大數據分析
(Big Data Analysis)
TensorFlow 深度學習
金融大數據分析
(Deep Learning for Finance Big Data Analysis with TensorFlow)

Min-Yuh Day
Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday
2020-11-25, 2020-12-09, 2020-12-16
課程大綱 (Syllabus)

週次 (Week)  日期 (Date)  內容 (Subject/Topics)
1  2020/09/16  大數據分析介紹 (Introduction to Big Data Analysis)
2  2020/09/23  AI人工智慧與大數據分析 (AI and Big Data Analysis)
3  2020/09/30  Python 大數據分析基礎 (Foundations of Big Data Analysis in Python)
4  2020/10/07  數位沙盒第一堂課：數位沙盒服務平台簡介 (Digital Sandbox Lesson 1: Introduction to FintechSpace Digital Sandbox)
5  2020/10/14  數位沙盒第二堂課：工程師操作說明與實作教學 (Digital Sandbox Lesson 2: Hands-on Practices)
6  2020/10/21  Python Pandas 大數據量化分析 (Quantitative Big Data Analysis with Pandas in Python)
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Talks</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2020/10/28</td>
<td>Python Scikit-Learn 機器學習 I (Machine Learning with Scikit-Learn in Python I)</td>
</tr>
<tr>
<td>8</td>
<td>2020/11/04</td>
<td>數位沙盒第三堂課：學生小組討論實作與成果發表 (Digital Sandbox Lesson 3: Learning Teams Hands-on Project Discussion and Project Presentation)</td>
</tr>
<tr>
<td>9</td>
<td>2020/11/11</td>
<td>期中報告 (Midterm Project Report)</td>
</tr>
<tr>
<td>10</td>
<td>2020/11/18</td>
<td>Python Scikit-Learn 機器學習 II (Machine Learning with Scikit-Learn in Python II)</td>
</tr>
<tr>
<td>12</td>
<td>2020/12/02</td>
<td>大數據分析個案研究 (Case Study on Big Data Analysis)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>13</td>
<td>2020/12/09</td>
<td>TensorFlow 深度學習金融大數據分析 II (Deep Learning for Finance Big Data Analysis with TensorFlow II)</td>
</tr>
<tr>
<td>14</td>
<td>2020/12/16</td>
<td>TensorFlow 深度學習金融大數據分析 III (Deep Learning for Finance Big Data Analysis with TensorFlow III)</td>
</tr>
<tr>
<td>15</td>
<td>2020/12/23</td>
<td>AI 機器人理財顧問 (Artificial Intelligence for Robo-Advisors)</td>
</tr>
<tr>
<td>16</td>
<td>2020/12/30</td>
<td>金融科技智慧型交談機器人 (Conversational Commerce and Intelligent Chatbots for Fintech)</td>
</tr>
<tr>
<td>17</td>
<td>2021/01/06</td>
<td>期末報告 I (Final Project Report I)</td>
</tr>
<tr>
<td>18</td>
<td>2021/01/13</td>
<td>期末報告 II (Final Project Report II)</td>
</tr>
</tbody>
</table>
Deep Learning for Finance
Big Data Analysis with TensorFlow
Outline

• Deep Learning for Finance Big Data Analysis with TensorFlow
  – Deep Learning
  – Financial Time Series Forecasting
  – TensorFlow
AI Acting Humanly: The Turing Test Approach (Alan Turing, 1950)

• Knowledge Representation
• Automated Reasoning
• Machine Learning (ML)
  – Deep Learning (DL)
• Computer Vision (Image, Video)
• Natural Language Processing (NLP)
• Robotics

Computer Vision: Image Classification, Object Detection, Object Instance Segmentation

Computer Vision: Object Detection

(a) Object Classification

(b) Generic Object Detection (Bounding Box)

(c) Semantic Segmentation

(d) Object Instance Segmentation

YOLOv4: Optimal Speed and Accuracy of Object Detection

Deep learning for financial applications: A survey

Applied Soft Computing (2020)

Source:
Financial time series forecasting with deep learning: A systematic literature review: 2005–2019

Applied Soft Computing (2020)

Source:
First and foremost, we clustered the various topics within the financial applications research and presented them in Fig. 8. A quick glance at the figure shows us financial text mining and algorithmic trading are the top two fields that the researchers most worked on followed by risk assessment, sentiment analysis, portfolio management and fraud detection, respectively. The results indicate most of the papers were published within the past 3 years implying the domain is very hot and actively studied.

When the papers were clustered by the DL model type as presented in Fig. 9, we observe the dominance of RNN, DMLP and CNN over the remaining models, which might be expected, since these models are the most commonly preferred ones in general DL implementations. Meanwhile, RNN is a general umbrella model which has several versions including LSTM, GRU, etc. Within the RNN choice, most of the models actually belonged to LSTM, which is very popular in time series forecasting or regression problems. It is also used quite often in algorithmic trading. More than 70% of the RNN papers consisted of LSTM models. Meanwhile, DMLP generally fits well for classification problems; hence it is a common choice for most of the financial application areas. However, since it is a natural extension of its shallow counterpart MLP, it has a longer history than the other DL models.

CNN started getting more attention lately since most of the implementations appeared within the past 3 years. Careful analysis of CNN papers indicates that a recent trend of representing financial data with a 2-D image view in order to utilize CNN is growing. Hence CNN based models might overpass the other models in the future. It actually passed DMLP for the past 3 years. Furthermore, we attempted to provide more details about associations between the DL models and the financial application areas.

Fig. 10 gives the distribution of the models for the research areas through a model-topic heatmap. Since most of the papers had multiple DL models, the amount of models is more than the number of covered papers. The results indicate the broad acceptance of RNN, DMLP and CNN models in almost all financial application areas.

We also wanted to elaborate on the particular feature selections for each financial application area to see if we could spot any pattern. Fig. 11 gives the distribution of the features for the research areas through a feature-topic heatmap. Unlike

Deep learning for financial applications: Deep Learning Models

Fig. 9. The histogram of publication count in model types.

Fig. 10. Topic-model heatmap.

The model-topic heatmap, in this case, we saw a distinction between the associations. Even though price data and technical indicators have been very popular for most of the research areas that are involved with time series forecasting, like algorithmic trading, portfolio management, financial sentiment analysis and financial text mining, the studies that had more significant spatial characteristics like risk assessment and fraud detection did not depend much on these temporal features. One other noteworthy difference came up with the adaptation of text related features. Highly text-based applications like financial sentiment analysis, financial text mining, risk assessment and fraud detection preferred to use features like text (extracted from tweets, news or financial data) and sentiments during their model development and implementation. However, the temporal characteristics of the financial time series data were also important for financial sentiment analysis and financial text mining, since a significant portion of these models were integrated into algorithmic trading systems.

Fig. 12 elaborates on the distribution of the dataset types for the research areas through a dataset-topic heatmap. If we analyze the heatmap, we see similarities with the feature-topic associations. However, this time, we had three main clusters of dataset types, the first one being the temporal datasets like Stock, Index, Cryptocurrency, ETF, Commodity price datasets, and the second one being the text-based datasets like News, Tweets, Microblogs and Financial Reports, and the last one being the datasets that had both numeric and textual components like Consumer Data, Credit Data and Financial Reports from companies or analysts. As far as the dataset vs. application area associations are concerned, these three main clusters were distributed as follows: Stock, Index, Cryptocurrency, ETF datasets were used almost in every application area except Risk Assessment and Fraud Detection which had less of temporal properties. Meanwhile, Credit Data, Financial Reports and Consumer Data were particularly used by these two application areas, namely Risk Assessment and Fraud Detection. Lastly, pure text based datasets like news, tweets, microblogs were preferred by Financial Sentiment Analysis and Financial Text Mining studies. However, as was the case in the feature-topic associations, temporal datasets like stock, ETF, Index price datasets were also used with these studies since some of them were tied with algorithmic trading models.

6. Discussion and open issues

After reviewing all the publications based on the selected criteria explained in the previous section, we wanted to provide our findings of the current state-of-the-art situation. Our discussions are categorized by the DL models and implementation topics.

6.1. Discussions on DL models

It is possible to claim that LSTM is the dominant DL model that is preferred by most researchers, due to its well-established structure for financial time series data forecasting. Most of the financial implementations have time-varying data representations requiring regression-type approaches which fits very well for LSTM and its derivatives due to their easy adaptations to the problems. As long as the temporal nature of the financial data remains, LSTM and its related family models will maintain their popularities.

Meanwhile, CNN based models started getting more traction among researchers in the last two years. Unlike LSTM, CNN works better for classification problems and is more suitable for either non-time varying or static data representations. However, since most financial data is time-varying, under normal circumstances,
Deep learning for financial applications: Topic-Model Heatmap

<table>
<thead>
<tr>
<th>Model</th>
<th>algorithmic trading</th>
<th>risk assessment</th>
<th>fraud detection</th>
<th>portfolio management</th>
<th>asset pricing and derivatives market</th>
<th>cryptocurrency and blockchain studies</th>
<th>financial sentiment analysis</th>
<th>financial text mining</th>
<th>theoretical or conceptual studies</th>
<th>other financial applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>LSTM</td>
<td>15</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>13</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GRU</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CNN</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>DMLP</td>
<td>10</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>DBN</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>AE</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>RL</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RBM</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Other</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The histogram of publication count in model types.

The topic-model heatmap, in this case, we saw a distinction between the associations. Even though price data and technical indicators have been very popular for most of the research areas that are involved with time series forecasting, like algorithmic trading, portfolio management, financial sentiment analysis and financial text mining, the studies that had more significant spatial characteristics like risk assessment and fraud detection did not depend much on these temporal features. One other noteworthy difference came up with the adaptation of text related features. Highly text-based applications like financial sentiment analysis, financial text mining, risk assessment and fraud detection preferred to use features like text (extracted from tweets, news or financial data) and sentiments during their model development and implementation. However, the temporal characteristics of the financial time series data were also important for financial sentiment analysis and financial text mining, since a significant portion of these models were integrated into algorithmic trading systems.

Fig. 12 elaborates on the distribution of the dataset types for the research areas through a dataset-topic heatmap. If we analyze the heatmap, we see similarities with the feature-topic associations. However, this time, we had three main clusters of dataset types, the first one being the temporal datasets like Stock, Index, ETF, Cryptocurrency, Forex and Commodity price datasets, and the second one being the text-based datasets like News, Tweets, Microblogs and Financial Reports, and the last one being the datasets that had both numeric and textual components like Consumer Data, Credit Data and Financial Reports from companies or analysts. As far as the dataset vs. application area associations are concerned, these three main clusters were distributed as follows: Stock, Index, Cryptocurrency, ETF datasets were used almost in every application area except Risk Assessment and Fraud Detection which had less of temporal properties. Meanwhile, Credit Data, Financial Reports and Consumer Data were particularly used by these two application areas, namely Risk Assessment and Fraud Detection. Lastly, pure text based datasets like news, tweets, microblogs were preferred by Financial Sentiment Analysis and Financial Text Mining studies. However, as was the case in the feature-topic associations, temporal datasets like stock, ETF, Index price datasets were also used with these studies since some of them were tied with algorithmic trading models.

6. Discussion and open issues

After reviewing all the publications based on the selected criteria explained in the previous section, we wanted to provide our findings of the current state-of-the-art situation. Our discussions are categorized by the DL models and implementation topics.

6.1. Discussions on DL models

It is possible to claim that LSTM is the dominant DL model that is preferred by most researchers, due to its well-established structure for financial time series data forecasting. Most of the financial implementations have time-varying data representations requiring regression-type approaches which fits very well for LSTM and its derivatives due to their easy adaptations to the problems. As long as the temporal nature of the financial data remains, LSTM and its related family models will maintain their popularities.

Meanwhile, CNN based models started getting more traction among researchers in the last two years. Unlike LSTM, CNN works better for classification problems and is more suitable for either non-time varying or static data representations. However, since most financial data is time-varying, under normal circumstances,
Deep learning for financial applications:

Topic-Feature Heatmap

---

Deep learning for financial applications: Topic-Dataset Heatmap

Deep learning for financial applications:

Algo-trading applications embedded with time series forecasting models

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[33]</td>
<td>GarantiBank in BIST, Turkey</td>
<td>2016</td>
<td>OCHLV, Spread, Volatility, Turnover, etc.</td>
<td>PLR, Graves LSTM</td>
<td>MSE, RMSE, MAE, RSE, Correlation R-square</td>
<td>Spark</td>
</tr>
<tr>
<td>[34]</td>
<td>CSI300, Nifty50, HSI, Nikkei 225, S&amp;P500, DJIA</td>
<td>2010–2016</td>
<td>OCHLV, Technical Indicators</td>
<td>WT, Stacked autoencoders, LSTM</td>
<td>MAPE, Correlation coefficient, THEIL-U</td>
<td>–</td>
</tr>
<tr>
<td>[36]</td>
<td>50 stocks from NYSE</td>
<td>2007–2016</td>
<td>Price data</td>
<td>SFM</td>
<td>MSE</td>
<td>–</td>
</tr>
<tr>
<td>[37]</td>
<td>The LOB of 5 stocks of Finnish Stock Market</td>
<td>2010</td>
<td>FI-2010 dataset: bid/ask and volume</td>
<td>WMTR, MDA</td>
<td>Accuracy, Precision, Recall, F1-Score</td>
<td>–</td>
</tr>
<tr>
<td>[38]</td>
<td>300 stocks from SZSE, Commodity</td>
<td>2014–2015</td>
<td>Price data</td>
<td>FDDR, DMLP+RL</td>
<td>Profit, return, SR, profit-loss curves</td>
<td>Keras</td>
</tr>
<tr>
<td>[42]</td>
<td>Singapore Stock Market Index</td>
<td>2010–2017</td>
<td>OCHL of last 10 days of Index</td>
<td>DMLP</td>
<td>RMSE, MAPE, Profit, SR</td>
<td>–</td>
</tr>
<tr>
<td>[43]</td>
<td>GBP/USD</td>
<td>2017</td>
<td>Price data</td>
<td>Reinforcement Learning + LSTM + NES</td>
<td>SR, downside deviation ratio, total profit</td>
<td>Python, Keras, TensorFlow</td>
</tr>
</tbody>
</table>


18
Deep learning for financial applications: 
**Algo-trading applications embedded with time series forecasting models**

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[46]</td>
<td>Bitcoin, Dash, Ripple, Monero, Litecoin, Dogecoin, Nxt, Namecoin</td>
<td>2014–2017</td>
<td>MA, BOLL, the CRIX returns, Euribor interest rates, OCHLV</td>
<td>LSTM, RNN, DMLP</td>
<td>Accuracy, F1-measure</td>
<td>Python, Tensorflow</td>
</tr>
</tbody>
</table>

Deep learning for financial applications:

Classification (buy–sell signal, or trend detection) based algo-trading models

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[51]</td>
<td>Stocks in Dow30</td>
<td>1997–2017</td>
<td>RSI</td>
<td>DMLP with genetic algorithm</td>
<td>Annualized return</td>
<td>Spark MLlib, Java</td>
</tr>
<tr>
<td>[52]</td>
<td>SPY ETF, 10 stocks from S&amp;P500</td>
<td>2014–2016</td>
<td>Price data</td>
<td>FFNN</td>
<td>Cumulative gain</td>
<td>MatConvNet, Matlab</td>
</tr>
<tr>
<td>[53]</td>
<td>Dow30 stocks</td>
<td>2012–2016</td>
<td>Close data and several technical indicators</td>
<td>LSTM</td>
<td>Accuracy</td>
<td>Python, Keras, Tensorflow, TALIB</td>
</tr>
<tr>
<td>[54]</td>
<td>High-frequency record of all orders</td>
<td>2014–2017</td>
<td>Price data, record of all orders, transactions</td>
<td>LSTM</td>
<td>Accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[55]</td>
<td>Nasdaq Nordic (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)</td>
<td>2010</td>
<td>Price and volume data in LOB</td>
<td>LSTM</td>
<td>Precision, Recall, F1-score, Cohen’s k</td>
<td>–</td>
</tr>
<tr>
<td>[56]</td>
<td>17 ETFs</td>
<td>2000–2016</td>
<td>Price data, technical indicators</td>
<td>CNN</td>
<td>Accuracy, MSE, Profit, AUROC</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[57]</td>
<td>Stocks in Dow30 and 9 Top Volume ETFs</td>
<td>1997–2017</td>
<td>Price data, technical indicators</td>
<td>CNN with feature imaging</td>
<td>Recall, precision, F1-score, annualized return</td>
<td>Python, Keras, Tensorflow, Java</td>
</tr>
<tr>
<td>[58]</td>
<td>FTSE100</td>
<td>2000–2017</td>
<td>Price data</td>
<td>CAE</td>
<td>TR, SR, MDD, mean return</td>
<td>–</td>
</tr>
<tr>
<td>[59]</td>
<td>Nasdaq Nordic (Kesko Oyj, Outokumpu Oyj, Sampo, Rautaruukki, Wartsila Oyj)</td>
<td>2010</td>
<td>Price, Volume data, 10 orders of the LOB</td>
<td>CNN</td>
<td>Precision, Recall, F1-score, Cohen’s k</td>
<td>Theano, Scikit learn, Python</td>
</tr>
<tr>
<td>[60]</td>
<td>Borsa Istanbul 100 Stocks</td>
<td>2011–2015</td>
<td>75 technical indicators and OCHLV</td>
<td>CNN</td>
<td>Accuracy</td>
<td>Keras</td>
</tr>
<tr>
<td>[61]</td>
<td>ETFs and Dow30</td>
<td>1997–2007</td>
<td>Price data</td>
<td>CNN with feature imaging</td>
<td>Annualized return</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[62]</td>
<td>8 experimental assets from bond/derivative market</td>
<td>–</td>
<td>Asset prices data</td>
<td>RL, DMLP, Genetic Algorithm</td>
<td>Learning and genetic algorithm error</td>
<td>–</td>
</tr>
<tr>
<td>[63]</td>
<td>10 stocks from S&amp;P500</td>
<td>–</td>
<td>Stock Prices</td>
<td>TDNN, RNN, PNN</td>
<td>Missed opportunities, false alarms ratio</td>
<td>–</td>
</tr>
</tbody>
</table>

Deep learning for financial applications:
Stand-alone and/or other algorithmic models

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[67]</td>
<td>Taiwan Stock Index Futures, Mini Index Futures</td>
<td>2012–2014</td>
<td>Price data to image</td>
<td>Visualization method + CNN</td>
<td>Accumulated profits, accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[70]</td>
<td>Taiwan stock index futures (TAIFEX)</td>
<td>2017</td>
<td>Price data</td>
<td>Agent based RL with CNN pre-trained</td>
<td>Accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[71]</td>
<td>Stocks from S&amp;P500</td>
<td>2010–2016</td>
<td>OCHLV</td>
<td>DCNL</td>
<td>PCC, DTW, VWL</td>
<td>Pytorch</td>
</tr>
<tr>
<td>[73]</td>
<td>489 stocks from S&amp;P500 and NASDAQ-100</td>
<td>2014–2015</td>
<td>Limit Order Book</td>
<td>Spatial neural network</td>
<td>Cross entropy error</td>
<td>NVIDIA’s cuDNN</td>
</tr>
<tr>
<td>[74]</td>
<td>Experimental dataset</td>
<td>–</td>
<td>Price data</td>
<td>DRL with CNN, LSTM, GRU, DMLP</td>
<td>Mean profit</td>
<td>Python</td>
</tr>
</tbody>
</table>

# Deep learning for financial applications: Credit scoring or classification studies

## Table 4

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[77]</td>
<td>The XR 14 CDS contracts</td>
<td>2016</td>
<td>Recovery rate, spreads, sector and region</td>
<td>DBN+RBM</td>
<td>AUROC, FN, FP, Accuracy</td>
<td>WEKA</td>
</tr>
<tr>
<td>[78]</td>
<td>German, Japanese credit datasets</td>
<td>–</td>
<td>Personal financial variables</td>
<td>SVM + DBN</td>
<td>Weighted-accuracy, TP, TN</td>
<td>–</td>
</tr>
<tr>
<td>[79]</td>
<td>Credit data from Kaggle</td>
<td>–</td>
<td>Personal financial variables</td>
<td>DMLP</td>
<td>Accuracy, TP, TN, G-mean</td>
<td>–</td>
</tr>
<tr>
<td>[80]</td>
<td>Australian, German credit data</td>
<td>–</td>
<td>Personal financial variables</td>
<td>GP + AE as Boosted DMLP</td>
<td>FP</td>
<td>Python, Scikit-learn</td>
</tr>
<tr>
<td>[81]</td>
<td>German, Australian credit dataset</td>
<td>–</td>
<td>Personal financial variables</td>
<td>DCNN, DMLP</td>
<td>Accuracy, False/Missed alarm</td>
<td>–</td>
</tr>
<tr>
<td>[82]</td>
<td>Consumer credit data from Chinese finance company</td>
<td>–</td>
<td>Relief algorithm chose the 50 most important features</td>
<td>CNN + Relief</td>
<td>AUROC, K-s statistic, Accuracy</td>
<td>Keras</td>
</tr>
<tr>
<td>[83]</td>
<td>Credit approval dataset by UCI Machine Learning repo</td>
<td>–</td>
<td>UCI credit approval dataset</td>
<td>Rectifier, Tanh, Maxout DL</td>
<td>–</td>
<td>AWS EC2, H2O, R</td>
</tr>
</tbody>
</table>

## Table 5

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[84]</td>
<td>966 french firms</td>
<td>–</td>
<td>Financial ratios</td>
<td>RBM+SVM</td>
<td>Precision, Recall</td>
<td>–</td>
</tr>
<tr>
<td>[85]</td>
<td>883 BHC from EDGAR 2006–2017</td>
<td>Tokens, weighted sentiment polarity, leverage and ROA</td>
<td>CNN, LSTM, SVM, RF</td>
<td>Accuracy, Precision, Recall, F1-score</td>
<td>Keras, Python, Scikit-learn</td>
<td></td>
</tr>
<tr>
<td>[86]</td>
<td>The event data set for large European banks, news articles from Reuters 2007–2014</td>
<td>Word, sentence</td>
<td>DMLP +NLP pre-process</td>
<td>Relative usefulness, F1-score</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>[87]</td>
<td>Event dataset on European banks, news from Reuters 2007–2014</td>
<td>Text, sentence</td>
<td>Sentence vector + DFFN</td>
<td>Usefulness, F1-score, AUROC</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>[89]</td>
<td>Macro/Micro economic variables, Bank character/performance variables from BHC 1976–2017</td>
<td>Macro economic variables and bank performances</td>
<td>CGAN, MVN, MV-t, LSTM, VAR, FE-QAR</td>
<td>RMSE, Log likelihood, Loan loss rate</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>[92]</td>
<td>Financial statements of several companies from Japanese stock market 2002–2016</td>
<td>Financial ratios</td>
<td>CNN</td>
<td>F1-score, AUROC</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>[95]</td>
<td>Private brokerage company's real data of risky transactions – 250 features: order details, etc.</td>
<td></td>
<td>CNN, LSTM</td>
<td>F1-Score</td>
<td>Keras, Tensorflow</td>
<td></td>
</tr>
<tr>
<td>[96]</td>
<td>Several datasets combined to create a new one</td>
<td>Index data, 10-year Bond yield, exchange rates,</td>
<td>Logit, CART, RF, SVM, NN, XGBoost, DMLP</td>
<td>AUROC, KS, G-mean, likelihood ratio, DP, BA, WBA</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

## Table 4: Credit scoring or classification studies.

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[84]</td>
<td>966 french firms</td>
<td>–</td>
<td>Financial ratios</td>
<td>RBM+SVM</td>
<td>Precision, Recall</td>
<td>–</td>
</tr>
<tr>
<td>[85]</td>
<td>883 BHC from EDGAR</td>
<td>2006–2017</td>
<td>Tokens, weighted sentiment polarity, leverage and ROA</td>
<td>CNN, LSTM, SVM, RF</td>
<td>Accuracy, Precision, Recall, F1-score</td>
<td>Keras, Python, Scikit-learn</td>
</tr>
<tr>
<td>[86]</td>
<td>The event data set for large European banks, news articles from Reuters</td>
<td>2007–2014</td>
<td>Word, sentence</td>
<td>DMLP +NLP preprocess</td>
<td>Relative usefulness, F1-score</td>
<td>–</td>
</tr>
<tr>
<td>[87]</td>
<td>Event dataset on European banks, news from Reuters</td>
<td>2007–2014</td>
<td>Text, sentence</td>
<td>Sentence vector + DFFN</td>
<td>Usefulness, F1-score, AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[89]</td>
<td>Macro/Micro economic variables, Bank characteristics/performance variables from BHC</td>
<td>1976–2017</td>
<td>Macro economic variables and bank performances</td>
<td>CGAN, MVN, MV-t, LSTM, VAR, FE-QAR</td>
<td>RMSE, Log likelihood, Loan loss rate</td>
<td>–</td>
</tr>
<tr>
<td>[92]</td>
<td>Financial statements of several companies from Japanese stock market</td>
<td>2002–2016</td>
<td>Financial ratios</td>
<td>CNN</td>
<td>F1-score, AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[95]</td>
<td>Private brokerage company's real data of risky transactions</td>
<td>–</td>
<td>250 features: order details, etc.</td>
<td>CNN, LSTM</td>
<td>F1-Score</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[96]</td>
<td>Several datasets combined to create a new one</td>
<td>1996–2017</td>
<td>Index data, 10-year Bond yield, exchange rates, Logit, CART, RF, SVM, NN, XGBoost, DMLP</td>
<td>AUROC, KS, G-mean, likelihood ratio, DP, BA, WBA</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
## Fraud detection studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[114]</td>
<td>Debit card transactions by a local Indonesia bank</td>
<td>2016–2017</td>
<td>Financial transaction amount on several time periods</td>
<td>CNN, Stacked-LSTM, CNN-LSTM</td>
<td>AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[115]</td>
<td>Credit card transactions from retail banking</td>
<td>2017</td>
<td>Transaction variables and several derived features</td>
<td>LSTM, GRU</td>
<td>Accuracy</td>
<td>Keras</td>
</tr>
<tr>
<td>[116]</td>
<td>Card purchases' transactions</td>
<td>2014–2015</td>
<td>Probability of fraud per country, origin, other fraud related features</td>
<td>DMLP</td>
<td>AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[117]</td>
<td>Transactions made with credit cards by European cardholders</td>
<td>2013</td>
<td>Personal financial variables to PCA</td>
<td>DMLP, RF</td>
<td>Recall, Precision, Accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[118]</td>
<td>Credit-card transactions</td>
<td>2015</td>
<td>Transaction and bank features</td>
<td>LSTM</td>
<td>AUROC</td>
<td>Keras, Scikit-learn</td>
</tr>
<tr>
<td>[119]</td>
<td>Databases of foreign trade of the Secretariat of Federal Revenue of Brazil</td>
<td>2014</td>
<td>8 Features: Foreign Trade, Tax, Transactions, Employees, Invoices, etc.</td>
<td>AE</td>
<td>MSE</td>
<td>H2O, R</td>
</tr>
<tr>
<td>[120]</td>
<td>Chamber of Deputies open data, Companies data from Secretariat of Federal Revenue of Brazil</td>
<td>2009–2017</td>
<td>21 features: Brazilian State expense, party name, Type of expense, etc.</td>
<td>Deep Autoencoders</td>
<td>MSE, RMSE</td>
<td>H2O, R</td>
</tr>
<tr>
<td>[121]</td>
<td>Real-world data for automobile insurance company labeled as fraudulent</td>
<td>–</td>
<td>Car, insurance and accident related features</td>
<td>DMLP + LDA</td>
<td>TP, FP, Precision, F1-score</td>
<td>–</td>
</tr>
<tr>
<td>[122]</td>
<td>Transactions from a giant online payment platform</td>
<td>2006</td>
<td>Personal financial variables</td>
<td>GBDT+DMLP</td>
<td>AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[123]</td>
<td>Financial transactions</td>
<td>–</td>
<td>Transaction data</td>
<td>LSTM</td>
<td>t-SNE</td>
<td>–</td>
</tr>
<tr>
<td>[124]</td>
<td>Empirical data from Greek firms</td>
<td>–</td>
<td>–</td>
<td>DQL</td>
<td>Revenue</td>
<td>Torch</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Portfolio management studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[129]</td>
<td>Chinese stock data</td>
<td>2012–2013</td>
<td>Technical, fundamental data</td>
<td>Logistic Regression, RF, DMLP</td>
<td>AUC, accuracy, precision, recall, f1, tpr, fpr</td>
<td>Keras, Tensorflow, Python, Scikit learn</td>
</tr>
<tr>
<td>[132]</td>
<td>Taiwans stock market</td>
<td>–</td>
<td>Price data</td>
<td>Elman RNN</td>
<td>MSE, return</td>
<td>–</td>
</tr>
<tr>
<td>[133]</td>
<td>FOREX (EUR/USD, etc.), Gold</td>
<td>2013</td>
<td>Price data</td>
<td>Evolino RNN</td>
<td>Return</td>
<td>Python</td>
</tr>
<tr>
<td>[137]</td>
<td>Analyst reports on the TSE and Osaka Exchange</td>
<td>2016–2018</td>
<td>Text</td>
<td>LSTM, CNN, Bi-LSTM</td>
<td>Accuracy, R²</td>
<td>R, Python, MeCab</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Asset pricing and derivatives market studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>Der. type</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[137]</td>
<td>Asset pricing</td>
<td>Analyst reports on the TSE and Osaka Exchange</td>
<td>2016–2018</td>
<td>Text</td>
<td>LSTM, CNN, Bi-LSTM</td>
<td>Accuracy, $R^2$</td>
<td>R, Python, MeCab</td>
</tr>
<tr>
<td>[142]</td>
<td>Options</td>
<td>Simulated a range of call option prices</td>
<td>–</td>
<td>Price data, option strike/maturity, dividend/risk free rates, volatility</td>
<td>DMLP</td>
<td>RMSE, the average percentage pricing error</td>
<td>Tensorflow</td>
</tr>
<tr>
<td>[143]</td>
<td>Futures, Options</td>
<td>TAIEX Options</td>
<td>2017</td>
<td>OCHLV, fundamental analysis, option price</td>
<td>DMLP, DMLP with Black scholes</td>
<td>RMSE, MAE, MAPE</td>
<td>–</td>
</tr>
<tr>
<td>[144]</td>
<td>Equity returns</td>
<td>Returns in NYSE, AMEX, NASDAQ</td>
<td>1975–2017</td>
<td>57 firm characteristics</td>
<td>Fama–French n-factor model DL</td>
<td>$R^2$, RMSE</td>
<td>Tensorflow</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Cryptocurrency and blockchain studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[140]</td>
<td>12 most-volumed cryptocurrency</td>
<td>2015–2016</td>
<td>Hash value, bitcoin address, public/private key, digital signature, etc.</td>
<td>CNN + RL</td>
<td>SR, portfolio value, MDD</td>
<td>–</td>
</tr>
<tr>
<td>[146]</td>
<td>Bitcoin data</td>
<td>2012, 2013, 2016</td>
<td>TransactionId, input/output Addresses, timestamp</td>
<td>Graph embedding using heuristic, laplacian eigen-map, deep AE</td>
<td>F1-score</td>
<td>–</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Financial sentiment studies coupled with text mining for forecasting

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[137]</td>
<td>Analyst reports on the TSE and Osaka Exchange</td>
<td>2016–2018</td>
<td>Text</td>
<td>LSTM, CNN, Bi-LSTM</td>
<td>Accuracy, R²</td>
<td>R, Python, MeCab</td>
</tr>
<tr>
<td>[154]</td>
<td>SCI prices</td>
<td>2013–2016</td>
<td>Text data and Price data</td>
<td>LSTM</td>
<td>Accuracy, F1-Measure</td>
<td>Python, Keras</td>
</tr>
<tr>
<td>[155]</td>
<td>Stocks of Google, Microsoft and Apple</td>
<td>2016–2017</td>
<td>Twitter sentiment and stock prices</td>
<td>RNN</td>
<td>–</td>
<td>Spark, Flume, Twitter API,</td>
</tr>
<tr>
<td>[156]</td>
<td>30 DJIA stocks, S&amp;P500, DJI, news from Reuters</td>
<td>2002–2016</td>
<td>Price data and features from news articles</td>
<td>LSTM, NN, CNN and word2vec</td>
<td>Accuracy</td>
<td>VADER</td>
</tr>
<tr>
<td>[157]</td>
<td>Stocks of CSI300 index, OCHLV of CSI300 index</td>
<td>2009–2014</td>
<td>Sentiment Posts, Price data</td>
<td>Naive Bayes + LSTM</td>
<td>Precision, Recall, F1-score, Accuracy</td>
<td>Python, Keras</td>
</tr>
</tbody>
</table>

Deep learning for financial applications:
Text mining studies without sentiment analysis for forecasting

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[173]</td>
<td>Price data, index data, news, social media data</td>
<td>2015</td>
<td>Price data, news from articles and social media</td>
<td>Coupled matrix and tensor</td>
<td>Accuracy, MCC</td>
<td>Jieba</td>
</tr>
<tr>
<td>[174]</td>
<td>HS300</td>
<td>2015–2017</td>
<td>Social media news, price data</td>
<td>RNN-Boost with LDA</td>
<td>Accuracy, MAE, MAPE, RMSE</td>
<td>Python, Scikit-learn</td>
</tr>
</tbody>
</table>

Deep learning for financial applications:
Text mining studies without sentiment analysis for forecasting

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[175]</td>
<td>News and Chinese stock data</td>
<td>2014–2017</td>
<td>Selected words in a news</td>
<td>HAN</td>
<td>Accuracy, Annual return</td>
<td>–</td>
</tr>
<tr>
<td>[176]</td>
<td>News, stock prices from Hong Kong Stock Exchange</td>
<td>2001</td>
<td>Price data and TF-IDF from news</td>
<td>ELM, DLR, PCA, BELM, KELM, NN</td>
<td>Accuracy</td>
<td>Matlab</td>
</tr>
<tr>
<td>[178]</td>
<td>Stock of Tsugami Corporation</td>
<td>2013</td>
<td>Price data</td>
<td>LSTM</td>
<td>RMSE</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[179]</td>
<td>News, Nikkei Stock Average and 10-Nikkei companies</td>
<td>1999–2008</td>
<td>news, MACD</td>
<td>RNN, RBM+DBN</td>
<td>Accuracy, P-value</td>
<td>–</td>
</tr>
<tr>
<td>[180]</td>
<td>ISMIS 2017 Data Mining Competition dataset</td>
<td>–</td>
<td>Expert identifier, classes</td>
<td>LSTM + GRU + FFNN</td>
<td>Accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[182]</td>
<td>APPL from S&amp;P500 and news from Reuters</td>
<td>2011–2017</td>
<td>Input news, OCHLV, Technical indicators</td>
<td>CNN + LSTM, CNN+SVM</td>
<td>Accuracy, F1-score</td>
<td>Tensorflow</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Financial sentiment studies coupled with text mining without forecasting

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[85]</td>
<td>883 BHC from EDGAR</td>
<td>2006–2017</td>
<td>Tokens, weighted sentiment polarity, leverage and ROA</td>
<td>CNN, LSTM, SVM, Random Forest</td>
<td>Accuracy, Precision, Recall, F1-score</td>
<td>Keras, Python, Scikit-learn</td>
</tr>
<tr>
<td>[185]</td>
<td>SemEval-2017 dataset, financial text, news, stock market data</td>
<td>2017</td>
<td>Sentiments in Tweets, News headlines</td>
<td>Ensemble SVR, CNN, LSTM, GRU</td>
<td>Cosine similarity score, agreement score, class score</td>
<td>Python, Keras, Scikit Learn</td>
</tr>
<tr>
<td>[190]</td>
<td>StockTwits</td>
<td>2008–2016</td>
<td>Sentences, StockTwits messages</td>
<td>CNN, LSTM, GRU</td>
<td>MCC, WSURT</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[191]</td>
<td>Financial statements of Japan companies</td>
<td>–</td>
<td>Sentences, text</td>
<td>DMLLP</td>
<td>Precision, recall, f-score</td>
<td>–</td>
</tr>
<tr>
<td>[192]</td>
<td>Twitter posts, news headlines</td>
<td>–</td>
<td>Sentences, text</td>
<td>Deep-FASP</td>
<td>Accuracy, MSE, R²</td>
<td>–</td>
</tr>
<tr>
<td>[194]</td>
<td>News from Financial Times related US stocks</td>
<td>–</td>
<td>Sentiment of news headlines</td>
<td>SVR, Bidirectional LSTM</td>
<td>Cosine similarity</td>
<td>Python, Scikit Learn, Keras, Tensorflow</td>
</tr>
</tbody>
</table>

Deep learning for financial applications: Other text mining studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[86]</td>
<td>The event data set for large European banks, news articles from Reuters</td>
<td>2007–2014</td>
<td>Word, sentence</td>
<td>DMLP + NLP preprocess</td>
<td>Relative usefulness, F1-score</td>
<td>–</td>
</tr>
<tr>
<td>[87]</td>
<td>Event dataset on European banks, news from Reuters</td>
<td>2007–2014</td>
<td>Text, sentence</td>
<td>Sentence vector + DFFN</td>
<td>Usefulness, F1-score, AUROC</td>
<td>–</td>
</tr>
<tr>
<td>[121]</td>
<td>Real-world data for automobile insurance company labeled as fraudulent</td>
<td>–</td>
<td>Car, insurance and accident related features</td>
<td>DMLP + LDA</td>
<td>TP, FP, Accuracy, Precision, F1-score</td>
<td>–</td>
</tr>
<tr>
<td>[123]</td>
<td>Financial transactions</td>
<td>–</td>
<td>Transaction data</td>
<td>LSTM</td>
<td>t-SNE</td>
<td>–</td>
</tr>
<tr>
<td>[195]</td>
<td>Taiwan's National Pension Insurance</td>
<td>2008–2014</td>
<td>Insured’s id, area-code, gender, etc.</td>
<td>RNN</td>
<td>Accuracy, total error</td>
<td>Python</td>
</tr>
</tbody>
</table>
Deep learning for financial applications: Other theoretical or conceptual studies

<table>
<thead>
<tr>
<th>Art.</th>
<th>SubTopic</th>
<th>IsTimeSeries?</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>[197]</td>
<td>Analysis of AE, SVD</td>
<td>Yes</td>
<td>Selected stocks from the IBB index and stock of Amgen Inc.</td>
<td>2012–2014</td>
<td>Price data</td>
<td>AE, SVD</td>
</tr>
<tr>
<td>[198]</td>
<td>Fraud Detection in Banking</td>
<td>No</td>
<td>Risk Management / Fraud Detection</td>
<td>–</td>
<td>–</td>
<td>DRL</td>
</tr>
</tbody>
</table>

Deep learning for financial applications:

Other financial applications

<table>
<thead>
<tr>
<th>Art.</th>
<th>Subtopic</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>Improving trading decisions</td>
<td>S&amp;P500, KOSPI, HSI, and EuroStoxx50</td>
<td>1987–2017</td>
<td>200-days stock price</td>
<td>Deep Q-Learning and DMLP</td>
<td>Total profit, Correlation</td>
<td>–</td>
</tr>
<tr>
<td>193</td>
<td>Identifying Top Sellers In Underground Economy</td>
<td>Forums data</td>
<td>2004–2013</td>
<td>Sentences and keywords</td>
<td>Recursive neural tensor networks</td>
<td>Precision, recall, f-measure</td>
<td>–</td>
</tr>
<tr>
<td>195</td>
<td>Predicting Social Ins. Payment Behavior</td>
<td>Taiwan's National Pension Insurance</td>
<td>2008–2014</td>
<td>Insured's id, area-code, gender, etc.</td>
<td>RNN</td>
<td>Accuracy, total error</td>
<td>Python</td>
</tr>
<tr>
<td>199</td>
<td>Speedup</td>
<td>45 CME listed commodity and FX futures</td>
<td>1991–2014</td>
<td>Price data</td>
<td>DNN</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>200</td>
<td>Forecasting Fundamentals</td>
<td>Stocks in NYSE, NASDAQ or AMEX exchanges</td>
<td>1970–2017</td>
<td>16 fundamental features from balance sheet</td>
<td>DMLP, LFM</td>
<td>MSE, Compound annual return, SR</td>
<td>–</td>
</tr>
<tr>
<td>201</td>
<td>Predicting Bank Telemarketing</td>
<td>Phone calls of bank marketing data</td>
<td>2008–2010</td>
<td>16 finance-related attributes</td>
<td>CNN</td>
<td>Accuracy</td>
<td>–</td>
</tr>
</tbody>
</table>

First and foremost, we clustered the various topics within the financial applications research and presented them in Fig. 8. A quick glance at the figure shows us financial text mining and algorithmic trading are the top two fields that the researchers most worked on followed by risk assessment, sentiment analysis, portfolio management and fraud detection, respectively. The results indicate most of the papers were published within the past 3 years implying the domain is very hot and actively studied.

When the papers were clustered by the DL model type as presented in Fig. 9, we observe the dominance of RNN, DMLP and CNN over the remaining models, which might be expected, since these models are the most commonly preferred ones in general DL implementations. Meanwhile, RNN is a general umbrella model which has several versions including LSTM, GRU, etc. Within the RNN choice, most of the models actually belonged to LSTM, which is very popular in time series forecasting or regression problems. It is also used quite often in algorithmic trading. More than 70% of the RNN papers consisted of LSTM models.

Meanwhile, DMLP generally fits well for classification problems; hence it is a common choice for most of the financial application areas. However, since it is a natural extension of its shallow counterpart MLP, it has a longer history than the other DL models.

CNN started getting more attention lately since most of the implementations appeared within the past 3 years. Careful analysis of CNN papers indicates that a recent trend of representing financial data with a 2-D image view in order to utilize CNN is growing. Hence CNN based models might overpass the other models in the future. It actually passed DMLP for the past 3 years. Furthermore, we attempted to provide more details about associations between the DL models and the financial application areas.

Fig. 10 gives the distribution of the models for the research areas through a model-topic heatmap. Since most of the papers had multiple DL models, the amount of models is more than the number of covered papers. The results indicate the broad acceptance of RNN, DMLP and CNN models in almost all financial application areas.

We also wanted to elaborate on the particular feature selections for each financial application area to see if we could spot any pattern. Fig. 11 gives the distribution of the features for the research areas through a feature-topic heatmap. Unlike...
## 6. Discussion and open issues

Even though financial time series forecasting is a subset of time-series forecasting, it has a relatively narrow focus, i.e., the implementations covered in this survey. Regardless, Python-related tools were the most influential within the ‘Other’ section, the usage of Pytorch has increased in the last year or so, even though it is not visible from the images. No further information was given, while some others mentioned the use of Keras or TensorFlow, providing more details. Also, no further information was given, while some researchers claimed they used Python, but did not give details, preventing us from a more thorough comparison provided their development environment. Also, most papers did not mention the programming language or framework used, which makes it difficult to compare the performances of different models. However, we must keep in mind that not every published paper will provide this information.

### 6.1. DL models for financial time series forecasting

The most commonly used models were DMLP and RNN, which mainly consist of LSTM. As a matter of fact, more than 70% of the papers used LSTM, and it was the natural choice in financial time series forecasting. Meanwhile, LSTM is a special DL model derived from a more general classifier family, namely RNN. However, in general, data order independence must be preserved if the RNN model category. Regardless of its problem type, price, or trend prediction, the ordinal nature of the data representation forced researchers to consider RNN, GRU, and LSTM as viable preferences for their model choices. Hence, RNN models were chosen, at least for benchmarking, in many studies for classification-type financial time series forecasting implementations. Because most financial time series prediction problems were chosen, at least for benchmarking, in many studies for classification-type financial time series forecasting implementations.

From an application perspective, even though financial time series forecasting is a subset of time-series forecasting problems, the usage of DMLP and RNN was more than 70% of the papers. Meanwhile, other models were also used for time series forecasting problems. Among those, DMLP had the most interest due to the market dominance of its shallow cousin (MLP) and its wide acceptance and long history within ML society. However, there is a fundamental difference in how DMLP- and RNN-based models accept input data. DMLP fits well for both regression and classification problems. In contrast, RNNs perform better with time series data, where the order of elements is crucial. As a result, CNN might increase its share of interest for financial time series forecasting as long as the data becomes stationary. Regardless, some careful preprocessing might be needed so that the resulting time series data becomes stationary.

Careful analysis of the time series data is necessary for a DMLP model to be successful. In contrast, RNNs are more robust to time series data, which are often non-stationary. However, in general, data order independence must be preserved if the RNN model category. Regardless of its problem type, price, or trend prediction, the ordinal nature of the data representation forced researchers to consider RNN, GRU, and LSTM as viable preferences for their model choices. Hence, RNN models were chosen, at least for benchmarking, in many studies for classification-type financial time series forecasting implementations.

As one final note, although financial time series forecasting is a subset of time-series forecasting problems, the usage of DMLP and RNN was more than 70% of the papers. Meanwhile, other models were also used for time series forecasting problems. Among those, DMLP had the most interest due to the market dominance of its shallow cousin (MLP) and its wide acceptance and long history within ML society. However, there is a fundamental difference in how DMLP- and RNN-based models accept input data. DMLP fits well for both regression and classification problems. In contrast, RNNs perform better with time series data, where the order of elements is crucial. As a result, CNN might increase its share of interest for financial time series forecasting as long as the data becomes stationary. Regardless, some careful preprocessing might be needed so that the resulting time series data becomes stationary.

### Topic-model heatmap

![Fig. 7. Topic-model heatmap.](image)

![Fig. 13](image)

![Fig. 8. The histogram of publication count in years.](image)

### Image references

- Fig. 7: Topic-model heatmap.
- Fig. 13: [Image]
- Fig. 8: The histogram of publication count in years.

Stock price forecasting using only raw time series data

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Lag</th>
<th>Horizon</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[80]</td>
<td>38 stocks in KOSPI</td>
<td>2010–2014</td>
<td>Lagged stock returns</td>
<td>50 min</td>
<td>5 min</td>
<td>DNN</td>
<td>NMSE, RMSE, MAE, MI</td>
<td>–</td>
</tr>
<tr>
<td>[83]</td>
<td>Daily returns of ‘BRD’ stock in Romanian Market</td>
<td>2012–2013</td>
<td>OCHLV, Price data, turnover and number of trades.</td>
<td>200 d</td>
<td>1..10 d</td>
<td>LSTM, RNN, CNN, MLP</td>
<td>MAPE</td>
<td>–</td>
</tr>
<tr>
<td>[84]</td>
<td>Stocks of Infosys, TCS and CIPLA from NSE</td>
<td>2014</td>
<td>Price data</td>
<td>–</td>
<td>–</td>
<td>RNN, LSTM and CNN</td>
<td>Accuracy</td>
<td>–</td>
</tr>
<tr>
<td>[85]</td>
<td>10 stocks in S&amp;P500</td>
<td>1997–2016</td>
<td>OCHLV, Price data</td>
<td>36 m</td>
<td>1 m</td>
<td>RNN, LSTM, GRU</td>
<td>Accuracy, Monthly return</td>
<td>Keras, Tensorflow</td>
</tr>
<tr>
<td>[87]</td>
<td>High-frequency transaction data of the CSI300 futures</td>
<td>2017</td>
<td>Price data</td>
<td>–</td>
<td>1 min</td>
<td>DNN, ELM, RBF</td>
<td>RMSE, MAE, Accuracy</td>
<td>Matlab</td>
</tr>
<tr>
<td>[93]</td>
<td>12 stocks from SSE Composite Index</td>
<td>2000–2017</td>
<td>OCHLV</td>
<td>60 d</td>
<td>1.7 d</td>
<td>DWNN</td>
<td>MSE</td>
<td>–</td>
</tr>
<tr>
<td>[94]</td>
<td>50 stocks from NYSE</td>
<td>2007–2016</td>
<td>Price data</td>
<td>–</td>
<td>1d, 3 d, 5 d</td>
<td>SFM</td>
<td>MSE</td>
<td>–</td>
</tr>
</tbody>
</table>

Stock price forecasting using various data

<table>
<thead>
<tr>
<th>Art.</th>
<th>Data set</th>
<th>Period</th>
<th>Feature set</th>
<th>Lag</th>
<th>Horizon</th>
<th>Method</th>
<th>Performance criteria</th>
<th>Env.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[97]</td>
<td>U.S. low-level disaggregated macroeconomic time series</td>
<td>1926–2016</td>
<td>Fundamental Features: GDP, Unemployment rate, Inventories, etc.</td>
<td>–</td>
<td>1 s</td>
<td>DNN</td>
<td>MSPE</td>
<td>Tensorflow</td>
</tr>
<tr>
<td>[99]</td>
<td>Stock of Tsunami Corporation Stocks in China’s A-share SCI prices</td>
<td>2013</td>
<td>Price data</td>
<td>–</td>
<td>–</td>
<td>LSTM</td>
<td>RMSE</td>
<td>–</td>
</tr>
<tr>
<td>[100]</td>
<td>10 stocks in Nikkei 225 and news TKC stock in NYSE and QQQQ ETF 10 Stocks in NYSE</td>
<td>2013</td>
<td>Price data, Technical indicators</td>
<td>50 d</td>
<td>1 d</td>
<td>RNN (Jordan–Elman) LSTM, MLP</td>
<td>Profit, MSE</td>
<td>Java</td>
</tr>
<tr>
<td>[101]</td>
<td>10 Stocks in NYSE</td>
<td>–</td>
<td>Technical indicators, Price</td>
<td>20 min</td>
<td>1 min</td>
<td>LSTM, MLP</td>
<td>RMSE</td>
<td>–</td>
</tr>
<tr>
<td>[103]</td>
<td>–</td>
<td>250 features: order details, etc.</td>
<td>–</td>
<td>–</td>
<td>CNN, LSTM</td>
<td>F1-Score</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>[104]</td>
<td>–</td>
<td>Fundamental, technical and market information</td>
<td>–</td>
<td>–</td>
<td>CNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>[105]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>WMTR, MDA</td>
<td>Accuracy, Precision, Recall, F1-Score</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>[106]</td>
<td>–</td>
<td>57 firm characteristics</td>
<td>–</td>
<td>–</td>
<td>Fama–French n-factor model DL</td>
<td>R², RMSE</td>
<td>Tensorflow</td>
<td></td>
</tr>
</tbody>
</table>

Stock Market Movement Forecast: Phases of the stock market modeling

Python in Google Colab (Python101)

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

Deep Learning for Financial Time Series Forecasting


```python
1 '''
2 Time series
3 [1,2,3,4,5,6]
4 X, y
5 [1, 2, 3] 4
6 [2, 3, 4] 5
7 [3, 4, 5] 6
8 '''

'\nTime series\n[1,2,3,4,5,6]\nX, t\ny\n[1, 2, 3] 4\n[2, 3, 4] 5\n[3, 4, 5] 6'''

[ ] 1 # univariate data preparation
2 from numpy import array
3 # split a univariate sequence into samples
4 def split_sequence(sequence, n_steps):
5     X, y = list(), list()
6     for i in range(len(sequence)):
7         # find the end of this pattern
8         end_ix = i + n_steps
9         # check if we are beyond the sequence
10        if end_ix > len(sequence)-1:
11            break
12        # gather input and output parts of the pattern
13        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
14        X.append(seq_x)
15        y.append(seq_y)

https://tinyurl.com/aintpupython101
O’Reilly Media, 2019

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

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

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

https://github.com/ageron/handson-ml2
Sequences using RNNs and CNNs

```python
np.random.seed(43)

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

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

An end-to-end open source machine learning platform

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

Get started with TensorFlow

https://www.tensorflow.org/
TensorFlow

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

https://www.tensorflow.org/
Why TensorFlow 2.0

Why TensorFlow

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

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

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

Powerful experimentation for research
A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.
# TensorFlow 2.0
outputs = f(input)

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

Source: [https://www.tensorflow.org/guide/effective_tf2](https://www.tensorflow.org/guide/effective_tf2)
```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

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

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

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

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

[https://www.tensorflow.org/overview/](https://www.tensorflow.org/overview/)
TensorFlow 2 quickstart for beginners

This short introduction uses Keras to:

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

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

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

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

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

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

Basic classification: Classify images of clothing

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

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

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

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

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

print(tf.__version__)
```

https://www.tensorflow.org/tutorials/keras/classification
Image Classification
Fashion MNIST dataset

https://www.tensorflow.org/tutorials/keras/classification
Text classification with TensorFlow Hub: Movie reviews

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

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

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

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

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

https://www.tensorflow.org/tutorials/keras/text_classification_with_hub
Text classification with preprocessed text: Movie reviews

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

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

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

Setup

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

Basic regression: Predict fuel efficiency

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

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

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

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

from __future__ import absolute_import, division, print_function, unicode_literals
import pathlib
```
Time series forecasting

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

```python
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

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

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

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

\[ [100, 110, 120, 130, 140, 150] \]
Long Short Term Memory (LSTM) for Time Series Forecasting

\[ X_t \rightarrow \text{LSTM} \rightarrow \text{LSTM} \rightarrow \text{LSTM} \rightarrow \text{LSTM} \rightarrow \text{LSTM} \]

\[ h_{t-2} \rightarrow h_{t-1} \rightarrow h_t \rightarrow h_{t+1} \rightarrow h_{t+2} \]

\[ X_{t-2} \rightarrow X_{t-1} \rightarrow X_t \rightarrow X_{t+1} \rightarrow X_{t+2} \]
**Time Series Data**

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

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[10 20 30]</code></td>
<td>40</td>
</tr>
<tr>
<td><code>[20 30 40]</code></td>
<td>50</td>
</tr>
<tr>
<td><code>[30 40 50]</code></td>
<td>60</td>
</tr>
<tr>
<td><code>[40 50 60]</code></td>
<td>70</td>
</tr>
<tr>
<td><code>[50 60 70]</code></td>
<td>80</td>
</tr>
<tr>
<td><code>[60 70 80]</code></td>
<td>90</td>
</tr>
</tbody>
</table>
Deep Learning and Neural Networks
Deep Learning Foundations:
Neural Networks
Deep Learning and Neural Networks

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

X1  X2

Y
Deep Learning and Neural Networks

Input Layer (X)  Hidden Layer (H)  Output Layer (Y)
Deep Learning and Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

Deep Neural Networks
Deep Learning
Deep Learning
and
Deep Neural Networks
LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton.

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

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

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, conventional systems were feedforward and used a single layer of non-linearities; they were thereby limited to capturing relatively simple and smooth functions of the input. More recent approaches incorporate many layers of non-linearities that can model more complex functions. Interpreting the model is still a challenge, however, because the parameters in each layer of the network are coupled in a complex way. Deep learning, in its many forms, is perhaps the most successful and influential modern approach to machine learning.
Deep Learning

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

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

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

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Neural Networks (NN)
Convolutional Neural Networks
(CNN or Deep Convolutional Neural Networks, DCNN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Recurrent Neural Networks (RNN)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Long / Short Term Memory (LSTM)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Gated Recurrent Units (GRU)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Generative Adversarial Networks (GAN)

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

Support Vector Machines (SVM)


Source: http://www.asimovinstitute.org/neural-network-zoo/
Neural networks (NN) 1960

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

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

X1  X2

Source: https://www.youtube.com/watch?v=bxetV8XRsl
Multilayer Perceptrons (MLP) 1985

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Support Vector Machine (SVM) 1995

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

Deep Belief Network (DBN)

New interests in deep learning and RBM

State of the art MNIST 2005

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Deep Recurrent Neural Network (RNN) 2009

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Convolutional DBN
2010

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Max-Pooling CDBN 2011

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
From image to text

A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

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

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

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

A group of **people** sitting on a boat in the water.
Convolutional Neural Networks (CNN)
Convolutional Neural Networks (CNN)


Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)
Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)
A friendly introduction to Convolutional Neural Networks and Image Recognition

Convolution Layer

Pooling Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-O7ZB0MmU
A friendly introduction to Convolutional Neural Networks and Image Recognition

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

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7Z80MmU
A friendly introduction to Convolutional Neural Networks and Image Recognition

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

**CNN Convolution Layer**

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

**3x3 convolution**

Input

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Filter / Kernel

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**CNN Convolution Layer**

*Input x Filter --> Feature Map*

receptive field: 3x3

<table>
<thead>
<tr>
<th>1x1</th>
<th>1x0</th>
<th>1x1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>1x1</td>
<td>1x0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x1</td>
<td>0x0</td>
<td>1x1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**CNN Convolution Layer**

**Input x Filter --> Feature Map**

receptive field: 3x3

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1x1</th>
<th>1x0</th>
<th>0x1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input x Filter**

**Feature Map**

### CNN Convolution Layer

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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>1x0</td>
<td>1x1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>1x1</td>
<td>1x0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x1</td>
<td>0x0</td>
<td>1x1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

CNN Convolution Layer

10 different filters  10 feature maps of size 32x32x1

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

CNN Convolution Layer
Sliding operation at 4 locations

CNN Convolution Layer

two feature maps
**CNN Convolution Layer**

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

---

**Stride 1**

**Feature Map**

---

**CNN Convolution Layer**

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

---

**Stride 2**

**Feature Map**

CNN Convolution Layer

Stride 1 with Padding

Feature Map

CNN Pooling Layer

Max Pooling

max pool with 2x2 window and stride 2

CNN Pooling Layer

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

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

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

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

Dropout

No Dropout                      With Dropout

Model Performance

Train Loss: 0.054, Val Loss: 1.345

Starts Overfitting

Train Accuracy: 0.981, Val Accuracy: 0.732

Visual Recognition

Image Classification
IS THIS A CAT or DOG?

CAT   DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP NEURAL NETWORK

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

(CNNs / ConvNets)

http://cs231n.github.io/convolutional-networks/
A regular 3-layer Neural Network

http://cs231n.github.io/convolutional-networks/
A ConvNet arranges its neurons in three dimensions (width, height, depth)

http://cs231n.github.io/convolutional-networks/
The activations of an example ConvNet architecture.
ConvNets

32x32x3 CIFAR-10 image

first Convolutional layer

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

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

Input Volume (+pad 1) (7x7x3)  Filter W0 (3x3x3)  Filter W1 (3x3x3)  Output Volume (3x3x2)

\[
\begin{array}{ccc}
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & 2 & 1 & 0 \\
0 & 2 & 2 & 2 & 1 & 1 & 0 \\
0 & 2 & 2 & 2 & 0 & 1 & 0 \\
0 & 2 & 2 & 1 & 2 & 1 & 0 \\
0 & 2 & 1 & 2 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
& \begin{bmatrix}
-1 & -1 & 0 \\
1 & 1 & 1 \\
-1 & 0 & 1 \\
0 & 1 & 1 \\
-1 & -1 & 1 \\
0 & -1 & 1 \\
-1 & 0 & 1
\end{bmatrix}
& \begin{bmatrix}
w[0][1] \\
w[0][2] \\
-1 & -1 & 0 \\
1 & 0 & -1 \\
0 & 0 & 0
\end{bmatrix}
& \begin{bmatrix}
w[1][1] \\
w[1][2] \\
0 & 1 & 1 \\
-1 & -1 & 1 \\
0 & 0 & 0
\end{bmatrix}
& \begin{bmatrix}
6 & 3 & 6 \\
7 & -1 & -2 \\
2 & 3 & -2 \\
4 & 3 & 2 \\
-1 & 0 & -1
\end{bmatrix}
\end{array}
\]

\[\text{Bias b0 (1x1x1)}\]

\[\begin{bmatrix}
1
\end{bmatrix}\]

\[\text{Bias b1 (1x1x1)}\]

\[\begin{bmatrix}
0
\end{bmatrix}\]

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

input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$

http://cs231n.github.io/convolutional-networks/
ConvNets
max pooling

http://cs231n.github.io/convolutional-networks/
Convolutional Neural Networks (CNN) (LeNet)

Source: [http://deeplearning.net/tutorial/lenet.html](http://deeplearning.net/tutorial/lenet.html)
Recurrent Neural Networks (RNN)
Recurrent Neural Networks (RNN)
Recurrent Neural Networks (RNN)
Time Series Forecasting

\[ X_t - 1 \]
\[ y_t \]
\[ h_t \]

\[ X_t - 2 \]
\[ y_{t-1} \]
\[ h_{t-1} \]

\[ X_t \]
\[ y_t \]
\[ h_t \]

\[ X_{t+1} \]
\[ y_{t+1} \]
\[ h_{t+1} \]

\[ X_{t+2} \]
\[ y_{t+2} \]
\[ h_{t+2} \]
Recurrent Neural Networks (RNN)
Recurrent Neural Networks (RNN) Sentiment Analysis

This movie is very good

Input: $X_t, X_{t-1}, X_{t-2}$
Hidden: $h_t, h_{t-1}, h_{t-2}$
Output: $y$
Recurrent Neural Networks (RNN)
Sentiment Analysis

\[ X_t \rightarrow h_t \rightarrow h_{t+1} \rightarrow h_{t+2} \]

**Input:**
- \( X_{t-2} \): This
- \( X_{t-1} \): movie
- \( X_t \): is
- \( X_{t+1} \): very
- \( X_{t+2} \): boring

**Output:**
- \( y \)
Recurrent Neural Network (RNN)

Course Description

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

\[ h_{t-1} \rightarrow y_{t-1} \]
\[ h_{t-1} \rightarrow h_t \]
\[ W \]
\[ x_{t-1} \rightarrow x_t \]
\[ y_t \]
\[ h_t \rightarrow y_t \]
\[ W \]
\[ x_t \rightarrow x_{t+1} \]
\[ h_{t+1} \rightarrow y_{t+1} \]

RNN long-term dependencies

I grew up in France… I speak fluent French.

Vanishing Gradient
Exploding Gradient
Recurrent Neural Networks (RNN)
RNN
Vanishing Gradient problem
Exploding Gradient problem

Error

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

RNN

Vanishing Gradient problem

W = 0.9 < 1 (Vanishing)

RNN
Exploding Gradient problem

W = 1.1 > 1 (Exploding)

RNN LSTM

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU)

reset gate  update gate

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.

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)

LSTM

Memory state (C)
LSTM

**forget gate (f)**

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

LSTM

input gate (i)

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

**LSTM**

**Memory state (C)**

\[ C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \]

**LSTM**

**output gate (o)**

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

\[
h_t = o_t \times \text{tanh}(C_t)
\]

**LSTM**

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

\[
\begin{align*}
    f_t &= \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \\
    i_t &= \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \\
    o_t &= \sigma (W_o \cdot [C_t, h_{t-1}, x_t] + b_o)
\end{align*}
\]
Gated Recurrent Unit (GRU)

update \( (z) \), reset \( (r) \) gates

\[
\begin{align*}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W \cdot [r_t \ast h_{t-1}, x_t]) \\
h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]

LSTM Recurrent Neural Network

- Traditional Neural Network
- Music Generation
- Sentiment Classification
- Name Entity Recognition
- Machine Translation

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent
Long Short Term Memory (LSTM) for Time Series Forecasting

\[ X_t, X_{t-1}, h_t, h_{t-1}, h_{t+1}, h_{t+2}, X_{t-2}, X_{t+1}, X_{t+2} \]
The Sequence to Sequence model (seq2seq)
Sequence to Sequence (Seq2Seq)

Source: https://google.github.io/seq2seq/
Transformer (Attention is All You Need) (Vaswani et al., 2017)

Transformer

INPUT
Je suis étudiant

THE TRANSFORMER

OUTPUT
I am a student

Transformer
Encoder Decoder

INPUT Je suis étudiant

OUTPUT I am a student

Transformer
Encoder Decoder Stack

Transformer
Encoder Self-Attention

Transformer Decoder

Transformer Encoder with Tensors
Word Embeddings

Transformer
Self-Attention Visualization

Transformer
Positional Encoding Vectors

Transformer Self-Attention Softmax Output

Input
Embedding
Queries
Keys
Values
Score
Divide by $8 \sqrt{d_k}$
Softmax
Softmax $X$
Value
Sum

Source: Jay Alammar (2019), The Illustrated Transformer, [http://jalammar.github.io/illustrated-transformer/]
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

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

BERT (Bidirectional Encoder Representations from Transformers)

BERT input representation

BERT, OpenAI GPT, ELMo

Fine-tuning BERT on Different Tasks

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

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

(c) Question Answering Tasks: SQuAD v1.1

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

Illustrated BERT

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

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

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

**Semi-supervised Learning Step**

Model: BERT


Objective: Predict the masked word (language modeling)

**Supervised Learning Step**

Classifier

75% Spam
25% Not Spam

Model: BERT (pre-trained in step #1)

Dataset:

<table>
<thead>
<tr>
<th>Email message</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy these pills</td>
<td>Spam</td>
</tr>
<tr>
<td>Win cash prizes</td>
<td>Spam</td>
</tr>
<tr>
<td>Dear Mr. Atreides, please find attached...</td>
<td>Not Spam</td>
</tr>
</tbody>
</table>

BERT Classification Input Output

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

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

85% Spam
15% Not Spam

Classifier
(Feed-forward neural network + softmax)

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

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

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

\[ x_1 \]
\[ x_2 \]
\[ \ldots \]
\[ x_n \]

\[ w_1 \]
\[ w_2 \]
\[ \ldots \]
\[ w_n \]

\[ y \]

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

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

\[ y = F \left( \sum_i w_i x_i \right) \]

\[ F(x) = \max(0, x) \]

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
\[ y = \max\left(0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3\right) \]
Neural Networks

Source: https://www.youtube.com/watch?v=bx8cA0r8vXRs&index=1&list.PLiaHhY2iBXdHaRt7b7XevZtgZRa1PoU
Neural Networks

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

X1

X2

Source: https://www.youtube.com/watch?v=bxT2-V8XR&index=1&list=PLiaHhY2iBX9hdHaR6b7XevZtgZRa1PoU
Neural Networks

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

Deep Neural Networks
Deep Learning

Source: https://www.youtube.com/watch?v=bxet-V8XRs&index=1&list=PLiaHhY2lBX9hdHaRt6b7XevZtgZRa1PoU
Neural Networks

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

Neuron

Synapse

X1

X2

Synapse

Neuron

Y

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2IBXh4hHaRr6b7XevZtgZRa1PoU
Neural Networks

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

![Neural Network Diagram]

- Hours
- Sleep
- Hours
- Study

Score

Source: https://www.youtube.com/watch?v=bx2T-V8XRses&index=1&list=PLiaHhY2l1BX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

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

Source: https://www.youtube.com/watch?v=bxet-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtZRa1PoU
<table>
<thead>
<tr>
<th>Hours Sleep</th>
<th>Hours Study</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>

Source: https://www.youtube.com/watch?v=bx2T-V8XR&index=1&list=PLiaHhY2lBX9hdHaRt6b7XevZtgZRa1PoU
<table>
<thead>
<tr>
<th>Training</th>
<th>Hours</th>
<th>Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing</th>
<th>Hours</th>
<th>Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>
\[ Y = W X + b \]
\[ Y = WX + b \]

Output \rightarrow W \rightarrow X \rightarrow Y

Weights \rightarrow bias

Trained

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
\[ W \mathbf{X} + \mathbf{b} = \mathbf{Y} \]

Scores → Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzOgqE
SoftMAX

\[ W X + b = Y \]

Logits

Scores

Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
\[ S(y_i) = \frac{e^{y_i}}{\sum_{j} e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7 \]

\[ S(y_i) = \frac{e^{y_i}}{\sum_{j} e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2 \]

\[ S(y_i) = \frac{e^{y_i}}{\sum_{j} e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1 \]

\[ W \times X + b = Y \]

Logits \rightarrow Scores \rightarrow Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOqqE
Training a Network

= Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxelT-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRte79XevZtgZRa1PoU
Training a Network

= Minimize the **Cost** Function

Minimize the **Loss** Function
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bx2T-V8XR&s=index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bx2T-V8XR&index=1&list=PLiaHhY2lBX9hdHaR6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
Activation Functions
Activation Functions

Sigmoid

TanH

ReLU (Rectified Linear Unit)

\[ f(x) = \max(0, x) \]
Activation Functions

Sigmoid: \( f(x) = \frac{1}{1 + e^{-x}} \)

Tanh: \( \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \)

ReLU: \( f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \)

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.

Learning Algorithm

While not done:

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

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
\[ y = \max(0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3) \]
Next time:

\[ y = \max ( \ 0, \ -0.23 \cdot x_1 + 0.31 \cdot x_2 + 0.65 \cdot x_3 ) \]

\[ y = \max ( \ 0, \ -0.21 \cdot x_1 + 0.3 \cdot x_2 + 0.7 \cdot x_3 ) \]

Weights
Optimizer:
Stochastic Gradient Descent (SGD)
This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Gradient Descent
how neural networks learn

Average cost of all training data...

Cost of $8$

\[
(0.18 - 0.00)^2 + \\
(0.29 - 0.00)^2 + \\
(0.58 - 0.00)^2 + \\
(0.77 - 0.00)^2 + \\
(0.20 - 0.00)^2 + \\
(0.36 - 0.00)^2 + \\
(0.93 - 0.00)^2 + \\
(1.00 - 0.00)^2 + \\
(0.95 - 1.00)^2 + \\
(0.35 - 0.00)^2
\]

What’s the “cost” of this difference?

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

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

While not done:

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

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Financial Time Series Forecasting
Time Series Data

![Graph of AAPL with different moving averages (MA05, MA20, MA60) over a time period from 2016-02 to 2018-10.](image-url)
Time Series Data

\[ [100, 110, 120, 130, 140, 150] \]
Deep Learning with TensorFlow
Deep Learning Software

• TensorFlow
  – TensorFlow™ is an open source software library for high performance numerical computation.

• Keras
  – Deep Learning library for TensorFlow, CNTK

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

• CNTK
  – Computational Network Toolkit by Microsoft Research

Source: http://deeplearning.net/software_links/
Keras: High-level API for TensorFlow
Keras: The Python Deep Learning library

You have just found Keras.

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

Use Keras if you need a deep learning library that:

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

Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.6.

http://keras.io/
FROM RESEARCH TO PRODUCTION

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

http://pytorch.org/
Keras is a high-level neural networks API.

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

Source: https://keras.io/
TensorFlow

An end-to-end open source machine learning platform

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

Get started with TensorFlow

https://www.tensorflow.org/
TensorFlow

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

https://www.tensorflow.org/
import tensorflow as tf
mnist = tf.keras.datasets.mnist

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

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

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

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

https://www.tensorflow.org/overview/
TensorFlow 2 quickstart for beginners

This short introduction uses Keras to:

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

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

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

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

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

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

Time Series Forecasting

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

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

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

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

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

http://playground.tensorflow.org/
TensorFlow is an Open Source Software Library for Machine Intelligence

https://www.tensorflow.org/
Tensor

• 3
  – # a rank 0 tensor; this is a **scalar** with shape []

• [1., 2., 3.]
  – # a rank 1 tensor; this is a **vector** with shape [3]

• [[1., 2., 3.], [4., 5., 6.]]
  – # a rank 2 tensor; a **matrix** with shape [2, 3]

• [[[1., 2., 3.]], [[7., 8., 9.]]]
  – # a rank 3 **tensor** with shape [2, 1, 3]

https://www.tensorflow.org/
Scalar

Vector

Matrix

Tensor
TensorFlow
Deep Learning for Financial Time Series Forecasting
Deep Learning for Financial Market Prediction

Stock Market Prediction

Stock Price Prediction

Time Series Prediction
Time Series Data

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

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

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

\[
\begin{align*}
&X_t1 & X_t2 & X_t3 & X_t4 & X_t5 \\
[100 & 110 & 120 & 130 & 140]
\end{align*}
\]

Y

150
Long Short Term Memory (LSTM) for Time Series Forecasting

\[ X_t, X_{t-1}, h_{t-2}, h_{t-1}, h_t, h_{t+1}, h_{t+2} \]

\[ LSTM, LSTM, LSTM, LSTM, LSTM \]
Time Series Data

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

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10, 20, 30]</td>
<td>40</td>
</tr>
<tr>
<td>[20, 30, 40]</td>
<td>50</td>
</tr>
<tr>
<td>[30, 40, 50]</td>
<td>60</td>
</tr>
<tr>
<td>[40, 50, 60]</td>
<td>70</td>
</tr>
<tr>
<td>[50, 60, 70]</td>
<td>80</td>
</tr>
<tr>
<td>[60, 70, 80]</td>
<td>90</td>
</tr>
</tbody>
</table>
Deep Learning for Financial Time Series Forecasting

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

```
# univariate data preparation
from numpy import array
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# summarize the data
for i in range(len(X)):
    print(X[i], y[i])
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

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

LSTM for Time Series Forecasting

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

# define dataset
X = array([[100, 110, 120], [110, 120, 130], [120, 130, 140], [130, 140, 150], [140, 150, 160]])
y = array([130, 140, 150, 160, 170])
# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))

# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# fit model
history = model.fit(X, y, epochs=2000, verbose=0)

# demonstrate prediction
x_input = array([150, 160, 170])
x_input = x_input.reshape((1, 3, 1))
yhat = model.predict(x_input, verbose=0)
print('yhat', yhat)

print(model.summary())
# list all data in history
print(history.history.keys())
# summarize history for loss
print('loss: ', history.history['loss'][-1])
print('loss: ', history.history['val_loss'][-1])
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()

```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

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

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

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

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

Using TensorFlow backend.

```
[[102.31296]]
yhat [[102.31296]]
```

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lstm_1 (LSTM)</td>
<td>(None, 50)</td>
<td>10400</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 1)</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total params: 10,451</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trainable params: 10,451</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-trainable params: 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
Deep Learning for Financial Time Series Forecasting

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

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

Fashion MNIST Image Classification

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

Train your first neural network: basic classification

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

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.
Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i_gror

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

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

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

```
# memory footprint support libraries/code
!ln -s /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
```
Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgD2_25K_AURjR6SCVBRdlj

Copyright 2018 The TensorFlow Authors.

Predict house prices: regression

The Boston Housing Prices dataset

Examples and features

Labels

Normalize features

Create the model

Train the model

Predict

Conclusion

Copyright 2018 The TensorFlow Authors.

→ 2 cells hidden

Predict house prices: regression

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

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

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

```python
# memory footprint support libraries/code
!ln -s f /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def print():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( process.virtual_memory().available ), " | Proc size: ", process.memory_info().rss)
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(2017, 1, 1)
df = web.DataReader('AAPL', 'yahoo', start, end)
df.to_csv('AAPL.csv')
print(df.tail())
df2 = pd.read_csv('AAPL.csv')
print(df2.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')
bottom.plot(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)
np.where
(df['MA20'] > df['MA60'], 12000, 9000)

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

df['Positions'] = np.where(df['MA20'] > df['MA60'], 12000, 9000)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Positions']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL', secondary_y='Positions').legend(bbox_to_anchor=(1.2, 0.5))

https://tinyurl.com/aintpuppython101
np.where
(df['MA20'] > df['MA60'],
1,
0)

# simple moving averages
def['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
def['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
def['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
def['Positions'] = np.where(df['MA20'] > df['MA60'], 1, 0)
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60'], 'Positions': df['Positions']})
Yves Hilpisch (2018),
Python for Finance: Mastering Data-Driven Finance, O'Reilly

https://github.com/yhilpisch/py4fi2nd

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

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

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

https://github.com/ageron/handson-ml2
Notebooks
1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training andDeploying TensorFlow Models at Scale

https://github.com/ageron/handson-ml2
Papers with Code
State-of-the-Art (SOTA)

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code
Follow on Twitter for updates

Computer Vision

- Semantic Segmentation
  - 33 leaderboards
  - 667 papers with code
- Image Classification
  - 52 leaderboards
  - 564 papers with code
- Object Detection
  - 54 leaderboards
  - 467 papers with code
- Image Generation
  - 51 leaderboards
  - 231 papers with code
- Pose Estimation
  - 40 leaderboards
  - 231 papers with code

See all 707 tasks

Natural Language Processing

- Machine Translation
- Language Modelling
- Question Answering
- Sentiment Analysis
- Text Generation

https://paperswithcode.com/sota
Papers with Code
Stock Market Prediction

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

Subtasks
- Stock Price Prediction: 3 papers with code
- Stock Trend Prediction: 2 papers with code
- Stock Prediction: 1 paper with code

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

• Deep Learning for Finance Big Data Analysis with TensorFlow
  – Deep Learning
  – Financial Time Series Forecasting
  – TensorFlow
References

- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5lCeMbQuqbbCOQWCs6OYBr5A
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHh1vSEYgvGod9wWiydumYI8hOXiXNu
- 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
- Jay Alammar (2019), The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning), http://jalammar.github.io/illustrated-bert/
- TensorFlow: https://www.tensorflow.org/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPF79uXLuQrlmzHfOV
- https://github.com/leriomaggio/deep-learning-keras-tensorflow