N-BEATS Perceiver: A Novel Approach for Robust Cryptocurrency Portfolio Forecasting

Attilio Sbrana (attilio.sbrana@ga.ita.br)
Instituto Tecnológico de Aeronáutica https://orcid.org/0000-0003-3540-426X
Paulo André Lima de Castro (pauloac@ita.br)
Instituto Tecnológico de Aeronáutica https://orcid.org/0000-0001-5515-1672

Research Article

Keywords: N-BEATS, Perceiver, Transformers, Deep Learning, Forecasting, Cryptocurrency

DOI: https://doi.org/10.21203/rs.3.rs-2618277/v1

N-BEATS Perceiver: A Novel Approach for Robust Cryptocurrency Portfolio Forecasting

Attilio Sbrana and Paulo André Lima de Castro

Autonomous Computational Systems Lab - LABSCA
Aeronautics Institute of Technology (ITA)
São José dos Campos, São Paulo, Brazil.

Contributing authors: attilio.sbrana@ga.ita.br; pauloac@ita.br;

Abstract

Cryptocurrencies are well-known for their high volatility and unpredictability, posing a challenge for forecasting using traditional methods. To address this issue, we explore variations of the N-BEATS deep learning (DL) architecture by adding convolutional network layers, Transformer mechanisms, and the Mish activation function, and propose a novel approach for forecasting cryptocurrency portfolios. Our comprehensive evaluation demonstrates that our model variations outperform other DL and traditional forecasting methods in numerous evaluation metrics, making them powerful tools for predicting cryptocurrency prices and portfolios in the rapidly-evolving cryptocurrency market. Furthermore, our newly proposed N-BEATS Perceiver model, a Transformer-based N-BEATS variation, exhibits a robust risk profile with less downside compared to other models and performs exceptionally well when evaluated using the TOPSIS method across a wide range of portfolio evaluation parameters. These results underscore the potential of our approach and specifically highlight the N-BEATS Perceiver's potential for selecting portfolios and forecasting cryptocurrency prices, offering valuable insights into the development of more accurate and reliable models for cryptocurrency forecasting.

Keywords: N-BEATS, Perceiver, Transformers, Deep Learning, Forecasting, Cryptocurrency

1 Introduction

Cryptocurrencies have earned a reputation for their extreme volatility and dramatic price movements, which are driven by a number of variables including trading volume, market beta, and volatility (Khedr et al., 2021). Due to the complexity of the matter, it is challenging to develop accurate price prediction models for cryptocurrencies, despite their increasing significance in the financial world. In recent years, both machine learning (ML) and deep learning (DL) communities have paid substantial attention to the application of ML algorithms for cryptocurrency price predictions (Fang et al., 2022). In this quickly expanding market, these ML-based approaches have increasingly demonstrated their capacity to enhance the precision of predictions, promote better portfolio decisions, and enable more effective market risk management.

In this paper, we describe novel methods for forecasting cryptocurrency prices based on the N-BEATS (Oreshkin, Carpov, Chapados, & Bengio, 2020) architecture. We investigate variations of the N-BEATS model, such as the addition of Convolutional Neural Networks (CNNs) and Transformer (Vaswani et al., 2017) mechanisms to extract spatial features and weight input features, as well as the application of the Mish (Misra, 2020) activation function. Next, we explore the use of N-BEATS for portfolio-level univariate forecasting. To evaluate the effectiveness of our proposed approach, we conduct three experiments on historical cryptocurrency price data. Two of these experiments focus on univariate time series point forecasting, while the third centers on a portfolio selection task. We compare the outcomes of our proposed approaches with those of existing ML and conventional statistical techniques. The purpose of these experiments is to assess the accuracy and robustness of various forecasting techniques for portfolio selection, including the ability to choose portfolios with the highest forecasted returns.

Our research provides strong evidence that the N-BEATS architecture, when combined with convolutional transformations, attention layers, and the Mish activation function, is a powerful approach for forecasting portfolio-level cryptocurrency prices. In evaluating a diverse set of forecasting methods, our newly proposed N-BEATS Perceiver model, a version of N-BEATS that leverages a Transformer architecture, emerged as the top performer against other variations. The N-BEATS Perceiver model showed exceptional results across all assessment criteria, exhibited a superior error distribution, and had superior mean and median error performance in comparison to other models and N-BEATS variations. In addition, the N-BEATS Perceiver exhibited remarkable scalability properties as inputs increased. Lastly, as a portfolio selection approach, N-BEATS Perceiver was discovered to have a robust risk profile, with less downside relative to other models, and performed remarkably well when evaluated using the TOPSIS (Uzun, Taiwo, Syidanova, & Uzun Ozsahin, 2021) method. Consequently, our research offers a highly promising method for forecasting cryptocurrency prices and portfolios in the volatile and rapidly-evolving cryptocurrency market.

In the following sections, we provide a detailed description of our research on forecasting cryptocurrency portfolios using variations of the N-BEATS architecture. The literature review in Section 2 covers recent advancements in DL for time series forecasting, including improvements to the N-BEATS architecture, and evaluates the effectiveness of diffeerent approaches in the domain of cryptocurrency forecasting. In Section 3, we describe our proposed architecture, the N-BEATS Perceiver, as well as competing architectures based on convolutional network layers, Transformer mechanisms, and the Mish activation function. We also present our preferred evaluation metrics, and introduce an alternative method for interpreting DL experiments through SHAP values. In Section 4, we detail our experimental setup, including data collection and preprocessing, and three different experiments aimed at evaluating various forecasting techniques for portfolio selection. Section 5 presents the results of our experiments, including main forecasting results, subsample forecasting results, and portfolio selection results. In Section 6, we provide a comprehensive discussion of our findings and implications for the use of the N-BEATS Perceiver model in the context of cryptocurrency portfolio forecasting and selection. Finally, Section 7 concludes our work by summarizing our contributions, implications for future research, and practical relevance of our findings.

2 Literature Review

The goal of this research is to improve the performance and versatility of the N-BEATS architecture for time series forecasting. To achieve this, we provide a comprehensive overview of the existing literature in the areas of DL for time series forecasting, improving DL architectures, and in cryptocurrency price forecasting. We focus on the most relevant studies that are directly related to this research, highlighting their key contributions and how they inform our proposed approach.

2.1 Deep Learning for Time Series Forecasting

The use of ML techniques for time series forecasting has gained considerable attention in recent years due to their potential to improve the accuracy of predictions and enable effective decision-making. In particular, the N-BEATS (Oreshkin et al., 2020) architecture has been proposed as a novel neural network architecture for univariate time series point forecasting. The architecture is based on backward and forward residual links and a stack of fully-connected layers, and is designed to be interpretable and applicable to a wide range of target domains. The authors of the paper have tested the model on several datasets, including the M3, M4, and TOURISM competition datasets, and achieved state-of-the-art performance.

More recently, the N-HiTS model (Challu et al., 2022) has been proposed as a state-of-the-art approach for time series forecasting that builds upon the N-BEATS model. This architecture includes a MaxPool layer at

the block level, which enables the model to learn short-term and long-term effects in the time series and combine these forecasts to make more accurate predictions. Additionally, the authors have added residual connections between blocks in each stack and modified the architecture of the blocks to further improve performance. Through experiments on large-scale datasets, the authors have demonstrated that N-HiTS provides an average accuracy improvement of nearly 20% over the latest Transformer architectures, while also reducing computation time by an order of magnitude.

Overall, the use of DL techniques for time series forecasting has demonstrated impressive performance, with the N-BEATS and N-HiTS architectures providing state-of-the-art results. The interpretability and applicability of these architectures make them particularly attractive for decision-making in various domains, such as finance, where accurate predictions are essential.

2.2 Improving the N-BEATS Architecture for Time Series Forecasting

N-BEATS is a powerful and effective architecture for time series forecasting, but it has some limitations that could be addressed by modifying its architecture. In this paper, we propose several modifications to the N-BEATS model that aim to improve its performance and make it more versatile.

Previous works have attempted to improve the N-BEATS architecture for time series forecasting, such as N-BEATS-RNN (Sbrana, Debiaso Rossi, & Coelho Naldi, 2020), which added a Recurrent Neural Network (RNN) architecture to the N-BEATS model and used a Neural Architecture Search (NAS) to optimize the RNN's architecture for improved performance. Recent advances in RNNs have the potential to capture long-term dependencies, but their complexity and computational cost have caused them to fall out of favor in many DL tasks. Moreover, NAS may be a way to optimize RNN architecture, but the search is highly computationally expensive. Therefore, in this research, we do not further explore a hybridization of N-BEATS with RNNs but instead focus on alternative architectures that are more computationally efficient.

Instead, we propose three specific modifications to the N-BEATS architecture. The first is to add a convolutional layer on top of each stack in the N-BEATS model. Convolutional layers are a key component of convolutional neural networks (CNNs) (LeCun & Bengio, 1998) and are widely used in image and video processing tasks. A convolutional layer applies a set of filters to an input image, each of which is designed to detect a specific spatial pattern. This allows the CNN to extract features from the input data that are robust to small translations and deformations. In the context of N-BEATS, a convolutional layer on top of each stack could help the model to better capture local patterns in the input time series data, which could improve its forecasting performance.

The second modification we propose is to add a Transformer Encoder on top of each stack in the N-BEATS model. Attention mechanisms and the Transformer have been widely used in natural language processing tasks, where they

allow a model to integrate information from different sources in a flexible and efficient manner. In the context of N-BEATS, a Transformer Encoder could be used to integrate information from distant points of the time series which could improve the model's forecasting performance. Additionally, the use of attention mechanisms could help to reduce the computational complexity of the N-BEATS model, allowing it to scale to larger inputs and outputs.

We finally propose incorporating a Perceiver Encoder into N-BEATS to improve the model's ability to handle inputs of various modalities and sizes. The Perceiver (Jaegle, Borgeaud, et al., 2021; Jaegle, Gimeno, et al., 2021) is based on the Transformer, and uses an asymmetric attention mechanism to iteratively distill inputs into a tight latent bottleneck, allowing it to scale to handle large inputs and outperform specialized models on classification tasks across various modalities. If only the encoder portion of a Perceiver model is used, the model would still be able to map inputs of a wide range of modalities to a fixed-size latent space, but it would not be able to generate outputs of various sizes and semantics. By using a Perceiver Encoder on top of each stack of an N-BEATS model, it could improve N-BEATS' ability to compactly encode inputs, which could help to reduce the computational complexity of the model, allowing it to scale to larger inputs and outputs.

The modifications we have proposed to the N-BEATS architecture could help to improve its performance and make it more versatile. In addition, these modifications could help to reduce the computational complexity of the model, allowing it to scale to larger inputs and outputs. We hypothesize that these modifications will be beneficial in improving the performance and scalability of N-BEATS for time series forecasting tasks.

2.3 Cryptocurrency Time Series Forecasting

To contextualize our research in the domain of cryptocurrency time series forecasting, it is essential to understand the current state of the field. Fang et al. (2022) have conducted a comprehensive survey of cryptocurrency trading research, covering 146 papers on various aspects of cryptocurrency, including trading systems, technical analysis, ML technology, portfolio construction, market condition research, and more. In their exhaustive examination of the literature on cryptocurrency trading, Fang et al. (2022) identify ML and DL technologies as the most popular researched topics. The authors provide an overview of various ML algorithms, including classification, clustering, regression, and reinforcement learning, and note that DL algorithms, particularly CNNs, RNNs, Gated Recurrent Units (GRU), Multi-Layer Perceptron (MLP), and Long-Short Term Memory (LSTM), are the most widely adopted technologies in cryptocurrency trading. Numerous studies covered by the survey have investigated the use of ML and DL models in cryptocurrency trading, demonstrating that they can perform some level of modeling and accuracy for financial time series, including cryptocurrencies, and, more importantly, that they can improve trading performance relative to conventional technical trading strategies.

Another recent survey paper (Khedr et al., 2021) provides a thorough analysis of classical statistical models and ML methods for predicting cryptocurrency prices for research conducted between 2010 and 2020. The paper highlights the dominance of artificial neural networks (ANNs) and Bayesian regression in predicting Bitcoin price fluctuations due to their ability to learn the nonlinear relationship between input and output variables in cryptocurrency datasets. The survey emphasizes the predictive potential of DL approaches, particularly LSTM models, for cryptocurrency pricing. The authors argue that because LSTMs incorporate memory states, they are more effective at solving time-series prediction problems and recognizing long-term relationships by deleting unnecessary information from the network. The paper recommends future research on the use of LSTM models, such as CNN-LSTMs and bidirectional LSTMs (Bi-LSTMs). In addition to highlighting the limitations of conventional methodologies, the survey article advises investigating ensemble techniques and employing meta-optimization to ML techniques to increase accuracy.

In a more recent comprehensive study not covered by the surveys, Kang, Lee, and Lim (2022), propose a 1-Conv-CNN stacked over a 2-layers GRU model for the prediction of Bitcoin, Ethereum, and Ripple prices. The model is designed to smooth the data before passing it to the 2-layered GRU, hence improving its performance. The authors also compared the performance of their proposed model with other popular models for cryptocurrency price prediction, including Simple RNNs, LSTMs, ARIMA, XGBoost, Facebook Prophet, and Bi-LSTMs. The results showed that models of RNN-nature and the ARIMA model outperformed other models. However, the authors found no statistically significant difference between their proposed model modification and a simple 2-layered GRU, a Simple RNN, ARIMA, or a CNN-LSTM model. Notably, these models performed slightly better than a 2-layered LSTM and substantially better than popular models such as XGBoost, Prophet, and Bi-LSTMs. The findings highlight the effectiveness of RNN-based models and the ARIMA model for cryptocurrency time series forecasting, while also suggesting that a simple 2-layered GRU, or other analogous models, can achieve competitive performance in this domain.

In another recent research, Patra and Mohanty (2022) propose a three-stage GRU model to predict the prices of three cryptocurrencies (Bitcoin, Ethereum, and Dogecoin) over a 21-day forecast period. The data for the study was gathered from the Quandl marketplace, and many pre-processing techniques, including min-max normalization and the removal of missing time-series data, were applied to it. The authors report that the suggested model outperformed simple LSTM and GRU models in terms of Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The authors suggest that a hybrid model may be necessary to overcome the constraints of the proposed model for longer forecasting windows of six months to one year.

Finally, Tripathi and Sharma (2022) present a DL framework for Bitcoin price forecasting using both technical and fundamental indicators. The framework was evaluated at four distinct time intervals (1, 3, 5, and 7 days in advance), and the results were compared to those of previous benchmark studies. The authors employed three input types and four models, including ANNs, LSTM, Bi-LSTM, and CNN-LSTM. The model with the best performance was determined to be the ANN which leveraged technical indicators. The research findings show a significant improvement in Bitcoin price forecasting when using technical indicators instead of fundamental indicators. The significance of technical analysis on the Bitcoin market is highlighted by these results, which have ramifications for portfolio managers and algorithmic traders.

In summary, the literature on cryptocurrency time series forecasting highlights the dominance of ML and DL algorithms in predicting cryptocurrency prices. While previous studies have demonstrated the effectiveness of several models in cryptocurrency forecasting, there is still a need to investigate new models and strategies to increase the accuracy and dependability of these forecasts. Thus, the proposed modifications to the N-BEATS architecture in this study offer promising means of improving the precision and resilience of cryptocurrency price forecasting.

3 Proposed Architecture: N-BEATS Perceiver

In this section, we present a detailed description of the novel architecture, N-BEATS Perceiver, which we propose for cryptocurrency price forecasting. We describe the design and development of the architecture, including the procedures and techniques employed. To evaluate the effectiveness of the N-BEATS Perceiver, we conducted experiments using historical cryptocurrency price data, and compare its performance against existing methods. The subsequent subsections provide detailed accounts of the data used, N-BEATS architectures tested, heuristic optimization algorithms employed, and the experimental setup.

3.1 Transforming the N-BEATS Architecture

N-BEATS is a DL model for univariate time-series forecasting. It is composed of a fully connected network and a basis layer. The fully connected network predicts expansion coefficients both forward (forecast) and backward (backcast). The basis layer is composed of blocks which are organized into stacks using doubly residual stacking principle. The forecasts are then aggregated in a hierarchical fashion, forming a deep neural network with interpretable outputs.

This architecture has several advantages. Firstly, it is a pure DL approach, meaning that it does not rely on time-series specific feature engineering or input scaling. This makes it a flexible and adaptable approach which can be applied to a wide range of forecasting problems. Secondly, it is designed to be interpretable, allowing practitioners to understand and explain the model's outputs in terms of traditional decomposition techniques, such as the

"seasonality-trend-level" approach. Thirdly, the architecture is composed of basic building blocks, which can be easily customized and extended to suit specific needs. Lastly, the model uses ensembles to improve accuracy, allowing it to capture multiple scales of the data and learn more complex representations.

We explored multiple variations of the N-BEATS architecture in our experiments, including the use of convolutional network layers and Transformer mechanisms, as well as the Mish activation function. One advantage of using the Mish activation function instead of the ReLU activation function in N-BEATS is that Mish has a smoother gradient. This characteristic can improve the model's performance by allowing gradient descent to converge to the optimal solution more quickly and accurately. Another advantage of Mish is that it has a bounded output, meaning that the output of the activation function will always be contained within a specific range. This could prevent the model's output from becoming excessively large or small, which could lead to numerical instability and degradation in performance. In contrast, the ReLU activation function has a non-smooth gradient and can produce unbounded outputs, which can make the optimization process more difficult and can reduce the performance of the model. To use Mish as the activation function in N-BEATS, we can simply replace all instances of the ReLU function with the Mish function.

This will replace the ReLU activation function with the Mish activation function, and can improve the performance of the N-BEATS model. It is important to note that this may not always be the case, and the performance of the model will depend on the specific characteristics of the time series data and the model architecture. It is best to conduct experiments and compare the performance of the N-BEATS model using different activation functions in order to determine the best activation function for your specific case.

In addition to the Mish activation function, we also explored adding CNNs or Attention Encoders, specifically the Transformer and Perceiver Encoders, to the N-BEATS architecture in order to improve the accuracy of the model. As shown in Figure 1, which depicts our N-BEATS Perceiver model, we have implemented these modifications by adding layers between the input and the 4-layer MLP block in order to form a new block that includes an additional step of input pre-processing. In the case of the Transformer and Perceiver Encoders, the 4-layer MLP blocks act as a decoder of the information compressed by the encoders. We have implemented the native Transformer Encoder from the Pytorch library, the Pytorch implementation of the Perceiver Encoder from the Hugging Face library (Wolf et al., 2020), and the native Pytorch 1-dimensional CNN layer with a stride of three.

We hypothesize that the incorporation of CNNs into the N-BEATS architecture will improve its performance by allowing it to capture spatially local information, thereby allowing the model to learn more local patterns in the data. Additionally, we anticipate that the integration of attention encoders will enable the model to better focus on important parts of the data, and to learn more complex relationships between features. Through these modifications, we

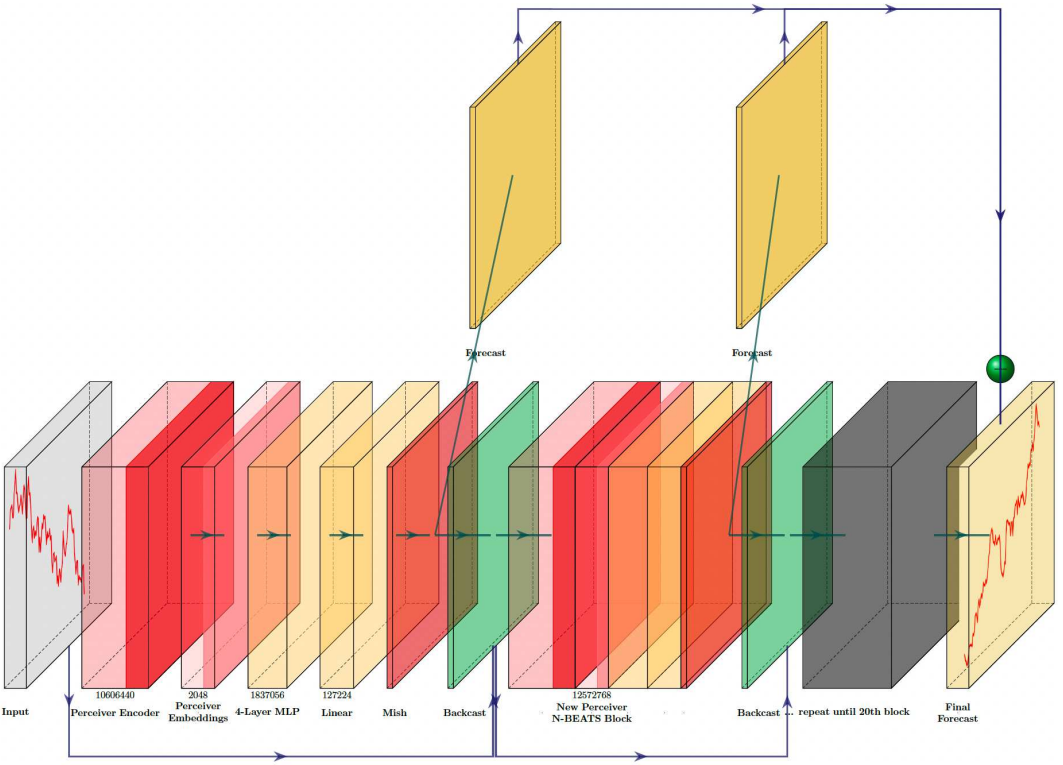

Fig. 1 Proposed Architecture for N-BEATS Perceiver Model. The figure depicts our proposed architecture for the N-BEATS Perceiver model, a deep neural network designed for time series forecasting. The network is composed of 20 N-BEATS blocks, each containing a Perceiver Encoder, Perceiver Embeddings, a 4-Layer MLP, a Linear layer, and a Mish activation function. The Perceiver Encoder layer is responsible for extracting features from the input time series, while the Perceiver Embeddings layer learns embeddings for the extracted features. The 4-Layer MLP acts as a decoder layer responsible for generating the Backcast and the Forecast, and the Linear layer applies a linear transformation to the input before the Mish activation function's final transformation of the outputs. The model contains 251,455,360 trainable parameters in total. All backcasts are transferred to the subsequent block, and all forecasts are added to produce the model's final forecast.

aim to enhance the versatility and performance of the N-BEATS model. We also aim to assess the trade-offs of computational complexity, by testing each model's scalability to different input sizes.

3.2 Interpreting Deep Learning Experiments with Shap Values

In this paper, we analyze the performance of the N-BEATS model with different combinations of architectures, loss functions, and activation functions. To interpret these results, we use Shapley Additive exPlanations (Shap Values) (Lundberg & Lee, 2017) to understand the individual contributions of each feature or variation to the model's predictions.

Shap Values are a method for interpreting and explaining the output of ML models by decomposing the model's predictions into the contributions of each individual feature. Decision trees are particularly useful for this task, as they can compute Shap Values by associating each split in the tree with a particular feature and computing the Shap Value for that feature as the difference in the prediction made by the tree before and after the split.

Using the decision tree model LightGBM (Ke et al., 2017), we trained a model with the different variations as features and required it to predict the expected forecasting error of the model. We were then able to compute Shap Values to gauge the contribution of each feature to the error. This allowed us to determine the unique contributions of each variation to the model's predictions, offering significant insight into which variations are most beneficial for a particular task. For instance, if the Shap Value for a particular loss function is large, it indicates that this loss function is a significant predictor of model performance. This could provide meaningful information into which variants are most effective for a given task and better guide our decisions when developing DL models.

3.2.1 Evaluation Metrics

It is essential, while evaluating the effectiveness of forecasting models, to employ metrics that are suited for the particular characteristics of the time series data and the objectives of the research, which are, in this case, the evaluation of a models ability to perform univariate point forecasting in discrete time. As a result, we have chosen, for our study, three of the most often employed loss functions for model training in forecasting: Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (sMAPE), and Mean Absolute Scaled Error (MASE).

MAPE measures the average error of a model as a percentage of the actual values. It is calculated as:

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{(Actual_i - Forecast_i)}{Actual_i} \tag{1}$$

where n is the number of forecasts, $Actual_i$ is the actual value of the i-th observation and $Forecast_i$ is the predicted value of the i-th observation. However, MAPE is not defined for cases where the actual value is zero, as it involves dividing by the absolute value of the actual value. This can be a problem if there are many zeros in the data, as MAPE will be undefined for cases in which a portfolio goes to zero.

sMAPE is similar to MAPE, but it is symmetric, meaning that it doesn't discriminate between positive and negative errors. It is calculated as:

$$sMAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{(Actual_i - Forecast_i)}{(Actual_i + Forecast_i)} \tag{2}$$

sMAPE is defined for all cases, even when the actual value is zero. However, sMAPE is generally considered to be more sensitive to outliers, as it involves dividing by the sum of the absolute values of the actual and forecast values.

To overcome this limitation, we also include the Mean Absolute Scaled Error (MASE) as a complementary metric. MASE is a scale-independent measure of forecast accuracy, and is calculated as:

$$MASE = \frac{1}{n}\sum_{i=1}^{n}\frac{(Actual_i - Forecast_i)}{MeanAbsoluteErrorofNaiveForecast} \tag{3}$$

where $MeanAbsoluteErrorofNaiveForecast$ is calculated as:

$$MeanAbsoluteErrorofNaiveForecast = \frac{1}{n}\sum_{i=1}^{n}(Actual_i - Actual_{i-1}) \tag{4}$$

MASE is useful for comparing different forecasting models in a relative sense, as it compares the forecast error to the errors that would be made by a naive forecast. It is a relative measure of forecast error, which means it is independent of the scale of the series, and it makes it particularly useful for comparing models when the data has different scales or units of measurement. Overall, using MASE as a metric allows us to accurately and intuitively compare the performance of different forecasting models in our study in a relative sense, and makes it particularly useful for comparing models in cases where the series has a trend or seasonal component.

For model performance comparison, in addition to these three metrics, we also use Root Mean Squared Scaled Error (RMSSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). While these metrics are not used as loss functions during the model training process, they provide valuable information for comparing the performance of different forecasting models. RMSSE, for instance, is particularly useful for comparing models when the series has a trend or seasonal component, while MAE and RMSE are widely used metrics for evaluating the accuracy of forecasts and provide a clear measure of the magnitude of the errors.

However, given the importance of accurately and intuitively comparing the performance of different forecasting models in our study, we have chosen to use sMAPE as our primary metric for evaluating the performance of models in much of the analysis of the results. This is because sMAPE is symmetric, easy to interpret, and defined for all cases, even when the actual value is zero. Additionally, the use of sMAPE allows us to understand the overall accuracy of the forecasting models, regardless of whether the errors are positive or negative.

4 Experimental Setup

In this study, we evaluate the performance of several time series forecasting models on hourly cryptocurrency portfolios with the objective of accurately

predicting their future values. To achieve this goal, we apply the problem of univariate point forecasting in discrete time to the specific case of univariate portfolio time series data of cryptocurrencies. In this section, we define the forecasting problem, describe the data collection and preprocessing methods used, and present the experimental designs implemented in our study.

4.1 Problem Definition

In our study, we focus on the problem of univariate point forecasting in discrete time for hourly cryptocurrency portfolios. The objective is to predict the future returns of a portfolio of cryptocurrencies given a historical series of length-T of hourly returns $[r_1, r_2, ..., r_T]$. Specifically, we seek to forecast the vector of future returns $r_{T+1:T+H}$, where H is the forecast horizon. To make the task more manageable, we use a lookback window of length $t = T$ as input to the forecasting models, denoted as $x_t = [r_{T-t+1}, ..., r_T]$. It's important to note that in this context, the observed series history $[r_1, ..., r_T]$ is indexed at 1, and the future returns $r_{T+1:T+H}$ are defined as a ratio of the future portfolio value to the indexed historical value of 1. This means that both past and future returns are expressed as a percentage change relative to the indexed value of 1. Although this problem shares similarities with the univariate point forecasting problem described in Oreshkin et al. (2020), our study focuses exclusively on portfolio time series data of cryptocurrencies with indexed values.

A sample time series demonstrating the problem is shown in Figure 2. The series fluctuates randomly until it reaches $r_t = 0$, where it has a locked value of 1. The task is to forecast the future returns relative to this indexed value of 1 over the next 48 hours.

4.2 Data Collection and Pre-processing

The dataset used in our experiments consists of historical cryptocurrency price data from Binance, the largest cryptocurrency exchange platform. We acquired the data through the Binance API (Binance, 2022) in minute format. The data was preprocessed so that, for each cryptocurrency, we calculated the volume-weighted average price (VWAP) using a 1-minute time series as input and outputting the VWAP hourly. The initial price, before the calculation of the volume, was calculated as the average of the Open, High, Low, and Close prices for each 1-minute time step. The VWAP at the hourly level is calculated as follows:

$$VWAP_i = \frac{\sum_{j=1}^{60} (\frac{open_j + high_j + low_j + close_j}{4} \cdot volume_j)}{\sum_{j=1}^{60} volume_j} \tag{5}$$

where $open_j$, $high_j$, low_j, $close_j$, and $volume_j$ are the minute-level prices of Open, High, Low, Close, and volume, respectively, for the j-th minute within the i-th hour. This formula calculates the hourly VWAP by first taking the average of the minute-level prices of Open, High, Low, and Close for each

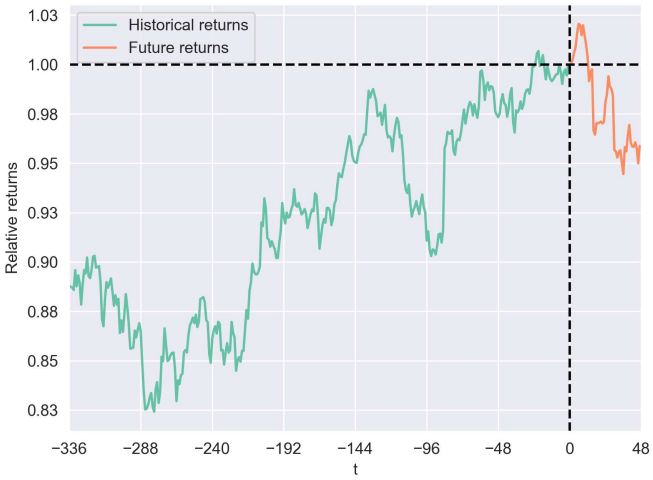

Fig. 2 Sample time series for forecasting. This figure demonstrates a sample time series of hourly cryptocurrency portfolio returns from the test period that ends at the indexed value of 1. The task is to forecast future returns relative to this indexed value of 1 over the next $H = 48$ hours, using a lookback window of $7H$ (equivalent to 336 hours). The horizontal and vertical dashed lines represent the point at which the value of the portfolio is indexed to one, which is the last observation.

minute within the hour, and then multiplying this average by the corresponding minute-level volume. The resulting products are summed over all minutes within the hour, and then divided by the total volume for the hour. This allows us to obtain a fair representation of the value of the cryptocurrency over time, taking into account both the price and volume of transactions.

The volume and prices were obtained from the volume of trade and prices in the main base cryptocurrencies, namely, $USDT$, USD, $USDC$, $BUSD$, BTC, ETH, BNB, which were all converted to USD. This was done in order to ensure that the VWAP accurately reflects the fair value of the cryptocurrencies, regardless of the denominations they were traded in.

To generate a large number of portfolio return time series, we developed a selection algorithm that ensured the portfolios consisted of assets that were traded throughout the entire life of the portfolio. This was done to avoid any scenarios of impossible portfolios, such as cases where certain currencies were de-listed. The resulting portfolio series were then utilized as input for both the time series forecasting and portfolio optimization experiments.

The training and test datasets were generated from historical data from the cryptocurrency exchange Binance. Specifically, we utilized portfolios beginning with Binance's earliest recorded datapoints on July 14, 2017, and continuing through June 30, 2022, to train and assess the performance of our models. All test data, on the other hand, was comprised of portfolios from July 1,

2022, through October 31, 2022, which we used to evaluate our models' generalization capabilities and performance on unseen data. Table 1 provides an overview of the specific periods and assets used in the analysis. By employing this temporal split and utilizing a diverse set of assets, we aimed to mitigate the risks of overfitting our models to the training data and to ensure their generalization ability to new data. This technique is a well-established practice in ML, particularly in financial applications where overfitting is a well-known issue (Lopez de Prado, 2018).

Table 1 Periods and Assets used in the study

Data	Period	Assets	Source
Train set	Jul 14, 2017 - Jun 30, 2022	500 unique tickers	Binance historical data
Test set	Jul 1, 2022 - Oct 31, 2022	387 unique tickers	Binance historical data

In both training and testing, all models were tasked with predicting the next 48 hours of the portfolio, called H. Different models would look at different lookback horizons ranging from $2H$ to $7H$, that is, looking on 96 hours of history up to 336 hours of history.

By using portfolios of real-world assets, we aimed to evaluate the performance of our methods in a more realistic setting and better understand their potential applications in practice. The results of this analysis are depicted in Figure 3, which presents a graphical representation of the distribution of values and returns for a sample of portfolios selected specifically for testing between July and October 2022.

Lastly, we would like to note that the train and test datasets used in this research have been published and made available in Sbrana (2023a) for reproducibility purposes.

4.3 Description of Experiments

We devised a series of experiments to evaluate the effectiveness of various forecasting techniques in the context of cryptocurrency portfolio selection. Our experiments were designed to assess the models' ability to produce accurate and robust forecasts for a wide range of portfolio possibilities, as well as their ability to select portfolios with the highest forecasted returns. We performed three experiments, which are described below: the Main Forecasting experiment, the Subsample Forecasting experiment, and the Evaluation of Forecasting Models through Portfolio Selection experiment.

4.3.1 Main Forecasting Experiment

In this experiment, we aim to forecast the returns of the portfolio over the next 48 hours, given the index value of 1 at hour zero. To provide benchmark comparisons for our DL models, we selected several methods from the Darts library (Herzen et al., 2022). These methods include four varieties of Naive

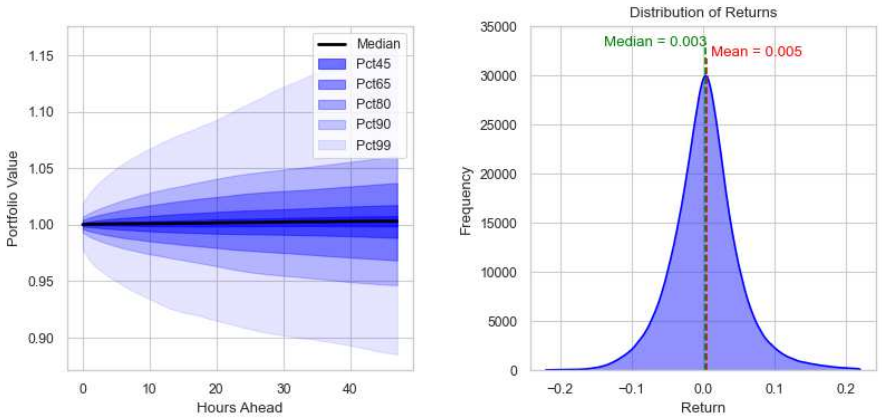

Fig. 3 Analyzing the Performance of the Test Portfolios and their Returns Distribution. The left chart illustrates the distribution of portfolio values for over 4,160,000 sampled test portfolios, depicted by a fan chart with percentiles at 45, 65, 80, 90, and 99. These portfolios were utilized to assess the effectiveness of our models in forecasting the portfolio's returns over the following 48 hours, with the value at hour zero serving as a reference point indexed at 1 (or 100%). The models are expected to predict the fluctuation of the portfolio's value with respect to hour zero. The right chart displays the distribution of returns for the same test portfolios, with the mean return of 0.005 and median return of 0.003, represented by the red and green dotted lines.

models, the Theta method, Croston's method, and a Fast Fourier Transform (FFT) model. These models were selected for benchmark comparison because they could provide a reasonable runtime response for the 4,160,000 portfolio forecasts that comprised the test set of our main experiment.

All other methods were DL models that we implemented ourselves. To further improve the performance of these models, we trained multiple versions of each model. Specifically, we trained 54 models per DL model, comprising of 6 types of lookbacks ($2H$ through $7H$), three types of loss functions (sMAPE, MAPE and MASE), and three versions of each of those combinations. This approach is similar to the one used in the original N-BEATS paper (Oreshkin et al., 2020), which repeated each of those experiments 10 times instead of three times while training the N-BEATS models. The reason for repeating the experiment multiple times is to avoid having "bad luck" from initialization and comparing noisy or unfortunate versions of each model, which would be unfair to the approach.

In recent years, researchers have proposed various optimization methods for training neural networks, recognizing the significant impact that such methods can have on performance. In our study, instead of the traditional Adam optimizer (Kingma & Ba, 2015), we adopt the Ranger optimizer (Wright, 2022), a state-of-the-art approach that combines two promising techniques: RAdam (Liu et al., 2020) and Lookahead (Zhang, Lucas, Ba, & Hinton, 2019). The RAdam approach incorporates a gradual warming up period to reduce the fluctuation in momentum during the initial training stages. Meanwhile, the

Lookahead technique involves searching for the best weight updates and then adjusting the current weights in that direction. This method aims to stabilize training by cleverly exploring the loss surfaces of neural networks. Both RAdam and Lookahead aim to decrease training variability and have been shown to improve convergence on various tasks.

In addition, we also implemented a "flat plus cosine" learning rate schedule, as shown in Figure 4. This strategy involves maintaining a constant learning rate for 70% of the epochs and gradually decreasing to a local minimum in the remaining 30%. The chosen starting rate for this study was 0.001.

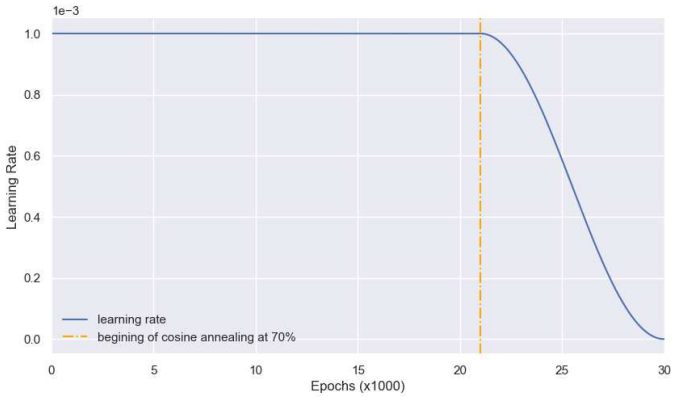

Fig. 4 Flat plus cosine learning rate schedule. The optimizer begins with a learning rate of 0.001 and maintains this pace for 21,000 iterations, gradually decreasing the rate until it reaches zero for the remaining iterations.

Table 2 shows the training parameters for the different DL models. The models were trained on randomly sampled portfolios at runtime from the training window between July 2017 and June 2022. Each model was assigned different epochs of training, which were determined by looking at train and validation curves within the training windows. The table illustrates an overview of the training parameters for each DL model, including the chosen number of epochs, batch size, number of training samples required for convergence, as well as the min-max range of training time, number of parameters, and memory allocation required for each model.

Most models were trained on 31 million randomly sampled portfolios with a batch size of 1,024. However, N-BEATS Transformer required half of the epochs and training samples to stop learning, and N-BEATS Perceiver required only 5,500 epochs of batches of 32, that is, 176,000 samples to stop learning. Additionally, N-BEATS Perceiver is a considerably larger model with close to 1GB of size, while models like N-HiTS, 1-Layer GRU and 1-Layer LSTM take up under 10MB of size. The complete data for the training parameters of the different DL models are presented in Table 2.

Table 2 Training Parameters and Characteristics for the DL models evaluated in this study. Min-max ranges depend on the size of lookback windows ($2H$ through $7H$, which affect model size.

Model	Epochs	Batch	Train samples	Train time (mins)	Params (M)	Memory (MB)
1-Layer GRU	30K	1,024	31M	[26, 36]	[1.0, 1.3]	[4, 5]
1-Layer LSTM	30K	1,024	31M	[12, 36]	[1.3, 1.8]	[5, 7]
2-Layer GRU	30K	1,024	31M	[13, 27]	[2.5, 2.9]	[10, 11]
2-Layer LSTM	30K	1,024	31M	[13, 28]	[3.4, 3.9]	[27, 29]
N-BEATS 1-Conv.	30K	1,024	31M	[44, 67]	[27, 35]	[105, 133]
N-BEATS Mish	30K	1,024	31M	[39, 69]	[27, 35]	[105, 133]
N-BEATS Orig.	30K	1,024	31M	[49, 65]	[27, 35]	[105, 133]
N-HiTS	30K	1,024	31M	[12, 29]	[1.0, 1.4]	[4, 5]
N-BEATS Transf.	15k	1,024	15M	[50, 67]	[42, 92]	[159, 349]
N-BEATS Perc.	5.5K	32	176K	[138, 206]	[248, 258]	[947, 983]

As shown in Figure 5, the scalability of the N-BEATS Transformer model is the worst among the models compared. The graph demonstrates how the memory utilization of the Transformer model increases steeply as the input size increases. This is further highlighted by the broken y-axis in the graph, which emphasizes the relative difference in memory utilization among the models. Furthermore, as seen in Figure 6, when the lookback period is increased from $2H$ to $7H$, that is, when the input size increases by a percentage of 250%, the increase in both the parameters and memory size of the N-BEATS Transformer model is a factor of 119%.

Conversely, the N-BEATS Perceiver has the best scalability profile, as seen in Figure 6. Although it has many more parameters and near 1GB in model size, it increases only 4% in size and parameters when the input size is scaled by 250%. This highlights the effectiveness of the N-BEATS Perceiver model in terms of scalability, despite the downside of having a larger model size and more parameters.

In conclusion, the Main Experiment provides a comprehensive evaluation of various DL models, comparing them to well-established forecasting methods. The combination of the Ranger optimizer and a "flat plus cosine" learning rate schedule proved to be an effective method for training these models, resulting in enhanced performance. By training multiple versions of each model and selecting the median forecast from 54 models as the final forecast of the ensemble, we were able to achieve a more accurate and robust prediction. The insights gained from this experiment will inform future investigations, including those aimed at exploring other optimization techniques and assessing the scalability of DL models on large datasets.

4.3.2 Subsample Forecasting Experiment

In addition to the main experiment, we also conducted a secondary experiment on a subsample of 10,000 test portfolios. We carried out this experiment

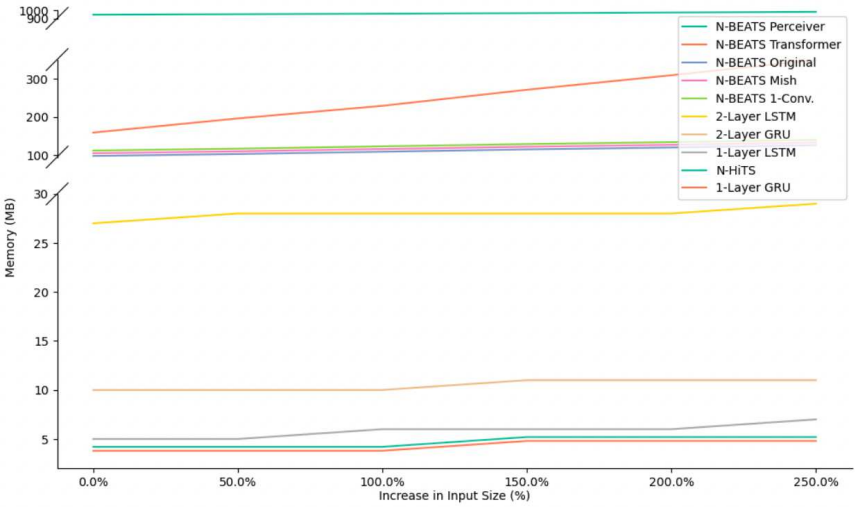

Fig. 5 Scalability of Model Memory Utilization with Increasing Input Size. This graph demonstrates the relationship between memory utilization and input size for various ML models. The x-axis represents the increasing input size, as measured by percentage increases in the lookback parameter from $2H$ (0%) to $7H$ (250% increase in input size). The y-axis, displayed in logarithmic scale, illustrates the memory utilization of each model. The broken y-axis emphasizes the relative difference in memory utilization among the models.

to compare the performance of a broader range of forecasting methods, including traditional statistical and ML methods, which do not have vectorized or parallelized versions that can efficiently handle the large scale test set of the main experiment.

The subsample was selected randomly from the over 4 million test portfolios, ensuring that the distribution of portfolio values and returns remained representative of the entire test set. The choice of using 10,000 samples as a subsample was made based on the statistical power and computational efficiency. A sample size of 10,000 is large enough to provide a good representation of the population, as per the central limit theorem, while being small enough to allow for efficient computation of the models.

In this secondary experiment, we used all time series methods available in the Pycaret library (Ali, 2020), in addition to the same methods from the Darts library that were used in the main experiment. The results of this experiment provide a comprehensive comparison of various forecasting methods in a more computationally feasible setting, and serve as a complementary analysis to the main experiment, while allowing us to make inferences about the population of portfolios with a high degree of accuracy and precision.

4.3.3 Evaluating Forecasting Models through Portfolio Selection

The main goal of this study is to evaluate the effectiveness of various forecasting models in the context of portfolio selection in real-world settings. To

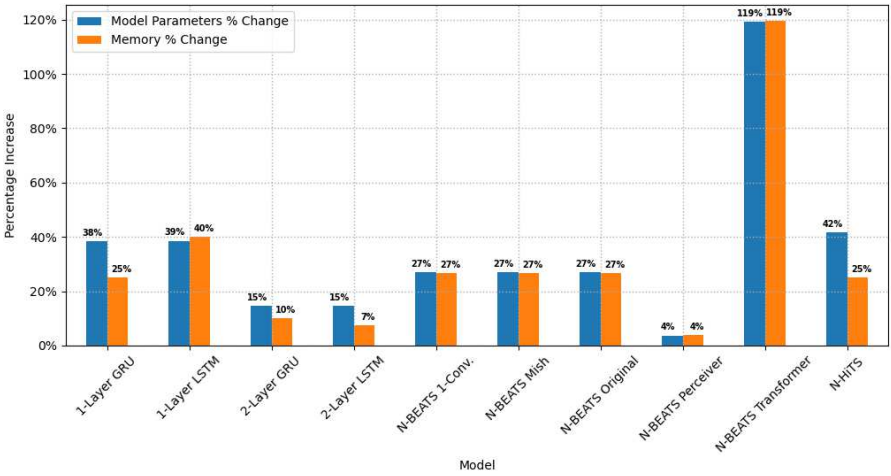

Fig. 6 Scalability of Models in terms of Parameters and Memory. Comparison of the percentage increase in model parameters and memory for different models when the lookback period is increased from $2H$ to $7H$, that is, when the input size increases by a percentage of 250%.

accomplish this, we designed an additional experiment in which we applied a diverse set of forecasting models to a large number of portfolio options, generating forecasts for each one. We then selected the portfolios with the highest expected returns based on these forecasts, as a way to evaluate the performance of the models in the context of portfolio selection. This approach allowed us to test the ability of the models to accurately predict future returns and make effective investment decisions, which is a key consideration in the real-world application of these models.

The experiment was carried out in two phases. In the first phase, we used the models to forecast the returns of 100,000 portfolio options on any given hour for 102,635 rounds. These portfolios were selected from the test dataset, and were representative of the distribution of portfolio values and returns in the test set. In the second phase, we used a portfolio selection algorithm to choose the portfolio with the highest expected return in each round.

In order to thoroughly evaluate the performance of the various models in the portfolio selection experiment, we employed a comprehensive set of portfolio selection performance criteria, including mean return, standard deviation, minimum and maximum returns, downside deviation, maximum drawdown, and various risk-adjusted return measures. To further analyze the models' relative performance, we also utilized the Multi-Criteria Decision Making (MCDM) method of Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) as described in Uzun et al. (2021). This technique, which is a part of analytical multi-criteria decision-making, involves ranking the models based on their proximity to the positive and negative ideal solutions, which are formed as a combination of the best and worst points of each criterion.

This method is only applicable for numerical data sets and and offers a comprehensive evaluation of the models' performance by comparing each model's performance to the best achievable performance among the pool of methods being evaluated.

The results of this experiment offer a thorough examination of multiple forecasting techniques in the context of portfolio selection, and serve as a practical complement to the primary experiment, as the models are being evaluated on their ability to generate actual profits in real-world settings.

5 Results

The results of our experiments to assess the efficacy of various forecasting techniques in the context of cryptocurrency portfolio selection are presented in this section. As described in the previous section, we conducted three experiments: the main forecasting experiment, the subsample forecasting experiment, and the portfolio selection experiment.

5.1 Main Forecasting Results

The main forecasting experiment evaluated a diverse set of forecasting methods, including DL models and established traditional methods, in the context of cryptocurrency portfolio selection on a large-scale test set of 4,160,000 portfolios. We trained multiple versions of each DL model, and used the median forecast from those 54 models as the final forecast of the ensemble.

Our results, shown in Table 3, indicate that the N-BEATS Perceiver achieved the best performance, with a sMAPE of 2.4964%. This represents a significant improvement over the N-BEATS Original (2.7641%), and the N-BEATS Perceiver model also performed better than all other models in the remaining metrics.

Figure 7, illustrates the variation in sMAPE across different N-BEATS architectures. N-BEATS Perceiver architectures has the lowest median sMAPE, as well as the lowest interquartile ranges, making it the most accurate.

Figure 8 presents a comparison of the original N-BEATS models using ReLU and Mish activations. The results demonstrate that the use of Mish activation yields lower sMAPE values than ReLU activation, indicating a more accurate prediction of sMAPE. Furthermore, the box plot analysis shows that Mish activation has a smaller median sMAPE and narrower interquartile range than ReLU activation, indicating a more consistent performance across different scenarios.

Figure 9 presents a comparison of sMAPE across N-BEATS architectures and lookback windows. The results indicate that the N-BEATS Perceiver architecture had a lower sMAPE across all lookback windows than other architectures, with no significant difference across different lookback horizons. Additionally, the figure illustrates the trade-off between lookback window size and sMAPE for different architectures, with the N-BEATS Transformer's

Table 3 Comparison of ensemble methods for cryptocurrency portfolio forecasting on the full test set.

Method	MASE	RMSSE	MAE	RMSE	MAPE %	sMAPE %
N-BEATS Perceiver	**5.1530**	**4.1134**	**0.0248**	**0.0294**	**2.4656**	**2.4679**
1-Layer GRU	5.1946	4.1497	0.0251	0.0297	2.4910	2.4964
1-Layer LSTM	5.2060	4.1585	0.0252	0.0298	2.4975	2.5034
2-Layer LSTM	5.2153	4.1648	0.0253	0.0299	2.5032	2.5093
N-BEATS Transf.	5.2398	4.1857	0.0253	0.0299	2.5044	2.5104
2-Layer GRU	5.2244	4.1721	0.0253	0.0299	2.5067	2.5128
N-BEATS 1-Conv.	5.3213	4.2478	0.0256	0.0303	2.5392	2.5456
N-HiTS	5.3577	4.2751	0.0259	0.0306	2.5634	2.5723
Naive Drift	5.3593	4.2619	0.0259	0.0305	2.5658	2.5734
N-BEATS Mish	5.4066	4.3119	0.0260	0.0307	2.5759	2.5842
Theta Method	5.4875	4.3444	0.0264	0.0310	2.6183	2.6277
Croston's Method	5.6415	4.4004	0.0272	0.0314	2.6953	2.7031
N-BEATS Original	5.7627	4.5711	0.0278	0.0326	2.7507	2.7641
Naive Ensemble	5.8877	4.6438	0.0283	0.0331	2.8087	2.8163
Naive Seasonal	6.4891	5.0766	0.0314	0.0364	3.1138	3.1216
Naive Mean	11.2237	7.9987	0.0548	0.0580	5.4483	5.4234
Fast Fourier Transf.	16.1153	11.4183	0.0799	0.0840	7.9471	7.8746

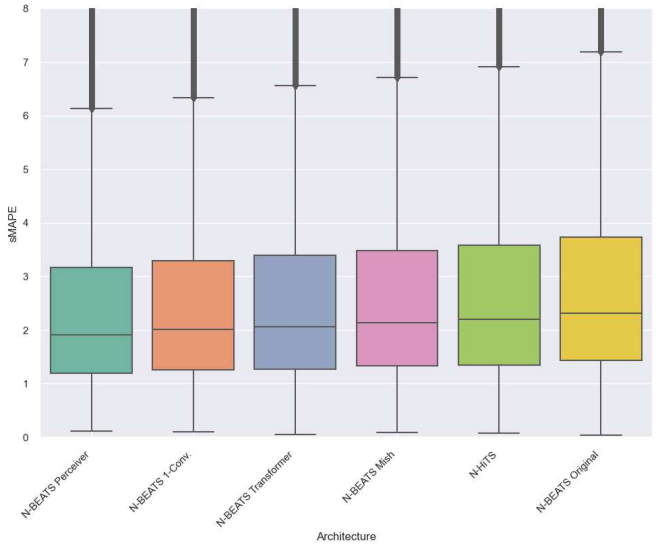

Fig. 7 Box plot Representation of sMAPE Across N-BEATS Architectures. Each box plot summarizes the distribution of sMAPE values for a single architecture, showing the median, quartiles, and outliers. The architectures are sorted by median sMAPE, lowest first. The x-axis indicates architectures and the y-axis sMAPE values.

longer lookback windows outperforming its shorter ones, while the opposite was true for all remaining architectures.

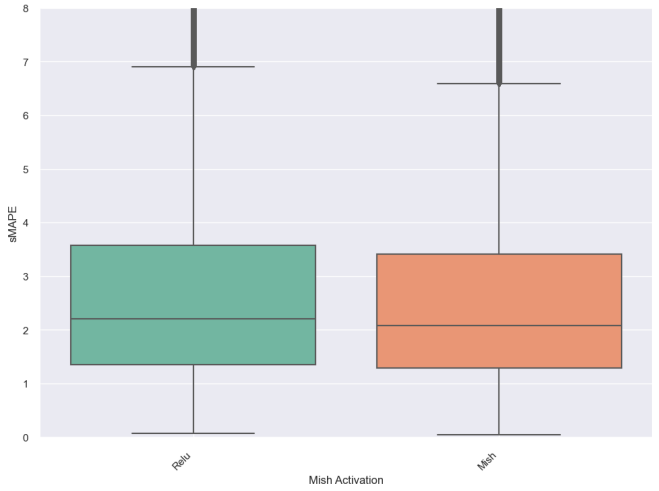

Fig. 8 Comparison of the original N-BEATS models using ReLU and Mish Activation. The box plot of the variation in sMAPE for N-BEATS using ReLU and Mish activation are presented. The x-axis depicts the activation functions and the y-axis shows the sMAPE values. The box plots depict the median, quartiles, and outliers of each activation function's sMAPE. The results confirm that the use of Mish activation leads to more accurate predictions and more consistent performance across different scenarios.

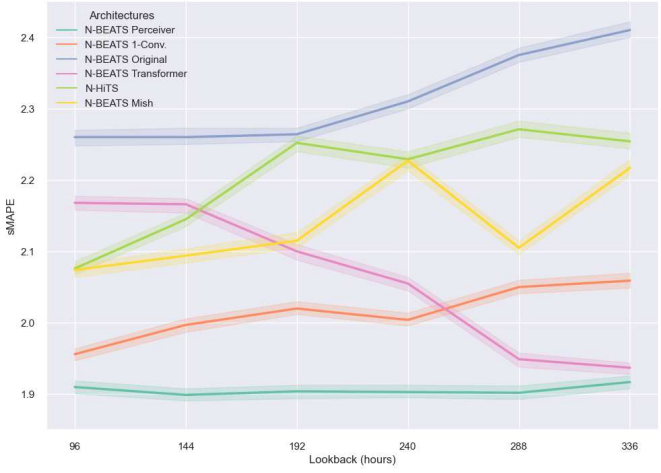

Fig. 9 Comparison of sMAPE across N-BEATS Architectures and Lookback Windows. The line chart displays the median sMAPE for N-BEATS models across architectures and lookback windows. The x-axis shows lookback windows, in hours, and the y-axis sMAPE values. Colored lines show each architecture's median sMAPE, with 99.99% confidence intervals, colored around the median. The N-BEATS Transformer's longer lookback windows are shown to outperform its shorter ones, while the opposite is true for all remaining architectures.

Figure 10, presents a comparison of sMAPE across N-BEATS architectures and loss functions during training. The figure shows that the N-BEATS Perceiver architecture had a lower sMAPE across all loss functions than other architectures, but with no significant difference across loss functions.

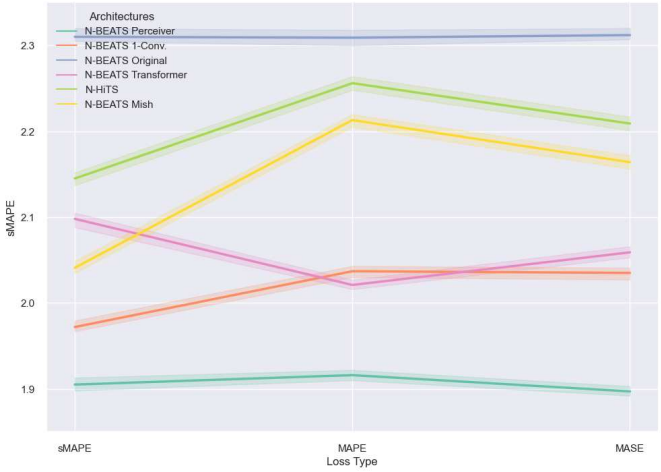

Fig. 10 Comparison of sMAPE across N-BEATS Architectures and Loss Functions. The line chart displays the median sMAPE for N-BEATS models across architectures and loss functions. The x-axis shows the loss functions and the y-axis sMAPE values. Colored lines show each architecture's median sMAPE, with 99.99% confidence intervals, colored around the median.

A LightGBM model was trained on the data to estimate the sMAPE performance of a model based on its characteristics of architecture, lookback, loss function and Mish activation. A SHAP analysis of the LightGBM model, presented in Figure 11, provides insight into the distribution of the top 10 most important features that impact the sMAPE estimation. The figure illustrates the impact of the various characteristics used in modeling. The examination of the figure reveals that the N-BEATS Original architecture, represented by the red dots in the feature line, has a positive marginal contribution to the estimation of sMAPE, since the red dots are to the right of the line of zero impact, indicating higher errors. Conversely, the N-BEATS Perceiver and N-BEATS 1-Conv. architectures have a negative marginal contribution to the estimation of sMAPE, indicating lower errors, since the red dots are to the left of the zero impact line. Additionally, the presence of Mish activation in the blocks tends to have a negative marginal contribution to the error, signifying that models with Mish activation tend to have lower errors. Furthermore, when the objective function minimizes MAPE, the errors tend to be higher for the sMAPE metric. However, when the objective function is MASE, there is a slight bias towards a lower sMAPE error, possibly because MAPE is non-symmetric.

5.2 Subsample Forecasting Results

In the subsample forecasting experiment, we applied a broader range of forecasting methods to a subsample of 10,000 test portfolios, in order to compare their performance. The aim of this experiment was to compare the performance of a broader range of methods with those of the main experiment. Our results, presented in Table 4, indicate that the N-BEATS Perceiver model

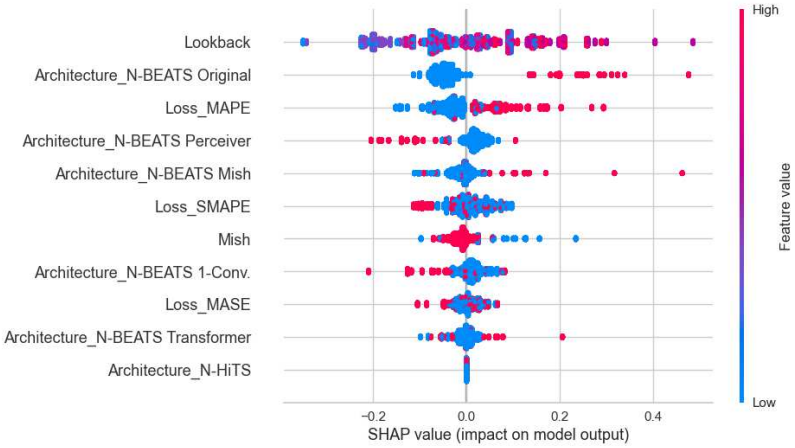

Fig. 11 SHAP Analysis of LightGBM Model for sMAPE Error Estimation. The figure illustrates the distribution of SHAP values for the top 10 most important features, providing insight into the model's error estimation based on various characteristics such as architecture, lookback, loss function, and Mish activation. The red dots represent that the feature is present or it is high, and the blue dots represent that the feature is not present or it is low. The x-axis indicates the marginal impact of the feature samples on the LightGBM model's predictions. The dots located to the left of the zero impact line indicate that the feature improves the model's predictions by decreasing the predicted error.

achieved the best performance across all metrics, including MASE, RMSSE, MAE, RMSE, MAPE, and sMAPE. These results are consistent with those of the main experiment, whereby none of the rankings of the original 4 million experiment changed.

5.3 Portfolio Selection Results

In this section, we present the results of our portfolio selection experiment. Firstly, we present the results of each portfolio selection in the form of a heatmap in Figure 12. The chart shows the return correlation of different investment strategies, as determined by choosing a winning portfolio out of 100,000 options on any given hour for 102,635 rounds. The heatmap highlights the lack of correlation of N-BEATS Perceiver and Theta Method with other methods, the high correlation between LSTM and GRU models, and the moderate correlation between N-BEATS Transformer and other N-BEATS variants and FFT. Notably, N-BEATS Mish and N-BEATS 1-Conv also have a particularly high correlation of 0.4.

Next, we present a visualization of the return characteristics of the different models in Figure 13. The figure shows the returns from a portfolio selection task where the goal was to choose the portfolio with the highest expected return out of 100,000 options, conducted over 102,635 rounds. N-BEATS Perceiver is the most consistent model with the narrowest range of returns. Other models, such as N-HiTS and N-BEATS Transformer, have higher median

Table 4 Comparison of ensemble methods for cryptocurrency portfolio forecasting on the 10,000 subsample test set.

Method	MASE	RMSSE	MAE	RMSE	MAPE %	sMAPE %
N-BEATS Perceiver	**5.1224**	**4.0871**	**0.0246**	**0.0291**	**2.4398**	**2.4436**
1-Layer GRU	5.1652	4.1245	0.0249	0.0295	2.4650	2.4717
1-Layer LSTM	5.1773	4.1336	0.0250	0.0295	2.4717	2.4789
2-Layer LSTM	5.2110	4.1599	0.0250	0.0296	2.4777	2.4853
N-BEATS Transf.	5.1877	4.1408	0.0250	0.0296	2.4779	2.4853
2-Layer GRU	5.1970	4.1481	0.0251	0.0296	2.4821	2.4895
Naive (Pycaret)	5.1743	4.1828	0.0252	0.0296	2.5136	2.5093
N-BEATS 1-Conv.	5.2890	4.2205	0.0254	0.0300	2.5126	2.5206
N-HiTS	5.3278	4.2486	0.0256	0.0303	2.5380	2.5482
Naive Drift	5.3308	4.2359	0.0257	0.0303	2.5418	2.5508
N-BEATS Mish	5.3763	4.2857	0.0258	0.0305	2.5506	2.5606
Theta Method	5.4819	4.3295	0.0263	0.0308	2.6021	2.6137
Croston's Method	5.6351	4.3877	0.0271	0.0313	2.6789	2.6895
N-BEATS Original	5.7287	4.5414	0.0275	0.0323	2.7206	2.7357
Auto ARIMA	5.7147	4.5936	0.0280	0.0328	2.7958	2.7914
Naive Ensemble	5.8754	4.6270	0.0282	0.0329	2.7881	2.7988
ARIMA	5.7915	4.6027	0.0283	0.0328	2.8192	2.8084
ETS	6.0522	4.8477	0.0298	0.0349	2.9757	2.9642
Exp. Smooth.	6.1704	4.9362	0.0302	0.0354	3.0191	3.0078
Gradient Boosting	6.1288	4.8979	0.0304	0.0353	3.0334	3.0280
K Neighbors	6.1510	4.9104	0.0305	0.0354	3.0461	3.0399
AdaBoost	6.2019	4.9375	0.0308	0.0357	3.0761	3.0704
Light Grad. Boost.	6.2711	4.9806	0.0311	0.0359	3.1070	3.1003
Random Forest	6.2808	5.0044	0.0312	0.0362	3.1137	3.1079
Naive Seasonal	6.4860	5.0682	0.0313	0.0363	3.0994	3.1107
Huber	6.3602	5.1006	0.0317	0.0369	3.1681	3.1458
Orth. M. Pursuit	6.7503	5.3640	0.0339	0.0392	3.3802	3.3615
Linear	6.7503	5.3640	0.0339	0.0392	3.3802	3.3615
Bayesian Ridge	6.7589	5.3699	0.0339	0.0393	3.3847	3.3664
Extra Trees	6.8119	5.4556	0.0340	0.0397	3.3930	3.3859
Decision Tree	7.2430	5.7841	0.0363	0.0422	3.6187	3.6124
Ridge	8.9369	6.6829	0.0451	0.0491	4.5025	4.4881
Polynomial Trend	8.9632	6.6978	0.0453	0.0493	4.5205	4.5057
Elastic Net	8.9661	6.6997	0.0453	0.0493	4.5221	4.5072
Lasso	8.9661	6.6997	0.0453	0.0493	4.5221	4.5072
Grand Means Forec.	9.3548	6.8515	0.0468	0.0502	4.6571	4.6360
Naive Mean	11.3396	8.0544	0.0552	0.0583	5.4824	5.4648
Fast Fourier Transf.	16.3439	11.5562	0.0808	0.0850	8.0436	7.9840

returns but are also associated with a relatively narrow range of returns and have more significant downside events compared to N-BEATS Perceiver.

To further investigate the performance of the different models, we present a table of various performance metrics in Table 5. This table includes metrics such as mean, standard deviation, minimum, 25th percentile, 50th percentile (median), 75th percentile, maximum, downside deviation, maximum drawdown, Sharpe ratio, Sortino ratio, Calmar ratio, Ulcer index, negative outliers, average loss, conditional tail ratio, Value at Risk (95%), Value at Risk (99%), Cond. VaR (95%), Cond. VaR (99%), Expected Shortfall (95%), Expected

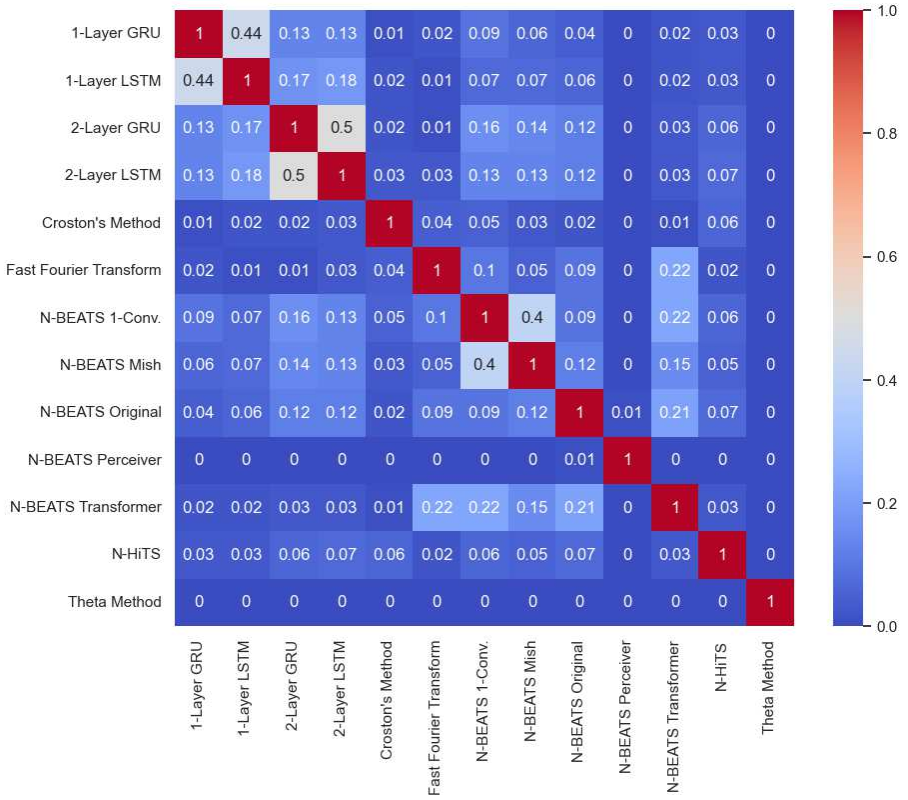

Fig. 12 Comparative Analysis of Model Return Correlation. The chart shows the return correlation of different investment strategies, as determined by choosing a winning portfolio out of 100,000 options for 102,635 rounds. Models with high return correlation are likely to pick similar winning portfolios on each round of the experiment.

Shortfall (99%), and Coherent VaR (95%). An overview of Expected Shortfall and Coherent risk metrics can be found in Chan and Nadarajah (2019); Tzagkarakis and Maurer (2022).

Finally, in 14, the results of a Multi-Criteria Decision Making (MCDM) method called TOPSIS are presented. This method is used to compare the relative closeness scores of various investment strategies, with the highest score indicating the top performing strategy. The chart displays the scores for each strategy, with the criteria used for the analysis listed in Table 5. This figure provides a clear comparison of the performance of different model selection strategies, allowing for an informed evaluation of their portfolio selection abilities.

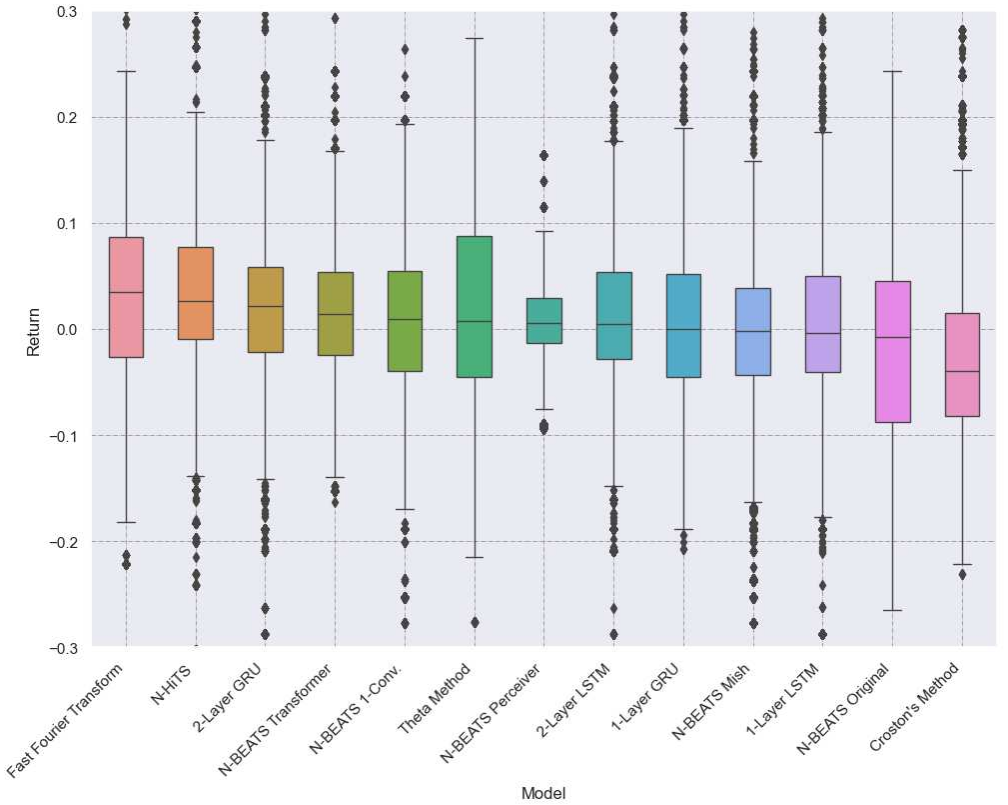

Fig. 13 Comparison of Model Returns in a Portfolio Selection Task. This box plot data presents the returns of the portfolio selection task conducted over 102,635 rounds and is ordered by the median return, from the highest to the lowest.

6 Discussion

This section discusses the outcomes of the three experiments conducted to examine the performance of time series forecasting models in the context of cryptocurrency price forecasting and portfolio selection. Specifically, we used historical price data from Binance, the largest cryptocurrency exchange platform, to calculate the volume-weighted average price (VWAP) utilizing the whole dataset of Binance's 1-minute pricing history. We then generated a large number of hourly portfolio series using a selection algorithm and used them as input for both the time series forecasting experiments and the portfolio optimization experiments.

In the main forecasting experiment, a diverse set of forecasting methods were evaluated in over 4 million portfolio test samples for the task of forecasting the next 48 hours. These experiments included DL models and established traditional methods as benchmarks. The N-BEATS Perceiver model achieved the best performance in all evaluation metrics and its box plot of errors showed

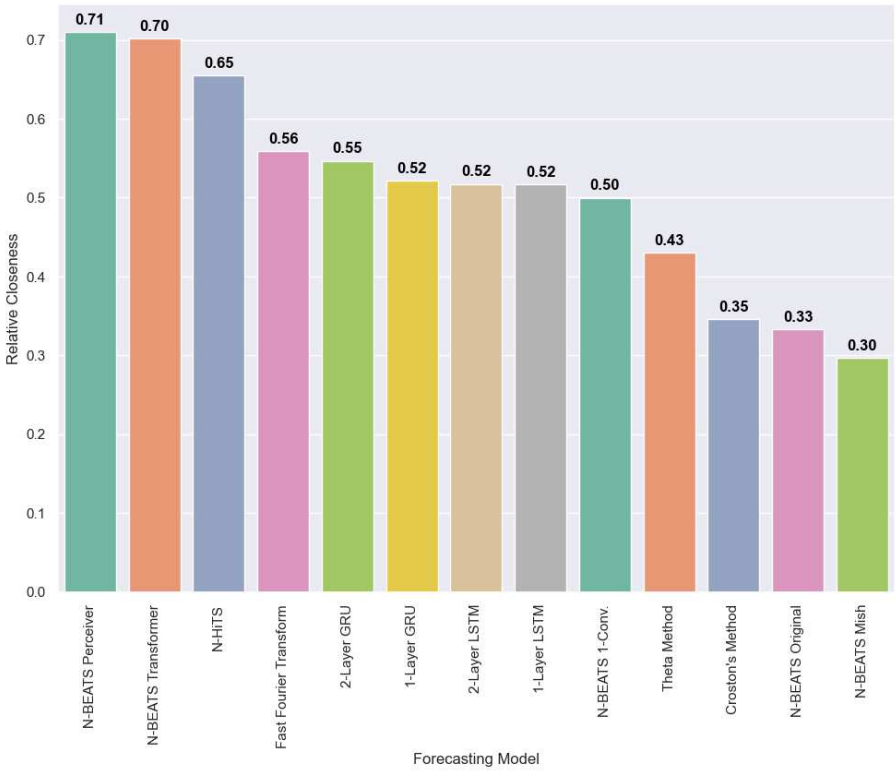

Fig. 14 Comparison of Models using TOPSIS Method. The chart shows the relative closeness scores for each investment strategy, with the highest score indicating the top strategy according to the Multi-Criteria Decision Making (MCDM) method of TOPSIS. The criteria used for the analysis are those depicted in Table 5: Mean, Standard deviation, Minimum, 25th percentile, 50th percentile (Median), 75th percentile, Maximum, Downside deviation, Maximum drawdown, Sharpe ratio, Sortino ratio, Calmar ratio, Ulcer index, Negative outliers, Average loss, Conditional tail ratio, Value at Risk (95%), Value at Risk (99%), Cond. VaR (95%), Cond. VaR (99%), Expected Shortfall (95%), Expected Shortfall (99%), Coherent VaR (95%).

superior performance compared to competing N-BEATS architectures, as seen in Figure 7. Specifically, the box plot depicts each architecture's sMAPE median, quartiles, and outliers, with N-BEATS Perceiver exhibiting the lowest median sMAPE. This performance difference was also observed in the subsample forecasting experiments where N-BEATS Perceiver presented the lowest error metrics across the board, as shown by Table 4.

The study also found that the simple addition of the Mish activation to the Original N-BEATS model significantly improved its performance, as seen in Figures 7 and 8. The SHAP Value analysis using the LightGBM model supported these findings and indicated that the Mish activation performed better than ReLU, and that certain architectures performed better than others.

Additionally, models trained with a MAPE loss function did not perform well in their sMAPE performance.

The final experiment of our research consisted of analyzing the portfolio selection capabilities of each model. The outstanding performance of the N-BEATS Perceiver model in the portfolio selection experiment was highlighted by its narrow distribution of returns and minimal correlation with other models, as illustrated in Figures 13 and 12. This resulted in a strong risk profile for the model, with less downside compared to other models, as evident in Table 5. Furthermore, the model performed exceptionally well when evaluated using the TOPSIS method, leading the pack with the N-BEATS Transformer and N-HiTS architectures closely following, as shown in Figure 14. Although the N-BEATS Perceiver model demonstrated a lower median return compared to some other models, the N-BEATS Transformer and N-HiTS models, despite their higher downside events, provided robust risk-reward alternatives with higher median returns. These models can potentially be integrated in portfolio selection tasks as they exhibit little return correlation, as indicated by our correlation analysis.

Our findings revealed that the N-BEATS Perceiver model outperformed other models with significantly less training samples and epochs. In addition, while the N-BEATS Perceiver model has a bigger model size and more parameters than other models, it exhibits the best scalability profile among the compared models, with a size and parameter increase of only 4% when the input size is increased by 250%. Despite a higher model size, the N-BEATS Perceiver's scalability demonstrates its success in its capacity to manage larger inputs and outputs. In contrast, the N-BEATS Transformer model exhibited poor scalability, with a sharp increase in memory use as input size rose and a 119% percent increase in size and parameters when input size was expanded by 250%.

7 Conclusion

This study performs an in-depth assessment of various time series forecasting models for cryptocurrency portfolio selection, with a specific focus on the N-BEATS Perceiver model. The study's emphasis on the volatile nature of cryptocurrencies and the availability of over 500 cryptocurrencies for portfolio construction highlights its significant and relevant contribution to both quantitative finance and financial technology, and provides valuable insights for developing trading strategies and risk management techniques in these fields. Additionally, the practical relevance of the study is enhanced by the use of a real-world dataset from Binance, the largest cryptocurrency exchange platform, and by the vast size of the test dataset, which includes over 4 million portfolio test samples.

While the study may have a focus on historical data from a single asset class and hourly time series, it still provides a comprehensive evaluation of

time series forecasting models in the context of cryptocurrency asset allocation. The outstanding performance of the N-BEATS Perceiver model in both time series forecasting and portfolio selection experiments demonstrated its potential utility in real-world applications, including cryptocurrency trading strategies and economic outcomes.

Future research should explore more advanced portfolio selection methods beyond the random selection of 100,000 portfolios used in this study. Techniques such as genetic algorithms and Bayesian optimization could be employed to improve the selection process. Combining optimization and search methods with forecasting models in a synergistic approach may lead to even better results. In addition, it would be valuable to investigate the economic benefits of more accurate portfolio construction methods and their potential impact on real-world financial outcomes.

Acknowledgments. *Fundação de Amparo a Pesquisa do Estado de São Paulo* (FAPESP - Grant 2022/01524-2).

Declarations

- **Funding.** This study was supported by the *Fundação de Amparo a Pesquisa do Estado de São Paulo* (FAPESP - Grant 2022/01524-2).
- **Conflict of interest/Competing interests.** The authors have no relevant financial or non-financial interests to disclose.
- **Availability of data and materials.** The Binance Portfolio Forecasting Hourly VWAP Dataset used in this study is available in the OSF repository referenced in Sbrana (2023a). Researchers can access the data for reproducibility purposes under the CC-By Attribution 4.0 International license, which is specified in the repository.
- **Code availability.** The source code for the N-BEATS Perceiver model and other models used in this study is accessible from the GitHub repository cited in Sbrana (2023b). The code is open-source and available under the MIT License, which is specified in the repository. Detailed instructions on how to run the code are provided in the repository's README file.
- **Authors' contributions.** Both authors contributed to the study's conception and design. Material preparation, data collection and analysis were performed by Attilio Sbrana. The first draft of the manuscript was written by Attilio Sbrana and Paulo André Lima de Castro commented on previous versions of the manuscript. Both authors read and approved the final manuscript.

References

Ali, M. (2020). *Pycaret: An open source, low-code machine learning library in python.* 2.3.10. GitHub. Retrieved from https://www.pycaret.org .

Binance (2022). *Binance vision.* 1.0. GitHub. Retrieved from https://data.binance.vision .

Challu, C., Olivares, K.G., Oreshkin, B.N., Garza, F., Mergenthaler-Canseco, M., Dubrawski, A. (2022). N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting. *arXiv preprint*. https://arxiv.org/abs/2201.12886

Chan, S., & Nadarajah, S. (2019). Risk: An R Package for Financial Risk Measures. *Computational Economics*, *53*(4), 1337–1351. https://doi.org/10.1007/s10614-018-9806-9

Fang, F., Ventre, C., Basios, M., Kanthan, L., Martinez-Rego, D., Wu, F., Li, L. (2022). Cryptocurrency trading: a comprehensive survey. *Financial Innovation*, *8*(1), 13. https://doi.org/10.1186/s40854-021-00321-6

Herzen, J., Lässig, F., Piazzetta, S.G., Neuer, T., Tafti, L., Raille, G., ... Grosch, G. (2022). Darts: User-friendly modern machine learning for time series. *J. Mach. Learn. Res.*, *23*(124), 1-6. https://arxiv.org/abs/2110.03224

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., ... Carreira, J. (2021). Perceiver IO: A General Architecture for Structured Inputs & Outputs. *arXiv preprint*. https://arxiv.org/abs/2107.14795

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., Carreira, J. (2021). Perceiver: General Perception with Iterative Attention. *arXiv preprint*. https://arxiv.org/abs/2103.03206

Kang, C.Y., Lee, C.P., Lim, K.M. (2022). Cryptocurrency Price Prediction with Convolutional Neural Network and Stacked Gated Recurrent Unit. *Data*, *7*(11). https://doi.org/10.3390/data7110149

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st Conference on Neural Information Processing Systems, NIPS 2017* (pp. 3146–3154).

Khedr, A.M., Arif, I., P V, P.R., El-Bannany, M., Alhashmi, S.M., Sreedharan, M. (2021). Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey. *Intelligent Systems in Accounting, Finance and Management*, *28*(1), 3-34. https://doi.org/10.1002/isaf.1488

Kingma, D.P., & Ba, J.L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015*,

1–15. https://arxiv.org/abs/1412.6980

LeCun, Y., & Bengio, Y. (1998). Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks*, 255–258. MIT Press. https://doi.org/10.5555/303568.303704

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J. (2020). On the variance of the adaptive learning rate and beyond. *8th Int. Conf. on Learning Representations, ICLR 2020*. https://arxiv.org/abs/1908.03265v4

Lopez de Prado, M. (2018). Cross-Validation in Finance. *Advances in Financial Machine Learning*, 103–111. Wiley Publishing. https://doi.org/10.2139/ssrn.3266136

Lundberg, S., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *arXiv preprint*. https://arxiv.org/abs/1705.07874

Misra, D. (2020). Mish: A Self Regularized Non-Monotonic Activation Function. *31st British Machine Vision Conference 2020, BMVC 2020*, 1–13. https://arxiv.org/abs/1908.08681

Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *8th Int. Conf. on Learning Representations, ICLR 2020*. https://arxiv.org/abs/1905.10437

Patra, G.R., & Mohanty, M.N. (2022). Price Prediction of Cryptocurrency Using a Multi-Layer Gated Recurrent Unit Network with Multi Features. *Computational Economics*. https://doi.org/10.1007/s10614-022-10310-1

Sbrana, A. (2023a). Binance Portfolio Forecasting Hourly VWAP Dataset. OSF. https://osf.io/fjsuh/. https://doi.org/10.17605/OSF.IO/FJSUH

Sbrana, A. (2023b). N-BEATS Perceiver. GitHub. https://github.com/attiliosbrana/N-BEATS_Perceiver. https://doi.org/10.5281/zenodo.7668405

Sbrana, A., Debiaso Rossi, A.L., Coelho Naldi, M. (2020). N-BEATS-RNN: deep learning for time series forecasting. *19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020*, 765-768. https://doi.org/10.1109/ICMLA51294.2020.00125

Tripathi, B., & Sharma, R.K. (2022). Modeling Bitcoin Prices using Signal Processing Methods, Bayesian Optimization, and Deep Neural Networks. *Computational Economics*. https://doi.org/10.1007/s10614-022-10325

-8

Tzagkarakis, G., & Maurer, F. (2022). Horizon-Adaptive Extreme Risk Quantification for Cryptocurrency Assets. *Computational Economics*. https://doi.org/10.1007/s10614-022-10300-3

Uzun, B., Taiwo, M., Syidanova, A., Uzun Ozsahin, D. (2021). The Technique For Order of Preference by Similarity to Ideal Solution (TOPSIS). *Application of Multi-Criteria Decision Analysis in Environmental and Civil Engineering*, 25–30. Springer International Publishing. https://doi.org/10.1007/978-3-030-64765-0_4

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., ... Polosukhin, I. (2017). Attention is All You Need. *31st International Conference on Neural Information Processing Systems, NIPS 2017*, 6000–6010. https://arxiv.org/abs/1706.03762

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. (2020, October). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Wright, L. (2022). *Ranger - a synergistic optimizer*. 20.9.4. GitHub. Retrieved from https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer .

Zhang, M.R., Lucas, J., Ba, J., Hinton, G.E. (2019). Lookahead Optimizer: k steps forward, 1 step back. *Annual Conference on Neural Information Processing Systems, NIPS 2019*, 9593–9604. https://arxiv.org/abs/1907.08610

Table 5 Portfolio selection performance metrics for the different forecasting models.

Metric	N-B. Transf.	N-B. Mish	2-Layer GRU	N-B. Orig.	1-Layer LSTM	N-HiTS	1-Layer GRU	2-Layer LSTM	N-B. Perc.	N-B. 1Conv.	Crost.	Theta	FFT
Mean	0.02	-0.01	0.02	-0.02	0.01	**0.04**	0.02	0.01	0.01	0.01	-0.00	0.01	**0.04**
Standard deviation	0.07	0.09	0.09	0.09	0.13	0.11	0.18	0.08	**0.04**	0.09	0.21	0.10	0.10
Minimum	-0.16	-0.28	-0.29	-0.26	-0.29	-0.30	-0.21	-0.29	-0.09	-0.28	**-0.35**	**-0.35**	-0.22
25th percentile	-0.02	-0.04	-0.02	**-0.09**	-0.04	-0.01	-0.05	-0.03	-0.01	-0.04	-0.08	-0.04	-0.03
50th percentile	0.01	-0.00	0.02	-0.01	-0.00	**0.03**	-0.00	0.00	0.01	0.01	-0.04	0.01	**0.03**
75th percentile	0.05	0.04	0.06	0.05	0.05	0.08	0.05	0.05	0.03	0.05	0.02	**0.09**	**0.09**
Maximum	0.32	1.90	**2.92**	0.39	2.07	1.54	**2.92**	**2.92**	0.16	2.07	1.54	0.27	0.39
Downside deviation	0.03	0.06	0.05	0.06	0.04	0.04	0.04	0.05	**0.02**	0.05	0.04	0.07	0.05
Maximum drawdown	0.47	2.18	3.21	0.64	2.36	1.84	3.13	3.21	**0.26**	2.35	1.89	0.63	0.61
Sharpe ratio	0.33	-0.06	0.26	-0.18	0.06	0.38	0.09	0.12	0.26	0.16	-0.01	0.07	**0.43**
Sortino ratio	0.68	-0.09	0.46	-0.30	0.19	**1.11**	0.41	0.22	0.49	0.28	-0.04	0.11	0.78
Calmar ratio	0.05	-0.00	0.01	-0.03	0.00	0.02	0.01	0.00	0.04	0.01	-0.00	0.01	**0.07**
Ulcer index	0.05	0.09	0.07	0.10	0.06	0.06	0.07	0.07	**0.03**	0.07	0.08	0.11	0.09
Average loss	0.04	0.06	0.05	0.09	0.05	0.04	0.05	0.05	**0.02**	0.06	0.07	0.09	0.07
Conditional tail ratio	0.53	0.60	0.65	0.64	0.69	0.28	0.61	0.69	0.49	0.65	0.23	**0.87**	0.73
Negative outliers	**0**	2306	145	957	**0**	29	**0**	204	**0**	344	**0**	23	**0**
Value at Risk (95%)	0.07	0.17	0.10	0.15	0.10	0.05	0.10	0.11	**0.04**	0.11	0.12	0.18	0.13
Value at Risk (99%)	0.11	0.25	0.19	0.19	0.16	0.15	0.19	0.19	**0.09**	0.19	0.17	0.20	0.22
Cond. VaR (95%)	0.10	0.21	0.15	0.18	0.13	0.10	0.13	0.14	**0.07**	0.15	0.16	0.19	0.17
Cond. VaR (99%)	0.12	0.27	0.20	0.22	0.21	0.17	0.19	0.19	**0.09**	0.20	0.20	0.20	0.22
Expected Short. (95%)	0.09	0.20	0.14	0.17	0.13	0.09	0.12	0.13	**0.06**	0.14	0.15	0.18	0.16
Expected Short. (99%)	0.12	0.26	0.19	0.22	0.21	0.17	0.19	0.19	**0.09**	0.20	0.19	0.20	0.22
Coherent VaR (95%)	0.11	0.22	0.15	0.14	0.10	0.11	0.14	0.16	**0.07**	0.15	0.17	0.15	0.18

Figures

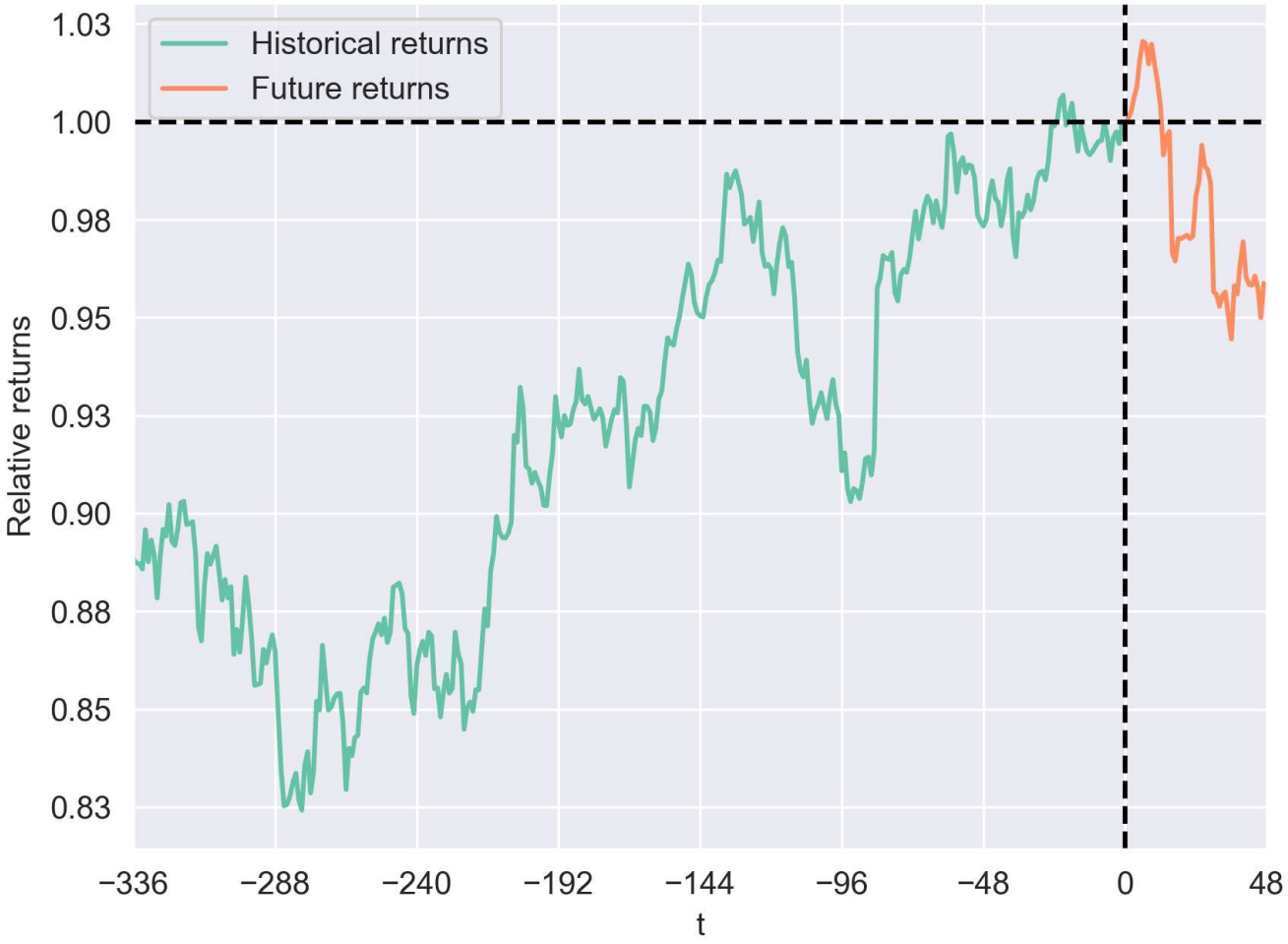

Figure 1

Sample time series for forecasting. This figure demonstrates a sample time series of hourly cryptocurrency portfolio returns from the test period that ends at the indexed value of 1. The task is to forecast future returns relative to this indexed value of 1 over the next H=48 hours, using a lookback window of 7H (equivalent to 336 hours). The horizontal and vertical dashed lines represent the point at which the value of the portfolio is indexed to one, which is the last observation.

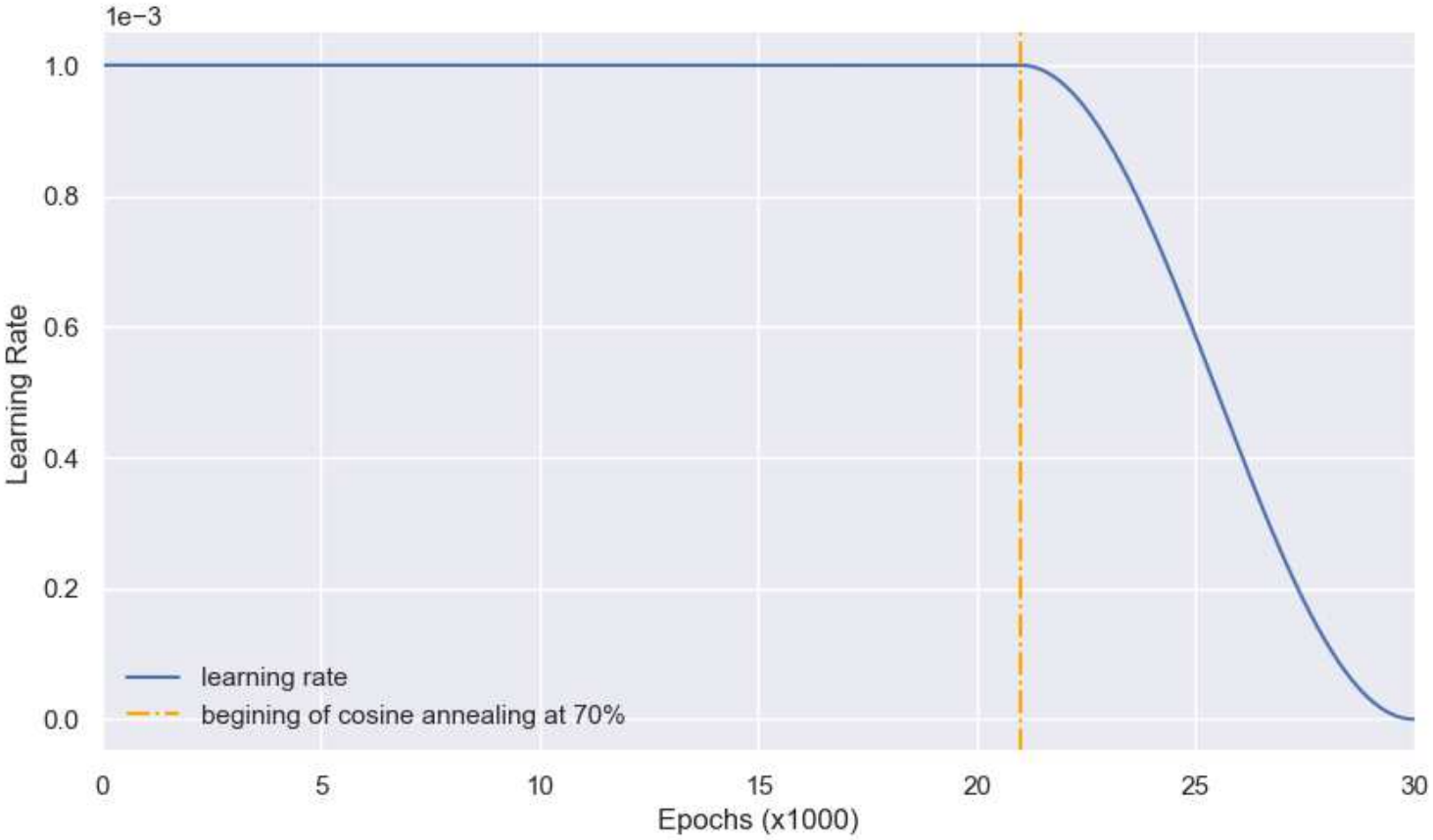

Figure 2

Flat plus cosine learning rate schedule. The optimizer begins with a learning rate of 0.001 and maintains this pace for 21,000 iterations, gradually decreasing the rate until it reaches zero for the remaining iterations.

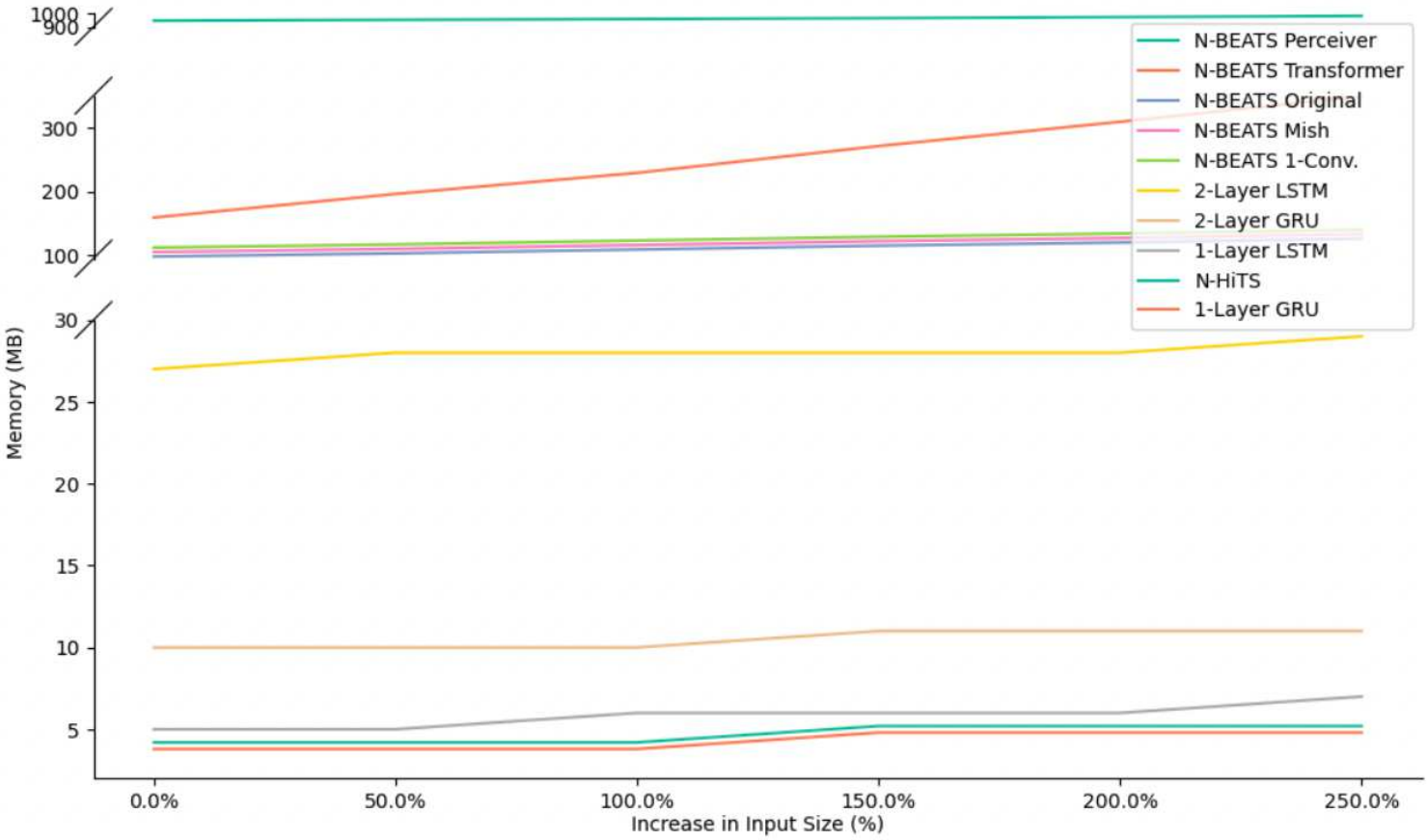

Figure 3

Scalability of Model Memory Utilization with Increasing Input Size. This graph demonstrates the relationship between memory utilization and input size for various ML models. The x-axis represents the increasing input size, as measured by percentage increases in the lookback parameter from 2H (0%) to 7H (250% increase in input size). The y-axis, displayed in logarithmic scale, illustrates the memory utilization of each model. The broken y-axis emphasizes the relative difference in memory utilization among the models.

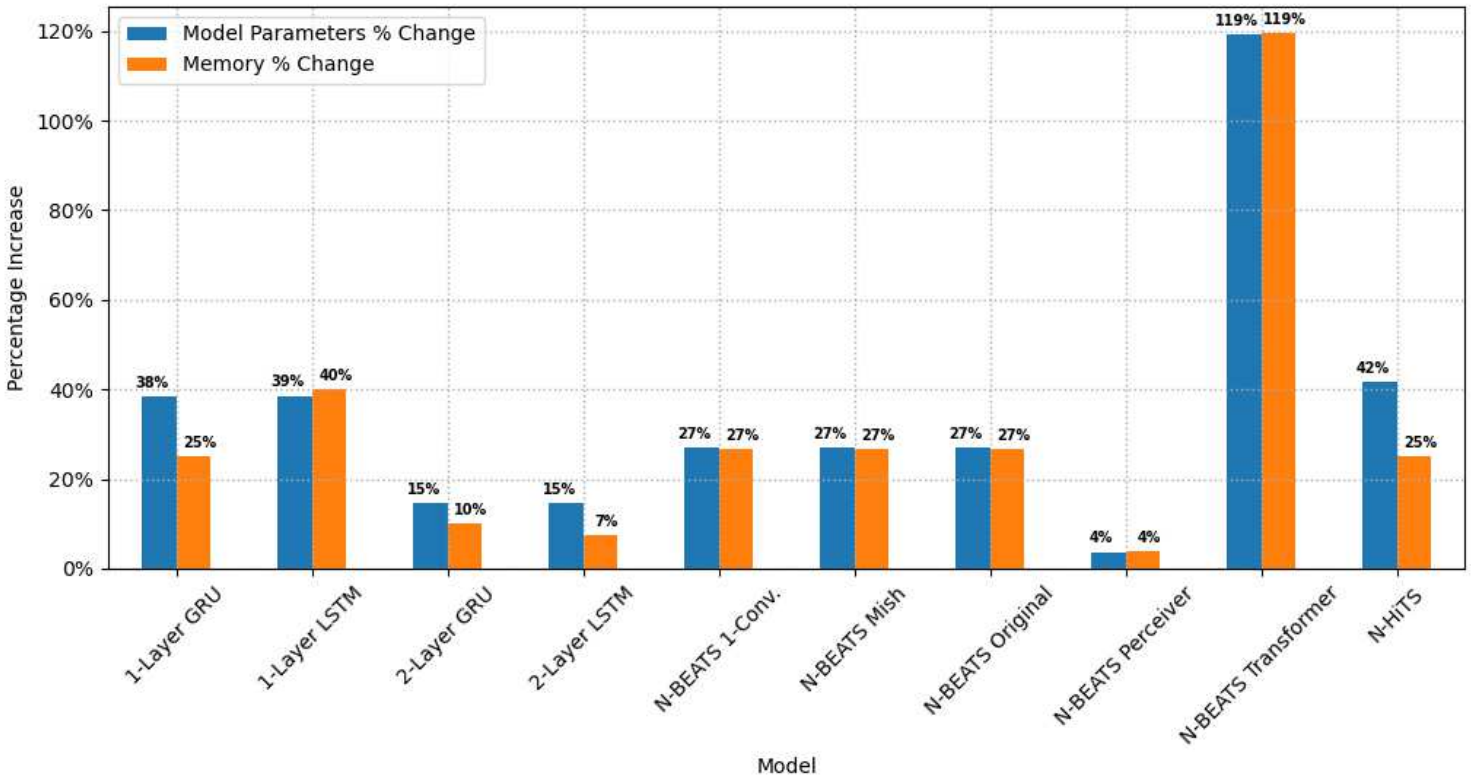

Figure 4

Scalability of Models in terms of Parameters and Memory. Comparison of the percentage increase in model parameters and memory for different models when the lookback period is increased from 2H to 7H, that is, when the input size increases by a percentage of 250%.

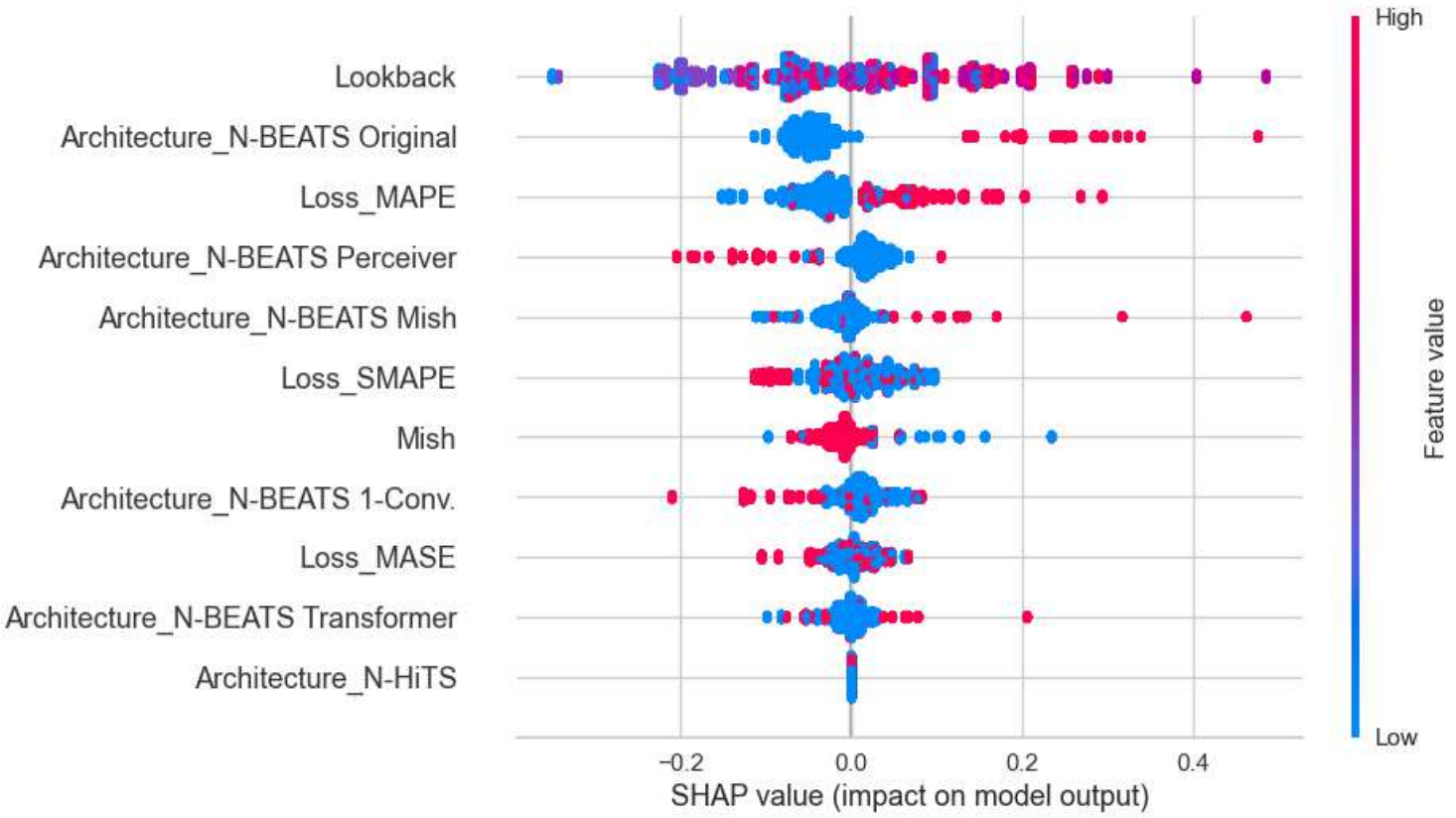

Figure 5

SHAP Analysis of LightGBM Model for sMAPE Error Estimation. The figure illustrates the distribution of SHAP values for the top 10 most important features, providing insight into the model's error estimation based on various characteristics such as architecture, lookback, loss function, and Mish activation. The red dots represent that the feature is present or it is high, and the blue dots represent that the feature is not present or it is low. The x-axis indicates the marginal impact of the feature samples on the LightGBM model's predictions. The dots located to the left of the zero impact line indicate that the feature improves the model's predictions by decreasing the predicted error.

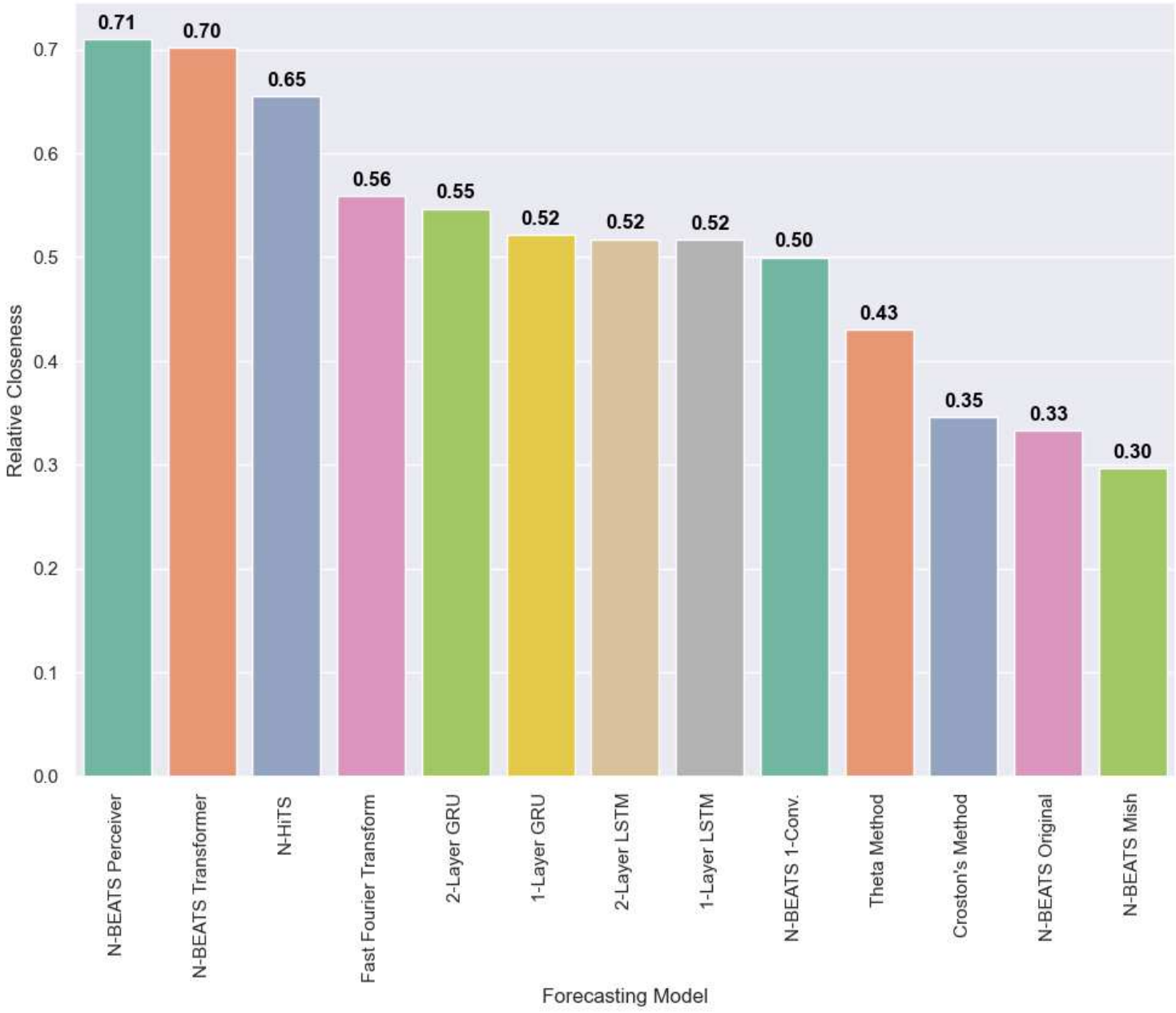

Figure 6

Comparison of Models using TOPSIS Method. The chart shows the relative closeness scores for each investment strategy, with the highest score indicating the top strategy according to the Multi-Criteria Decision Making (MCDM) method of TOPSIS. The criteria used for the analysis are those depicted in Table 5: Mean, Standard deviation, Minimum, 25th percentile, 50th percentile (Median), 75th percentile, Maximum, Downside deviation, Maximum drawdown, Sharpe ratio, Sortino ratio, Calmar ratio, Ulcer index, Negative outliers, Average loss, Conditional tail ratio, Value at Risk (95%), Value at Risk (99%), Cond. VaR (95%), Cond. VaR (99%), Expected Shortfall (95%), Expected Shortfall (99%), Coherent VaR (95%).

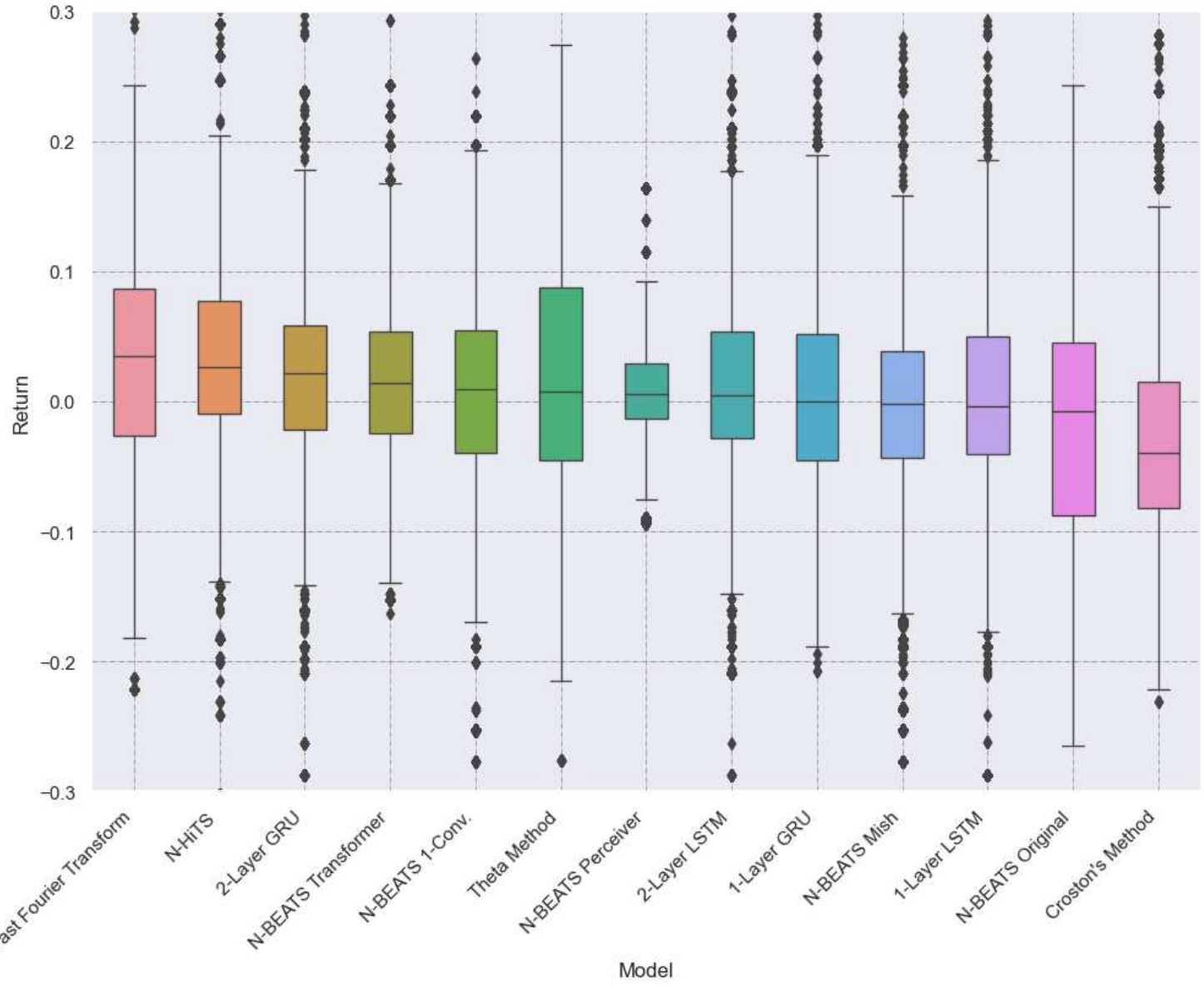

Figure 7

Comparison of Model Returns in a Portfolio Selection Task. This box plot data presents the returns of the portfolio selection task conducted over 102,635 rounds and is ordered by the median return, from the highest to the lowest.

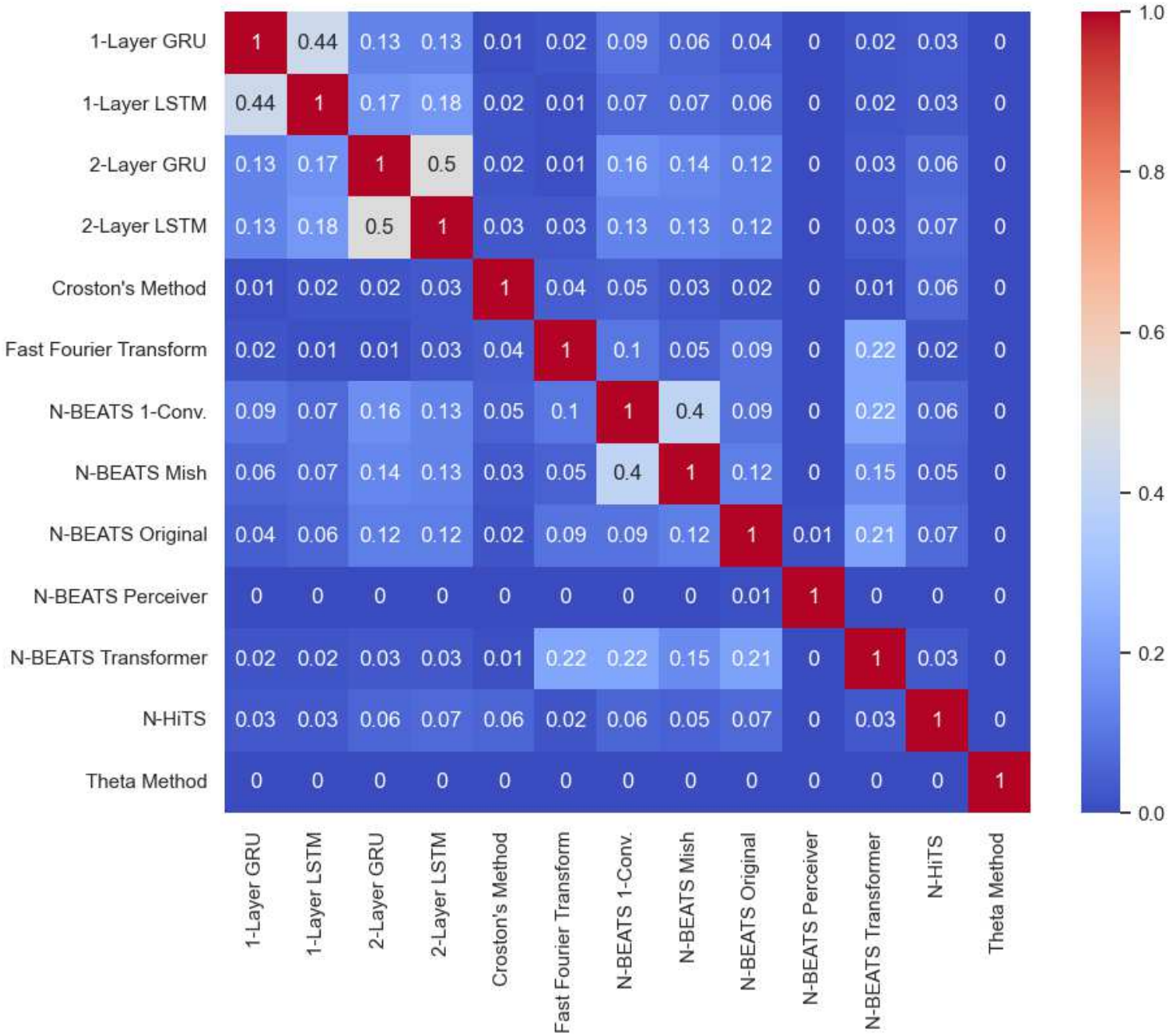

Figure 8

Comparative Analysis of Model Return Correlation. The chart shows the return correlation of different investment strategies, as determined by choosing a winning portfolio out of 100,000 options for 102,635 rounds. Models with high return correlation are likely to pick similar winning portfolios on each round of the experiment.

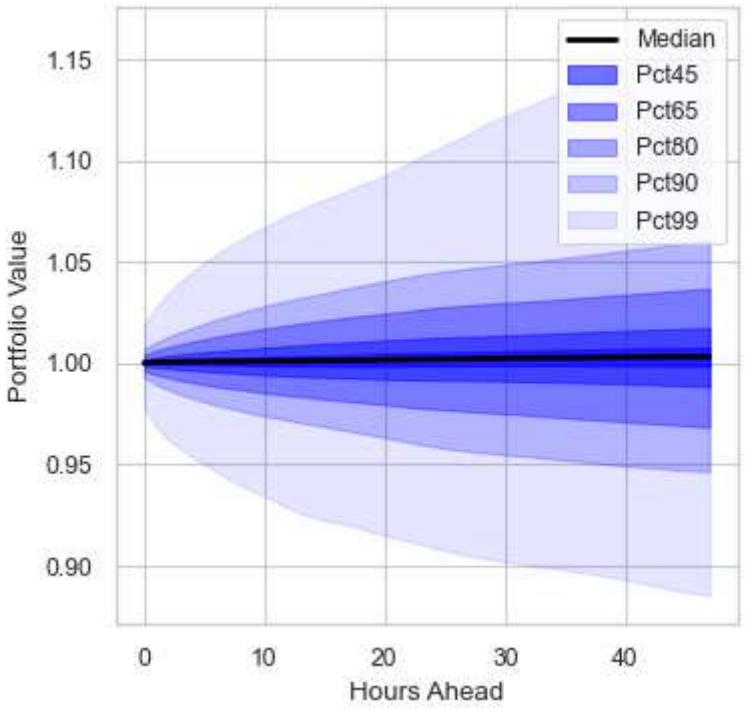

Figure 9

Analyzing the Performance of the Test Portfolios and their Returns Distribution. The left chart illustrates the distribution of portfolio values for over 4,160,000 sampled test portfolios, depicted by a fan chart with percentiles at 45, 65, 80, 90, and 99. These portfolios were utilized to assess the effectiveness of our models in forecasting the portfolio's returns over the following 48 hours, with the value at hour zero serving as a reference point indexed at 1 (or 100%). The models are expected to predict the fluctuation of the portfolio's value with respect to hour zero. The right chart displays the distribution of returns for the same test portfolios, with the mean return of 0.005 and median return of 0.003, represented by the red and green dotted lines.

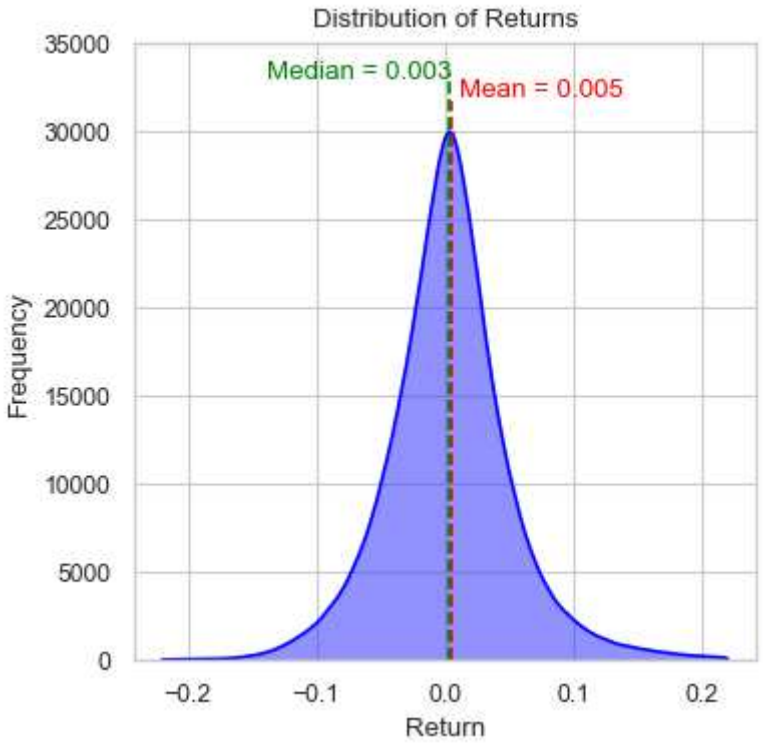

Figure 10

Analyzing the Performance of the Test Portfolios and their Returns Distribution. The left chart illustrates the distribution of portfolio values for over 4,160,000 sampled test portfolios, depicted by a fan chart with percentiles at 45, 65, 80, 90, and 99. These portfolios were utilized to assess the effectiveness of our models in forecasting the portfolio's returns over the following 48 hours, with the value at hour zero serving as a reference point indexed at 1 (or 100%). The models are expected to predict the fluctuation of the portfolio's value with respect to hour zero. The right chart displays the distribution of returns for the same test portfolios, with the mean return of 0.005 and median return of 0.003, represented by the red and green dotted lines.

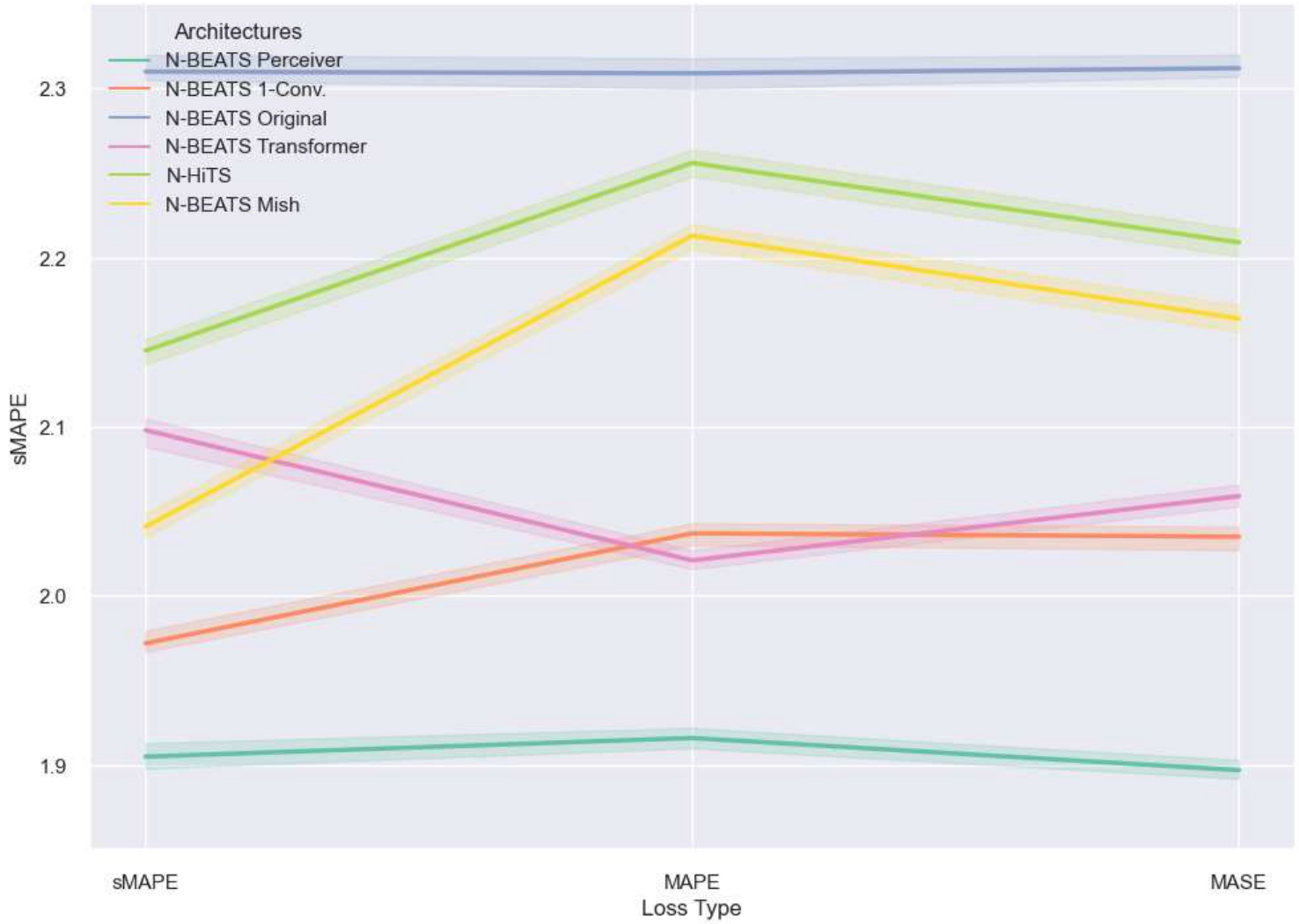

Figure 11

Comparison of sMAPE across N-BEATS Architectures and Loss Functions. The line chart displays the median sMAPE for N-BEATS models across architectures and loss functions. The x-axis shows the loss functions and the y-axis sMAPE values. Colored lines show each architecture's median sMAPE, with 99.99% confidence intervals, colored around the median.

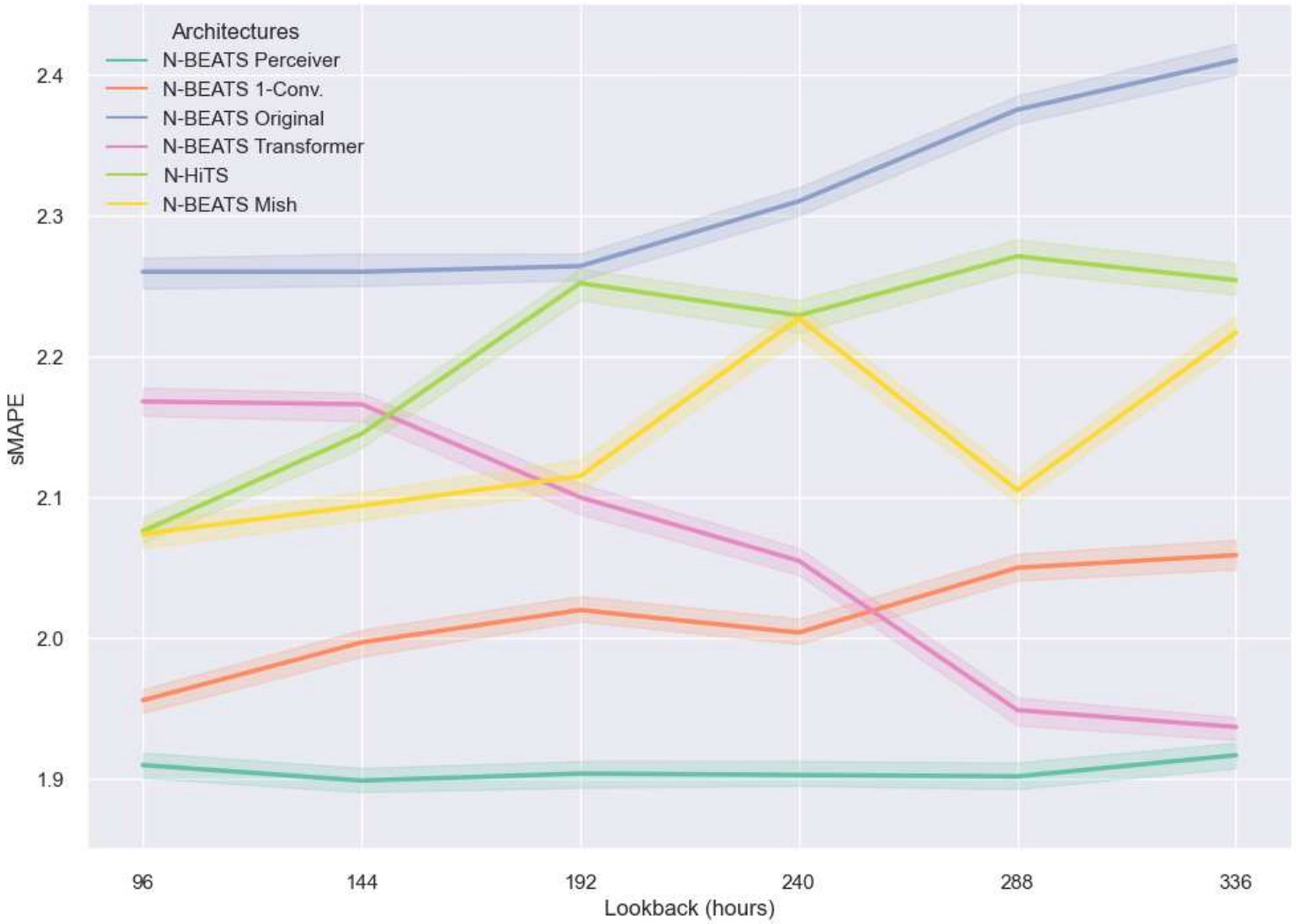

Figure 12

Comparison of sMAPE across N-BEATS Architectures and Lookback Windows. The line chart displays the median sMAPE for N-BEATS models across architectures and lookback windows. The x-axis shows lookback windows, in hours, and the y-axis sMAPE values. Colored lines show each architecture's median sMAPE, with 99.99% confidence intervals, colored around the median. The N-BEATS Transformer's longer lookback windows are shown to outperform its shorter ones, while the opposite is true for all remaining architectures.

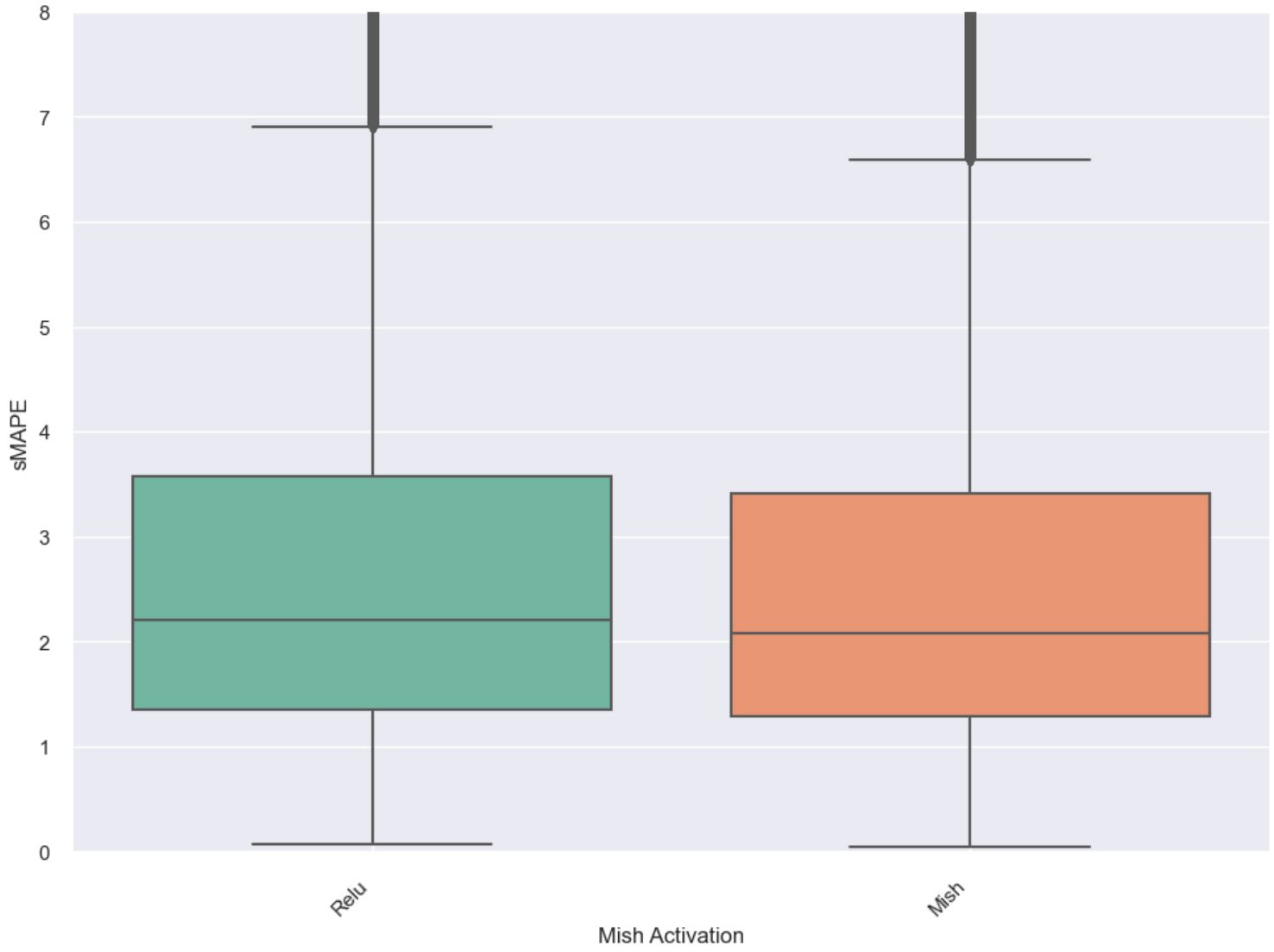

Figure 13

Comparison of the original N-BEATS models using ReLU and Mish Activation. The box plot of the variation in sMAPE for N-BEATS using ReLU and Mish activation are presented. The x-axis depicts the activation functions and the y-axis shows the sMAPE values. The box plots depict the median, quartiles, and outliers of each activation function's sMAPE. The results confirm that the use of Mish activation leads to more accurate predictions and more consistent performance across different scenarios.

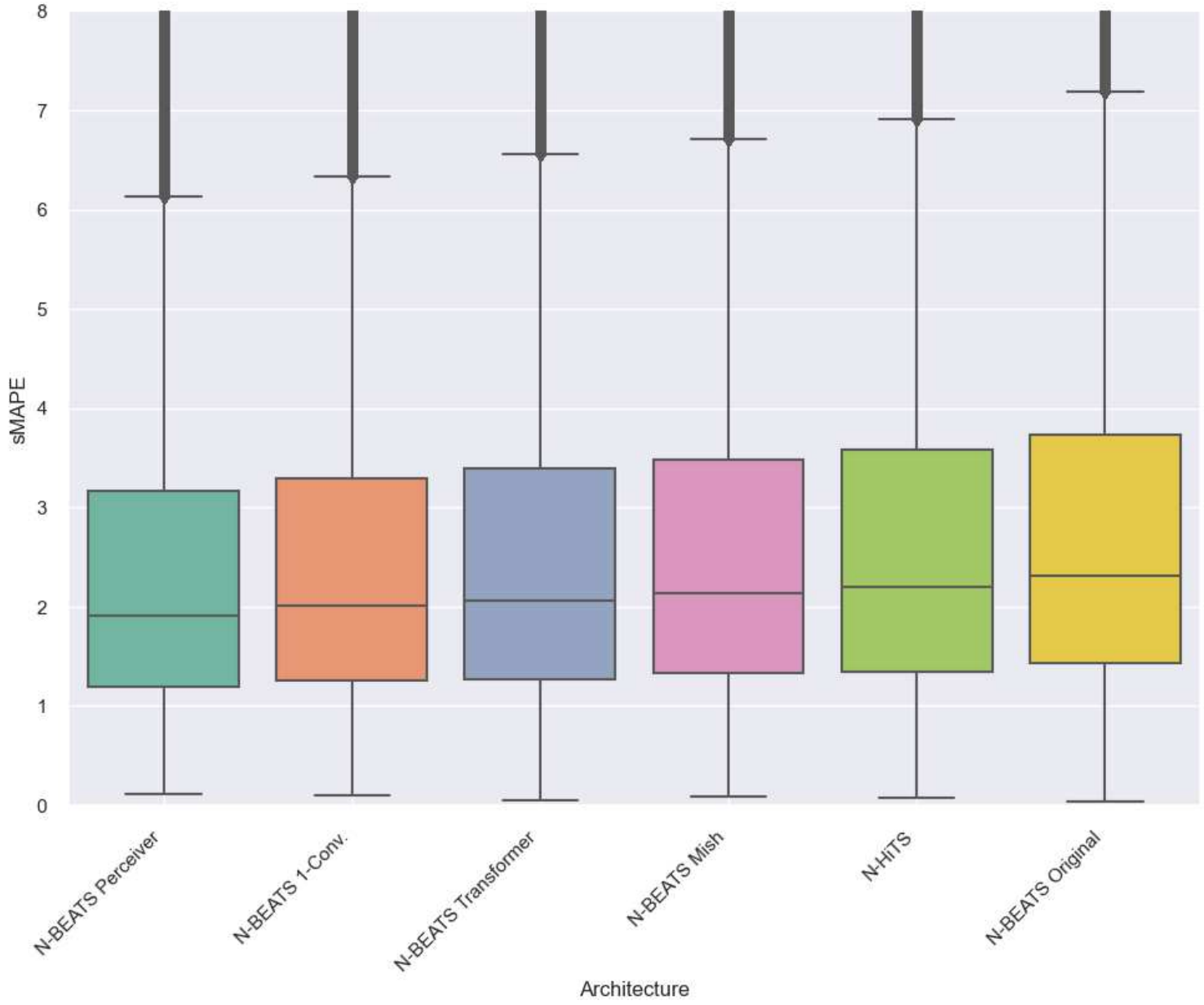

Figure 14

Box plot Representation of sMAPE Across N-BEATS Architectures. Each box plot summarizes the distribution of sMAPE values for a single architecture, showing the median, quartiles, and outliers. The architectures are sorted by median sMAPE, lowest first. The x-axis indicates architectures and the y-axis sMAPE values.

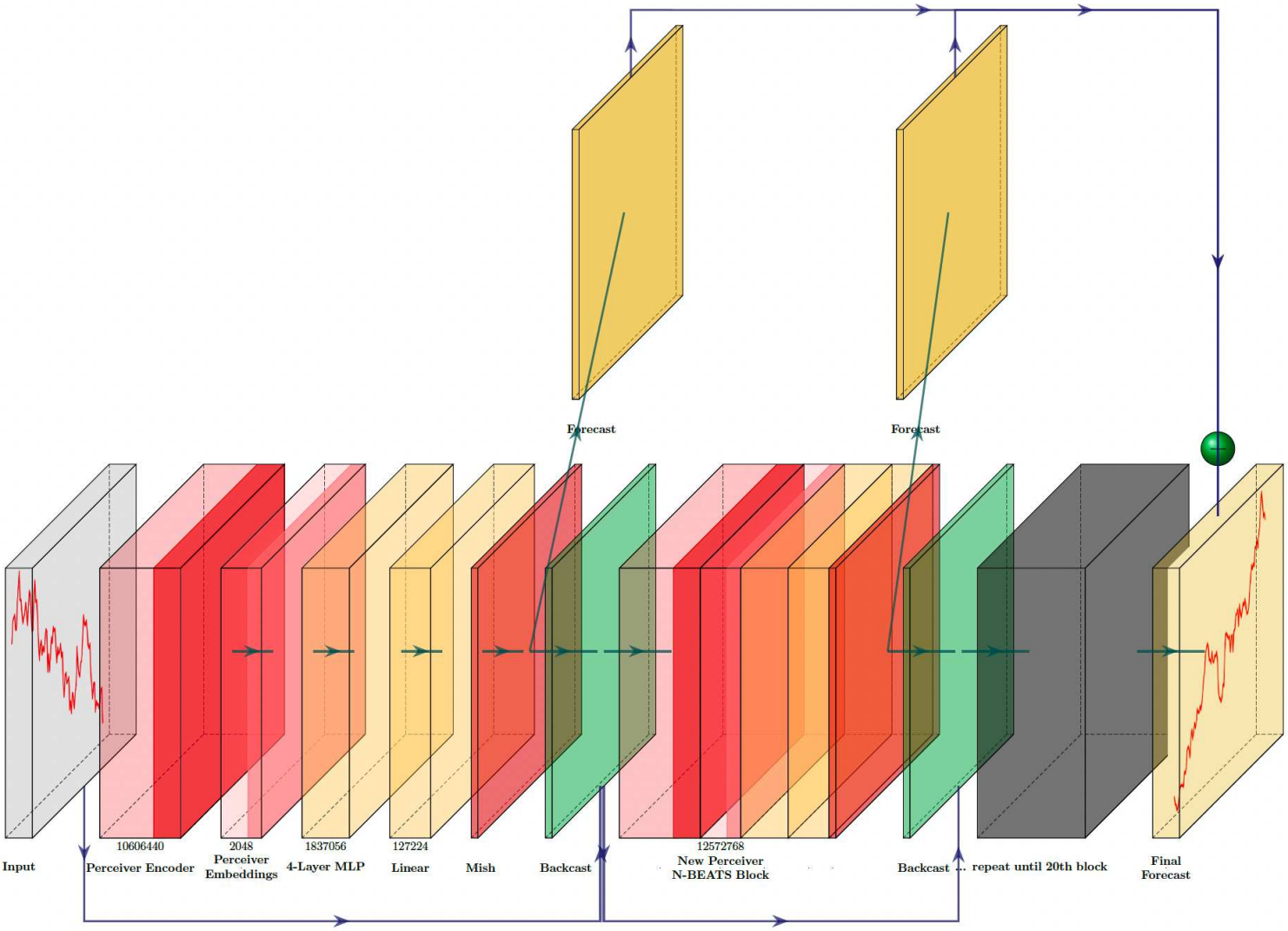

Figure 15

Proposed Architecture for N-BEATS Perceiver Model. The figure depicts our proposed architecture for the N-BEATS Perceiver model, a deep neural network designed for time series forecasting. The network is composed of 20 N-BEATS blocks, each containing a Perceiver Encoder, Perceiver Embeddings, a 4-Layer MLP, a Linear layer, and a Mish activation function. The Perceiver Encoder layer is responsible for extracting features from the input time series, while the Perceiver Embeddings layer learns embeddings for the extracted features. The 4-Layer MLP acts as a decoder layer responsible for generating the Backcast and the Forecast, and the Linear layer applies a linear transformation to the input before the Mish activation function's final transformation of the outputs. The model contains 251,455,360 trainable parameters in total. All backcasts are transferred to the subsequent block, and all forecasts are added to produce the model's final forecast.