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

Tamkang University
淡江大學

財務金融大數據深度學習
(Deep Learning for Finance Big Data)

1061BDAF11
MIS EMBA (M2322) (8605)
Thu 12,13,14 (19:20-22:10) (D503)

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系
http://mail. tku.edu.tw/myday/

2017-12-28

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/09/21	財務金融大數據分析課程介紹 (Course Orientation for Big Data Analytics in Finance)
2	2017/09/28	金融科技商業模式 (Business Models of Fintech)
3	2017/10/05	人工智慧投資分析與機器人理財顧問 (Artificial Intelligence for Investment Analysis and Robo-Advisors)
4	2017/10/12	金融科技對話式商務與智慧型交談機器人 (Conversational Commerce and Intelligent Chatbots for Fintech)
5	2017/10/19	事件研究法 (Event Study)
6	2017/10/26	財務金融大數據分析個案研究 I (Case Study on Big Data Analytics in Finance I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2017/11/02	Python 財務大數據分析基礎 (Foundations of Finance Big Data Analytics in Python)
8	2017/11/09	Python Numpy大數據分析 (Big Data Analytics with Numpy in Python)
9	2017/11/16	Python Pandas 財務大數據分析 (Finance Big Data Analytics with Pandas in Python)
10	2017/11/23	期中報告 (Midterm Project Report)
11	2017/11/30	Python Keras深度學習 (Deep Learning with Keras in Python)
12	2017/12/07	文字探勘分析技術與自然語言處理 (Text Mining Techniques and Natural Language Processing) [Invited Speaker: Irene Chen, Consultant, Teradata]

課程大綱 (Syllabus)

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

Deep Learning for Finance Big Data

Python Pandas for Finance

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

Python Pandas for Finance

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

```
#Python for Stocks: 1
#Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
#Import Packages Required
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline
```

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

Python Pandas for Finance

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

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

Finance Data from Quandl

```
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
import quandl
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2017-10-25	156.91	157.5500	155.27	156.405	20126554.0	0.0	1.0	156.91	157.5500	155.27	156.405	20126554.0
2017-10-26	157.23	157.8295	156.78	157.410	16751691.0	0.0	1.0	157.23	157.8295	156.78	157.410	16751691.0
2017-10-27	159.29	163.6000	158.70	163.050	43904150.0	0.0	1.0	159.29	163.6000	158.70	163.050	43904150.0
2017-10-30	163.89	168.0700	163.72	166.720	43923292.0	0.0	1.0	163.89	168.0700	163.72	166.720	43923292.0
2017-10-31	167.90	169.6499	166.94	169.040	35474672.0	0.0	1.0	167.90	169.6499	166.94	169.040	35474672.0

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

Python Pandas for Finance

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

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

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

Python Pandas for Finance

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

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

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

Python Pandas for Finance

```
<matplotlib.text.Text at 0x115630860>
```

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

Python Pandas for Finance

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

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

<matplotlib.text.Text at 0x1153630b60>

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

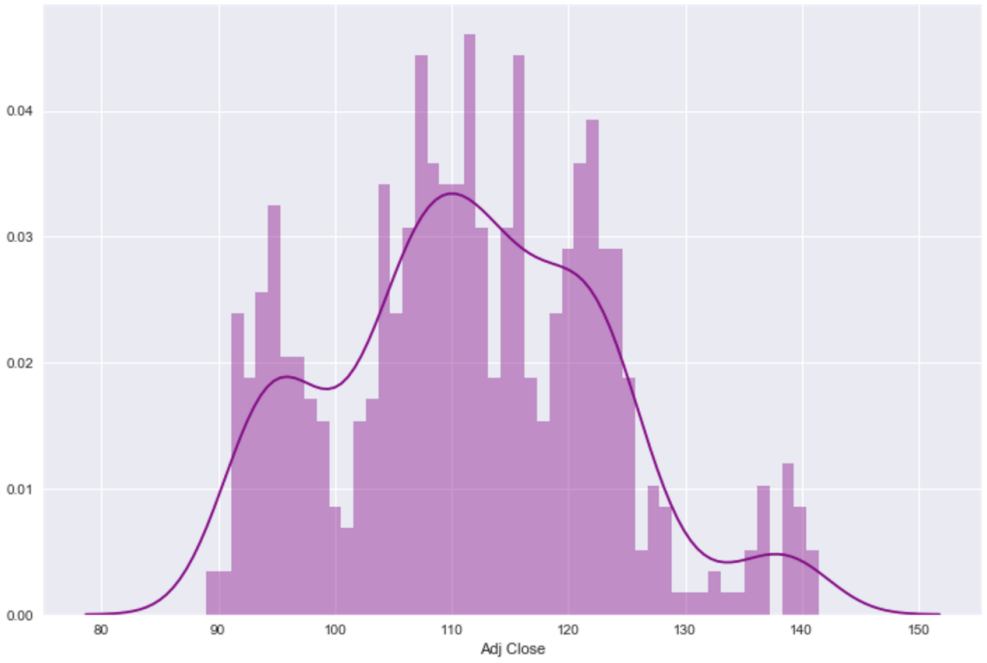
Python Pandas for Finance

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

```
plt.figure(figsize=(12,8))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
```

```
/Users/imyday/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools
g: using a non-integer number instead of an integer will result in an error in the fu
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x116309780>
```



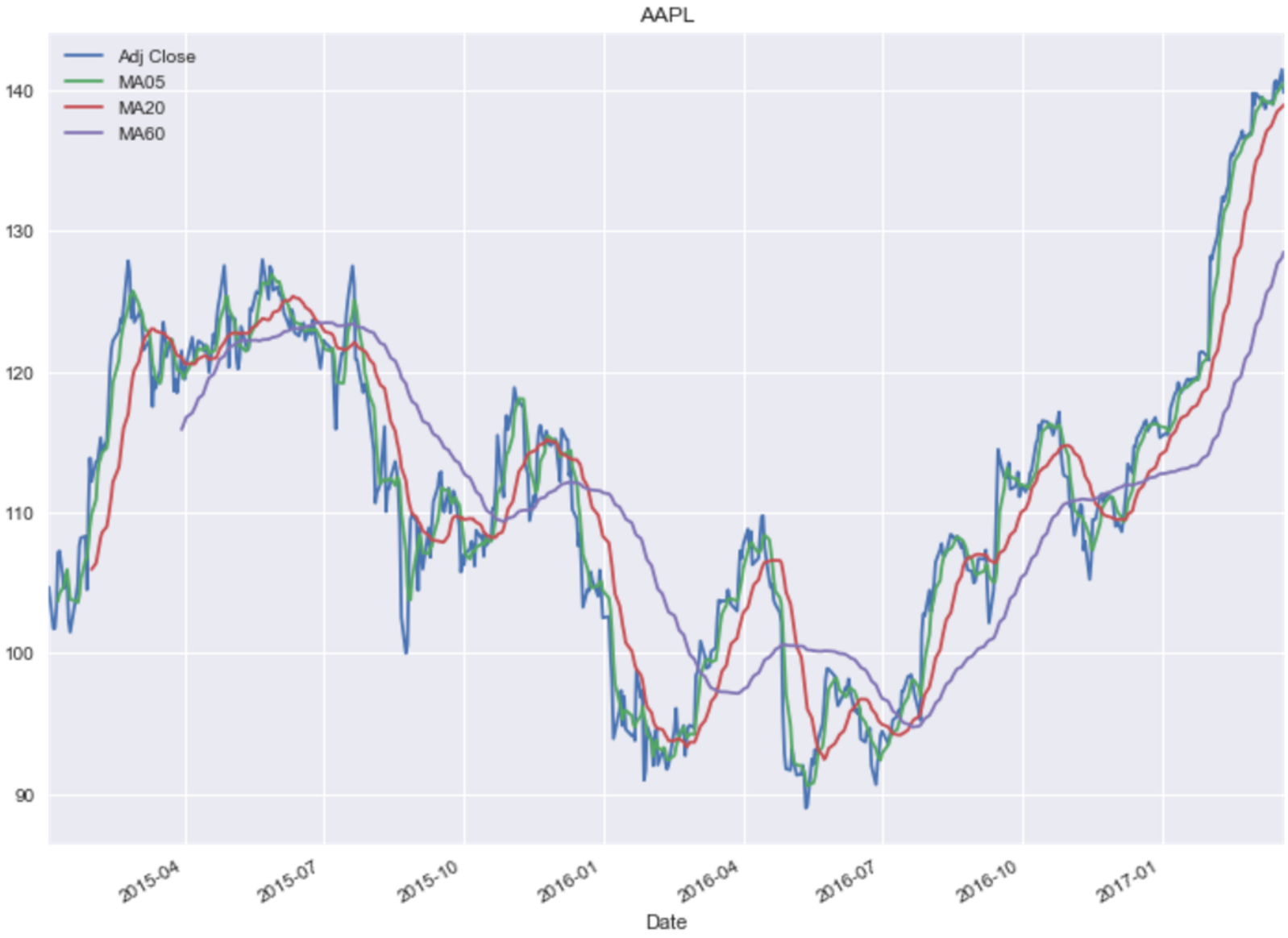
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

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

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

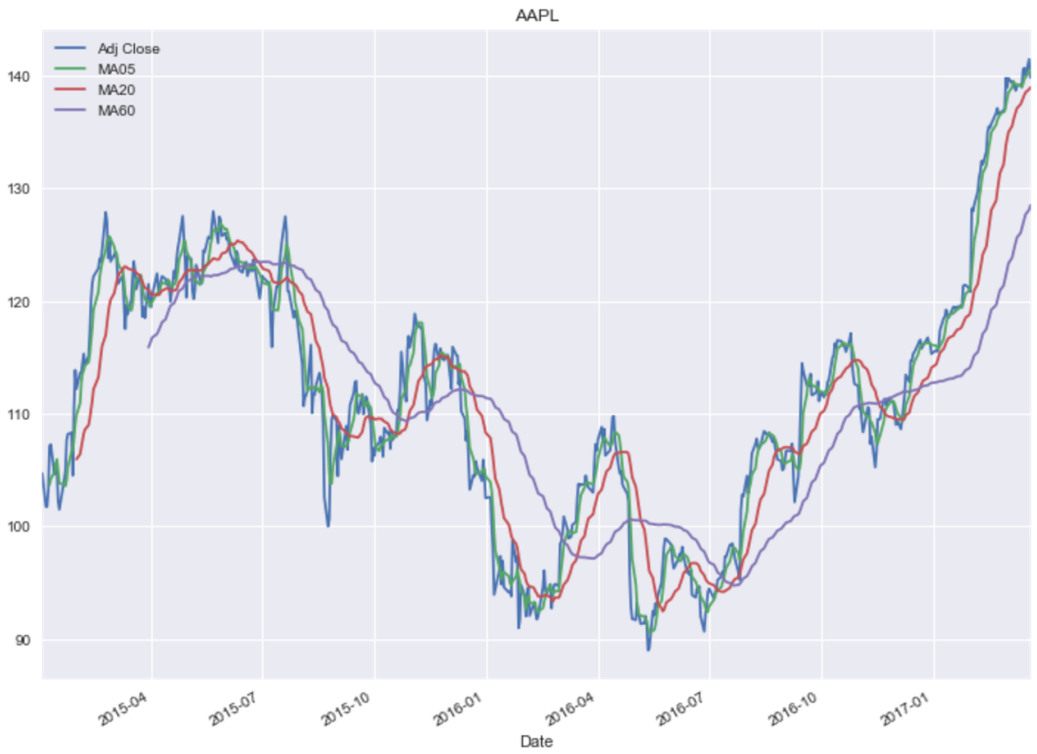
Python Pandas for Finance



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

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

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

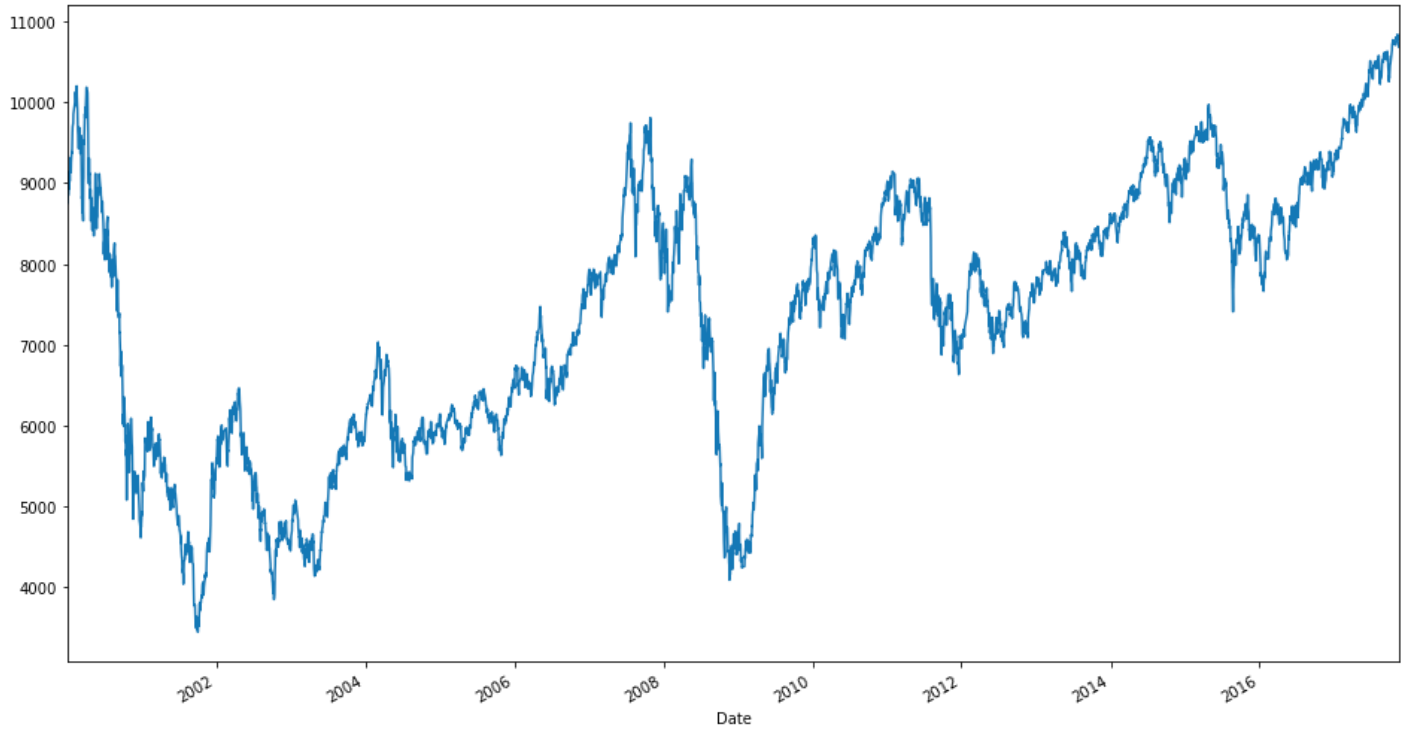


Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

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

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

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

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

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

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

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

```
import matplotlib.pyplot as plt
%matplotlib inline
import fix_yahoo_finance as yf
df = yf.download("^TWII", start="2000-01-01", end="2017-11-15")
df.to_csv('YF_TWII_2000_2017.csv')
print(df.head())
fig = plt.figure(figsize=(16,9))
df["Adj Close"].plot()
fig.show()
```



candlestick_ohlc

```
import matplotlib.pyplot as plt

from matplotlib.finance
import candlestick_ohlc
```

Source: https://matplotlib.org/examples/pylab_examples/finance_demo.html

```
import matplotlib.pyplot as plt
from matplotlib.finance import candlestick_ohlc
```

Source: https://ntguardian.wordpress.com/2016/09/19/introduction-stock-market-data-python-1/

daily_to_weekly

```
#Convert Daily Data to Weekly Data
def daily_to_weekly(df):
    #dfWeekly = daily_to_weekly(df)
    #df.sort_index(axis=0, level=None, ascending=True, inplace=True)
    Open = df.Open.resample('W-Fri').first() #W #W-MON #W-Fri
    High = df.High.resample('W-Fri').max()
    Low = df.Low.resample('W-Fri').min()
    Close = df.Close.resample('W-Fri').last()
    Volume = df.Volume.resample('W-Fri').sum()
    Adj_Close = df["Adj Close"].resample('W-Fri').last()
    dfWeekly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfWeekly = dfWeekly[pd.notnull(dfWeekly['Adj Close'])]
    return dfWeekly
```

daily_to_monthly

```
#Convert Daily Data to Monthly Data
def daily_to_monthly(df):
    #dfMonthly = daily_to_monthly(df)
    Open = df.Open.resample('M').first()
    High = df.High.resample('M').max()
    Low = df.Low.resample('M').min()
    Close = df.Close.resample('M').last()
    Volume = df.Volume.resample('M').sum()
    Adj_Close = df["Adj Close"].resample('M').last()
   dfMonthly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfMonthly = dfMonthly[pd.notnull(dfMonthly['Adj Close'])]
    return dfMonthly
```

TA-Lib:
Technical Analysis Library

TA-Lib : Technical Analysis Library

Home

Products
Downloads
Purchase
Support

Function List

Source Code
Community Forum
Useful Links

About Us

Multi-Platform Tools for Market Analysis ...

TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data.

- Includes 200 indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands etc... (more info)
- Candlestick pattern recognition
- Open-source API for C/C++, Java, Perl, Python and 100% Managed .NET

Free Open-Source Library

TA-Lib is available under a BSD License allowing it to be integrated in your own open-source or commercial application. (more info)

Commercial Application

TA-Lib is also available as an easy to install Excel Add-Ins. Try it for free!

http://ta-lib.org/

Stochastic Oscillator (KD)

```
#Stochastic oscillator %D
def KDJ(df, n, m1, m2):
    #KDJ(df, 9, 3, 3)
    KDJ_n = n
    KDJ_m1 = m1
    KDJ_m2 = m2

    df['Low_n'] = pd.rolling_min(df['Low'], KDJ_n)
    df['Low_n'].fillna(value=pd.expanding_min(df['Low']), inplace=True)
    df['High_n'] = pd.rolling_max(df['High'], KDJ_n)
    df['High_n'].fillna(value=pd.expanding_max(df['High']), inplace=True)

    df['RSV'] = (df['Close'] - df['Low_n']) / (df['High_n'] - df['Low_n']) * 100

    df['KDJ_K'] = pd.ewma(df['RSV'], KDJ_m1)
    df['KDJ_D'] = pd.ewma(df['KDJ_K'], KDJ_m2)
    df['KDJ_J'] = 3 * df['KDJ_K'] - 2 * df['KDJ_D']
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#Bollinger Bands
def BBANDS20(df, n):
    MA = pd.Series(pd.rolling_mean(df['Close'], n))
    MSD = pd.Series(pd.rolling_std(df['Close'], n))
    b1 = 4 * MSD / MA
    B1 = pd.Series(b1, name = 'BollingerB_' + str(n))
    df = df.join(B1)
    b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
    B2 = pd.Series(b2, name = 'Bollinger%b_' + str(n))
    df = df.join(B2)
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#BB Bollinger Bands BB_20
def BB_20(df):
    df['BB_MA20'] = pd.stats.moments.rolling_mean(df["Adj Close"], 20)
    df['BB_SD20'] = pd.stats.moments.rolling_std(df['Adj Close'],20)
    df['BB_UpperBand'] = df['BB_MA20'] + (df['BB_SD20']*2) # Default 2*SD
    df['BB_LowerBand'] = df['BB_MA20'] - (df['BB_SD20']*2)
    df['BB_PB'] = (df['Adj Close'] - df['BB_LowerBand']) / (df['BB_UpperBand'] -
df['BB_LowerBand'])
    df['BB_BW'] = (df['BB_UpperBand'] - df['BB_LowerBand']) / df['BB_MA20']
    df['BB_UpperBand_1SD'] = df['BB_MA20'] + (df['BB_SD20'])
    df['BB_LowerBand_1SD'] = df['BB_MA20'] - (df['BB_SD20'])
    #BB_PB: Bollinger Band Percent b (PB)
    #BB_BW: Bollinger Band Band Width (BW)
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Examples: Python Pandas for Finance

In [11]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas.io.data as web
import random

import datetime
import time
import timeit

import io
import os

import re
import codecs
import requests
get_ipython().magic('matplotlib inline')

from scipy import stats

#pd.set_option('display.notebook_repr_html', False)
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 10)
pd.set_option('precision', 3)

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

print('Hello Pandas')
print(getDateTimeNow())
```

```
Hello Pandas
20160323_145708
```

```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [13]:
```
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
#"^TWII"
#"AAPL"
#SHA:000016"
#"600000.SS"
#"2330.TW"
df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df_01 = web.DataReader("2330.TW", 'yahoo')
sSymbol = sSymbol.replace(":","_")
sSymbol = sSymbol.replace("^","_")
sPath = "data/financedata/"
#sPath = "/users/imyday/SCDBA/data/"  #Mac OS X
#sPath = "C:\data\"                   #Windows

sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
print(sPathFilename)
df.to_csv(sPathFilename)
df.head(5)
```

```
data/financedata/AAPL_Yahoo_Finance.csv
```

Out[13]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2000-01-03	104.875	112.500	101.688	111.938	133949200	3.702
2000-01-04	108.250	110.625	101.188	102.500	128094400	3.390

df.tail(5)

```
In [14]: df.tail(5)
```

Out[14]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-12-24	109.00	109.00	107.95	108.03	13596700	107.447
2015-12-28	107.59	107.69	106.18	106.82	26704200	106.243
2015-12-29	106.96	109.43	106.86	108.74	30931200	108.153
2015-12-30	108.58	108.70	107.18	107.32	25213800	106.741
2015-12-31	107.01	107.03	104.82	105.26	40912300	104.692

```
sSymbol = "AAPL”

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [15]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

```
data/AAPL.csv
```

data/AAPL.csv

Out[15]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-22	105.25	107.29	105.21	106.72	32232600	106.72
2016-03-21	105.93	107.65	105.14	105.91	35180800	105.91
2016-03-18	106.34	106.50	105.19	105.92	43402300	105.92
2016-03-17	105.52	106.47	104.96	105.80	34244600	105.80
2016-03-16	104.61	106.31	104.59	105.97	37893800	105.97

```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
   df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

In [16]:
```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
    df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

```
sSymbol = "AAPL”
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime,
sDir)
print(sPathFilename)
```

In [17]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime, sDir)
print(sPathFilename)
```

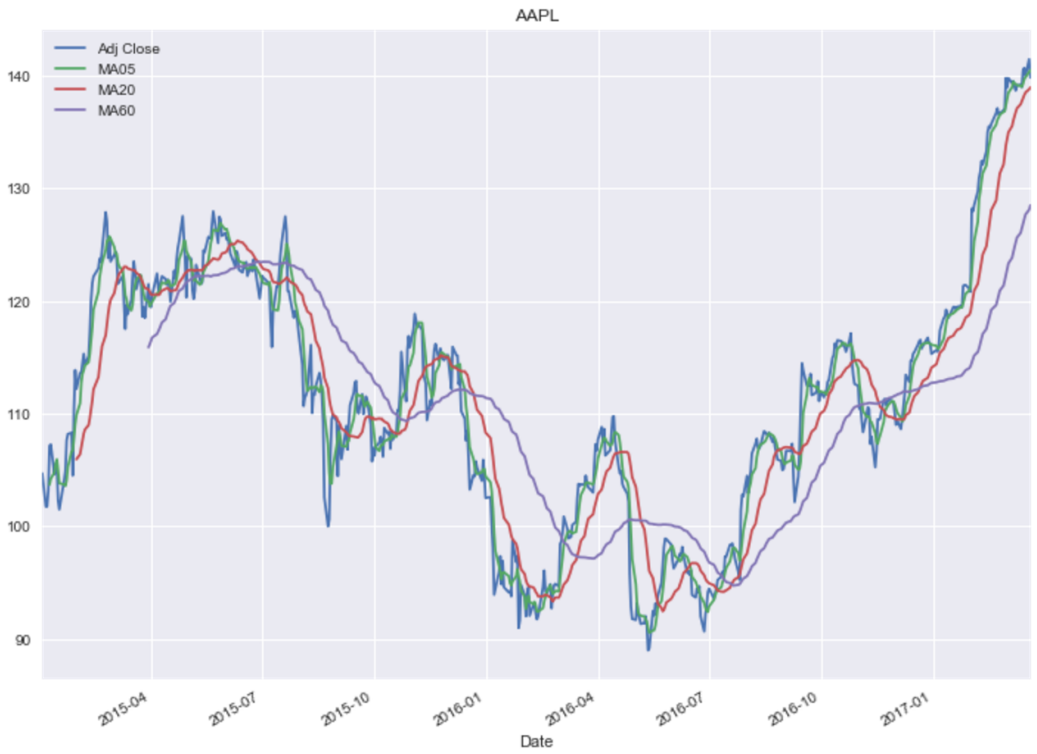
```
data/financedata/AAPL_Yahoo_Finance.csv
```

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

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

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

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

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

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

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

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

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

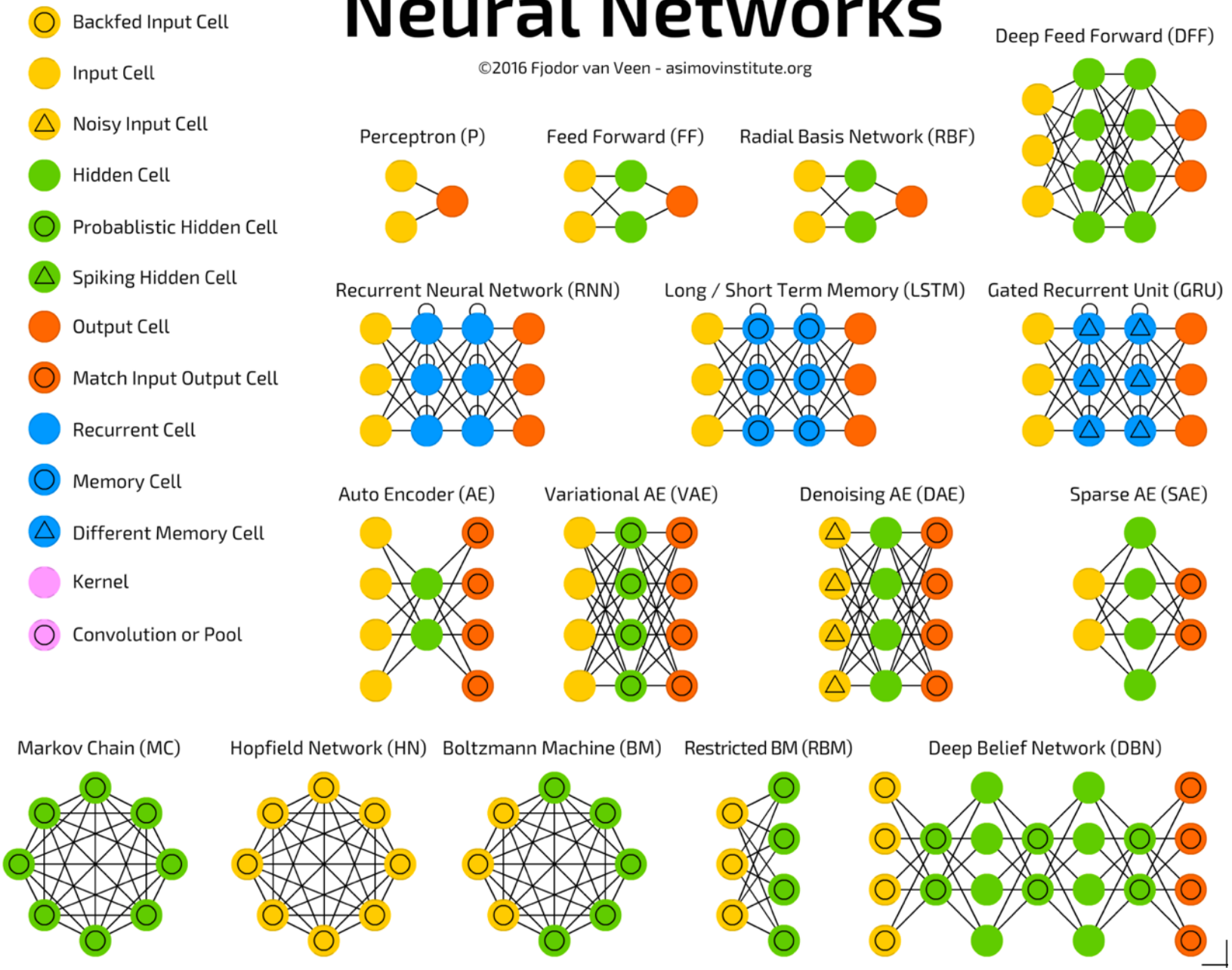


Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Deep Learning

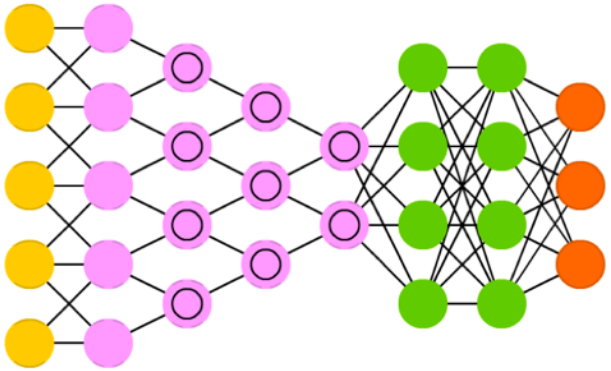
A mostly complete chart of

Neural Networks

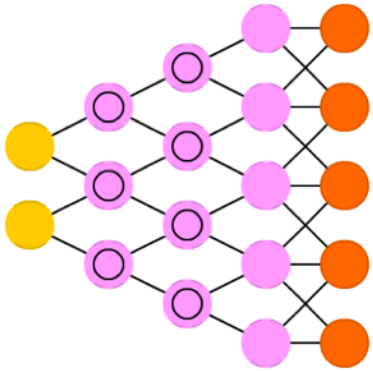


Source: http://www.asimovinstitute.org/neural-network-zoo/

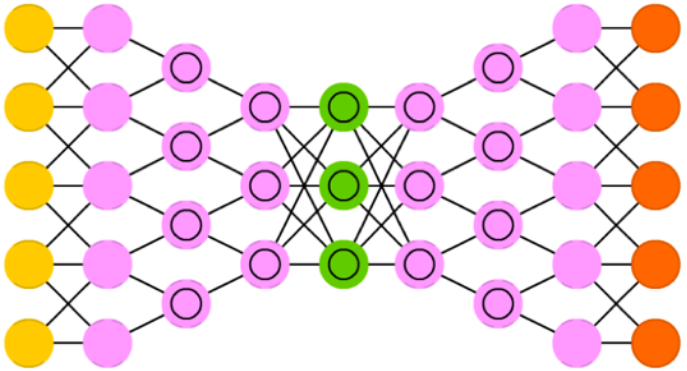
Deep Convolutional Network (DCN)



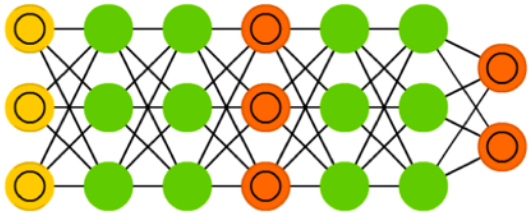
Deconvolutional Network (DN)



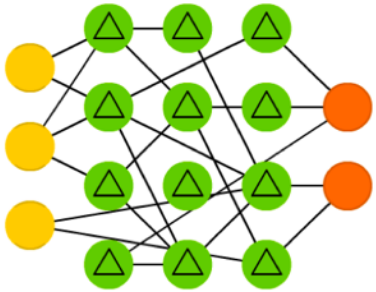
Deep Convolutional Inverse Graphics Network (DCIGN)



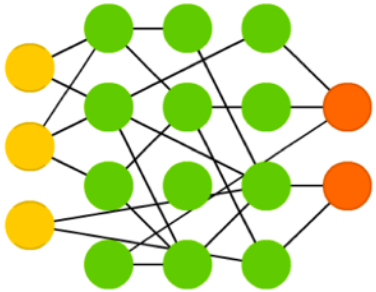
Generative Adversarial Network (GAN)



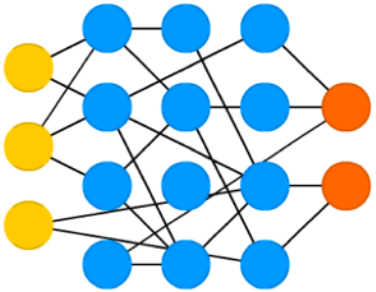
Liquid State Machine (LSM)



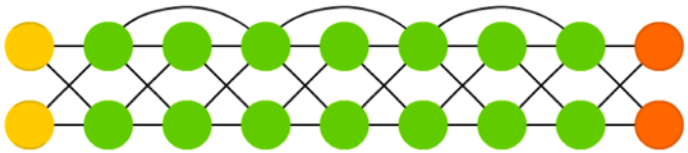
Extreme Learning Machine (ELM)



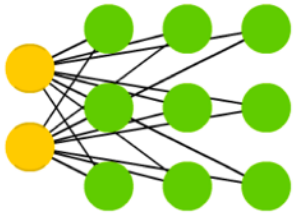
Echo State Network (ESN)



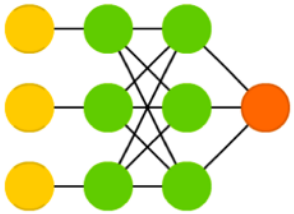
Deep Residual Network (DRN)



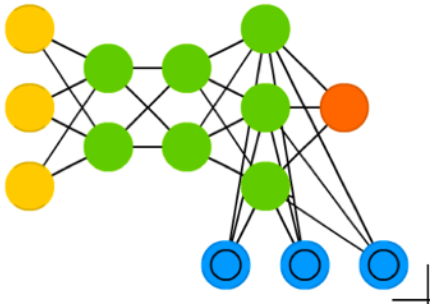
Kohonen Network (KN)



Support Vector Machine (SVM)



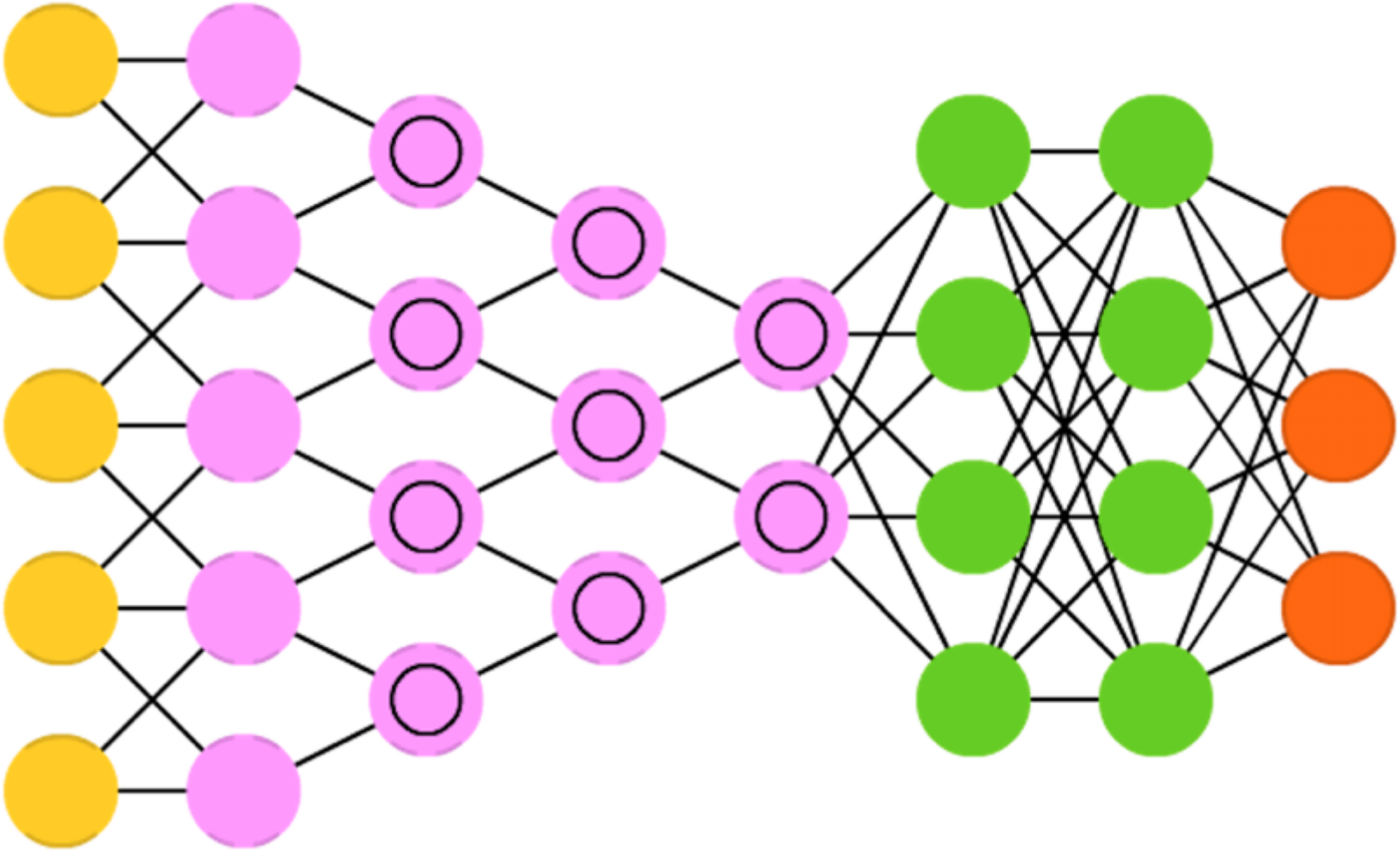
Neural Turing Machine (NTM)



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

Convolutional Neural Networks

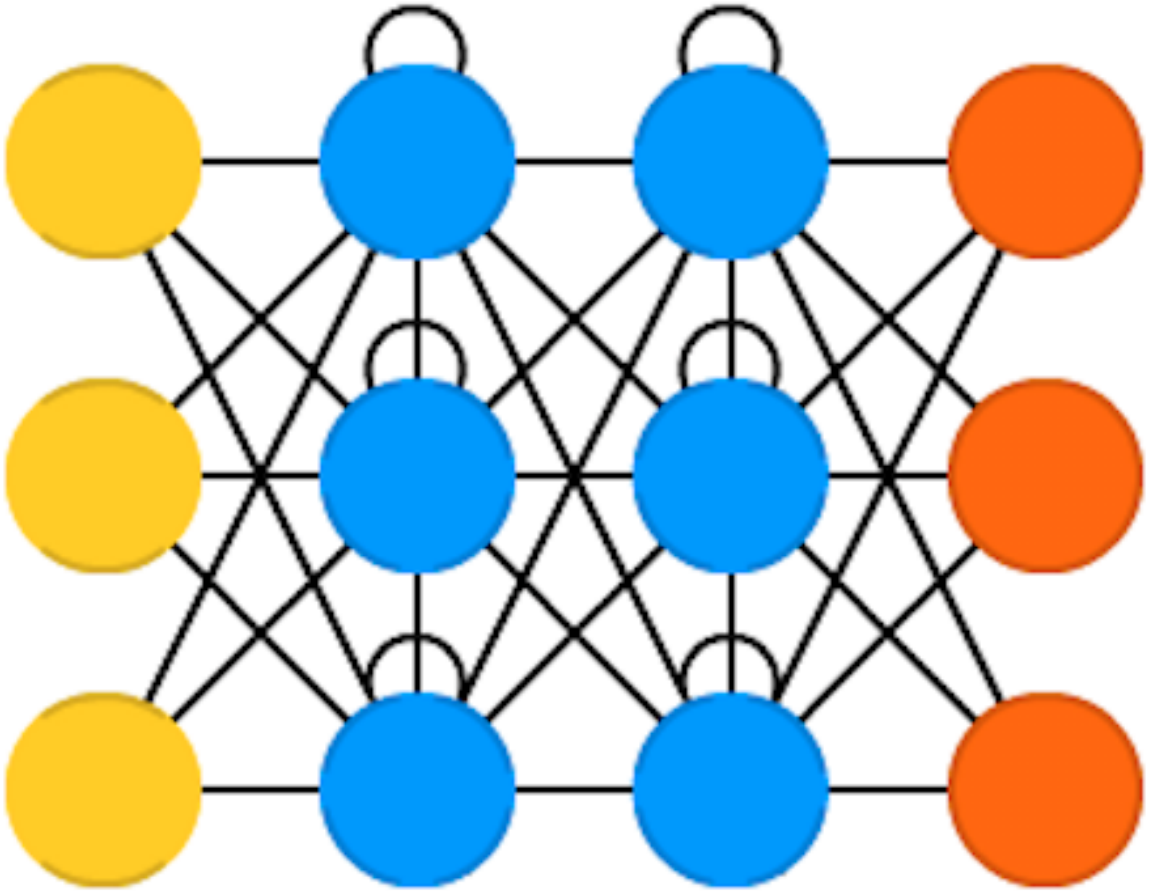
(CNN or Deep Convolutional Neural Networks, DCNN)



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

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

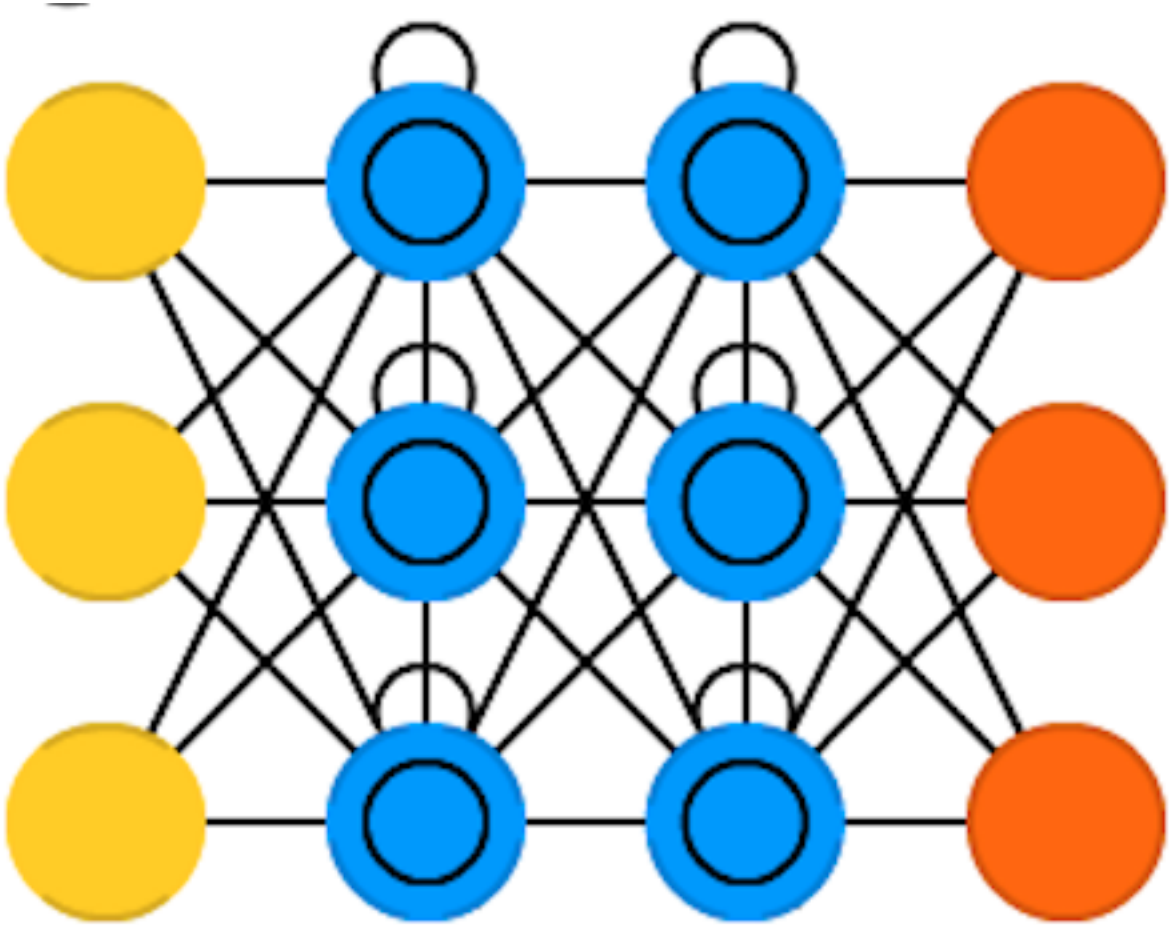
Recurrent Neural Networks (RNN)



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

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

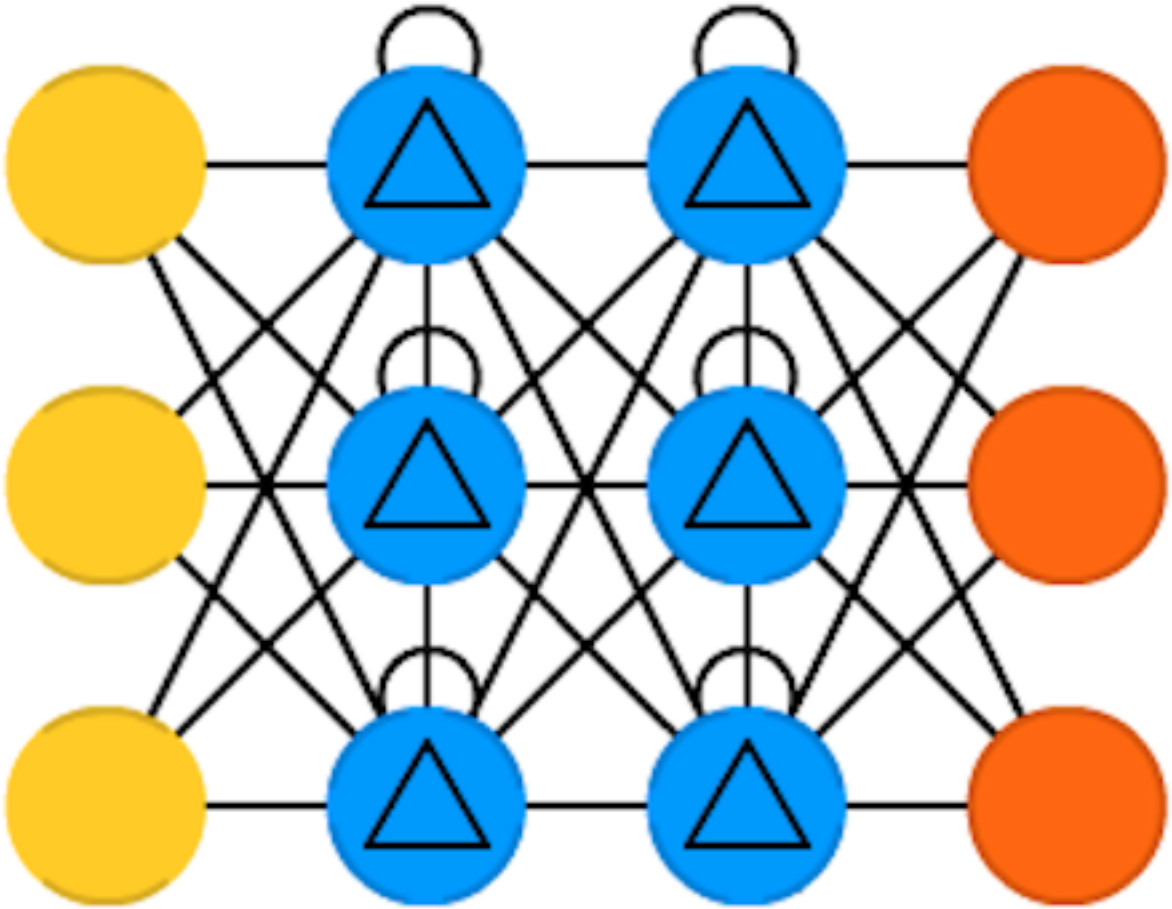
Long / Short Term Memory (LSTM)



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

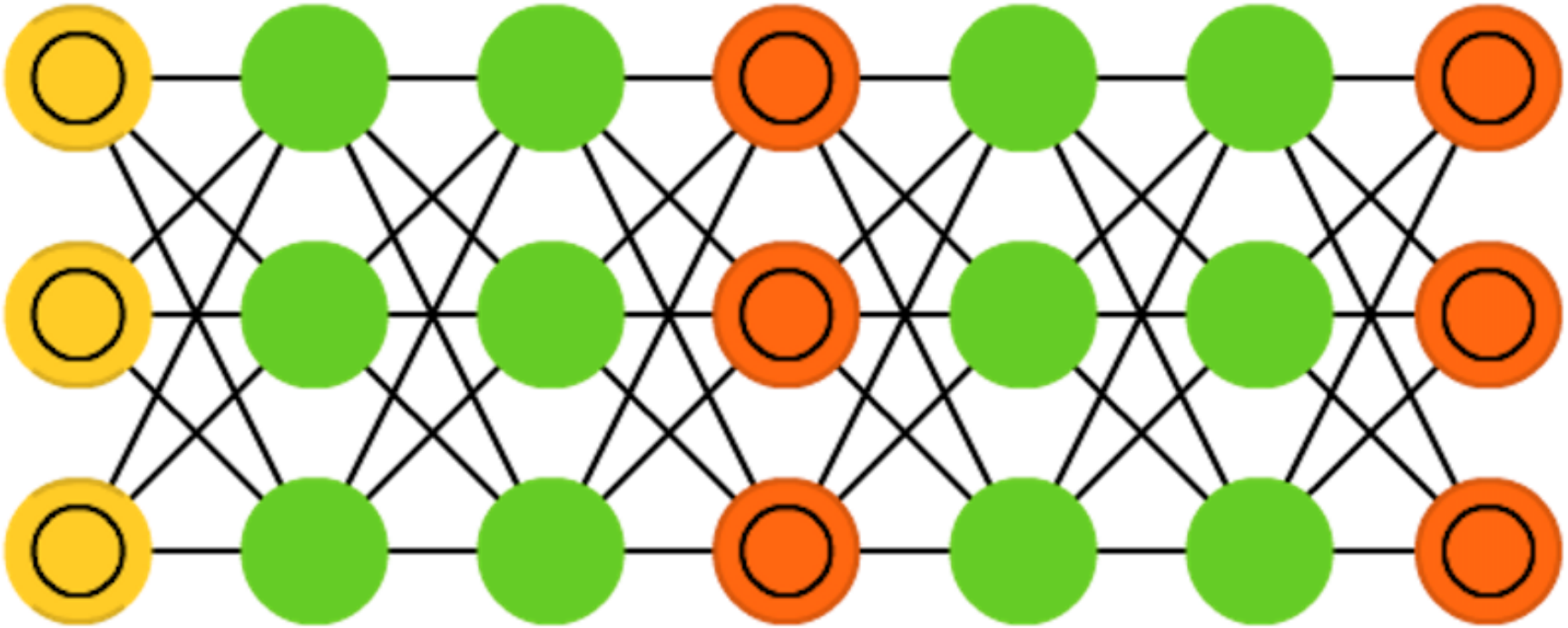
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



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

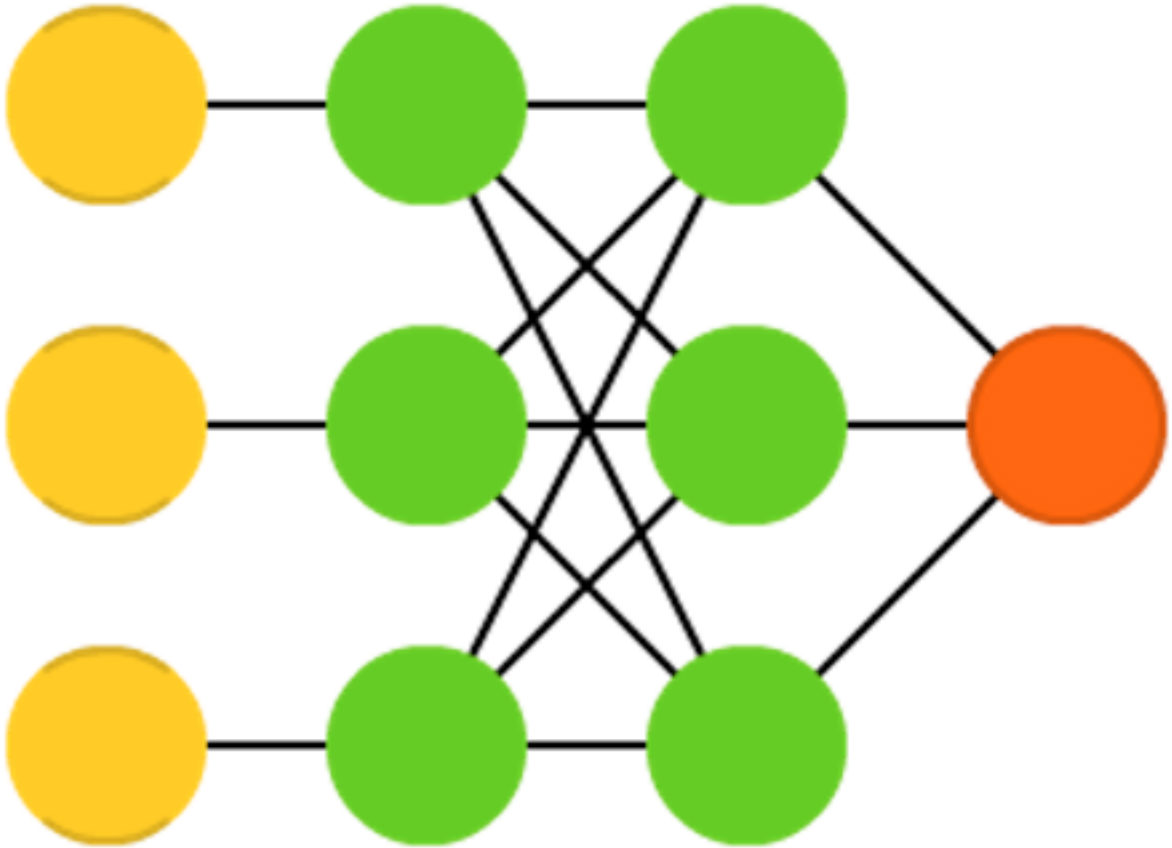
Generative Adversarial Networks (GAN)



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

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

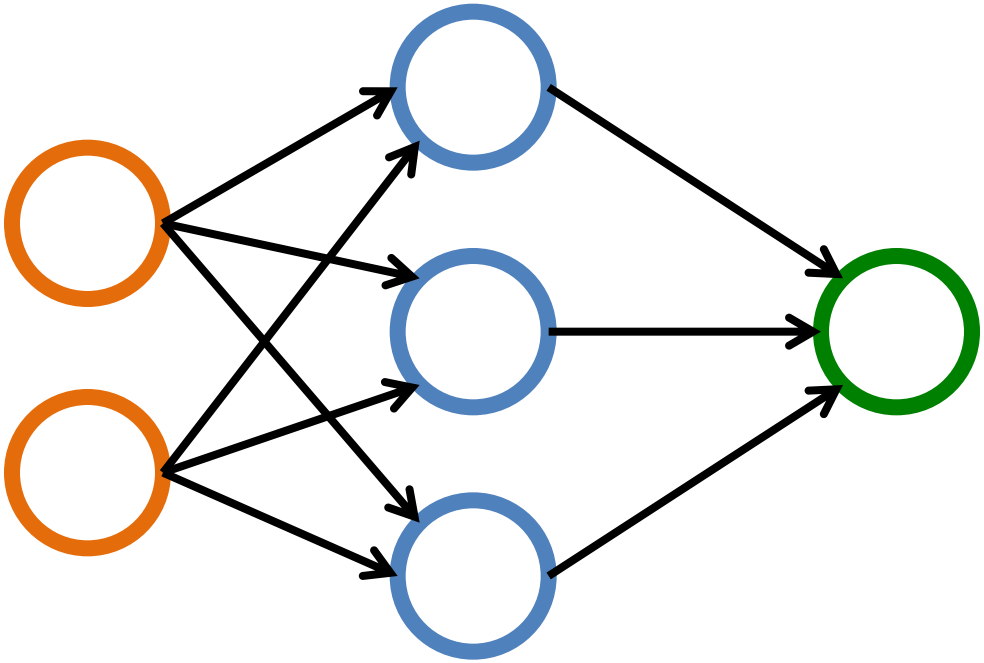
Support Vector Machines (SVM)



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

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

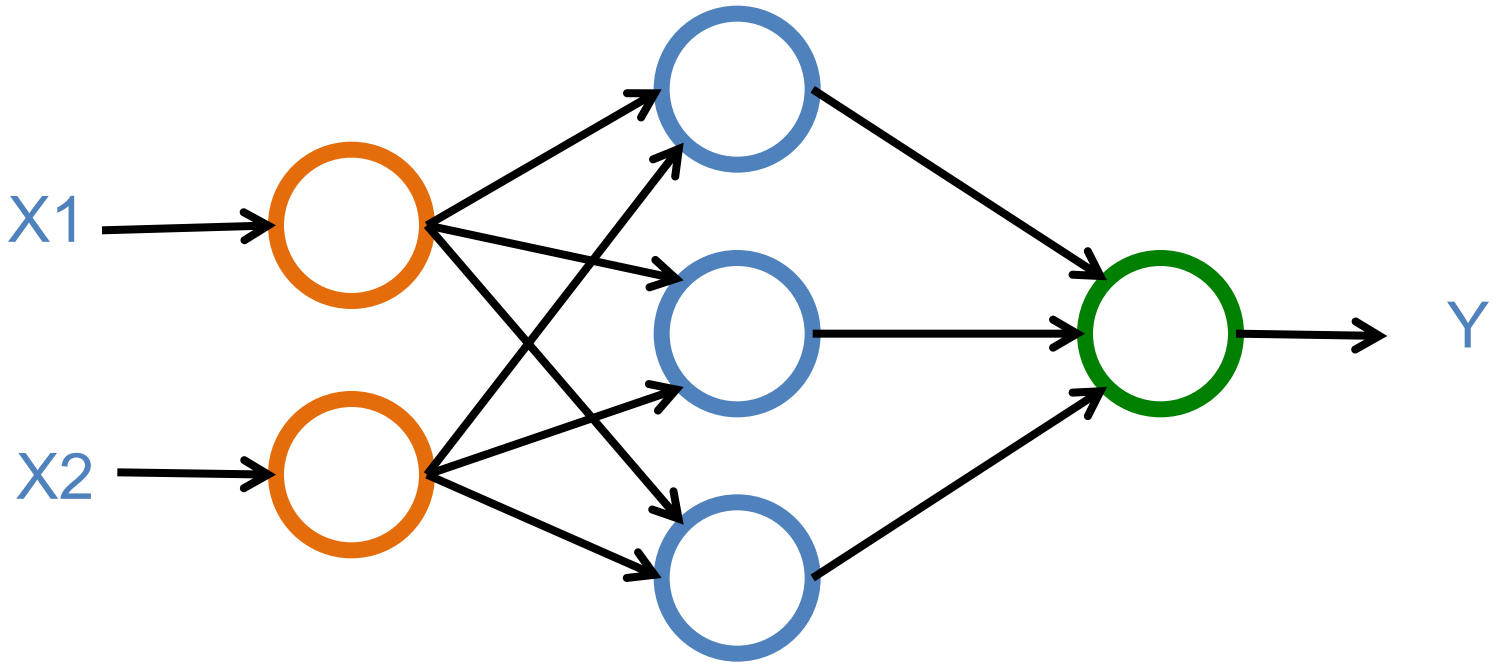
Neural Networks



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

Neural Networks

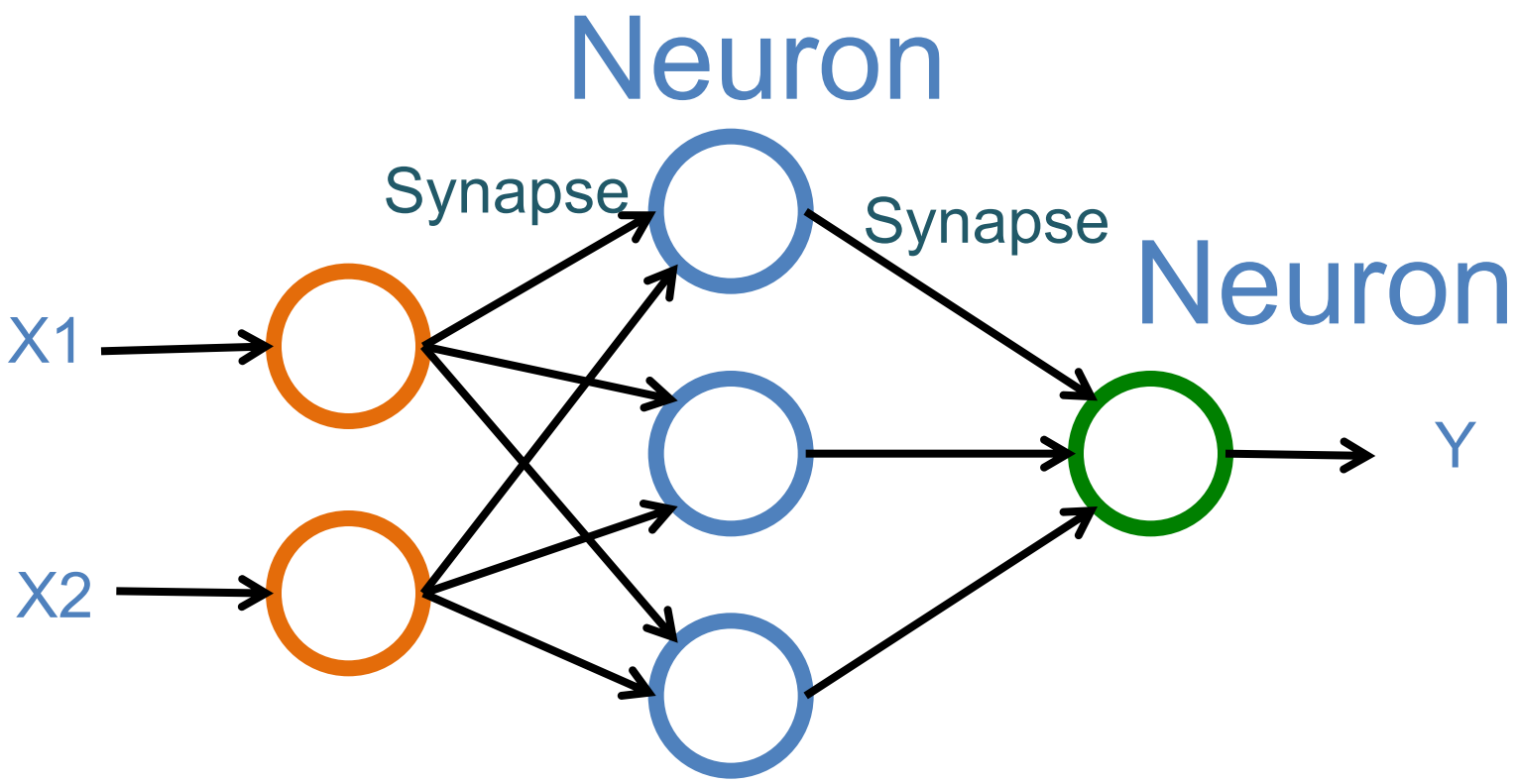
Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



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

Neural Networks

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



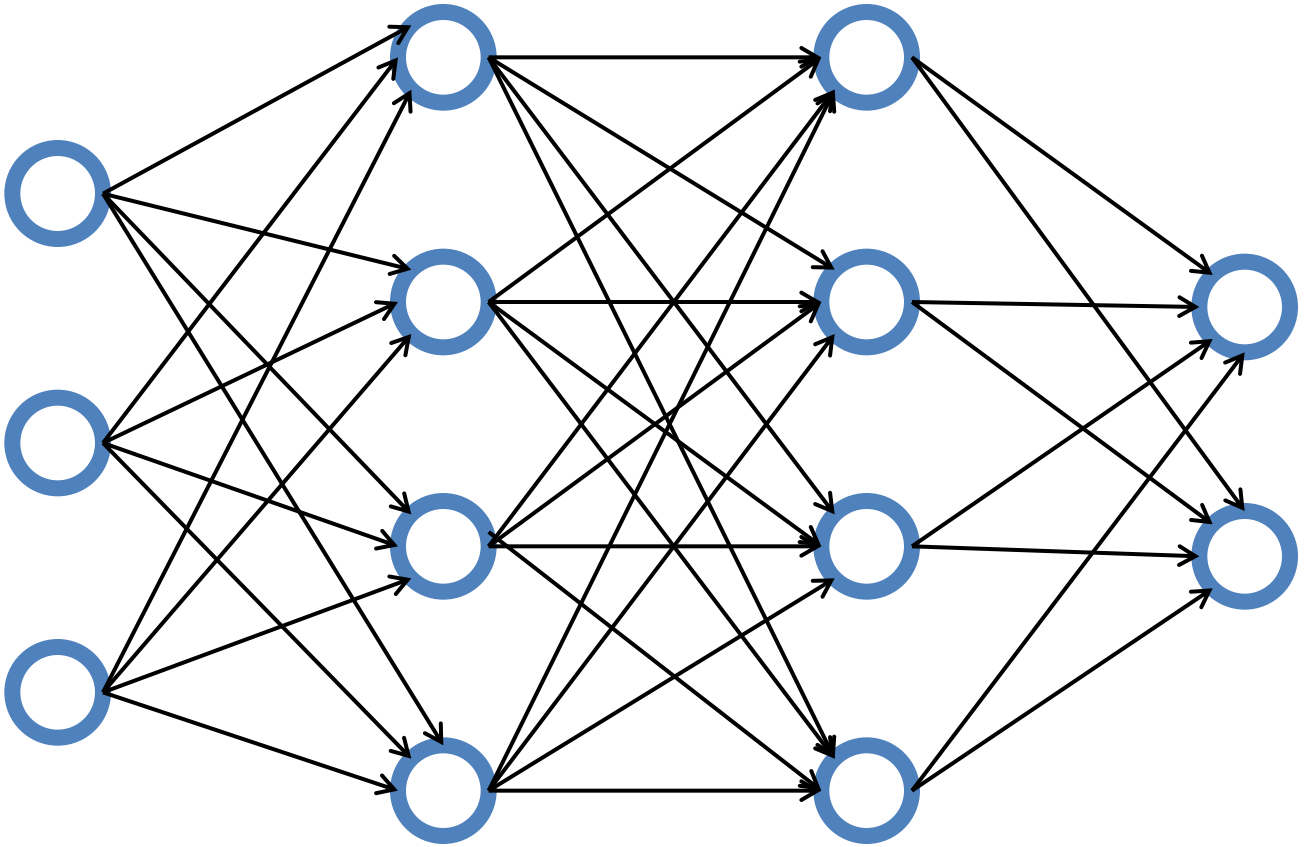
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Neural Networks

Input Layer (X)

Hidden Layer (H)

Output Layer (Y)



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

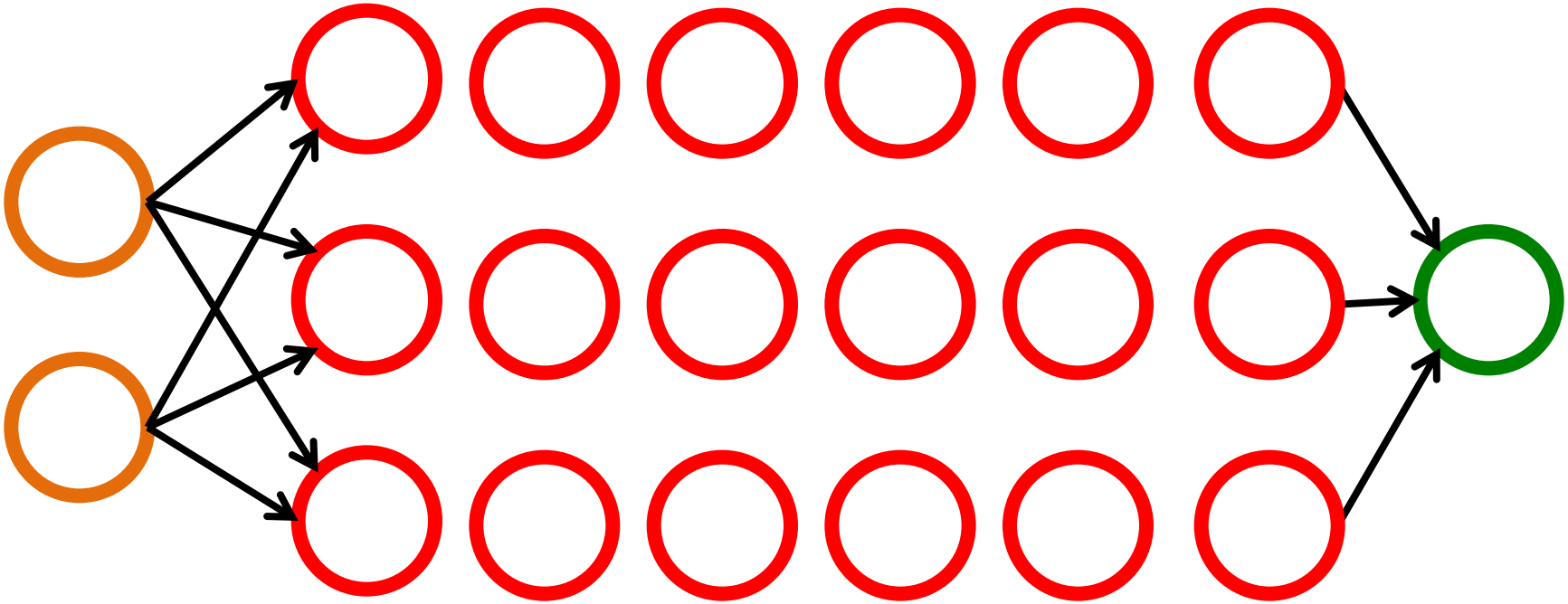
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

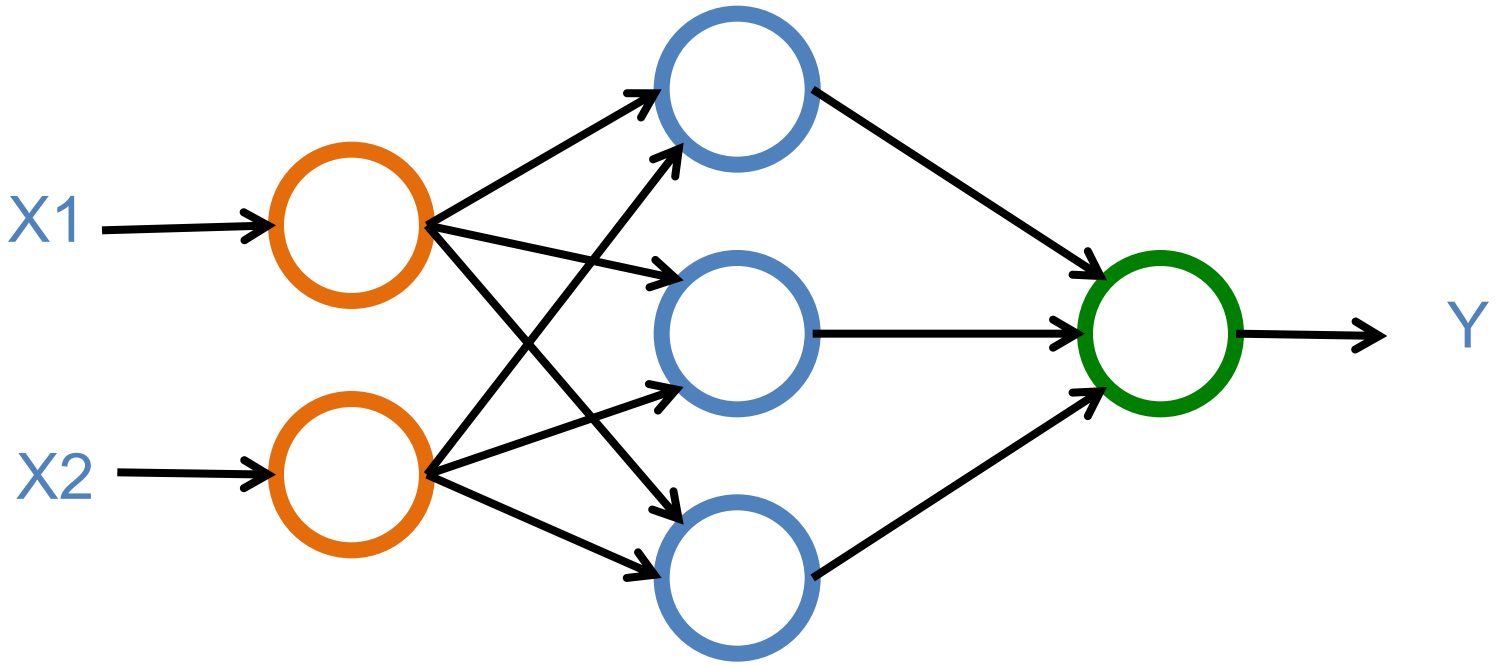
Deep Neural Networks
Deep Learning



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

Neural Networks

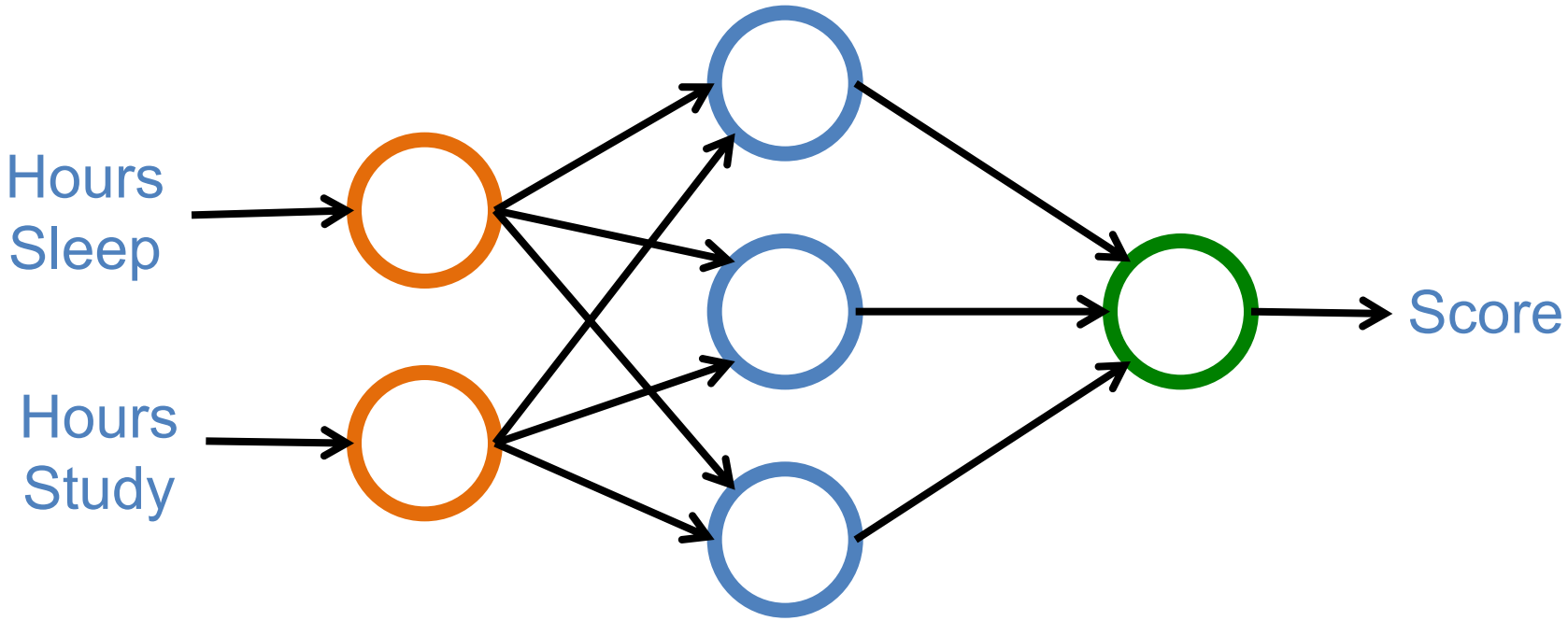
Input Layer (X) Hidden Layer (H) **Output Layer (Y)**



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

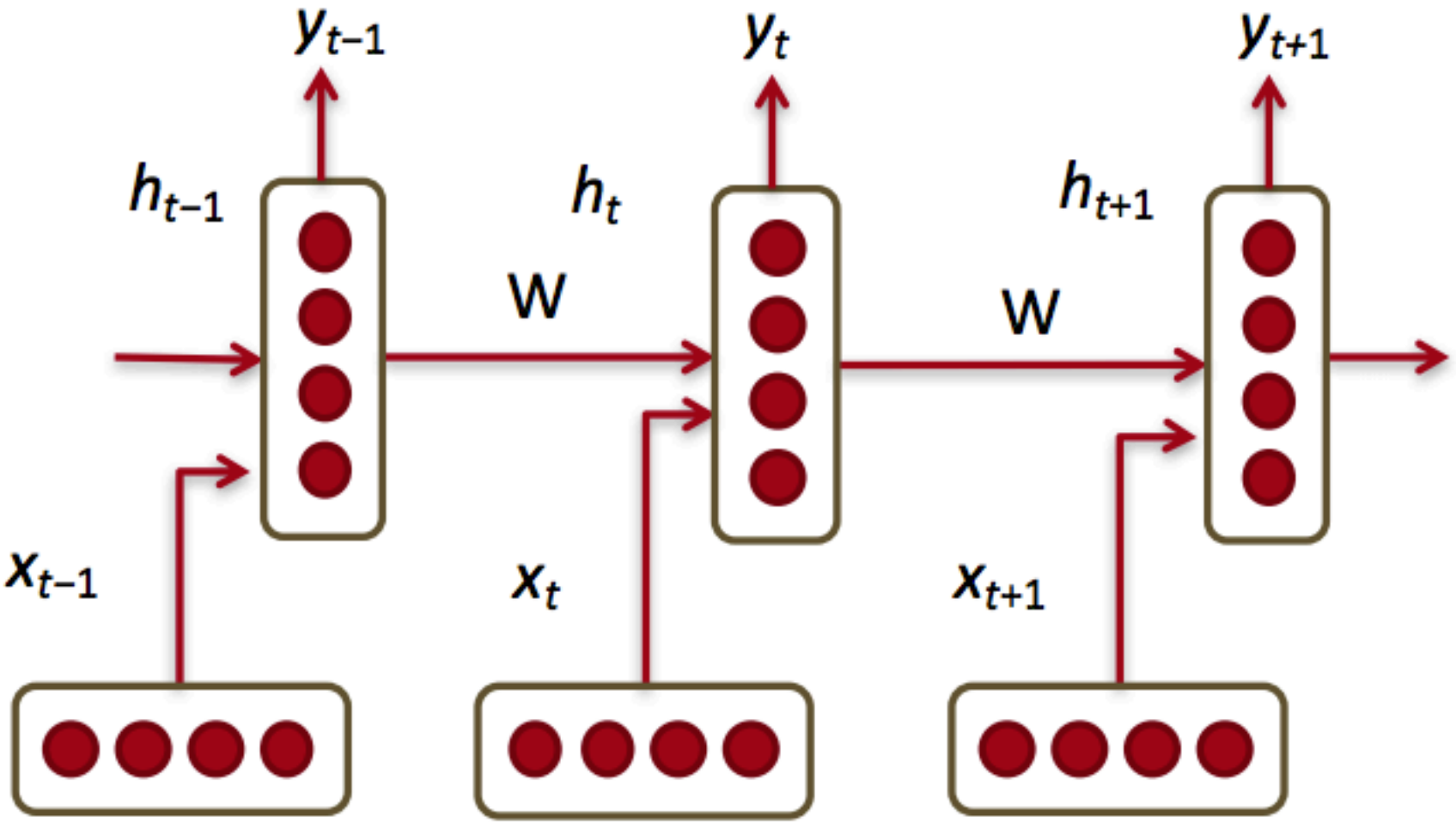
Neural Networks

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



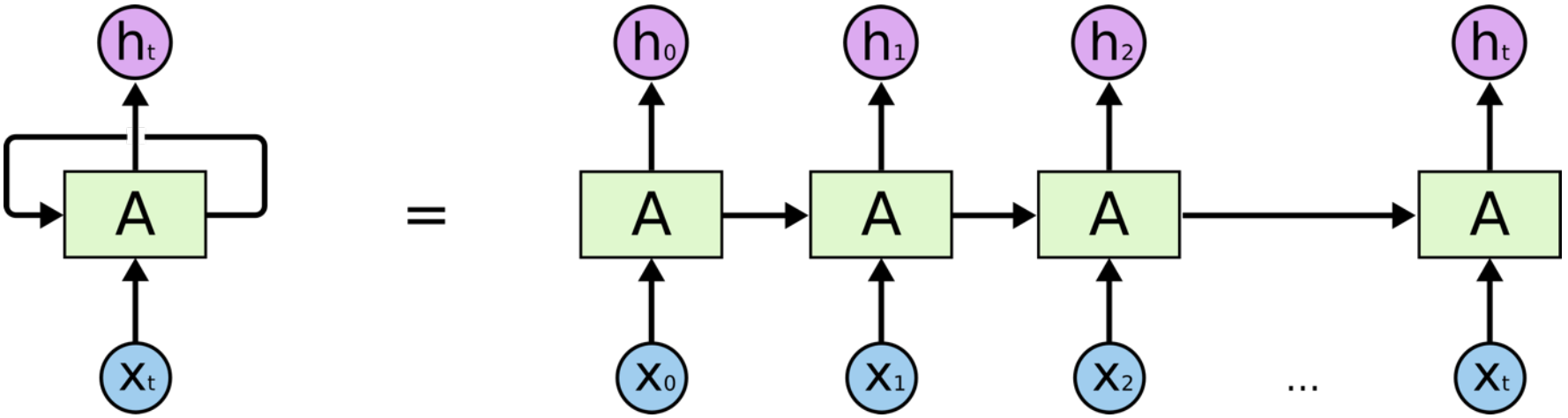
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Recurrent Neural Networks (RNNs)



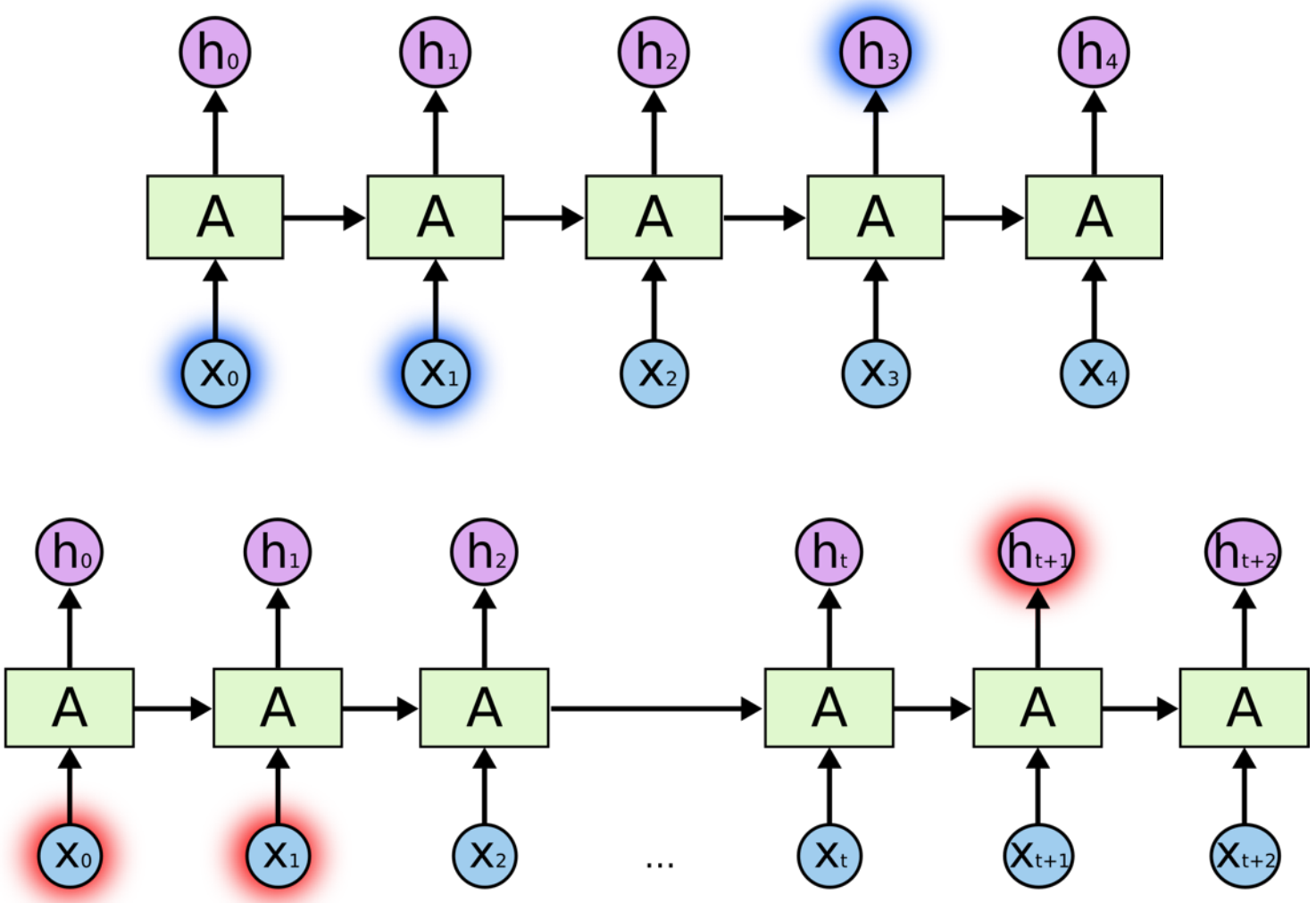
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN



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

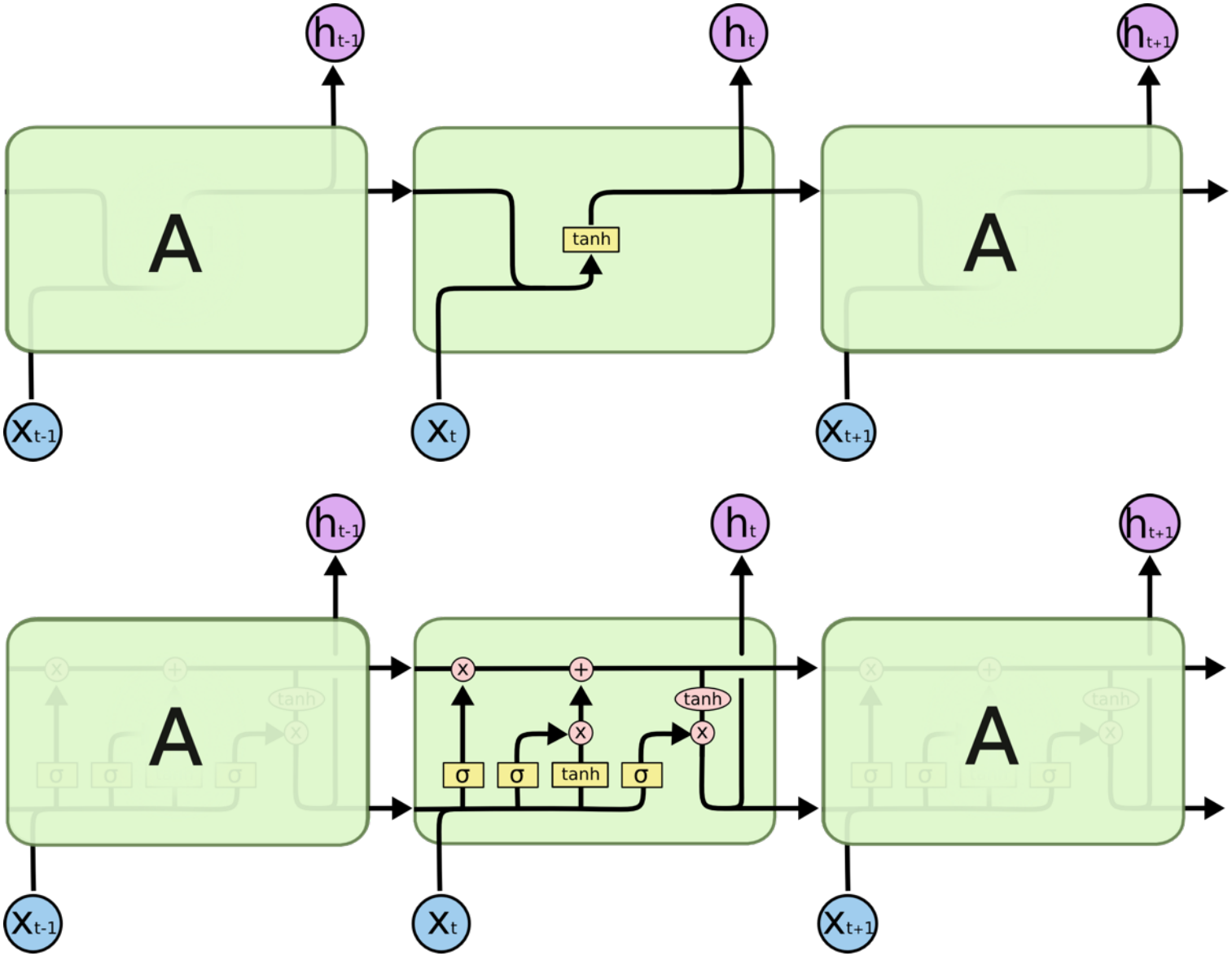
RNN long-term dependencies



I grew up in France… I speak fluent French.

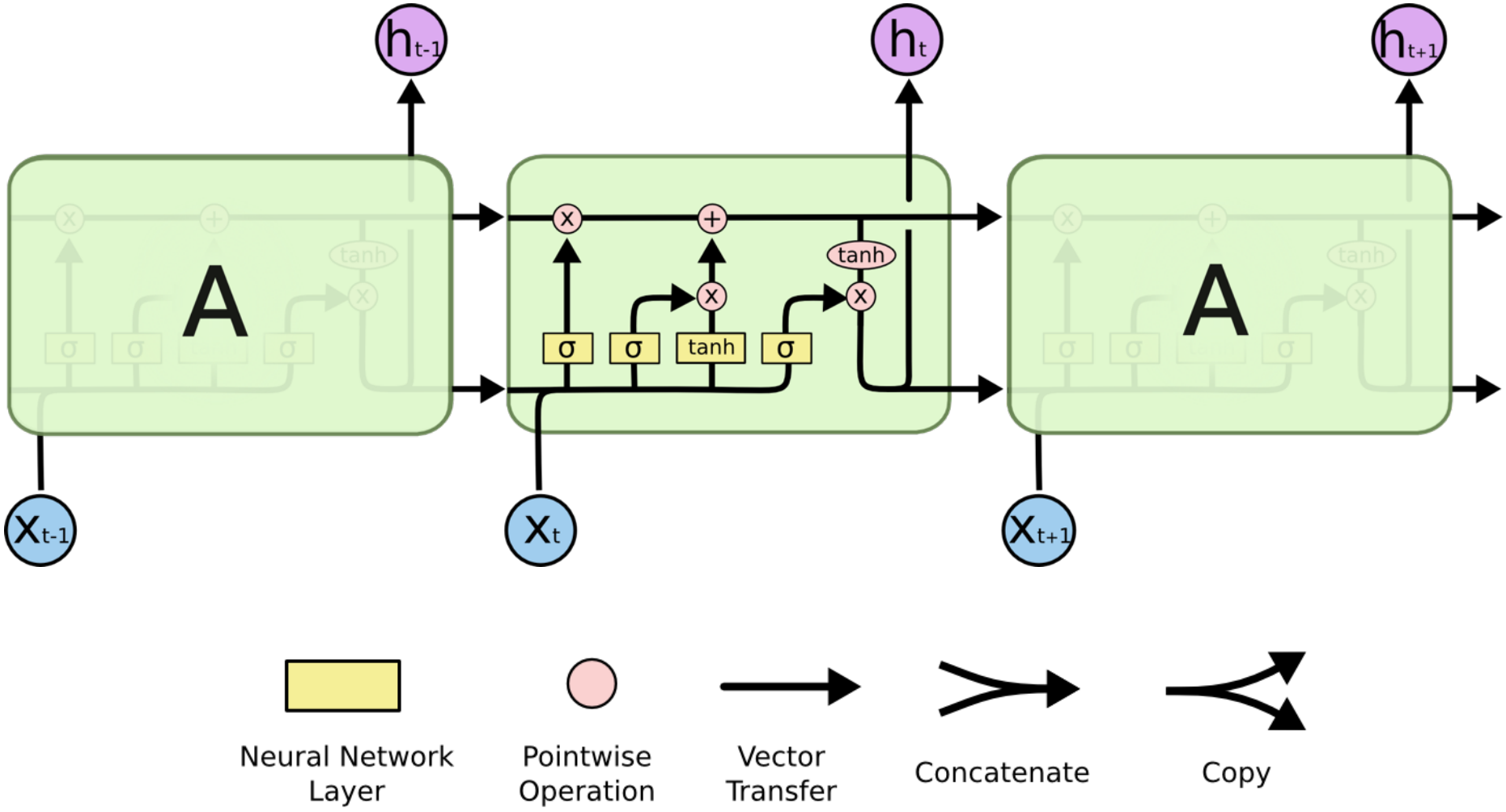
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN LSTM



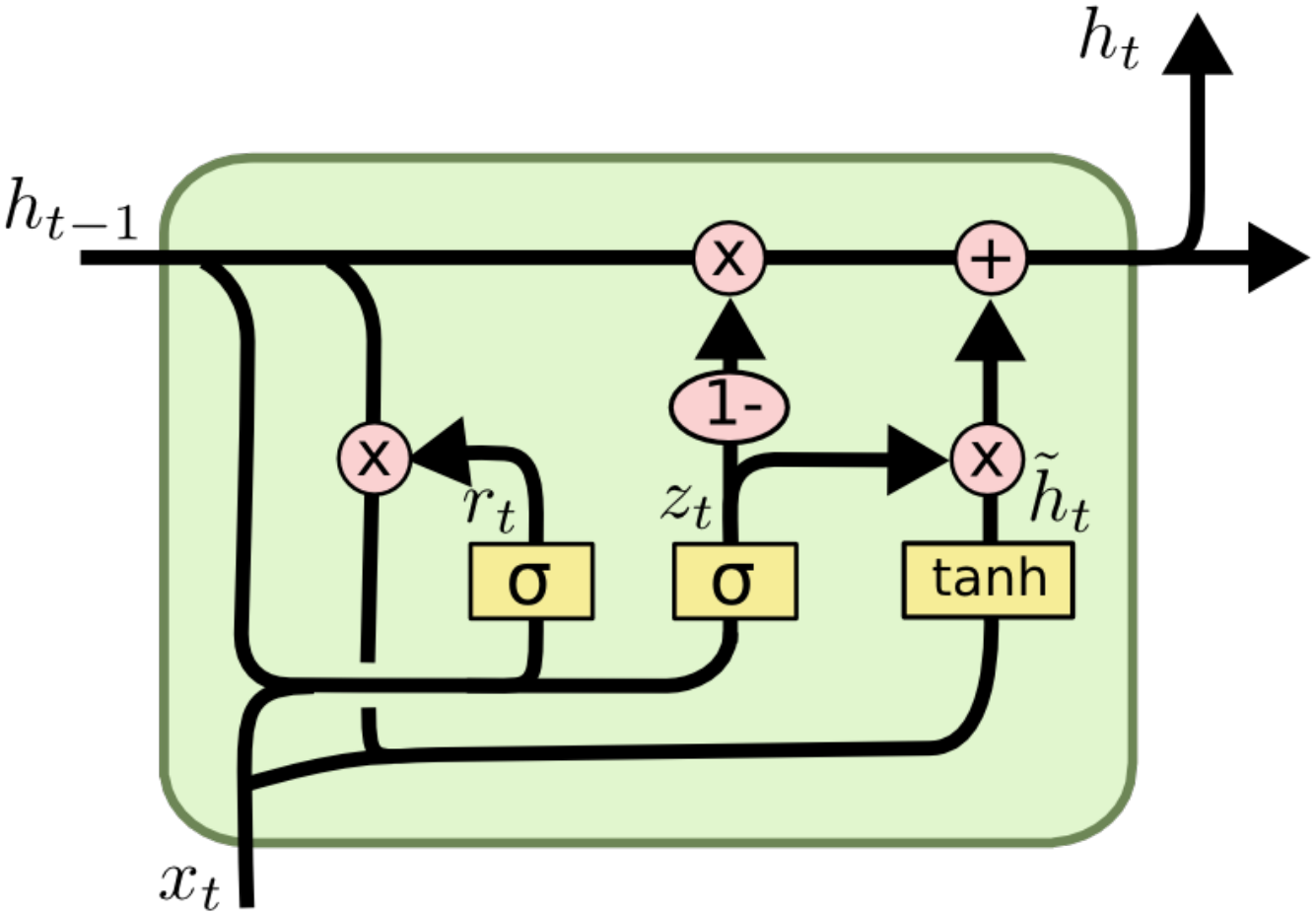
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



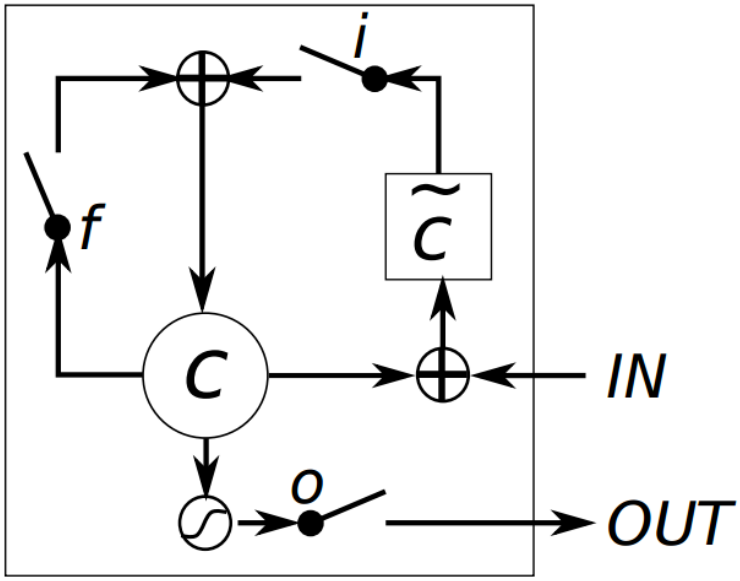
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

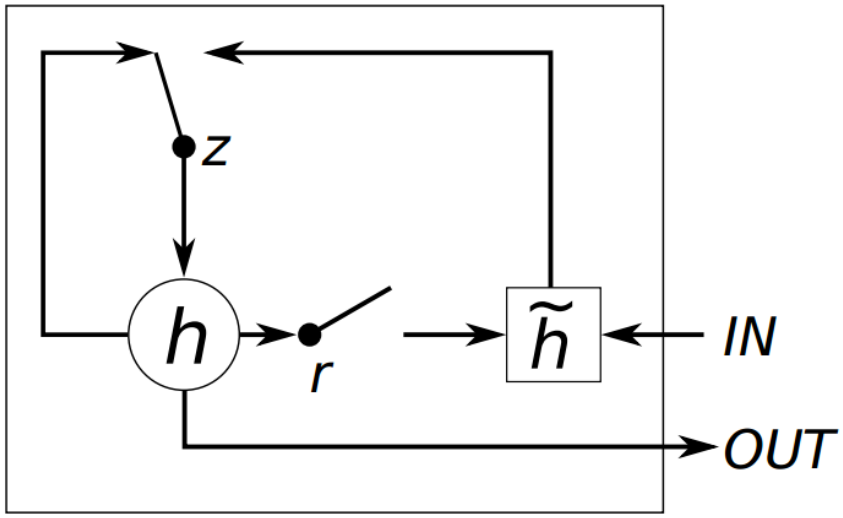
Gated Recurrent Unit (GRU)



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

LSTM vs GRU





LSTM

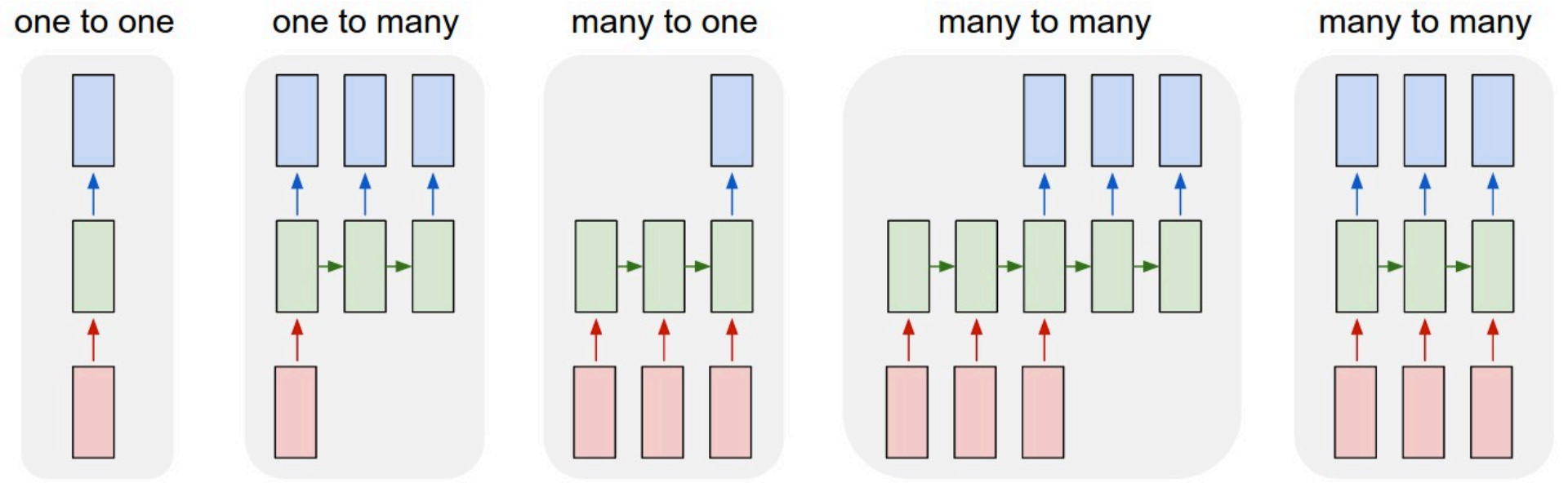
i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

GRU

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

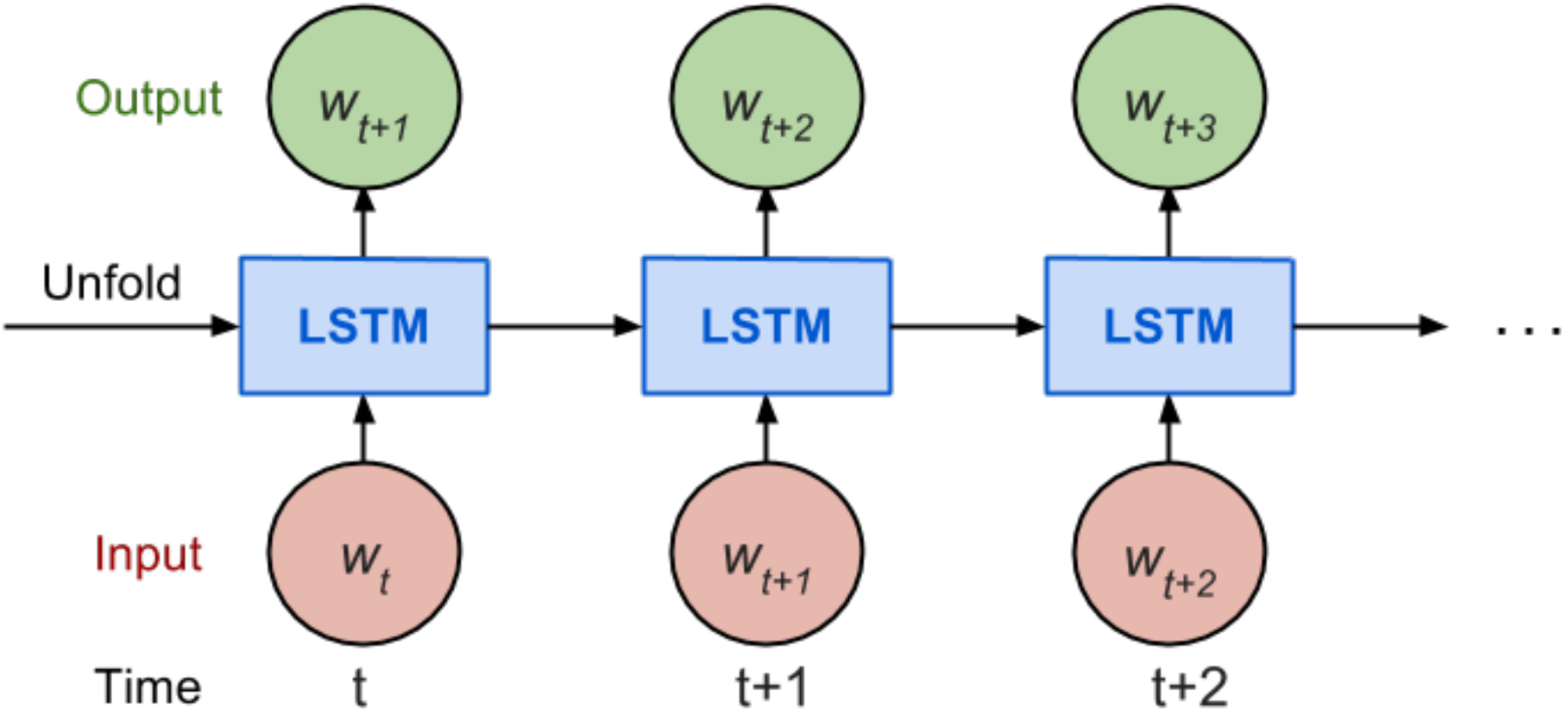
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

LSTM Recurrent Neural Network



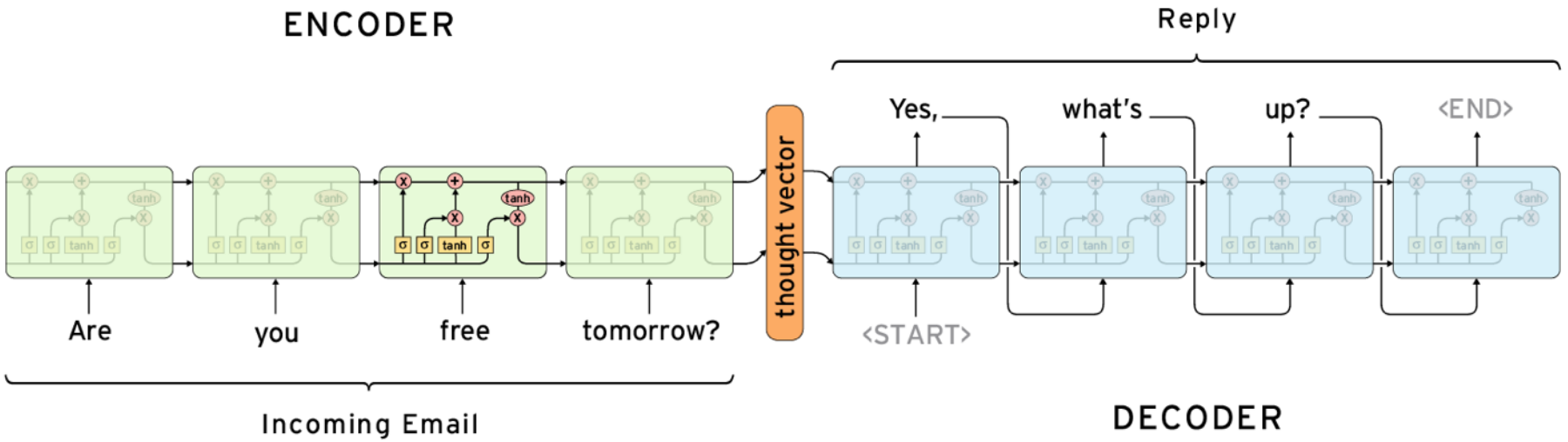
Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

Predict Stock Prices Using LSTM RNN



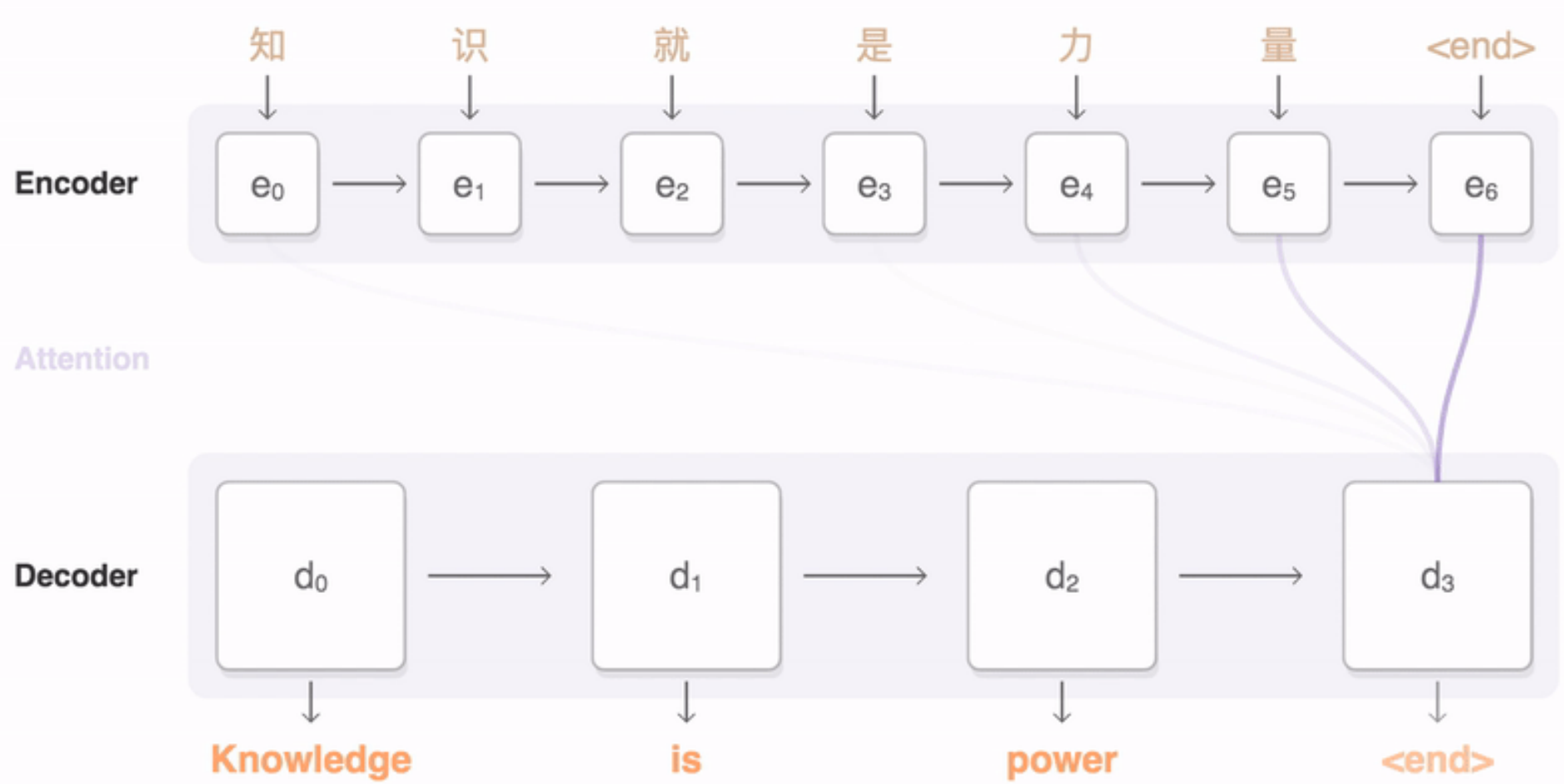
Source: https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html

The Sequence to Sequence model (seq2seq)



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

Sequence to Sequence (Seq2Seq)



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

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

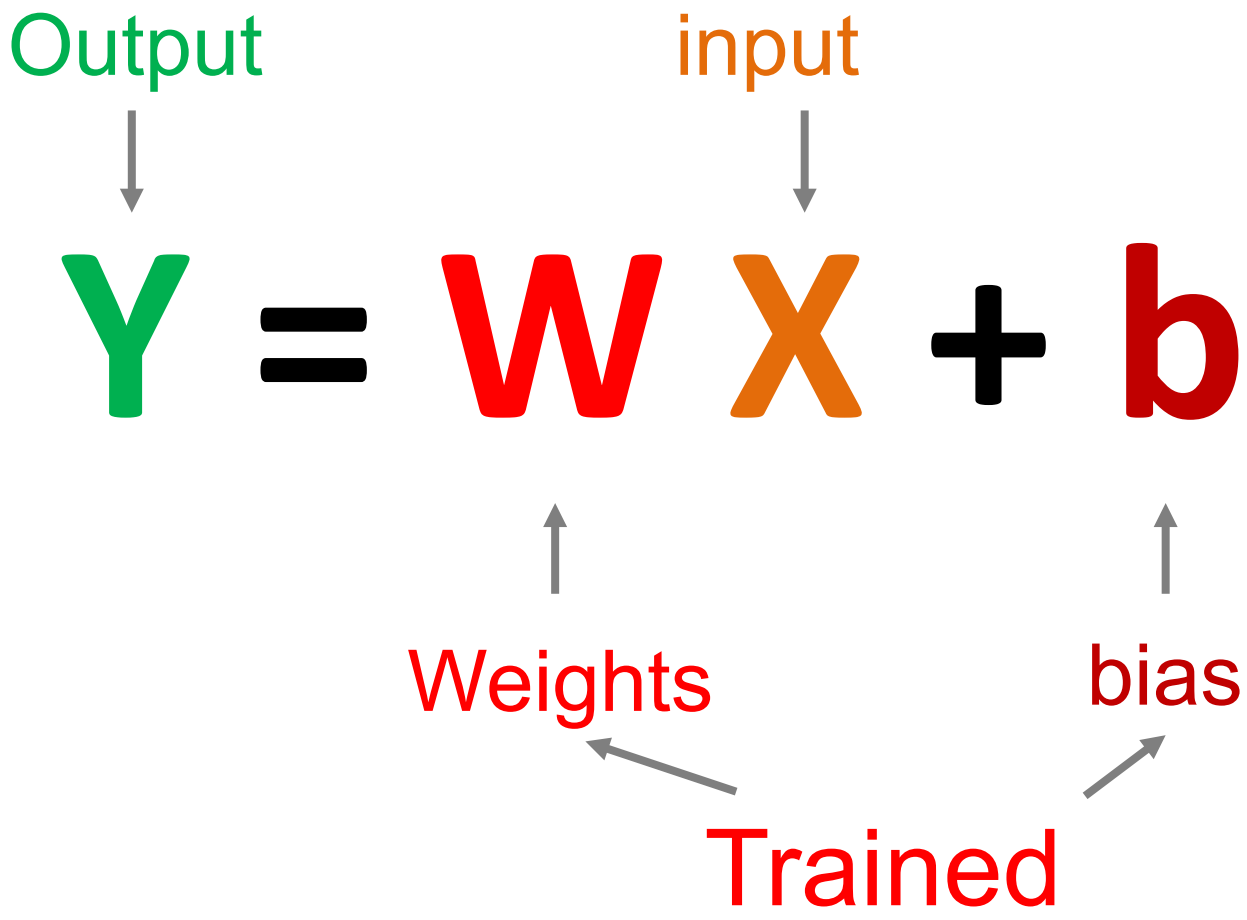
Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

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

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

$$Y = W X + b$$

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



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

Training a Network
=
Minimize the Cost Function

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

Training a Network

=

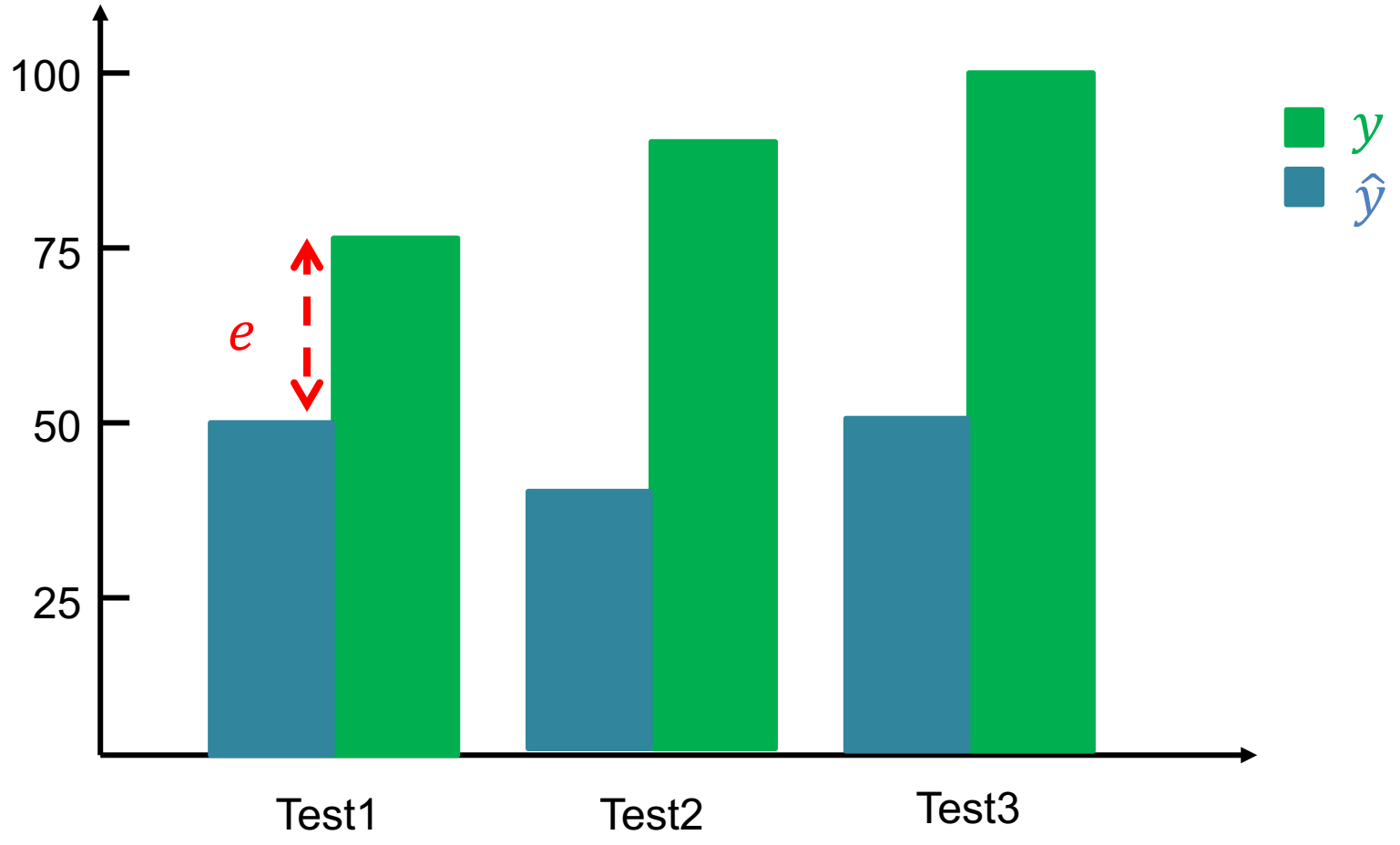
Minimize the Cost Function

Minimize the Loss Function

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

Error = Predict Y - Actual Y

Error : Cost : Loss



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

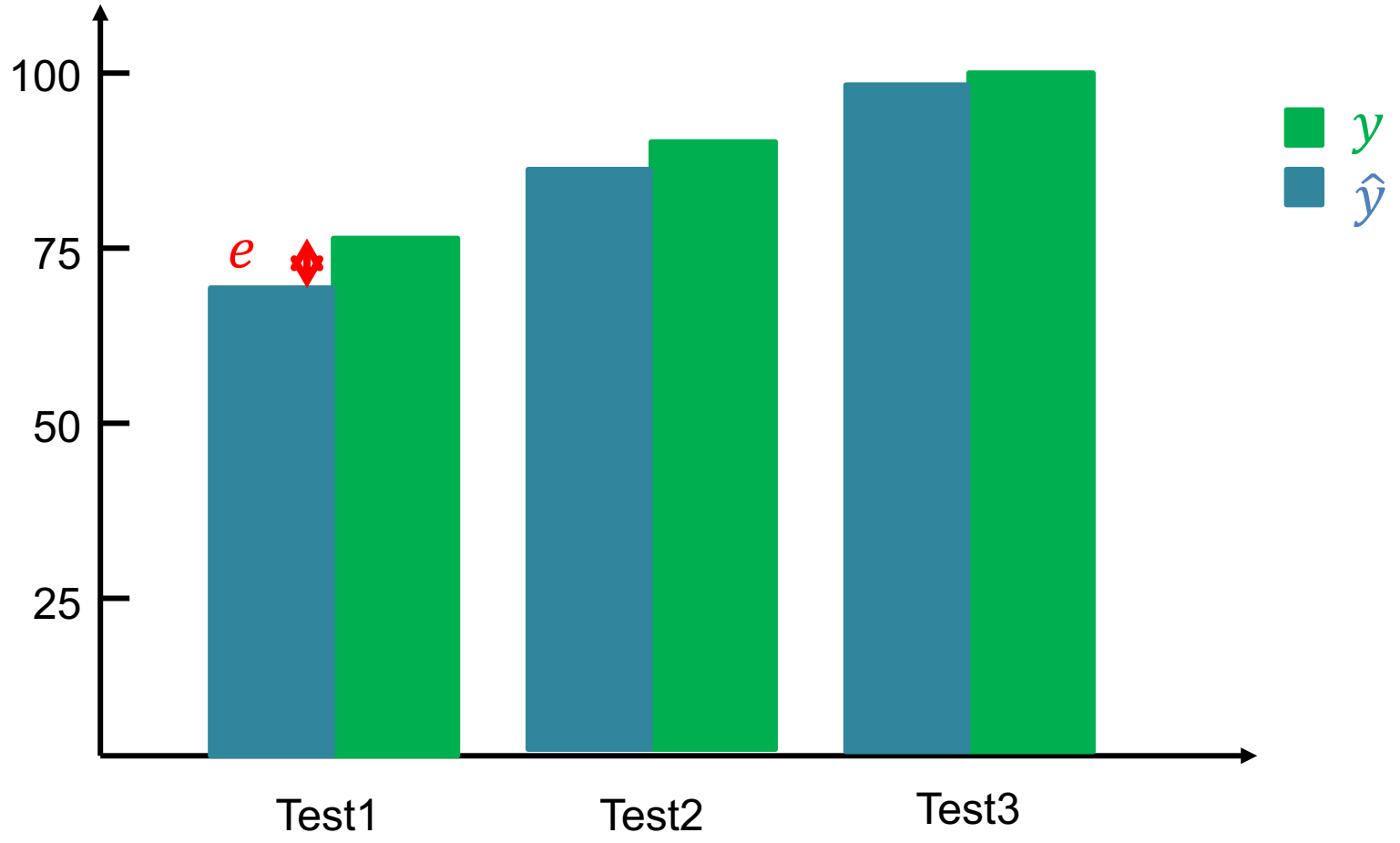
Error = Predict Y - Actual Y

Error : Cost : Loss

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

Error = Predict Y - Actual Y

Error : Cost : Loss

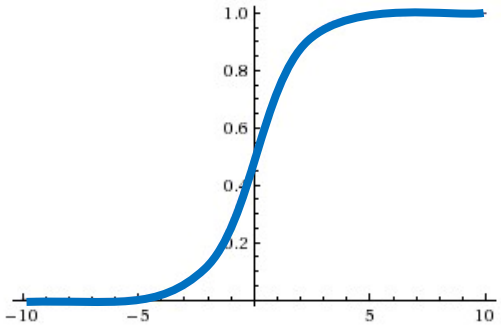


Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

Activation Functions

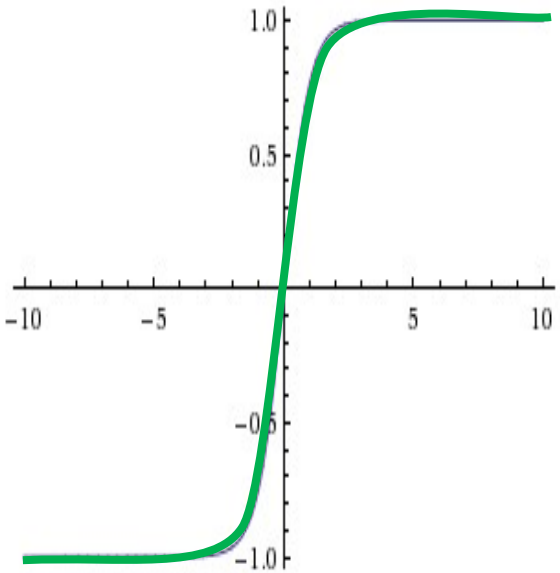
Activation Functions

Sigmoid



[0, 1]

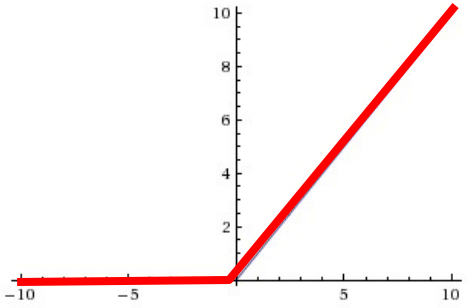
TanH



[-1, 1]

ReLU

(Rectified Linear Unit)

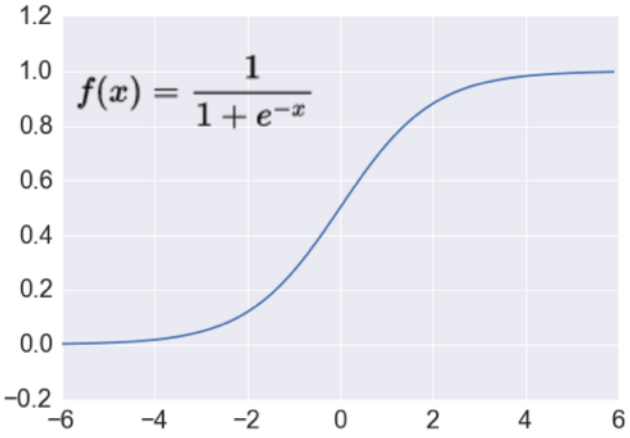


$f(x) = max(0, x)$

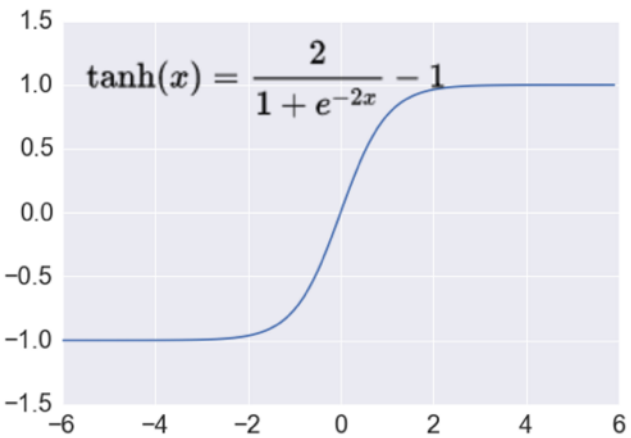
Source: http://cs231n.github.io/neural-networks-1/

Activation Functions

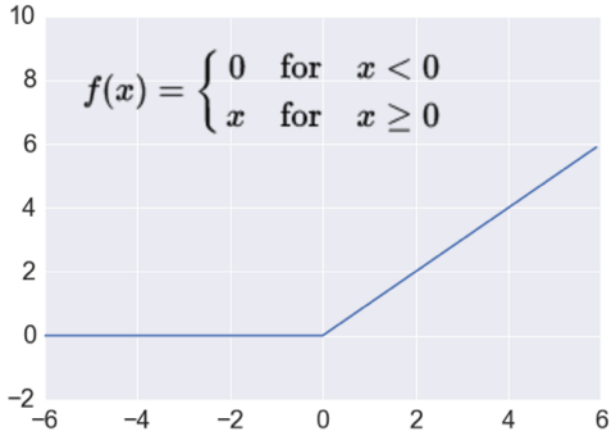
Sigmoid



TanH



ReLU



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

Loss Function

Binary Classification: 2 Class

Activation Function: Sigmoid

Loss Function: Binary Cross-Entropy

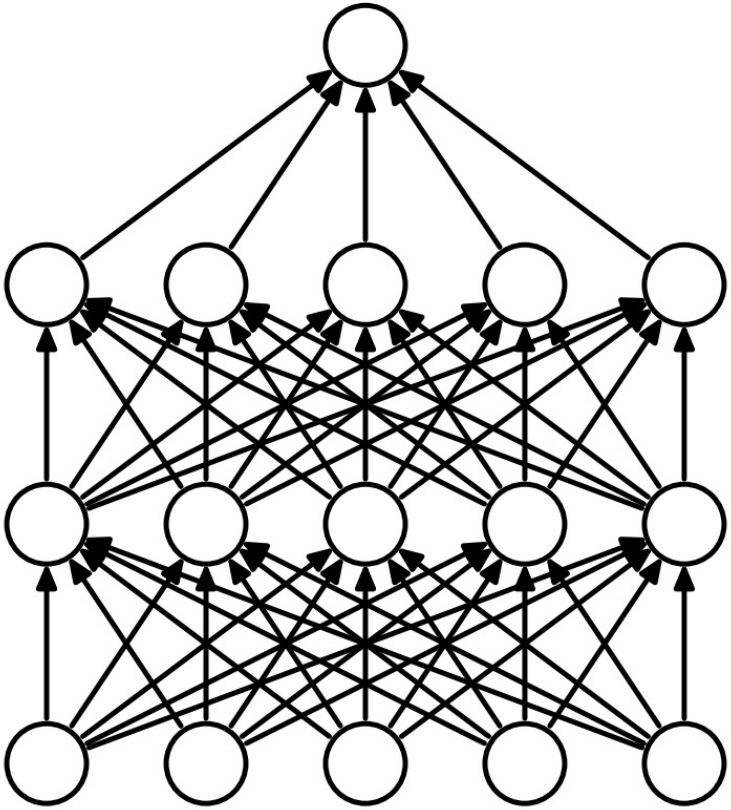
Multiple Classification: 10 Class

Activation Function:
SoftMAX

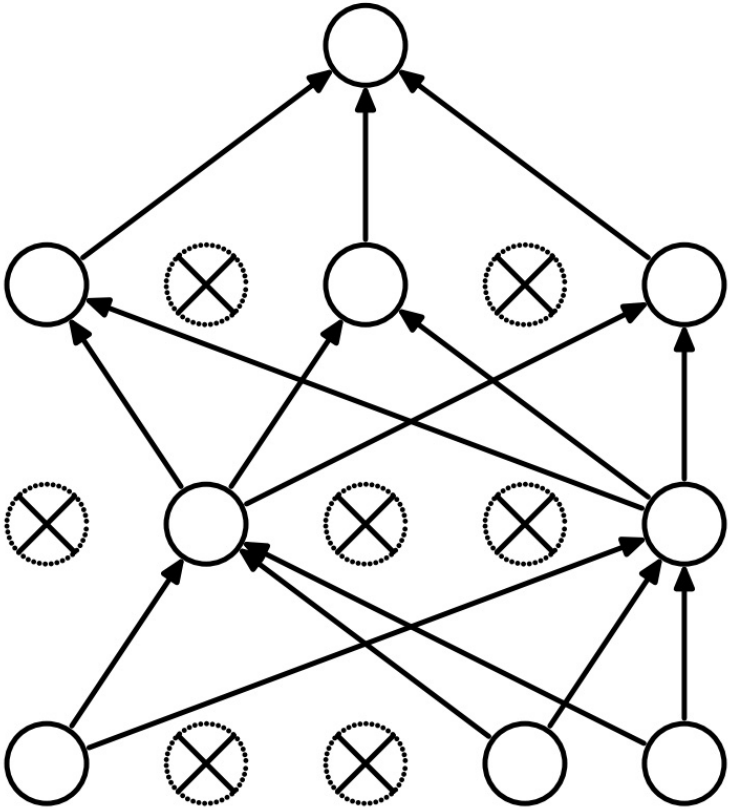
Loss Function:
Categorical Cross-Entropy

Dropout

Dropout: a simple way to prevent neural networks from overfitting



(a) Standard Neural Net

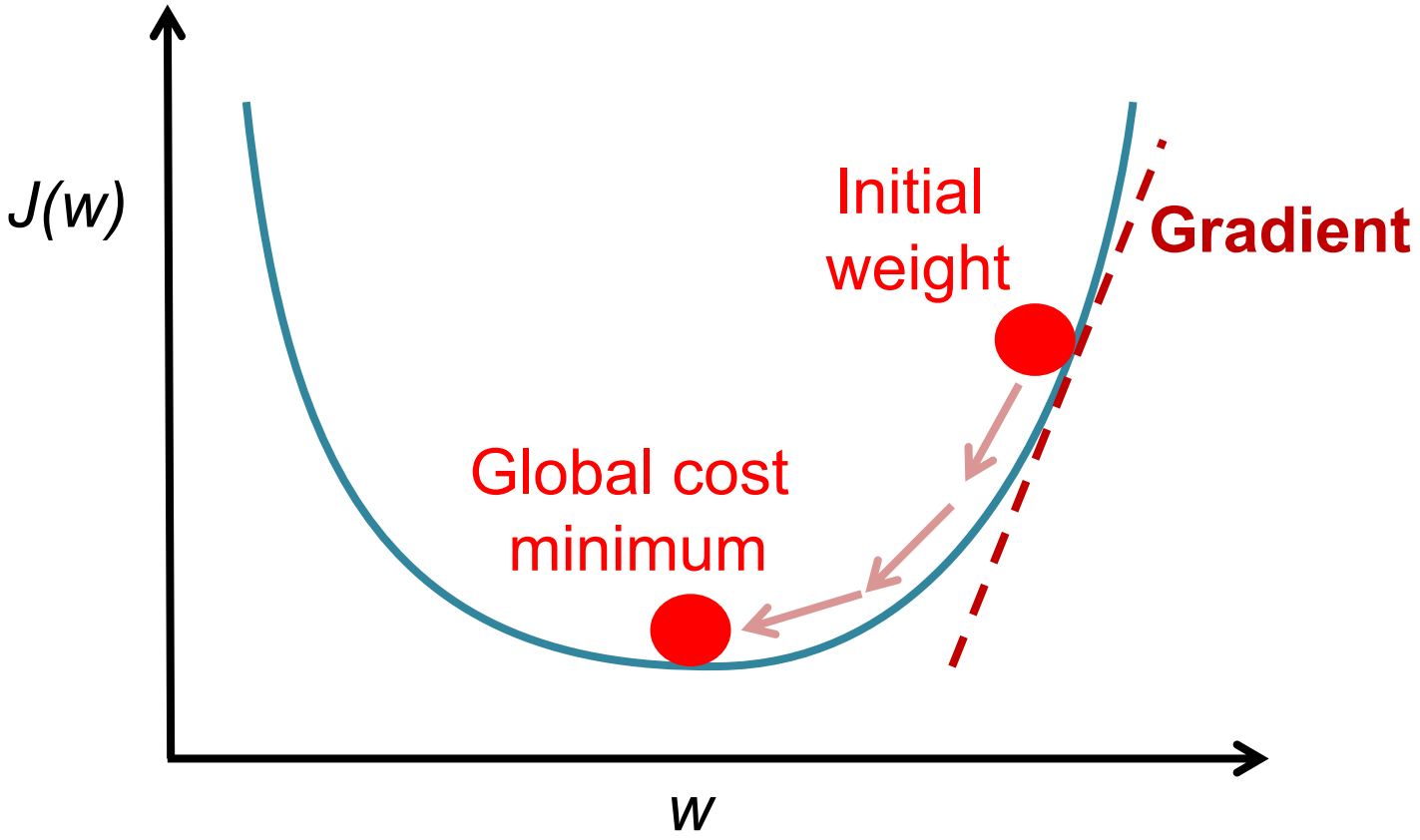


(b) After applying dropout.

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.

Optimizer:

Stochastic Gradient Descent (SGD)



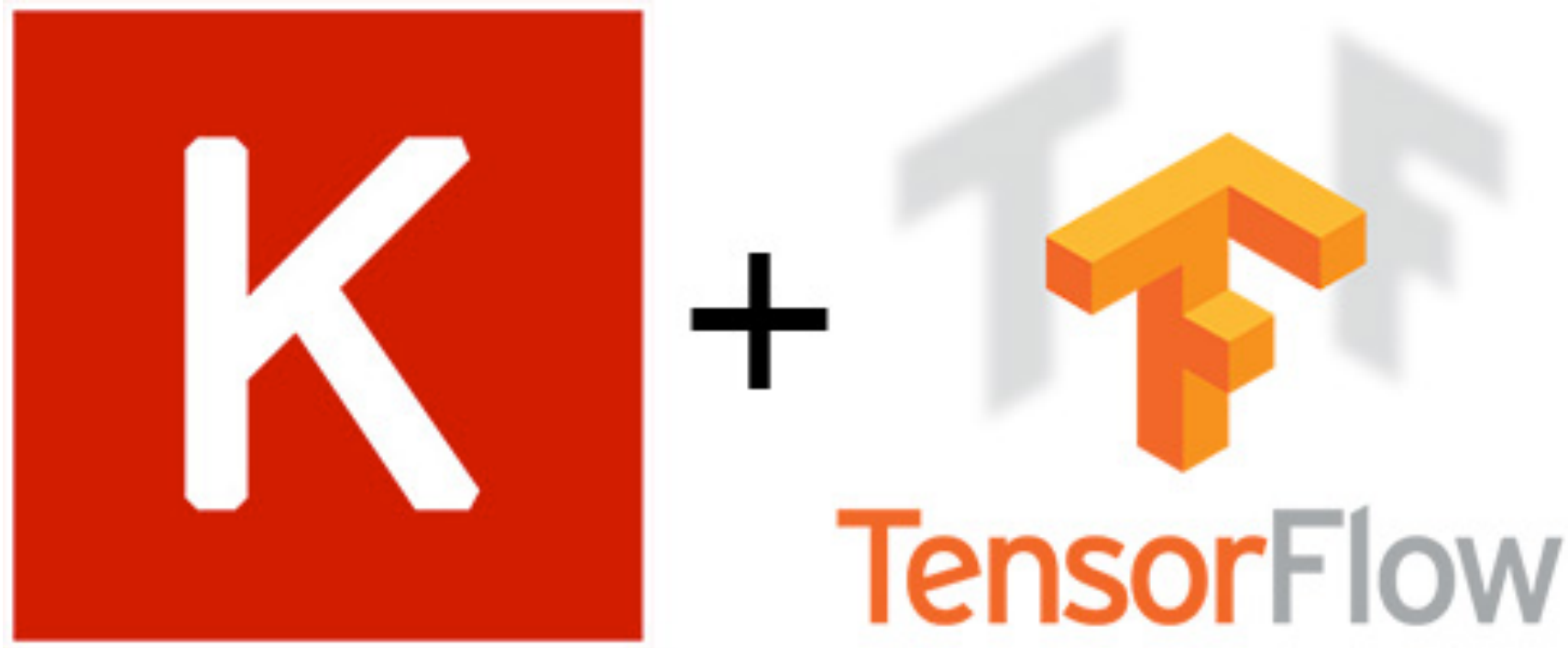
Learning Algorithm

While not done:

- Pick a random training example “(input, label)”
- Run neural network on “input”
- Adjust weights on edges to make output closer to “label”

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

Deep Learning
Keras + TensorFlow



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

```
source activate tensorflow
```

```
jupyter notebook
```

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

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

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

```
1.4.0
```

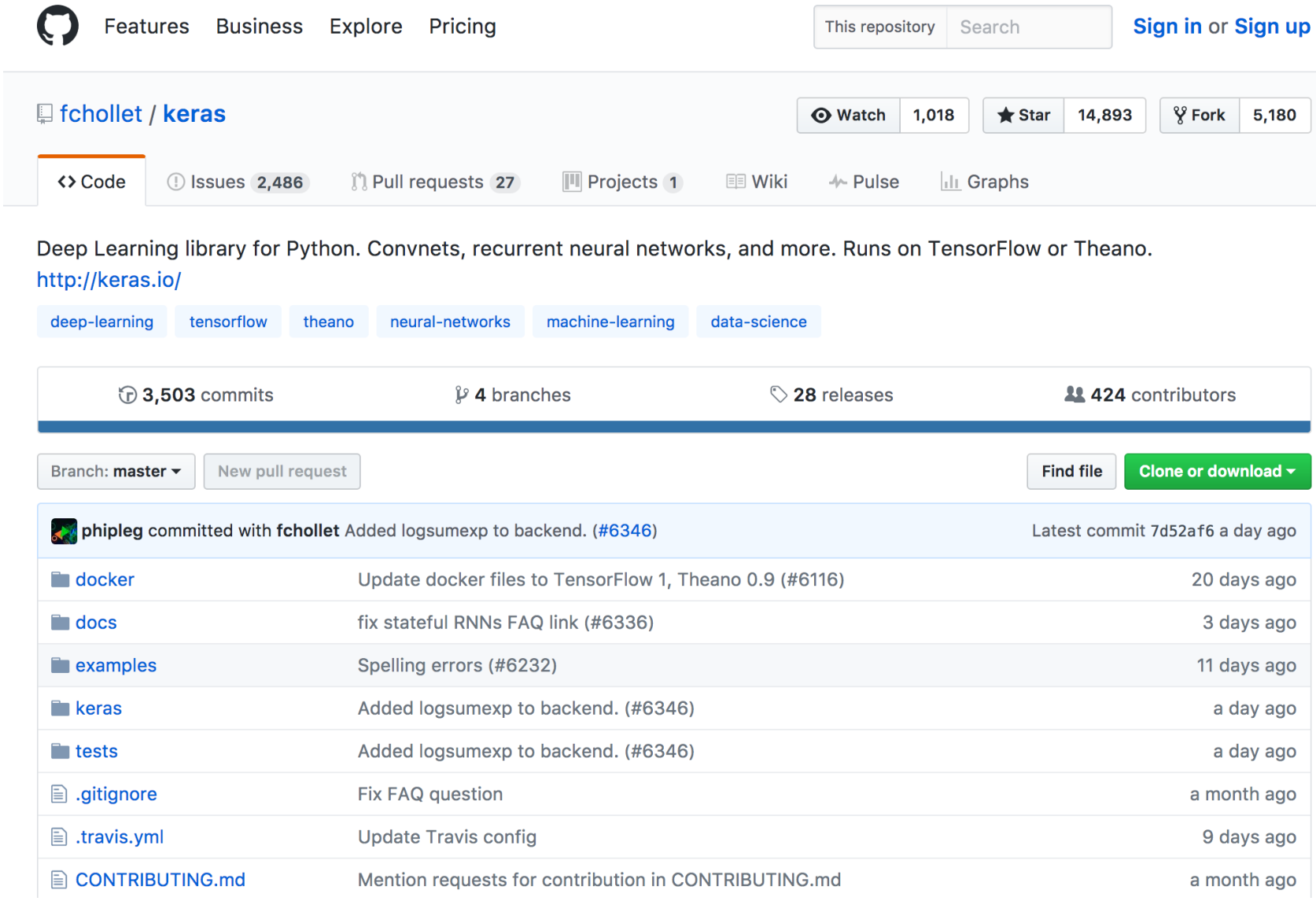
```
import keras
```

```
Using TensorFlow backend.
```

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

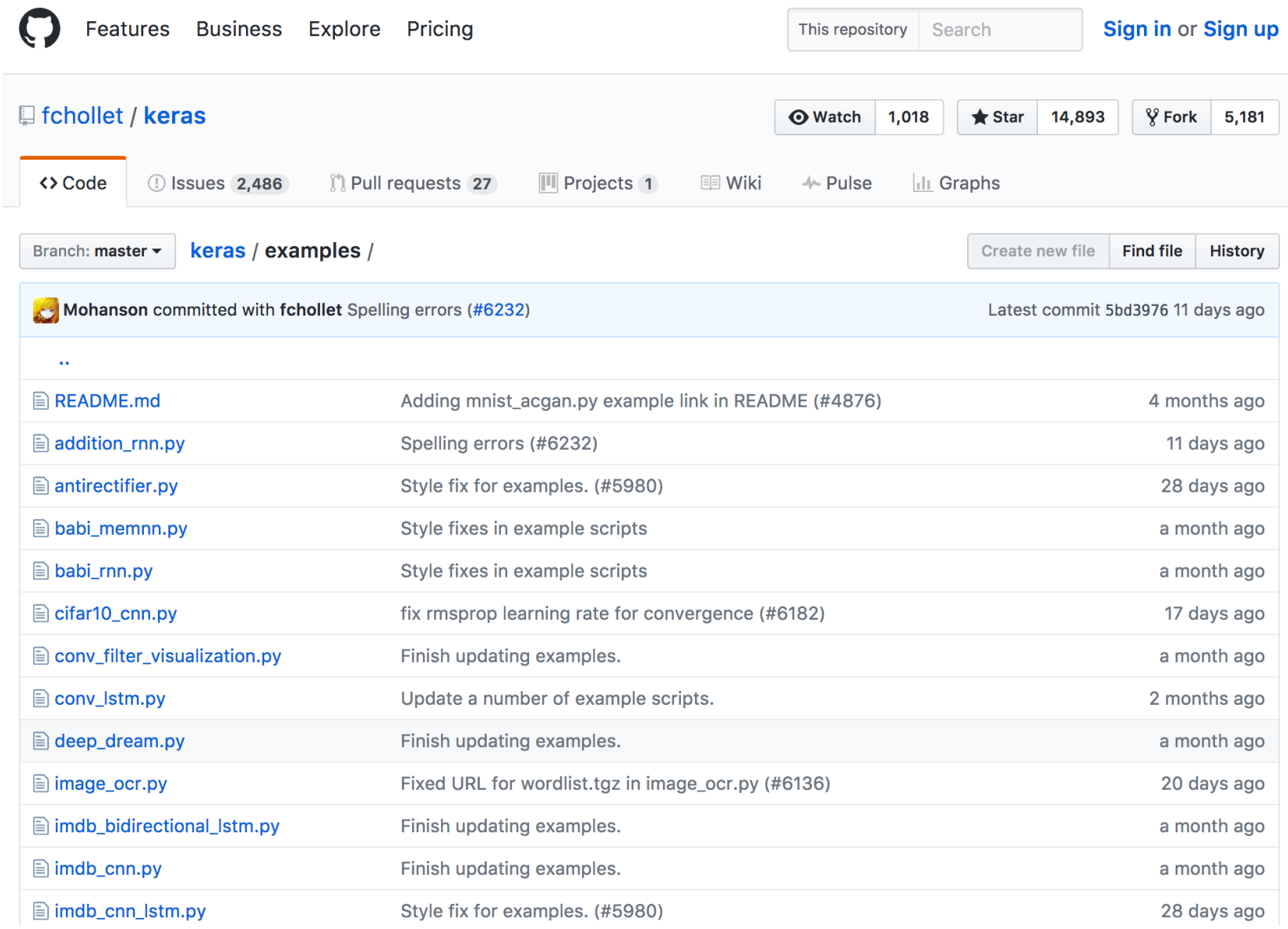
```
2.1.2
```

Keras Github



https://github.com/fchollet/keras

Keras Examples



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

Keras Examples

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

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

Keras MINST CNN

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

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

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

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

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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

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

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

Keras MINST CNN

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

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

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

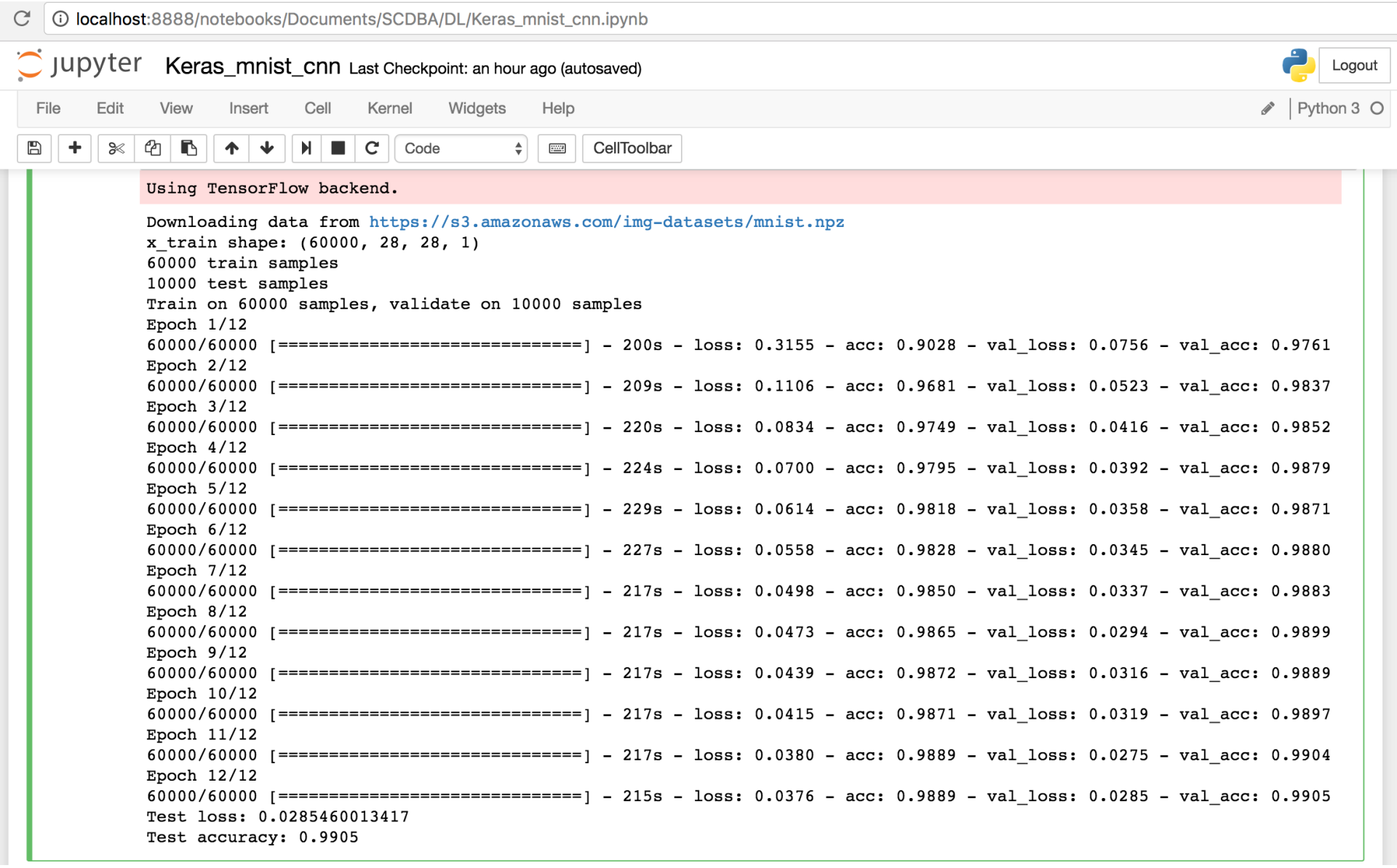
```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

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

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

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

```
Using TensorFlow backend.
```

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

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

Keras MINST CNN



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

Keras MINST CNN

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

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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

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

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

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

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

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

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

Keras MINST CNN

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

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

Keras MINST CNN

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

# input image dimensions
img_rows, img_cols = 28, 28

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

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

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

Keras MINST CNN

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

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

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

Keras MINST CNN

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

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

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

Keras MINST CNN

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

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

Keras MINST CNN

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

IMDB
Large Movie Review Dataset

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

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

IMDB Dataset (Mass et al., 2011)

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

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

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

Keras IMDB Movie reviews sentiment classification

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

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

Keras IMDB load_data

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

Source: keras/datasets/imdb.py

Keras IMDB get_word_index

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

Source: keras/datasets/imdb.py

Keras IMDB CNN

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

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

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

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

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

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

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

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

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

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

Keras IMDB CNN

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

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

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

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

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

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

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

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

```
Using TensorFlow backend.
```

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

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

Keras IMDB CNN

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

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

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

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

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

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

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

```
Using TensorFlow backend.
```

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

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

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

Keras IMDB CNN

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

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

Keras IMDB LSTM

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

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

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

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

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

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

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

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

Keras IMDB LSTM

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

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

Keras IMDB LSTM

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

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

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

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

Keras IMDB LSTM

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

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

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

Keras IMDB LSTM

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

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

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

Keras IMDB LSTM

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

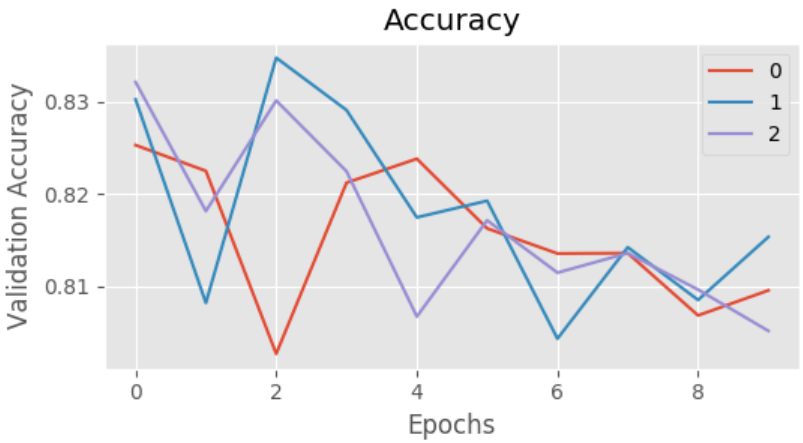
Keras IMDB FastText

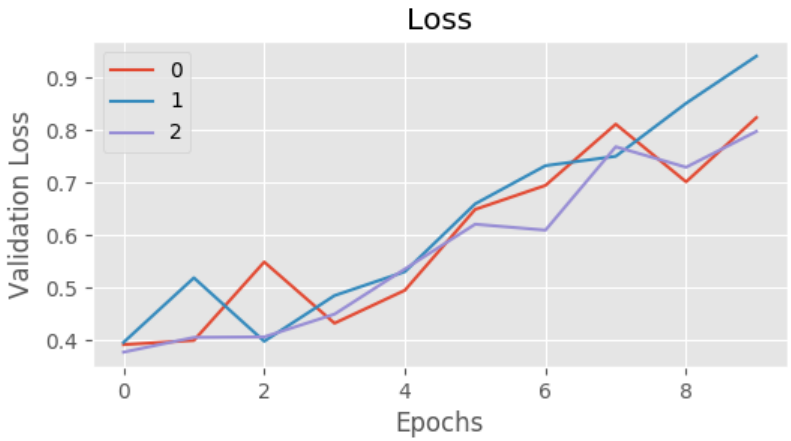
```
python imdb_fasttext.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Average train sequence length: 238
Average test sequence length: 230
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples,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

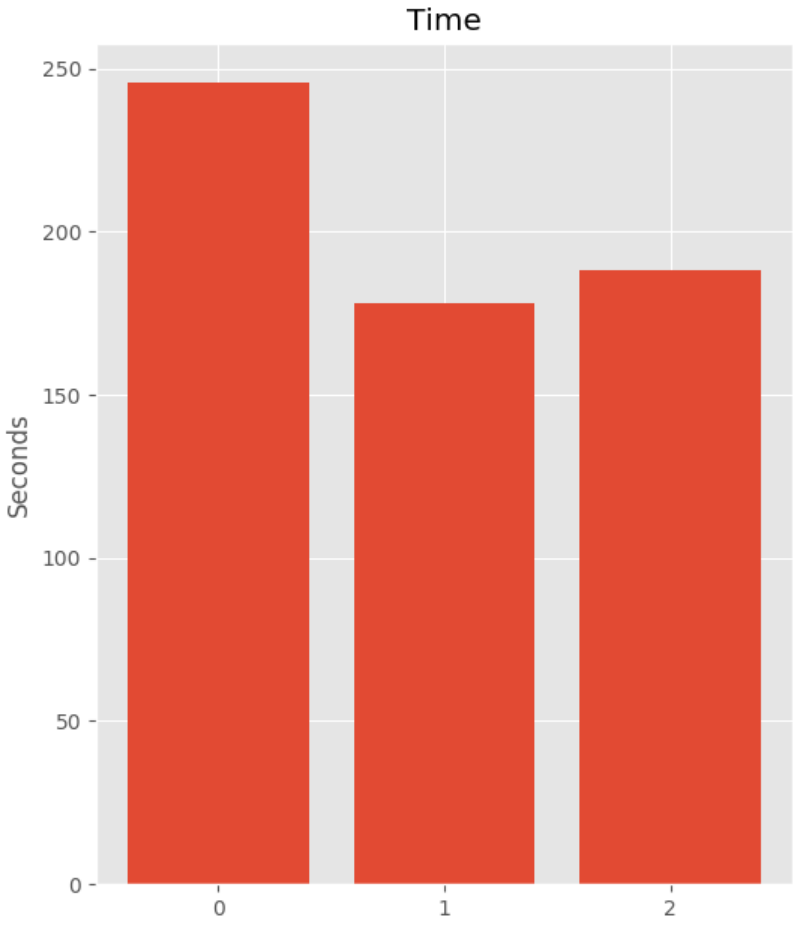
Keras IMDB CNN LSTM

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

Keras LSTM Benchmark







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

imdb_lstm_2.py

```
from __future__ import print_function

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

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

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

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

imdb_lstm_2.py

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

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

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

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

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

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

imdb_lstm_2.py

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

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

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

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

imdb_lstm_2.py

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

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

imdb_lstm_2.py

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

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

imdb_lstm_2.py

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

imdb_lstm_2.py

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

python filename.py

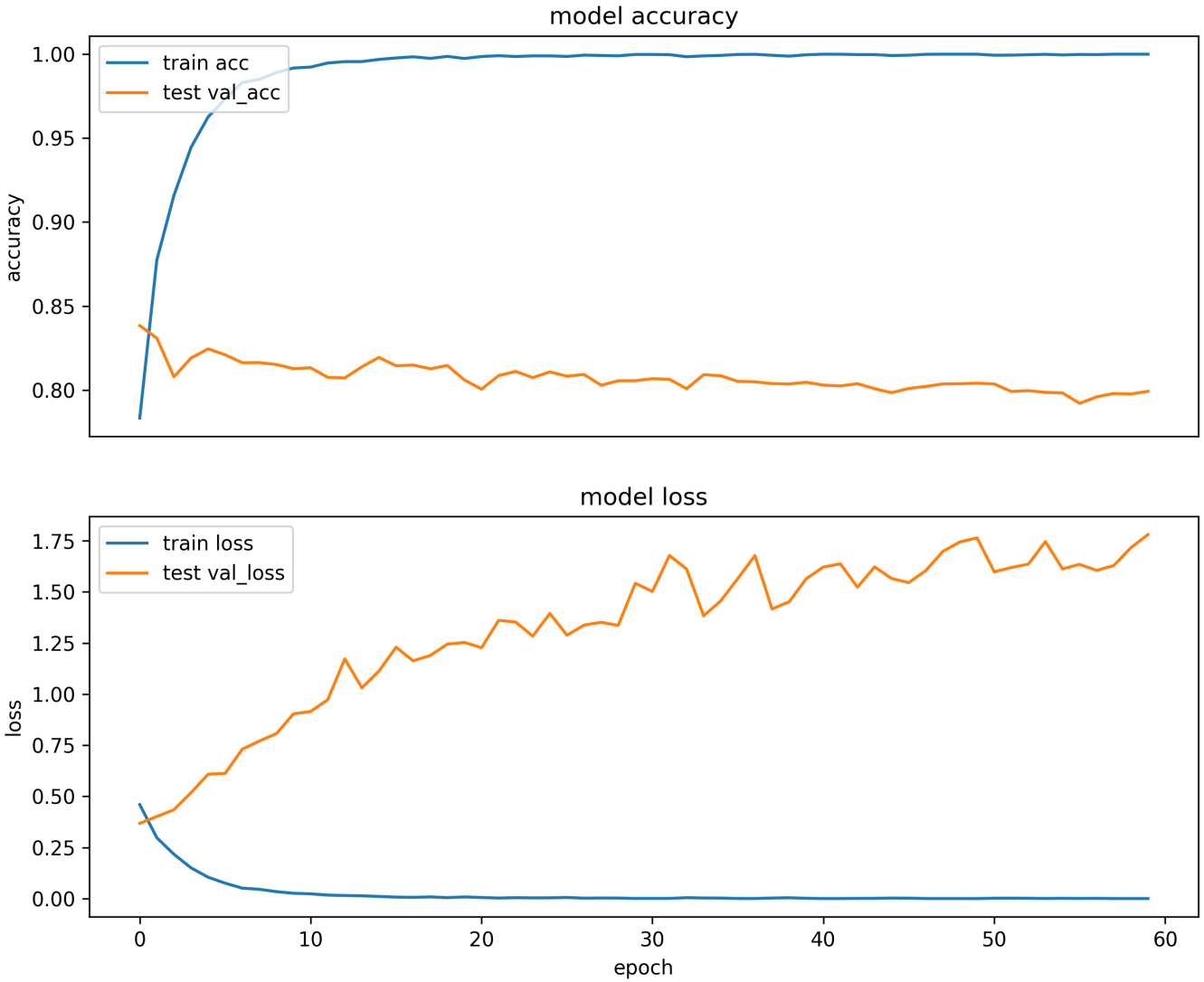
```
python imdb_fasttext_2.py
python imdb_cnn_2.py
python imdb_lstm_2.py
python imdb_cnn_lstm_2.py
python imdb_bidirectional_lstm_2.py
```

Deep Learning Summary

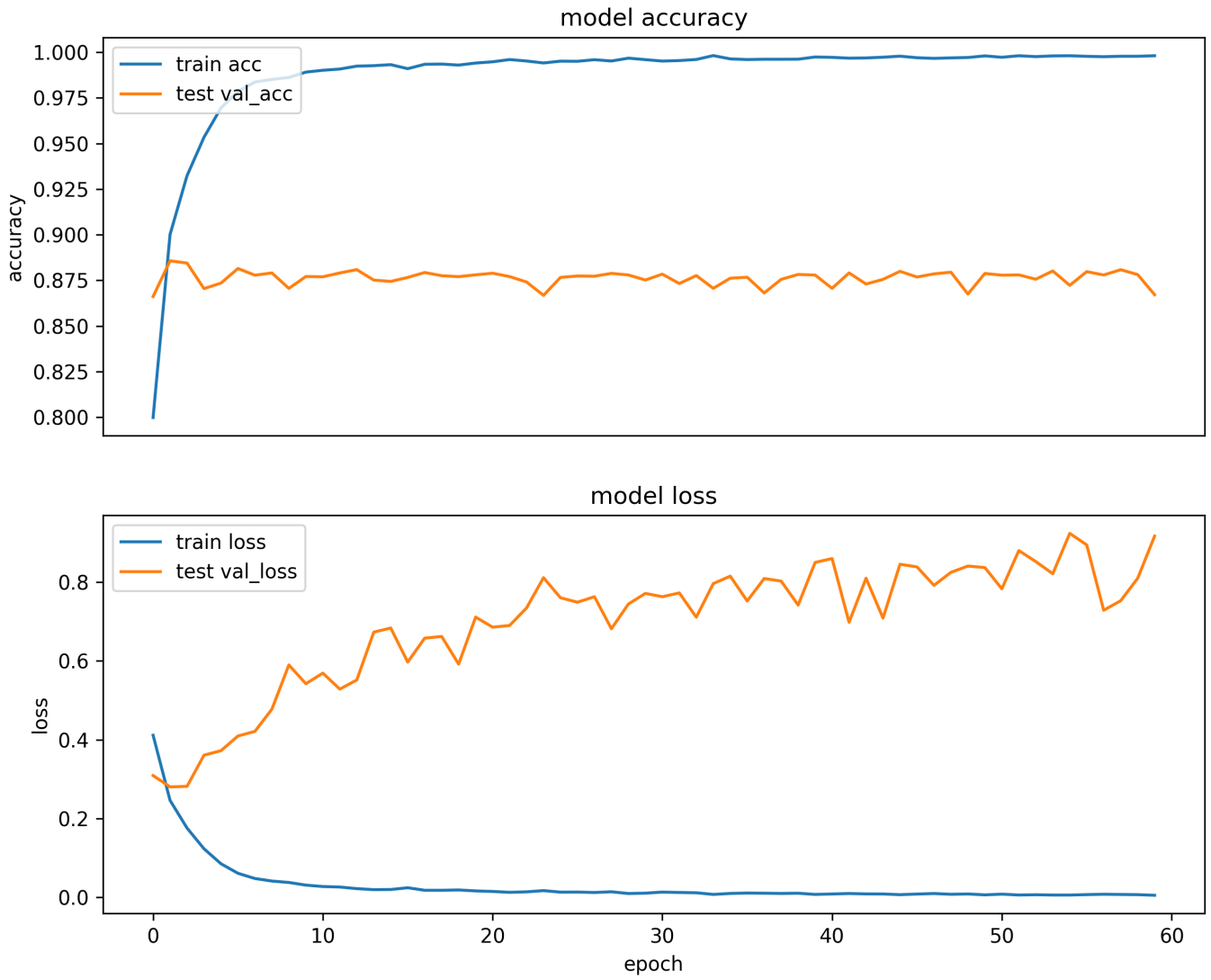
```
#Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\t'  + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t ' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))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

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

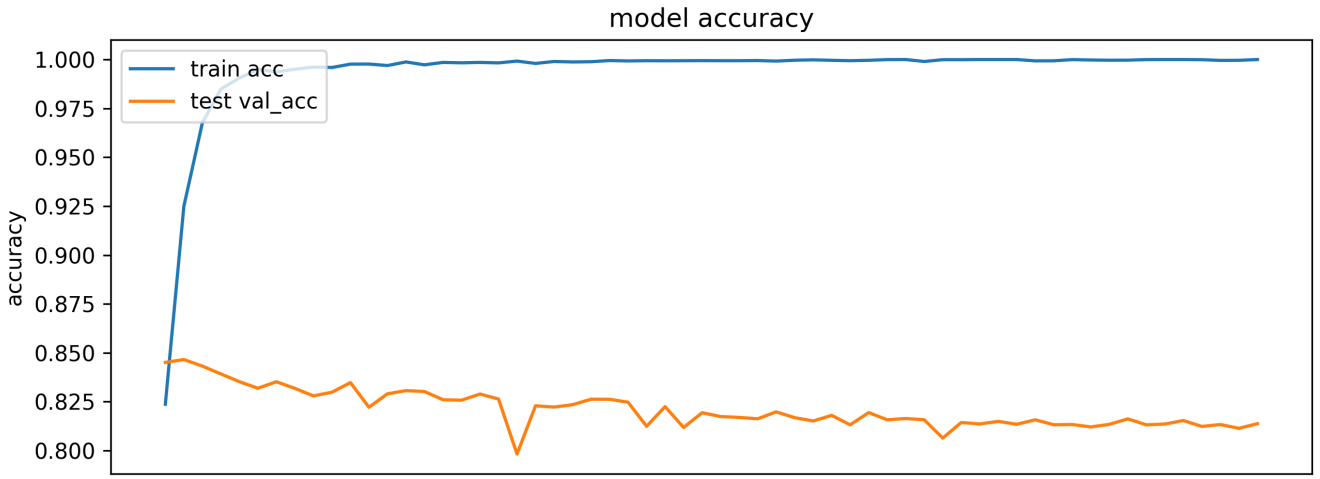
imdb_lstm_2.py

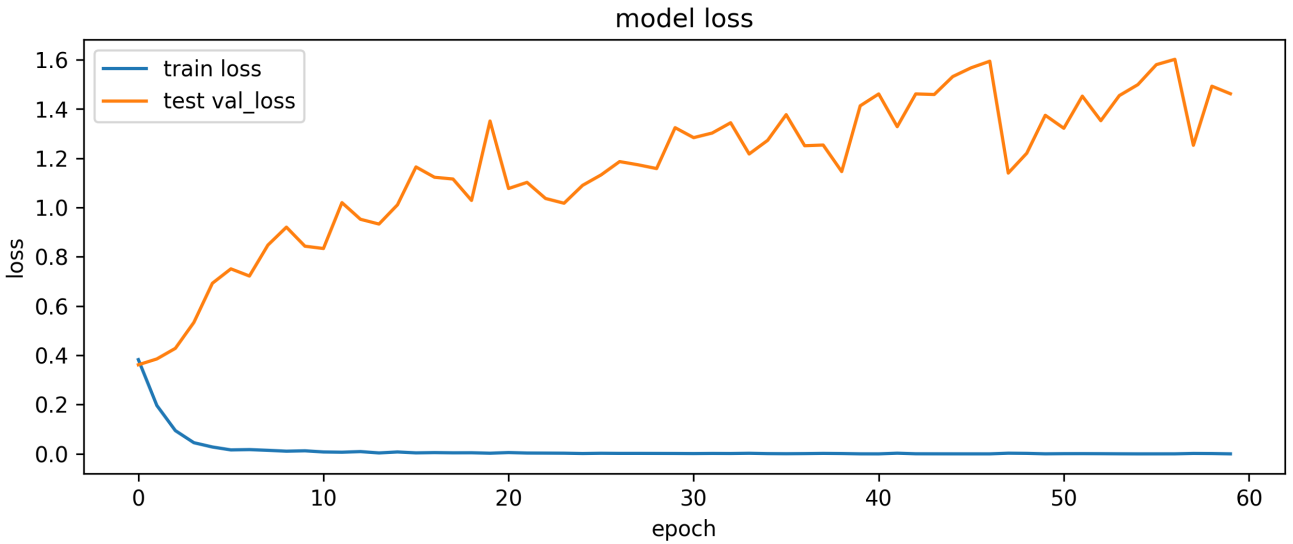


imdb_cnn_2.py

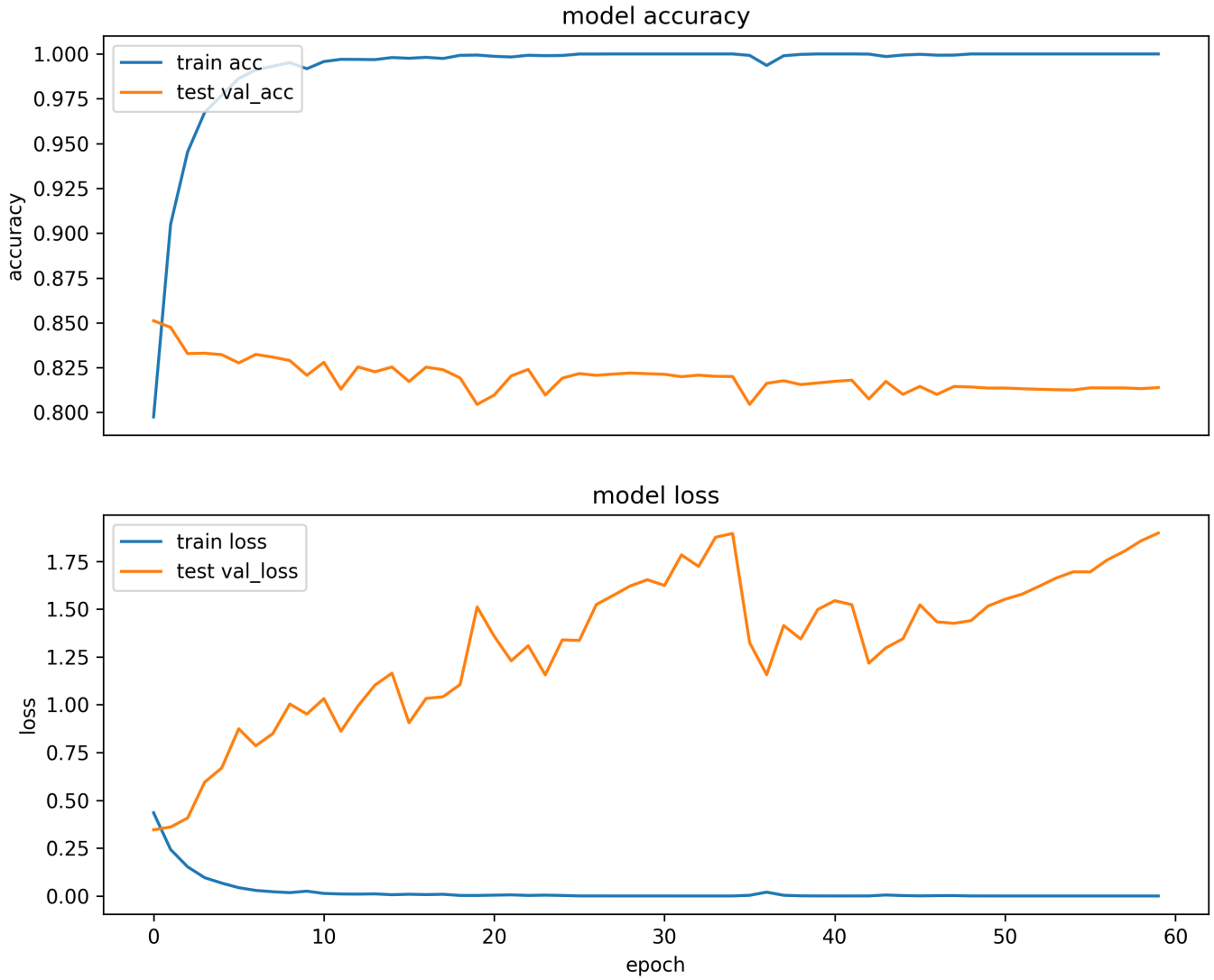

model accuracy
train acc
test val_acc
accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
model loss
train loss
test val_loss
loss
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
epoch

imdb_cnn_lstm_2.py

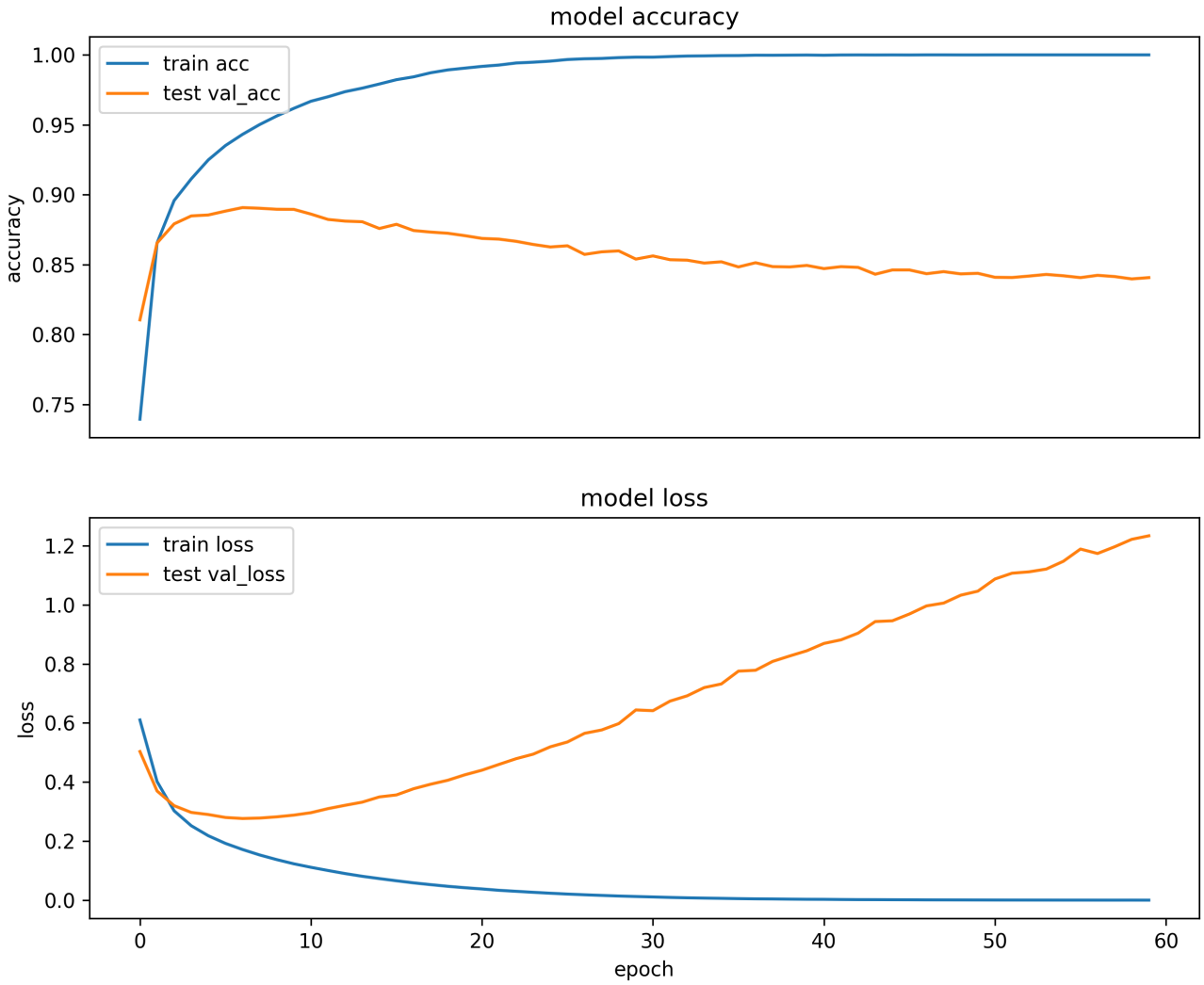




imdb_bidirectional_lstm_2.py



imdb_fasttext_2.py



Deep Learning with CPU vs. GPU

Timings:

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

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

Deep Learning for Financial Market Prediction

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

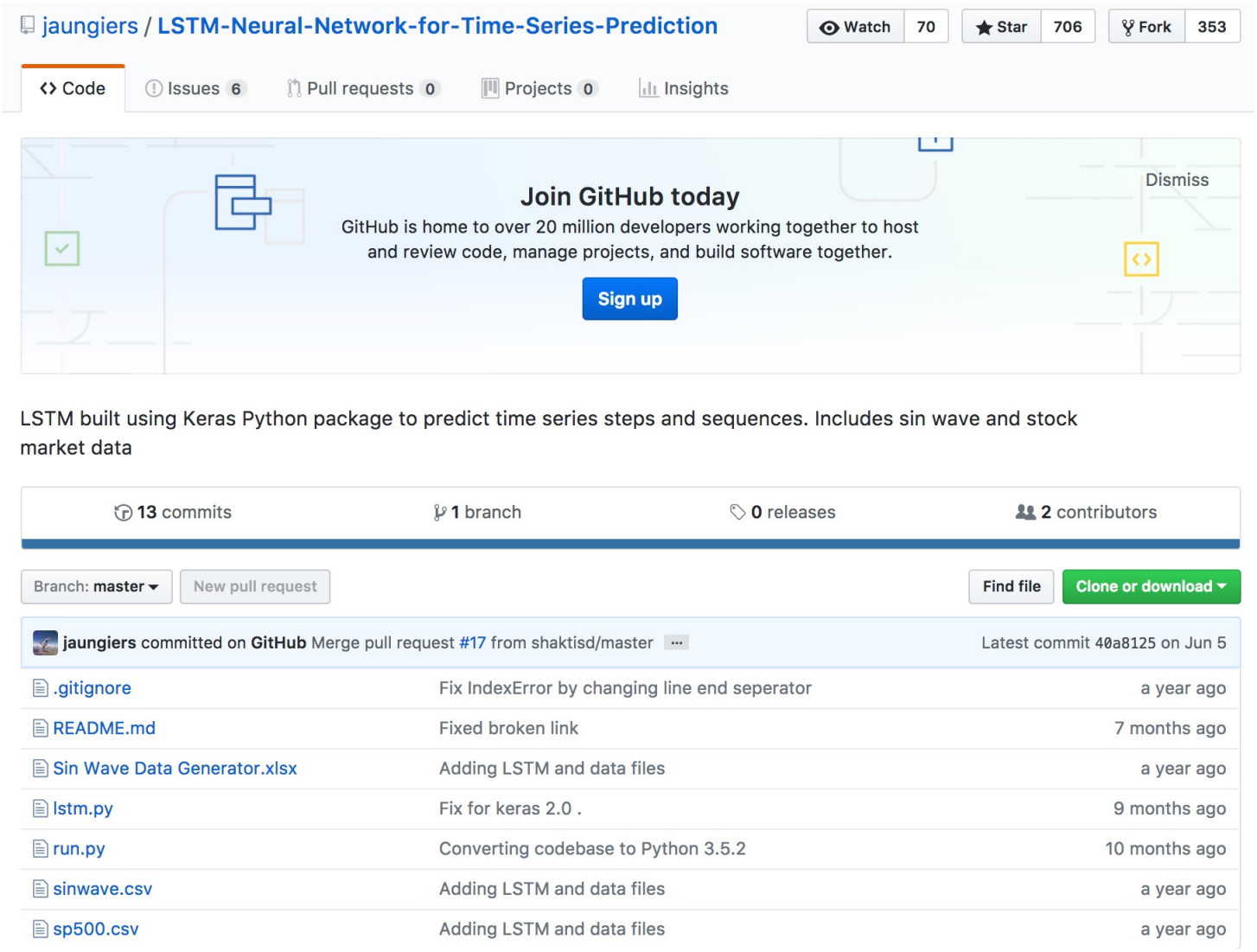
Time Series Data

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

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

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

LSTM Neural Network for Time Series Prediction



jaungiers / LSTM-Neural-Network-for-Time-Series-Prediction

Watch 70 | Star 706 | Fork 353

Code | Issues 6 | Pull requests 0 | Projects 0 | Insights

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

LSTM built using Keras Python package to predict time series steps and sequences. Includes sin wave and stock market data

13 commits | 1 branch | 0 releases | 2 contributors

Branch: master | New pull request | Find file | Clone or download

jaungiers committed on GitHub Merge pull request #17 from shaktisd/master ... Latest commit 40a8125 on Jun 5

.gitignore	Fix IndexError by changing line end seperator	a year ago
README.md	Fixed broken link	7 months ago
Sin Wave Data Generator.xlsx	Adding LSTM and data files	a year ago
lstm.py	Fix for keras 2.0 .	9 months ago
run.py	Converting codebase to Python 3.5.2	10 months ago
sinwave.csv	Adding LSTM and data files	a year ago
sp500.csv	Adding LSTM and data files	a year ago

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

LSTM Neural Network for Time Series Prediction

```
from
keras.layers.recurrent
import LSTM
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

LSTM Neural Network for Time Series Prediction

```
import os
import time
import warnings
import numpy as np
from numpy import newaxis
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
import matplotlib.pyplot as plt

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' #Hide TensorFlow warnings
warnings.filterwarnings("ignore") #Hide Numpy warnings
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

load_data

```
def load_data(filename, seq_len, normalise_window):
    f = open(filename, 'rb').read()
    data = f.decode().split('\n')

    sequence_length = seq_len + 1
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index: index + sequence_length])

    if normalise_window:
        result = normalise_windows(result)

    result = np.array(result)

    row = round(0.9 * result.shape[0])
    train = result[:int(row), :]
    np.random.shuffle(train)
    x_train = train[:, :-1]
    y_train = train[:, -1]
    x_test = result[int(row):, :-1]
    y_test = result[int(row):, -1]

    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

    return [x_train, y_train, x_test, y_test]
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

normalise_windows

```
def normalise_windows(window_data):
    normalised_data = []
    for window in window_data:
        normalised_window = [((float(p) / float(window[0])) - 1) for p in window]
        normalised_data.append(normalised_window)
    return normalised_data
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

build_model

```
def build_model(layers):
    model = Sequential()

    model.add(LSTM(
        input_shape=(layers[1], layers[0]),
        output_dim=layers[1],
        return_sequences=True))
    model.add(Dropout(0.2))

    model.add(LSTM(
        layers[2],
        return_sequences=False))
    model.add(Dropout(0.2))

    model.add(Dense(
        output_dim=layers[3]))
    model.add(Activation("linear"))

    start = time.time()
    model.compile(loss="mse", optimizer="rmsprop")
    print("> Compilation Time : ", time.time() - start)
    return model
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

predict_point_by_point

```
def predict_point_by_point(model, data):
    #Predict each timestep given the last sequence of true data,
    #in effect only predicting 1 step ahead each time
      predicted = model.predict(data)
      predicted = np.reshape(predicted, (predicted.size,))
    return predicted
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

predict_sequence_full

```
def predict_sequence_full(model, data, window_size):
    #Shift the window by 1 new prediction each time, re-run predictions on new window
    curr_frame = data[0]
    predicted = []
    for i in range(len(data)):
        predicted.append(model.predict(curr_frame[newaxis,:,:])[0,0])
        curr_frame = curr_frame[1:]
        curr_frame = np.insert(curr_frame, [window_size-1], predicted[-1], axis=0)
    return predicted
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

predict_sequences_multiple

```
def predict_sequences_multiple(model, data, window_size, prediction_len):
    #Predict sequence of 50 steps before shifting prediction run forward by 50 steps
    prediction_seqs = []
    for i in range(int(len(data)/prediction_len)):
        curr_frame = data[i*prediction_len]
        predicted = []
        for j in range(prediction_len):
            predicted.append(model.predict(curr_frame[newaxis,:,:])[0,0])
            curr_frame = curr_frame[1:]
            curr_frame = np.insert(curr_frame, [window_size-1], predicted[-1], axis=0)
        prediction_seqs.append(predicted)
    return prediction_seqs
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

plot_results

```
def plot_results(predicted_data, true_data, outputfilename):
    fig = plt.figure(facecolor='white', figsize=(14, 8))
    ax = fig.add_subplot(111)
    ax.plot(true_data, label='True Data')
    plt.plot(predicted_data, label='Prediction')
    plt.legend()
    plt.show()
    plt.savefig(outputfilename + '.png')
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

plot_results_multiple

```
def plot_results_multiple(predicted_data, true_data, prediction_len):
    fig = plt.figure(facecolor='white', figsize=(14, 8))
    ax = fig.add_subplot(111)
    ax.plot(true_data, label='True Data')
    #Pad the list of predictions to shift it in the graph to it's correct start
    for i, data in enumerate(predicted_data):
        padding = [None for p in range(i * prediction_len)]
        plt.plot(padding + data, label='Prediction')
        plt.legend()
    plt.show()
    plt.savefig('plot_results_multiple.png')
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

plot_history

```
def plot_history(history, epochs, filename):
    plt.figure(figsize=(10, 8))  # make separate figure
    ax = plt.subplot(1, 1, 1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train loss', 'test val_loss'], loc='upper left')
    plt.show()
    plt.savefig("plot_history_" + filename + "_" + str(epochs) + ".png", dpi= 300)
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

model.fit

```
global_start_time = time.time()
epochs  = 1
seq_len = 50
print('> Loading data... ')
X_train, y_train, X_test, y_test = load_data('sp500.csv', seq_len, True)

print('> Data Loaded. Compiling...')

model = build_model([1, 50, 100, 1])
model.fit(X_train, y_train, batch_size=512, nb_epoch=epochs,
validation_split=0.05)

predictions = predict_sequences_multiple(model, X_test, seq_len, 50)
print('predictions.shape:', np.array(predictions).shape)
predictions_full = predict_sequence_full(model, X_test, seq_len)
print('predictions_full.shape:', np.array(predictions_full).shape)
predictions_point_by_point = predict_point_by_point(model, X_test)
print('predictions_point_by_point.shape:', np.array(predictions_point_by_point).shape)

print('Training duration (s) : ', time.time() - global_start_time)

plot_results_multiple(predictions, y_test, 50)
plot_results(predictions_full, y_test,'predictions_full')
plot_results(predictions_point_by_point, y_test,'predictions_point_by_point')
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

model.fit

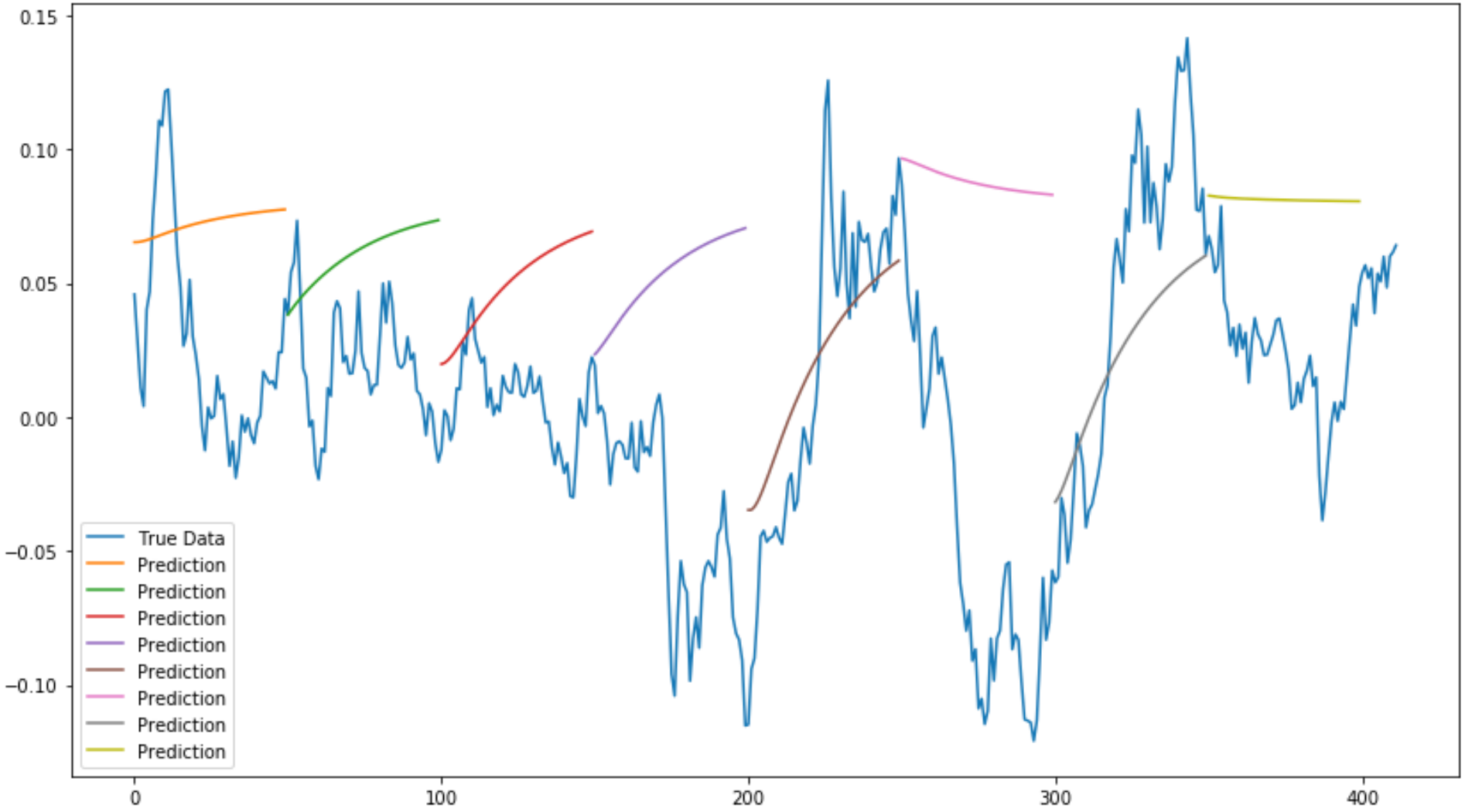
```
global_start_time = time.time()
epochs  = 100
seq_len = 50
print('> Loading data... ')
X_train, y_train, X_test, y_test = load_data('sp500.csv', seq_len, True)

print('> Data Loaded. Compiling...')

model = build_model([1, 50, 100, 1])
history = model.fit(X_train, y_train, batch_size=512, nb_epoch=epochs,
validation_split=0.05)

predictions = predict_sequences_multiple(model, X_test, seq_len, 50)
print('predictions.shape:', np.array(predictions).shape)
predictions_full = predict_sequence_full(model, X_test, seq_len)
print('predictions_full.shape:', np.array(predictions_full).shape)
predictions_point_by_point = predict_point_by_point(model, X_test)
print('predictions_point_by_point.shape:', np.array(predictions_point_by_point).shape)

print('Training duration (s) : ', time.time() - global_start_time)

plot_results_multiple(predictions, y_test, 50)
plot_results(predictions_full, y_test,'predictions_full')
plot_results(predictions_point_by_point, y_test,'predictions_point_by_point')

filename = "LSTM_Time_Series_Keras"
plot_history(history, epochs, filename)
```

Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

Deep Learning Training…

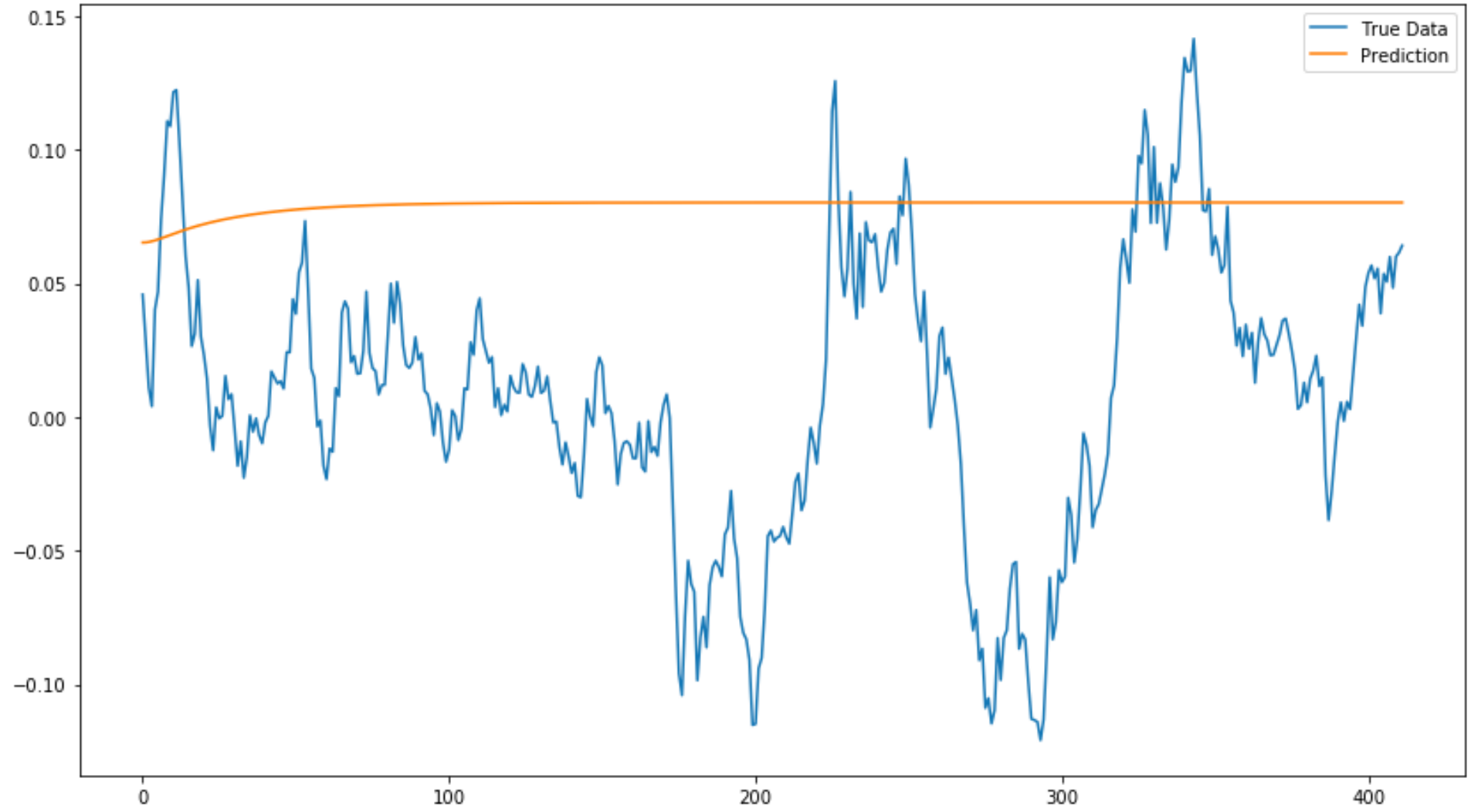
```
> Loading data...
X_train.shape: (3709, 50, 1)
y_train.shape: (3709,)
X_test.shape: (412, 50, 1)
y_test.shape: (412,)
> Data Loaded. Compiling...
> Compilation Time : 0.03888273239135742
Train on 3523 samples, validate on 186 samples
Epoch 1/100
3523/3523 [==============================] - 16s 4ms/step - loss: 0.0024 - val_loss: 8.0286e-04
Epoch 2/100
3523/3523 [==============================] - 6s 2ms/step - loss: 7.5175e-04 - val_loss: 6.8021e-04
Epoch 3/100
3523/3523 [==============================] - 6s 2ms/step - loss: 6.7244e-04 - val_loss: 5.9541e-04
…

Epoch 99/100
3523/3523 [==============================] - 6s 2ms/step - loss: 2.0188e-04 - val_loss: 1.9586e-04
Epoch 100/100
3523/3523 [==============================] - 6s 2ms/step - loss: 1.9659e-04 - val_loss: 3.1408e-04
predict_sequences_multiple shape: (8, 50)
predict_sequence_full shape: (412,)
predict_point_by_point shape: (412,)
Training duration (s) : 650.2197120189667
```

plot_results_multiple (predictions, y_test, 50)



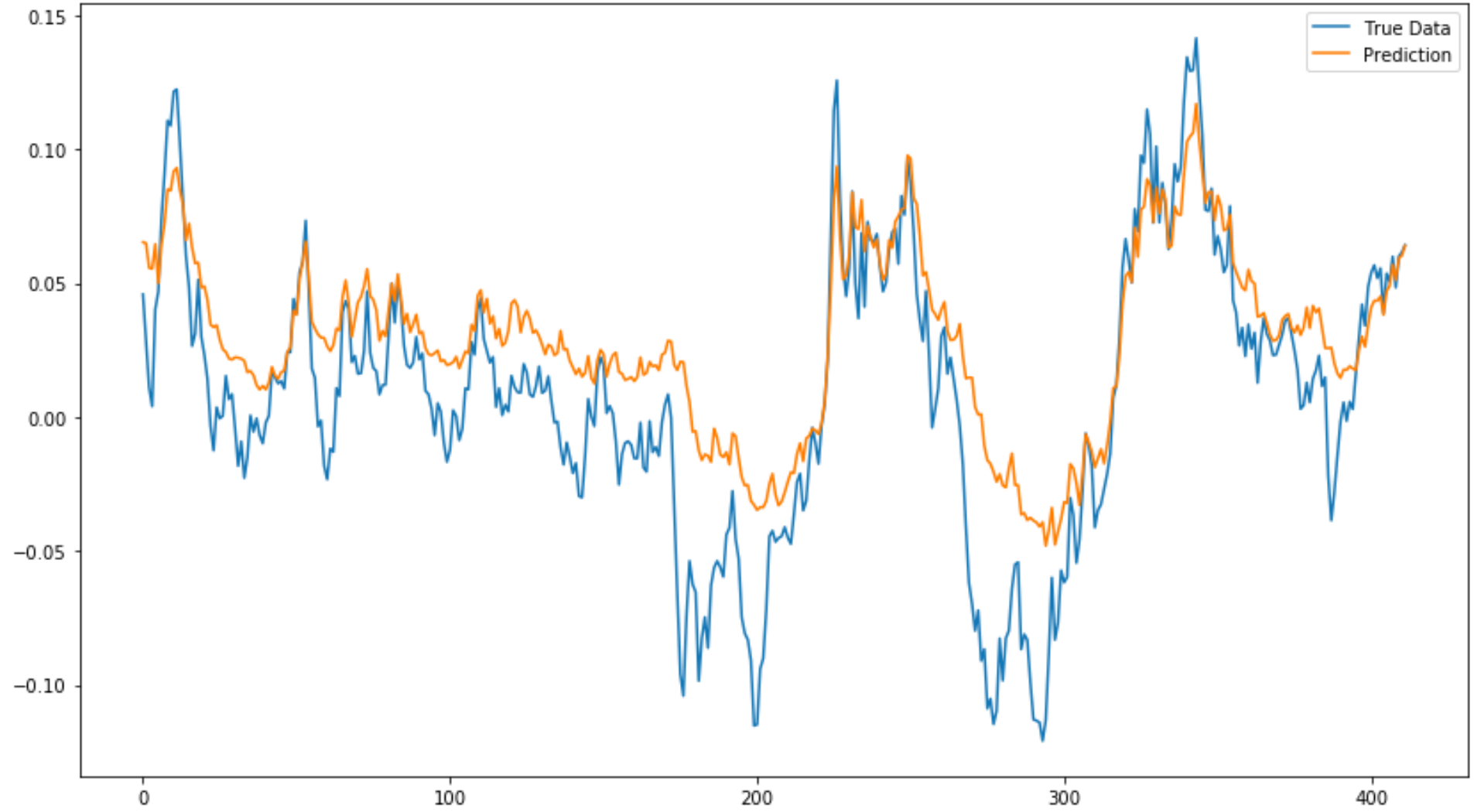
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

plot_results(predictions_full, y_test,'predictions_full')



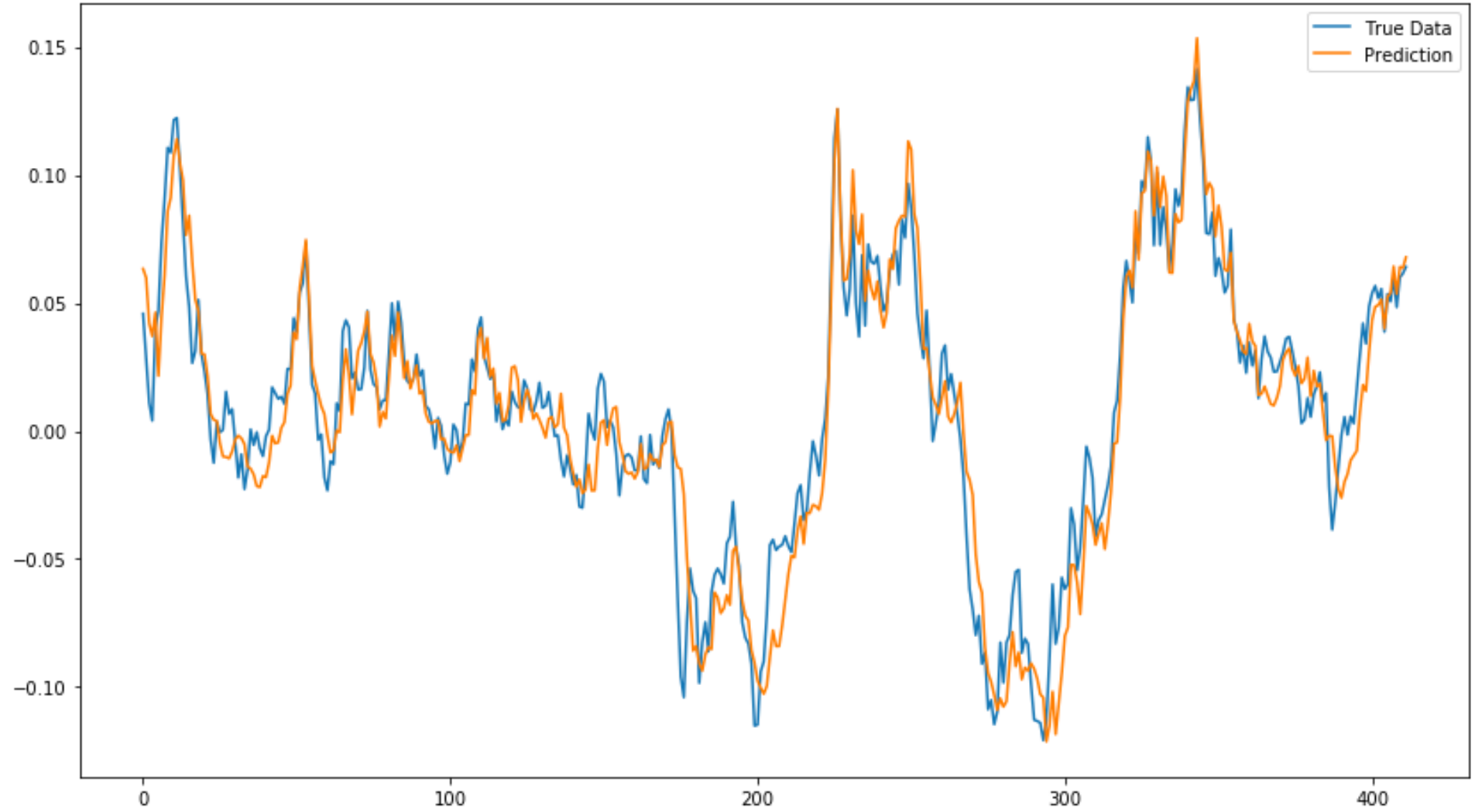
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

epochs = 1
seq_len = 50



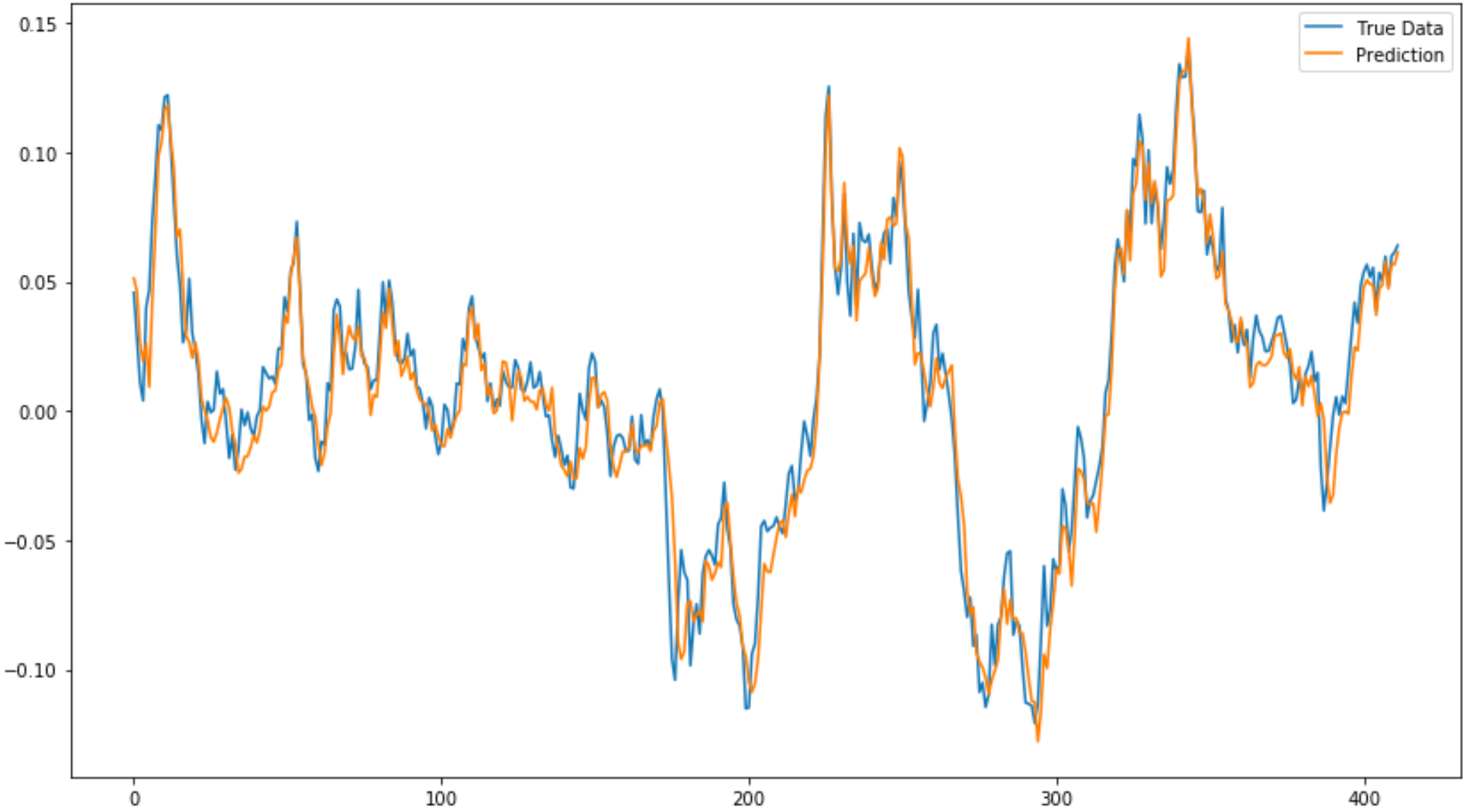
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

epochs = 10
seq_len = 50



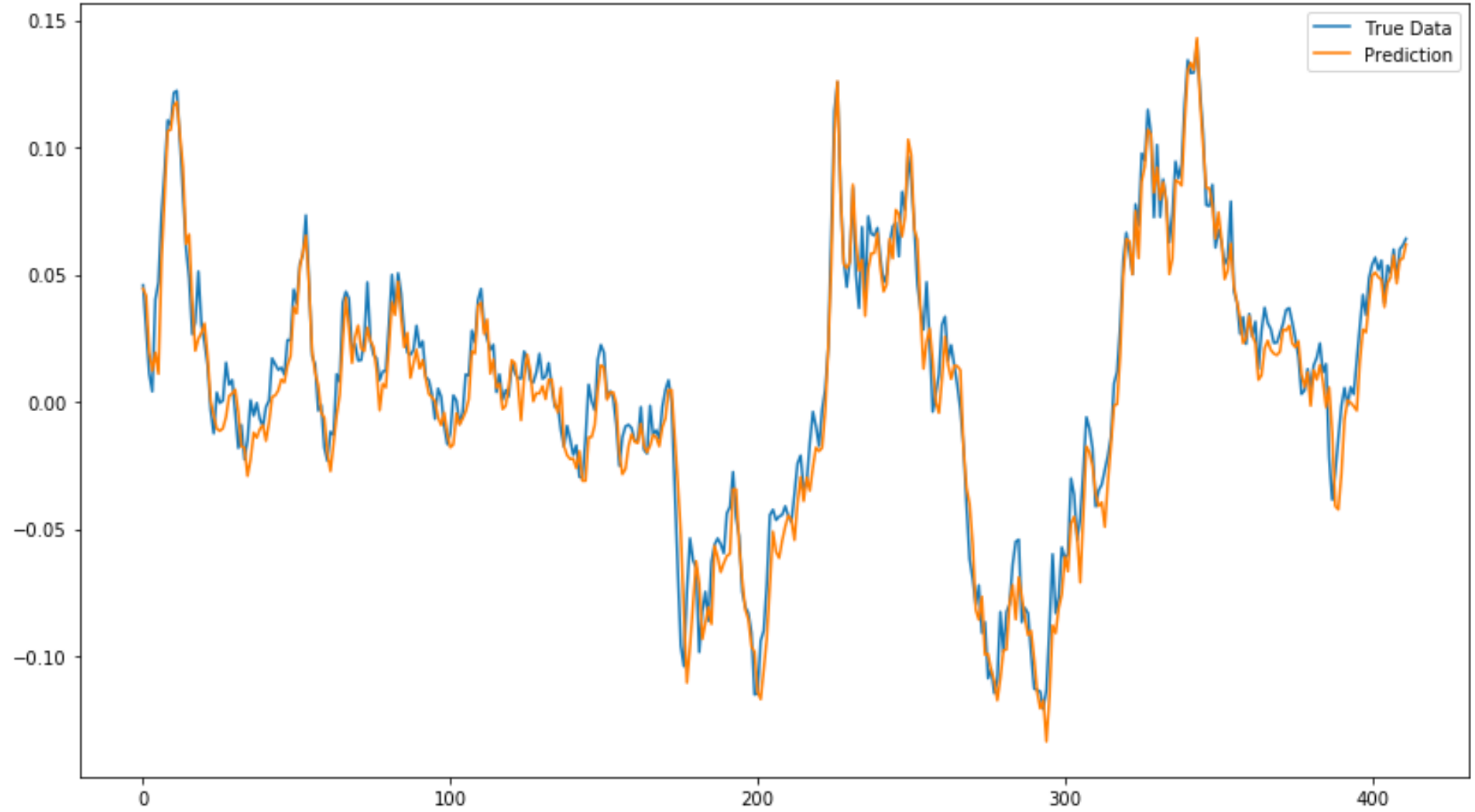
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

epochs = 30
seq_len = 50



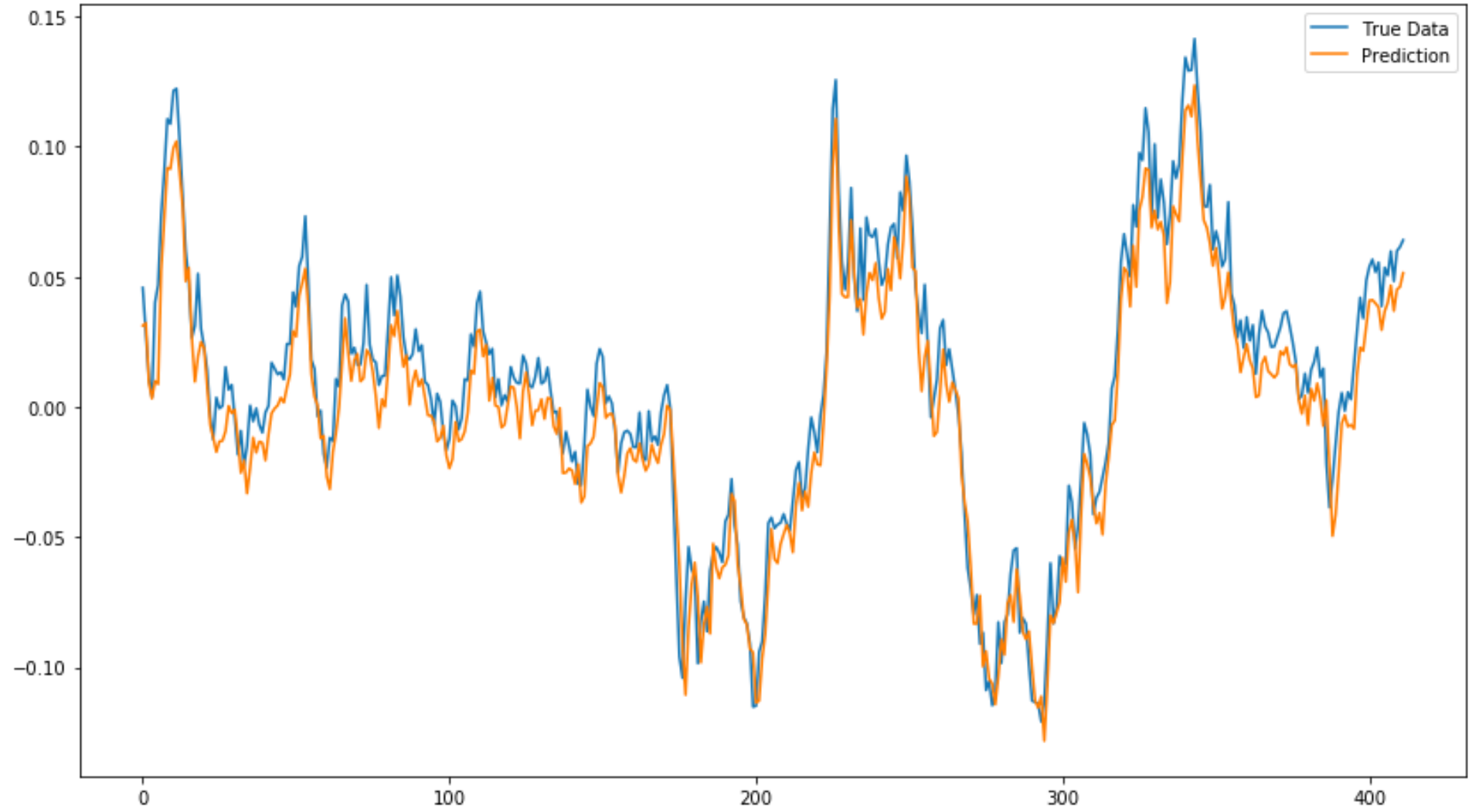
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

epochs = 60
seq_len = 50



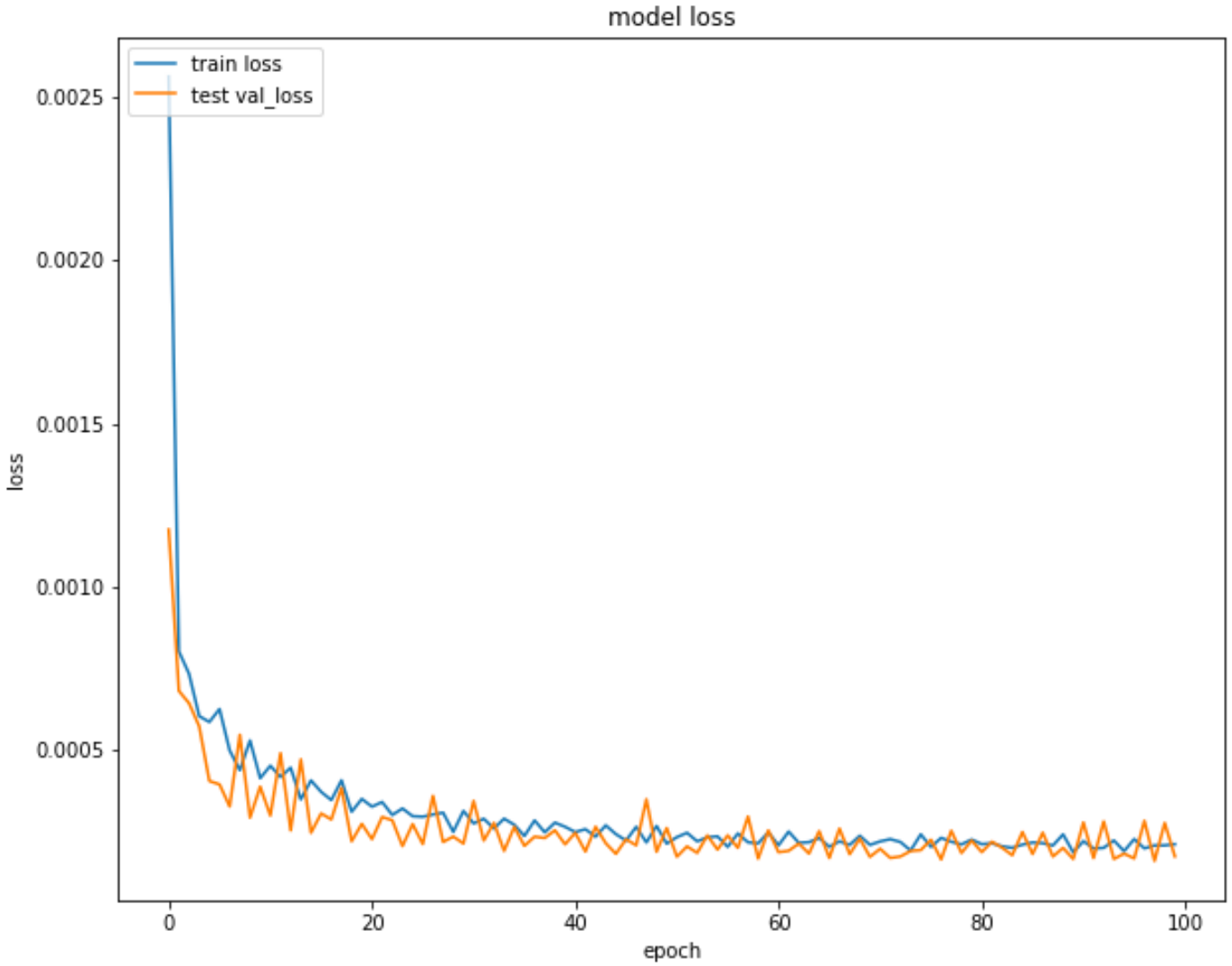
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

epochs = 100
seq_len = 50



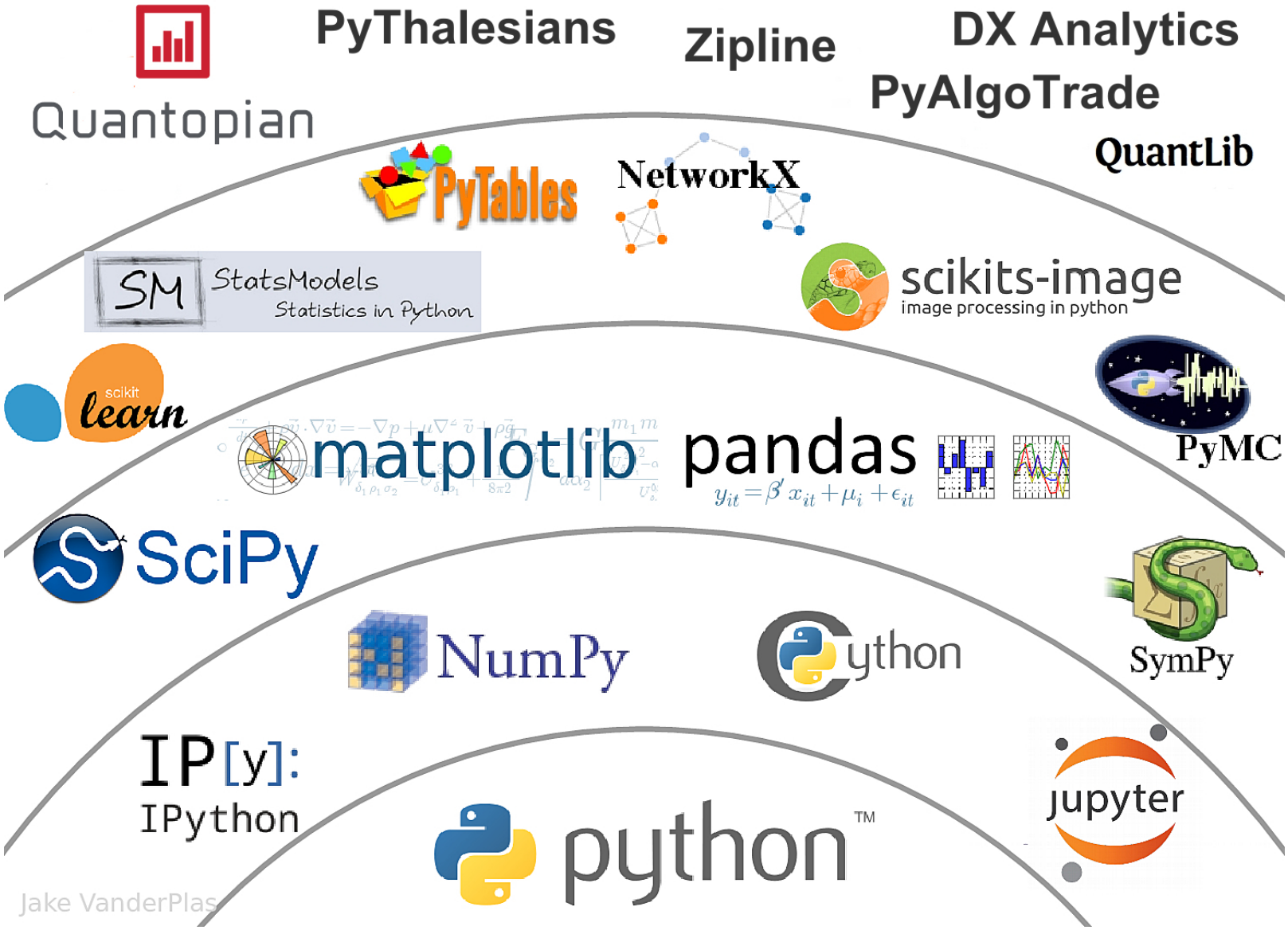
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

Model Fit History
Model Loss



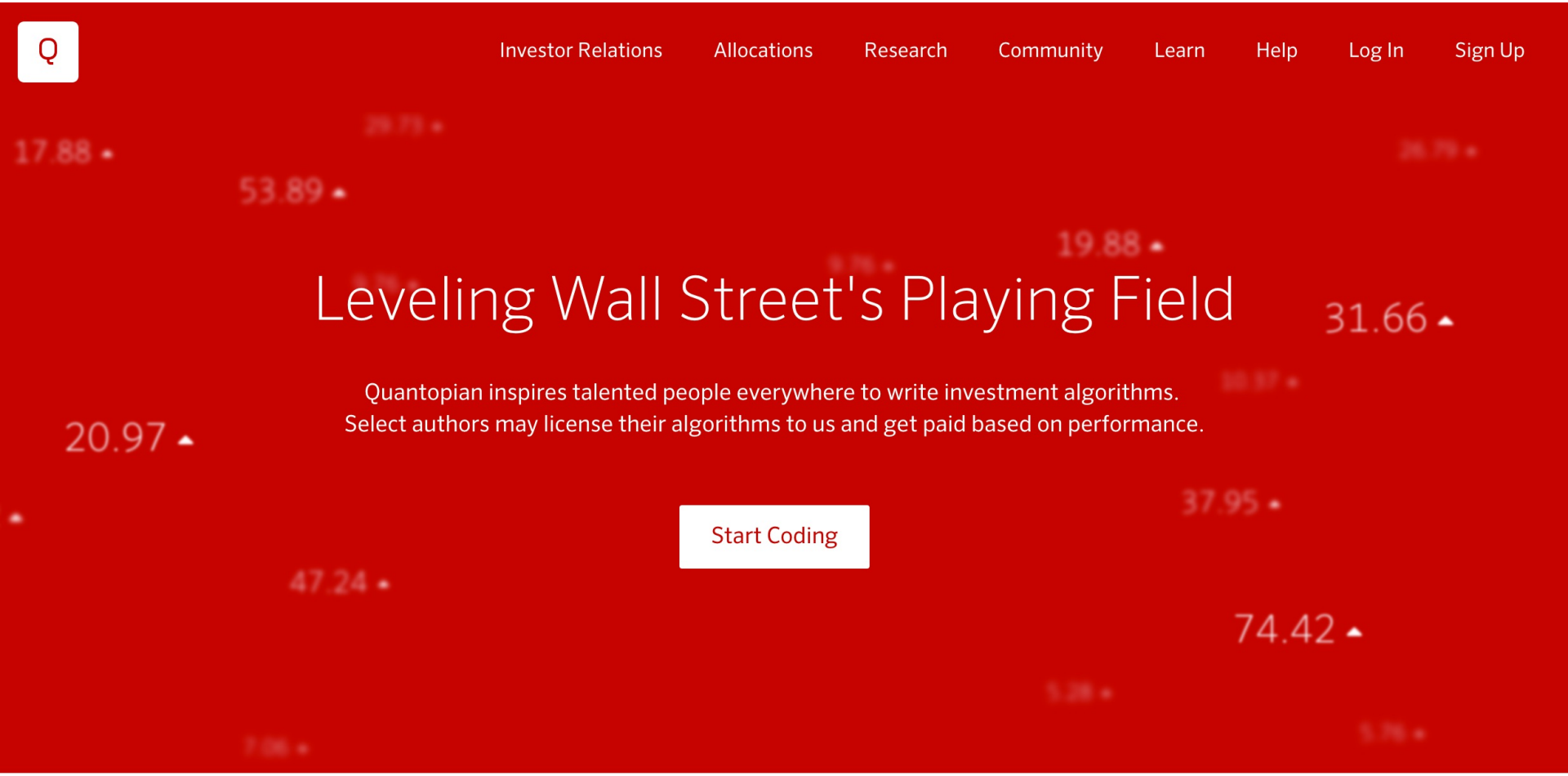
Source: https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction

The Quant Finance PyData Stack



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

Quantopian



https://www.quantopian.com/

References

- Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Yves Hilpisch (2015), Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging, Wiley
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
- James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
- Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
- Wes McKinney (2013), 10-minute tour of pandas, https://vimeo.com/59324550
- Jason Wirth (2015), A Visual Guide To Pandas, https://www.youtube.com/watch?v=9d5-Ti6onew
- Edward Schofield (2013), Modern scientific computing and big data analytics in Python, PyCon Australia, https://www.youtube.com/watch?v=hqOsfS3dP9w
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Python Programming Language, http://pythonprogramminglanguage.com/
- Numpy, http://www.numpy.org/
- Pandas, http://pandas.pydata.org/

References

- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media. https://github.com/wesm/pydata-book
- Avinash Jain (2017), Introduction To Python Programming, Udemy, https://www.udemy.com/pythonforbeginnersintro/
- Alfred Essa (2015), Awesome Data Science: 1.0 Jupyter Notebook Tour, https://www.youtube.com/watch?v=e9cSF3eVQv0
- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Ivan Idris (2015), Numpy Beginner's Guide, Third Edition, Packt Publishing
- Numpy Tutorial, https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), https://www.youtube.com/watch?v=u4alGiomYP4
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFfI8
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, https://goo.gl/pHeXe7, https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Deep Learning Studio: Cloud platform for designing Deep Learning AI without programming, http://deepcognition.ai/
- Natural Language Processing with Deep Learning (Winter 2017), https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow
- Jakob Aungiers (2016), LSTM Neural Network for Time Series Prediction, https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction
- Francois Chollet (2017), Deep Learning with Python, Manning Publications, https://github.com/fchollet/deep-learning-with-python-notebooks