KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Technology

Second cycle, 30 credits

End-to-End Portfolio Optimization using Time-series Generative Adversarial Networks and Transfer Learning

MERVAN CAN KAYA
FILIP ISAK MATTSSON

Stockholm, Sweden, 2024

End-to-End Portfolio Optimization using Time-series Generative Adversarial Networks and Transfer Learning

MERVAN CAN KAYA

FILIP ISAK MATTSSON

Master’s Programme, Industrial Engineering and Management, 120 credits
Date: August 8, 2024

Supervisors: Jingwen Fu, Mikael Anveden, Victor Gruselius
Examiner: Ming Xiao
School of Electrical Engineering and Computer Science
Host company: Fjärde AP-fonden (AP4)
Swedish title: End-to-End Portföljoptimering med Tidsserie Generativa Motståndarnätverk och Överföringslära

Abstract

This thesis introduces an end-to-end portfolio optimization framework utilizing Time Series Generative Adversarial Networks (TimeGANs) and Sequence-to-Sequence (Seq2Seq) models integrated with Mean-Variance Optimization (MVO). The research aims to improve the accuracy of market trend predictions and improve automatic investment decision-making by capturing the temporal dependencies in financial markets. The proposed framework utilizes TimeGANs where the network components are trained jointly, and then the embedding and recovery networks are extracted and used as encoders and decoders, respectively, in Seq2Seq models for forecasting. The integration of neural networks with traditional MVO provides an effective and adaptive tool for portfolio management. The effectiveness of the framework is validated through empirical analysis of information coefficients, Sharpe ratio, and comparisons with traditional benchmarks, demonstrating significant improvements in the accuracy of forecasts and portfolio performance. The mean information coefficient for the 10-week analysis is 0.069 with a p-value of 0.022 indicating a statistical significance at the 5% level. The long-only optimized portfolio obtained a 1.7 mean Sharpe ratio over a 3-year period.

Keywords

Financial Markets, Generative Adversarial Networks, Timeseries, Portfolio Optimization, Mean-Variance, End-to-end, Transfer Learning, Timeseries Generative Adversarial Networks, Recurrent Neural Networks, Sequence-to-Sequence

Sammanfattning

Denna rapport introducerar en ny end-to-end portföljoptimeringsramverk som använder Time Series Generative Adversarial Networks (TimeGANs) och Sequence-to-Sequence (Seq2Seq) modeller integrerade med Mean-Variance Optimization (MVO). Forskningen syftar till att förbättra noggrannheten i marknadstrendprognoser och förbättra automatiserade investeringsbeslut genom att fånga de temporala beroendena i finansiella marknader. Det föreslagna ramverket använder TimeGANs där nätverkskomponenterna tränas gemensamt, och sedan extraheras Embedding- och Recovery-nätverken och används som Encoder och Decoder respektive i Seq2Seq-modeller för prognostisering. Integrationen av neurala nätverk med traditionell MVO tillhandahåller ett effektivt och adaptivt verktyg för portföljhantering. Ramverkets effektivitet valideras genom empirisk analys av informationskoefficienter, Sharpe-kvot och jämförelser med traditionella referensramar, vilket visar på betydande förbättringar i prognosernas noggrannhet och portföljens prestanda. Den genomsnittliga informationskoefficienten sett över en 10-veckors analys är 0.069 med ett p-värde på 0.022 som indikerar en statistisk signifikans på 5% nivån. Den optimerade lång-portföljen erhöll en genomsnittlig Sharpe-kvot på 1,7 över en 3-årsperiod.

Nyckelord

Finansiella Marknader, Generativa Motståndsnätverk, Tidsserier, Portföljoptimering, Mean-Variance, End-to-end, Överföringsinlärning, Tidsserie Generativa Motståndsnätverk, Recurrent Neural Networks, Sequence-to-Sequence

Acknowledgments

Firstly, our sincere thanks go to Victor Gruselius and Nils Everling for their invaluable guidance throughout the duration of this project. Their insights and advice were crucial to our success.

Secondly, we extend our heartfelt appreciation to Mikael Anveden and Magdalena Högberg for their support and for providing the necessary resources to make this project possible.

Finally, we are profoundly grateful to Bálint Gabulya for his guidance, inspiration, and support throughout the project. His contributions were instrumental in shaping the outcome of our work.

Stockholm, August 2024
Mervan Can Kaya Filip Isak Mattsson

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

10W	10 week
1W	1 week
5W	5 week
AP4	Fjärde AP-fonden
BPTT	Backpropagation Through Time
GAN	Generative Adverserial Network
GRU	Gated Recurrent Unit
IC	Information Coefficient
KDE	Kernel Density Estimate
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MCFD	Mean Correct Forecast Direction
ML	Machine Learning
MLP	Multi-layer Perceptron
MPT	Modern Portfolio Theory
MSE	Mean Squared Error
MVO	Mean-Variance Optimization
OHLCV	Open, High, Low, Close and Volume
PAGAN	Portfolio Analysis with Generative Adversarial Networks
PCA	Principal Component Analysis
RHN	Recurrent Highway Network
RMSE	Root Mean Squared Error
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
Seq2Seq	Sequence-to-Sequence

t-SNE	t-Distributed Stochastic Neighbor Embedding
TimeGAN	Time Series Generative Adversarial Network

Chapter 1

Introduction

In the world of finance, finding the best way to manage investment portfolios is a key challenge. Traditional methods often struggle with capturing the temporal dependencies of financial markets. This thesis introduces a novel end-to-end approach using Time Series Generative Adversarial Networks (TimeGANs), Sequence-to-Sequences (Seq2Seqs) models, and Mean-Variance Optimization (MVO). TimeGANs offers improved time series data synthesizing by capturing temporal dependencies, allowing Seq2Seq models to forecast future data more accurately. Together with MVO for portfolio optimization, the result is a suitable end-to-end framework for asset managers.

This research aims to combine these technologies to develop a system that not only predicts market trends more accurately but also makes investment decisions automatically. In doing so, it seeks to provide a better tool for investors that adapts to new information and helps secure profitable results. The following sections will explore how these tools work together and the benefits they offer over traditional methods.

1.1 Background

The development of portfolio optimization principles has evolved significantly since Harry Markowitz introduced Modern Portfolio Theory (MPT) in 1952 [1], highlighting the importance of optimizing portfolios based on risk tolerance and expected return. This shift has led to more diversified investment strategies, reducing overall risk. Despite the widespread academic influence of Markowitz's MVO, its practical application is limited because of its propensity to produce concentrated portfolios and sensitivity to changes in expected excess returns [2]. To address these issues, Fisher Black and Robert Litterman developed an alternative model, the Black-Litterman model, which incorporates investor insights, using the equilibrium portfolio as a baseline and applying a Bayesian approach to blend market expectations with specific asset views, offering a more intuitive and adaptable framework for portfolio management [2].

However, the complexity of financial markets has increased significantly in the past 10 years, driven by the surge in high-frequency data, fragmented markets, and advanced trading algorithms. This evolution, as discussed by O'Hara [3], suggests that traditional market modeling methodologies might no longer be as effective in this dynamic landscape.

The advancement of Machine Learning (ML) in portfolio optimization and financial forecasts offers a powerful solution to the challenges raised by O'Hara [3], using computational algorithms to analyze large datasets and extract complex patterns that traditional statistical methods may overlook. The Portfolio Analysis with Generative Adversarial Networks (PAGAN) model is a notable example, employing deep generative models to learn the joint probability distribution of price trends, effectively replicating the real market's probability dynamics. This innovative approach has shown significant potential in optimizing portfolios to strike a balance between risk and expected returns, as demonstrated in Mariani's [4] research. This integration of new technology with established financial theories marks a significant advance in the realm of financial investment, highlighting the dynamic nature of this field.

1.2 Problem

These traditional frameworks, while foundational, often fail to adequately address the complex dynamics of contemporary markets, resulting in portfolios that may be overly concentrated or sensitive to estimation errors. Described by Haugh [5], mean variance analysis tends to produce extreme portfolios combining extreme shorts and extreme longs, usually caused by estimation errors of the mean return and the covariance matrix. Secondly, according to Haugh [5], the extreme sensitivity of the portfolio weight to small changes in expected returns can lead to large errors in the optimal portfolio composition, especially when the number of assets is large but the sample size is small. This instability is due to the nature of the covariance matrix used in MVO, where the estimation errors are magnified, making MVO unreliable without additional techniques to manage these sensitivities.

Additionally, as described by Schmitt et al. [6], financial markets are highly nonstationary systems, meaning sampled observed means and covariances strongly varies depending on the time window they are sampled in. This imposes limitations on models that estimate these statistical properties. This necessitates the need for a model that is able to capture the temporal dependencies of the time series.

This thesis seeks to extend current research, by implementing forecasting models into a practical application, useful to many types of investors, thus filling the gap between research and application. The developed models are used to forecast future stock prices and movements. The forecasts can then be used later on for portfolio optimization and portfolio management in the face of market uncertainties.

The question the thesis therefore seeks to answer is the following:

Will the integration of TimeGAN and Seq2Seq forecasting model with Mean-Variance Optimization (MVO) improve the accuracy of market trend predictions and enhance automatic investment decision-making in portfolio management compared to traditional benchmarks?

1.3 Purpose

The purpose of this thesis is to develop and evaluate a novel end-to-end framework that integrates TimeGANs with MVO algorithms for improved portfolio management. This research aims to address the limitations of traditional investment portfolio management methods, which often struggle to capture the temporal dependencies of financial markets. Using TimeGANs for improved time series forecasting and MVO for optimized investment decisions, the proposed system aims to offer a more accurate and adaptive tool for predicting market trends and making automatic investment decisions. Ultimately, this thesis attempts to bridge the gap between theoretical advancements and practical applications in financial investment, providing investors with a robust framework that adapts to new information and secures profitable outcomes in the dynamic landscape of contemporary markets.

1.4 Goals

The goal of this project is to develop an end-to-end portfolio optimization model using TimeGAN and MVO. This has been divided into the following three subgoals:

1. **Industry Application** The goal is to create a complete portfolio optimization model that integrates TimeGAN with MVO algorithms for enhanced portfolio management. Thus, providing actionable insights and adaptive portfolio strategies that improve investment performance and risk management in real-world market conditions.
2. **Academic Contribution** Produce a comprehensive analysis, validation and comparison of the theoretical foundations and practical results of the model, focusing on its contribution to the existing body of knowledge in financial engineering and machine learning offering new research on robust portfolio optimization.
3. **Preparation for Future Challenges** Facilitate a deep learning experience through the hands-on application of advanced machine learning techniques and financial theories, enhancing our analytical, programming and financial modeling skills. This goal emphasizes the practical understanding and application of complex concepts for future challenges in both academia and the financial industry.

This project will deliver an end-to-end portfolio optimization framework and a thorough analysis of the possibilities of practical applications of such frameworks for investing purposes.

1.5 Delimitations

The project is bounded by some major limitations, which drastically determine the outcome of the paper.

Firstly, given the vast amount of data available, the computational requirements would be too high. Therefore, the universe of stocks has been drastically narrowed to include only the top 10 largest stocks by weight in the MSCI USA index (with some exceptions described in 3.3.1). We also restrict the data to consist only of daily Open, High, Low, Close and Volume (OHLCV) values. Computational limitations again restrict us from adding technical indicators, which could provide the models further context of market conditions and improve prediction accuracy.

Secondly, due to the complexity of modeling and training machine learning models, the implementation of TimeGAN was only tested using a statistical analysis and not a comparative analysis. A comparative analysis would also be desired, as it allows us to benchmark our model versus similar models and check the reliability of our model's performance.

Lastly, due to the long training and testing times and the duration of the project, portfolio optimization was limited to only including MVO versus index benchmarks. This is because it would simply take too long to implement custom models, or models based on reinforcement learning, for example. Improvements in this part of the thesis will be discussed in more detail in 7.3.

1.6 Structure of the Thesis

Chapter 1 introduces the thesis, including background, problem statement, purpose, goals, delimitations, and the thesis structure.

Chapter 2 provides theoretical background on Generative Adversarial Networks, time series forecasting methods, and portfolio optimization concepts.

Chapter 3 outlines the research methodology, including the research process, data collection, experimental design, data analysis, and evaluation framework.

Chapter 4 details the implementation of the models, covering the preparation of data, model configuration, training, evaluation, and financial evaluation framework.

Chapter 5 presents the results and analysis, including evaluations of the TimeGAN, Seq2Seq models, and portfolio optimization results.

Chapter 6 discusses the interpretation of key results, the validity of the findings, and the reliability of the implementation.

Chapter 7 concludes the thesis, summarizing the findings, limitations, future work, and reflections.

The References section lists all the sources cited.

Chapter 2

Theoretical Background

This chapter will describe the theoretical knowledge that underlies the project.

2.1 Generative Adverserial Networks

2.1.1 Adverserial Games

The adversarial modeling framework as described in the work of Goodfellow et al. [7] on Generative Adverserial Networks (GANs) leverages the dynamics between two models: a generator (G) and a discriminator (D). This framework is designed to train the generator to produce synthetic data that closely mimic a real data distribution, while the discriminator distinguishes between real and generated data.

The generator begins by taking a noise variable from a predefined noise distribution as input. This noise is then mapped to the data space using a differentiable function defined by the generator's neural network, aiming to replicate the real data distribution. In contrast, the discriminator assesses the incoming data and outputs a value between 0 and 1 indicating the confidence level of the data being real.

Training these two networks unfolds in an adversarial manner, following the principles of a minimax game. The discriminator aims to maximize its accuracy in classifying data as real or generated, while the generator aims to fool the discriminator. This creates a feedback loop where both models continuously improve: the discriminator becomes better at identifying real versus fake data, and the generator improves its ability to create realistic data.

Due to computational limitations and to avoid overfitting, the training alternates between optimizing *D* and *G*. Typically, this involves several steps of optimizing *D* followed by a single step of optimizing *G*, ensuring *D* remains near its optimal solution, providing effective feedback for *G*'s improvement.

Initially, when *G* is less capable of generating convincing data, the objective function might not provide enough strong gradients for effective learning. To address this, the training objective for *G* can be shifted from minimizing the probability of *D* correctly identifying fake data to maximizing the likelihood of *D* mistakenly identifying the generated data as real. This adjustment ensures that *G* receives robust gradients early in the training process.

The adversarial game is formulated as a differentiable function represented by a neural network. Given a prior input noise distribution $p_z(z)$, the generator $G(z;\theta_g)$ aims to map this noise to the data space to match the real data distribution p_{data}. The generator parameters θ_g are adjusted to minimize the difference between the generated data distribution and the real data distribution. The optimization goal can be formally expressed as part of the minimax game with the discriminator D:

$$\begin{aligned}\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] \\ + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]\end{aligned} \qquad (2.1)$$

where $V(D,G)$ is the value function, $D(x)$ is the discriminator's estimate of the probability that the real data instance x is real, and $D(G(z))$ is the probability estimated for a data point generated by G being real [7].

2.1.2 Loss Functions

The loss function in GANs is designed to capture the adversarial game between G and D. It consists of two parts: one to optimize D and the other to optimize G. Each part in the minimax game, encouraging D to accurately classify real and generated data, while motivating G to produce data that D cannot distinguish from real data. Both G and D's loss functions are derived from equation 2.1.

2.1.2.1 Discriminator Loss

The loss of the discriminator, L_D, aims to maximize the probability of correctly classifying both real and generated data. It is defined as:

$$L_D = -E_{x \sim p_{data}(x)}[\log D(x)] - E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.2}$$

This loss function combines two terms: the first term penalizes D for incorrectly classifying real data, and the second term penalizes it for incorrectly classifying generated data. By minimizing this loss, D learns to improve its classification accuracy.

2.1.2.2 Generator Loss

The loss of the generator, L_G, is designed to increase the error rate of the discriminator on the generated data. Traditionally, it is expressed as:

$$L_G = -E_{z \sim p_z(z)}[\log D(G(z))] \tag{2.3}$$

This formulation encourages G to produce data that D is likely to classify as real. By maximizing L_G, G improves its ability to generate convincing data.

2.1.3 Optimizers

Optimizers are algorithms that are used to change the learnable parameters of a network, such as weights and learning rate, to reduce loss and thus maximize predictive performance of the network. The optimizer reduces the loss based on a loss function that, as described above, measures the difference between the network output and the actual output. Optimizers allow the network, therefore, to change the parameters such that the network output loss converges to a loss below a certain threshold.

Moreover, to optimize loss, the optimization algorithm provides additional benefits such as increased learning speed by adapting the learning rate during training, avoiding being stuck in local minima during gradient calculations, and handling noisy data by adapting the learning rate for each parameter individually. This allows the model to be better generalized for training and testing data.

2.1.3.1 Adaptive Moment Estimation (Adam)

Adam combines the benefits of AdaGrad and Root Mean Square Propagation (RMSProp), offering an adaptive learning rate for each parameter. This optimizer adjusts the learning rates based on the first and second moments of the gradients, making it particularly suitable for problems involving large datasets and high-dimensional spaces. Despite its widespread use and effectiveness, Adam's aggressive adaptation to learning rate can sometimes lead to overshooting issues [8].

2.2 TimeGAN

2.2.1 Overview of TimeGAN Architecture

The TimeGAN framework, as proposed by Yoon et al. [9], integrates both supervised and unsupervised learning methods within a generative adversarial network architecture. The main components of TimeGAN include the embedding network, recovery network, generator, and discriminator. The embedding and recovery networks are used to map the high-dimensional time-series data to a lower-dimensional latent space and back, ensuring accurate reconstruction of the original data.

2.2.2 Embedding and Recovery Networks

The embedding network in TimeGAN serves to reduce the dimensionality of the input time-series data, facilitating the learning process by creating a compact representation in the latent space. This process is critical as it simplifies the complex temporal dependencies present in the data. The recovery network then takes the latent space representation and reconstructs it back to the original feature space, aiming to minimize reconstruction loss. This ensures that the latent representations retain the essential characteristics of the original data.

2.2.3 Generator and Discriminator Networks

The generator in TimeGAN is responsible for producing synthetic time-series data from the latent space. It works in conjunction with the discriminator, which aims to distinguish between real and synthetic data. The adversarial training between the generator and discriminator improves the quality of the synthetic data, making it increasingly indistinguishable from real data. The generator's ability to capture temporal dynamics and generate realistic sequences is enhanced through the combined use of supervised and unsupervised losses.

2.2.4 Supervised and Unsupervised Learning Components

TimeGAN uniquely combines supervised learning, which leverages actual data embeddings to guide the generation process, with unsupervised learning, which focuses on the reconstruction of the data. The supervised component ensures that the generated sequences maintain accurate stepwise transitions, while the unsupervised component drives the overall realism of the generated sequences. This dual approach allows TimeGAN to achieve high fidelity in both short-term and long-term sequence generation.

2.2.5 Hyperparameter Tuning and Optimization

The performance of TimeGAN is sensitive to the choice of hyperparameters, such as learning rates, batch sizes, and the relative weights of supervised and unsupervised losses. Effective hyperparameter tuning is essential for balancing the trade-offs between reconstruction accuracy and the adversarial learning process. The robustness of TimeGAN to various hyperparameter settings simplifies the tuning process, making it more accessible for different time-series datasets.

2.2.6 Applications of TimeGAN

TimeGAN has been successfully applied in various domains requiring time-series data synthesis, such as financial forecasting, healthcare, and anomaly detection. Its ability to generate realistic and accurate synthetic data makes it a valuable tool for augmenting datasets, performing stress tests, and creating robust models in scenarios where real data is scarce or costly to obtain. The integration of TimeGAN with other advanced models further enhances its applicability and performance across different fields.

2.3 Time Series Forecasting

As described by Dorffner in his paper [10], the problem of predicting future values of data can be seen as finding a function.

$$\mathcal{F} : \mathcal{R}^{k\times(n+1)} \rightarrow \mathcal{R}^{k} \tag{2.4}$$

which obtains an estimate $\hat{\vec{x}}(t+d)$ of the data vector $\vec{x}$ at time $t+d$, where d is the delay in prediction.

Forecasting thus becomes a problem of approximation of the function , where the goal is to approximate the function $\mathcal{F}$ as closely as possible. Performance evaluation is typically performed by computing an error between the forecasted data point and the actual data point. There are many methods for error estimation, such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), which are types of Euclidean distances.

Again, as described by Dorffner (ibid.), for real-world applications, the assumptions of generating a model which approximates the function exactly are not realistic. Measurements of errors and unknown / uncontrollable factors will generate a residual error ϵ that cannot be removed. Therefore, the equation of finding can be expressed as

$$\hat{x}(t+d) = \mathcal{F}(\hat{x}(t), \hat{x}(t-1), ...) + \hat{\epsilon}(t) \tag{2.5}$$

This noise cannot be modeled explicitly; however, many models (including the models used in this paper) assume it to be Gaussian distributed with $\mu = 0$, σ^2 and I.I.D [11].

The Universal Approximation Theorem states that any continuous function $\mathcal{F} : [0,1]^n \rightarrow [0,1]$ can be arbitrarily well approximated by a neural network [12]. Thus, the function 2.4 can theoretically be approximated with an error given by equation 2.5.

2.3.1 Recurrent Neural Networks

Schmidt [13] describes Recurrent Neural Networks (RNNs) as a type of neural network primarily used for pattern recognition in sequential data where the order of the sequences is significant. Traditional Multi-layer Perceptron (MLP) models process information in a single direction, passing data from input nodes through hidden layers to output nodes without cycles. In contrast, RNNs extend this functionality by incorporating feedback loops, allowing the network to consider previous inputs $\hat{x}_{0:t-1}$ in addition to the current input $\hat{x}_t$.

The hidden state at time step t can be expressed as:

$$\mathbf{H}_t = \phi_h(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h) \tag{2.6}$$

where $\mathbf{H}_t$ represents the hidden state at time step t, $\mathbf{X}_t$ is the input at time step t, $\mathbf{W}_{xh}$ is the input-to-hidden weight matrix, $\mathbf{W}_{hh}$ is the hidden-to-hidden weight matrix, and $\mathbf{b}_h$ is the bias term. These components are passed through an activation function ϕ_h [13].

The output at time step t is given by:

$$\mathbf{O}_t = \phi_o(\mathbf{H}_t \mathbf{W}_{ho} + \mathbf{b}_o) \tag{2.7}$$

It is important to note the recursive nature of RNNs: $\mathbf{H}_t$ encapsulates information from all previous hidden states up to $\mathbf{H}_{t-1}$.

The Backpropagation Through Time (BPTT) algorithm used for training RNNs can become complex. As detailed by Schmidt, the derivative $\frac{\partial \mathbf{H}_t}{\partial \mathbf{H}_k}$ involves matrix multiplications over potentially long sequences. For small values of the gradient (less than 1), this can lead to vanishing gradients, while large values (greater than 1) can cause exploding gradients. To mitigate these issues, Long Short-Term Memory (LSTM) units are introduced [13].

2.3.1.1 Long Short-Term Memory

Integration of LSTM into the generator architecture significantly enhances its ability to handle sequential data, making it particularly suited for applications in natural language processing and time series prediction. Although the description of LSTMs highlights their advantages, it would be beneficial to provide a comparison with other methods and discuss potential limitations. For example, LSTMs can be computationally intensive and challenging to train in very long sequences.

RNNs are distinguished from standard feedforward neural networks by their ability to maintain a memory of previous inputs through internal states, which is crucial for processing sequences of data. This statement could be expanded to discuss specific scenarios where this memory aspect is particularly advantageous, such as in handling dependencies over extended sequences.

LSTMs, a special kind of RNN, are designed to overcome the problem of vanishing gradients, allowing the learning of long-term dependencies [14]. It would be helpful to mention that while LSTMs mitigates vanishing gradients, they do not completely eliminate them, and alternative architectures like Gated Recurrent Units (GRUs) or attention mechanisms might offer competitive performance with potentially lower complexity.

The mathematical formulation for an LSTM-based generator can be expressed as follows. At each time step t, the LSTM unit receives the current input z_t, the previous output h_{t-1}, and the previous cell state c_{t-1}, producing the new output h_t and updating its cell state too c_t. This process can be formalized through the following set of equations.

$$f_t = \sigma(W_f \cdot [h_{t-1}, z_t] + b_f) \tag{2.8}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, z_t] + b_i) \tag{2.9}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, z_t] + b_C) \tag{2.10}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t \tag{2.11}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, z_t] + b_o) \tag{2.12}$$

$$h_t = o_t * \tanh(c_t) \tag{2.13}$$

where σ denotes the sigmoid function, $*$ represents element-wise multiplication, f_t, i_t, and o_t are to forget, input and output gates, respectively, $\tilde{C}_t$ is the candidate cell state, and W and b denote the weights and biases associated with each gate [15]. Clarification of how these equations integrate into the larger GAN framework would enhance understanding, especially for readers not familiar with GANs.

Incorporating these dynamics within the GAN framework, the generator $G(z; \theta_g)$, now equipped with LSTM units, generates sequential data by adapting its output based on both immediate input and contextual information retained in its memory. This is particularly effective for generating time-series data, where the temporal dependencies play a critical role. More examples and case studies showcasing the effectiveness of generators based on LSTM in different applications would provide practical insights.

The optimization challenge in training such a model lies in balancing the generator's ability to produce realistic sequences with the discriminator's skill in distinguishing between real and generated data. The overall objective function for the GAN, incorporating an LSTM-based generator, remains as described in the equations above, but with the understanding that $G(z; \theta_g)$ now encapsulates the sequential processing capabilities of LSTMs. A discussion on the practical aspects of training such models, including computational requirements and hyperparameter tuning, would be beneficial for practitioners looking to implement this approach.

2.3.1.2 Gated Recurrent Unit

Chung et al. [16] describes a GRU proposed by Cho et al. [17] as being a recurrent unit that can adaptively capture dependencies of different time scales. The GRU has gating units which modulate the flow of information without having separate memory cells. The activation of GRU is a linear interpolation between the previous activation and the candidate activation, where and update gate decides how much the unit updates its activation. The linear sum between the existing state and the computed state is similar to LSTM, however, GRUs exposes the entire state each time.

The GRU has a set of reset gates that control the behavior of the GRU. When the reset value is close to 0, the gate makes the unit act as if it is only reading the first symbol of an input sequence, making it to *forget* the previous state. We refer to the paper by Chung et al. for the mathematical formulations.

The empirical results in the Chung paper, of applying *tanh-RNN* (an RNN containing tanh units), LSTM and GRU to different sequential data set data sets show GRUs performing overall best with the lowest average negative log-probabilities of the training and test sets for most of the data sets. Although the authors emphasize that these results are preliminary, they state that there are clear advantages of using gated units over the more traditional recurrent units. However, the choice between LSTM and GRUs could not be concluded concretely. However, the findings of the article validated the use of LSTMs and GRUs in our neural network architectures for networks in the model TimeGAN.

2.3.1.3 Recurrent Highway Networks

Recurrent Highway Networks (RHNs) extend the capabilities of LSTM units by incorporating the concept of highway layers into the recurrent transitions of the network [18]. This enhancement enables RHNs to handle more complex nonlinear transition functions from one time step to the next, which is particularly advantageous for tasks requiring deep sequential processing, such as language modeling and time-series prediction.

Traditional RNNs often face difficulties with vanishing and exploding gradients when training on long sequences, which limits their ability to learn long-term dependencies. LSTM networks address this by using specialized gating mechanisms to maintain a more stable gradient flow. However, even with LSTMs, the transition functions between consecutive time steps are typically shallow, often consisting of a single trainable linear transformation followed by a nonlinearity [14].

RHNs build upon this by introducing depth into the recurrent transitions themselves. Inspired by the success of highway networks in feedforward settings, RHNs utilize highway layers within the recurrent loop, allowing multiple nonlinear transformations at each time step. This approach not only improves the representational capacity of the network but also maintains the stability of training by effectively managing the gradients through the use of gating mechanisms.

The mathematical formulation for an RHN at each time step t can be described as follows. Let x_t be the current input, y_{t-1} be the previous hidden state, and L be the desired recurrence depth. The RHN layer with depth L is defined by:

$$s_0^{[t]} = y_{t-1} \tag{2.14}$$

$$h_\ell^{[t]} = \tanh(W_H x_t \cdot I\{\ell = 1\} + R_{H\ell} s_{\ell-1}^{[t]} + b_{H\ell}) \tag{2.15}$$

$$t_\ell^{[t]} = \sigma(W_T x_t \cdot I\{\ell = 1\} + R_{T\ell} s_{\ell-1}^{[t]} + b_{T\ell}) \tag{2.16}$$

$$c_\ell^{[t]} = \sigma(W_C x_t \cdot I\{\ell = 1\} + R_{C\ell} s_{\ell-1}^{[t]} + b_{C\ell}) \tag{2.17}$$

$$s_\ell^{[t]} = h_\ell^{[t]} \cdot t_\ell^{[t]} + s_{\ell-1}^{[t]} \cdot c_\ell^{[t]} \tag{2.18}$$

for $\ell \in \{1, \ldots, L\}$, where $h_\ell^{[t]}$, $t_\ell^{[t]}$, and $c_\ell^{[t]}$ are the outputs of the nonlinear transform, the transform gate, and the carry gate in layer ℓ, respectively. The indicator function $I\{\ell = 1\}$ ensures that the input x_t is only used in the first layer. The final output of the RHN layer is $s_L^{[t]}$, which serves as the hidden state y_t for the next time step.

Using the flexibility of highway layers, RHNs achieve more nuanced control over the flow of information through the network, dynamically adjusting the depth of processing at each time step. This capability allows RHNs to outperform traditional LSTM architectures on various benchmarks, particularly in tasks involving long sequences and complex temporal dependencies [18].

Weight-tying is a regularization technique that further enhances the efficiency and performance of an RHN layer by tying the input and output embedding. Moreover, weight-tying reduces the total number of parameters, leading to a more compact model. This not only mitigates the risk of overfitting, but also decreases computational requirements, making the network more scalable and faster to train. Furthermore, weight-tying can help maintain consistency and coherence in learned representations, contributing to improved generalization on various tasks [18, 19].

2.3.2 Directional Loss Function

There are a multitude of different loss functions applicable for forecasting. Lee [20] describes four different financial forecasting loss functions; mean squared forecast errors (MSFE), mean absolute forecast errors (MAFE), mean forecast trading returns (MFTR), and Mean Correct Forecast Directions (MCFDs). While the first three are relevant and interesting, this thesis will focus, and implement the last one; MCFD.

The loss function is defined as:

$$MCFD = -N^{-1} \sum_{t=R}^{T} \mathbf{1}(\text{sign}(f_{t,1}) \cdot \text{sign}(Y_{t+1}) > 0). \tag{2.19}$$

where the correct direction $c(y, \hat{y}) = -\text{sign}(f) \cdot \text{sign}(y)$, where $\text{sign}(x) = \mathbf{1}(x > 0) - 1(x < 0)$ and $\mathbf{1}(\cdot)$ is effectively the same as the Kronecker Delta function where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.20}$$

The negative sign in the beginning of equation 2.19 is to make the loss function minimize instead of maximize.

Equity market timing, as described by Baker and Wurgler [21], is an important aspect of real corporate financial policy where the intention is to exploit temporary fluctuations in the cost of equity relative to the cost of other forms of capital. In practice, this can be seen as ”selling high” and ”buying low”. Being able to predict these market timings allows the asset manager to adjust their portfolios in time and thus returning higher returns.

2.4 Sequence-to-Sequence (Seq2Seq) Models

Seq2Seq models are a class of deep learning models designed to transform a sequence of elements from one domain to a sequence in another domain. These models were initially introduced for machine translation tasks, where the goal is to translate sentences from one language to another. The architecture typically consists of an encoder and a decoder, both of which are RNNs or their variants, such as LSTM networks or GRUs [22].

The encoder processes the input sequence and compresses its information into a fixed-size context vector, which captures the essential features of the input sequence. This context vector is then fed into the decoder, which generates the output sequence one element at a time. The decoder's ability to produce accurate outputs relies heavily on the quality of the context vector produced by the encoder. Various improvements, such as attention mechanisms [23], have been introduced to enhance the performance of Seq2Seq models by allowing the decoder to focus on different parts of the input sequence at each step of the output generation.

2.4.1 Transfer Learning from TimeGAN

Transfer learning enables reuse of pre-trained embedding and recovery networks from TimeGAN. Integrating the embedding and recovery networks from TimeGAN into the Seq2Seq model offers several advantages, including improved representation learning, as the pre-trained networks have already captured essential temporal dependencies, providing a richer data representation. Furthermore, enhanced forecast accuracy is achieved due to the latent representations learned by TimeGAN, which contribute to more accurate and stable forecasts, particularly over long horizons [24].

2.4.2 Loss Function

During fine-tuning, the Seq2Seq model is trained to minimize a loss function that quantifies the difference between the predicted and actual future values. Common loss functions include MSE and Mean Absolute Error (MAE). The optimization process involves backpropagation and gradient descent methods, where the gradients of the loss function with respect to the model parameters are calculated and used to update the weights, ensuring accurate forecasts.

Integrating the MCFD loss function described in Section 2.3.2 into the Seq2Seq model's training process emphasizes the correct prediction of market direction, rather than just minimizing the numerical error between predicted and actual values, which is challenging to accomplish within the original TimeGAN framework, due to its intricate and sensitive construction.

The combined approach leverages the strengths of traditional loss functions like MSE and MAE to predict precise values while incorporating MCFD for directional accuracy. This hybrid loss function strategy aims to enhance the model's overall performance, particularly in financial contexts where directionality is crucial for decision-making.

2.5 Portfolio Optimization

MPT introduced by Markowitz in his seminal *Portfolio Selection* paper [1], describes a mathematical framework for selecting an optimal portfolio of assets that maximizes expected returns for a given risk tolerance. The theory posits that maximizing expected returns is desirable, while the variance of returns (i.e., risk) is undesirable. Therefore, the term ***expected returns - variance of returns*** rule is coined, representing the balance of returns maximization for a given risk tolerance.

In general, we can express the expected returns of a portfolio as follows:

$$\mathbf{E}(R_P) = \sum_i w_i \mathbf{E}(R_i) \tag{2.21}$$

where R_P is the return of the portfolio, w_i is the weight of the asset i in the portfolio such that the sum of all weights is equal to 1, and R_i is the return of the asset i.

The return variance of the portfolio is given by the following:

$$\sigma_P^2 = \sum_i \sum_j w_i w_j \sigma_i \sigma_j \rho_{ij} \tag{2.22}$$

where σ_i is the standard deviation of the returns of the asset i and ρ_{ij} is the correlation of the returns of the assets i and j. It should be noted that $\sigma_i \sigma_j \rho_{ij}$ is the covariance of the returns of the two assets, which allows the equation to be rewritten as:

$$\sigma_P^2 = \sum_i \sum_j w_i w_j \sigma_{ij} \tag{2.23}$$

The volatility of the portfolio return, σ_P, is simply the square root of 2.23. For a few assets, this problem can be expressed analytically, but as the number of assets grows, the complexity of the equation increases. Therefore, using matrix algebra is much more efficient, resulting in the following two key expressions.

$$\mathbf{E}(R_P) = \mathbf{w}^{\mathbf{T}} \boldsymbol{\mu} \tag{2.24}$$

where $\boldsymbol{\mu}$ is a column vector of expected asset returns, and:

$$\sigma_P^2 = \mathbf{w}^{\mathbf{T}} \boldsymbol{\Sigma} \mathbf{w} \tag{2.25}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the assets.

2.5.1 Efficient Frontier

The central insight in Markowitz's framework was the identification that two different portfolios can have the same expected returns but different return variances. Consider the following example and explanation given in a lecture by Haugh [5]:

Portfolio A: Single stock portfolio such that $w_1 = 1$ and $w_i = 0$ for $i = 2, ..., n$

Portfolio B: An equal-weighted portfolio such that $w_i = \frac{1}{n}$ for $i = 1, ..., n$

We thus have the following:

$$\begin{aligned} \mathbf{E}(R_A) = \mathbf{E}(R_B) &= \mu \\ \sigma_A^2 &= \sigma^2 \\ \sigma_B^2 &= \frac{\sigma^2}{n} \end{aligned}$$

An investor would logically choose portfolio B in this case since the expected returns are the same, while the variance is lower in B. Portfolio B thus gains the benefits of diversification without the loss of expected returns.

To find ”efficient” portfolios, i.e., portfolios that maximize returns while minimizing variance, Markowitz proposed the *Efficient Frontier*. First, it is assumed that the returns of a portfolio with n assets are multivariate normally distributed as follows:

$$\mathbf{R} \sim MVN_n(\mu, \mathbf{\Sigma}) \tag{2.26}$$

The mean-variance optimization problem is thus as follows:

$$\begin{aligned} & \min_w \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} && (2.27) \\ \text{subject to} \quad & \mathbf{w}^T\mu = \gamma && (2.28) \\ \text{and} \quad & \mathbf{w}^T 1_n = 1 && (2.29) \end{aligned}$$

where γ is a relative risk-aversion parameter of the investor. This is a simple quadratic optimization problem and can be solved using Lagrange multiplier methods. However, in many cases, including the one discussed in this paper, additional portfolio constraints such as no short-selling make the problem not analytically solvable, but instead numerically solvable. Given the chosen constraints, these types of problems are convex, and convex optimization methods can be utilized [25].

2.5.2 Sharpe Ratio and Information Coefficient

The Sharpe ratio is a financial metric used to evaluate the risk-adjusted return of an investment or portfolio. It was introduced by William F. Sharpe in 1966 and has since become a fundamental tool in modern portfolio theory and investment analysis [26]. The Sharpe ratio measures the performance of an investment compared to a risk-free asset, after adjusting for its risk.

The formula for the Sharpe ratio is given by:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

where:

- R_p is the average return of the portfolio or investment.
- R_f is the risk-free rate of return, typically represented by the return on short-term government bonds.
- σ_p is the standard deviation of the portfolio's excess return, which serves as a proxy for the investment's risk.

The risk-free rate (R_f) represents the return of an investment with zero risk, serving as a benchmark for evaluating the performance of other investments. The excess return ($R_p - R_f$) is the return earned beyond the risk-free rate. The standard deviation of the excess return (σ_p) is used to quantify the investment's risk or volatility.

The Information Coefficient (IC) is a statistical measure used to evaluate the skill of a financial analyst or an investment manager. It quantifies the correlation between predicted and actual returns, providing insights into the accuracy of the predictions. The IC is particularly useful in the context of quantitative finance, where it helps assess the effectiveness of predictive models or strategies [27].

Mathematically, the Information Coefficient is defined as the Pearson correlation coefficient between the predicted returns ($\hat{R}_i$) and the actual returns (R_i):

$$\text{IC} = \frac{\text{Cov}(\hat{R}, R)}{\sigma_{\hat{R}}\sigma_R}$$

where:

- $\text{Cov}(\hat{R}, R)$ is the covariance between the predicted returns ($\hat{R}$) and the actual returns (R).
- $\sigma_{\hat{R}}$ is the standard deviation of the predicted returns.
- σ_R is the standard deviation of the actual returns.

The IC ranges from -1 to 1, where:

- An IC of 1 indicates a perfect positive linear relationship between predicted and actual returns, meaning the predictions are perfectly accurate.
- An IC of -1 indicates a perfect negative linear relationship, meaning the predictions are perfectly inversely related to actual returns.
- An IC of 0 indicates no linear relationship between predicted and actual returns, implying the predictions have no value.

A higher IC suggests better predictive accuracy and, therefore, higher skill in forecasting returns. Typically, an IC above 0.1 is considered good, while an IC above 0.2 is considered very strong in financial contexts.

The Information Coefficient is a key component in evaluating the performance of quantitative investment strategies and models. It helps in determining the strength of the signals generated by these models and their potential to generate alpha, which is the excess return over a benchmark index.

However, it is important to note that the IC only measures linear relationships and may not capture more complex, non-linear relationships between predicted and actual returns. Additionally, the IC can be sensitive to the time period and the sample size over which it is calculated, so it is crucial to consider these factors when interpreting the IC.

Chapter 3

Method

3.1 Research Process

The UML diagram in 3.1 outlines the research process, which includes defining the research question, conducting a literature review, and synthesizing the findings to guide the model design. The initial step is to create a research question, followed by a literature search to collect existing research relevant to the topic. This involves selecting the papers collected and categorizing them according to their content. After the literature search, the papers are analyzed and the results are synthesized to be used to steer the model design phase.

The first model designed in this research is based on the TimeGAN approach. Following its design, the performance of the model is evaluated. If it meets an adequate performance level, the process continues with the extraction of networks from the TimeGAN model and the design of the Seq2Seq model. Should the performance level not be met, the process will repeat the previous steps of collecting more research and redesigning the models in an iterative way.

Subsequently, the Seq2Seq model undergoes similar stages of training, performance evaluation, and optimization. The research process includes iterative cycles of redesign and retraining of both models until they achieve satisfactory performance.

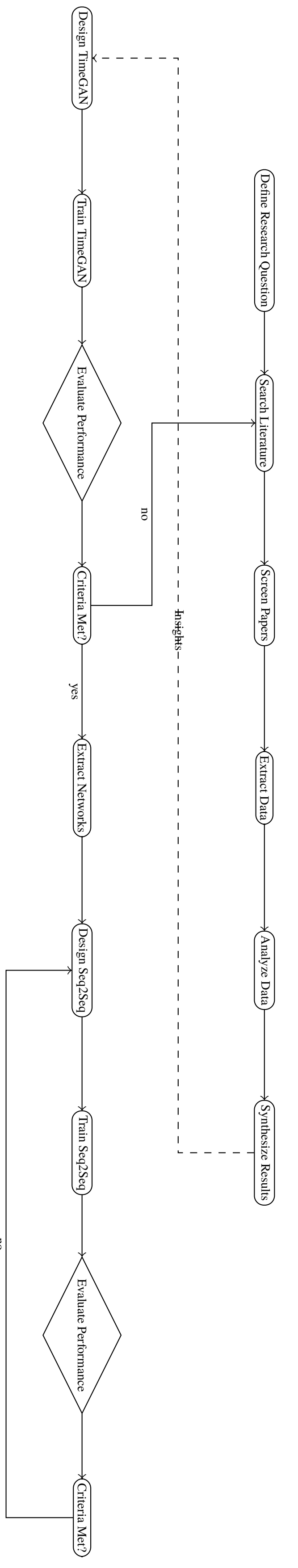


Figure 3.1: UML-diagram of the Research Process and Model Implementation

3.2 Research Paradigm

Stock price prediction and portfolio optimization align with a positivist research paradigm, given the underlying assumption that financial markets exhibit statistical patterns that can be measured and quantified. Quantitative methods, including statistical analysis and machine learning models, are employed to model these patterns objectively.

Market assumptions, which aim to explain specific market behaviors, have been extensively researched and empirical analysis and evidence is prevalent. Theories such as Gaussian noise (as described in Section 2.3) and the Universal Approximation Theorem are widely studied and supported by empirical evidence and statistical analysis.

This research also depends on the use of empirical data, such as historical data OHLCV, to generate evidence through analysis and validate models and hypotheses. Furthermore, the evidence can be replicated using the same data and models OHLCV, enhancing the reliability of the results and models.

3.3 Data Collection

The data collected was in collaboration with Fjärde AP-fonden (AP4). The fund provided us with access to a broad number of different indices from multiple market data suppliers, but for this research's purposes, specifically MSCI.

MSCI is a global supplier of financial services and market data. With more than 4500 equity indices available, data was vastly available for our research and allowed us the freedom to tailor our strategy as we wanted. Access to high-frequency data and liquid constituents provided us with the possibility to create a more reliable model and also higher degrees of freedom to experiment with different setups of hyperparameters.

3.3.1 Sampling and Sample Size

As stated above, access to 4,500 equity indices provided us with the opportunity to select from almost any market in the world. Based on recommendations from the analysts at AP4, we chose the MSCI US index. This index is designed to measure the performance of the large- and mid-cap segment of the US market, comprising 612 constituents and covering approximately 85% of the market. The primary reason for selecting this index was to ensure reliable high-frequency data from liquid constituents, minimizing the need for extensive pre-modeling data cleaning.

Of the 612 constituents, we filtered and selected the 10 largest constituents by index weight due to computational constraints. This selection process was necessary due to the lack of available data or exclusions made by AP4, most notably by excluding companies like ExxonMobil (for ESG reasons) and Boeing (for manufacturing components for nuclear weapons).

We identified the 10 largest constituents according to Morningstar [28] and S&P [29]. Companies with higher weights have a more significant impact on index performance than those with smaller weights, as the returns of each company are weighted by its index weight. However, the number of constituents selected can be adjusted on the basis of the computational power available to the user.

3.4 Assessing Reliability and Validity of the Data

MSCI is a leading financial services and data provider serving some of the largest companies worldwide. Their clients depend on accurate, clean, and timely data delivery, as even minor errors can significantly impact their strategies. According to their 2019 report [30], MSCI employs 150 researchers to identify risk and return drivers in global markets. This research is used to construct indices that accurately reflect global markets.

The research team is also responsible for data cleaning, achieving an accuracy rate of 99.6%. MSCI calculates 160,000 indices daily at the end of the day and 8,500 indices in real-time, maintaining an almost perfect on-time delivery record.

Since the data consists of OHLCV market data, it can be easily verified against other sources of financial data, such as the Yahoo Finance API, ensuring its validity.

3.5 Experimental Design

3.5.1 Hardware and Software Specifications for Implementation

The primary data pre-processing was carried out using ***pandas*** [31] and ***NumPy*** [32]. The models were developed using ***Python*** version 3.10.4 and ***TensorFlow*** version 2.10 with ***Keras*** version 2.10 [33]. It should be noted that the migration from TensorFlow 1.x and Keras 1.x to 2.x involves significant changes, and the 2.x versions offer benefits such as eager execution, automatic differentiation, and tighter integration into the TensorFlow framework.

To monitor performance and the impact of hyperparameter adjustments and model structure changes, ***MLflow*** [34] was utilized due to its lightweight and streamlined API with extensive functionality.

The models were trained on a single NVIDIA RTX 4070 Ti using default settings. Training a single TimeGAN model for one ticker, with data ranging from 2010 to 2024, 20,000 epochs, a batch size of 32, and a learning rate of 0.001, took approximately two hours. The Seq2Seq models, trained with the same hardware setup, a batch size of 16, a learning rate of 0.001, and 100 epochs, took about 70 ms per step and around 1 second per epoch, depending on the date interval set during training.

3.5.2 Test Environment

The testing environment for the TimeGAN models consisted of a script that initialized an MLflow run, logged parameters, and saved artifacts such as plot outputs and model summaries. Testing was carried out on data from 2010-01-01 to the end of the time series, using weekly values to minimize the impact of noise on stock prices.

A set of necessary hyperparameters was defined and the test proceeded with a train/test split of the data followed by training. After training, the model was evaluated on the test data and sequences of predictions were generated. These predictions were then inputted into a plotting algorithm that produced Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) plots, along with sample plots comparing the predictions with actual values. Finally, the models were saved locally for future use.

After training and testing the TimeGAN models, the saved models were loaded into a script that created a Seq2Seq model. This script tuned the model for the β parameter of the Seq2Seq loss function and generated predictions.

3.6 Data Analysis

Little data analysis and cleaning was performed in the preprocessing stage. The quality of the MSCI data allowed us to skip major parts of the processing. The preprocessing of data can be summarized into the following steps.

1. Extract the desired ticker from the dataset
2. Split the data into train, test and validation sets
3. Scale the data with MinMaxScaler
4. Create sliding windows of a certain sequence length and forecast length.
5. The dates of the sliding windows for the different data sets are indexed.
6. The *Close* feature values are extracted and kept as targets for the training and testing. The close feature value is the closing price that day for the company's stock.

7. Convert the arrays containing the data into Tensors, and then combine the corresponding Tensors into train and test TensorFlow Datasets. The TensorFlow Dataset objects allow for dynamic and simple batching.

3.7 Evaluation Framework

3.7.1 Evaluation of TimeGAN and Seq2Seq

Evaluating the performance of the TimeGAN model was done using PCA and t-SNE as in the paper by Yoon and Jarrett [9]. Both t-SNE and PCA are used to represent and visualize high-dimensional data in lower dimensions. The overlap between the sampled data and the actual data shows the ability of the model to generate realistic samples from an estimated distribution.

The evaluation of the Seq2Seq model was carried out mainly using the ***Alphalens*** Python library [35]. Alphalens is used for the performance analysis of predictive stock factors, where our predicted prices are used as stock factors. The analysis consists of statistics and plots for the following:

- Returns Analysis
- Information Coefficient (IC) Analysis
- Turnover Analysis
- Grouped Analysis

The focus of this report lies within the **IC** analysis, which is a measure of the degree to which the financial forecasts match the actual financial results.

3.7.2 Evaluation of Portfolio Optimization

The portfolio optimization algorithm will be evaluated versus a benchmark. As the constituents and data were sampled from the MSCI US index, we also chose to benchmark our portfolio versus the index. We perform two benchmarks. First, we benchmark our portfolio against the entire benchmark with all 611 constituents. Secondly, we benchmarked our portfolio versus a fictional portfolio consisting of the same constituents as ours, however, with their MSCI US weights reweighted over time to make a complete portfolio where the sum of the weights is 1. For example, suppose that we chose three stocks from the MSCI index, and they have weights of 3%, 4% and 5%, respectively, then a scaling factor can be calculated by simply summing the weights and dividing each weight by the sum to scale them up.

Chapter 4

Implementation

4.1 TimeGAN

The implementation of TimeGAN was large based on the code written by the *YData* team and their open-source Python package *ydata-synthetic* which contains a multitude of synthetic data generators for tabular and time series data [36]. The code itself is implemented according to the original paper by Yoon et al. [9].

However, there are some key differences between our revised version of the code and theirs. Firstly, we have chosen to implement most of the networks using GRUs. Through tests, we found that the GRU implementation performed better than the LSTM implementations, which is in accordance with the previous research described in section 2.3.1.2. Furthermore, we chose to increase the amount of GRU units in each network to 1024, 512 and 256 for the first, second, and third layers, respectively. This increase in units prevented, in most cases, the model from underfitting and flat-lining its predictions.

Secondly, through further research, we decided to change the entire architecture of the Embedder and Recovery networks. The original GRU based implementations of these were not adequate for our Seq2Seq model, and thus needed to be improved. We therefore introduced RHN layers with weight-tying, as used in the original paper [18].

Lastly, through Hyperband tuning implemented by the Keras team based on the paper by Li et al. [37], the hyperparameters could be somewhat tuned for loss minimization. However, this tuning was limited due to the long training time of the TimeGAN. The final model configuration for TimeGAN can be seen in table 4.1.

Parameter	**Value**
Batch Size	32
Learning Rate	0.001
Latent Dimension	24
Gamma	0.01
Sequence Length	6
Number of Features	5
Epochs	20000

Table 4.1: Model Configuration of TimeGAN

4.2 Seq2Seq

The Seq2Seq model is a simple implementation in which the Embedder network from TimeGAN acts as the Encoder, and the Recovery network acts as the Decoder. The networks are loaded into a model, we call it Seq2Seq. The first RHN layer of both networks is frozen. This is done to prevent these layers from having their underlying temporal dependencies gained during TimeGAN training altered in future training. During the backpropagation step of the Seq2Seq training, these layers will not be affected. Freezing these layers made a big difference in the performance of the model to predict closing prices, since the deep temporal dependencies are kept.

The loss function for the Seq2Seq model incorporates the MCFD loss described in Section 2.3.2 together with the traditional MSE function. These two are weighted together with a β parameter that controls the impact of the directional loss. The final loss is as follows:

$$L_{Seq2Seq} = MSE + \beta \cdot MCFD \tag{4.1}$$

The Seq2Seq model, like the TimeGAN model, was tuned with the Hyperband tuner, resulting in the final configuration of the model seen in Table 4.2 where the value β_{TICKER} represents the best weighting factor for that ticker.

Parameter	**Value**
Batch Size	16
Learning Rate	0.001
Epochs	100
Sequence Length	6
Forecast Length	5
Beta (β)	β_{TICKER}

Table 4.2: Model Configuration of Seq2Seq.

4.2.1 Seq2Seq Training Process

The training process for the Seq2Seq model involves two main phases:

Phase 1 The embedding and recovery networks are pre-trained as part of the TimeGAN framework. During this phase, these networks learn to accurately map and reconstruct time-series data, establishing robust latent space representations.

Phase 2 Post pre-training, the embedding and recovery networks are integrated into the Seq2Seq model. The entire model, including the pre-trained networks, is then fine-tuned on the forecasting task. This phase involves training the model to minimize the prediction error on future time steps, using supervised learning techniques to adjust the parameters for optimal forecasting performance.

4.3 Overview of Training-Test Periods

The following section provides an overview of the different training and testing periods for the models used in our study.

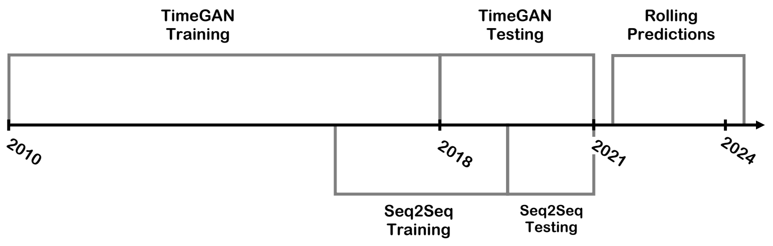


Figure 4.1: Overview of different train and test splits

4.3.1 TimeGAN

The model TimeGAN was trained on data spanning 2010 to 2018. The training period aimed to capture a wide range of temporal dynamics present in the data. After the training phase, the model was tested on data from 2018 to 2021 to evaluate its performance in predicting future sequences.

4.3.2 Seq2Seq

Similarly, the Seq2Seq model underwent its training phase from 2016 to 2018. After training, the Seq2Seq model was tested in a different test period compared to TimeGAN, specifically from 2019 to 2021.

4.3.3 Rolling Predictions

After the initial testing phases for both models, we employed rolling predictions for the final Seq2Seq from June 2021 to 2024. This phase is crucial for assessing the model performance in a real-time forecasting environment. In the rolling phase, the model was retrained on the data going back 1 year and then generated a forecast for the upcoming 5 weeks; this was repeated for all forcasts during this period.

4.4 Portfolio Optimization

After the rolling predictions were generated, the portfolio optimization step could begin. For simplicity and precision sake, we decided to use the ***skfolio*** Python library [38]. The library is built on top of the well-established scikit-learn library, which is used for data analysis and machine learning. Using the ***skfolio*** library, we built our own prior estimator using a shrinkage expected returns, and a denoising covariance estimator. This was then fed into the Mean-Risk optimization model, with the objective of maximizing Sharpe ratio and variance as the risk measure. The minimum weights were set to 0 to constrain the model into being long only for the sake of being realistic in AP4's case, and the risk-free rate was set to 0. The model was then trained using a Walk Forward cross-validator, which splits the time series data samples in a walk forward logic with 5 training data points, and 3 testing data points. For the full description, please see the ***skfolio*** API-references in the footnotes. [*]

The results could then be visualized as plots of the Sharpe ratio and the asset weights over time.

4.5 Financial Evaluation Framework

In the implementation of the financial evaluation framework, a comprehensive approach was taken to assess and analyze financial predictions. This section details the steps and methodologies applied to achieve the evaluation goals.

4.5.1 Data Preparation

To start with, financial predictions and true values were read. The prediction data was then transformed to represent percentage

[*] Skfolio Model Selection - WalkForward and Skfolio Optimization - MeanRisk

changes and appropriately shifted to align with future returns. This transformation was necessary to capture the financial returns based on the predictions.

4.5.2 Factor Data Calculation

Using the Alphalens library, clean factor and forward returns data were generated. This involved the computation of quantiles, which facilitated a more granular analysis of the data. The choice to use quantiles was to categorize the predictions into discrete buckets for a better performance evaluation.

4.5.3 Performance Metrics

Several performance metrics were computed to evaluate the efficacy of the predictions. The factor returns were calculated to determine the overall return associated with the predictions. The decision to compute long-short and group-neutral returns was made to provide a comprehensive view of performance under different strategies. Additionally, the mean return by quantile metric was essential for understanding the average return per quantile, providing insights into the effectiveness of predictions within different buckets.

Alpha and beta values were derived to measure excess returns and sensitivity to market movements, respectively. Furthermore, IC was used to gauge the predictive power of the factors. The turnover and autocorrelation metrics were calculated to assess the stability and consistency of the factors over time.

4.5.4 Visualization and Reporting

Comprehensive tear sheets were generated using ***Alphalens*** [35] to visually represent the analysis results. This step was crucial in effectively communicating the findings and providing a clear visual summary of the performance metrics.

The design decisions made throughout the implementation process were driven by the goal of obtaining a detailed and nuanced understanding of the prediction performance. The choice of using ***Alphalens*** for factor analysis, the computation of various performance metrics, and the detailed visualization aimed to ensure a robust and comprehensive evaluation framework. By preparing the data, calculating a wide array of performance metrics, and utilizing visualization tools, the implementation effectively met the goals of providing a comprehensive financial evaluation.

Chapter 5

Results and Analysis

5.1 TimeGAN and Seq2Seq

5.1.1 PCA and t-SNE Analysis

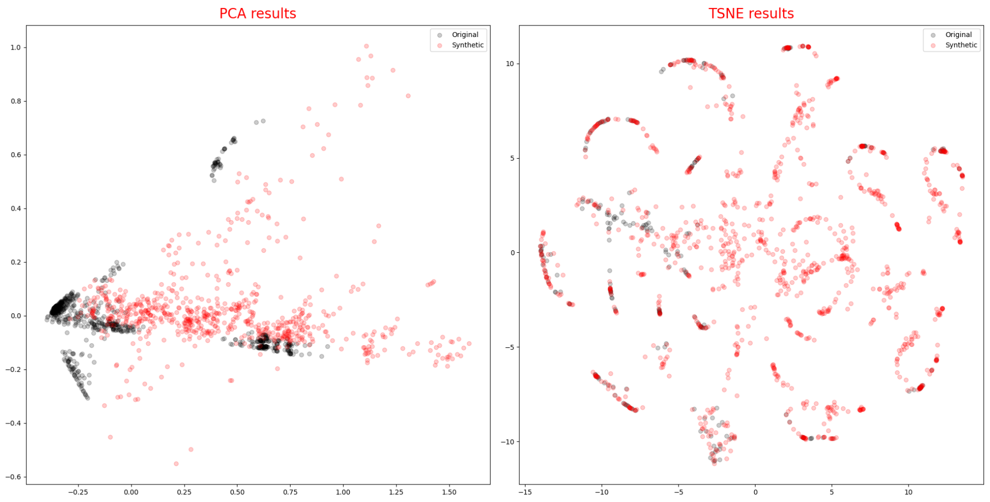


Figure 5.1: PCA and t-SNE analysis of TimeGAN Generator output for Microsoft [*]

In the PCA results for Microsoft (left graph) as seen in Figure 5.1, the original data, represented by black dots, appear to cluster

[*]All plots for the analysis of PCAs and t-SNEs of the models can be found in Appendix A.

more tightly in specific regions. In contrast, the synthetic data, represented by red dots, show a broader spread, indicating that the TimeGAN generator has introduced some variability. Although there is some overlap between the original and synthetic data points, there are noticeable differences.

In the results t-SNE (right graph) of Figure 5.1, the graph reveals more pronounced clustering and separation patterns compared to the graph PCA. Original data and synthetic data form distinct clusters. The synthetic data, though forming patterns similar to the original data, is more dispersed. There are regions where the synthetic and original data overlap, suggesting that the TimeGAN generator has captured some of the original data's structure but has also introduced its own artifacts.

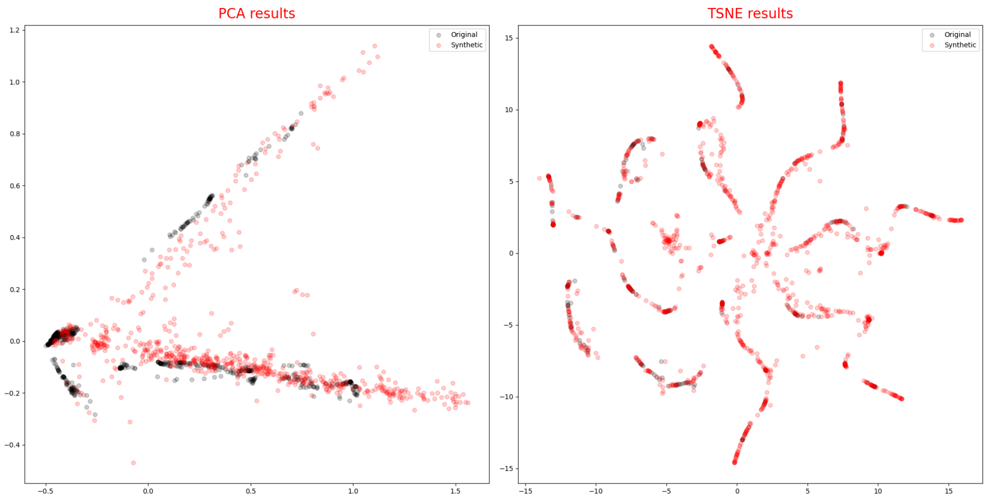


Figure 5.2: PCA and t-SNE analysis of TimeGAN Generator output for Netflix

Turning to Netflix, the PCA results in Figure 5.2 similarly show that the original data cluster more tightly and align in specific directions. The synthetic data, represented by red dots, are broadly spread out, mirroring the variability seen in the Microsoft dataset. The overlap between original and synthetic data points is comparable to that in the Microsoft dataset but greater, indicating a higher degree of structural capture by the TimeGAN generator, but there still remains a significant introduction of variability.

The t-SNE results for Netflix shows that the original data form distinct clusters, similar to the Microsoft data, but the synthetic data exhibit a more intricate pattern of dispersion. The red dots follow a more detailed and elaborate structure, suggesting that the generator has captured a higher degree of the original data' complexity. The overlap between original and synthetic data points is more pronounced in certain regions, highlighting the generator's ability to mimic the original data's structural nuances more effectively than in the Microsoft dataset.

5.1.2 Prediction of Closing Prices

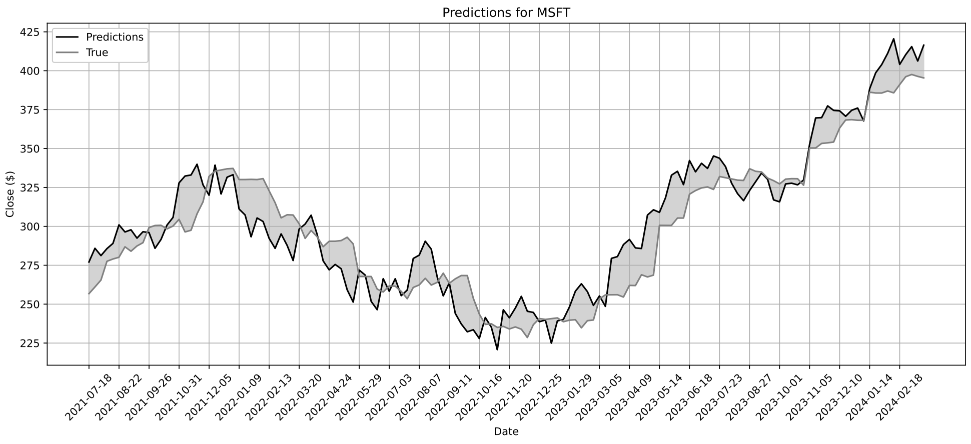
Figure 5.3 presents the 5-week rolling predictions and actual closing prices for Microsoft, Netflix, and Nvidia. The plots include the actual prices represented by a solid gray line and the predicted prices by a solid black line, along with a shaded area indicating the difference between the predicted and actual prices.

In subplot 5.3a, the predicted closing prices for Microsoft (MSFT) are shown alongside the actual closing prices. The plot demonstrates that the model's predictions closely follow the actual price movements, particularly during periods of significant trends. The differences are generally small, suggesting a high level of accuracy in the predictions. However, there are minor deviations where the predictions slightly lag behind or exceed the actual prices.

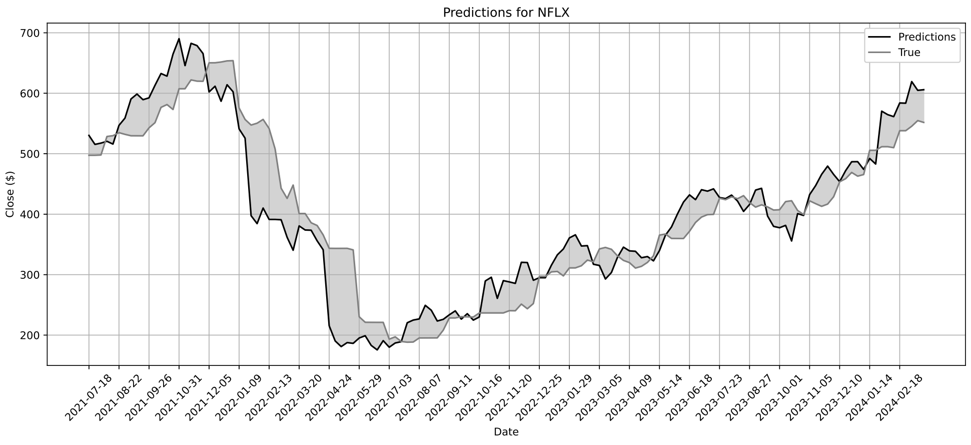
The subplot 5.3b displays the predicted and actual closing prices for Netflix (NFLX). Similarly to Microsoft, the predictions for Netflix generally align well with the actual prices. The model captures the general trend and the major price movements accurately, with the differences remaining relatively small throughout the period. In particular, the model effectively predicts the recovery phase of Netflix's stock price after a period of decline. There are occasional discrepancies, particularly during rapid price changes, where the model's predictions may not fully capture volatility.

The subplot 5.3c shows the predicted and actual closing prices for Nvidia (NVDA). The model demonstrates strong predictive performance for Nvidia, with the predicted prices closely

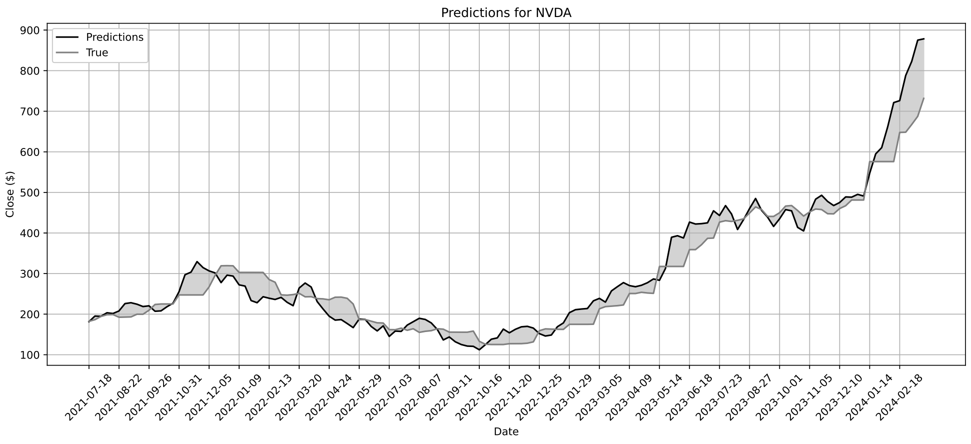
*All plots for the closing prices can be found in Appendix B



(a) Prediction for Microsoft



(b) Predictions for Netflix



(c) Predictions for Nvidia

Figure 5.3: Prediction of closing prices for MSFT, NFLX and NVDA *

matching the actual prices. The differences are consistently small, indicating a high degree of precision in the predictions. The model successfully captures both the upward and downward trends in Nvidia's stock price, including periods of rapid growth. As with the other stocks, minor deviations are observed, particularly during highly volatile periods, but overall predictive accuracy remains high.

5.1.3 Information Coefficient Analysis

Table 5.1: IC Analysis

Statistic	**1W**	**5W**	**10W**
Mean IC	-0.019	0.056	**0.069**
IC Std.	0.367	0.376	**0.337**
Risk adjusted IC	-0.052	0.150	**0.205**
t-statistic (IC)	-0.587	1.698	**2.320**
p-value (IC)	0.559	0.092	**0.022**
IC Skew	0.188	0.092	-0.142
IC Kurtosis	-0.281	-0.955	-0.053

Table 5.1 presents the IC Analysis for three different evaluation horizons: 1 week (1W) , 5 weeks (5Ws), and 10 weeks (10Ws); first week, one forecast, and two forecasts, respectively. The IC is a measure used to evaluate the predictive power of our financial models, with a higher IC indicating better predictive accuracy.

For the 1W horizon, the mean IC is -0.019, suggesting that the model has a slightly negative predictive power over this short-term period. The IC standard deviation is 0.367, indicating moderate variability in the IC values. The adjusted risk IC is -0.052, which remains negative, highlighting that even when adjusted for risk, the predictive power is not favorable. The t-statistic for IC is -0.587 with a p-value of 0.559, indicating that the mean IC is not statistically significant. The skewness of the distribution IC is 0.188, suggesting a slight positive skew, and kurtosis is -0.281, indicating a flatter distribution compared to a normal distribution.

For the 5W horizon, the mean IC improves to 0.056, indicating a positive but modest predictive power. The IC standard deviation is slightly higher at 0.376. The risk adjusted IC is 0.150, suggesting a favorable risk adjustment. The t-statistic for IC increases to 1.698 with a p-value of 0.092, approaching statistical significance at the level 10%. The IC skew is 0.092, indicating a near-normal distribution, while the kurtosis is -0.955, suggesting a more pronounced flatness in the distribution.

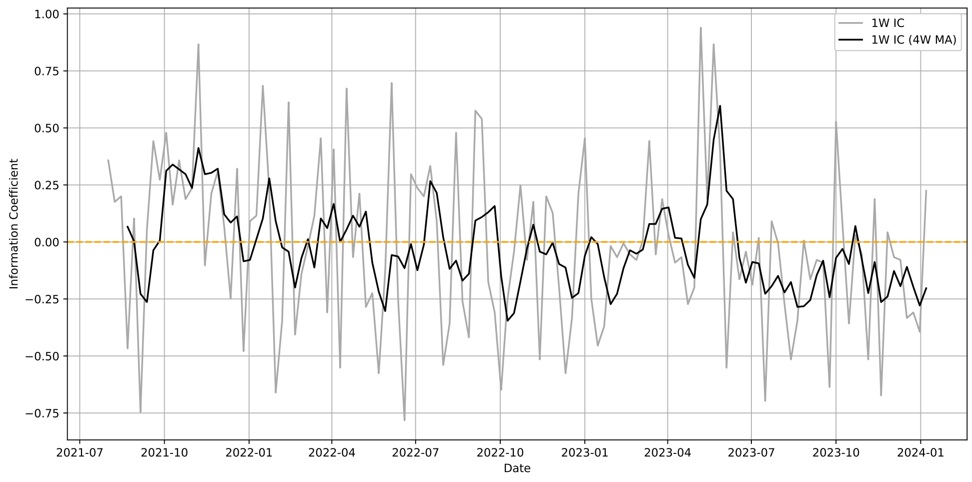
For the 10W horizon, the mean IC is 0.069, reflecting a stronger positive predictive power over a longer period. The standard deviation IC decreases to 0.337, indicating less variability in the values IC. The risk adjusted IC improves to 0.205, suggesting a favorable adjustment for risk. The t-statistic for IC is 2.320 with a p-value of 0.022, indicating that the mean IC is statistically significant at the level 5%. The skewness of the distribution IC is -0.142, showing a slight negative skew, and the kurtosis is -0.053, which is close to a normal distribution.

5.1.3.1 IC Analysis over Time

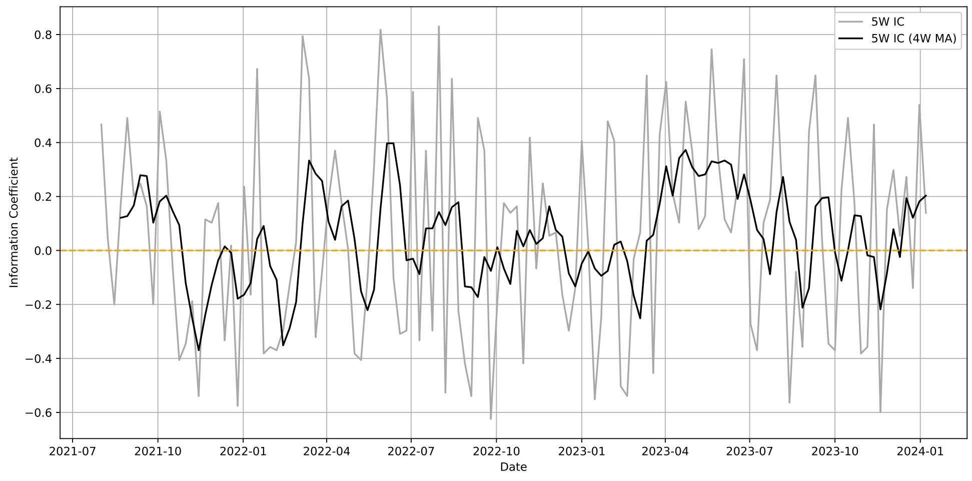
Figure 5.4 presents the time series of IC for three different evaluation horizons: 1W, 5W, and 10Ws. These graphs illustrate IC over time, providing insight into the consistency and reliability of the predictive power of the model at different periods.

In subplot 5.4a, the 1W Period Forward Return IC shows significant fluctuations around the mean value IC, indicated by the orange dashed line. The black line represents the moving average of IC, which smooths out short-term variations and highlights the overall trend. Throughout the period 2021-07 to 2024-01, the 1W IC oscillates widely, frequently crossing the zero line, indicating alternating periods of positive and negative predictive power. Despite these fluctuations, there is a slight negative trend in 1W IC, reflecting the challenges of achieving consistent short-term predictive accuracy.

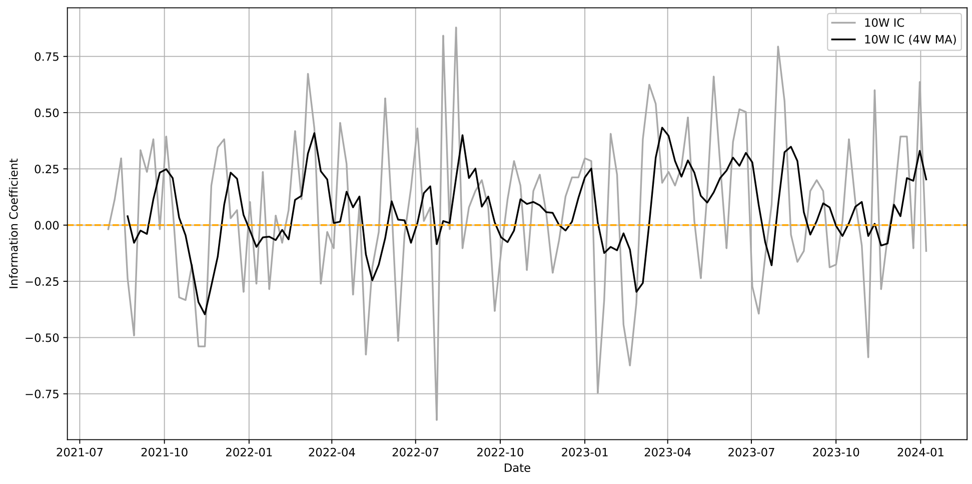
The subplot 5.4b illustrates the 5W Period Forward Return IC. Compared to 1W IC, 5W IC exhibits somewhat smoother fluctuations and a more stable pattern around its mean value. The moving average (black line) suggests that 5W IC maintains a



(a) 1W Period Forward Return IC



(b) 5W Period Forward Return IC



(c) 10W Period Forward Return IC

Figure 5.4: IC Timeseries

relatively consistent level of predictive power, with fewer extreme deviations. In particular, there are periods where the IC are consistently positive, particularly in the latter part of the analysis, indicating periods of more reliable predictions.

The subplot 5.4c shows the 10W Period Forward Return IC. This longer horizon exhibits the smoothest IC time series among the three horizons, with fewer extreme fluctuations and a clearer trend. The moving average line indicates that 10W IC maintains a generally positive level throughout the analyzed period. There are extended periods where IC is above zero, particularly from mid-2022 onward, suggesting that the predictive power of the model is more robust and reliable over longer periods.

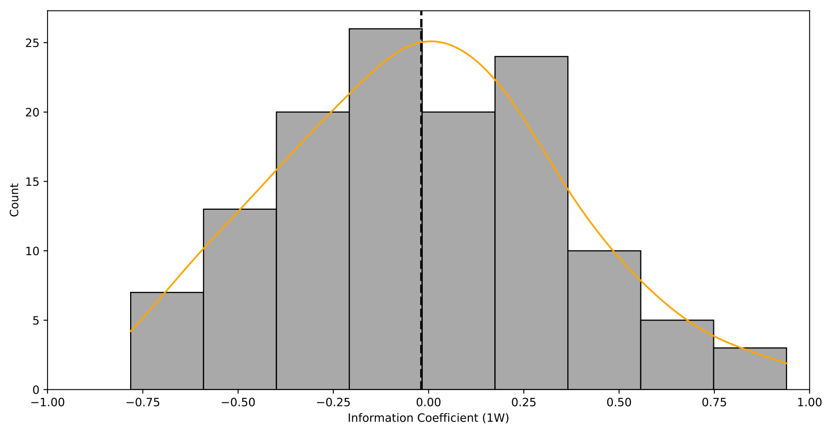
5.1.3.2 Normal Distribution of IC

Figure 5.5 presents the distribution of IC for three different evlauation horizons: 1W, 5Ws, and 10Ws. These histograms, overlaid with Kernel Density Estimates (KDEs), provide information on the spread, central tendency, and shape of the distributions IC.

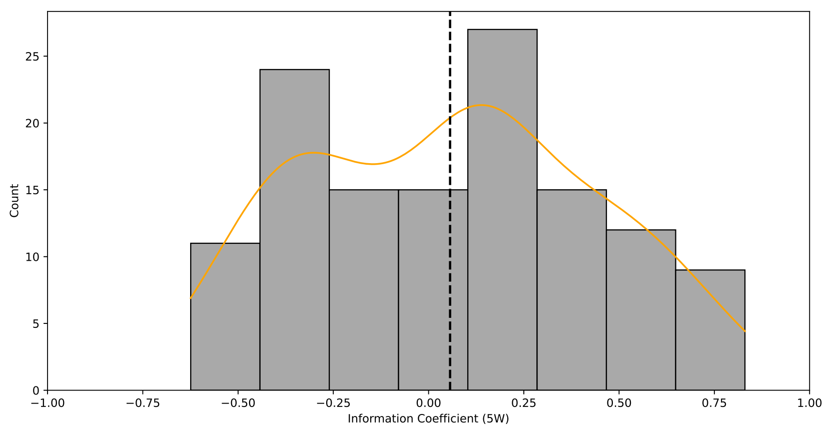
In subplot 5.5a, the 1W period IC distribution appears relatively broad with a mean close to zero, as indicated by the vertical dashed line. The histogram shows a wide range of IC values, with a noticeable skewness toward the negative side. This distribution reflects the high variability and frequent negative predictive power observed in the short term. The KDE curve also highlights the multimodal nature of the distribution, suggesting multiple peaks and troughs in the IC values over the 1W periods.

The subplot 5.5b depicts the 5W period IC distribution, which shows slightly narrower spread compared to the 1W IC distribution. The mean IC for this period is also close to zero, as indicated by the vertical dashed line. The distribution shows a clear bimodal pattern, with significant frequencies around both -0.25 and 0.25, suggesting that the information coefficient values tend to cluster around these two points, however with a slight positive skewness.

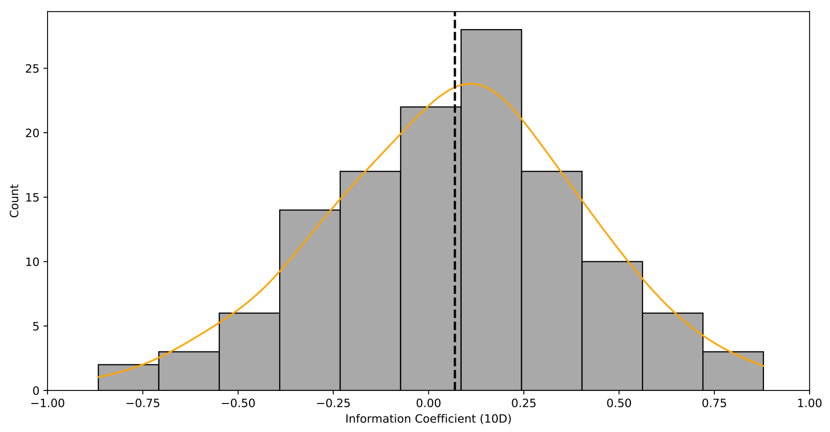
The subplot 5.5c illustrates the distribution of the period 10W IC, which exhibits the narrowest spread and the most symmetrical shape among the three horizons. The mean IC is slightly positive,



(a) 1W Period IC



(b) 5W Period IC



(c) 10W Period IC

Figure 5.5: Distribution of IC

reflecting a trend toward better predictive power over longer periods. The histogram and the KDEs curve show a bell-shaped distribution, closely resembling a normal distribution. This indicates that the IC values are more consistently clustered around the mean, with fewer extreme values. The reduced kurtosis and skewness in the distribution 10W IC suggest a more reliable and robust predictive power over the longer horizon.

5.2 Portfolio Optimization

5.2.1 Cumulative Returns

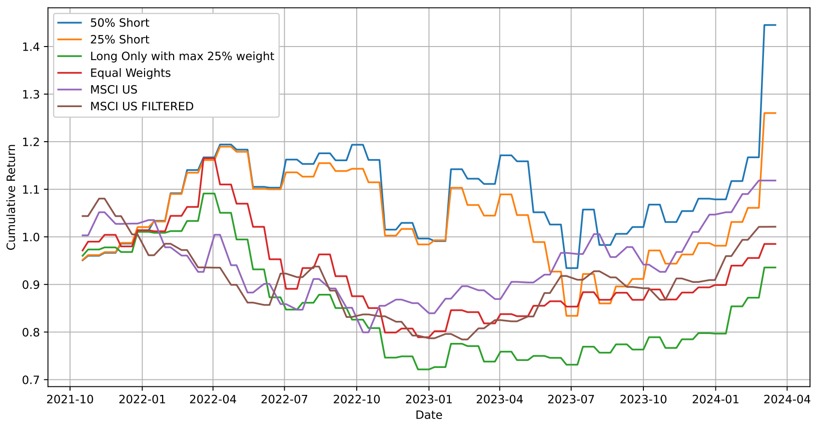


Figure 5.6: Cumulative returns of MVO portfolios with Seq2Seq predictions versus MSCI US filtered, and unfiltered index.

The results of the cumulative return analysis for the specified period are presented in Figure 5.6. The figure compares the performance of various strategies: 50% Short, 25% Short, Long Only with a maximum weight of 25%, and the benchmarks: Equal Weights, MSCI US and MSCI US Filtered.

From the graph, it is evident that the 50% Short strategy consistently outperformed the other strategies, especially in the later period of the analysis, showing a sharp increase in cumulative return starting around the beginning of 2024. This strategy reached a cumulative return of approximately 1.45 at the end of the analysis period. The 25% short strategy also showed strong performance, closely following the 50% short strategy, and achieving a cumulative return slightly above 1.4 at the end of the period.

The Equal Weights and Long Only strategies exhibited moderate performance, with cumulative returns gradually increasing throughout the period but not matching the peak levels of the short strategies. The long-only strategy with a maximum weight of 25% showed more fluctuation and a lower overall return, indicating higher volatility and less consistency in performance compared to the short strategies.

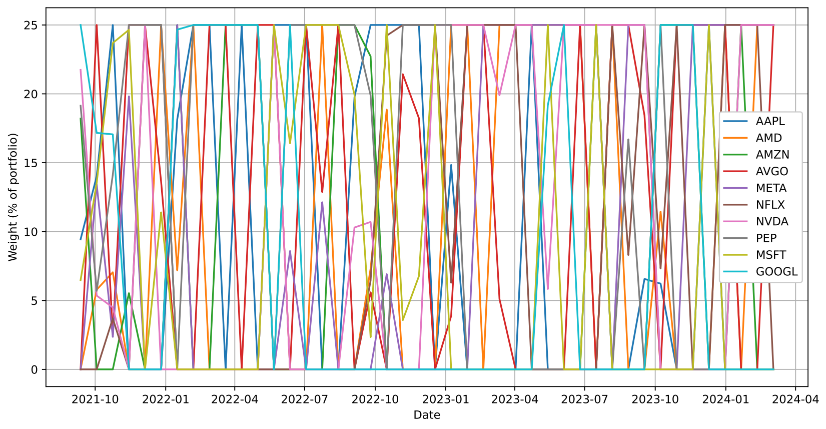
The MSCI US index and its filtered variant displayed more stable but lower returns compared to the other strategies. Both indices had cumulative returns that topped around 1.2, indicating that while they provided steady growth, they outperformed strategic allocations involving short positions.

5.2.2 Portfolio Weights

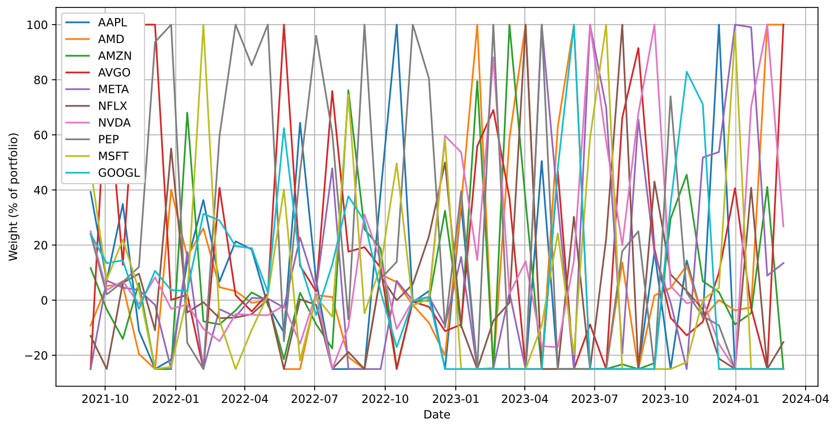
The portfolio weights for various strategies during the specified period are illustrated in Figure 5.7. The figure includes three subplots: 5.7a Weights of the long-only portfolio with a 25% maximum weight constraint, 5.7b Weights of the long-short portfolio with a 25% maximum short weight constraint, and 5.7c Weights of the long-short portfolio with a 50% maximum short weight constraint.

In subplot 5.7a, the long-only portfolio with a 25% maximum weight constraint shows significant variation in the weights of individual stocks over time. The constraint prevents any single stock from exceeding 25% of the portfolio, ensuring diversification. However, frequent and substantial shifts in weights suggest a highly dynamic adjustment strategy in response to market conditions. Stocks such as AAPL, AMZN, NVDA, and MSFT frequently reach the upper weight limit, indicating their significant impact on portfolio performance.

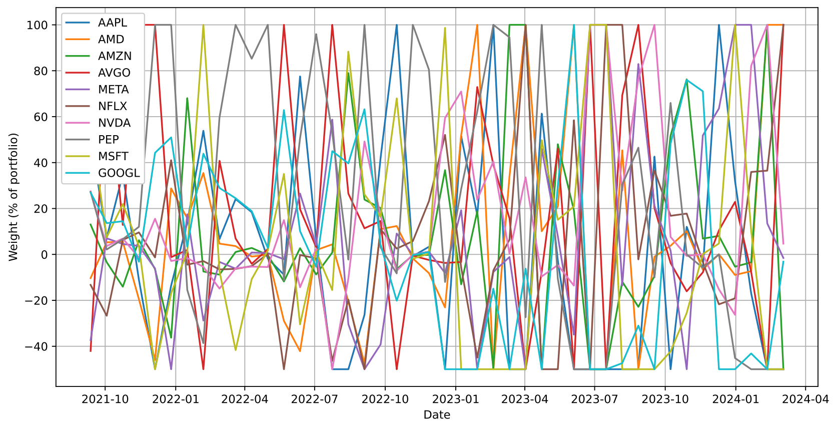
The subplot 5.7b presents the weights of the long-short portfolio with a 25% maximum short weight constraint. This strategy



(a) Weights of long only portfolio with 25% maximum weight constraint.



(b) Weights of long-short portfolio with 25% maximum short weight constraint.



(c) Weights of long-short portfolio with 50% maximum short weight constraint.

Figure 5.7: Portfolio asset weights from MVO over time.

exhibits even greater variability in the allocation of stocks, both long- and short-term, throughout the period. The inclusion of short positions introduces additional complexity, as reflected in the more erratic weight adjustments. Stocks like TSLA and NVDA show substantial short positions at various times, indicating periods where these stocks were expected to go down in price.

The subplot 5.7c details the weights of the long-short portfolio with a 50% maximum short weight constraint. Similarly to the 25% short constraint portfolio, this strategy shows significant fluctuations in weights. However, the higher short constraint allows for more aggressive short positions. Stocks such as AAPL, TSLA, and NVDA frequently appear with higher short weights, reflecting strategic bets against these stocks during specific periods. This increased flexibility in shorting could potentially improve returns during bearish market conditions but also introduces higher risk.

5.2.3 Sharpe Ratios

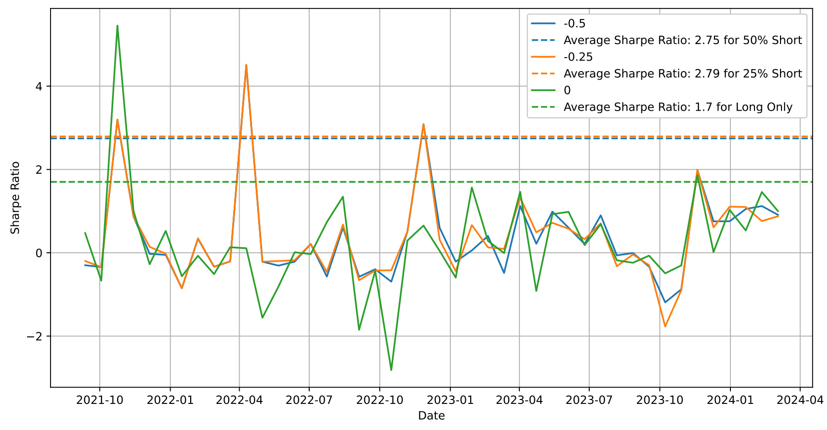


Figure 5.8: Sharpe ratio of MVO portfolios with Seq2Seq predictions.

The Sharpe ratios for various optimization techniques during the specified period are presented in Figure 5.8. This figure provides insight into the risk-adjusted returns of different portfolio strategies using MVO with predictions Seq2Seq.

The graph shows the Sharpe ratios for three portfolio strategies: 50% Short, 25% Short, and Long Only, along with their average Sharpe ratios over the analysis period.

The 50% short strategy (blue line) demonstrates a high average Sharpe ratio of 2.75. This indicates that the portfolio has achieved substantial risk-adjusted returns, particularly noticeable during the periods where the Sharpe ratio spikes above 4, such as in early 2022 and mid-2022. However, there are also periods, particularly in mid-2023, where the Sharpe ratio dips below zero, indicating negative returns relative to the risk taken.

The 25% short strategy (orange line) has the highest average Sharpe ratio of 2.79 among the strategies. This strategy maintains consistently high Sharpe ratios throughout the period, with multiple peaks above 3, highlighting its effectiveness in optimizing returns relative to risk. Similarly to the 50% Short strategy, it also experiences fluctuations, although the dips are generally less severe.

The Long Only strategy (green line) exhibits a lower average Sharpe ratio of 1.7, indicating lower risk-adjusted returns compared to the short strategies. The Long Only strategy has periods of strong performance, particularly evident in late 2021 and early 2024, where the Sharpe ratio exceeds 5 and 2 respectively. However, it also shows more frequent and deeper dips into negative values, suggesting higher volatility and less consistent performance in terms of risk-adjusted returns.

Chapter 6

Discussion

6.1 Quality of Synthetic Data and Temporal Dependencies

The analyses PCA and t-SNE provide insight into the synthetic data generated by the TimeGAN model, particularly in how it learns and replicates temporal dependencies. For Microsoft, the PCA results show that the original data are tightly clustered, whereas the synthetic data are more dispersed. This dispersion indicates that while the TimeGAN generator captures some aspects of the original data, it also introduces variability. The t-SNE analysis further emphasizes this by showing distinct clusters for the original and synthetic data, although there is some overlap. This suggests that the generator retains some structural characteristics of the original data while generating novel patterns that reflect the temporal dependencies learned during training.

For Netflix, a similar pattern is observed in the PCA results, with the original data being more tightly clustered compared to the synthetic data. The t-SNE analysis for Netflix reveals a more complex structure in the synthetic data, indicating that the TimeGAN generator captures a higher degree of the complexity and temporal dynamics of the original data. The increased overlap in the t-SNE graph for Netflix suggests a better mimicry of the structural nuances and temporal dependencies of the original data compared to the Microsoft dataset.

This highlights that the TimeGAN generator can produce synthetic data that partially reflect the structure and temporal dependencies of the original data but also introduce significant variability. This variability can be advantageous for certain applications, such as stress testing models or generating diverse scenarios for analysis, where capturing the dynamic temporal relationships in data is crucial.

6.2 Forecast Performance and Stability

The forecasts generated by the final Seq2Seq model are assessed using the IC analysis across different time horizons. Figures 5.3, 5.4, and 5.5 provide comprehensive information on the performance of the model and its predictive stability over time.

Figure 5.3 presents the 5-week rolling forecasts and the actual closing prices for Microsoft, Netflix and Nvidia. The model forecasts closely follow the actual price movements, particularly during significant trends, with minor deviations during periods of rapid price changes. This demonstrates the model's strong predictive capabilities over short- to medium-term horizons, capturing both upward and downward trends effectively.

Table 5.1 summarizes the IC analysis for three different evaluation horizons: 1W, 5W, and 10W. For the 1W horizon, the mean IC is slightly negative, indicating poor short-term predictive power. The IC standard deviation and risk-adjusted IC also suggest high variability and inconsistency in short-term forecasts. In contrast, for the 5W horizon, the mean IC improves to a positive value, and the risk-adjusted IC indicates better predictive consistency, although it remains modest. The 10W horizon shows the highest mean IC, which is statistically significant, reflecting stronger and more reliable forecasting power over longer periods.

Figure 5.4 illustrates the time series of IC for the different prediction horizons. The 1W period (subplot 5.4a) exhibits significant fluctuations and frequent crossings of the zero line, indicating inconsistent forecasting power. The 5W period (subplot 5.4b) shows smoother fluctuations and more stable forecasting performance, with periods of consistently positive IC, particularly in the latter part of the analysis. The 10W period (subplot 5.4c) demonstrates the most stable and reliable forecasting power, with fewer extreme fluctuations and extended positive periods IC.

The distribution of the IC values, shown in Figure 5.5, further supports these observations. The 1W IC distribution (subplot 5.5a) is broad and negatively skewed, reflecting high variability and frequent negative predictive power. The 5W IC distribution (subplot 5.5b) is more symmetrical and balanced, indicating more consistent forecasts. The 10W IC distribution (subplot 5.5c) is the narrowest and most symmetric, similar to a normal distribution, suggesting reliable and robust predictive power over longer periods.

These findings indicate that, while the model struggles with short-term forecasts, its performance improves significantly over longer horizons. The improved consistency and reliability of the forecasting power over medium- to long-term periods suggest that the model is more suitable for longer-term financial forecasting and investment decision-making. The integration of TimeGAN and Seq2Seq forecasting models with MVO enhances the model's ability to capture temporal dependencies, leading to more accurate and stable forecasts over extended periods.

6.3 Portfolio Optimization

The optimized portfolio aims to maximize the expected Sharpe Ratio while managing the associated volatility. The model's rolling forecasts (5W) are used in the optimization process, serving as the covariance matrix used for predicting the portfolio weights for the upcoming (3W). The expected return for the optimized portfolio is calculated based on the predicted asset return, with higher weights assigned to assets with stronger predicted performance.

Given that the covariance matrix used for MVO is relatively small, multiple problems can arise. For example, given the small amount of assets in our model portfolio, optimization might fail to diversify the weights between assets, evident in our long only strategy (see Figure 5.7). Moreover, a small covariance matrix can cause instability in optimization during high volatility periods, evident in Figure 5.3, resulting in diminished returns and a lower Sharpe Ratio (see Figures 5.8 and 5.6).

The optimized portfolio with 50% and 25% maximum short weight shows an improved Sharpe ratio compared to the baseline (see Figure 5.8), reflecting a better compensation for the risk taken. This shows that the final Seq2Seq model successfully predicts downturns and upturns in stock price, allowing the MVO to effectively short stocks to the same accuracy as the long-only strategy, giving us an almost twice as high cumulative return (see Figure 5.6). Moreover, this statement is reinforced by Figure 5.3 where it is evident that the forecasts generated successfully capture upturns and downturns in the stock price, which can also be seen by the greater than zero IC for our forecast period of 5Ws (see Table 5.1).

6.3.1 Implications for Investors

The findings of the portfolio optimization process have significant implications for investors. The improved expected return and Sharpe ratio suggest that incorporating the forecasts of the Seq2Seq model into the portfolio construction process can improve investment outcomes. Investors can achieve better risk-adjusted returns by focusing on medium- to long-term horizons, where the model is able to capture the temporal dynamics more effectively.

However, high volatility and inconsistent short-term performance highlight the need for caution when incorporating the model into short-term trading strategies (see Table 5.1), where temporal dynamics over time are undermined. Investors should consider the strengths and limitations of the model, utilizing its forecasts primarily for longer-term investment decisions.

Chapter 7

Conclusions

7.1 Conclusion

This thesis aimed to explore whether the integration of TimeGAN and Seq2Seq forecasting models with MVO improves market trend predictions and investment decision-making in portfolio management, compared to traditional benchmarks. The research used a novel end-to-end framework designed to take advantage of the strengths of TimeGAN to capture temporal dependencies and Seq2Seq models in forecasting, combined with MVO for optimized portfolio management.

The analyses of PCA and t-SNE revealed that the synthetic data generated by the TimeGAN model capture some structural characteristics of the original data while introducing variability. This variability, while potentially advantageous for applications like stress testing, underscores the model's limitations in perfectly mimicking original data.

The IC analysis demonstrated the predictive power of the model on different time horizons. It was found that while the model's short-term predictive power is limited, its performance improves significantly over medium- to long-term horizons. This suggests that the integrated framework is better suited for longer-term financial forecasting.

The results of the portfolio optimization highlighted the practical implications for investors. The optimized portfolio, particularly with short-weight constraints, showed improved Sharpe ratios and expected returns, indicating that the model can effectively capture market downturns and upturns. This underscores the model's potential in enhancing investment outcomes over longer horizons, although caution is needed for short-term strategies due to inherent volatility.

Based on the results, the integration of TimeGAN for learning temporal dependencies and Seq2Seq forecasting models with MVO improves the accuracy of forecasts of market trend and improves automatic investment decision-making in portfolio management compared to traditional benchmarks. The combined approach offers superior performance in terms of risk adjusted returns, particularly over medium to long-term investment horizons. This integration enables the capture of complex temporal dependencies in financial data, resulting in more accurate and adaptive investment strategies.

In conclusion, the integrated framework of the TimeGAN and the forecasting model Seq2Seq with MVO presents a robust and innovative approach to portfolio management. Bridges the gap between theoretical advancements and practical applications in financial investment, providing investors with a more accurate tool for predicting market trends and making informed investment decisions.

7.2 Limitations

Several limitations were encountered during the course of this degree project, which impacted the overall outcomes and interpretations of the results.

Firstly, the variability introduced by the TimeGAN generator posed a significant challenge. Although the synthetic data captured some structural aspects of the original datasets, the broader spread and dispersion indicated that the generated data did not perfectly mimic the density and clustering of the original data. This discrepancy highlights a key limitation in the fidelity of the synthetic data produced by the current model configuration.

The IC analysis revealed limitations in predictive power, particularly over shorter time horizons. The results of the 1W horizon showed a slightly negative predictive power with high variability, suggesting that the model struggles with the accuracy of the short-term prediction. This limitation is critical for applications requiring precise short-term forecasts, as the model's inconsistency could lead to unreliable predictions.

Another limitation was the scope of the datasets used in the analysis. The focus on specific tickers, such as Microsoft and Netflix, provided valuable insight but may not generalize to other financial instruments or markets. The limited dataset range restricts the breadth of conclusions that can be drawn about the model performance across different sectors or market conditions.

Computational constraints also played a role in limiting the depth of the analysis. Training complex generative models such as TimeGAN requires significant computational resources and time, which restricts the ability to perform extensive hyperparameter tuning and more exhaustive evaluations. These limitations potentially affected the optimization of the model and the robustness of the generated synthetic data.

Furthermore, the evaluation metrics used, while comprehensive, are not exhaustive. The reliance on PCA, t-SNE, and IC analyses, although informative, may not capture all aspects of the model's performance or the nuances in synthetic data quality. Other metrics and evaluation techniques could provide additional insight, but were beyond the scope of this project due to resource and time constraints.

7.3 Future Work and Unaddressed Issues

Based on the findings and limitations of this degree project, several avenues for future work can be identified to further enhance the capabilities and applications of generative models, such as TimeGAN, in financial data analysis. One critical area for future work is the refinement of the synthetic data generation process to reduce variability and improve fidelity.

This could involve experimenting with advanced techniques for model tuning, such as hyperparameter optimization and ensemble methods, to achieve a closer alignment between synthetic and original data distributions. In addition, integrating techniques such as adversarial training could help mitigate the introduction of artifacts and improve the structural accuracy of synthetic data.

Expanding the scope of datasets used in the analysis is another vital step. Future research should consider incorporating a wider range of financial instruments, including stocks from different sectors, commodities, and forex data. This expansion would allow for a more comprehensive evaluation of the performance of the model in diverse market conditions and increase the generalizability of the findings. Further exploration of alternative evaluation metrics is also recommended. Although the analyzes of PCA, t-SNE, and IC provided valuable information, other metrics such as dynamic time warping (DTW) and domain-specific financial metrics could offer additional perspectives on the quality and applicability of synthetic data. Developing a more holistic evaluation framework would allow for a deeper understanding of the strengths and weaknesses of the model.

Addressing computational constraints is another area for future work. Using advances in high-performance computing and cloud-based solutions could enable more extensive training and evaluation processes. This would facilitate a broader exploration of model configurations and potentially uncover more effective generative strategies. Furthermore, future work could explore the integration of TimeGAN with other machine learning models to create hybrid systems that take advantage of the strengths of multiple approaches. For example, combining TimeGAN with reinforcement learning or Bayesian networks could improve predictive accuracy and provide more robust decision-making tools in financial contexts.

For researchers continuing this work, it is recommended to focus on a rigorous evaluation and validation of the results. Documenting the impacts of different model configurations and training processes will be crucial to building upon this foundation. Collaborative efforts that bring together experts from financial engineering, machine learning, and data science could also accelerate advancements and foster innovative solutions.

Due to the project's narrow timeframe, the portfolio optimization aspect of the thesis was quite limited. The drawbacks of the MVO algorithm became evident, even with the high returns and Sharpe ratio. More advanced optimization algorithms would limit portfolio concentration while retaining the Sharpe ratio and returns. Additionally, computational limitations prevented the training of the model for many stocks. Ideally, we would have liked to reconstruct the MSCI US index with our models and compare the performance of our models versus the actual index to evaluate the performance of neural network models in financial applications. Large pension funds, e.g. AP4, strategize their investments to take a certain minimum risk level to achieve sufficient expected long-term returns while maintaining low costs. They achieve this by creating their reference portfolios against global equity indices and government bonds. If we can reconstruct the reference portfolios using our models and produce better returns while maintaining the same risk level, we can justify the reason for the model's existence.

Furthermore, from the results, we are unable to conclude if the performance of our model is high because the model itself is performing well, or if the selected time period and stocks allow for easy training of neural network models. We would have liked to benchmark our model versus other neural models to evaluate its performance.

7.4 Cost Analysis

Training the models for the entire MSCI US index of 612 tickers with cloud computing, at the time of writing this thesis, using an ***a2-highgpu-2g*** instance based in the western region of Europe would cost roughly $\$2{,}326.64$ assuming that each model requires 10 hours of training, and we can load 12 models onto each GPU. While this was not an option for us as master's students, for a large organization it is very doable.

7.5 Reflections

7.5.1 Economic, Social, Environmental, and Ethical Aspects of the Work

The economic implications of this work are significant, especially in the context of financial modeling and investment strategies. By evaluating models such as TimeGAN and Seq2Seq, this project provides information on their potential to generate synthetic time series data and improve predictive accuracy. The analysis demonstrated that longer prediction horizons (e.g., the 10W horizon) offer more robust and reliable predictive power, leading to more informed investment decisions and potentially higher returns. Furthermore, synthetic data generation can reduce the need for extensive historical data, thus lowering the costs associated with data acquisition and storage. However, training complex generative models like TimeGAN requires significant computational resources, which can be expensive and potentially limit accessibility for smaller organizations. Furthermore, the variability in synthetic data introduces uncertainty, impacting the reliability of financial models and leading to economic risks if predictions are inaccurate.

The social aspects of this work are primarily related to its impact on the financial industry and stakeholders. The use of generative models for synthetic data generation can make advanced financial modeling tools more accessible to a wider range of users, including smaller firms and individual investors. Advancements in AI and machine learning in finance can create new job opportunities in data science, financial engineering, and related fields. However, there are ethical considerations regarding the transparency and explainability of AI models in finance. Stakeholders need to understand the limitations and potential biases of the models to ensure fair and responsible use.

The environmental impact of this work is primarily related to the computational resources required. Training large-scale generative models consumes significant amounts of energy, contributing to the carbon footprint of computational research. Efforts to optimize models and improve computational efficiency can help mitigate these environmental impacts. Exploring cloud-based solutions and high-performance computing that take advantage of renewable energy sources can reduce the environmental footprint associated with training AI models.

Ethical considerations are critical in the application of AI in finance. Ensuring the privacy and security of financial data used in training models is paramount. Synthetic data can help mitigate privacy concerns, but models must be designed to prevent the leakage of sensitive information. Variability in synthetic data generation can introduce biases that can affect financial predictions and decisions. Rigorous evaluation and continuous refinement of the models are necessary to address and mitigate these biases. Financial models must be transparent and accountable. Stakeholders should be able to understand the decision-making process of AI models to ensure that they are used responsibly and ethically.

References

[1] H. Markowitz, “Portfolio Selection,” *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952. doi: 10.2307/2975974 Publisher: [American Finance Association, Wiley]. [Online]. Available: https://www.jstor.org/stable/2975974 [Pages 1 and 21.]

[2] F. Black and R. Litterman, “Global Portfolio Optimization,” *Financial Analysts Journal*, vol. 48, no. 5, pp. 28–43, 1992, publisher: CFA Institute. [Online]. Available: https://www.jstor.org/stable/4479577 [Page 2.]

[3] M. O’Hara, “High frequency market microstructure,” *Journal of Financial Economics*, vol. 116, no. 2, pp. 257–270, May 2015. doi: 10.1016/j.jfineco.2015.01.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304405X15000045 [Page 2.]

[4] G. Mariani, Y. Zhu, J. Li, F. Scheidegger, R. Istrate, C. Bekas, and A. C. I. Malossi, “PAGAN: Portfolio Analysis with Generative Adversarial Networks,” Sep. 2019, arXiv:1909.10578 [q-fin]. [Online]. Available: http://arxiv.org/abs/1909.10578 [Page 2.]

[5] M. Haugh, “Mean-Variance Optimization and the CAPM,” Columbia University, New York, 2016. [Online]. Available: https://www.columbia.edu/~mh2078/FoundationsFE/MeanVariance-CAPM.pdf [Pages 2 and 22.]

[6] T. A. Schmitt, D. Chetalova, R. Schäfer, and T. Guhr, “Non-Stationarity in Financial Time Series and Generic Features,” *EPL (Europhysics Letters)*, vol. 103, no. 5, p. 58003, Sep. 2013. doi: 10.1209/0295-5075/103/58003

ArXiv:1304.5130 [q-fin]. [Online]. Available: http://arxiv.org/abs/1304.5130 [Page 3.]

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," Jun. 2014, arXiv:1406.2661 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1406.2661 [Pages 7 and 8.]

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980 [Page 10.]

[9] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series Generative Adversarial Networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html [Pages 10, 33, and 35.]

[10] G. Dorffner, "Neural Networks for Time Series Processing," 1996. [Online]. Available: https://www.semanticscholar.org/paper/Neural-Networks-for-Time-Series-Processing-Dorffner/b9a1c422c29b8e4dfb1a9c91d426fc3894726e88 [Page 12.]

[11] E. ZIvot, "6 The Gaussian White Noise Return Model | Introduction to Computational Finance and Financial Econometrics with R," May 2021. [Online]. Available: https://bookdown.org/compfinezbook/introFinRbook/The-Gaussian-White-Noise-Model.html [Page 13.]

[12] I. Mitliagkas, "IFT 6085 - Lecture 10 Expressivity and Universal Approximation Theorems Part 1," Montreal, 2020. [Online]. Available: https://mitliagkas.github.io/ift6085-2020/ift-6085-lecture-10-notes.pdf [Page 13.]

[13] R. M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," Nov. 2019, arXiv:1912.05911 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1912.05911 [Pages 13 and 14.]

[14] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. doi: 10.1162/neco.1997.9.8.1735 [Pages 15 and 17.]

[15] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019, arXiv:1909.09586 [cs]. [Online]. Available: http://arxiv.org/abs/1909.09586 [Page 15.]

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," Dec. 2014, arXiv:1412.3555 [cs]. [Online]. Available: http://arxiv.org/abs/1412.3555 [Page 16.]

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Oct. 2014, arXiv:1409.1259 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1409.1259 [Page 16.]

[18] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent Highway Networks," Jul. 2017, arXiv:1607.03474 [cs]. [Online]. Available: http://arxiv.org/abs/1607.03474 [Pages 17, 18, and 35.]

[19] O. Press and L. Wolf, "Using the Output Embedding to Improve Language Models," Aug. 2016. [Online]. Available: https://arxiv.org/abs/1608.05859v3 [Page 18.]

[20] T.-H. Lee, "Loss Functions in Time Series Forecasting," 2007. [Online]. Available: https://www.semanticscholar.org/paper/Loss-Functions-in-Time-Series-Forecasting-Lee/dcd34f5077aa96df43c814e436b12d2a33401e9a [Page 18.]

[21] M. Baker and J. Wurgler, "Market Timing and Capital Structure," *The Journal of Finance*, vol. 57, no. 1, pp. 1–32, 2002, publisher: [American Finance Association, Wiley]. [Online]. Available: https://www.jstor.org/stable/2697832 [Page 19.]

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," Dec. 2014, arXiv:1409.3215 [cs]. [Online]. Available: http://arxiv.org/abs/1409.3215 [Page 20.]

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 2016, arXiv:1409.0473 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1409.0473 [Page 20.]

[24] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Sep. 2023, arXiv:1910.10683 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1910.10683 [Page 20.]

[25] A. Lesniewski, "Optimization Techniques in Finance 5. Convex optimization. II," New York, 2019. [Page 23.]

[26] W. F. Sharpe, "Mutual Fund Performance," *The Journal of Business*, vol. 39, no. 1, pp. 119–138, 1966, publisher: University of Chicago Press. [Online]. Available: https://www.jstor.org/stable/2351741 [Page 24.]

[27] R. C. Grinold and R. N. Kahn, *Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk*, 2nd ed. New York: McGraw Hill, Oct. 1999, vol. 1999. ISBN 0-07-024882-6 [Page 24.]

[28] I. Morningstar, "Do You Have the Wrong Index Funds for 2024's Stock Market?" Jan. 2024, section: Markets. [Online]. Available: https://www.morningstar.com/markets/do-you-have-wrong-index-funds-2024s-stock-market [Page 30.]

[29] S&P, "Methodology Matters | S&P Dow Jones Indices," 2024. [Online]. Available: https://www.spglobal.com/spdji/en/research-insights/index-literacy/methodology-matters/ [Page 30.]

[30] MSCI, "THE INDEX MATTERS," MSCI, Tech. Rep., May 2019. [Online]. Available: https://www.msci.com/documents/10199/8e9db7fc-6abf-463b-9210-c059e2e74cd5 [Page 30.]

[31] The pandas development team, "pandas-dev/pandas: Pandas," May 2024. [Online]. Available: https://github.com/pandas-dev/pandas [Page 31.]

[32] NumPy, “numpy/numpy,” May 2024, original-date: 2010-09-13T23:02:39Z. [Online]. Available: https://github.com/numpy/numpy [Page 31.]

[33] Keras-Team, “keras-team/keras,” May 2024, original-date: 2015-03-28T00:35:42Z. [Online]. Available: https://github.com/keras-team/keras [Page 31.]

[34] MLflow, “mlflow/mlflow,” May 2024, original-date: 2018-06-05T16:05:58Z. [Online]. Available: https://github.com/mlflow/mlflow [Page 31.]

[35] S. Jansen, “stefan-jansen/alphalens-reloaded,” Jun. 2024, original-date: 2021-02-23T19:11:27Z. [Online]. Available: https://github.com/stefan-jansen/alphalens-reloaded [Pages 33 and 40.]

[36] YData, “ydata-synthetic,” Seattle, May 2024. [Online]. Available: https://github.com/ydataai/ydata-synthetic [Page 35.]

[37] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization,” *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-558.html [Page 36.]

[38] H. Delatte and C. Nicolini, “skfolio,” 2023. [Online]. Available: https://github.com/skfolio/skfolio [Page 39.]

Appendix A

PCA and t-SNE Analyses for all Tickers and Models

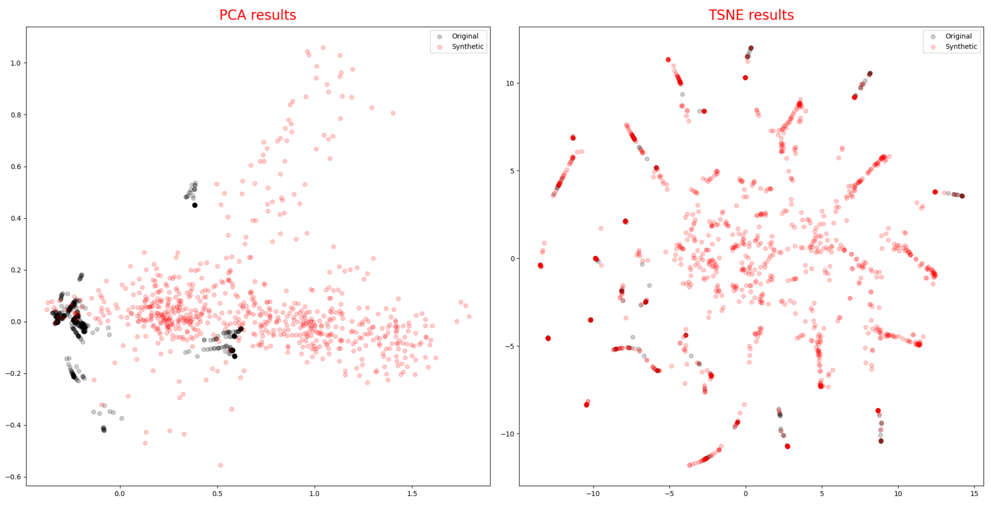


Figure A.1: PCA and t-SNE for AAPL

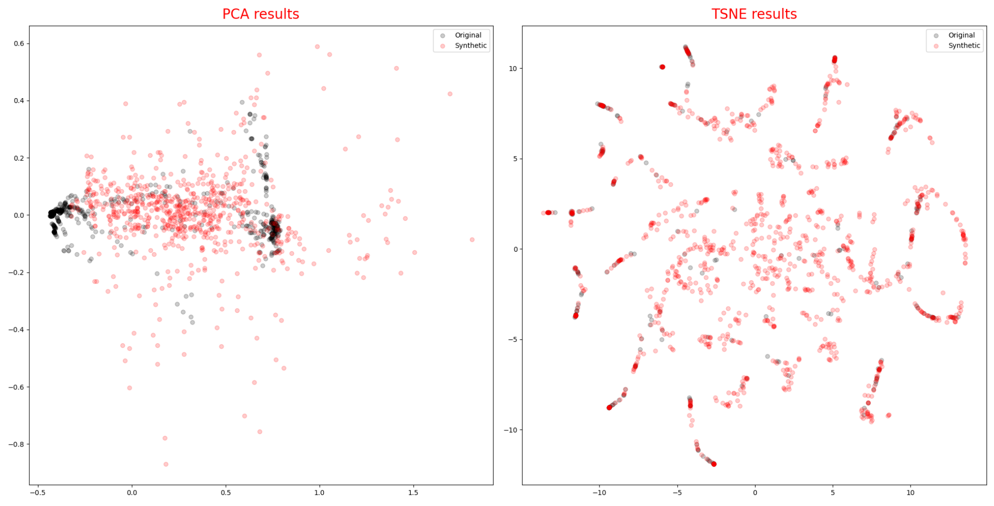


Figure A.2: PCA and t-SNE for AMD

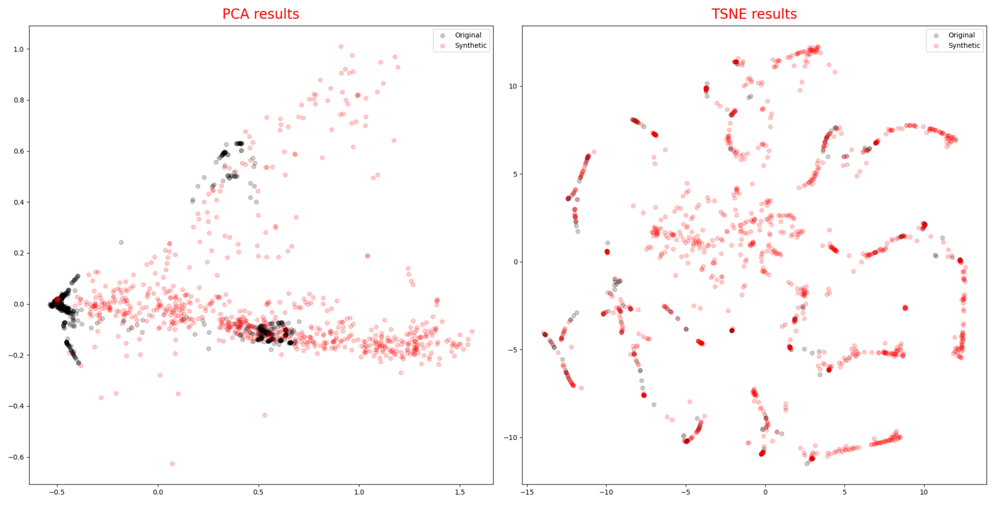


Figure A.3: PCA and t-SNE for AMZN

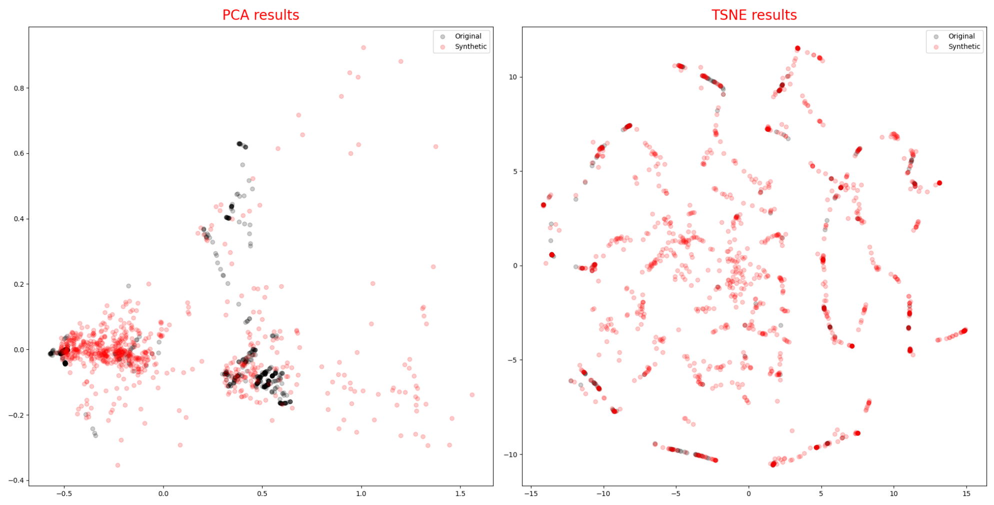


Figure A.4: PCA and t-SNE for AVGO

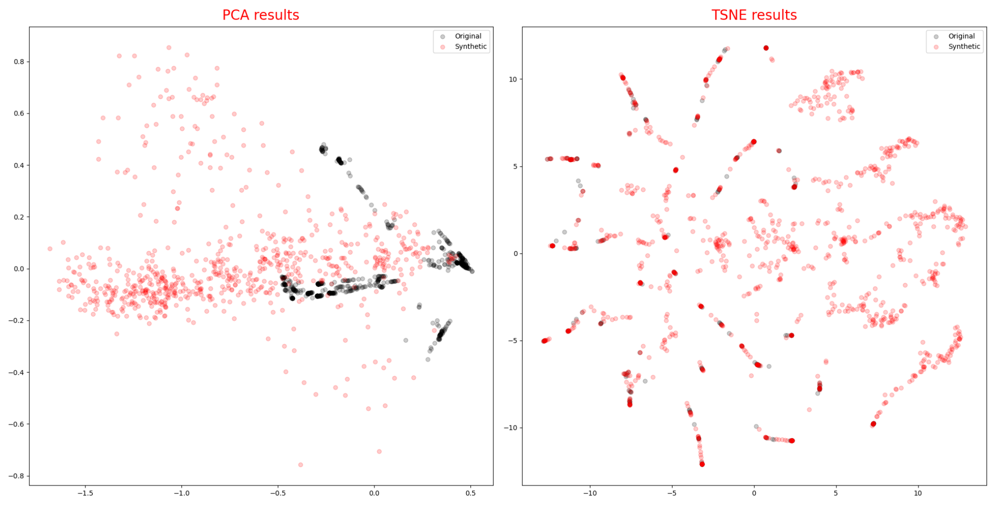


Figure A.5: PCA and t-SNE for GOOGL

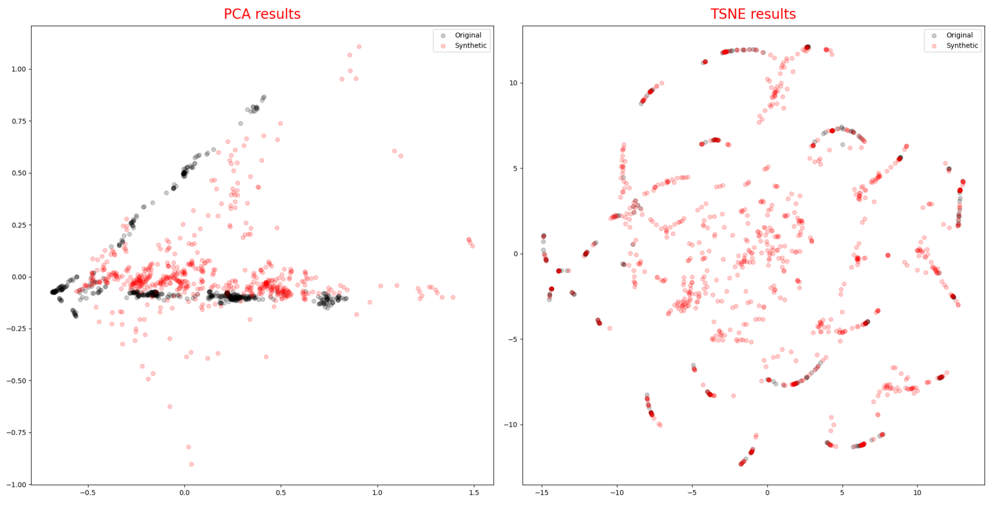


Figure A.6: PCA and t-SNE for META

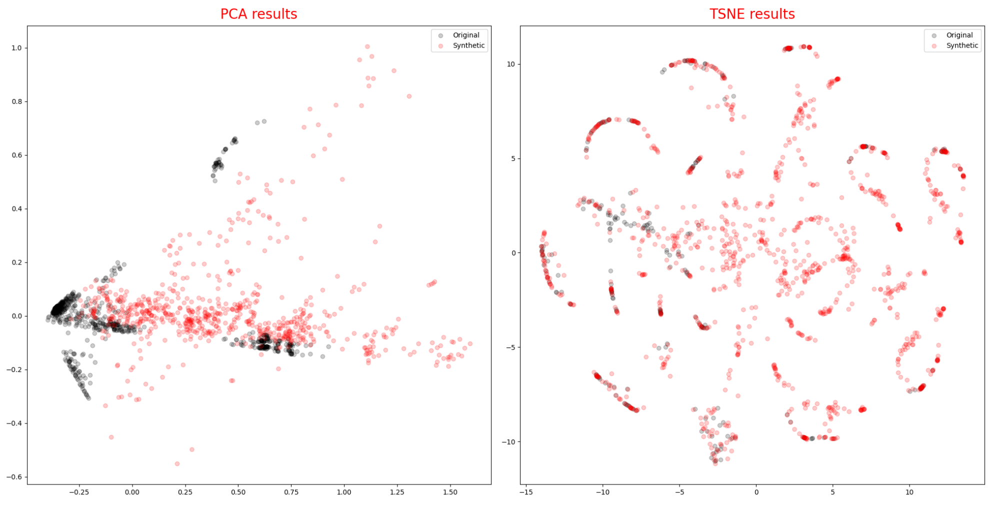


Figure A.7: PCA and t-SNE for MSFT

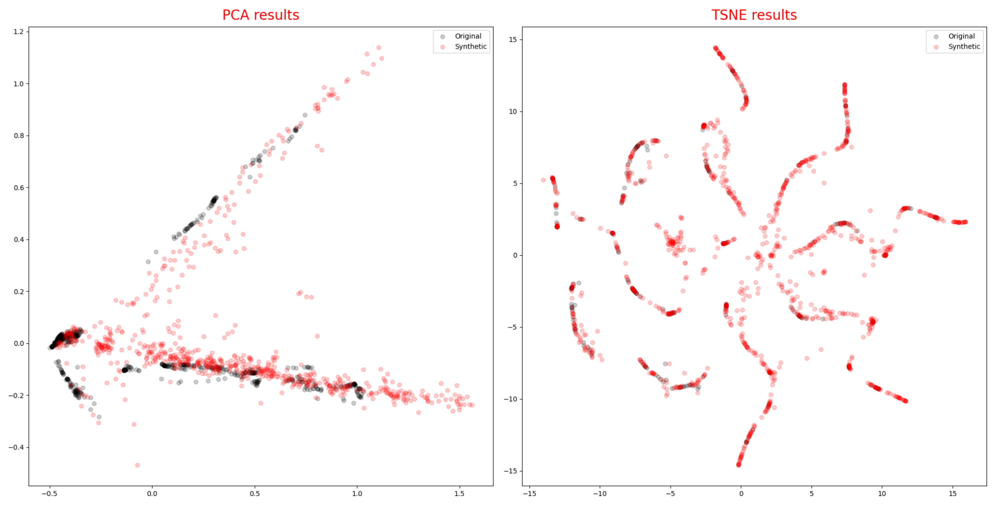


Figure A.8: PCA and t-SNE for NFLX

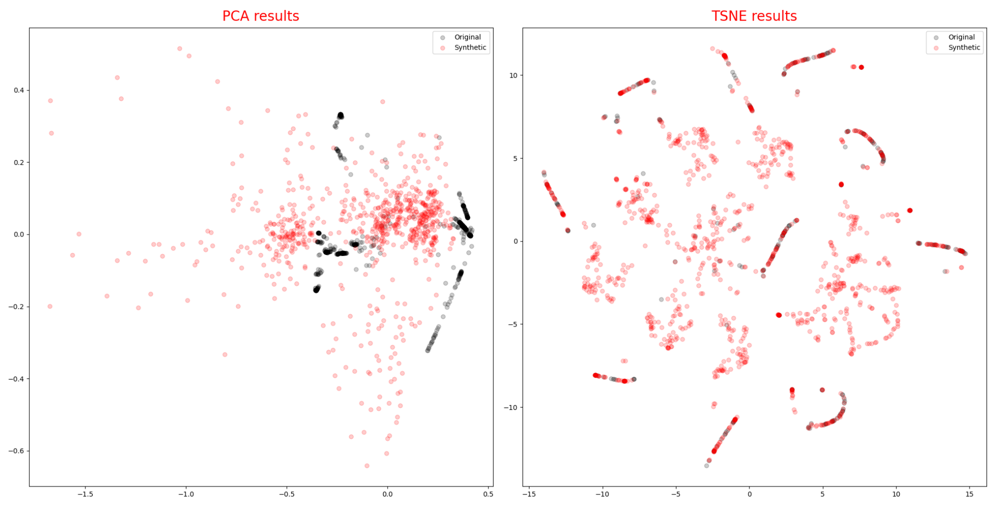


Figure A.9: PCA and t-SNE for NVDA

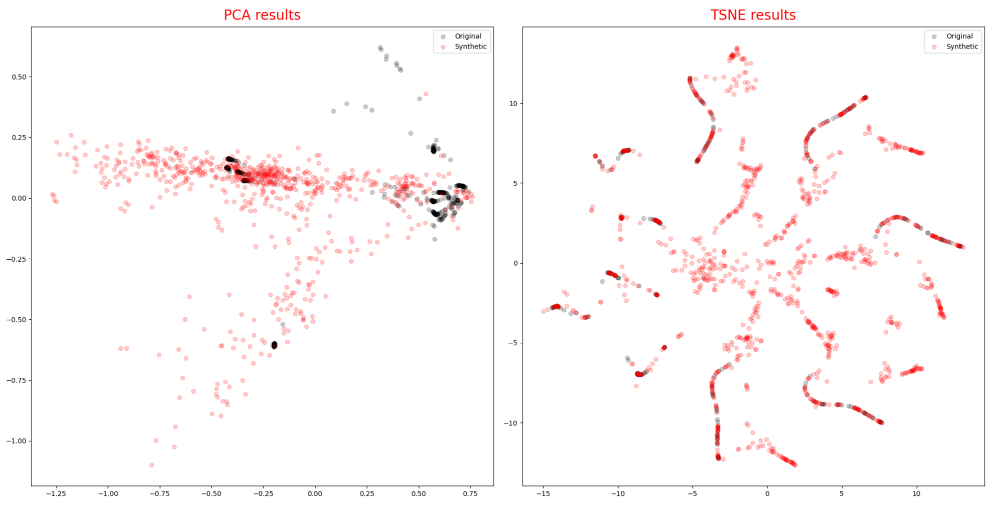


Figure A.10: PCA and t-SNE for PEP

Appendix B

Predicted Closing Prices for all Tickers and Models

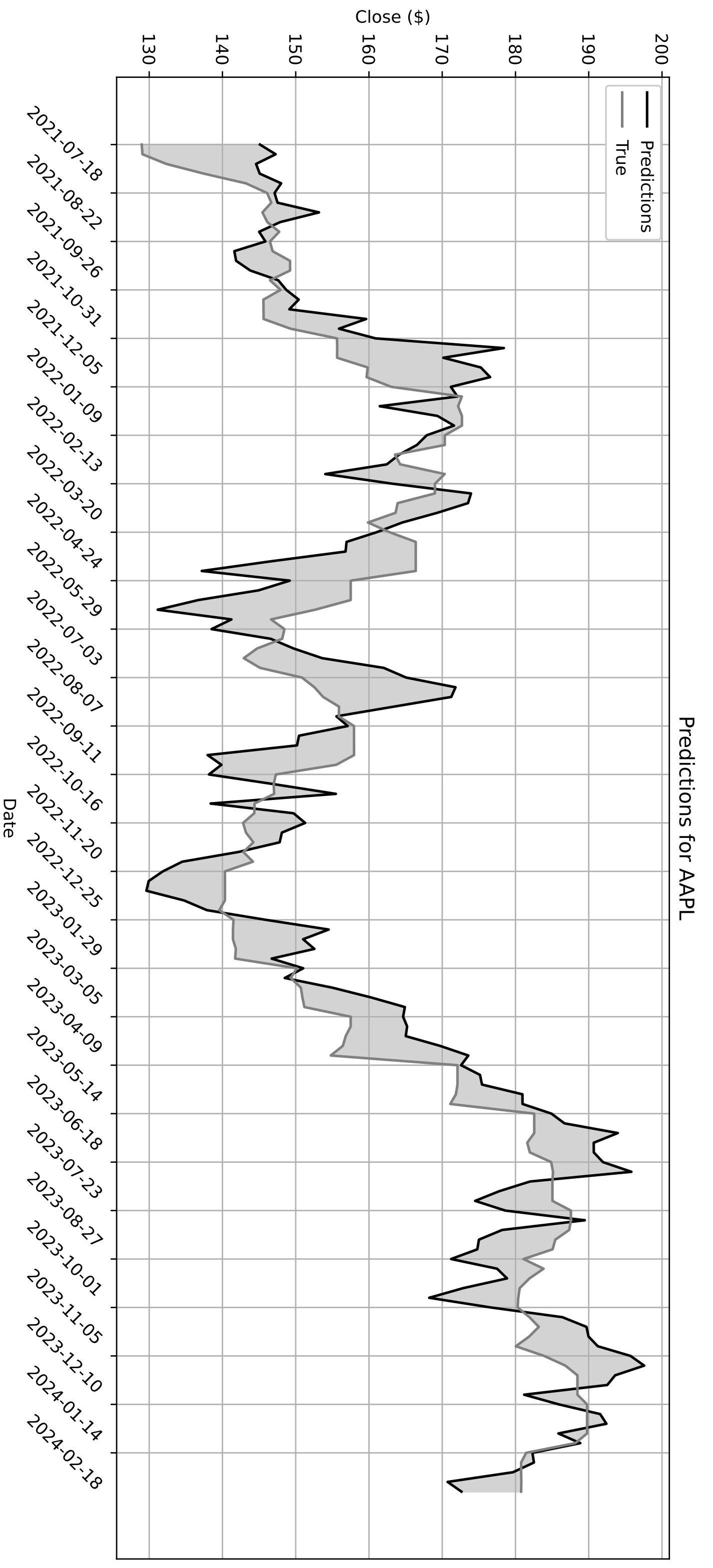


Figure B.1: AAPL

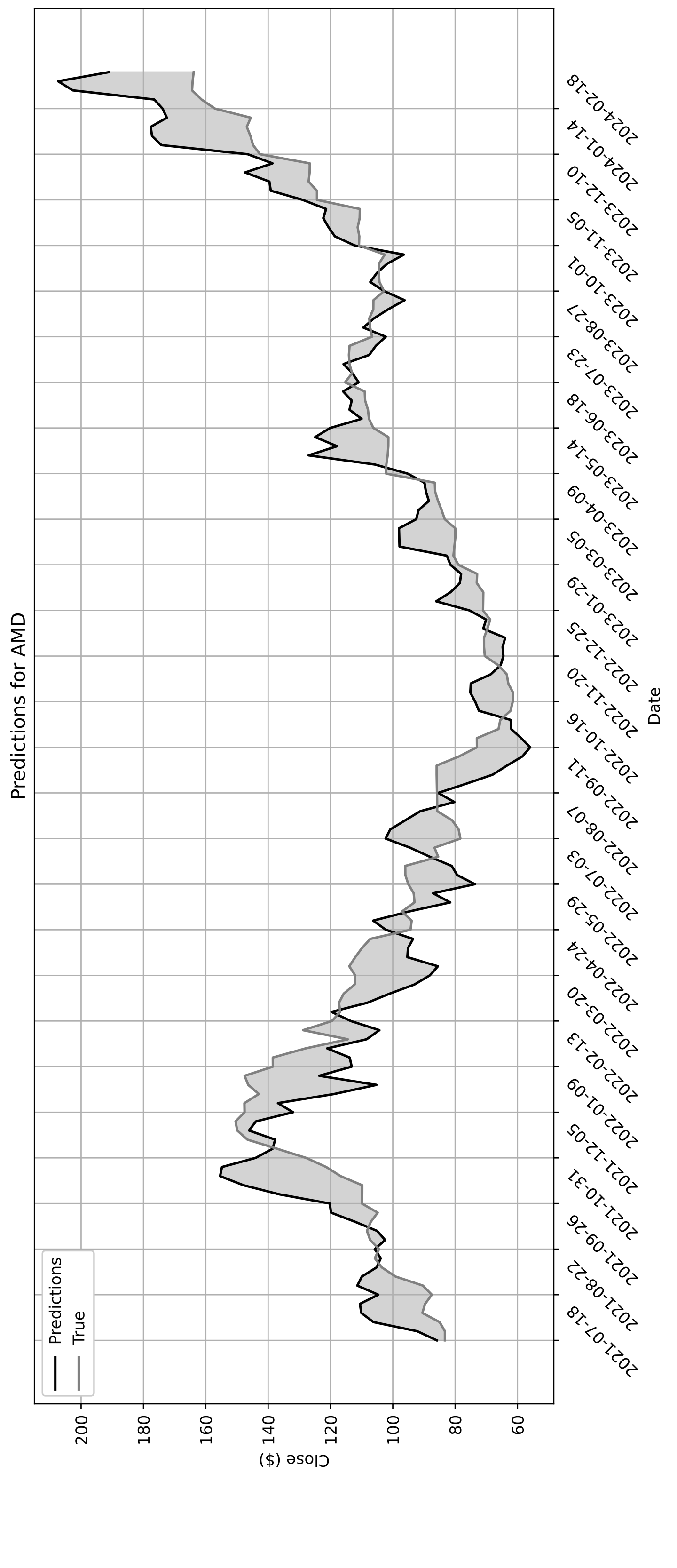


Figure B.2: AMD

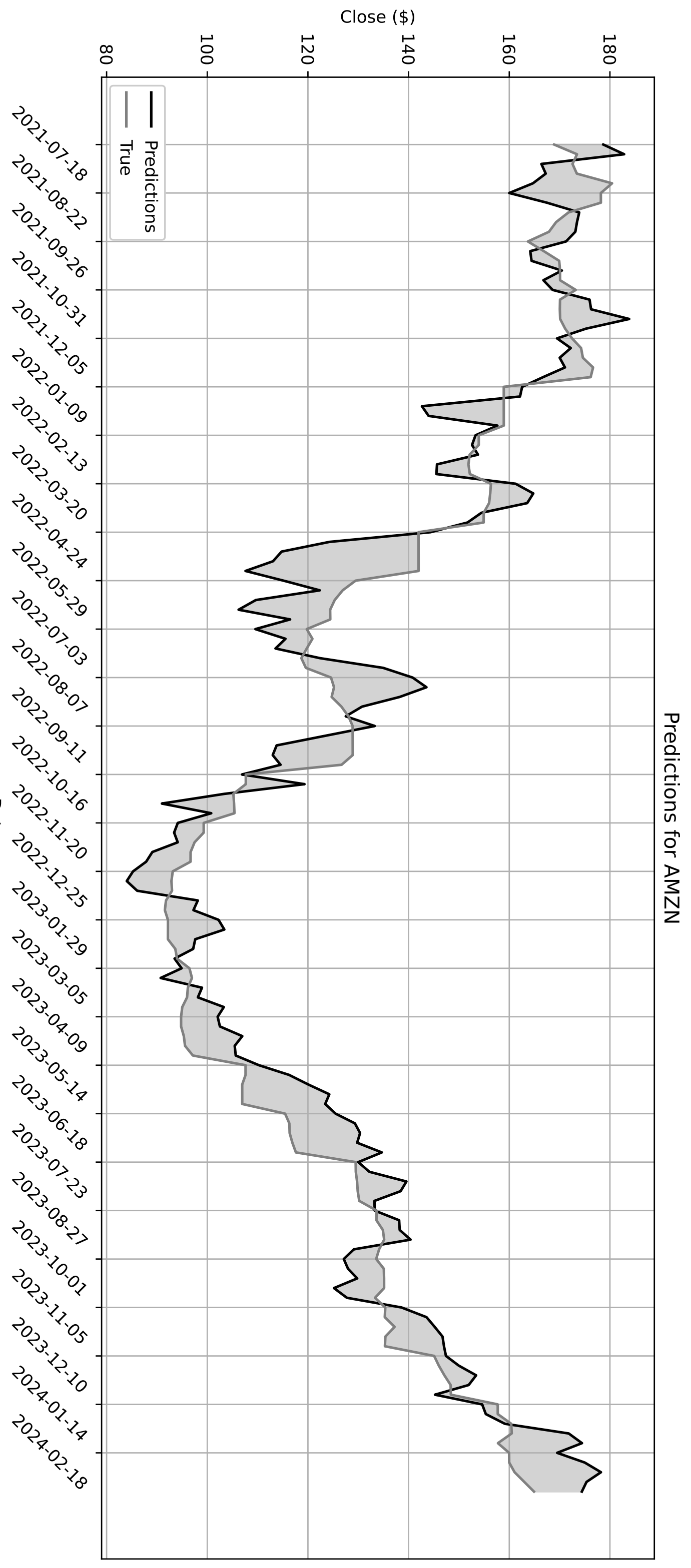


Figure B.3: AMZN

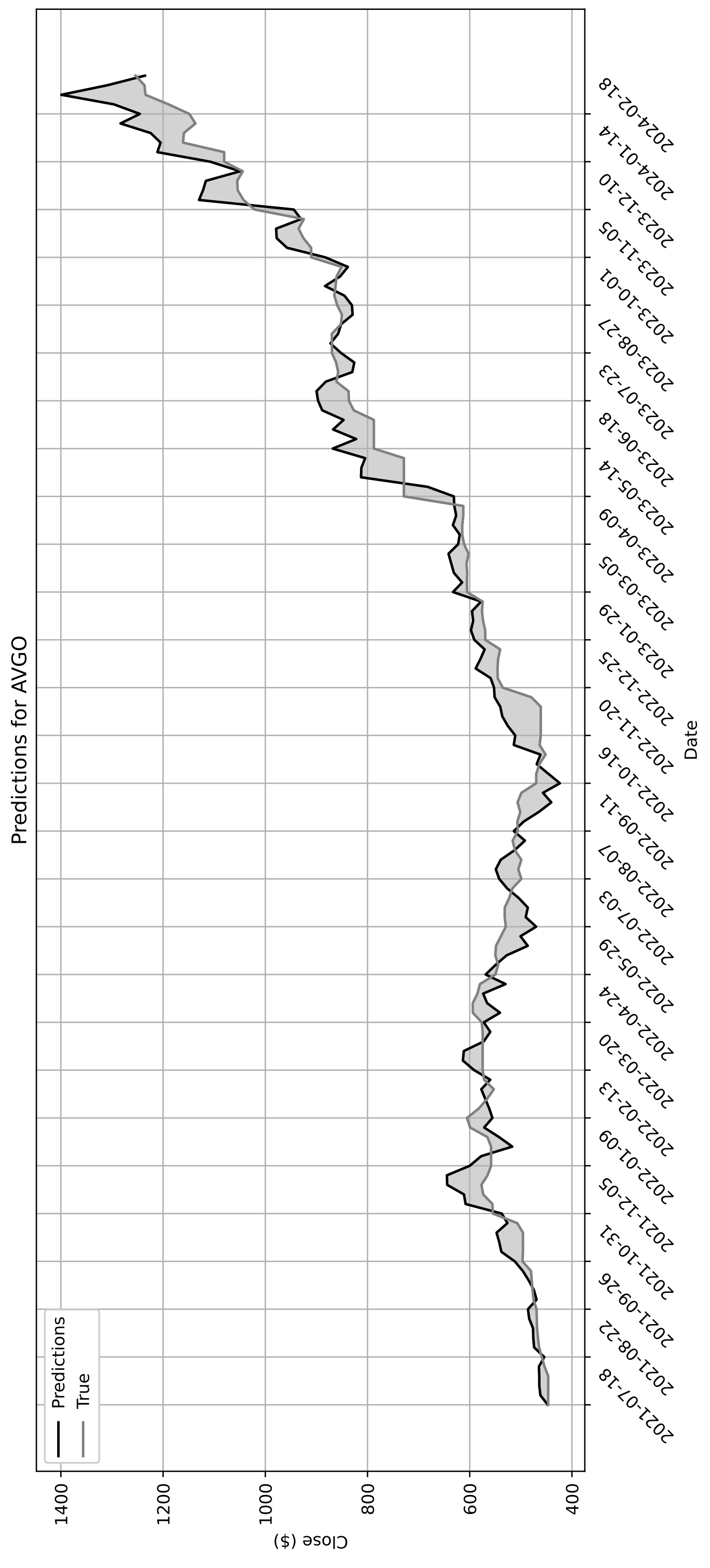


Figure B.4: AVGO

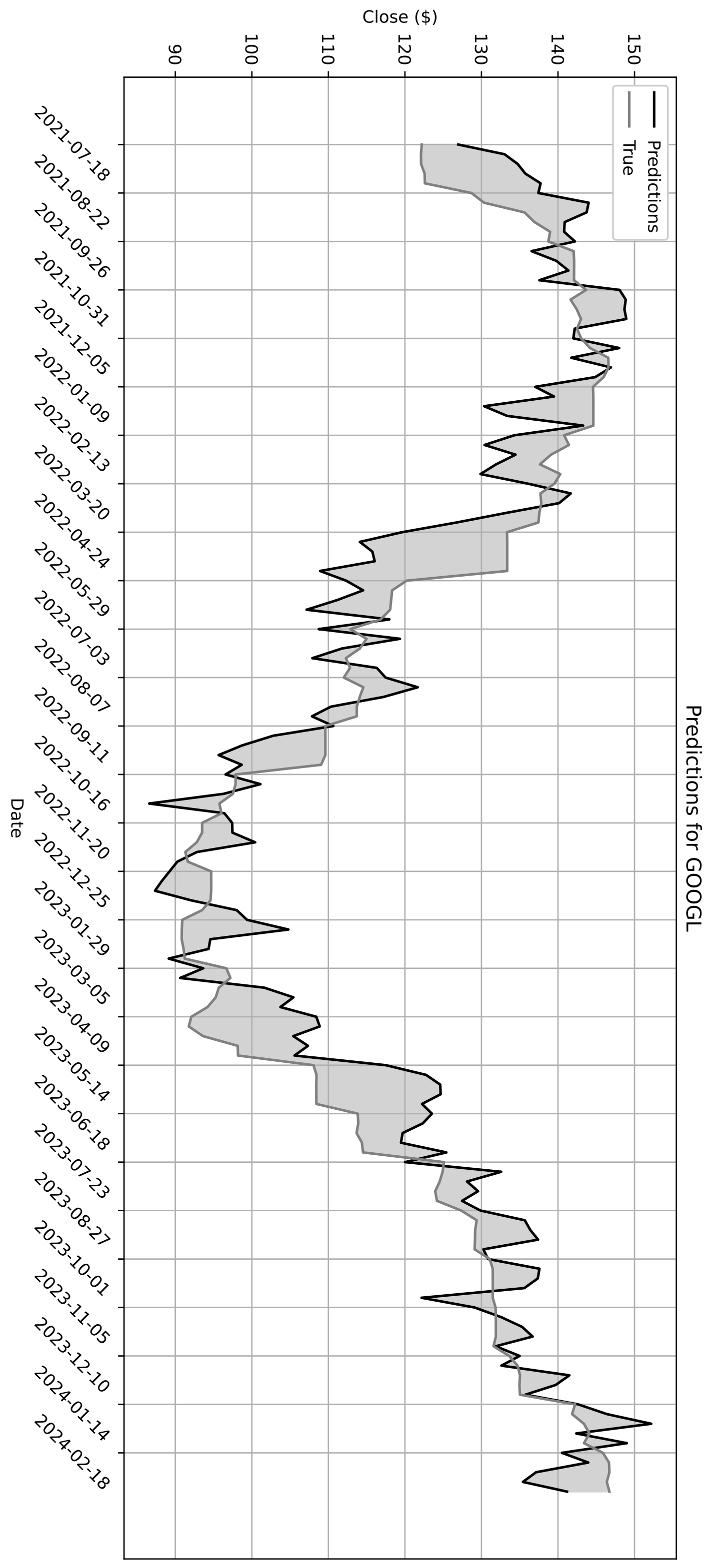


Figure B.5: GOOGL

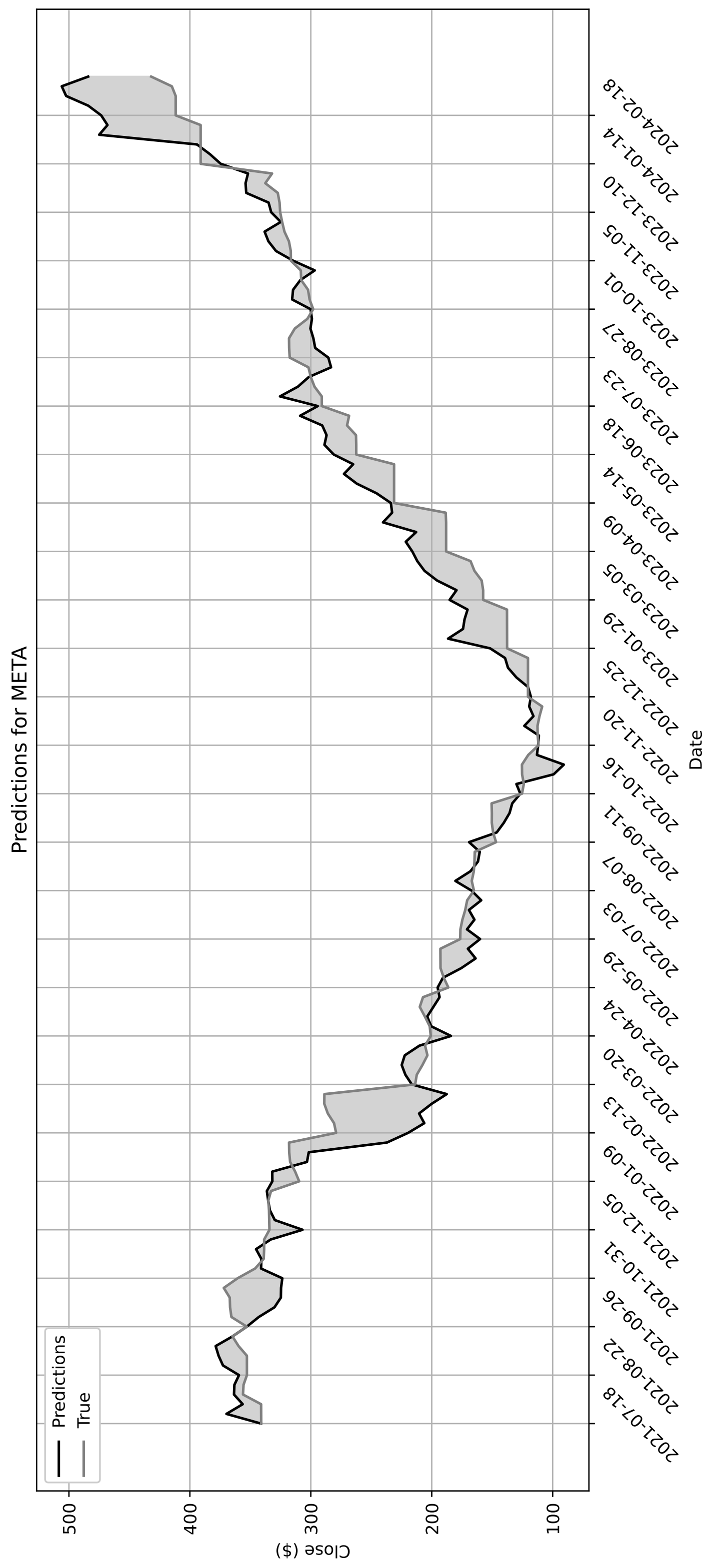


Figure B.6: META

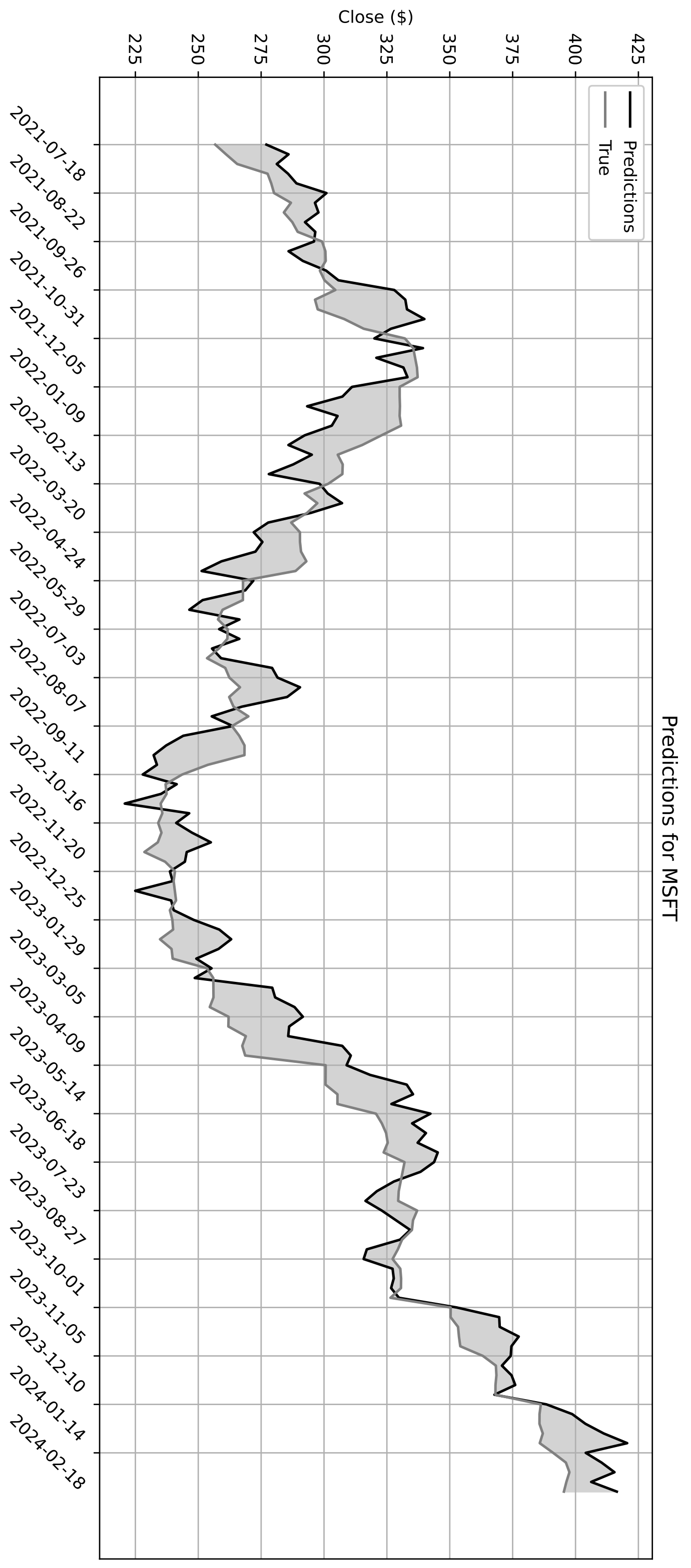


Figure B.7: MSFT

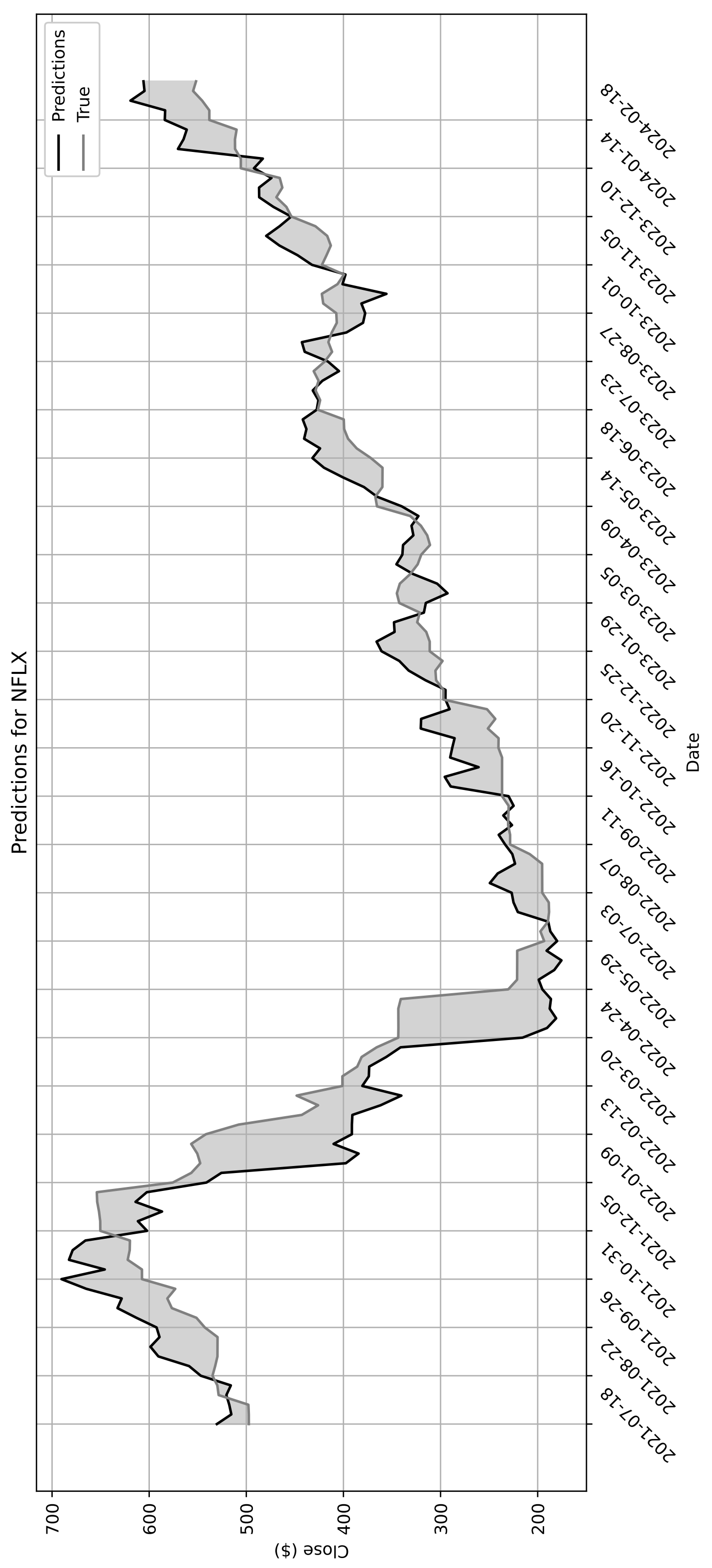


Figure B.8: NFLX

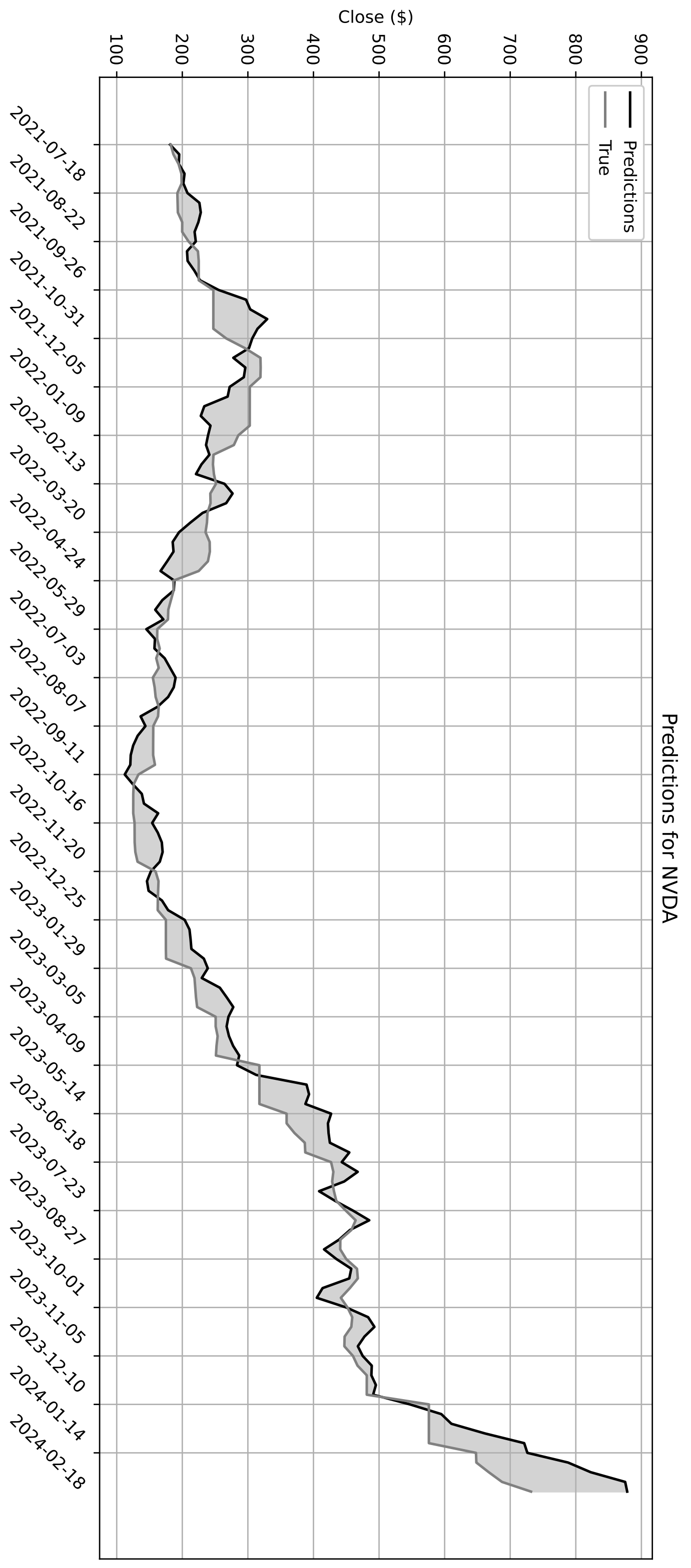


Figure B.9: NVDA

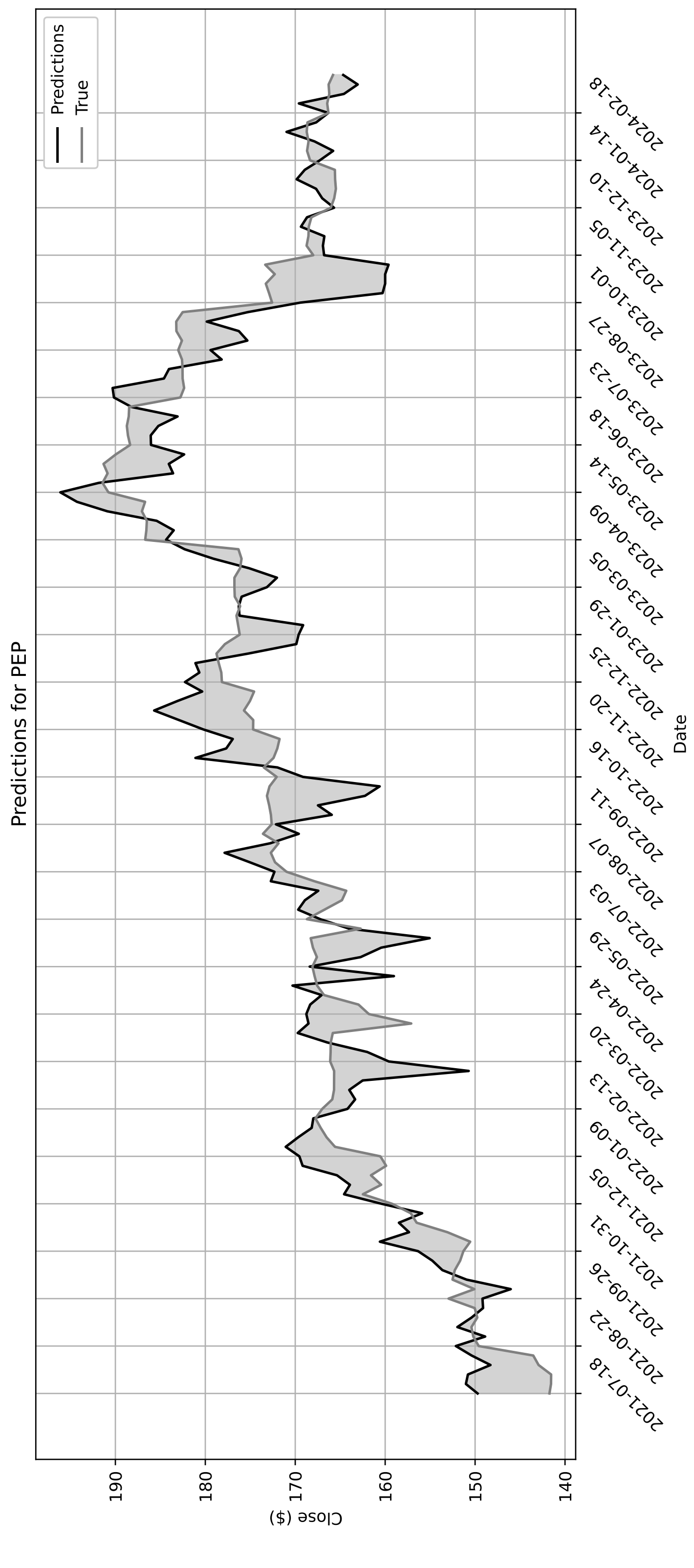


Figure B.10: PEP

TRITA-EECS-EX-2024:759