Capstone Project

Machine Learning Engineer Nanodegree

Yves D'hondt **August 12th, 2022**

Definition

Project Overview

Stock markets are the second largest public market, valued at roughly $70 trillion (cmc markets, n.d.). With a market that large and a lot of money to be gained, it is no surprise that stock price prediction is a big topic. Traditionally, stock price prediction has been analyzed through *factors*. In essence, factors are common sources of risks (& returns) among different stocks. Fama & French defined the de facto baseline factor model by identifying 5 different factors (Fama & French, 2015). They identified the following 5 factors: market returns, size, value, profitability & investment patterns.

Looking for new sources of stock predictions, researchers have turned to analysing stock chart images using machine learning. Jin & Kwon investigate the impact of chart image characteristics on stock price returns using CNNs (Jin & Kwon, 2021). Ozbayoglu & Sezer also investigate the potential of using CNN models on bar chart images (CNN-BI) to predict stock returns (Ozbayoglu & Sezer, 2019).

This project aims to build a CNN model in line with this body of research that predicts stock returns.

Problem Statement

This project investigates whether CNNs can be successfully designed to predict stock returns. Concretely, the following research question is addressed:

Can a CNN be designed that accurately predicts stock returns into one of the following three classes?

- Label 2: Positive returns (defined as returns > 2% over the next calendar month)
- Label 1: Neutral returns (defined as returns <= 2% and >= -2% over the next calendar month)
- Label 0: Negative returns (defined as returns < -2% over the next calendar month)

Metrics

The model will be evaluated based on its accuracy. This is simply defined as the % of samples that is classified correctly. While it is a basic metric and has some limitations (especially for unbalanced classes), it still gives a quick and easy to understand idea of the quality of the model.

To offer some additional insights, the model will be evaluated based on its precision and recall as well. However, since this is a multi-class problem, the precision and recall will have to be calculated separately for each class. Ultimately, with stock return predictions, the aim is to catch most opportunities (high recall) and avoid (costly) wrong investments (high precision).

Analysis

Dataset

The raw data used for this project will consist of adjusted prices for (almost) all US stocks & ETFs between 1984 and 2017 (Marjanovic, 2017). The dataset consists of adjusted prices, meaning that stock splits and dividends have been taken into account. This dataset was released to the public domain under CC0.

The raw data will be cleaned and transformed into a collection of (labelled) images according to the three predictive classes from the problem statement.

Concretely, the data is randomly sampled and many observations with the following features are created:

- Historical returns
- Historical standard deviation
- Area chart of historical prices
- Gramian Angular Difference field plot of historical prices
- A dummy that indicates whether the observation is from a stock or ETF

Each feature is calculated for a 1 month, 6 month, and 12 month lookback period.

Next to the features, a label is created that corresponds to the 3 classed defined in the problem statement.

This new dataset of observations is the one that is used in the remainder of the project.

Data Exploration

The transformed dataset contains the following columns:

- **date**: the start date of the observation
- **asset_file**: the path of the file from which this observation was created
- **stock**: a boolean indicator that is 1 if this observation comes from a stock or 0 if it comes from an ETF
- **1_month_return**: the return of the asset in the previous calendar month (lagged by one day to omit look-ahead bias)
- **6_month_return**: the return of the asset in the previous 6 calendar months (lagged by one day to omit look-ahead bias)
- **12_month_return**: the return of the asset in the previous 12 calendar months (lagged by one day to omit look-ahead bias)
- **1_month_volatility**: the (daily) volatility of the asset in the previous calendar month (lagged by one day to omit look-ahead bias)
- **6_month_volatility**: the (daily) volatility of the asset in the previous 6 calendar months (lagged by one day to omit look-ahead bias)
- **12_month_volatility**: the (daily) volatility of the asset in the previous 12 calendar months (lagged by one day to omit look-ahead bias)
- **1_month_img**: the name of the .png file containing a GADF chart of the asset price in the previous calendar month (lagged by one day to omit look-ahead bias)

- **6_month_img**: the name of the .png file containing a GADF chart of the asset price in the previous 6 calendar months (lagged by one day to omit look-ahead bias)
- **12_month_img**: the name of the .png file containing a GADF chart of the asset price in the previous 12 calendar months (lagged by one day to omit look-ahead bias)
- **1_month_img_bar**: the name of the .png file containing an area chart of the asset price in the previous calendar month (lagged by one day to omit look-ahead bias)
- **6_month_img_bar**: the name of the .png file containing an area chart of the asset price in the previous 6 calendar months (lagged by one day to omit look-ahead bias)
- **12_month_img_bar**: the name of the .png file containing an area chart of the asset price in the previous 12 calendar months (lagged by one day to omit look-ahead bias)
- **label_1m**: the classification target. This can be 0 (next-month returns < 2%), 1 (next-month returns >= -2% and <= 2%), or 2 (next-month returns > 2%)
- **1m_return**: the return over the next 1 month (used for baseline regression model)

Two types of images representing the stock price time-series are present in this dataset. First is a Gramian Angular Difference Field (GADF) of the historical stock prices. This is a 2D-representation of a time series. It has been shown that this representation of time series helps in classification an imputation tasks (Wang & Oates, 2015). Next is a simple area chart of the historical stock prices. For similar images (bar charts), Ozbayoglu & Sezer have already shown that a CNN model trained on these images has some predictive power in predicting future stock returns (Ozbayoglu & Sezer, 2019). They used a 30-day lookback window, but their stock universe was rather small and limited to only the Dow 30. This is in contrast with the dataset used for this project which covers the entire US stock market.

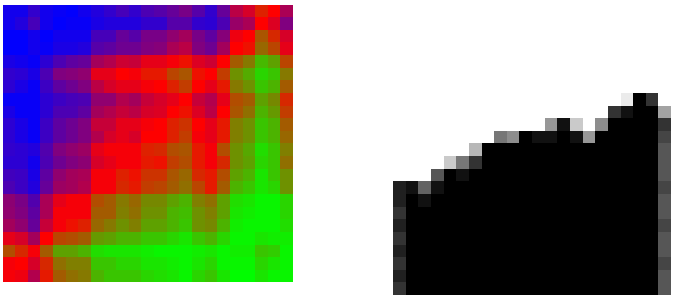

Figure 1 Gramian Angular Difference Field (left) and Area Chart (right) of the same stock price history over a 360 day period.

Both the GADF and area chart images are plotted based on smoothed prices (5-day average) that were normalized relative to the price on the first trading day of the lookback window. Furthermore, each picture is 30x30 pixels large. The GADF charts are represented by 3-channel RGB images, and the area charts as 1-channel greyscale images.

The dataset contains 20,989 observations.

Exploratory Visualization

First, regarding the labels, the dataset is very imbalanced (see table below**)**. This can create problems for the model training. Therefore, the training, test, & validation sets are sampled from this dataset in such a way that it contains a balanced sample of the different labels.

Label	Count
Negative Returns	8,131
Neutral Returns	3,059
Positive Returns	9,799

Table 1 Class Imbalance

Second, as can be seen in the plot below, the data is skewed towards more recent dates. This can introduce some bias to the model as more recent events are taken into account more.

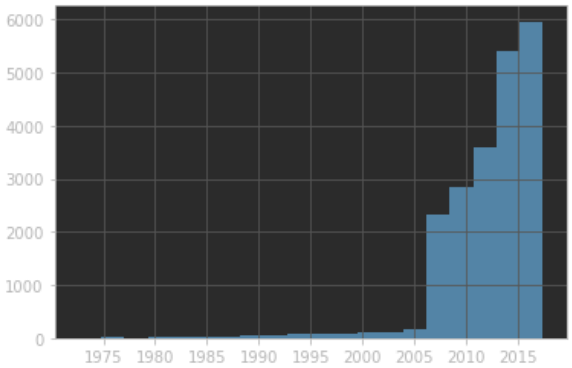

Figure 2 Distribution of the observations over time

Third, some observations display extremely large (or small returns). Therefore, the dataset is filtered so that only observations with returns between -50% and 100% are retained. This is still a large range, but is within a realistic range based on historical stock market performance and crashes. The goal of this classifier is to predict returns under "normal" conditions, not to predict extreme bull-runs or crashes which is a much harder problem.

Algorithms and Techniques

A Convolutional Neural Network (CNN) is used for this project. It is in line with the models used in previous research and is still a state-of-the-art algorithm for image classification. The algorithm classifies images by assigning a probability that they belong to each class. For this project, the following parameters were tuned:

- The number of epochs (the number of cycles that are trained)
- The batch size (the number of images used in a single training step)
- The learning rate (the rate at which the algorithm learns from new information)

The structure of the CNN consists of a convolutional layer followed by a fully connected linear layer to predict the three labels.

- A CNN structure depending on the input type
    - A CNN structure for the GADF images
    - A CNN structure for the area chart images
- An FC to turn the output of the CNN structure into predictions for the three labels

The CNNs used for the image classification are based on the model from Ozbayoglu & Sezer (Ozbayoglu & Sezer, 2019). An overview of the entire model will be provided below in the implementation section of the methodology.

An Adam optimizer was used so momentum is dynamic rather than being chosen upfront. Since this is a multi-class problem, Cross-Entropy Loss is minimized during the training.

The model is trained on the CPU, but could easily be modified to train on a GPU instead.

Benchmark

For the benchmark model, a simple OLS regression model was first constructed. Due to poor performance however, a Logistic Regression model was constructed as a benchmark model instead. This model takes the historical 1 month, 6 month, and 12 month returns and volatilities into account and predicts probabilities that an observation belongs to each of the three classes. Similar to the CNN model, this Logistic Regression model was trained on the training set and will be evaluated on the test set.

Given the simplicity of this model, and the complexity of the problem, this model is not expected to perform well & might even perform worse than a simple random allocation.

The model has an overall accuracy of 43.69% on the test set and the precision & recall for the different classes are as follows:

Label	**Precision**	**Recall**
Negative Returns	43.65%	44.14%
Neutral Returns	46.23%	63.51%
Positive Returns	38.10%	23.42%

Table 2 Benchmark performance metrics

The accuracy, precision, and recall are all slightly better than a random allocation except for the positive return recall. Given the balanced sample of 3 classes, a random allocation would hover around 33.33% for the accuracy, precision, and recall. Although the model does not perform bad on negative and neutral return predictions, it fails to predict and recall positive returns well.

Methodology

Data Preprocessing

Given that the model data was constructed from scratch based on the raw data no further preprocessing was necessary. The only preprocessing that took place was to construct meaningful train, test, and validation subsamples.

The unbalanced labels and outlier returns are taken into account to create a subsample of observations that are considered for this problem. This leads to a training dataset with 6,210 observations and a validation and test dataset with 1,332 and 1,329 observations respectively.

Implementation

2 model designs were tested: a model that only uses area chart images and a model that only uses GADF chart images. All models only used the 1 month historical return images. (The 6 month and 12 month images could be used for future improvements of this model.)

Both models were finetuned using the "optuna" package and Pytorch. The search space for the hyperparameters was as follows:

- Learning rate: 0.0001 to 0.01
- Epochs: 50 to 200
- Batch size: 64 to 2056

During the finetuning, 5 models are constructed and the model with the highest accuracy (on the validation set) is retained as the best model for that specific design.

Overall, the models were trained by minimizing their Cross-Entropy Loss and using an adaptive momentum optimizer (Adam).

The structure of both models is equal, with the one exception that the model trained on area chart images only uses 1 input channel (greyscale), while the model trained on GADF images uses 3 input images (RGB). The structure of the models is inspired by Ozbayoglu & Sezer (Ozbayoglu & Sezer, 2019) and can be seen in the image below.

As marked on the image below, the CNN model consists of 3 parts. First, there is the input image which is a 30x30 pixel image. Next, there is a convolutional structure consisting of two convolutional layers, followed by one max pooling layer (to compress the output information). Finally, there is a fully connected linear structure consisting of 2 dropout layers (to reduce overfitting) and 2 linear layers. Throughout the entire structure, ReLu activation is used as the activation function given its attractive properties.

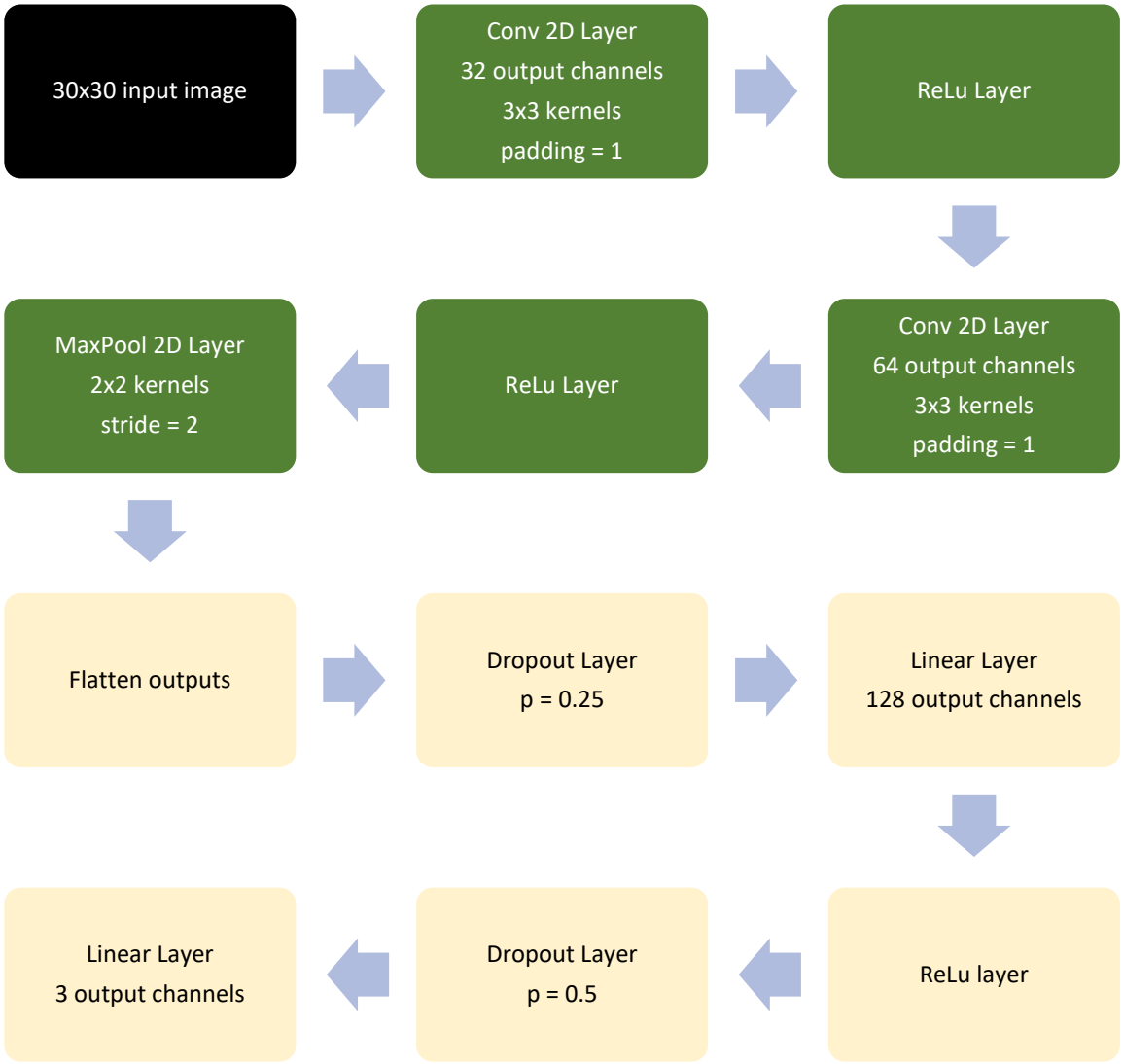

Figure 3 High-level structure of the CNN model

Refinement

The CNN models are largely in line with the model suggested by Ozbayoglu & Sezer (Ozbayoglu & Sezer, 2019). The model incorporates 2 dropout layers to prevent overfitting. Next to this, the model uses an adaptive momentum optimizer (Adam) during the training so that the learning rate is automatically decreased once the training losses start to stop decreasing.

When training the model on area charts with random hyperparameters, the model failed to learn and remained stuck at a 33.33% accuracy on the validation dataset. After adding a hyperparameter optimization step, this accuracy increased to 42.64%. The hyperparameter optimization suggests that smaller learning rates in combination with larger epochs and batch sizes are more optimal for this problem. Training the model on Gramian Angular Difference Field plots rather than on area charts, failed to produce any improvements to the model. The accuracy decreased to 37.99%. This suggests that (at least when using the same model structure), the GADF images do not add any value.

Results

Model Evaluation and Validation

The two trained models will be evaluated separately. First, the model trained on area charts will be analyzed and second, the model trained on GADF images will be analyzed.

The models were trained on a training dataset which contained 70% of the data, the hyperparameter tuning took place using a validation dataset which contained 15% of the data, and the final evaluation took place using a test dataset which contained the remaining 15% of the data.

The area chart model used the following hyperparameters:

```
params = {
    "learning_rate": 0.00048304275749704115,
    "epochs": 67,
    "batch_size": 657
}
```

The area chart model has an accuracy on the test set of 42.64%. This is higher than a random allocation, but also not too far from it. The precision and recall metrics can be seen in the table below.

Label	Precision	Recall
Negative Returns	39.81%	55.41%
Neutral Returns	52.89%	51.58%
Positive Returns	33.10%	20.95%

Table 3 Area chart model performance metrics

Given the context, the precision of the negative and positive returns is one of the most important metrics. Nonetheless, the model fails to perform better than a random allocation for the positive returns and only does slightly better for the negative returns. The model seems to perform relatively well on the precision of neutral returns however.

The charts below show the performance of the model during the training. Neither the training/validation losses nor the accuracies show signs of over- or underfitting.

Figure 4 Training losses & accuracies for the area chart model

The GADF model used the following hyperparameters:

```
params = {
    "learning_rate": 0.001480635942514392,
    "epochs": 50,
    "batch_size": 1431
}
```

The GADF model has an accuracy on the test set of 37.99%. This is only slightly higher than a random allocation and below both the area chart model and the benchmark model. The precision and recall metrics can be seen in the table below.

Label	Precision	Recall
Negative Returns	37.44%	34.91%
Neutral Returns	38.05%	44.82%
Positive Returns	38.48%	34.23%

Table 4 Area chart model performance metrics

Given the context, the precision of the negative and positive returns is one of the most important metrics. Nonetheless, the model only performs slightly better than a random allocation on both of those metrics. Overall, the accuracy, precisions and recalls all point to a model that fails to perform much better than a random allocation.

The charts below show the performance of the model during the training. These charts show a different story than the area chart model. Here there is a clear case of overfitting. While the training loss keeps decreasing and the training accuracy keeps increasing, the validation loss starts to increase after epoch 20 and the validation accuracy plateaus after epoch 20. This suggests that the model can clearly learn something from the GADF images (as it is recognizing patterns and overfitting), but with the current model structure, the model is unable to generalize these learnings.

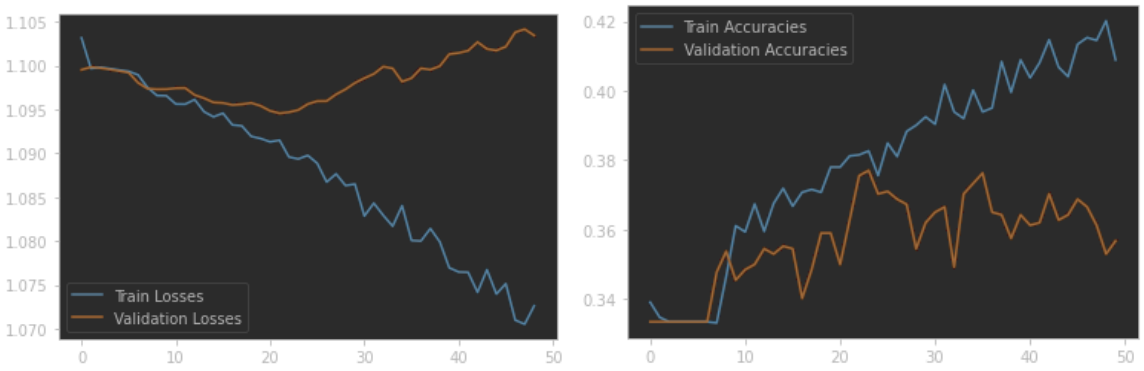

Figure 5 Training losses & accuracies for the area chart model

Justification

Below, the performance metrics of all three models are shown again.

The benchmark model has an overall accuracy of 43.69% on the test set.

Label	Precision	Recall
Negative Returns	43.65%	44.14%
Neutral Returns	46.23%	63.51%
Positive Returns	38.10%	23.42%

Table 5 Benchmark performance metrics

The area chart model has an accuracy on the test set of 42.64%.

Label	Precision	Recall
Negative Returns	39.81%	55.41%
Neutral Returns	52.89%	51.58%
Positive Returns	33.10%	20.95%

Table 6 Area chart model performance metrics

The GADF model has an accuracy on the test set of 37.99%.

Label	Precision	Recall
Negative Returns	37.44%	34.91%
Neutral Returns	38.05%	44.82%
Positive Returns	38.48%	34.23%

Table 7 Area chart model performance metrics

Across all models, the benchmark model is actually the best performer. This is surprising as similar CNN models from previous studies have shown promising performances. A key difference between this project and previous projects is the scope of the data. While previous studies limited themselves to small universes (such as the DOW50), this project looks at (almost) all stocks and ETFs listed on US stock exchanges. This means that there is a very wide range of assets which might behave differently (for instance small vs large companies, tech vs cyclical …). Perhaps the patterns recognized for certain types of assets are not transferable to other types of assets.

Nonetheless, there are some points where the CNN models perform better than the benchmark model. First, the area chart model has a much higher recall of negative returns. Roughly 55% of all negative returns were captured by this model. This could be useful to avoid investing in assets that will have a negative future return. Second, the GADF model has a higher recall of positive returns, although at 34% it is not too far from a random choice either. Finally, the GADF model seems to have the most balanced precisions. It does not seem to be skewed to either of the three classes.

In conclusion, while the area chart and GADF model do not offer much value in their current form, they do offer some interesting metrics already. Perhaps with a different structure (for instance to omit the overfitting seen in the GADF model), the accuracy, precision and recall can be bumped up to beat the benchmark model. In its current form, the area chart model is the best performer out of the two CNN models, with an accuracy close to the benchmark model.

Conclusion

Reflection

The problem of predicting future stock returns is a problem that, in my humble opinion, will never be solved with 100% accuracy. While the model constructed above shows some nice characteristics, it is still far from being usable in a real-life investment setting. However, the model suggested above could (with some improvement) become a part of a larger ensemble model that takes many different predictions into account as simple directionality of prices (as suggested by this model) is likely too vague of a signal to make a clear investment decision. Next to this, a more elaborate investment algorithm should also take provisions against increased volatility and potential market downturns into account.

Regardless of the direct usability of this model, the fact that a machine learning model based on stock chart images has a higher than random accuracy is surprising given that technical analysis (the manual analysis of stock chart images) is often discarded as a bad investment strategy.

Improvement

There are a number of ways to improve the model. First of all, the other images from the cleaned dataset could be incorporated into the CNN model (6 month & 12 month historical return images). The numerical features (returns, standard deviations …) could also be incorporated into the CNN model to create a multi-input type model.

Next to these rather "quick" potential improvements, there are also more elaborate potential improvements. First of all, the training, validation and test data could be balanced over time. Currently there is a skew towards more recent observations, which might introduce unwanted biases. Second, the training, validation and test datasets could also be set up in such a way that they only contain observations from completely separate time periods (to omit lookahead biases). Third, other images, such as the evolution of return standard deviation over time or factor scores (Value, Momentum …) over time could be incorporated as they could also contain potentially valuable information for the model.

Sources

cmc markets. (n.d.). *Bonds vs stocks*. cmc markets. Retrieved from https://www.cmcmarkets.com/en/trading-guides/bonds-vs-stocks

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of financial economics*, *116*(1), 1-22.

Jin, G., & Kwon, O. (2021). *Impact of chart image characteristics on stock price prediction with a convolutional neural network*. PLOS ONE. Retrieved from https://doi.org/10.1371/journal.pone.0253121

Marjanovic, B. (2017). *Huge Stock Market Dataset*. Kaggle. Retrieved from https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

Ozbayoglu, M., & Sezer, O. B. (2019). *Financial Trading Model with Stock Bar Chart Image Time Series with Deep Convolutional Neural Networks.* ResearchGate. Retrieved from https://www.researchgate.net/publication/331700676_Financial_Trading_Model_with_Stock_Bar_Chart_Image_Time_Series_with_Deep_Convolutional_Neural_Networks

Wang, Z., & Oates, T. (2015). *Imaging Time-Series to Improve Classification and Imputation.* ResearchGate. Retrieved from https://www.researchgate.net/publication/277603742_Imaging_Time-Series_to_Improve_Classification_and_Imputation