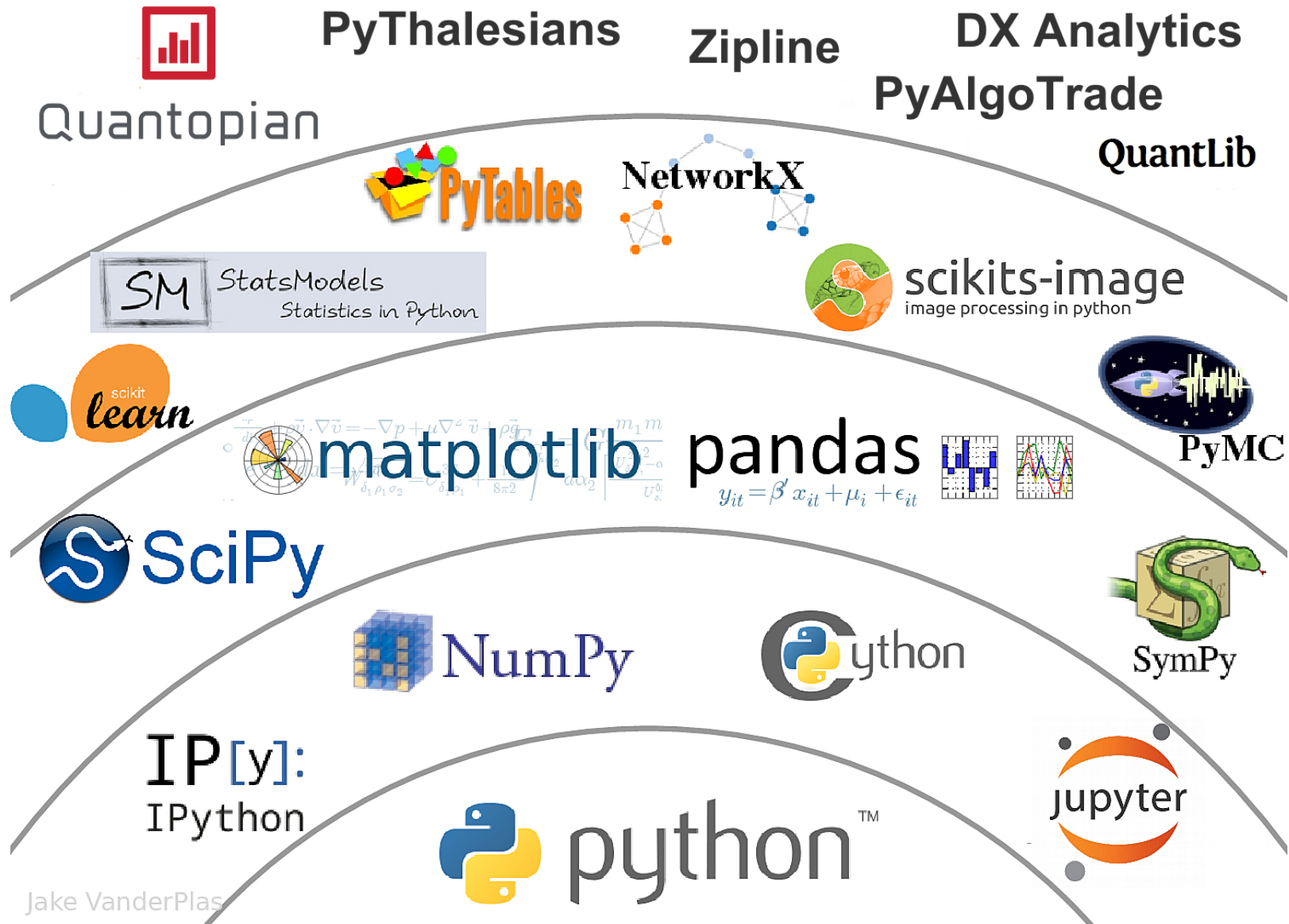
 **Stock Trend Prediction**

2 papers with code

 **Stock Prediction**

1 papers with code

The Quant Finance PyData Stack



Jake VanderPlas

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

Summary

- **Deep Learning for Investment Analysis with TensorFlow**
 - **Deep Learning**
 - **Financial Time Series Forecasting**
 - **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-da3qGB5lCeMbQuqbbCOQWcS6OYBr5A>
- 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=PLiaHhY2iBX9hdHaRr6b7XevZtgZR1PoU>
- 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=IHZwWFHw-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_OWpFT9uXLuQrImzHfOV
- <http://p.migdal.pl/2017/04/30/teaching-deep-learning.html>
- <https://github.com/leriomaggio/deep-learning-keras-tensorflow>