BSc Data Science and Artificial Intelligence

Application of Transformer Models and Mixture-of-Experts Architectures on Pairs Trading with ETFs

David Moerdijk

Supervisors:
Marc Hilbert & Mitra Baratchi

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

July 1, 2025

Abstract

Pairs Trading (PT) is a widely used quantitative trading strategy that exploits the price relationships between asset pairs on the stock market. Recent advancements in deep learning have opened up possible enhancements through more accurate time series forecasting. This study contributes to the field of machine learning by applying proven models to a novel context. Experiments forecast time series data using the classical statistical Kalman Filter Regression as a baseline, and benchmarked this against a Transformer model and the model Time-MoE. Time series consisted of price spreads between pairs of two statistically cointegrated ETFs, found through the Engle-Granger test. The results demonstrate that the benchmarked deep learning models are able to capture temporal dependencies with more predictive accuracy than the baseline Kalman Filter Regression across all tested time periods. Time-MoE consistently yielded the most accurate predictions. The Transformer model exhibited higher variance and less robustness across time periods. This superior prediction accuracy did not propagate through to returns in trading simulations, as the Kalman Filter was able to more accurately capture z-score breaches, triggering trading signals correctly in the final simulation more often caused by specifics in the types of forecast errors each model makes. Based on current findings, pre-trained and sparse architectures hold the greatest promise for practical deployment in machine learning in financial contexts under alternative strategies.

Contents

1. Introduction

Machine-learning techniques have long been prevalent across different sectors, including finance. In this sector, statistical arbitrage is a rigorously studied econometric framework that exploits statistically significant price divergences between related assets or markets [HH12]. As an applied case, pairs trading (PT) is one way to translate the statistical-arbitrage concept into executable long-short positions.

> ”It is hard to overestimate the influence that pairs trading has had on the industry.”
> — Hudson and Thames [Tha25]

Among trading strategies used in algorithmic trading by propietary firms, **pairs trading** (PT) stands out for providing an approach to apply time series predictions, based on assumptions of mean-reversion rather than the alternative of using direct price predictions to inform trading signals. Moreover, the method is well-documented in scientific literature, being heralded as the simplest [Vid04] and one of the most valuable [SH20] market-neutral strategies. Despite the surge in machine-learning advancements, most PT studies still rely on classical models or modest recurrent networks. Transformers and Mixture-of-Experts (MoE) architectures have exhibited performance gains in language, vision, and generic time-series tasks, yet their value for PT remains largely unexplored. Addressing that gap, this thesis aims to serve as exploratory research into differences between these two deep learning architectures and classical statistical methods.
Only a handful of papers benchmark *Transformers* for PT, and so far no published study has tested a sparse, pre-trained *Mixture-of-Experts* architecture such as Time-MoE in the current context. This is analyzed in further detail in section 2.4. By measuring how these state-of-the-art models perform compared to a classical Kalman Filter across five rolling out-of-sample windows, this thesis fills a clear empirical gap and eventually analyzes the trade-off between raw predictive power and economic pay-off in PT with ETFs.
The purpose of this investigation is to determine to what extent the two deep learning architectures can improve one-day-ahead ETF-spread forecasts, and how those improvements translate into higher returns. Answering this question requires three operational steps that unfold within the same experimental pipeline. First, the study identifies cointegrated ETF pairs. Second, it assesses the prediction accuracy of the models receiving identical features. Third, it converts those forecasts into a common z-score mean-reversion strategy using a simulation and compares the realised Return Scores. The remainder of the document unfolds as follows. chapter 2 reviews the theoretical foundations, surveys deep learning advances relevant to PT and sharpens the research gap. chapter 3 then explains the full experimental design, including the implementation details of each model. chapter 4 presents the empirical results, pair-selection diagnostics, predictive accuracy figures and trading simulation outcomes, while chapter 5 interprets those findings, relates them back to the research questions and discusses limitations such as trend sensitivity and assumptions proven to be false in section 5.1, whereas section 5.2 draws the main conclusions and answers the central research question. Finally, section 5.3 sketches avenues for future work, while also emphasizing how methodological design could be improved.

2. Background and Theory

2.1. The Research Problem

This study looks into the application of different models into **Pairs Trading**; PT, which can be explained as follows. When one asset's price is predicted to drop compared to the other, capital is allocated to the cheaper asset (*go long*), while selling short [1] the more expensive asset (*go short*). The expectation is that their prices will realign over time, creating a profitable opportunity. This specific method will be examined because the problem of forecasting a univariate spread for the purpose of guiding a PT strategy is well-suited to be solved by machine learning techniques. Machine learning performs well at predicting inherently noisy, complex and nonlinear time-series data. Besides the methods being well-suited for application, there is a notable research gap regarding the application of advanced deep learning architectures to this domain, as discussed in greater detail in section 2.4.

2.2. Existing Literature

Since Gerry Bambeger introduced PT in the 1980s, the method has generated trading strategies exhibiting positive excess returns [Boo06]. In recent years, particularly over the past decade, PT has undergone a transformation, driven by the integration of deep learning techniques and exponential growth in computational resources.

2.2.1. Model History

Kalman Filter

In statistics, the self-descriptive term linear quadratic estimation is better known as the eponymous Kalman Filtering. It describes an algorithm specialized in modeling noisy data to produce estimates of unknown variables accurately. The choice to use this model is motivated mainly by the fact that it is one of the most basic statistical methods to forecast time series in a way similar to more complicated machine learning models. It works similar to the machine learning models in the way that it minimizes mean squared error [Kra16] . It is also chosen for its advantageous properties. First, it is particularly effective at modelling noisy data, making it well-suited for financial time series. Additionally, it is versatile: the Kalman Filter is applicable across a wide range of environments. It is also rooted in engineering and control theory. This origin outside of econometrics mirrors the other used deep learning architectures, Transformer and MoE.

Transformer

In 2017, two revolutional new deep learning methods have been discovered; the Transformer model architecture [VSP+17] and the Mixture-of-Experts architecture [SMM+17]. These methodological advances have yielded performance gains in natural language processing and have subsequently been found to be readily adaptable to time-series modeling [WZZ+22]. The Transformer model was introduced in 2017 with the seminal paper named "Attention is all you need". It introduced a revolutionary concept, the self-attention mechanism, to existing sequence-to-sequence models.

[1]In statistical arbitrage, short selling involves selling a borrowed asset with the expectation that its price will fall. This is often referred to as *selling short*.

This enabled the processing of entire sequences of data in parallel. Soon after, Transformer models also proved useful for time-series forecasting, specifically being applied to financial markets [LALP21]. As all models under current methodology take a 20-day period look-back tensor as the context to predict spreads with single-step forecasting, a multi-head Transformer encoder is used, in accordance with literature for its ability to process entire sequences of data [SAD+23].

MoE

Parallel to Transformer's advances, the Mixture-of-Experts approach has gained momentum as an innovative framework. Mixture-of-Experts (MoE) models, originally associated with ensemble-based methods in the early 1990s, have undergone a significant transformation with the introduction of sparsely-gated deep learning architectures by Shazeer et al. [SMM+17]. This modern formulation enables scalable model capacity with limited computational overhead, allowing for pipelines more efficient than more traditional dense architectures in deep learning. The foundational paper laid the groundwork for scalable large-scale neural networks, being valuable in many fields. MoE architectures scale to parameter counts on the order of billions while preserving computational efficiency. Because only a small subset of experts is active for each input, the effective floating-point operations per token increase only marginally. This sparsity makes MoE models suitable both for data-scarce scenarios (e.g., crisis-period economic forecasting, where sample sizes can drop by an order of magnitude) and for high-frequency applications that demand sub-minute training or inference latency.

2.2.2. Model PT Application

Previous studies have examined the application of Deep Learning architectures to PT. Daniels et al. found long-short term memory (LSTM) architectures to be the most effective for finding PT opportunities through predicting residuals between factor models[2] and actual prices [Dan22]. Kaur reported the same, but by more directly predicting pair spreads with LSTMs [Kau18]. In the end of Daniels' work, the use of Transformers was recommended for better results in future research. Later, Gradzki et al. explored the utility of Transformers in the context of foreign exchange markets, including PT. They found it to exhibit 'high predictive power' [GW24]. Therefore, this study adopts the model recommended by Daniels, using Kaur's methodology as the reference baseline. Although Gradzki's work provides additional evidence of Transformers' potential, it did apply this methodolofy to ETFs, as was examined by Kaur and Daniels. To get a deeper understanding of differences in deep learning architectures, this study will also examine the application of another deep learning architecture.

The Mixture-of-Experts approach, an architecture originally introduced to deep learning by Shazeer at al. in 2017, has also shown high predictive power in time-series models [SMM+17]. In contrast to the widespread adoption of Transformer-based models, Mixture-of-Experts (MoE) architectures remain underexplored. To date, existing literature includes only a single study focused on portfolio optimization [SWX+23], another on stock price prediction [Val24], and, most notably, no known studies into its use for PT. As discussed in section 2.4, the MoE model has demonstrated empirical robustness. This motivates its evaluation in the present study. More specifically, the foundation model Time-MoE [SWN+25], published in 2025, integrates MoE with transformer-based architectures for time series modeling. It represents a recent effort to improve time series models, and its superior capability and efficiency through outperforming dense models of equivalent computation budgets position Time-MoE as a state-of-the-art solution for time-series forecasting.

2.2.3. Feature Importance

In machine learning, identifying which features significantly contribute to performance is a crucial step in developing a robust pipeline. Through analysis of variance (ANOVA), Niaki and Hoseinzade found that important features for binary ETF price prediction were the returns of the 5 biggest companies and current exchange rates excluding past returns, volume and indicators [NH13]. In foreign exchanges, Aloud used Genetic Algorithms (GAs) to find that fundamental analysis contributed very little to the model's accuracy [Alo20]. Across extended prediction windows, specifically the 20- to 60-day look-ahead horizons, Liew and Mayster reported that a univariate model leveraging trading volume as the sole explanatory feature minimized forecast error relative to any higher-dimensional feature sets. For shorter timespans, they recommended "that model builders use a wide range of features guided by financial experiences and intuition" [LM17]. In the context of 5- to 60-day horizons[3], prior research and domain knowledge suggest that fundamental analysis contributes minimally to predictive performance. Accordingly, this study defines the input tensor as $\mathbf{X}_t = [p_0, ..., p_{19}]$ where p_t is the closing price, thereby minimizing feature space dimensionality and eliminating a priori assumptions on feature importance. The look-back window of 20 days is chosen based on literature research rather than a systematic optimisation of the window size.

[2]Factor models, such as e.g. the Fama-French model, explain asset prices using a linear combination of economic or financial *factors*.

[3]This is a similar range to the range used in this experimental design. For the trading strategy, a z-score is calculated based on a short-term mean and standard deviation (5-day time horizon) and long-term mean (30-day time horizon)

2.2.4. Data Selection and Rationale

The choice of which assets to use is a critical factor in ensuring the robustness of the end of the pipeline; the trading simulation. The following factors underpin the selection of exchange-traded funds (ETFs) listed on the NYSE and NASDAQ. Firstly, according to Rudy et al., "one of the greatest risks known for pair trading" is the fact that companies behind shares can go bankrupt. They found that this risk is effectively eliminated by limiting the total set of possible assets to ETFs. ETFs do not have a possibility of a total loss as one cannot become bankrupt. This results in a more consistently profitable strategy. Second, one of the biggest advantages of ETF pairs over stock pairs is how the fundamental economics of ETFs are more mean-reverting, and thus pair cointegration is less likely fo fall apart [Cha13]. Third is the identified research gap. As mentioned in section 2.4, other markets such as the Brazilian market have been tested for ETFs, but not the U.S. market. Fourth is the consideration of liquidity. U.S. markets are known for their high liquidity, which is crucical for efficient execution of pairs trading strategies. In a later step of the pipeline, illiquid ETFs will also be removed from the total set to limit the exponential growth of the combinatorial problem. Fifth is the specific suitability for deep learning architectures through dataset size. Transformers and MoE-architectures are are data hungry models. Therefore, a market that has a large size of possible ETFs and a long history of prices ensures them to reach their potential more fully. Considering these factors, this research focuses on ETFs listed on the U.S. market (NYSE & NASDAQ). They are expected to provide the most relevant and insightful results for the deep learning models explored in this study given the research objectives.

2.2.5. Pair Selection Strategy

For pair selection strategy, there are two viable approaches. The first approach would be a sector-based strategy [tH22]. Many ETFs are created as a bundle of assets inside a specific business sector, with the aim of giving the investor the possibility to invest in an entire sector at once. There is ongoing academic debate regarding how this affects the predictability and profitability of the resulting pairs. Ter Horst found a significant difference in accuracy between different sectors, ranging from a test RMSE of 5.97 in the real estate sector to 60.21 in the information technology sector [tH22]. To further support this, Do and Faff report that within-industry pairs tend to have higher absolute returns [DF10]. Other studies however, find the opposite to be true. Singh and Khushi fail to reject the null hypothesis that there is no significant difference, finding no significant improvements in performance for their ML model [SK21]. Moreover, Sarmento and Horta found that using sector-based analysis for finding ETF pairs leaves "a small margin for profit" [SH20].
The second possible approach is to select pairs based on the statistical properties of their time series. In 2006, Gatev, Goetzmann, and Rouwenhorst found a performance of negative mean returns (-1.37%) for random pairs picking, as compared to 1.44% MoM [GGR06]. Zhu examined whether these findings still hold in more modern contexts. They found in 2024 that the classic distance-based method still yields 6.2% annual excess return and a Sharpe ratio of 1.35. Therefore, they conclude that opportunities still exist in modern markets [Zhu24].
This study makes use of the Engle-Granger method for finding mean-reverting ETF pairs [EG87]. This is firstly due to the absence of consensus on sector-based analysis. The modern shift in ML-based pairs trading towards statistical approaches also substantiates this. As further support, this study focuses on finding models that provide robust performance under varying market conditions. Statistically grounded pair selection strategies offer a more data-driven foundation than sector-based approaches. Correlations within a sector can become unstable under volatile market conditions.

2.3. Overview of Applied Architectures

This section examines the underlying architecture of each model. The fundamental structure of each model is analyzed as well as the characteristics of forecasts the model might be expected to make. Eventually, the time series forecasting problem is mathematically formulated.

2.3.1. Kalman Filter

Model Architecture

In statistics, the self-descriptive term *linear quadratic estimation*, which is better known as the eponymous *Kalman Filtering*, describes an algorithm specialized in modeling noisy data to produce estimates of unknown variables accurately. For the application in this study, it is utilized as a baseline classical statistical model[4]. While originally often used in guidance, navigation and control of vehicles, Kalman Filtering has also been found to be useful in applications in time series analysis, such as signal processing and econometrics [ZM00]. Kalman filtering comprises two sequential steps. The first is the prediction phase, used for predicting the spread in the methodology (chapter 3):

$$\mathbf{x}_{t|t-1} = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_{t-1} + \mathbf{w}_t \tag{2.1}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}^T + \mathbf{Q} \tag{2.2}$$

With $\mathbf{x}_{t|t-1}$ the predicted value at time step t (using all values, known as the state, up to time $t-1$ to make this prediction), before applying the new measurement, $\mathbf{F}$ the transition matrix (controls to what extent the *state* is updated), $\mathbf{B}$ the control matrix, $\mathbf{u_t}$ the control input vector at time t, $\mathbf{P}$ the covariance matrix (representing the uncertainty in the prediction of the value), $\mathbf{Q}$ the process noise covariance matrix, $\mathbf{w}_t$ the noise at time t (with the error noise not perfectly Gaussian in the case of the spread in the current study).
Then, the second step is correction of the current model, consisting of:

$$\tilde{\mathbf{y}}_t = \mathbf{z}_t - \mathbf{H}_t\mathbf{x}_{t|t-1}, \tag{2.3}$$

with $\tilde{\mathbf{y}}_t$ the measurement residual (how unexpected the actual measurement is), $\mathbf{z}_t$ the actual measurement at timestep t, $\mathbf{H}$ the observation matrix,

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^T + \mathbf{R})^{-1}, \tag{2.4}$$

with $\mathbf{K}_t$ the Kalman Gain at time t (determines the weight given to the measurement in updating the estimate) and $\mathbf{R}$ the measurement noise covariance matrix.

$$\mathbf{x}_{t|t} = \mathbf{x}_{t|t-1} + \mathbf{K}_t\tilde{\mathbf{y}}_t, \text{and} \tag{2.5}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t|t-1} \tag{2.6}$$

with $\mathbf{I}$ the identity matrix. Conceptually, the prediction step uses no new data, it projects forward based on what is expected of the system. The correction step adjusts this prediction based on the actual observation at the current time step. The residual from that is then used to update the state estimate $\mathbf{x}_{t|t}$ and the uncertainty $\mathbf{P}_{t|t}$.
This theoretical foundation is crucial for understanding the behavior and interpretability of the filter's outputs. However, practical implementation is streamlined through modern libraries, as further detailed in chapter 3.

[4]In this paper, Kalman filters are classified as classical statistical models because they rely on explicit Gaussian-noise assumptions, identical to those employed in linear regression and ARIMA models. This classification draws a distinction between traditional econometric frameworks and deep learning approaches to time-series prediction.

Implementation Details

Applying Kalman Filters to PT can be done in several ways. This study applies a two-step Kalman approach, based on Kaur's methodology [Kau18]: (1) individual smoothing of each ETF price series using a simple [5] Kalman Filter, followed by (2) recursive Kalman Filter Regression in the form of a linear model $y_t = \beta_t \cdot x_t + \alpha_t + \epsilon_t$, which models a linear relationship between the smoothed series. In short, this is done by recursively updating the estimates α and β at each timestep t, based on the incoming price data x_t at timestep t.

Considerations of Gaussian Assumptions

The Kalman Filter is theoretically an optimal estimator under the assumption that the noise is Gaussian. This assumption represents a known limitation of the method. Empirical studies reveal that the distribution of spreads between cointegrated asset pairs exhibits statistically significant deviations from Gaussian distributions, exhibiting heteroskedasticity [MPV05] and "much heavier tails" [Shu23]. It must be noted that a misconception in literature is that applying Kalman Filters to non Gaussian data is not applicable, with Uhlmann and Julier finding roughly a dozen such instances [UJ24]. They report that it is still a rigorous application.
Using exclusively classical statistical methods, it is difficult to overcome such assumptions, as these methods often rely on fixed distributional assumptions. Therefore, this study proposes to apply deep learning-based approaches to time-series prediction, which are more flexible and can learn from data without requiring explicit parametric assumptions about underlying noise distributions.

2.3.2. Transformer

Model Architecture

Deep learning models offer a powerful alternative to classical statistical methods for time series prediction. Transformer, being a deep learning model, makes no strong assumptions about the underlying data distributions. The core advantage of the Transformer model is its ability to capture long-range contextual relationships between datapoints through self-attention mechanisms.
Transformer doesn't have an inherent understanding of the order of input tokens as Recurrent Neural Networks (RNNs) do, because all its layers are feed-forward. To still have contextual understanding of position, Transformers inject *positional encodings* into input tokens. The positional encodings utilized in the traditional Transformer model are absolute. Subsequent sections will examine alternative forms of positional encoding [6].
The traditional architecture, in the form it was originally proposed in the paper by Vaswani et al [VSP+17], consists of N encoder layers, and N decoder layers, where $N = 6$. The encoder layer consists of (1) a multi-head self-attention sublayer, and (2) a fully connected feed-forward network. All sublayers receive a residual connection, followed by normalization, which is referred to in the Figure 2.1 as *Add & Norm.*
The decoder layer contains the same two sublayers as the encoder, but with the addition of a third sub-layer, which comes before the two, applying masked self-attention. Decoders will not be explained in further detail, as only the Encoder of a Transformer is explicitly applied in this study. It is implicitly applied as the *experts* in the subsequent model Time-MoE are Transformer decoders. Of multiple forms of attention that exist, this architecture uses scaled dot-product

[5]'Simple' refers to using it as a denoising tool, without applying regression. It is modeled using a 1D model rather than the 2D model used for regression. This model is of the form $x_t = x_{t-1} + w_t, z_t = x_t + v_t$, where w_t and v_t represent process and measurement noise respectively. More explanation is detailed in chapter 3.

[6]See subsection 2.3.3 for Rotary versions of encoding, where the transformer *experts* from the Time-MoE architecture use this alternative form.

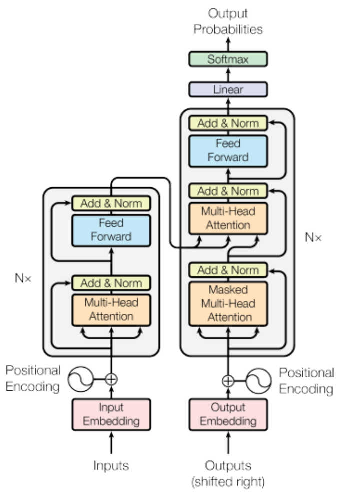


Figure 2.1.: General Transformer Architecture, Using Encoder and Decoder [VSP+17]

attention. It is multi-headed and self-oriented. Attention is calculated by applying the formula

$$\text{Attention}\,(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.7}$$

scaling the dot-product terms by $1/d_k$. This scaling, first proposed by Vaswani et al.[VSP+17], mitigates the vanishing-gradient problem by keeping the soft-max logits within a range where the resulting gradient magnitudes remain computational.
Then, to achieve multi-headed attention, multiple heads of attention are calculated (and later trained during backpropagation) in parallel. This means that the model is not only able to attend to a single part of the input tokens at one time, but many different tokens in parallel.

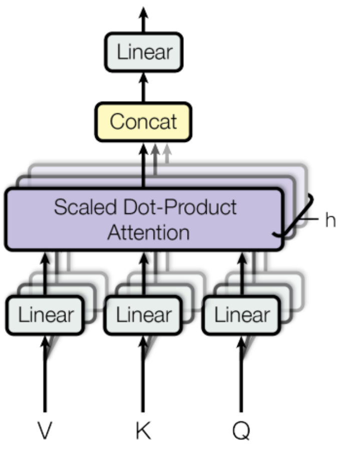


Figure 2.2.: Multi-headed attention [VSP+17]

As seen in Figure 2.2, these heads are then concatenated h times. Conceptually, this is what gives the Transformer architecture its ability to capture such complex contextual relationships.

Implementation Details

Applying a Transformer model to forecast a univariate spread does not *necessitate* applying modifications, as the model exhibits "high predictive power" in time-series prediction for intraday and end-of-day (EOD) prices, in contexts such as forex trading [GW24], stock price prediction [GOPZ21], bonds [PHL24] and generally financial time series [ZKS+23] as was mentioned in subsection 2.2.1. So while the Transformer architecture is already capable of producing competitive timeseries predictions in its standard form, some changes can improve Transformer's performance in this specific context. Single-step forecasting does not benefit from including a decoder in the architecture. Therefore, a transformer Encoder is used as the architecture, allowing for its ability to process entire sequences of data, namely the 20-period look-back window in the methodology of this study. chapter 3 explains the details of this implemented Encoder architecture.

2.3.3. Mixture-of-Experts

Model Architecture

A core advantage of MoE architectures over dense architectures is its sparse activation mechanism. This allows getting significant improvements in model efficiency, while still enjoying the benefits of multiple billion-parameter architectures. The gating network, as seen in Figure 2.3, has an output $G(X)_i$ for each expert i, known as the *gate value*. This value details how much that expert should contribute given the current input x. In the case of Figure 2.3, $G(x)_1$ and $G(x)_2$ are the only non-zero gate values, and therefore the only experts used in the context of the figure.

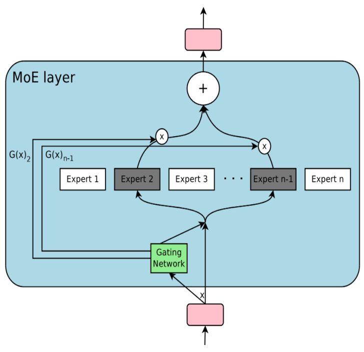


Figure 2.3.: Graph of a single MoE layer. In the specific case above, we see there are n experts, and only 2 out of n are chosen. This demonstrates the *sparse* element of the MoE architecture.

Sparse traninig in MoE does have some limitations. Most notably, MoE architectures often face the problem of *routing collapse*[7], where a few experts dominate in learning when focusing solely on optimizing prediction error. This problem can be mitigated, as explained in more detail hereafter.

[7]the term seems simultaneously coined in October 2023 by Chen et al. [CCD+23] as well as Ostapenko [OCS+23].

Implementation Details

To implement the MoE architecture, an existing architecture is used. Time-MoE is a pre-trained MoE model. Released in 2025 by Shi et al., it introduces several critical innovations beyond standard MoE architectures. As a way to mitigate routing collapse, an auxiliary loss has been found to mitigate this effect by Dai et al. for the development of Switch Transformer [FZS22]. The formula for this auxiliary loss as used in Time-MoE is given in Equation 2.8:

$$\begin{aligned}\mathcal{L}_{\text{aux}} &= N \sum_{i=1}^{N} f_i r_i, \quad \text{where} \\ f_i &= \frac{1}{KT} \sum_{t=1}^{T} I, \quad \text{and} \\ r_i &= \frac{1}{T} \sum_{t=1}^{T} s_{i,t}\end{aligned} \tag{2.8}$$

where f_i represents the fraction of tokens assigned to expert i, r_i denotes the proportion of router probability allocated to expert i, N the number of experts, and I the indicator function [SWN+25].
Another advantage of the model, distinguishing it from task-specific models with fixed horizons, is its flexible horizon length and context window. This is possible due to its decoder-only transformer architecture with rotary positional embeddings, as explained in the section before. See Figure 2.4 for the architecture of a single attention layer with RoPE. Third, the model utilizes multiple-resolution forecasting. With P forecasting heads, each head predicts the next p_j steps (p_j being the forecasting horizon for P, like 1, 8, 32, ..). This makes the model more robust and generalizable across applications. Fourth, the model is pre-trained on a dataset coined *Time-300B*, "the largest open-access time series data collection comprising over 300 billion time points across 9 domains" [SWN+25]. Lastly, Time-MoE emphasizes practical deployment through its sparsity by scaling to 2.4B total parameters with only 1B activated, making it efficient for inference on modest hardware.
Rotary Positional Embeddings (RoPE) improve upon traditional positional embeddings by encoding *relative distance* between time steps rather than absolute position. This allows the model to better generalize to longer or unseen sequences. RoPE was introduced by Su et al in 2021 [SLP+21]. Figure 2.4 shows an attention layer which incorporates RoPE inside the layer.

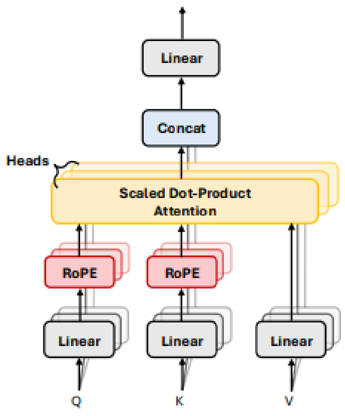


Figure 2.4.: A single attention layer in the Time-MoE architecture, utilizing RoPE to achieve flexible inputs.

This embedding is used in the Transformer models inside of the Time-MoE model.

2.3.4. Trading Strategy Based On Forecasts

The preceding sections have outlined the theoretical foundations for generating price spread predictions using various modelling techniques. Despite the difference in underlying methodology, the output of these predictions are of identical form regardless of the model. The z-score trading strategy is a method to simulate returns based on the generated predictions. Note that the main aim of this paper is to compare models across various measures. This strategy allows for an additional dimension in metrics. Specifically, the focus lies on assessing the returns a model can achieve compared to its maximum theoretical ability, as detailed further in subsection 3.5.2. A straightforward method to translate spread predictions into signals is to use the z-score to generate these signals. The exact methodology for this is described in section 3.5. Once the model has been used to predict the spread between two cointegrated assets, the trading strategy is constructed based on the principle of *mean reversion*, meaning that the spread fluctuates around a stable long-term equilibrium. Deviations from this mean equilibrum are used to create a signal for a trading opportunity.
As mentioned before, the aim is not comparing portfolio allocation methods to find the optimal trading strategy. Therefore the simplest method using the z-score is applied. For completeness, some other methods that are comparatively more complicated of translating spread predictions into signals are the Bollinger-Band rule [Bol92, MRD+24] or Ornstein–Uhlenbeck (OU) entry/exit levels (also based on mean-reversion) [EVDHM05]. Moreover, using the forecasted version of other series such as directly forecasting price of each asset can be used to generate signals through the same methods as mentioned before.

2.4. Research Gap

2.4.1. Existing Applications of Transformers

Previous studies have applied Transformers to stock price prediction in general contexts [Hu21, XDB24]. In contrast, research of specifically ETF price prediction has predominantly utilized RNN architectures such as LSTM [tH22], or BiLSTM and CNNs [Zhe21]. These models remain the current state-of-the-art for ETF price forecasting, while more advanced deep learning architectures have not yet been extensively explored or benchmarked in this domain.
One such example of the use of mean-reverting ETF pairs for statistical arbitrage with deep learning is the application of Transformers for ETF pairs arbitrage in the Brazilian market [HHLS24]. However, no studies have yet been known to address this approach for ETFs in the U.S. market using Transformers. Their effectiveness has been empirically demonstrated in U.S. pairs-trading across multiple asset classes; Gradzki (2024) reports that Transformer architectures outperform LSTM networks in high-frequency foreign-exchange trading [GW24]. This presents a promising avenue for further research into the application of Transformers to ETF-based pairs trading.

2.4.2. Existing Applications of MoE

To the best of my knowledge, there is currently no scientific research available on the application of MoE architectures to statistical-arbitrage applied on ETF pairs. MoE models are beginning to demonstrate their potential in stock time series modeling; one study explored its application to portfolio optimization [SWX+23] and one at general stock price prediction [Val24].
The recently proposed Time-MoE architecture has potential for this specific approach. It has been pre-trained on financial data as one of nine domains, and it is possible that it provides a more nuanced and reliable forecast compared to a vanilla Transformer Encoder architecture [SWN+25]. An other possible MoE-based time series foundation fodel is Moirai-MoE, which is not chosen

due to underperformance compared to Time-MoE [LLW+24]. Many other open-source large time series models are dense architectures, such as Chronos [AST+24] and Timer [LZL+24]. Seeing as Time-MoE is released in the same year as the current work, direct applications of Time-MoE in the literature remain limited. A thorough review of papers citing the original Time-MoE model reveals that most citations originate from studies proposing alternative architectures or methodological comparisons, rather than practical implementations. However, there are exceptions: at least three recent studies have successfully applied Time-MoE to real-world forecasting tasks, including supply chain management [WZL+25], agricultural price prediction [ZWZ+25], and continual learning for time series data [LJF+25]. Detailed analysis of the findings of these studies respectively show that fine-tuning the model results in consistent improvements. As well as the model forecasting the most accurately in monthly vegetable-price predictions, and finding that it "struggles with the irregular and volatile nature of short-term price fluctuations". Lastly, the model performed best in *incremental learning*, an approach where a machine learning model learns from a continuous stream of data over time. Incremental learning is not implemented in the current methodology. It does provide interesting insights for future work, as discussed in section 5.3. These works overall highlight Time-MoE's strong performance in challenging settings, such as forecasting after fine-tuning, and underscore its potential for robust adaptation across domains. The demonstrated robustness of the model provides a compelling rationale for its deployment in this setting.

2.5. Motivation and Significance

Bridging this research gap is significant for several reasons. First, the gap represents a research area with limited prior exploration despite "machine learning and neural network models consistently [outperforming] traditional forecasting methods in terms of precision and robustness" [HMP+25]. While both architectures have shown significant promise in time-series prediction, their performance in statistical arbitrage remains untested in this context of ETF PT in the US market. Second, by grounding the study in this domain, the work facilitates the analysis of core differences between Transformer-based and MoE-based models. Further, the comparison highlights trade-offs in architectures such as absolute vs rotary positional encoding, sparse vs dense training, pre-trained vs random[8] initialization. Third, ETF pair spread prediction is a testbed from which we can learn many aspects. This can help bridge the gap between academic model innovation and applied machine learning in noisy, high-dimensional environments. Application to other domains is further explored in subsubsection 5.3.1. For example, this allows for a comparison against findings of previous applications of the model to other contenxts, such as previous claims that the model struggles with the volatility of shorter time frames [ZWZ+25].

2.6. Academic Contribution

In quantitative finance academia, research often explores trading strategies for two primary purposes: to challenge the Efficient Market Hypothesis (EMH) [Fam70] by finding inefficiencies in the pricing of certain assets, or to compare state-of-the-art techniques to existing techniques for comparison in terms of profitability and risk. This paper will contribute in the way of the latter. This will have multiple consequences for prioritisations. Absolute profitability will be of secondary importance. Instead, emphasis is placed on the relative performance of advanced models against existing proven techniques, rather than against baselines such as a buy-and-hold

[8]Transformer often initializes its weights *strategically random*, using specific methods such as Xavier or Kaiming initialization. Radford et al. found a weight initialization method to scale across layers when developing GPT-2, where scaling the weights of residual layers by $\frac{1}{\sqrt{n}}$, with N the number of layers, preventing the accumulation of activations in layers by scaling down. This study uses Kaiming, Xavier and standard normal distribution initialization for several parts of the Transformer initialization.

strategy. As a consequence, a measure coined *Return Score* is introduced in subsection 3.5.3. Return scores are compared between models, rather than the *Sharpe Ratio*, a common metric in quantitative finance academia. This paper will contribute academically by applying proven state-of-the-art architectures in time series stock prediction to a new dataset: prediction of the price spread between two ETFs.

2.7. Research Questions

Despite the rise of deep learning architectures, there remains limited empirical understanding of how the examined models compare quantitatively in accuracy and trading performance, and qualitatively in terms of robustness and diverse applicability. This study aims to bridge that gap by evaluating whether the recent advances of Transformers and Time-MoE offer measurable improvements over classical statistical methods.

Research Question:

> **To what extent can the Transformer and finetuned Time-MoE model improve price spread prediction between two ETFs relative to a classical statistical Kalman-Filter approach?**

Operational Sub-Question 1 (Pair Discovery). Among all sufficiently liquid ETFs listed on the NASDAQ and NYSE, which pairs of two ETFs exhibit the strongest cointegration scores according to the Engle-Granger two-step cointegration test?

Operational Sub-Question 2 (Prediction Accuracy). Which model architecture attains the lowest mean-squared error (MSE) when forecasting that pair's price spread?

Operational Sub-Question 3 (Profitability). Which model architecture yields the highest Return Score when its forecasts drive a z-score mean-reversion strategy?

3. Methodology & Study Design

This section outlines the steps taken to develop a data pipeline for pairs trading using Kalman Filters, Transformers and the Time-MoE architecture, with the goal of describing the exact methods. The methodology consists of multiple steps: data collection and preprocessing, picking pairs, baseline model implementation (Kalman Filter), Transformer implementation and Time-MoE implementation, followed by a trading simulation based on all model's predictions.

3.1. Methodological Foundation

A key aspect of the methodology is that it builds upon the foundational work by Kaur, particularly the Python codebase described in [Kau18]. The findings of the work were first manually validated. Elements of the original implementation were then directly incorporated and subsequently modularized into distinct functions to enhance code clarity and reusability.

3.1.1. Adapted Components

Other components from Kaur's methodology were initially adapted in their existing form and then modified to align with the specific requirements of this research: *(1) finding cointegrated pairs through the Engle-Granger test, (2) Kalman Filter Regression and (3) A z-score trading strategy.* An important caveat in Kaur's implementation of Kalman Filter Regression is the introduction of look-ahead bias due to applying z-score normalization using statistics computed from the full time series, thereby incorporating future information into past predictions. Further explanation on z-score scaling is given in subsection 3.3.4

3.1.2. New Components

Furthermore, several components were designed and implemented independently to address unique aspects of this study: *(1) gathering data of all NYSE and NASDAQ-listed ETFs*[1]*, (2) caching logic for retrieving exact same data to improve replicability of results (2) filtering raw data based on incomplete time series and liquidity, (3) filtering resulting pairs data, (4) designing, training and inferencing the Transformer model, (5) training and inferencing the Time-MoE model, (6) designing supporting functionalities for gathering results, e.g. the Return Score as mentioned in subsection 3.5.3 and (7) all logic for diagnostic analyses.* The methodology of (7) is not described, but results can be found in Appendix B.

Accordingly, in the following methodological subsections, each step is described in detail based on the final implementation, rather than documenting the granular differences between this work and Kaur's original approach.

[1]Though Kaur's study incorporated the collection of stock price data, this process was facilitated using a deprecated software package that has been nonfunctional since the release of Python 3.

3.2. Dataset Construction

3.2.1. Data Gathering

The dataset, comprising historical closing prices for all ETFs traded on the NYSE and NASDAQ, was obtained using the *yfinance* Python module [Aro19], which retrieves financial data from the Yahoo Finance platform [Yah25a]. Python's *yfinance* library does not directly offer a method to gather the data on which assets exist, what type of asset they are, and on which exchange they can be found. For that first step, yahoofinance's *ETF screener* was used [Yah25b]. A webscrape of all ETFs gave a total of 871 *tickers*[2] for the largest time period 2008-2024, as it is a superset of ETFs found in the subperiods introduced further in the methodology through expanding windows. 777 of the tickers provided were valid. Given a valid ticker symbol, the *yfinance* library enables the retrieval of comprehensive historical market data for a specified asset. This dataset includes, but is not limited to, the asset's opening and closing prices, intraday price points, and daily trading volume. While the present study will focus on utilizing closing prices as the primary feature for model training, the established data pipeline is readily adaptable for alternative price intervals (with the shortest intraday interval being supplied by yahoofinance specifically being per-minute) or additional metrics, such as trading volume forecasting.

3.3. Data Preprocessing

3.3.1. Incomplete Data

From the 777 ETFs in the sample, some ETFs do not have observations spanning the full analysis period due to varying inception and closure dates. For example, certain ETFs were only launched at a date after the start of the training window, while others ceased trading before the end of the test window. As a result, the set of ETFs included in the analysis for a given period depends on data availability. To illustrate, 101 out of the initial 777 ETFs have a complete series of closing prices and are retained for analysis for the period starting *2008/01/01* and ending *2024/12/31*. Differences can be found within the expanding windows ending between 2016 and 2024. A further analysis on this concept, further referred to as *data availability* is given in subsubsection 5.1.1.

[2]Tickers are unique identifiers for assets on an exchange. For example, accessing *Vanguard Total International Stock Index Fund ETF* can be done through its unique accessor *VXUS*.

3.3.2. Liquidity

To ensure higher data quality, another step of filtering has been added; a minimum threshold for liquidity. Given that the intended strategy will be executed on a per-day basis, the required liquidity levels are not as stringent as those necessary for higher-frequency contexts (per-minute or per-second). An illustration of the liquidity of the current pool is given in Figure 3.1 using a logarithmic histogram. For the remainder of 101 ETFs, the average volume is taken to give an idea of the distribution of trading volumes across assets. This illustrates that the average Daily Trading Volume (DTV) predominantly ranges between 10^4 and 10^6.

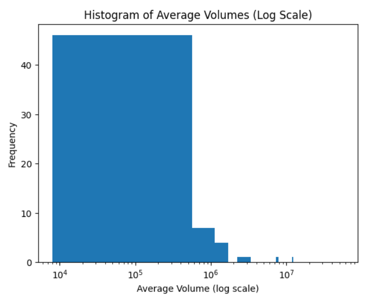


Figure 3.1.: Histogram of Daily Trading Volume (DTV) for all ETFs

Based on this distribution, a threshold of 10^5 was selected, as it lies near the center of the observed volume range and serves to exclude the lower tail of the distribution, thereby retaining a sufficiently large pool of assets while mitigating the risk of low-liquidity artifacts. It is important to note, however, that daily trading volumes can fluctuate throughout the observed period. In addition to enhancing data quality, applying a minimum volume threshold aligns the experimental conditions more closely with real-world market environments. This reduces the price impact of individual trades. Given that the subsequent simulations at the end of the pipeline operate under the assumption of negligible market impact, restricting the asset pool to higher-volume securities minimizes the discrepancy between this assumption and actual market behavior. As such, this filtering step not only improves the robustness of the dataset but also increases the external validity of the simulation results.

3.3.3. Finding Pairs

Once the data has been filtered on completeness and liquidity, the Engle-Granger test is performed on the train window. Using a brute force search, a p-value indirectly based on the closing price of all respective pairs is calculated [3]

For each possible pair of stocks S^1 and S^2, an ordinary least-squares (OLS) regression is executed with the first ETF's closing price time series P^1 [4] as the independent variable and P^2 as the dependent variable plus a constant shift c. Including the constant c allows the spread to revert to a non-zero mean; omitting it would force the long-run mean to zero. The regression equation is given as:

$$p_t^1 = \alpha + \beta \cdot p_t^2 + \epsilon_t \tag{3.1}$$

where α represents the equilibrium spread level between the two assets, β is the hedge ratio, and ϵ_t is the spread residual representing temporary deviations from the equilibrium relationship.
To test whether the spread is stationary, an Augmented Dickey–Fuller (ADF) test is applied to

[3]the formula is applied directly from the *stattools* implementation [SP10]. An examination of equations applied for this implementation is given in this section.

[4]$P^1 = [p_t^1]_{t=0}^T$

the spread residual ϵ_t, not adjusted for constants. The ADF test regression is:

$$\Delta\epsilon_t = \gamma\epsilon_{t-1} + \sum_{i=1}^{l} \phi_i \Delta\epsilon_{t-i} + u_t \tag{3.2}$$

where l is the number of lags (determined by AIC criterion), and u_t is the error term. The ADF test statistic is:

$$\text{ADF} = \frac{\hat{\gamma}}{SE(\hat{\gamma})} \tag{3.3}$$

Lastly p-values are extracted from the ADF statistic , which is estimated using MacKinnon's approximate p-value [Mac94]:

$$p\text{-value} = \Phi\left(\sum_{i=0}^{n} \tau_i \cdot (\text{ADF})^i\right) \tag{3.4}$$

where Φ is the cumulative distribution function of the standard normal distribution, and τ_i are the MacKinnon coefficients that depend on the regression type (in this case, only a constant) and sample size. All pairs for which the closing prices P^1 and P^2 yield a p-value less than 0.05 are retained for further analysis in the pipeline. Figure 3.2 shows the amount of resulting pairs with sufficiently low p-values out of the total set of combinations over all studied years.

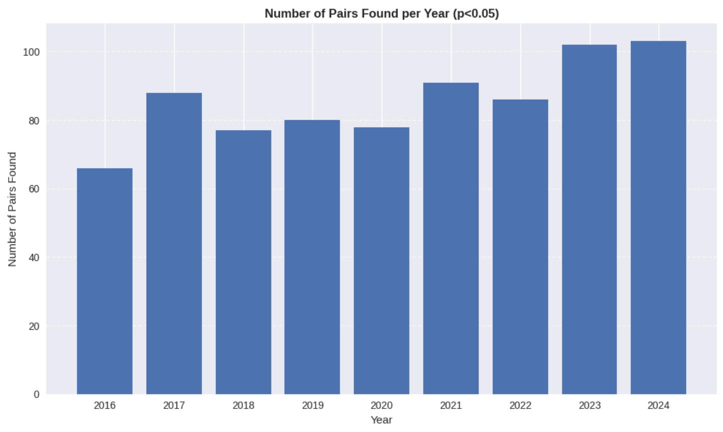


Figure 3.2.: Number of sufficiently cointegrated pairs for all validation and test windows.

For further filtering, pairs with a perfect cointegration score of 0.00 are filtered out, as these stem from different tickers tracking the exact same closing price. The spread is always zero, a time series not possible to exploit or meaningfully predict.

3.3.4. Scaling

In this study, z-score normalization is chosen as the principal scaling method. Under z-score normalization, each time series is transformed to have zero mean and unit variance, according to:

$$p_t^{norm} = \frac{p_t - \mu_w}{\sigma_w} \tag{3.5}$$

with p_t^{norm} the normalized price point of at time t, μ_w the mean of a chosen window w and σ_w the standard deviation of a chosen window w. The window w is in practice at least the length D of an entire year, and in the case of the train window (where look-ahead bias cannot occur) the chosen window w equals itself w_{train}. For validation and test, the window must precede the target window and is therefore a preceding window of length D; $w = w_{prev}$. The length of w_{train} is $k * D$, with k the amount of years in a train window (see section 3.6). To prevent look-ahead bias, test time series are normalized with the statistics from the window before, according to rolling z-score normalization, using:

$$p_t^{1,norm} = \frac{p_t^1 - \mu_{w_{prev}}}{\sigma_{w_{prev}}}, \tag{3.6}$$

with the test window w_{test} spanning from $t = (k) * D$ through $t = (k+1) * D$ (again, see section 3.6). Here, $\mu_{w_{test}-1}$ and $\sigma_{w_{test}-1}$ are computed from a window $w_{prev} = w_{test} - 1$ that precedes the window w_{test} of length D and these are used to scale the test window. This approach ensures temporal causality is maintained and future information is not leaked into earlier phases of model development. In the case of calculating validation metrics using time series cross-validation, the exact same calculation is used as for test metrics, but these are gathered from sliding time windows. Further explanation of details is given in section 3.7

3.4. Application of Prediction Models

Each predictive model operates on input windows consisting of 20 consecutive data points, utilizing this sequence to forecast the subsequent day's value. This approach is consistently employed during both the training and inference phases of the models. Figure 3.3 shows an illustration of an arbitrary time series, where the next-day prediction is made based on a look-back window of the previous 20 days.

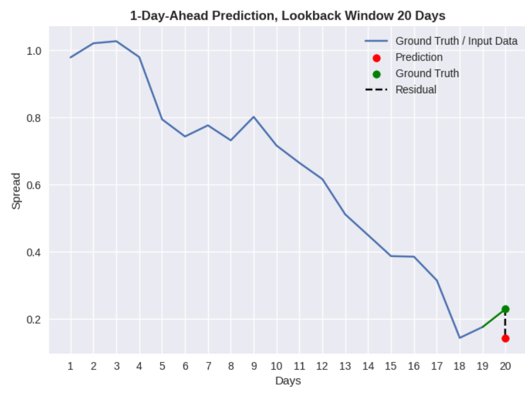


Figure 3.3.: Example of set of input data points used to predict a 1-day-ahead

3.4.1. Kalman Filter

The Kalman Filter is applied consecutively in two steps. The first step consists of a one-dimensional kalman filter, referred to as *Kalman Filter Averaging*. The second step executes regression to forecast future values, referred to as *Kalman Filter Regression*. The two steps are implemented with the library *pykalman* [C+10]. For each asset pair, a distinct time series is defined, denoted as P^1 and P^2 respectively. Univariate *Kalman Filter Averaging* is applied independently to each time-series in order to smooth out noisy price movements through choosing the state vector x_t to be one-dimensional; $\mathbf{x}_t = \mu_t$, where μ_t is the running mean of the observed series. This is then applied to the target time series $\mathbf{y}_t = P^1$, and later $\mathbf{y}_t = P^2$, resulting respectively in $P^1_{filtered}$ and $P^2_{filtered}$.
Then, for the next step, *Kalman Filter Regression* is applied on these time series $P^1_{filtered}$ and $P^2_{filtered}$. For this, an object is initialized taking in the observation matrix $H_t = \begin{bmatrix} x_t & 1 \end{bmatrix}$. Implementing prediction through the application of Equation 2.1 and Equation 2.2, where x_t is chosen to be a 2D vector in this case; $x_t = \begin{bmatrix} \beta_t \\ \alpha_t \end{bmatrix}$ such that β_t is the slope, also referred to as hedge ratio, and α_t is the intercept. This lets me define the regression equation 3.7, which is to be used internally to estimate the regression parameters via filtered data:

$$p^1_{filtered,t} = \beta_t \cdot p^2_{filtered,t} + \alpha_t + \epsilon_t \tag{3.7}$$

The β_t that results from fitting these state means is used to predict the spread $\hat{s}_{t+1}$ using Equation 3.8. This does not use filtered data as input.

$$\hat{s}_{t+1} = p^1_t + \beta_t \cdot p^2_t + \epsilon_t \tag{3.8}$$

Here, $\hat{s}_{t+1}$ is the predicted spread at time t and ϵ_t is the observation noise, assumed to be Gaussian white noise. Note that these Gaussian assumptions imperfectly model the time series data, as mentioned in subsubsection 2.3.1. Now that approximations have been made for the state vector, Equation 3.8 is used to predict the spread using β_t as input. Predictions are made over sliding windows as detailed in section 3.6.

Hyperparameter Search Space

For the Kalman Filter model, the following hyperparameters are optimized:

- **State Evolution Variance** δ: A parameter, with similar effects to learning rate in deep learning methods, that controls the update speed of the Kalman gain, optimized over the range $[10^{-5}, 10^{-1}]$ (log-uniform). Is used to calculate transition covariance Q.
- **Observation Covariance (Regression)**: A parameter controlling the role of observation noise in regression phase, optimized over the range $[0.5, 4.0]$ (log-uniform).
- **Transition Covariance (Averaging)**: The average value of the process (transition) covariance, regulating how much the system dynamics are allowed to deviate, optimized over the range $[0.001, 0.1]$ (log-uniform).
- **Observation Covariance (Averaging)**: A parameter controlling the role of observation noise in averaging phase, optimized over the range $[0.1, 10.0]$ (log-uniform).

Hyperparameter ranges are chosen through starting all hyperparameters in a small range, and expanding this range until the optimal hyperparameter found does not reach either limit of the range anymore, using the partial dependence plot to gauge the forecasting acuraccy's dependence on each hyperparameter:

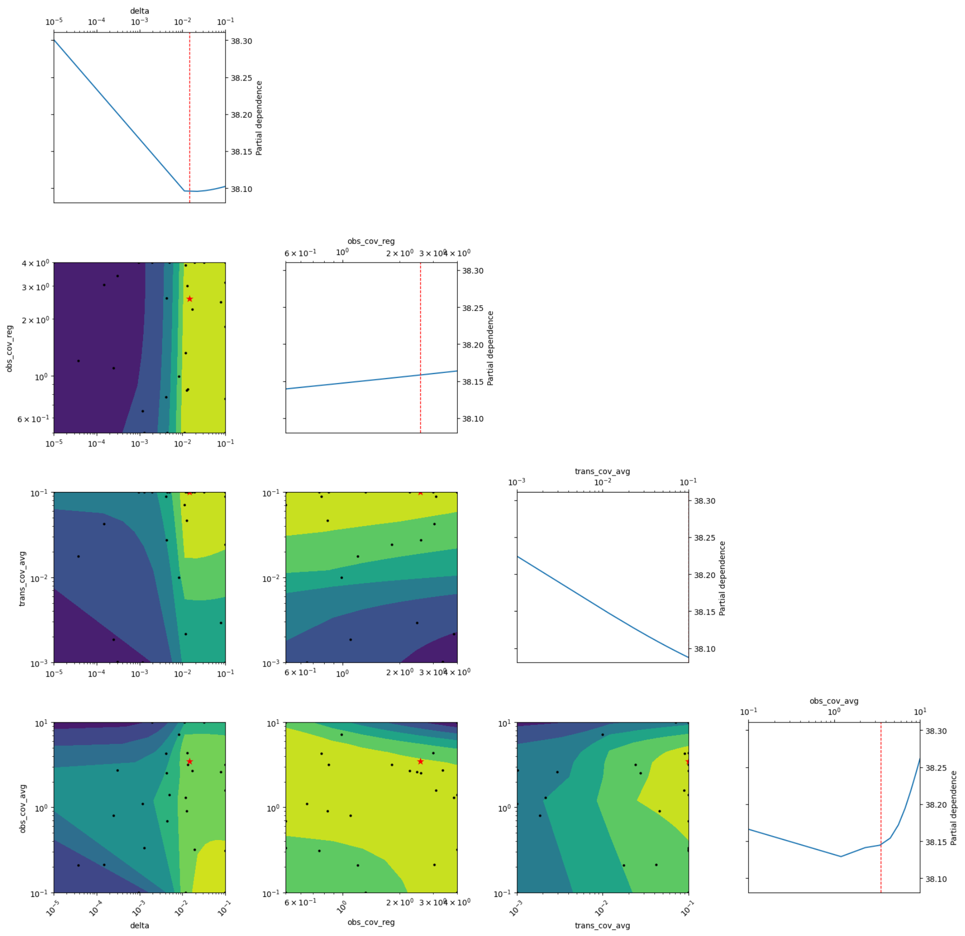


Figure 3.4.: Partial Dependence of Each Hyperparameter on Resulting NMSE.
Example: Reducing δ (top plot) from 10^{-5} to 10^{-2} results in an NMSE decrease of 38.30-38.10=0.20.

3.4.2. Transformer

The core architecture is a Transformer encoder, implemented in PyTorch, that processes sliding windows of univariate time series data. Time series splits are done in a rolling manner. Each input sample consists of a look-back window of features s_{t-19} to s_t for 20 timesteps:

$$X_t = \{p_{t-\ell+1}, \dots, p_t\} \tag{3.9}$$

with ℓ the look-back window of 20 time steps, and X_t the time series of the previous look-back window at time t. As mentioned in subsection 3.3.4 All features are scaled using z-score normalization. The model employs a Transformer encoder architecture tailored for time series regression. The core of the model is a stack of Transformer encoder layers n_{layers}, each employing multi-head self-attention with n_{heads} heads, and feed-forward sublayers of dimension $n_{layers} \times d_{\text{model}}$, with dropout regularization to prevent overfitting. Manual evaluations of loss behavior further confirmed the absence of overfitting. After encoding, the sequence is flattened and passed through a small feed-forward regression head: a fully connected layer with ReLU activation and dropout, followed by a linear layer projecting to a scalar output. This head produces the predicted value of $\hat{s}_{t+1}$ for each input sequence. Hyperparameters pertaining to the model architecture, model dimension (d_{model}), number of attention heads, dropout, and learning rate are selected empirically through optimization (see subsubsection 3.4.2).

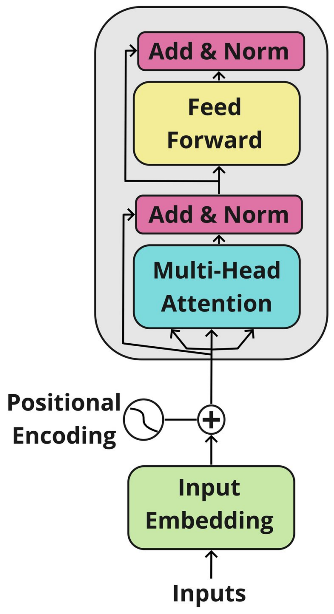


Figure 3.5.: Single Transformer Encoder Layer

Training is performed using Mean Squared Error loss (MSE) and the AdamW optimizer. Each training pair is subjected to a maximum of 400 epochs. An early stopping mechanism is employed after the initial 150 epochs, utilizing a tolerance threshold of 10^{-5} for the validation loss improvement. During evaluation, predictions are produced for the test set and then presented in the original scale. As mentioned in section 3.7, final test and validation accuracy is measured in NMSE. Using the test window, 1-day predictions are made for each of the inputs by rolling through inputs X_t for all t in the test window. The resulting predicted spread $\hat{S}$ of this testset is used as input to the trading strategy, as detailed in section 3.5, given by

$$\widehat{s}_{t+1} = \text{Transformer}(X_t) \tag{3.10}$$

where $\widehat{s}_t$ represents the predicted spread at time t. The entire predicted spread then consists of

$$\hat{S} = \{\widehat{s}_{t_{start}}, ..., \widehat{s}_{t_{end}}\} \tag{3.11}$$

where a test window runs from $t_{start} = (w + k) * D + 1$ through $t_{end} = (w + k + 1) * D$ as explained in section 3.6.

Hyperparameter Search Space

The Transformer model's hyperparameters are divided into two categories; (1) architecture hyperparameters, and (2) training algorithm hyperparameters. This is to ensure that the training algorithm hyperparameters can be matched exactly to the training algorithm hyperparameters optimized through bayesian optimization for in the case of Time-MoE.
For the architecture, the following hyperparameters are optimized:

- **Model depth** d_{model}: Specifies the dimensionality of the Transformer's internal layers, which impacts the model's capacity to learn comparatively more or less complex temporal dependencies, optimized over the range $[256, 512, 1024]$ (categorical).
- **Amount of heads** n_{heads}: The number of attention heads used in the multi-head self-attention mechanism. See Figure 2.2, optimized over the range $[2, 8, 16]$ (categorical).
- **Number of layers** n_{layers}: The total number of Transformer encoder layers stacked in the model, optimized over the range $[2, 6]$ (integer).
- **Dropout Rate** $p_{\mathbf{drop}}$: The dropout probability applied within the Transformer encoder and regression head, serving as a regularization mechanism to prevent overfitting during training, optimized over the range $[0.0, 0.3]$ (linear).

For the training algorithm, the following hyperparameters are optimized through bayesian optimization:

- **Learning rate**: The initial step size used by the optimizer for updating model parameters. A cosine learning rate scheduler is used to further progress this learning rate, optimized over the range $\left[10^{-6}, 10^{-4}\right]$ (log-uniform).
- **Minimum learning rate**: The minimum value to which the learning rate is decayed during the course of training, optimized over the range $\left[10^{-6}, 10^{-4}\right]$ (log-uniform).
- **Warmup ratio**: The fraction of total training steps used for learning rate warmup, gradually increasing the learning rate from zero to its initial value, optimized over the range $[0.0, 0.05]$ (linear).
- **Weight decay**: A regularization term that penalizes large weights to prevent overfitting, optimized over the range $[0.0, 0.3]$ (linear).
- **Batch Size**: The number of time series sequences processed in parallel during each training step, used to compute the gradient and update model parameters, optimized over the range $[64, 128]$ (categorical).

  For both deep learning architectures, the AdamW optimizer is used with the following hyperparameters:

- β_1: The exponential decay rate for the first moment estimates in the AdamW optimizer, optimized over the range $[0.85, 0.99]$ (linear).

- β_2: The exponential decay rate for the second moment estimates in the AdamW optimizer, optimized over the range $[0.9, 0.999]$ (linear).
- ϵ: A small constant added for numerical stability in the Adam optimizer. Affects stability and smoothness of parameter updates, optimized over the range $\left[10^{-11}, 10^{-8}\right]$ (log-uniform).

Hyperparameter ranges are chosen through starting all hyperparameters in a small range, and expanding this range until the optimal hyperparameter does not reach either limit of the range anymore, using the partial dependence plot similarly as explained in subsubsection 3.4.1.

3.4.3. Time-MoE

The Time-MoE model represents a model with modern transformer-style attention mechanisms. In this application, the publicly available *TimeMoE-50M* model is used as the base model, which is further fine-tuned on a train window. Training is carried out via the library's built-in learning algorithms according to hyperparameters such as batch size, learning rate, and all others mentioned in subsubsection 3.4.2 [Tim25]. After training on the train window, the finetuned version of the model is used for evaluation of validation or test metrics. During the inference phase, a rolling window approach is used on the test and validation sets, in the same method as is done for the Transformer. The finetuned model generates a one-step-ahead forecast ($\hat{s}_{t+1}$). The predicted spread series $\hat{S}$ is used as input to the trading strategy described in section 3.5.

Hyperparameter Search Space

For Time-MoE, the identical set of training hyperparameters is used as was done in the Transformer's workflow. See subsubsection 3.4.2 for an exhaustive list. Below, I outline several hyperparameters that, while potentially subject to optimization, remain fixed in the current experimental setup:

- **Warmup steps**: This parameter is already implicitly determined by the chosen warmup ratio. Therefore, it is not independently optimized.
- **Evaluation batch size**: The batch size used during model evaluation does not influence the training process or its outcomes; it is not considered as a hyperparameter for optimization.

Architecture hyperparameters are not optimized, as the model is pre-trained. Remaining hyperparameter ranges are chosen through starting all hyperparameters in a small range, and expanding this range until the optimal hyperparameter does not reach either limit of the range anymore, using the partial dependence plot similarly as explained in subsubsection 3.4.1. This is done separately from the ranges in Transformer, resulting in a different set of ranges.

3.5. Application of Trading Simulation from Forecasts

This simulation trades based on expected mean-reverting behavior of the spread between two assets, under the assumptions that deviations from equilibrium are temporary and will revert over time. The forecasts predict when exactly the spread will deviate from its mean, informing the generation of signals. At each timestep t, a model predicts the spread $\hat{s}_t$ between asset A and asset B. The set of predictions $\{\hat{s}_{t-\ell}, \hat{s}_t\}$ is then used to compute a short-term rolling mean $\mu_{short}(t)$, defined as

$$\mu_{short}(t) = \frac{\sum_{\tau=t-\ell_{short}}^{t} \hat{s}_\tau}{\ell_{short}}, \tag{3.12}$$

as well as a long-term rolling mean $\mu_{long}(t)$, given by

$$\mu_{long}(t) = \frac{\sum_{\tau=t-\ell_{long}}^{t} \hat{s}_\tau}{\ell_{long}}, \tag{3.13}$$

and a short-term rolling standard deviation $\sigma_{short}(t)$, calculated as

$$\sigma_{\text{short}}(t) = \sqrt{\frac{1}{\ell_{\text{short}}} \sum_{\tau=t-\ell_{\text{short}}}^{t} (\hat{s}_\tau - \mu_{\text{short}}(t))^2}. \tag{3.14}$$

Finally, the z-score at time t is computed as

$$z_{score}(t) = \frac{\mu_{long}(t) - \mu_{short}(t)}{\sigma_{short}(t)}. \tag{3.15}$$

Here, $\hat{s}_\tau$ denotes the predicted spread at timestep τ, while ℓ_{short} $(= 5)$ and ℓ_{long} $(= 30)$ are the sizes of the short and long windows respectively, and t is the current timestep for which the signal is being created. Figure 3.6 illustrates an example of z-scores, including position and clearing thresholds.

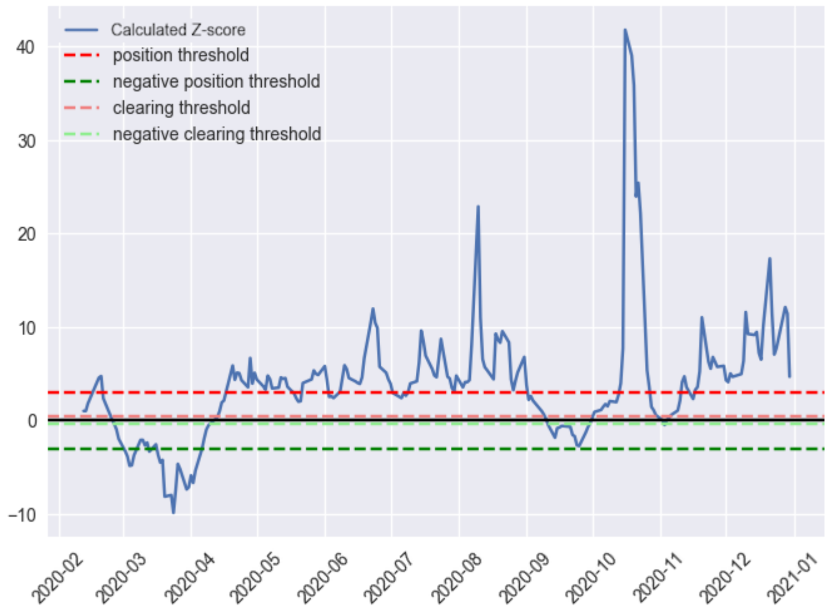


Figure 3.6.: Example of z-score Resulting from Arbitrarily Chosen Spread
"Threshold" Labels Show Thresholds for Signal Generation

The z-score calculated from this quantifies the extent to which the spread deviates from its recent mean, normalized by its volatility. Or as this z-score is calculated from predictions, the z-score quantifies the *predicted* deviation from its historical mean. Based on the value of the z-score, four scenarios can be distinguished, each associated with a specific action:

1. If $z_t > \theta_p$: sell asset 1 and buy asset 2 (short entry).
2. If $z_t < -\theta_p$: buy asset 1 and sell asset 2 (long entry).
3. If $|z_t| < \theta_c$: close all open positions (deviation from mean is not strong enough to bet on reversion)

4. If $\theta_p > z_{score} > \theta_c$ or $-\theta_p < z_{score} < \theta_c$: no action is undertaken (maintaining existing positions without liquidating current holdings)

The two threshold parameters θ_p and θ_c allow for a manual control of risk. As explained in subsection 3.5.1, uncertainty calculation is executed using a grid sweep. Moreover, the ground truth spread is used to calculate an upper bound for strategy performance. Using this upper bound and the mean return, a metric called the Return Score is calculated, which is explain in detail in subsection 3.5.3.

3.5.1. Returns from Simulation

To translate the z-score signals calculated from raw forecasts into an investment performance metric, the simulation is executed across a grid of configuration parameters. The stream of trading signals is converted into a path that the equity follows $E = \{e_0, .., e_t\}$ with each point e_t showing the equity on that day. First, it walks through time, updating available liquidity and the number of both pairs in the portfolio every time the z-score breaches either the positioning or the clearing threshold. A short z-score signal (referred to in item 1 as "short entry") removes a difference of $S_1(t) - S_2(t)\hat{s}_t$ from the liquidity account, and a long signal (referred to in item 2 as "long entry") performs the opposite operation. A clearing z-score signal liquidates every open position at the prevailing market price, resets both inventories to zero, and transfers the proceeds to cash. The first element e_0 of the resulting equity ath E coincides with the arbitrary starting capital $e_0 = C_0 = 10.000$, so the path can reflect an interpretable returns metric, rather than an absolute liquidity return. This allows for a calculation of the return percentage r using

$$r = (\frac{e_t}{e_0} - 1) * 100\%. \tag{3.16}$$

Though in the current methodology test windows are always exactly a year long (such that $r = r_{YoY}$ by default), one could ensure returns are yearly using

$$r_{YoY} = ((\frac{e_t}{e_0})^{D/T} - 1) * 100\%, \tag{3.17}$$

where D is the amount of trading days in a year, and T the length of the test window and equity path E.

Uncertainty Calculation

Uncertainty in the equity return paths is assessed by repeating the entire exercise on a parameter grid of different combinations of position threshold θ_p and clearing threshold θ_c. Calculating equity paths E under these different thresholds, each separate return percentage per run i is referred to as $r^{(i)}$. Mean annual returns and uncertainty can then be calculated using

$$\begin{aligned} \mu_r &= \frac{1}{N}\sum_{i=1}^{N} r^{(i)} \\ \sigma_r &= \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(r^{(i)} - \mu_r\right)^2}, \end{aligned} \tag{3.18}$$

where μ_r is the mean, σ_r the standard deviation and N the total amount of return percentages r. These metrics can be used for calculating the Return Score described in, after being converted to scalars subsection 3.5.3.

Total Loss of Equity

If the computed strategy equity e_t hits zero at any timepoint t, this conceptually means there is no more available liquidity. To prevent any possibility of overstating performance after extinction of liquid capital, any further point of equity e_τ for $\tau > t$ is set to zero. This guarantees that simulations which reach bankrupcy cannot re-enter the simulation on unrealistically favorable terms. Translating this into year-over-year metrics results in the following: because the terminal-to-initial capital ratio $\frac{e_t}{e_0}$ becomes exactly zero, the expression $r = (\frac{e_t}{e_0} - 1) * 100\%$ in turn returns -100%. In combination with the term D/T, this can give hard-to-interpret percentages[5], and therefore any such occurrences in the result are directly reported as a total loss of equity, referred to as *TLOE*.

3.5.2. Theoretical Return Under Perfect Information

This section establishes the metric that will later be referred to as the theoretical return under perfect information (TR-PI). It represents a theoretical upper bound on profitability for the z-score trading framework described in section 3.5 when the future one-day-ahead spread, s_{t+1}, is known without error. Let $\hat{s}_{t+1}$ denote the spread forecast produced by a model. In the simulation every forecast $\hat{s}_t$ is replaced with the ground truth spread s_t, after which, a simulation using ground truth spread s_t gives a resulting return percentage r. This percentage is the TR-PI. Figure 4.2 illustrates $\hat{s}_t$ and s_t respectively as "Predicted Spread" and "Actual Spread". The trading logic (Equations 3.12–3.15) and all hyper-parameters ($\ell_{\text{short}} = 5$, $\ell_{\text{long}} = 30$, position threshold θ_p, clearing threshold θ_c) remain unchanged. Hence, any discrepancy between TR-PI returns and model-generated returns can be attributed exclusively to forecast error and the residual stochastic variability inherent in the return-generating process.

Limitations of TR-PI

The upper bound TR-PI is consistently referred to as a *theoretical* upper bound, as it is still possible for predictions with an error to probabilistically outperform the strategy; the z-score strategy operates under the assumption that a spread will likely revert to its mean over time. This event is probabilistic and will only happen sometimes. A z-score-based trading strategy is not deterministically profitable, so even under perfect information negative returns can be attained, or total loss of equity (TLOE) as mentioned in subsubsection 3.5.1.

[5]A percentage of -100% YoY might imply that a yearly loss of exactly -100% could go on forever under those circumstances, whereas it actually means the strategy terminated at some point in that year. Therefore, direct percentages are replaced with "TLOE".

3.5.3. Return Score

The trading simulation gives different return percentages even when supplied with perfect information, which makes these percentages unfairly comparable to each other. For example, a model reaching 2% YoY returns when TR-PI was 30% should be deemed as a less profitable forecast then a 2% YoY return when TR-PI was 3%. To offer a more reliable metric to compare the profitability performance of different models with each other, the *Return Score* is introduced. In other literature, this score is similar to the *performance ratio*. The mathematical formula is given by

$$\rho = \begin{cases} -1, & \text{if } r_{model} = -1 \\ \text{N/A}, & \text{if TR-PI} = -1 \\ \frac{1+r_{model}}{1+\text{TR-PI}}, & \text{otherwise} \end{cases} \tag{3.19}$$

where ρ is the Return Score and r_{model} the actual return resulted from a model's forecast. The main concept of the score is that the returns resulting from ground truth information serve as a theoretical maximum return, represented by TR-PI. The return percentage achieved by feeding a model's predictions into the trading strategy is represented by r_{model}. r_{model} is likely to be somewhere below TR-PI. The closer r_{model} is to TR-PI, the better the model has performed. Any score above 1.0 means r_{model} is better than theoretical maximum returns. If the returns resulting from the perfect noiseless information are a loss of all equity, then $\rho = -1$, which will in turn result in a TR-PI represented in final results as N/A. In chapter 4, Return Scores that did not result in a *N/A* are discussed more in-depth, as the comparison between TR-PI and r_{model} is less meaningful if TR-PI results in a total loss of equity.

3.6. Time Series Cross-Validation

To ensure results are less dependent on regime shifts[6], a standard practice in time series forecasting is implemented; Time Series Cross-Validation. The time series cross-validation strategy employs an expanding-window framework, which respects the chronological order of observations by sliding the validation or test window by one year in each *window*. Unlike traditional k-fold cross-validation, this method ensures that future information never leaks into the training process as the training windows are, by construction, temporally constrained to precede its respective validation or test window.

3.6.1. Hyperparameter Tuning

To evaluate model performance, data from the years 2020 through 2024 are withheld for use as a testset. That held-out time series is later used for separately gathering test metrics and cross-averaging those metrics. This time series will be referred to as the *testset*, and is utilized for reporting the final metrics. Figure 3.7 shows the exact train, validation and test windows used in the current methodology. With the remaining set, referred to as the *complete train/validation set*, hyperparameter tuning is performed using expanding-window time series cross validation. This takes place as follows. The beginning of the complete train/validation set is defined at t = 0 at *2008-01-01*, as also illustrated in Figure 3.7. All closing prices from the beginning at $t = 0$ up to time $t = k * D$ are used as the first training window for window $w = 0$, where $k = 8$ is the amount of years skipped at the beginning and D is the length of a valdation window. In this case, $D \approx 252$, the amount of trading days in a year, as a window is exactly one year. D can differ

[6]**Regime shift** refers to a change in the overall structure of a financial time series as seen in statistical markers such as volatility, mean returns, correlations, or others.

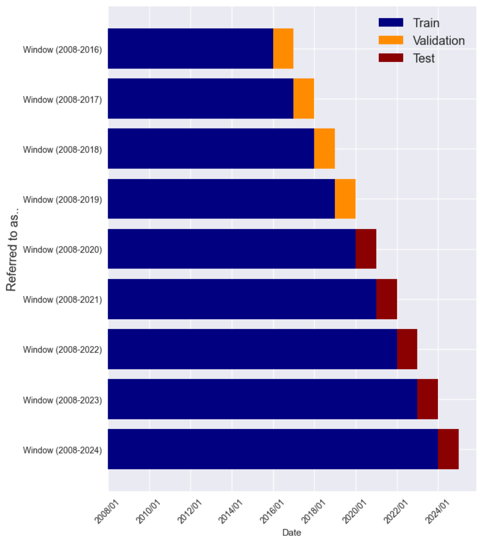


Figure 3.7.: Illustration of time structure of the expanding-window procedure. The y-axis indicates how each window is referenced throughout the study. The x-axis shows the corresponding calendar dates for the periods.

per calendar year. Then, the accompanying validation window begins at $t = k * D + 1$ and ends at $t = (k + 1) * D$. For all further windows w, the train window takes place from $t = 0$ through $t = (w + k) * D$ and the validation window from $t = (w + k) * D + 1$ through $t = (w + k + 1) * D$. To *roll* further to the next expanded window, the origin w is rolled forward by a fixed step size of one year such that the train window is exactly one year longer, and the validation window consists of a year for which data has not yet been accessed. This process continues until the validation window would exceed the end of the *complete train/validation set*. Model performance is averaged across all folds to obtain robust hyperparameter estimates.

3.6.2. Gathering Test Metrics

The sliding-window algorithm used for gathering validation metrics is the same as for gathering test metrics. Similarly, all closing prices from the beginning[7] at $t = 0$ up to time $t = (w + k) * D$ are used. However, this time starting at window $w = 4$ and ending at window $w = 8$ with endpoint $t = (w + k + 1) * D$ belonging to the date *2024/31/12*. Figure 3.7 illustrates each test and validation window.

3.6.3. Hyperparameter Optimization Algorithm

Each model allows for several hyperparameters to be set, impacting the model's accuracy. Kalman Filter's hyperparameters are based on noise assumptions and learning rates. Transformer's hyperparameters regard model architecture and train-specific parameters such as learning rates. Time-MoE, being a pre-trained architecture, only allows changes in train-specific parameters. For each model, Bayesian Optimization is performed using a Gaussian Process, as implemented in the scikit-optimize package [HKN+20]. The search space for each hyperparameter is given in model-specific sections. The optimization proceeded for 30 function evaluations, with 10 random initial points to ensure adequate exploration. The objective minimized was the validation mean squared error (NMSE) obtained from the respective workflow. All experiments are executed

[7]Each window originates again from the same period; *2008-01-01*.

with a fixed random seed (3178749) to ensure reproducibility, as this is the seed used in the entire methodology for reproducability.

3.7. Accuracy Metrics

Performance is quantified by normalized mean squared error (NMSE). This is the MSE normalized by the variance of the ground truth series. This normalization enables the comparison of error metrics across different data pairs. By normalizing the errors in this manner, the metric provides a variance-independent measure of predictive accuracy. Furthermore, the runs for all metrics reported in this study are executed using the seed (3178749).
All accuracy metrics are computed using the original, unprocessed form of the input data as ground truth. This is then compared with the model outputs that have been correspondingly processed to have the same scale. For validation, four time windows from 2016 through 2019 are gathered and cross-averaged, and time windows from 2020 through 2024 are gathered and cross-averaged for the test metrics using the optimized hyperparameters from the validation set.

4. Results

This chapter presents the empirical findings of predictive accuracy and profitability, evaluated using NMSE and Return score as introduced in subsection 3.5.3, for the baseline Kalman Filter, Transformer model and Time-MoE model. These models forecasted the spread of the top 10 cointegrated ETF pairs according to the Engle-Granger test [EG87] across five sliding test windows from 2020 through 2024, with results averaged across windows detailed in Table 4.1 and results per year detailed in Table 4.2. Hyperparameters, which are detailed in Table 4.3, are optimized using time-series cross validation across four sliding windows from 2016 through 2019. For each of the three models, identical preprocessing procedures (filtering, z-score normalization) were performed in accordance with the methodology described in chapter 3.

4.1. Pair Profitability

The brute-force search over an average of 101 sufficiently liquid NYSE/NASDAQ ETFs, totalling $\binom{101}{2} = 5050$ unique pairs, produced an average of 86 pairs if filtered for $p < 0.05$ using Engle-Granger [EG87]. Selecting the top 10 most cointegrated pairs per window resulted in a total of 50 pairs (from a total of 5 sliding test windows). Resulting p-values ranged between $p = 1.38 * 10^{-4}$ and $p = 5.3 * 10^{-3}$. Across 50 pairs (10 per window), 11 suffered total loss of equity ($TLOE$). The other 39 averaged a TR-PI of 5.80% YoY. The highest TR-PI was 43.99% YoY for SHV–PDP. Overall, 28 pairs resulted in a TR-PI between -5% and +5% YoY, and 11 exceeded +5% YoY. Figure 4.1 illustrates TR-PI further.

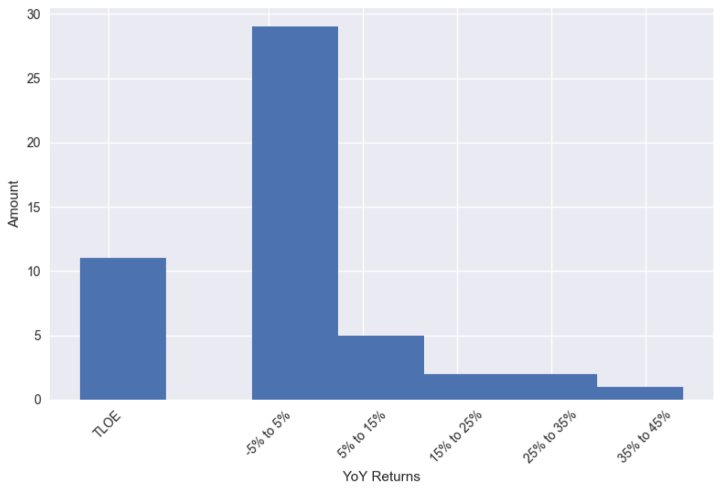


Figure 4.1.: *TR-PI* over All Test Windows

4.2. Kalman Filter

Bayesian optimization for 30 function evaluations[1] resulted in a reduction of 0.43 NMSE, with specific values detailed in subsubsection 3.4.1. Across periods, test MSE values range less than an order of magnitude from 8.63 to 47.45, with Kalman exhibiting the highest mean MSE among all models at 28.02. In the best-case scenario, the Kalman Filter was also the least accurate, with a Test MSE of 0.80. Return scores across test windows span the least wide range of all three models from 0.8667 to 1.006 with the highest mean Return Score of 0.9579.

4.3. Transformer

Bayesian optimization for 30 function evaluations resulted in a reduction of 2.42 NMSE, with specific values detailed in subsubsection 3.4.2. Across periods, test MSE values range almost two orders of magnitude from 0.17 to 8.64, scoring 2.24 in the average case. In its best case, the Transformer achieved a Test MSE of 0.024. In both cases it outperforms the Kalman filter but trails Time-MoE. Better performance was observed at the optimum than on average. Return scores across test windows span the widest spectrum from 0.531 to 1, with the lowest mean Return Score of 0.793.

4.4. Time-MoE

Bayesian optimization for 30 function evaluations resulted in a reduction of less than 0.01 NMSE, with specific values detailed in subsubsection 3.4.3. Across periods, Test MSE values range the most narrow of all models from 0.072 to 0.113, with a mean of 0.091. The lowest of all three models. In the best-case scenario, Time-MoE also scored the most accurate of all models with a Test MSE of 0.018. Return scores across test windows span a spectrum from 0.820 to 0.975, outperforming Transformer but trailing Kalman with a mean of 0.902.

[1]One function evaluation consists of four sliding windows, returning cross-window average validation MSE.

4.5. Overview Profitability & Predictability Averages Across Time Periods

	Kalman	**Transformer**	**Time-MoE**
Cross-Year Average Results			
Validation Metrics			
Validation MSE	42.49	0.5857	0.1818
Test Metrics			
Test MSE	28.0203	2.2436	0.0913
Return Score	0.9579	0.7932	0.9027
Overall Best-Performing Pair*			
Test Metrics			
Test MSE	0.80221	0.02447	0.0186
Returns YoY (%)	$1.58\% \pm 0.25\%$	$1.12\% \pm 0.02\%$	$0.87\% \pm 0.09\%$
TR-PI	0.53%	1.15%	0.88%
Return Score	1.01	1.0	1.0

Table 4.1.: Test metrics gathered and averaged across 5 testing windows (2020-2024). Validation metrics are cross-validated across four validation windows (2016-2019). *with *best-performing* defined as the lowest Test MSE. Taken across the top 10 pairs for each of the 5 testing windows.

4.6. Overview Profitability & Predictability Averages per Time Period

	Kalman	Transformer	Time-MoE
Average Validation Results per Year			
2016			
Validation MSE	2.9447	0.3326	0.30723
Test MSE	-	-	-
Return Score	-	-	-
2017			
Validation MSE	72.1505	0.3567	0.1530
Test MSE	-	-	-
Return Score	-	-	-
2018			
Validation MSE	57.9314	1.2846	0.0864
Test MSE	-	-	-
Return Score	-	-	-
2019			
Validation MSE	19.6823	0.3689	0.1805
Test MSE	-	-	-
Return Score	-	-	-
Average Cross-Validation MSE	42.49	0.5857	0.1818
Average Test Results per Year			
2020			
Validation MSE	-	-	-
Test MSE	8.6394	0.7749	0.0968
Return Score	0.8667	0.5311	0.8200
2021			
Validation MSE	-	-	-
Test MSE	26.3820	6.1121	0.0725
Return Score	0.9489	0.7289	0.9000
2022			
Validation MSE	-	-	-
Test MSE	32.7204	1.2895	0.0904
Return Score	1.0057	0.7143	0.9100
2023			
Validation MSE	-	-	-
Test MSE	24.9083	0.1694	0.0832
Return Score	0.9900	1.0000	0.9750
2024			
Validation MSE	-	-	-
Test MSE	47.4512	2.8719	0.1138
Return Score	0.9783	0.9917	0.9083
Cross-Year Average Test MSE	28.0203	2.2436	0.0913
Cross-Year Average Return Score	0.9579	0.7932	0.9027

Table 4.2.: Average results per year.
Pairwise results that result in the per year averages are explicitly given in the appendix in section A.1.
Validation metrics are cross-validated across four validation windows (2016-2019).
Test metrics gathered and averaged across 5 testing windows (2020-2024).

4.7. Hyperparameter Optimization Results

	Kalman	**Transformer**	**Time-MoE**
Classical Hyperparameters			
δ	0.01	–	–
Observation Covariance For Regression	2.56	–	–
Transition Covariance For Averaging	0.10	–	–
Observation Covariance For Averaging	3.48	–	–
Deep Learning Hyperparameters			
Architecture			
Model Depth	–	256	–
Amount of Heads	–	8	–
Number of Layers	–	3	–
Dropout	–	0.14	–
Training Algorithm			
Learning Rate	–	$2.43 * 10^{-5}$	$2.05 * 10^{-6}$
Minimum Learning Rate	–	8.42^{-5}	$2.61 * 10^{-6}$
Warmup Ratio	–	0.01	0.05
Weight Decay	–	0.28	0.24
Batch Size	–	64	128
Optimizer-specific			
β_1	–	0.96	0.88
β_2	–	0.97	0.98
ϵ	–	$5.23 * 10^{-10}$	$2.74 * 10^{-10}$
Average Cross-Validation MSE	42.49	0.5857	0.1818

Table 4.3.: Optimized hyperparameters using bayesian optimization cross-validated across four windows (2016-2019) as described in section 3.6.
All hyperparameters rounded to two decimals.

4.8. Visual Results of Predicted Spreads

To enhance the interpretability of forecasted time series and their error characteristics, this section visualizes the predicted and actual spread time series for the best-performing asset pair identified by each model. Following Table 4.1, best-performing is defined as the pair with the lowest Test MSE over all 50 pairs from the 5 test windows. Notable deviations may reveal unique error structures, indicating strengths and weaknesses inherent to each model, implications of which are further discussed in section 5.1.

4.8.1. Kalman Filter

The Kalman Filter's predictions least closely track the actual spread for the best-performing pair, which is in accordance with the highest test MSE of 0.8022. The Kalman Filter forecast timeseries is able to capture the approximate directions of the actual spread, while its largest deviations occur due to periods of rapid change or its unability to follow small oscillations. This error structure is similar to the smoothed window illustrated in Figure B.6.

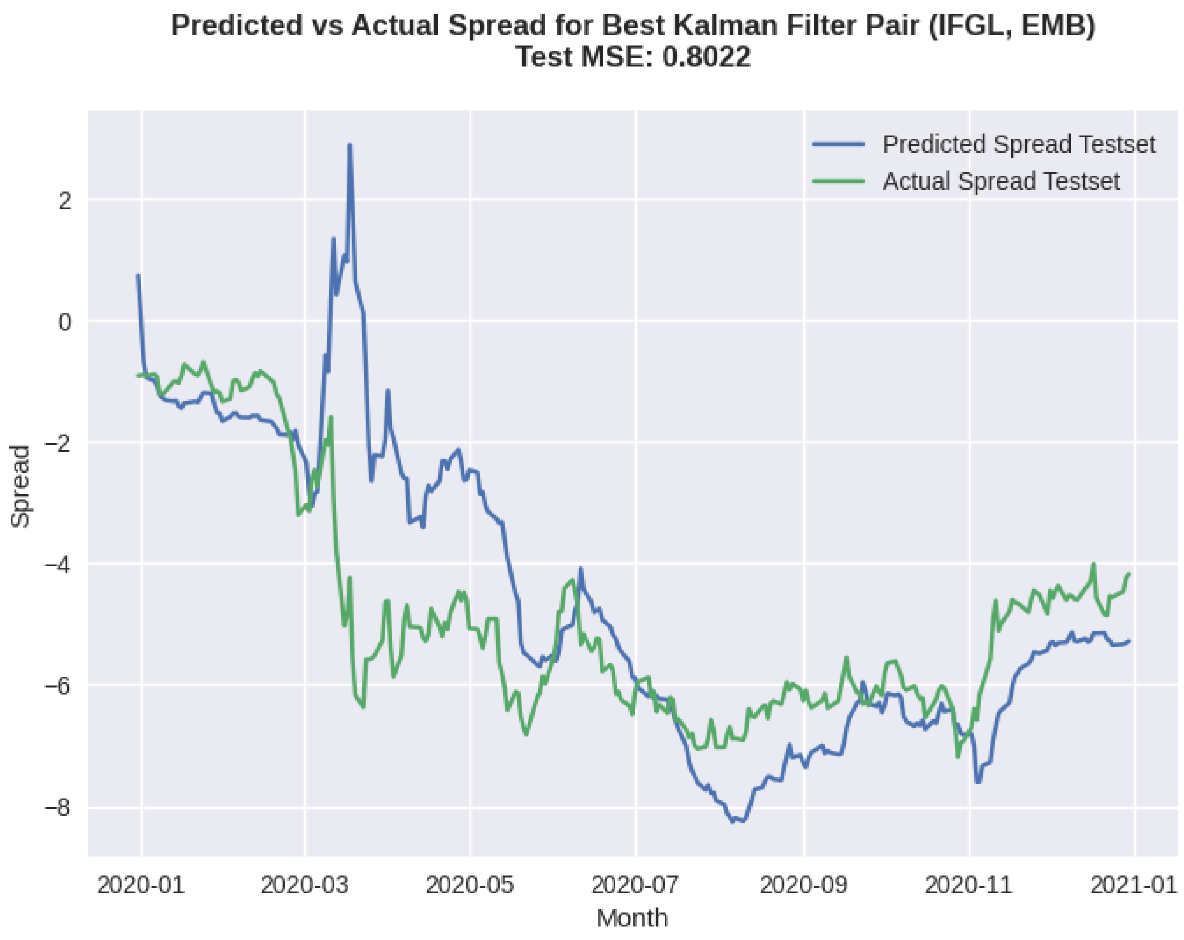


Figure 4.2.: Predicted vs Actual Spread for Kalman Filter's Best Pair

4.8.2. Transformer

The Transformer model demonstrates stronger alignment with the actual spread, capturing both trend and volatility, in accordance with the lower forecast error compared to Kalman Filter. Residual differences between the forecast and ground truth values remain discernible.

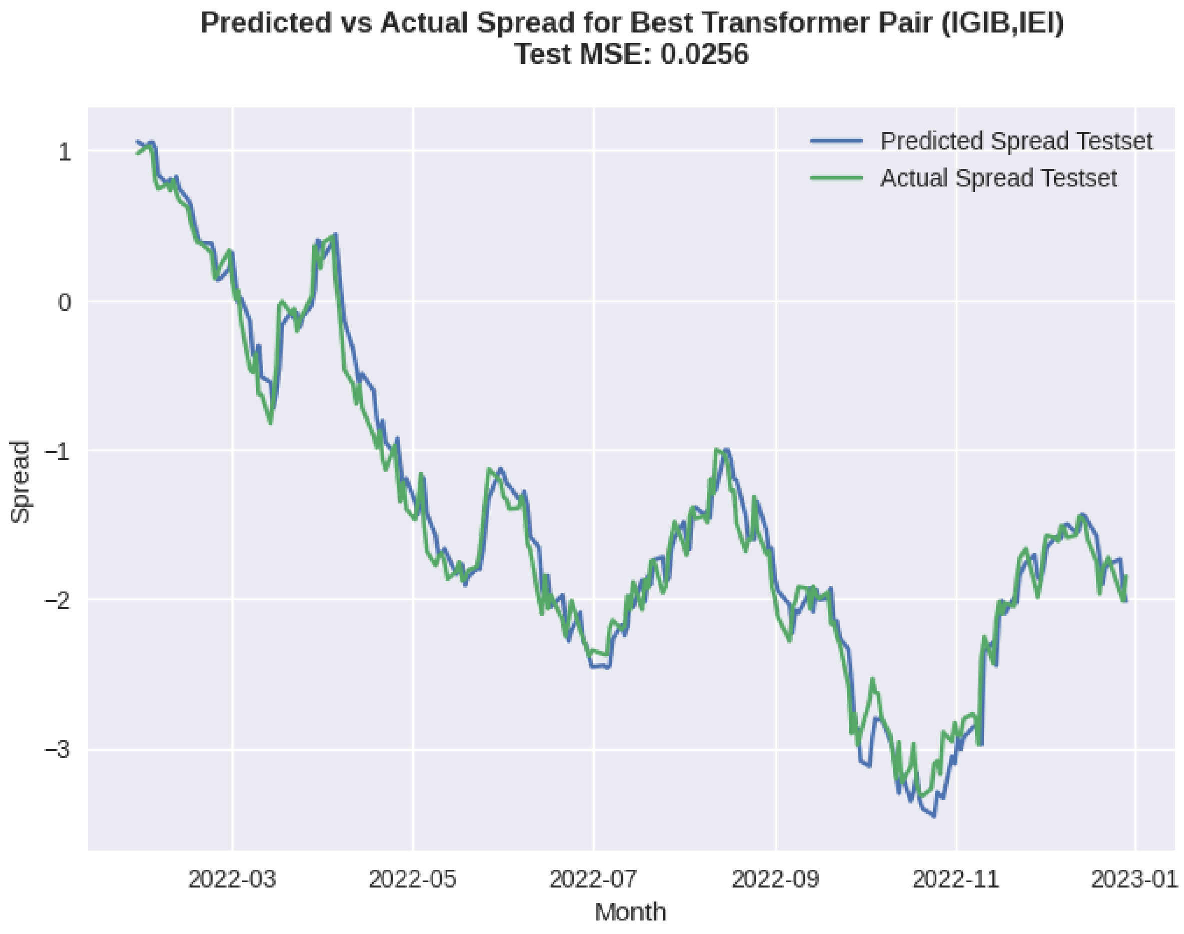


Figure 4.3.: Predicted vs Actual Spread for Transformer's Best Pair

4.8.3. Time-MoE

The Time-MoE model demonstrates stronger alignment with the actual spread compared to Transformer's spread. Residual differences between the predicted and ground truth values are less discernible than for Transformer. In the best-case, the error characteristics of the forecasts are the same as the Transformer's discernible error characteristics, not being caused by scaling inaccuracies or an error structure approximable by smoothing ground truth values, as is partly the case for the Kalman Filter.

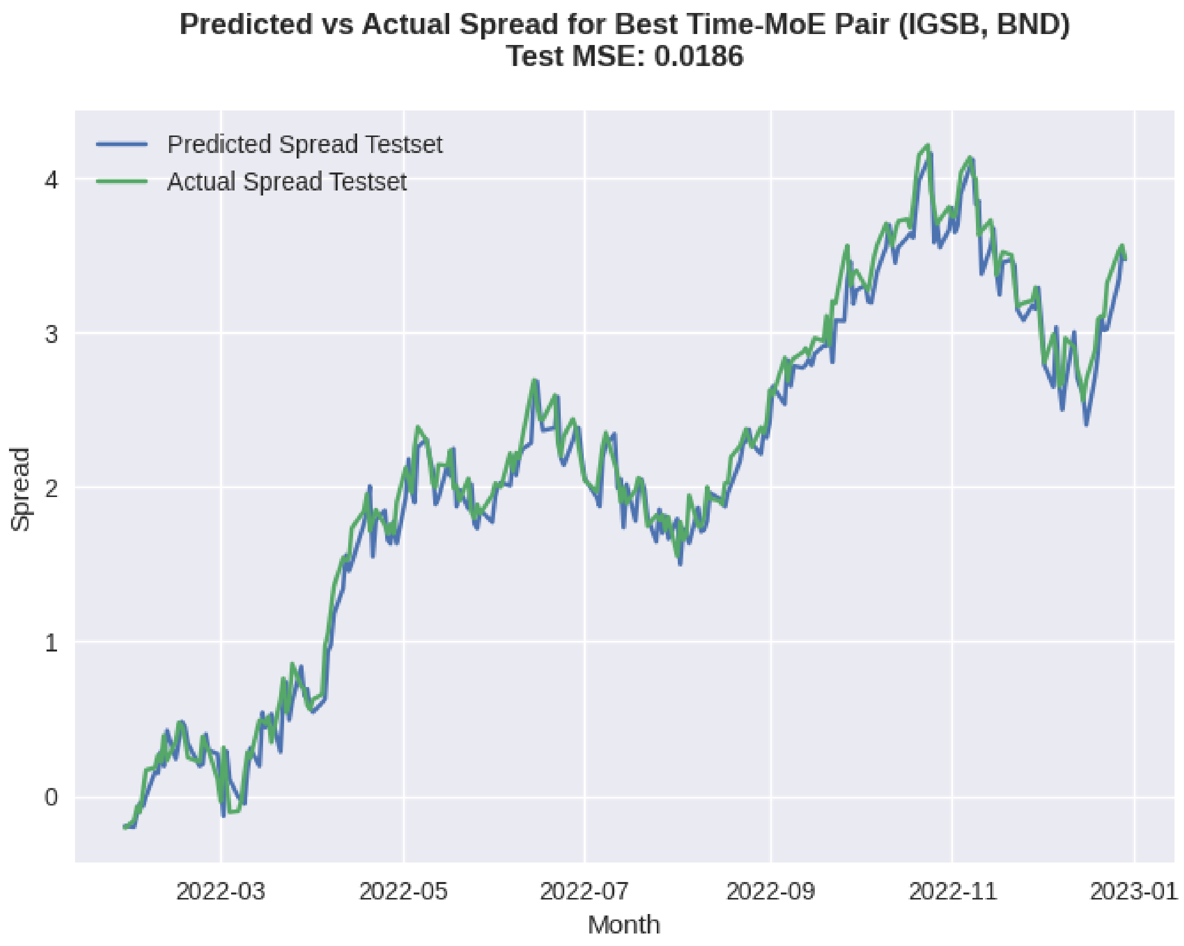


Figure 4.4.: Predicted vs Actual Spread for Time-MoE's Best Pair

4.8.4. Interpretation Clarification

Showing these graphs as predicted vs actual spread might misleadingly imply the entire blue graph is predicted using multi-step forecasting. However, as mentioned in the methodology in section 3.4, single-step forecasting is applied. A multi-step forecasting approach would be to apply one model to predict the value for the next time step, using the existing predicted value as an input to forecast the value for the next time step [WBB+22]. As only ground truth values are used to predict a day ahead, the current approach is single-step forecasting. Therefore, the graph in Figure 3.3 is deemed important context for interpreting these three graphs, as it illustrates how a single prediction is made with data points of length ℓ, the size of the look-back window.

5. Discussion & Conclusion

5.1. Discussion

Building on the empirical results presented in the previous chapter, this discussion interprets how the observed predictive and economic performance of the Kalman Filter, Transformer and Time-MoE model align with expectations based on existing theory. I highlight the methodological choices such as cointegration-based pair selection and time rolling window cross-averaging test metrics that most strongly influenced the findings. Finally, I examine the practical implications and limitations of the work, outlining avenues for further research.
Supplementary findings are delineated in Appendix B. In that section, their significance and characteristics are discussed in the conventional format of a results section. Its implications are discussed in the current discussion section.

5.1.1. Research Questions & Hypotheses

Sub-Research Question 1: Pair Discovery

Out of all sufficiently liquid ETFs listed on the NASDAQ and NSYE, an average of 86 per time window passed the Engle-Granger test out of an average of 5050 pairs in total. This provided a set of pairs large enough to proceed further in the pipeline to address subsequent research questions. The least cointegrated pair of the top 10 pairs examined scored a value of $5.3 * 10^{-3}$. This is an order of magnitude more signifact than the threshold of $p < 0.05$. Therefore, any issues in predictability or profitability are unlikely to be caused by insufficiently cointegrated pairs. Furthermore, the chosen combination of this pair selection method and trading strategy yield outcomes consistent with the premise of profitability. With an average TR-PI of 5.80% YoY when excluding pairs resulting in Total Loss of Equity (TLOE), it achieves a percentage similar to returns found in other literature (see subsection 2.2.5). Out of 50 pairs (10 across the 5 sliding windows), 11 resulted in TLOE. Table B.1 shows the difference per window, with 1 of the top 10 cointegrated pairs resulting in total loss of equity in 2020, a difference from 4 out of 10 in 2024. Figure B.2a and Figure B.2b demonstrate that selecting 2007 rather than 2008 as the first year of a train window materially alters data availability. A substantial number of ETFs had either not yet been launched or lacked sufficient trading volume in 2007, highlighting the necessity for the methodology to take into account this significant surge in availability of ETFs starting from 2008. Subsequent studies should equally take note of this. It also suggests a constraint in the models' further outcomes: even with perfect predictive accuracy, the proportion of profitable strategies is constrained to 22% by the underlying properties of the pairs identified via the Engle-Granger method. Consequently, the initial assumption that higher predictive accuracy would directly lead to proportionally higher profitability appears less tenable, as perfect accuracy does not invariably translate into profitability.

Sub-Research Question 2: Prediction Accuracy

Cross-averaged over sliding test windows, the Time-MoE model attained most accurate forecast of the ETF pairs' price spread. In the best-case scenario, Time-MoE also achieved the lowest Test MSE among all models. Moreover, the model empirically demonstrates to be more robust across time periods by having the tightest MSE range. Figure B.4a illustrates some observed inconsistency between validation and test MSE is caused by inaccurate scaling. With a larger inconsistency shown in Figure B.4b. From this difference, it can be inferred that cross-averaging over windows mitigates the inconsistency between validation MSE and test MSE. However, it does not completely get rid of it. The Transformer model demonstrated superior forecasting accuracy relative to the Kalman baseline but attained lower accuracy than that achieved by Time-MoE, in the per-year, cross-year average, and best case. Furthermore, the Transformer exhibited the highest variability in test MSE suggesting susceptibility to poor performance based on time periods. These results indicate it is the least robust to regime shifts compared to its best-case. The baseline Kalman Filter achieved less accurate results than both Transformer and Time-MoE. As seen in Figure 4.2, it serves well as a baseline for prediction, following the general trend with one-day-ahead predictions. It shows less variability in prediction accuracy to Transformer, implying similar but more robustness to regime shifts. Overall, the results reveal a consistent ordering across all evaluation metrics, including per-year average, cross-year average, and best-case scenario. The empirical results reveal outperformance of deep learning architectures compared to the baseline model. subsection 5.1.2 examines the observed trend sensitivity by the Transformer model and the Kalman Filter in further detail.

Sub-Research Question 3: Profitability

Cross-averaged over sliding test windows from 2020 through 2024, the baseline model Kalman Filter yielded the highest average Return Score of 0.9579 according to the back-tested simulations. The original assumption was that profitability scales in proportion to prediction accuracy. Earlier results (subsubsection 5.1.1) have previously suggested that this assumption is less tenable. Finally, this assumption is empirically observed to be false, reflected in the profitability outcomes. Accuracy is decoupled from profitability due to some underlying reason. The most accurate model Time-MoE scored a lower mean than the Kalman Filter. Transformer ranked the lowest. Nevertheless, a consistent ordering is shown. Per-year averages and cross-year averages all show the same ordering. Only with Kalman Filter ranking most profitable and Time-MoE second, the ordering is different from the one expected due to predictability. Transformer consistently delivered the lowest Return Score across the averages of all five back-tested time frames. To elucidate the observed decoupling between accuracy and profitability, the structure of forecast errors is inspected in the following section.

Predictive Accuracy vs. Economic Performance

The inverted logarithmic relationship between forecast error (artificially caused by increased noise levels) and returns as seen in the Gaussian plot in Figure B.5 initially supports the assumption that accuracy is a primary driver of profitability in z-score trading strategies. However, the error-returns relationship shows other relationships when simulated differently. These all provide insight into the missing translation of MSE into profitability. First, a smoother prediction causes more inaccurate forecasts reflected in higher MSE, but more profitable trading. This is likely because the ability to capture z-score breaches is not affected by this smoothing. Second, higher scaling is observed to improve profitability in spite of increasing error metrics. This originates from specific mechanics in the trading strategy. Position sizing is related to the size of the spread, such that greater absolute spread values produce proportionally larger trades, while diminished spreads yield smaller position sizes. Consequently, any amplification of the spread scale magnifies the trade size. Therefore, any ampliciation in scale amplifies profitability (and conversely amplifies losses in an unprofitable case). Thus, scaling changes result in the scaling of profitability, causing decoupling between relationship of MSE vs returns. When considered alongside the finding that smoothed versions of ground truth do not diminish resulting z-score profitability, the observation that Kalman Filters yield lower predictive accuracy yet still produce profitable simulation outcomes relative to deep learning architectures becomes understandable, rendering original assumptions false.

5.1.2. Limitations

Trend Sensitivity

The wider accuracy range observed in the Transformer and Kalman Filter is partly caused by inaccurate scaling. The current methodology calculates the mean and standard used for z-scale normalization from a previous window. In Figure B.3, this window is labelled as "Groundtruth Spread Previous Window". The figure shows such an an example of larger predictive error being caused by inaccurate scaling. Certain models can correct for this trend sensitivity better than others. Future research can improve predictive accuracy by normalization methodologies less sensitive to trends. Examples of this are correcting for trends in the input time series, using normalizations more temporally proximate to tested windows, or using preprocessing techniques other than normalization that rely less on trends.
Despite a thorough understanding of its underlying causes, the trend sensitivity remains a limitation in the predictive performance of models examined in this study. Time-sensitive models will stay inherently susceptible to performance degradation in the presence of trend differences. As the most reliable alternative, this study provides empirical evidence supporting the effectiveness of a model with low temporal sensitivity, *Time-MoE*, to address the limitation most effectively compared to the other models.

Predictive Accuracy Metrics

Although analyses like subsubsection 5.1.1 go further into depth about specifics of the errors and suspected error structures, a more accurate examination would have been possible if more forecasting error metrics besides NMSE were used. For example, scale-invariant measures such as Mean Absolute Percentage Error (MAPE) could have allowed for better analysis of systematic scaling inaccuracies in the predicted spread amplitudes, and Quasi-Likelihood Loss could have allowed for better analysis of heteroskedasticity in the forecast errors [CGR18], as well as being in line with other studies on financial time series.

Profitability Metrics

The research question focuses on improving price spread prediction through machine learning techniques. As discussed in section 2.6, the analysis of profitability metrics then serves as a complementary means of evaluating model performance, offering an additional perspective. Therefore, emphasis is placed on relative performance under a consistent trading framework, rather than on the absolute level profitability achieved, resulting in an emphasis on *Return Score* rather than YoY returns. While the trading strategy was not examined to understand how well it performs relative to other possible strategies, it nevertheless serves to offer an insightful additional dimension to view the model's performance. Consequently, the predictive accuracy metrics in this report are presented with more confidence than the profitability metrics. Reflecting the profitability metrics' dependence on secondary factors such as transaction costs, strategy design, and the fact that trading simulations are conducted subsequent to the predictive modeling phase within the workflow. Overall, this step in the pipeline incorporates a broader set of assumptions and dependencies than the predictability.

5.2. Conclusions

In conclusion, to answer the research question, sufficienty predictable ETFs were found using the Engle-Granger test [EG87]. This allowed Time-MoE to achieve the most accurate forecasts of all examined models. Using the forecasts, the Kalman Filter resulted in the most profitable back-tested simulations. The current empirical results reinforce and extend the evidence for deep learning-based architectures in finance. The use of Transformer was recommended for better results in the end of Daniels' work (as an improvement to LSTM models) [Dan22]. This study built further on that concept, implementing Transformer and Time-MoE by using a foundational component of the methodology introduced by Kaur [Kau18]. This component is adapted to the methodological context of the current study, while also eliminating poor procedures such as the introduction of look-ahead bias. This finding of superior accuracy challenges findings by Zhao et al. that Time-MoE would struggle with short-term price fluctuations [ZWZ+25]. The finding that the highest attainable profit, TR-PI, can still probabilistically result in a Total Loss Of Equity (TLOE) confirms that a statistically significant cointegration relationship is a *necessary* but not a *sufficient* condition for profitability. Using five non-overlapping sliding test windows, the Time-MoE model consistently achieved the lowest NMSE (0.0725–0.1138 across sliding windows, with an average NMSE of 0.0913), confirming the strength of its sparse architecture followed by a pre-training and finetuning approach. Cross-validation demonstrated that the MoE's accuracy exhibited minimal sensitivity to the choice of hyperparameters. The Transformer model ranked second in accuracy with an average of 2.2436, and last in profitability with an average Return Score of 0.7932. Its error variance was an order of magnitude larger than Time-MoE's, indicating trend sensitivity to regime shifts. Kalman Filter regression, constrained by linear-Gaussian assumptions, lagged behind in accuracy with an average MSE of 28.0203 and under-reacted to rapid changes in spread. Kalman's outperformance in terms of profitability appears to stem from residual error structures that better accommodate a z-score trading framework despite its lower accuracy when measured with MSE. Lastly, due to a difference in the examined model architectures, the results do not yet specifically highlight whether pre-training, sparse activation, or a combination of the two is responsible for Time-MoE's most accurate forecasts.

5.3. Recommendations

5.3.1. Future Work

After applying Transformers as recommended by Daniels [Dan22] and Time-MoE for its pre-trained and sparse architecture to a research context introduced by Kaur [Kau18], I anticipate that it is most promising not to apply other models to the same context, but rather applying the strengths exhibited by new underexplored models to novel contexts. This study's central takeaway for future work is that the pre-trained and sparse Time-MoE architecture can *quickly* capture *complex* dependencies in time series data. This could be applied most promisingly to financial time series in three ways.

Financial Time Series

Firstly, the model's ability to accurately predict prices with small data budgets positions it for use in financial time series prediction in contexts of crisis, where a small data budget is a definitive limitation. For example, few-shot financial forecasting has been proposed to be applied in the advent to the economic crisis of 2020 following the the COVID-19 pandemic [WKRZ23]. The newly found accurate Time-MoE model could prove to be one of the models well adapted to quickly adapt to new market conditions in a benchmarking study compared to many other models. This could prove a great application of online inferencing where the model runs continuously [CGK+24].

Secondly, it could be applied promisingly in an area where the shorter training and inference times that were observed in the model can be especially useful; High-Frequency Trading (HFT) represents a significant context for the application of Pairs Trading. The sparse and pre-trained nature of Time-MoE demonstrates considerable potential in time windows that are orders of magnitude smaller than the current feature selection of per-day closing prices. Compared to dense architectures trained from scratch with randomly initialized weights, the sparse pre-trained nature of Time-MoE grants it rapid inference capability[1] combined with enhanced predictive accuracy relative to all other models as empirically found in this study. This indicates suitability for deployment within HFT contexts. The main remaining task would be a rigorous examination of the underlying theoretical assumptions that may differ from the current, whereas the required methodological adjustments are minimal. As mentioned in subsection 3.2.1, the current pipeline is readily adaptable to train and inference models on different time scales.
Thirdly, The application of both Transformers and MoE-architectures applied to a specific quasi-multivariate framework, called *ETF arbitrage*[2], is a promising context. A multivariate framework in PT, as opposed to a univariate framework, is when one security is traded against a weighted portfolio of comoving securities, rather than separate securities being traded [Kra16]. Quasi-multivariate refers to only one of the two in a pair consisting of a group of assets. *Multivariate PT* is not to be confused with a multivariate time series, which means a timeseries constituting multiple dimensions. Importantly, Figueira et al. chose to feed their ML model a multivariate timeseries as feature input to machine learning models and reported that this yielded superior predictive accuracy [FH22]. The biggest indicator for this promising context is that in spite of the deep learning architectures' outperformance in terms of predictive accuracy, their maximum ability of capturing complex interdependent relationships has not been tested to its maximum. Using more complex multivariate features, as suggested by Figueira et al., with features such as the ETF's market price, the Net Asset Value (NAV) of underlying holdings, and premiums or discounts at which it is trading as an input to predict expected discrepancies could prove to use the strengths of deep learning architectures to a larger extent, further utilizing its superior complex predictive abilities over classical statistical models. Moreover, despite these anticipated advantages, ETF arbitrage is less saturated regarding the application of Machine Learning models, according to publicly available research, with mainly use classical arbitrage techniques [BDR20] [PZ17b] [MNV10]. However, techniques have been applied in recent years to my best knowledge, such as genetic algorithms [SKCS+25], or Random Forests and Gradient Boosting Decision Trees [CLLK23].

General Time Series

While the present study focuses on financial data, the empirical advantages of the sparse, pre-trained Time-MoE architecture (rapid fine-tuning, efficient inference, and an ability to model long-range, non-linear dependencies with limited task-specific data) extend naturally to other domains that share analogous statistical characteristics.
For example, electric-power systems require accurate forecasts at 15-minute to hourly horizons for unit commitment and demand-response scheduling. This lends such energy demand forecasting well to the empirically found advantages. These series exhibit pronounced daily and annual seasonality through weather-dependent regime shifts. Future work could provide insights into whether the model responds better to such seasonal trends compared to classical model or models.

[1]Specifically, training on a single time series using an NVIDIA A100 GPU required tens of seconds, with inference times taking a few seconds. While these figures are not definitive benchmarks, they provide a practical approximation of feasibility.
[2]Not to be confused with regular statistical arbitrage methods such as pair trading *applied to* ETFs. ETF Arbitrage entails finding price irregularities between an ETFs Net Asset Value (NAV) and its current trading price [PZ17a].

Moreover, Electrocardiogram (ECG) recordings are high-frequency signals where periodicity is of greater importance than both contexts mentioned before. The ability of Time-MoE to capture both fine-grained morphology and long-range contextual information makes it a strong candidate for anomaly detection in that context.
Building on the cross-benchmark analysis presented in the original study [SWN+25], investigations within individual domains are warranted to examine the model's domain-specific behavior and to facilitate the exchange of insights across these domains.

5.3.2. Amendments to Current Methodology

Besides novel contexts offering promising applications, amendments to the current methodology could also provide more insights. As detailed in subsubsection 5.1.1, reliable data on exchange-traded funds (ETFs) are sparse before 2008. Accordingly, future studies should explicitly assess whether their research questions can be addressed using the shorter, modern sample interval for which comprehensive ETF coverage *is* available.
The principal finding of this study can be summarized as follows. Time-MoE and Transformer produced the most accurate models, but the z-score based simulation strategy failed to reflect this performance. As this discrepancy has been discussed to be likely caused by error structures and probabilistic outcomes of trading simulations, future research could examine the input of accurate forecasts into more simulations that are 1) more deterministic and 2) decoupled from raw signal magnitude. Examples include altered versions of the current strategy decoupled from raw signal magnitude, or fixed-threshold Ornstein–Uhlenbeck entry/exit rules as mentioned in subsection 2.3.4. Testing accurate forecasts within these more deterministic, magnitude-decoupled frameworks should reveal a more easily interpretable link between forecast accuracy and simulated trading performance. As a further recommendation, future research could adopt methodologies less sensitive to trends, such as correcting for trends using detrending procedure or anchoring z-score normalizations to sliding reference periods that are shorter and therefore exhibit statistics closer to those of the evaluation window. Moreover, research could also explore preprocessing strategies that do not hinge on trend stationarity as z-score normalization does, such as seasonal–trend decomposition or using more robust versions of scaling using median or inter-quartile ranges, ensuring that inferential results remain robust to evolving baseline dynamics.

Bibliography

[Alo20] Monira Essa Aloud. The role of attribute selection in deep anns learning framework for high-frequency financial trading. *Intelligent Systems in Accounting, Finance and Management*, 2020. First published: 12 March 2020.

[Aro19] Ran Aroussi. yfinance: Download market data from yahoo! finance's api. https://github.com/ranaroussi/yfinance, 2019. Accessed: 2025-03-31.

[AST$^+$24] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[BDR20] David C Brown, Shaun William Davies, and Matthew C Ringgenberg. Etf arbitrage, non-fundamental demand, and return predictability*. *Review of Finance*, 25(4):937–972, 10 2020.

[Bol92] John Bollinger. Using bollinger bands. *Stocks & Commodities*, 10(2):47–51, 1992.

[Boo06] Richard Bookstaber. *A Demon of Our Own Design.* Wiley, 2006.

[C$^+$10] Jesse Cooke et al. pykalman: Kalman filter, smoother, and em algorithm for python. https://github.com/pykalman/pykalman, 2010. Accessed: 2025-06-11.

[CCD$^+$23] Tianlong Chen, Xuxi Chen, Xianzhi Du, Abdullah Rashwan, Fan Yang, Huizhong Chen, Zhangyang Wang, and Yeqing Li. Adamv-moe: Adaptive multi-task vision mixture-of-experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17346–17357, October 2023.

[CGK$^+$24] Lucas Correia, Jan-Christoph Goos, Philipp Klein, Thomas Bäck, and Anna V Kononova. Online model-based anomaly detection in multivariate time series: Taxonomy, survey, research challenges and future directions. *Engineering Applications of Artificial Intelligence*, 138:109323, 2024.

[CGR18] Leopoldo Catania, Stefano Grassi, and Francesco Ravazzolo. Predicting the volatility of cryptocurrency time-series. *Mathematical and Statistical Methods for Actuarial Sciences and Finance: MAF 2018*, pages 203–207, 2018.

[Cha13] Ernest P. Chan. *Algorithmic Trading: Winning Strategies and Their Rationale*, volume 625. John Wiley & Sons, 2013.

[CLLK23] Jinhyung Eric Cho, GUN HEE LEE, Woneung Lee, and Bongjun Kim. Machine learning approach for predicting us etfs' tracking errors–implications on us invested fund. *Machine Learning Approach for Predicting US ETFs' Tracking Errors–Implications on US Invested Fund (October 16, 2023)*, 2023.

[Dan22] Prateek Samuel Daniels. *Machine Learning Techniques for Pricing, Hedging and Statistical Arbitrage in Finance.* PhD thesis, University of Technology Sydney, Faculty of Business, 2022. Under the supervision of Assoc. Prof Christina Skiibosios Nikitopoulos, Dr Otto Konstandatos, Prof. Xue-Zhong (Tony) He, Dr Mesias Alfeus.

[DF10] Bailey Do and Robert Faff. Does simple pairs trading still work? *Financial Analysts Journal*, 66(4):83–95, 2010.

[EG87] Robert F. Engle and Clive W. J. Granger. Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–276, 1987.

[EVDHM05] Robert J Elliott, John Van Der Hoek*, and William P Malcolm. Pairs trading. *Quantitative Finance*, 5(3):271–276, 2005.

[Fam70] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

[FH22] Miguel Figueira and Nuno Horta. Machine learning-based pairs trading strategy with multivariate. *Available at SSRN 4295303*, 2022.

[FZS22] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. In *Transactions on Machine Learning Research (TMLR)*, 2022.

[GGR06] Evan Gatev, William N. Goetzmann, and K. Geert Rouwenhorst. Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3):797–827, 02 2006.

[GOPZ21] Jorge Guijarro-Ordonez, Markus Pelger, and Greg Zanotti. Deep learning statistical arbitrage. *SSRN Electronic Journal*, 2021.

[GW24] Przemysław Gradzki and Piotr Wójcik. Is attention all you need for intraday forex trading? *Expert Systems*, 41(2), 2024.

[HH12] Thomas A Hanson and Joshua Hall. Statistical arbitrage trading strategies and high frequency trading. *Available at SSRN 2147012*, 2012.

[HHLS24] Eli Hadad, Sohail Hodarkar, Beakal Lemeneh, and Dennis Shasha. Machine learning-enhanced pairs trading. *Forecasting*, 6(2):434–455, 2024.

[HKN$^+$20] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize, September 2020.

[HMP$^+$25] Mahmudul Hasan, Zannatul Mifta, Sumaiya Janefar Papiya, Paromita Roy, Pronay Dey, Nafisa Atia Salsabil, Nahid-Ur-Rahman Chowdhury, and Omar Farrok. A state-of-the-art comparative review of load forecasting methods: Characteristics, perspectives, and applications. *Energy Conversion and Management: X*, page 100922, 2025.

[Hu21] Xiaokang Hu. Stock price prediction based on temporal fusion transformer. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 60–66, 2021.

[Kau18] Simerjot Kaur. Quantitative trading strategies using deep learning: Pairs trading. *CS230*, 2018.

[Kra16] Christopher Krauss. Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys*, 31(2):513–545, May 2016.

[LALP21] Bryan Lim, Sercan O Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[LJF+25] Jia Liu, Cheng Jinguo, Xia Fang, Zhenyuan Ma, and Yuankai Wu. Evaluating temporal plasticity in foundation time series models for incremental fine-tuning, 2025.

[LLW+24] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024.

[LM17] Jim Kyung-Soo Liew and Boris Mayster. Forecasting etfs with machine learning algorithms. *The Journal of Alternative Investments*, 20(3):58–78, 2017.

[LZL+24] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. *arXiv preprint arXiv:2402.02368*, 2024.

[Mac94] James G. MacKinnon. Approximate asymptotic distribution functions for unit-root and cointegration tests. *Journal of Business & Economic Statistics*, 12(2):167–176, 1994.

[MNV10] Ben R Marshall, Nhut H Nguyen, and Nuttawat Visaltanachoti. Etf arbitrage. *Massey University*, 2010.

[MPV05] Matthew Q. McPherson, Joseph Palardy, and Jon Vilasuso. Are international stock returns predictable?: An application of spectral shape tests corrected for heteroskedasticity. *Journal of Economics and Business*, 57(2):103–118, 2005.

[MRD+24] Amit Milstein, Guy Revach, Haoran Deng, Hai Morgenstern, and Nir Shlezinger. Neural augmented kalman filtering with bollinger bands for pairs trading. *IEEE Transactions on Signal Processing*, 2024.

[NH13] Seyed Taghi Akhavan Niaki and Saeid Hoseinzade. Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1), 2013.

[OCS+23] Oleksiy Ostapenko, Lucas Caccia, Zhan Su, Nicolas Le Roux, Laurent Charlin, and Alessandro Sordoni. A case study of instruction tuning with mixture of parameter-efficient experts. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[PHL24] Weiying Ping, Yuwen Hu, and Liangqing Luo. Price forecast of treasury bond market yield: Optimize method based on deep learning model. *IEEE Access*, 12:194521–194539, 2024.

[PZ17a] Kevin Pan and Yao Zeng. Etf arbitrage under liquidity mismatch. *SSRN Electronic Journal*, 2017.

[PZ17b] Kevin Pan and Yao Zeng. Etf arbitrage under liquidity mismatch. Working Paper 59, European Systemic Risk Board, Frankfurt am Main, Germany, December 2017. Also circulated as ESRB Working Paper No. 2017/59.

[SAD+23] Md Rasel Sarkar, Sreenatha G Anavatti, Tanmoy Dam, Mahardhika Pratama, and Berlian Al Kindhi. Enhancing wind power forecast precision via multi-head attention transformer: An investigation on single-step and multi-step forecasting. In *2023 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2023.

[SH20] Simão Moraes Sarmento and Nuno Horta. Enhancing a pairs trading strategy with the application of machine learning. *Expert Systems with Applications*, 158:113490, 2020.

[Shu23] Alexander Shulzhenko. Copula-based deviation measure of cointegrated financial assets. *arXiv preprint arXiv:2312.02081*, 2023.

[SK21] Jaideep Singh and Matloob Khushi. Feature learning for stock price prediction shows a significant role of analyst rating. *Applied System Innovation*, 4(1):17, March 2021.

[SKCS$^+$25] Jens Soerlie Kvaerner, Julio A Crego, Dag Einar Sommervoll, Aavald Sommervoll, and Niek Stevens. Genetic mimicking portfolios for etf arbitrage. *Available at SSRN 5302810*, 2025.

[SLP$^+$21] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[SMM$^+$17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

[SP10] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. *Proceedings of the 9th Python in Science Conference*, 57:92–96, 2010.

[SWN$^+$25] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations (ICLR)*, 2025.

[SWX$^+$23] Shuo Sun, Xinrun Wang, Wanqi Xue, Xiaoxuan Lou, and Bo An. Mastering stock markets with efficient mixture of diversified trading experts. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, page 11 pages. ACM, August 2023.

[tH22] Jonas ter Horst. Machine learning for exchange traded fund price predictions. *TScIT 37*, 37, July 2022. © 2022 University of Twente. Permission granted for personal and classroom use.

[Tha25] Hudson & Thames. The comprehensive introduction to pairs trading. https://hudsonthames.org/pairs-trading-introduction, 2025. Accessed: 2025-04-11.

[Tim25] Time-MoE (Xiaoming Shi et al.). Time-MoE: billion-scale time-series foundation models (official code). https://github.com/Time-MoE/Time-MoE, 2025. Accessed: 2025-06-30.

[UJ24] Jeffrey Uhlmann and Simon Julier. Gaussianity and the kalman filter: A simple yet complicated relationship. *arXiv preprint arXiv:2405.00058*, 2024.

[Val24] Diego Vallarino. A dynamic approach to stock price prediction: Comparing rnn and mixture of experts models across different volatility profiles. *Independent Research*, October 2024.

[Vid04] Ganapathy Vidyamurthy. *Pairs Trading: quantitative methods and analysis.* John Wiley & Sons, 2004.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[WBB$^+$22] Can Wang, Mitra Baratchi, Thomas Bäck, Holger H Hoos, Steffen Limmer, and Markus Olhofer. Towards time-series feature engineering in automated machine learning for multi-step-ahead forecasting. *Engineering Proceedings*, 18(1):17, 2022.

[WKRZ23] Kieran Wood, Samuel Kessler, Stephen J Roberts, and Stefan Zohren. Few-shot learning patterns in financial time-series for trend-following strategies. *arXiv preprint arXiv:2310.10500*, 2023.

[WZL$^+$25] Shiyu Wang, Xinyue Zhong, Jiawei Li, Rongwei Liu, Yidong Feng, Congcong Hu, Fan Huang, and Zhou Ye. Enhancing e-commerce supply chain management through large time series model. In *Companion Proceedings of the ACM Web Conference Workshop 2025 (WWW'25 AI4TS Workshop)*, pages 1–8, Sydney, NSW, Australia, April 2025. ACM.

[WZZ$^+$22] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. 2022.

[XDB24] Jue Xiao, Tingting Deng, and Shuochen Bi. Comparative analysis of lstm, gru, and transformer models for stock price prediction. In *Proceedings of the International Conference on Digital Economy, Blockchain and Artificial Intelligence*, DEBAI '24, page 103–108, New York, NY, USA, 2024. Association for Computing Machinery.

[Yah25a] Yahoo Finance. Yahoo finance website. https://finance.yahoo.com, 2025. Accessed: 2025-03-31.

[Yah25b] Yahoo Finance. Yahoo finance website specific subpage: Etf screener, 2025. Accessed: 2025-03-31.

[Zhe21] Wenjian Zheng. Exchange-traded fund price prediction based on the deep learning model. In *2021 China Automation Congress (CAC)*, pages 7429–7434, 2021.

[Zhu24] Xuanchi Zhu. Examining pairs trading profitability. Senior Essay, Department of Economics, Yale University, April 2024.

[ZKS$^+$23] Zhen Zeng, Rachneet Kaur, Suchetha Siddagangappa, Saba Rahimi, Tucker Balch, and Manuela Veloso. Financial time series forecasting using cnn and transformer, 2023.

[ZM00] Paul Zarchan and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach.* American Institute of Aeronautics and Astronautics, Incorporated, 2000.

[ZWZ$^+$25] Chenyun Zhao, Xiaodong Wang, Anping Zhao, Yunpeng Cui, Ting Wang, Juan Liu, Ying Hou, Mo Wang, Li Chen, Huan Li, et al. A vegetable-price forecasting method based on mixture of experts. *Agriculture*, 15(2):162, 2025.

A. Appendix

A.1. Pairwise Results per Time Period

A.1.1. Kalman Filter

Pairwise Results Top 10 Cointegrated Pairs 2020

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (SHV,SMH)	2.46×10^{-4}	6.24861	$4.96\% \pm 0.35\%$	TLOE*	N/A
2. (SHV,ONEQ)	4.04×10^{-4}	7.31014	$3.23\% \pm 0.22\%$	33.16%	0.78
3. (SHV,PHO)	4.13×10^{-4}	8.91065	$1.14\% \pm 0.06\%$	14.67%	0.88
4. (SHV,PDP)	9.15×10^{-4}	7.17672	$3.39\% \pm 0.39\%$	43.99%	0.72
5. (DVY,PEY)	1.41×10^{-3}	22.46558	$0.03\% \pm 0.02\%$	0.10%	1.00
6. (PFF,EMB)	1.45×10^{-3}	10.39903	$1.36\% \pm 0.24\%$	0.71%	1.01
7. (IGSB,BND)	1.56×10^{-3}	3.46026	$0.10\% \pm 0.16\%$	-0.36%	1.00
8. (IFGL,SHV)	3.95×10^{-3}	4.17398	$-59.52\% \pm 49.03\%$	0.41%	0.40
9. (PRFZ,SCZ)	4.47×10^{-3}	15.44644	$0.76\% \pm 0.24\%$	0.31%	1.00
10. (IFGL,EMB)	4.61×10^{-3}	0.80221	$1.58\% \pm 0.25\%$	0.53%	1.01

Table A.1.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2021

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	7.71×10^{-4}	1.23317	$0.68\% \pm 0.07\%$	0.35%	1.00
2. (IFGL,EMB)	1.98×10^{-3}	6.62449	$0.74\% \pm 0.16\%$	-0.03%	1.01
3. (IFGL,SHV)	2.30×10^{-3}	4.47843	$1.17\% \pm 0.16\%$	0.32%	1.01
4. (IGSB,BND)	2.75×10^{-3}	31.00205	$0.32\% \pm 0.09\%$	0.19%	1.00
5. (IFGL,SOXX)	2.88×10^{-3}	23.39865	$3.17\% \pm 0.64\%$	23.38%	0.84
6. (IFGL,SMH)	2.91×10^{-3}	24.67326	$3.31\% \pm 0.77\%$	19.67%	0.86
7. (IFGL,PHO)	2.93×10^{-3}	34.76899	$1.62\% \pm 0.05\%$	12.06%	0.91
8. (IFGL,PDP)	2.97×10^{-3}	72.84399	$0.78\% \pm 0.06\%$	TLOE*	N/A
9. (IFGL,FTCS)	2.99×10^{-3}	44.07794	$1.71\% \pm 0.19\%$	12.47%	0.90
10. (IFGL,USIG)	3.01×10^{-3}	20.71949	$0.49\% \pm 0.08\%$	-0.11%	1.01

Table A.2.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2022

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	5.58×10^{-4}	20.34508	$4.49\% \pm 0.77\%$	0.61%	1.04
2. (IFGL,EMB)	1.40×10^{-3}	16.96470	$0.87\% \pm 0.04\%$	0.43%	1.00
3. (IGSB,BND)	1.89×10^{-3}	31.51380	$5.10\% \pm 0.26\%$	0.88%	1.04
4. (USIG,IEI)	2.78×10^{-3}	2.49067	$0.73\% \pm 0.14\%$	1.30%	0.99
5. (IGF,DVY)	2.82×10^{-3}	129.67840	$0.48\% \pm 0.11\%$	TLOE*	N/A
6. (DVY,PEY)	3.45×10^{-3}	3.14572	$0.18\% \pm 0.03\%$	0.27%	1.00
7. (IGIB,IEI)	4.50×10^{-3}	4.97859	$1.25\% \pm 0.27\%$	1.15%	1.00
8. (IFGL,SOXX)	4.65×10^{-3}	30.83984	$0.45\% \pm 0.06\%$	TLOE*	N/A
9. (IFGL,SMH)	4.65×10^{-3}	35.56703	$0.43\% \pm 0.06\%$	TLOE*	N/A
10. (IFGL,PHO)	5.30×10^{-3}	51.68065	$0.46\% \pm 0.10\%$	3.70%	0.97

Table A.3.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2023

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	2.82×10^{-4}	2.44415	$0.26\% \pm 0.04\%$	-0.06%	1.00
2. (IFGL,EMB)	7.70×10^{-4}	7.37119	$0.33\% \pm 0.07\%$	0.65%	1.00
3. (IGF,DVY)	1.17×10^{-3}	38.25796	$0.95\% \pm 0.04\%$	TLOE*	N/A
4. (IGIB,IEI)	1.25×10^{-3}	56.77418	$1.34\% \pm 0.22\%$	0.50%	1.01
5. (DVY,PEY)	1.57×10^{-3}	2.53301	$0.01\% \pm 0.01\%$	0.15%	1.00
6. (USIG,IEI)	1.67×10^{-3}	36.09742	$1.06\% \pm 0.14\%$	0.61%	1.00
7. (IFGL,BND)	1.93×10^{-3}	37.54439	$0.22\% \pm 0.02\%$	0.25%	1.00
8. (IFGL,SMH)	2.58×10^{-3}	9.56665	$4.55\% \pm 0.44\%$	16.05%	0.90
9. (IFGL,MBB)	2.63×10^{-3}	45.48907	$0.28\% \pm 0.04\%$	-0.27%	1.01
10. (IFGL,SOXX)	2.70×10^{-3}	13.00471	$3.08\% \pm 0.21\%$	TLOE*	N/A

Table A.4.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2024

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	1.38×10^{-4}	47.79868	$0.77\% \pm 0.13\%$	0.10%	1.01
2. (IFGL,MBB)	4.45×10^{-4}	55.19024	$1.20\% \pm 0.14\%$	TLOE*	N/A
3. (IFGL,EMB)	7.27×10^{-4}	2.23287	$1.09\% \pm 0.12\%$	0.39%	1.01
4. (IGIB,IEI)	7.97×10^{-4}	52.11113	$1.13\% \pm 0.15\%$	0.05%	1.01
5. (IGF,DVY)	8.40×10^{-4}	55.44654	$1.55\% \pm 0.21\%$	TLOE*	N/A
6. (USIG,IEI)	1.09×10^{-3}	141.67952	$0.74\% \pm 0.16\%$	-0.25%	1.01
7. (IFGL,BND)	1.20×10^{-3}	56.49230	$0.79\% \pm 0.07\%$	TLOE*	N/A
8. (IFGL,SMH)	1.64×10^{-3}	16.84568	$11.50\% \pm 0.67\%$	26.74%	0.88
9. (IFGL,SOXX)	1.82×10^{-3}	35.82744	$6.31\% \pm 0.64\%$	11.34%	0.95
10. (IGF,PPH)	2.29×10^{-3}	10.88732	$0.78\% \pm 0.07\%$	TLOE*	N/A

Table A.5.: Model performance and return statistics for all tested pairs.

A.1.2. Transformer

Pairwise Results Top 10 Cointegrated Pairs 2020

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (SHV,SMH)	2.46×10^{-4}	2.22423	TLOE*	TLOE*	N/A
2. (SHV,ONEQ)	4.04×10^{-4}	1.27594	TLOE*	33.16%	-1.00
3. (SHV,PHO)	4.13×10^{-4}	1.61198	$-10.92\% \pm 46.61\%$	14.67%	0.78
4. (SHV,PDP)	9.15×10^{-4}	0.63493	TLOE*	43.99%	-1.00
5. (DVY,PEY)	1.41×10^{-3}	0.16805	$0.07\% \pm 0.01\%$	0.10%	1.00
6. (PFF,EMB)	1.45×10^{-3}	0.32286	$0.32\% \pm 0.01\%$	0.71%	1.00
7. (IGSB,BND)	1.56×10^{-3}	0.98751	$0.44\% \pm 0.02\%$	-0.36%	1.01
8. (IFGL,SHV)	3.95×10^{-3}	0.13834	$-0.17\% \pm 0.04\%$	0.41%	0.99
9. (PRFZ,SCZ)	4.47×10^{-3}	0.08079	$0.19\% \pm 0.02\%$	0.31%	1.00
10. (IFGL,EMB)	4.61×10^{-3}	0.30447	$0.22\% \pm 0.05\%$	0.53%	1.00

Table A.6.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2021

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	7.71×10^{-4}	0.04681	$0.28\% \pm 0.02\%$	0.35%	1.00
2. (IFGL,EMB)	1.98×10^{-3}	0.07437	$-0.08\% \pm 0.03\%$	-0.03%	1.00
3. (IFGL,SHV)	2.30×10^{-3}	0.08526	$0.24\% \pm 0.02\%$	0.32%	1.00
4. (IGSB,BND)	2.75×10^{-3}	0.07438	$0.18\% \pm 0.02\%$	0.19%	1.00
5. (IFGL,SOXX)	2.88×10^{-3}	18.40974	TLOE*	23.38%	-1.00
6. (IFGL,SMH)	2.91×10^{-3}	16.40023	$3.48\% \pm 0.16\%$	19.67%	0.86
7. (IFGL,PHO)	2.93×10^{-3}	9.25424	$7.40\% \pm 0.15\%$	12.06%	0.96
8. (IFGL,PDP)	2.97×10^{-3}	12.14490	TLOE*	TLOE*	N/A
9. (IFGL,FTCS)	2.99×10^{-3}	4.48763	$8.12\% \pm 0.08\%$	12.47%	0.96
10. (IFGL,USIG)	3.01×10^{-3}	0.14323	$-21.66\% \pm 41.49\%$	-0.11%	0.78

Table A.7.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2022

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	5.58×10^{-4}	0.07043	$0.59\% \pm 0.06\%$	0.61%	1.00
2. (IFGL,EMB)	1.40×10^{-3}	0.04347	$0.35\% \pm 0.03\%$	0.43%	1.00
3. (IGSB,BND)	1.89×10^{-3}	0.24700	$0.83\% \pm 0.03\%$	0.88%	1.00
4. (USIG,IEI)	2.78×10^{-3}	0.04934	$1.10\% \pm 0.05\%$	1.30%	1.00
5. (IGF,DVY)	2.82×10^{-3}	8.62643	TLOE*	TLOE*	N/A
6. (DVY,PEY)	3.45×10^{-3}	0.11978	$0.36\% \pm 0.02\%$	0.27%	1.00
7. (IGIB,IEI)	4.50×10^{-3}	0.02447	$1.12\% \pm 0.02\%$	1.15%	1.00
8. (IFGL,SOXX)	4.65×10^{-3}	0.95163	TLOE*	TLOE*	N/A
9. (IFGL,SMH)	4.65×10^{-3}	0.78920	TLOE*	TLOE*	N/A
10. (IFGL,PHO)	5.30×10^{-3}	1.97333	TLOE*	3.70%	-1.00

Table A.8.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2023

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	2.82×10^{-4}	0.16529	$0.09\% \pm 0.01\%$	-0.06%	1.00
2. (IFGL,EMB)	7.70×10^{-4}	0.09328	$0.43\% \pm 0.07\%$	0.65%	1.00
3. (IGF,DVY)	1.17×10^{-3}	0.08200	TLOE*	TLOE*	N/A
4. (IGIB,IEI)	1.25×10^{-3}	0.06271	$0.48\% \pm 0.03\%$	0.50%	1.00
5. (DVY,PEY)	1.57×10^{-3}	0.05569	$0.21\% \pm 0.01\%$	0.15%	1.00
6. (USIG,IEI)	1.67×10^{-3}	0.18978	$0.35\% \pm 0.03\%$	0.61%	1.00
7. (IFGL,BND)	1.93×10^{-3}	0.14912	$0.16\% \pm 0.09\%$	0.25%	1.00
8. (IFGL,SMH)	2.58×10^{-3}	0.48091	$15.78\% \pm 0.31\%$	16.05%	1.00
9. (IFGL,MBB)	2.63×10^{-3}	0.22387	$0.01\% \pm 0.08\%$	-0.27%	1.00
10. (IFGL,SOXX)	2.70×10^{-3}	0.19171	TLOE*	TLOE*	N/A

Table A.9.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2024

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	1.38×10^{-4}	0.14245	$0.11\% \pm 0.01\%$	0.10%	1.00
2. (IFGL,MBB)	4.45×10^{-4}	0.29027	$-72.77\% \pm 47.72\%$	TLOE*	N/A
3. (IFGL,EMB)	7.27×10^{-4}	0.15208	$0.35\% \pm 0.02\%$	0.39%	1.00
4. (IGIB,IEI)	7.97×10^{-4}	0.09651	$0.09\% \pm 0.00\%$	0.05%	1.00
5. (IGF,DVY)	8.40×10^{-4}	0.22685	TLOE*	TLOE*	N/A
6. (USIG,IEI)	1.09×10^{-3}	0.25826	$-0.21\% \pm 0.03\%$	-0.25%	1.00
7. (IFGL,BND)	1.20×10^{-3}	0.44944	TLOE*	TLOE*	N/A
8. (IFGL,SMH)	1.64×10^{-3}	17.19063	$15.15\% \pm 0.46\%$	26.74%	0.91
9. (IFGL,SOXX)	1.82×10^{-3}	9.49619	$15.51\% \pm 0.73\%$	11.34%	1.04
10. (IGF,PPH)	2.29×10^{-3}	0.41636	TLOE*	TLOE*	N/A

Table A.10.: Model performance and return statistics for all tested pairs.

A.1.3. Time-MoE

Pairwise Results Top 10 Cointegrated Pairs 2020

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (SHV,SMH)	2.46×10^{-4}	0.02923	$-47.57\% \pm 69.90\%$	TLOE*	N/A
2. (SHV,ONEQ)	4.04×10^{-4}	0.02700	$-31.11\% \pm 59.98\%$	33.16%	0.52
3. (SHV,PHO)	4.13×10^{-4}	0.04419	$-59.32\% \pm 55.14\%$	14.67%	0.35
4. (SHV,PDP)	9.15×10^{-4}	0.02771	$-26.66\% \pm 63.59\%$	43.99%	0.51
5. (DVY,PEY)	1.41×10^{-3}	0.14001	$0.04\% \pm 0.02\%$	0.10%	1.00
6. (PFF,EMB)	1.45×10^{-3}	0.22588	$0.41\% \pm 0.15\%$	0.71%	1.00
7. (IGSB,BND)	1.56×10^{-3}	0.29892	$0.06\% \pm 0.12\%$	-0.36%	1.00
8. (IFGL,SHV)	3.95×10^{-3}	0.04271	$0.13\% \pm 0.23\%$	0.41%	1.00
9. (PRFZ,SCZ)	4.47×10^{-3}	0.08381	$0.25\% \pm 0.10\%$	0.31%	1.00
10. (IFGL,EMB)	4.61×10^{-3}	0.04807	$0.81\% \pm 0.45\%$	0.53%	1.00

Table A.11.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2021

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	7.71×10^{-4}	0.05746	$0.24\% \pm 0.08\%$	0.35%	1.00
2. (IFGL,EMB)	1.98×10^{-3}	0.08541	$-0.15\% \pm 0.03\%$	-0.03%	1.00
3. (IFGL,SHV)	2.30×10^{-3}	0.07188	$0.26\% \pm 0.08\%$	0.32%	1.00
4. (IGSB,BND)	2.75×10^{-3}	0.09011	$0.12\% \pm 0.05\%$	0.19%	1.00
5. (IFGL,SOXX)	2.88×10^{-3}	0.07939	$-1.31\% \pm 51.50\%$	23.38%	0.80
6. (IFGL,SMH)	2.91×10^{-3}	0.08343	$-44.97\% \pm 59.89\%$	19.67%	0.46
7. (IFGL,PHO)	2.93×10^{-3}	0.03839	$9.21\% \pm 2.02\%$	12.06%	0.97
8. (IFGL,PDP)	2.97×10^{-3}	0.09252	$-91.43\% \pm 32.19\%$	TLOE*	N/A
9. (IFGL,FTCS)	2.99×10^{-3}	0.02703	$9.89\% \pm 2.83\%$	12.47%	0.98
10. (IFGL,USIG)	3.01×10^{-3}	0.09892	$-10.79\% \pm 31.19\%$	-0.11%	0.89

Table A.12.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2022

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	5.58×10^{-4}	0.05012	$0.54\% \pm 0.12\%$	0.61%	1.00
2. (IFGL,EMB)	1.40×10^{-3}	0.04573	$0.33\% \pm 0.13\%$	0.43%	1.00
3. (IGSB,BND)	1.89×10^{-3}	0.01860	$0.87\% \pm 0.09\%$	0.88%	1.00
4. (USIG,IEI)	2.78×10^{-3}	0.02754	$0.89\% \pm 0.30\%$	1.30%	1.00
5. (IGF,DVY)	2.82×10^{-3}	0.20267	TLOE*	TLOE*	N/A
6. (DVY,PEY)	3.45×10^{-3}	0.11510	$0.07\% \pm 0.09\%$	0.27%	1.00
7. (IGIB,IEI)	4.50×10^{-3}	0.02443	$0.83\% \pm 0.34\%$	1.15%	1.00
8. (IFGL,SOXX)	4.65×10^{-3}	0.17117	TLOE*	TLOE*	N/A
9. (IFGL,SMH)	4.65×10^{-3}	0.19464	TLOE*	TLOE*	N/A
10. (IFGL,PHO)	5.30×10^{-3}	0.05408	$-61.78\% \pm 52.34\%$	3.70%	0.37

Table A.13.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2023

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	2.82×10^{-4}	0.09098	$-0.02\% \pm 0.02\%$	-0.06%	1.00
2. (IFGL,EMB)	7.70×10^{-4}	0.08899	$0.53\% \pm 0.17\%$	0.65%	1.00
3. (IGF,DVY)	1.17×10^{-3}	0.05588	TLOE*	TLOE*	N/A
4. (IGIB,IEI)	1.25×10^{-3}	0.06727	$0.27\% \pm 0.19\%$	0.50%	1.00
5. (DVY,PEY)	1.57×10^{-3}	0.05366	$0.12\% \pm 0.03\%$	0.15%	1.00
6. (USIG,IEI)	1.67×10^{-3}	0.08134	$0.38\% \pm 0.24\%$	0.61%	1.00
7. (IFGL,BND)	1.93×10^{-3}	0.15046	$0.46\% \pm 0.14\%$	0.25%	1.00
8. (IFGL,SMH)	2.58×10^{-3}	0.02390	$-8.15\% \pm 48.13\%$	16.05%	0.79
9. (IFGL,MBB)	2.63×10^{-3}	0.18685	$0.25\% \pm 0.10\%$	-0.27%	1.01
10. (IFGL,SOXX)	2.70×10^{-3}	0.03220	$-58.97\% \pm 55.89\%$	TLOE*	N/A

Table A.14.: Model performance and return statistics for all tested pairs.

Pairwise Results Top 10 Cointegrated Pairs 2024

Pair	Cointegration Score	test MSE	YoY Returns (std)	Theoretical Return Under Perfect Information	Return Score
1. (PFF,EMB)	1.38×10^{-4}	0.09364	$0.08\% \pm 0.02\%$	0.10%	1.00
2. (IFGL,MBB)	4.45×10^{-4}	0.09236	$-42.19\% \pm 51.16\%$	TLOE*	N/A
3. (IFGL,EMB)	7.27×10^{-4}	0.07025	$0.44\% \pm 0.07\%$	0.39%	1.00
4. (IGIB,IEI)	7.97×10^{-4}	0.08799	$0.02\% \pm 0.01\%$	0.05%	1.00
5. (IGF,DVY)	8.40×10^{-4}	0.16539	TLOE*	TLOE*	N/A
6. (USIG,IEI)	1.09×10^{-3}	0.18845	$-0.21\% \pm 0.05\%$	-0.25%	1.00
7. (IFGL,BND)	1.20×10^{-3}	0.09422	$-52.57\% \pm 52.18\%$	TLOE*	N/A
8. (IFGL,SMH)	1.64×10^{-3}	0.14566	$-21.83\% \pm 54.49\%$	26.74%	0.62
9. (IFGL,SOXX)	1.82×10^{-3}	0.16738	$-7.09\% \pm 32.49\%$	11.34%	0.83
10. (IGF,PPH)	2.29×10^{-3}	0.03281	TLOE*	TLOE*	N/A

Table A.15.: Model performance and return statistics for all tested pairs.

B. Diagnostic Analyses

To gain deeper insight into the observed results, this section presents additional interpretative and diagnostic analyses. These analyses present graphs and describe what is seen in the graphs following the conventions of a results section. Actual implications of those findings are discussed in section 5.1.

B.1. Pair Discovery

B.1.1. TR-PI All Pairs

Using the combination of Engle-Granger cointegration testing and a z-score-based trading strategy simulation, a portion results in a total loss of equity even when supplied with the actual future spread. Over all the pairs for which data was gathered, as detailed in Table B.1, p-values were scored in a total range between a p-value of $p = 1.38 * 10^{-4}$ and a p-value of $p = 5.3 * 10^{-3}$. During the years 2020 through 2024 the total loss of equity occurred as follows: once in 2020, once in 2021, three times in 2022, twice in 2023, and four times in 2024. This means that across 50 pairs, 11 pairs resulted in total loss of equity (TLOE). The other 39 pairs averaged a TR-PI of 5.80% YoY. The highest TR-PI among all pairs was 43.99% YoY for the pair (SHV-PDP). Examining all pairs, including those that did not result in top 10 pairs per period, an analysis for the year 2024 showed the differences in statistics for time series that resulted in a $TLOE$ TR-PI and those that did not. This is illustrated in Figure B.1, with standard deviation (σ) values ranging from $\sigma = 3.16$ to $\sigma = 32.40$ over the observation period. In contrast, non-$TLOE$ pairs display a more narrow volatility range, from $\sigma = 5.43$ to $\sigma = 21.44$. Furthermore, the testing period for unprofitable pairs ends its time window with the spread approximately 8 ticks lower than at its commencement, whereas profitable pairs experience an increase of 3 ticks in spread over the same period.

Pair	Cointegration Score	TR-PI
2020		
1. (SHV,SMH)	2.46×10^{-4}	TLOE*
2. (SHV,ONEQ)	4.04×10^{-4}	33.16%
3. (SHV,PHO)	4.13×10^{-4}	14.67%
4. (SHV,PDP)	9.15×10^{-4}	43.99%
5. (DVY,PEY)	1.41×10^{-3}	0.10%
6. (PFF,EMB)	1.45×10^{-3}	0.71%
7. (IGSB,BND)	1.56×10^{-3}	-0.36%
8. (IFGL,SHV)	3.95×10^{-3}	0.41%
9. (PRFZ,SCZ)	4.47×10^{-3}	0.31%
10. (IFGL,EMB)	4.61×10^{-3}	0.53%
2021		
1. (PFF,EMB)	7.71×10^{-4}	0.35%
2. (IFGL,EMB)	1.98×10^{-3}	-0.03%
3. (IFGL,SHV)	2.30×10^{-3}	0.32%
4. (IGSB,BND)	2.75×10^{-3}	0.19%
5. (IFGL,SOXX)	2.88×10^{-3}	23.38%
6. (IFGL,SMH)	2.91×10^{-3}	19.67%
7. (IFGL,PHO)	2.93×10^{-3}	12.06%
8. (IFGL,PDP)	2.97×10^{-3}	TLOE*
9. (IFGL,FTCS)	2.99×10^{-3}	12.47%
10. (IFGL,USIG)	3.01×10^{-3}	-0.11%
2022		
1. (PFF,EMB)	5.58×10^{-4}	0.61%
2. (IFGL,EMB)	1.40×10^{-3}	0.43%
3. (IGSB,BND)	1.89×10^{-3}	0.88%
4. (USIG,IEI)	2.78×10^{-3}	1.30%
5. (IGF,DVY)	2.82×10^{-3}	TLOE*
6. (DVY,PEY)	3.45×10^{-3}	0.27%
7. (IGIB,IEI)	4.50×10^{-3}	1.15%
8. (IFGL,SOXX)	4.65×10^{-3}	TLOE*
9. (IFGL,SMH)	4.65×10^{-3}	TLOE*
10. (IFGL,PHO)	5.30×10^{-3}	3.70%
2023		
1. (PFF,EMB)	2.82×10^{-4}	-0.06%
2. (IFGL,EMB)	7.70×10^{-4}	0.65%
3. (IGF,DVY)	1.17×10^{-3}	TLOE*
4. (IGIB,IEI)	1.25×10^{-3}	0.50%
5. (DVY,PEY)	1.57×10^{-3}	0.15%
6. (USIG,IEI)	1.67×10^{-3}	0.61%
7. (IFGL,BND)	1.93×10^{-3}	0.25%
8. (IFGL,SMH)	2.58×10^{-3}	16.05%
9. (IFGL,MBB)	2.63×10^{-3}	-0.27%
10. (IFGL,SOXX)	2.70×10^{-3}	TLOE*
2024		
1. (PFF,EMB)	1.38×10^{-4}	0.10%
2. (IFGL,MBB)	4.45×10^{-4}	TLOE*
3. (IFGL,EMB)	7.27×10^{-4}	0.39%
4. (IGIB,IEI)	7.97×10^{-4}	0.05%
5. (IGF,DVY)	8.40×10^{-4}	TLOE*
6. (USIG,IEI)	1.09×10^{-3}	-0.25%
7. (IFGL,BND)	1.20×10^{-3}	TLOE*
8. (IFGL,SMH)	1.64×10^{-3}	26.74%
9. (IFGL,SOXX)	1.82×10^{-3}	11.34%
10. (IGF,PPH)	2.29×10^{-3}	TLOE*

Table B.1.: Table showing TR-PI of top 10 cointegrated pairs over all five test windows (2020-2024).

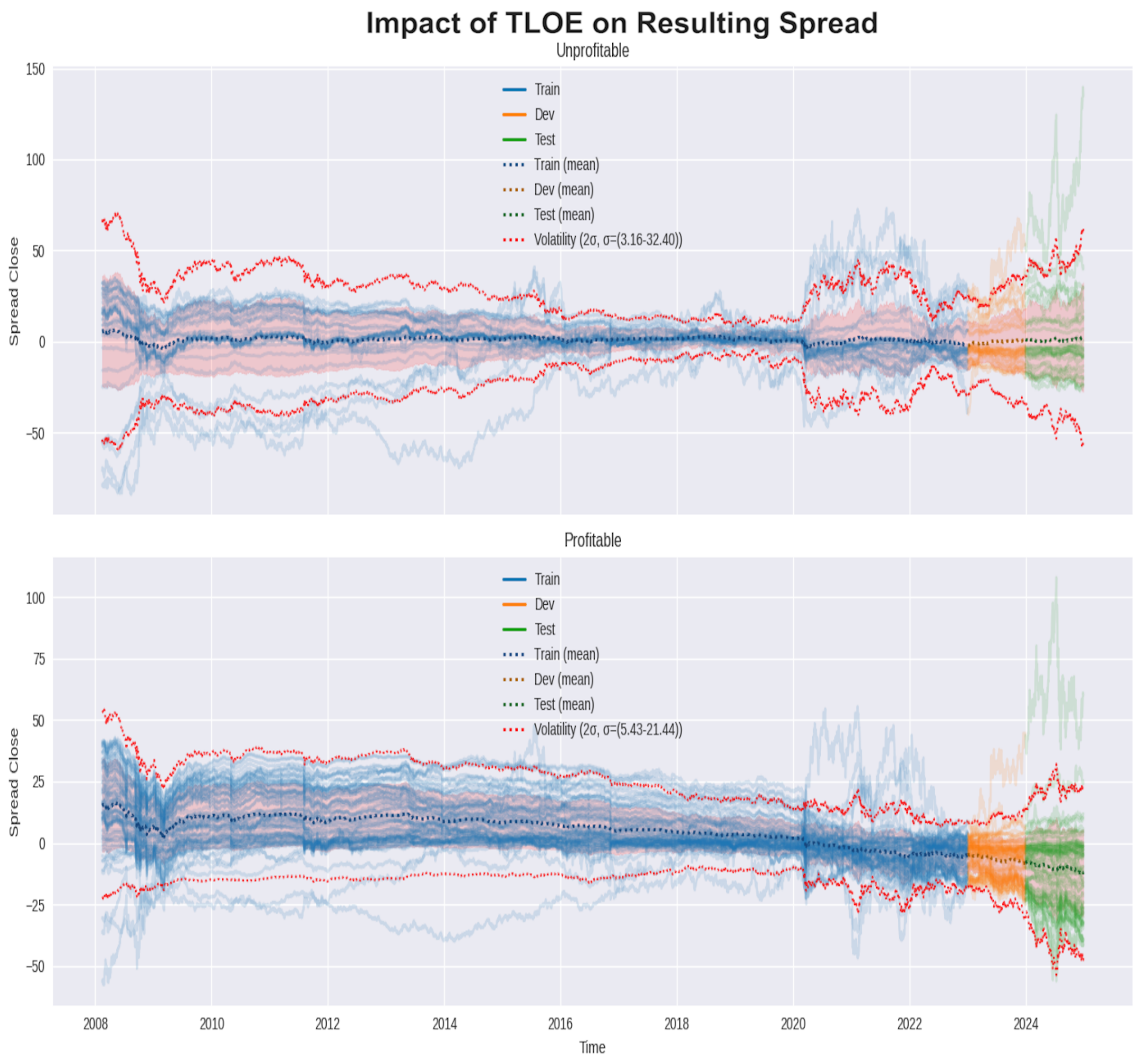
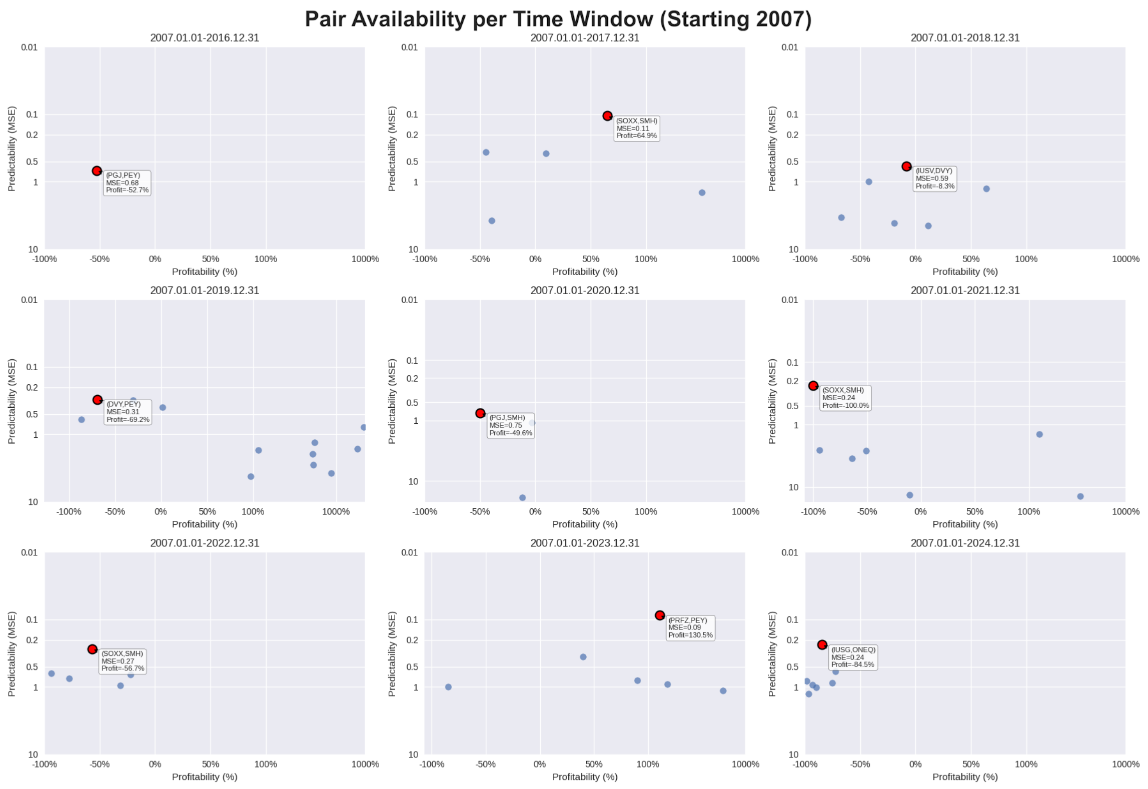


Figure B.1.: Above: Complete time series of pairs that result in a -100% (total loss of equity) return
Below: Complete time series of remaining pairs

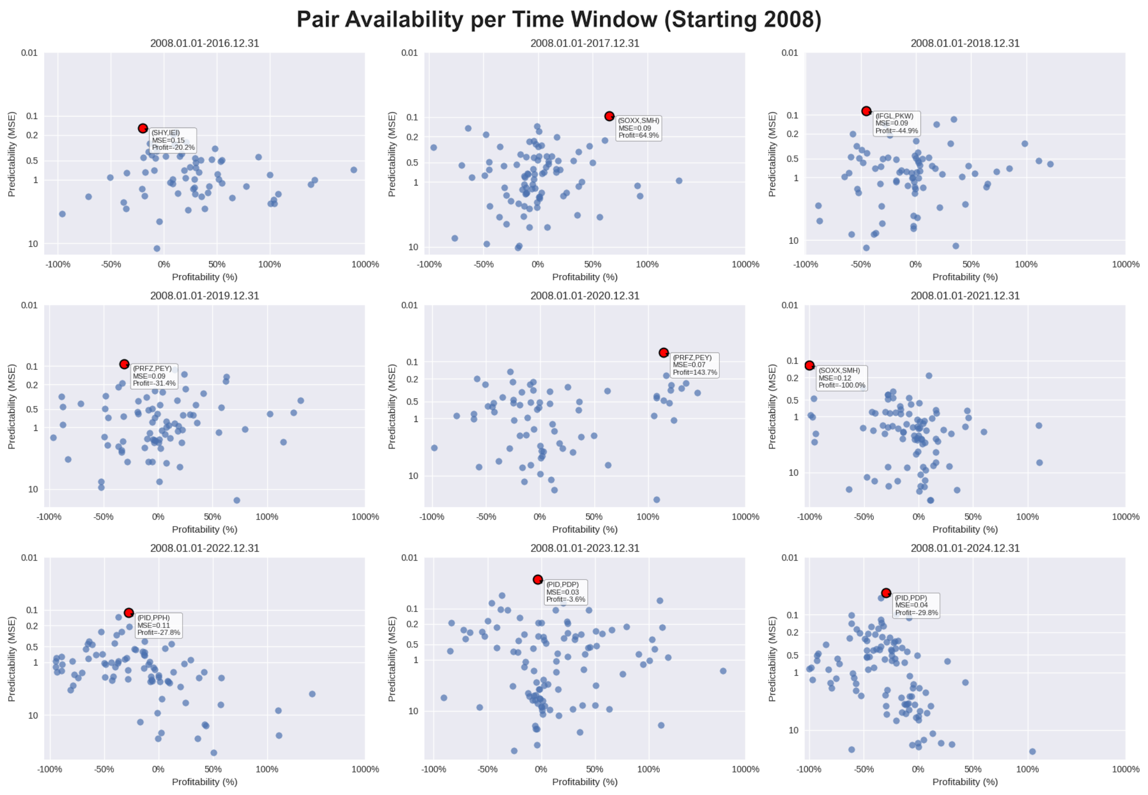
B.1.2. Profitability, Predictability ETF Pairs

To assess the viability of the cointegrated ETF pairs discovered via the Engle-Granger two-step cointegration test (threshold $p < 0.05$), this proxy-based analysis shows the two key dimensions that are tested in this study; profitability and predictability. The y-axis in the resulting scatter plot, as shown in Figure B.2a and Figure B.2b, represent the Mean Squared Error (MSE) from the baseline Kalman Filter model over the set of all available pairs[1]. The x-axis represents the theoretical maximum profitability - defined as the cumulative return assuming perfect foresight, with access to future values of the spread.
For the period starting in 2008, between 43 and 95 ETFs remain across all time periods. This graph also shows the great difference in profitable opportunities across years. Some years, such as 2024, shows only 9 possibly profitable trading pairs under perfect information, whereas the previous year shows 35 such pairs from all pairs with $p < 0.05$.

[1]A larger set than the total of top 10 cointegrated pairs



(a) With 2007 as starting year.



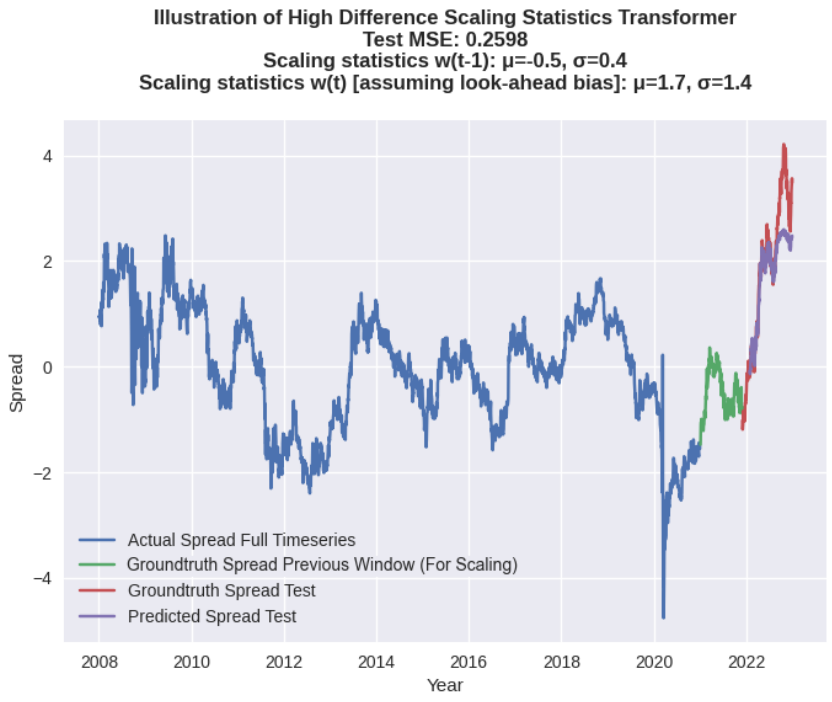
(b) With 2008 as starting year.

Figure B.2.: Profitability vs Predictability

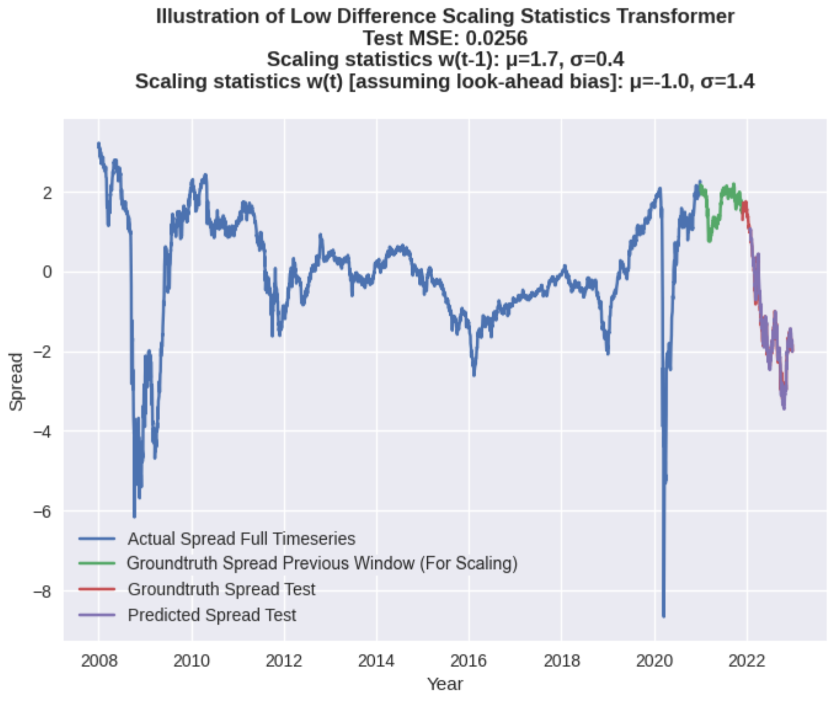
B.2. Predictability

B.2.1. Trend Sensitivity

To examine the suspected observation of trend sensitivity in the Transformer and Kalman Filter model, Figure B.3a shows an empirical example of a pair with high test MSE. The figure shows on both sides the window w_{t-1} in red, and window w_t in yellow and purple (respectively the prediction and groundtruth). As z-score rolling normalization is used, a difference of mean μ_{t-1} and σ_{t-1} at time $t-1$ are observed, with normalization of the prediction at time t at a different scale. Specifically, the statistics are observed to be $\mu_{t-1} = 0.5$ and $\sigma_{t-1} = 0.4$, whereas the intended values should be $\mu_t = 1.7$ and $\sigma_{t-1} = 1.4$.
On the other hand, Figure B.3b shows a figure where scaling statistics of window w_{t-1} are comparatively more similar. Because $\mu_{t-1} = 1.7$, $\sigma_{t-1} = 0.4$, which are comparatively similar in scale to the actual scaling statistics of the current window, with $\mu_t = -1.0$ and $\sigma_t = 1.4$. The prediction for this combination, labeled as predicted spread test in the figure, is also observed to be more accurate. In the figures, compared to the difference in scales, there is no easily discernible difference in the ability of the forecasts to follow the form of the ground truth spread.



(a) Transformer: Empirical example of high test MSE caused by scaling differences.

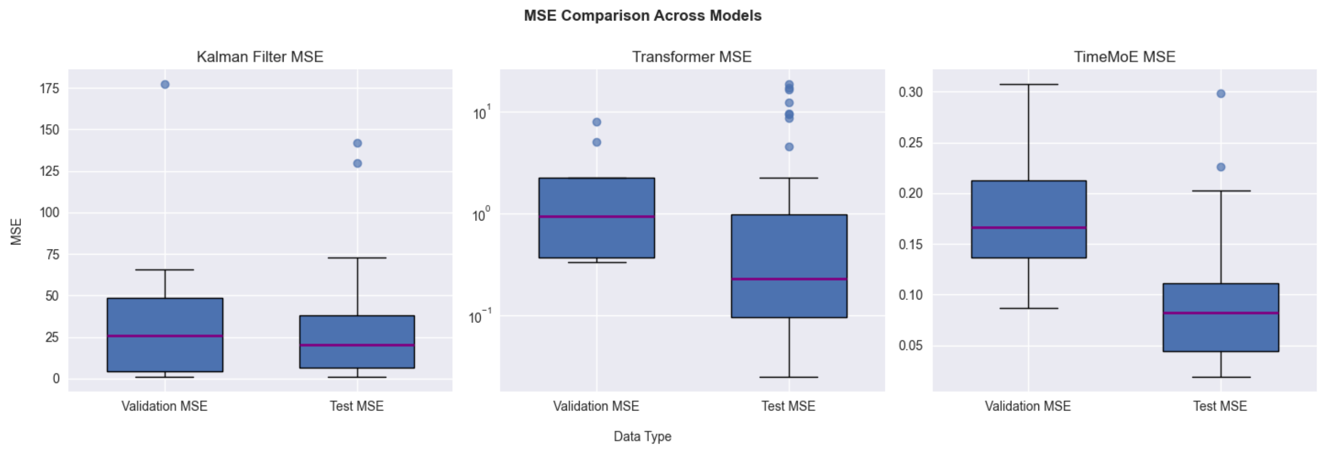


(b) Transformer: Empirical example of low test MSE caused by similar scaling statistics.

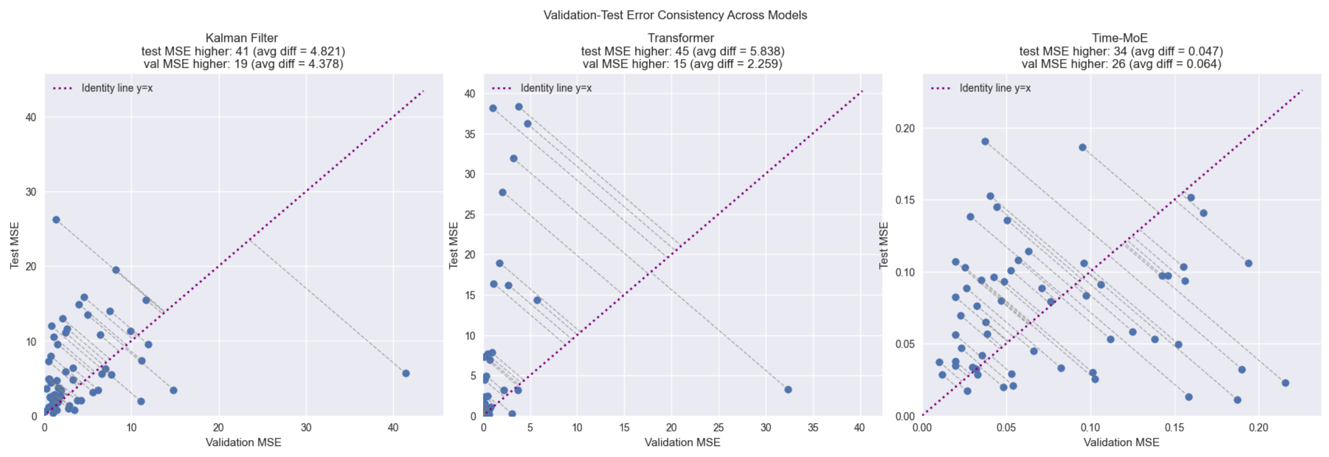
Figure B.3.: Comparison of Transformer and Time-MoE: Both show low test MSE and high validation MSE due to similar train and test timeseries patterns.

B.2.2. Discrepancy in Model Accuracy Between Validation and Test

Empirical observations indicated that there was a divergence between validation and test accuracy, illustrated in figure Figure B.4b. This result was mitigated through TSCV. It has exhibited lower validation MSE relative to test MSE with a moderate degree of consistency. This analysis aims to elucidate the potential causes underlying these observations and to determine whether the apparent magnitudes of these differences are statistically significant.
Refer to Figure B.4b for a graphical representation of divergence for the three models before implementation of Time Series Cross-Validation. This figure presents validation MSE on the x-axis and test MSE on the y-axis for each model. In a standard machine learning pipeline, it is anticipated that the plotted data points would approximate the identity line $y = x$ within a predictable range of standard deviation. While the Time-MoE model adheres to this expected relationship, the Kalman Filter and Transformer model demonstrate a deviation characterized by elevated test MSE values.



(a) Comparison of validation and test MSE over all three models, **with Expanding Window TSCV**. Boxplot used due to lack of direct pairing between validation and test MSE values across years (2016–2019 vs. 2020–2024). Aggregated distributions are compared as temporal correspondence is absent.



(b) Comparison of validation and test MSE over all three models, **before Expanding Window TSCV**. Scatter plot used due to direct pairing between validation and test MSE values across years (test windows 2020-2024 used, with old window methodology). Pointwise comparisons are therefore enabled.

Figure B.4.: Comparison of validation and test MSE over all three models, with and without Expanding Window TSCV.

B.3. Profitability

B.3.1. Predictive Accuracy vs. Economic Performance

To evaluate the robustness of predictive signals utilized as inputs in the z-score-based trading strategy, it is crucial to establish that high predictive accuracy translates into sustained profitability. To quantitatively investigate this relationship, various methods were employed to introduce mean squared error (MSE) into the ground truth spread values, thereby enabling a systematic analysis of the impact of types of erratic predictions the models made. Specifically, the following perturbations were applied: (i) the addition of synthetic Gaussian noise to the predictions, (ii) temporal lagging of predicted values, (iii and iv) scaling the spread by multiplicative factors to simulate both amplified (higher) and attenuated (lower) predictions, and (v) simulating trend-following behavior with reduced accuracy at turning points, as is observed with the Kalman Filter-based approach, achieved via the application of smoothing windows to ground truth values. Figure B.6 shows an illustration of an arbitrary time series to allow for better interpretability of the effects of the perturbations.
Figure B.5 illustrates all five perturbations. Some perturbations caused expected effects to the resulting profitability. Introducing Gaussian error results in a logarithmic fit showing a profitability of approximately 30% when test MSE is zero[2], whereas the profitability reduces to 0% for a test MSE of 12, and logarithmically grows more unprofitable the further the MSE rises. Similarly, prediction error caused by innacurate scaling shows a decrease in profitability for higher test MSE. However, this finding is juxtaposed with inaccurately high scaling causing the exact opposite effect.

The results of this analysis, averaged across the top 30 cointegrated pairs during the 2023 and 2024 periods, are presented in Figure B.5. It shows a relationship, with the highest returns accompanying a test MSE under a value of 2, rising the closer it gets to zero for both time periods.

[2]Note that a YoY return percentage under a Test MSE of zero is the same concept as the concept of TR-PI, as often mentioned in this paper. A complete explanation on TR-PI is given in subsection 3.5.2

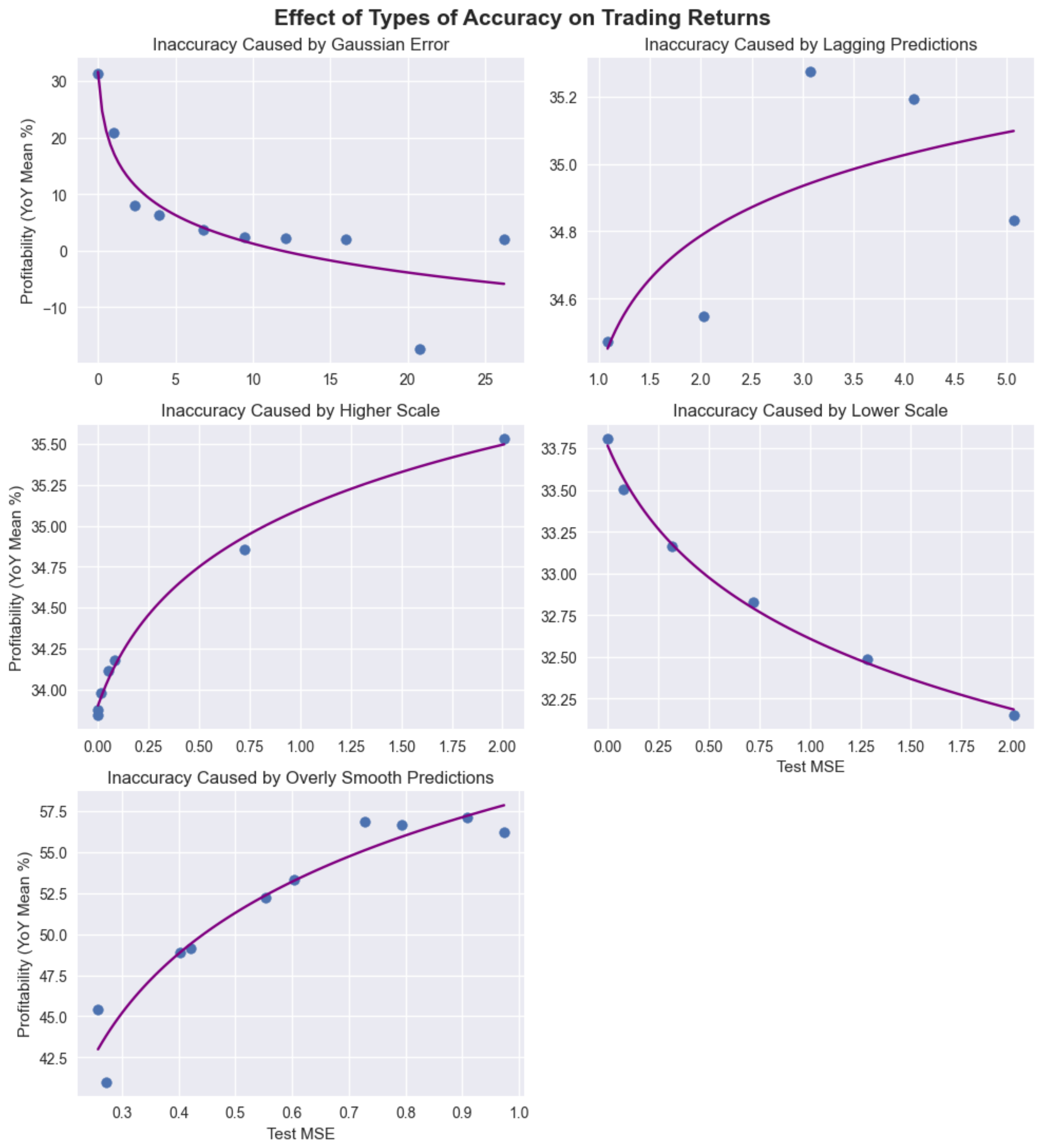


Figure B.5.: Predictive Accuracy vs. Economic Performance Comparison Between Types of Errors
Using artificially added noise to see effect on returns.

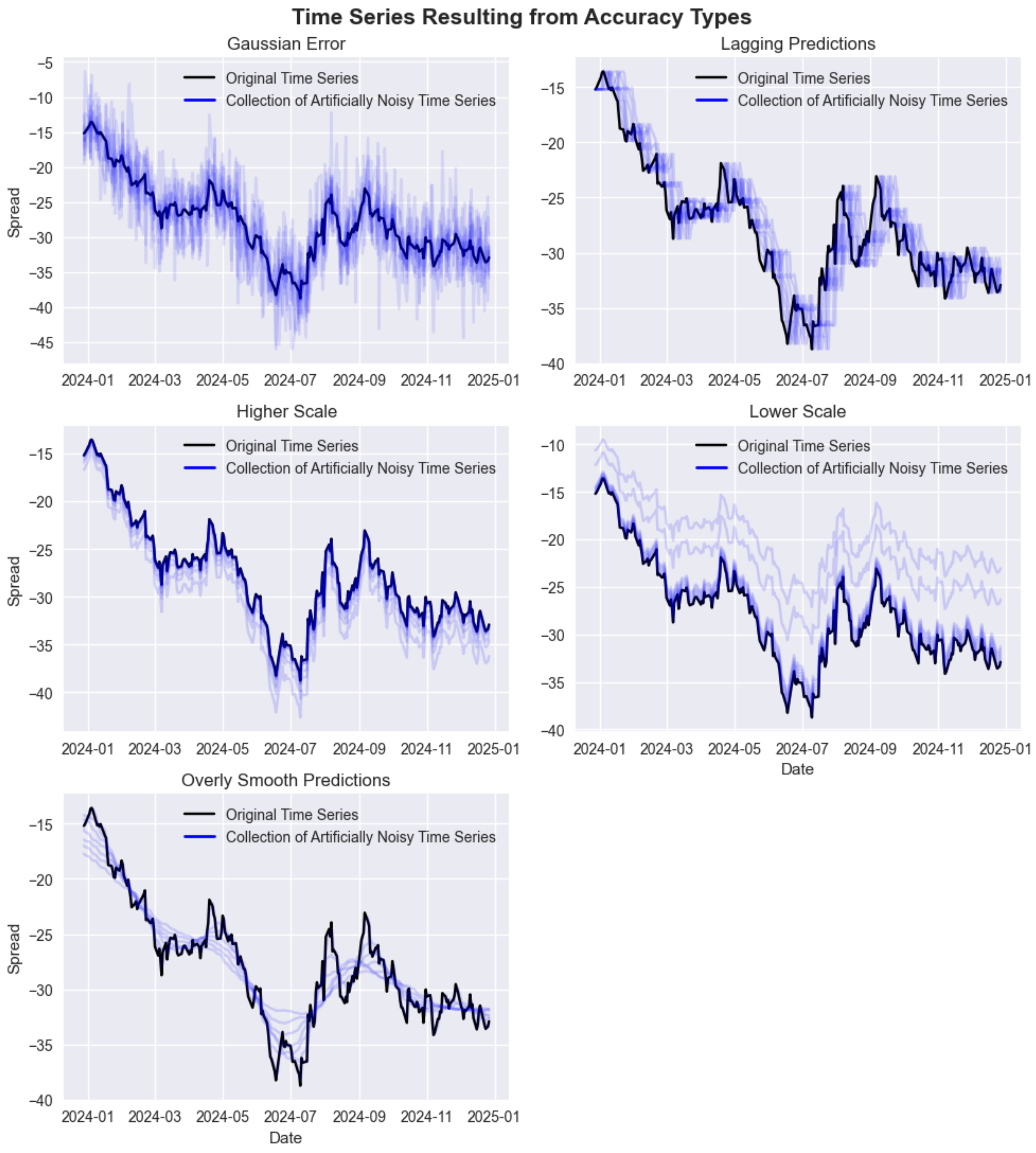


Figure B.6.: Illustrated Time Series For Five Error Structures:
i) Gaussian Error: simulating normally distributed residuals
ii) Lagging Predictions: simulating autocorrelated residuals
iii) Higher Scale: simulating heteroskedasticity resulting in over-amplified predictions due to inaccurate normalization
iv) Lower Scale: simulating heteroskedasticity resulting in under-amplified predictions due to inaccurate normalization
v) Overly Smooth Predictions: simulating low variance residuals