MSc Artificial Intelligence
Master Thesis

Rethinking Models and Evaluations for Financial Time Series Forecasting

by
Didier Merk
11037172

December 17, 2024

48 ECTS
January - October, 2024

Supervisor:
Yongtuo Liu

Examiner:
Prof. Dr. Efstratios Gavves

Second reader:
Yongtuo Liu

Universiteit van Amsterdam

Abstract

Foundation Models, and particularly Large Language Models, have recently demonstrated remarkable success across a wide range of applications, from natural language processing to computer vision. This thesis explores the application of such models on financial time series forecasting, focusing on the Chronos foundation model; a time-series specific foundation model trained using the T5 large language model architecture. Financial time series data is influenced by various unpredictable factors, making forecasting difficult. While classic statistical models and, more recently, machine learning and deep learning models have been the primary tools for forecasting, recent research shows promising results for foundation models dedicated to forecasting. In this thesis, Chronos is compared against multiple state-of-the-art deep learning models and statistical baselines to evaluate its potential for financial forecasting. The accuracy of each model is evaluated across different forecasting horizons and the influence of specific characteristics, such as seasonality or entropy, on model performance is analyzed. Additionally, the reliability of Chronos' probabilistic output is compared to those of the baseline models. Chronos demonstrates strong forecasting abilities, achieving zero-shot results on par with deep learning models trained specifically on the data; however its probabilistic outputs show high unreliability. Further results provide insight into the potential and limitations of using large language model architectures for financial time series forecasting, and the landscape of forecasting as a whole. All code of this thesis project can be found on `https://github.com/didiermerk/forecasting`.

Acknowledgements

First and foremost, I would like to thank my supervisor at the ING WBAA department, Fabian Jansen. Your enthusiasm and guidance throughout this project have been one of the main reasons working at ING has been such a pleasure. The discussions we had and your encouragement to critically evaluate every step were invaluable to this research.

I would also like to extend my thanks to Yongtuo Liu. During our progress meetings you always seemed to know exactly what needed to be done next, and your guidance helped keep the project on the right track every time. The tips you gave me, also during the time we worked together before this thesis, have been super helpful and made a real difference.

In addition, I would like to thank the full Data Science Chapter at ING. My fellow interns, Riccardo, Filippos, Svara and Chenhao, thank you for the great times both in and out of the office. Max, Hazal, Nilay, Ilan, Morris, Ralph, Simon and Tomasz, I have appreciated all your help tremendously. It has been an absolute pleasure to work with all of you over the past few months. I also promise that I will forever classify vanilla ice cream as flavorful.

Lastly, I would like to thank my family, Marcel, Esther and Hélène, my girlfriend, Daantje, and my friends, Steinar, Ben and Timo, for their complete support throughout this project, reading through the thesis and listening to all of my ideas.

Contents

1 Introduction

Time series analysis plays an essential role in understanding and predicting dynamic systems across various domains. From stock market prices, to blood sugar levels and audio samples, real world data is often recorded with a notion of time attached to it. When such data is collected, it forms a sequence of data points that is known as a *time series*. These sequences, when analyzed, can reveal patterns or trends that allow the detection of anomalies, classification of data, or the prediction of future values; a principle known as *forecasting*.

In 2024, millions of people living in Florida, United States, evacuated their homes in anticipation of a forecasted hurricane. The hurricane, later called Milton, eventually became the fifth-most intense Atlantic hurricane on record. In an interview with TIME magazine, meteorologist Matt Lanza explained that novel Artificial Intelligence (AI) forecasting models such as Google's GraphCast [1] correctly predicted where Milton would make landfall about 12 to 18 hours earlier than classic physics-based models, significantly reducing the amount of casualties and damages [2].

In the medical field, where time series data is abundant, deep learning models are increasingly being employed as anomaly detectors [3, 4]. These developments have allowed wearables such as the Apple Watch to analyze a person's electrocardiogram (ECG) and detect conditions like atrial fibrillation, a type of irregular heartbeat. By continuously monitoring time series data, these models can provide early warnings, allowing users to seek medical attention when necessary.

Even though time series analysis is used in a broad spectrum of fields, it is perhaps most prominently applied in the financial sector. Financial time series, which include data such as interest rates, GDP growth or transactions, are central to decision-making processes in investment management, risk assessment and stock trading.

The importance of time series analysis in finance was further underlined in 2003 when Robert Engle and Clive Granger were awarded the Nobel Prize in Economic Sciences. Engle's work on market volatility, particularly through his development of the ARCH model, allowed for more accurate predictions of how financial markets fluctuate over time [5].

Granger's research on cointegration highlighted how long-term relationships between different economic variables could be crucial in understanding their shared trends [6]. Their contributions have since become fundamental in financial modeling, influencing risk management, investment strategies and macroeconomic analysis.

Despite its widespread applications, forecasting financial time series remains challenging. More than in other domains, financial data is influenced by unpredictable factors such as market sentiment, geopolitical events and economic policies, making it inherently noisy and prone to sudden unpredictable changes. Classic statistical forecasting models such as ARIMA [7] and Exponential Smoothing [8], however, assume that patterns from the past, will continue in a stable and predictable manner. This often causes these models to struggle to capture unpredictable fluctuations and fail to accurately forecast financial time series.

In the past two decades, the development of machine learning models has improved the quality of forecasting significantly. These models offer the ability to learn more complex and non-linear patterns in the data. In 2020, a so-called Gradient Boosting Decision Tree by Microsoft, LightGBM [9], won the biggest benchmarking competition in the field of time series forecasting [10].

A major breakthrough in the field of Artificial Intelligence came in 2006 when Geoffrey Hinton introduced Deep Belief Networks [11]. These networks have now evolved into what are commonly known as *neural networks*, architectures designed to mimic the way the human brain learns by recognizing patterns in data. Unlike earlier models, these neural networks could handle far more complex and high-dimensional data, causing breakthrough in tasks like image recognition and computer vision [12].

The advancement in time series forecasting accelerated through the development of a particular type of neural network architecture, called Recurrent Neural Networks (RNNs), and more specifically Long Short-Term Memory (LSTM) networks [13]. These architectures are designed to handle sequential data using *recurrent connections*, which can be best described as loops in the network. This allows LSTMs to remember information over long sequences and filter out irrelevant data, and compete with state-of-the-art non-deep learning models, such as LightGBM.

While LSTMs marked a significant advancement in handling sequential data, they faced limitations when it came to processing very long sequences. As sequences grow in length, LSTMs tend to "forget" important information that occurred earlier in the sequence, leading to a deteriorating performance. In financial time series, this could mean that an LSTM may gradually "forget" significant trends that occurred several months ago. Additionally, the sequential nature of LSTMs makes them computationally slow, as they process information one step at a time.

In 2017, the introduction of the Transformer architecture by Vaswani et al. revolutionized sequence modeling by addressing these limitations [14]. Unlike LSTMs transformers do not rely on sequential processing. Instead, a mechanism called self-attention allows the model to process all data points in a sequence simultaneously. This transformer architecture lays at the heart of the successes in many Natural Language Processing (NLP) tasks such as machine translation and generating text based on next-token-prediction.

The introduction of Large Language Models (LLMs), built on the transformer architecture, has further revolutionized the field of AI. These models, trained on large amounts of text data -hence the term *Large Language Models*-, demonstrate remarkable performance in a wide range of tasks, including text generation, summarization and translation. Within just two years of its release, OpenAI's ChatGPT [15] attracts over 200 million weekly users worldwide. The success of these models lies in their ability to learn complex patterns and relationships from the massive datasets, enabling them to predict and construct coherent sentences.

Recent research has begun exploring the potential of applying pre-trained language models to time series forecasting [16, 17, 18]. Early studies suggest that LLM architectures can be leveraged to forecast time series data at a level comparable to state-of-the-art statistical and deep learning models [19, 20, 21, 22], *without needing to be specifically trained on the given data.*

However, there remains a gap in understanding how well LLMs perform in financial domains, especially when dealing with data such as transaction records. This data is task-specific and noisy and often highly protected, making it difficult to determine how well pre-trained models can adapt to it, given that this kind of data is typically absent from

the models' initial training. Indeed, the eight most widely used benchmarking datasets for time series forecasting consist of weather, traffic, electricity and medical data, but exclude financial datasets [23]. As a result, the application of LLMs in financial time series forecasting remains largely unexplored.

This thesis aims to bridge this gap by investigating whether the principles that make large language models successful in sequencing tasks like text generation and next-token prediction, can also be applied to the forecasting of transaction data. The research question this thesis seeks to answer is defined as follows:

"To what extent can large language model architectures be applied to financial time series forecasting, in comparison to traditional statistical and deep learning models?"

To answer this, four sub-questions are formulated, which together will provide an answer to the above question.

1. How well does a large language model architecture, pre-trained on time series data, forecast *unseen transaction data*, compared to state-of-the-art statistical and deep learning models?

2. How well does a large language model architecture forecast transaction data, when it is pre-trained on time series data *and fine-tuned on transaction data*?

3. How do *intrinsic characteristics* of financial time series affect the performance of a large language model architecture and other forecasting models?

4. How accurate is the *probabilistic output* produced by a large language model, and how do they compare to those generated by traditional models?

The primary contribution of this thesis is to provide valuable insights into the forecasting capabilities of a pre-trained large language model architecture within the context of financial time series. This research compares its performance to traditional statistical and deep learning models, evaluates its adaptability through fine-tuning, and investigates the accuracy of its probabilistic outputs. These findings offer a deeper understanding of the potential and limitations of applying LLMs to financial forecasting, contributing to ongoing efforts in adapting language models for time series analysis.

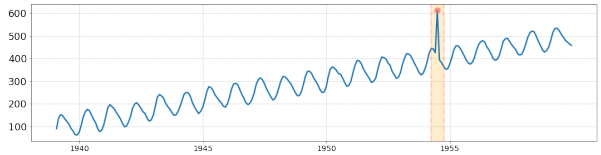
2 Theoretical Background

2.1 Time Series

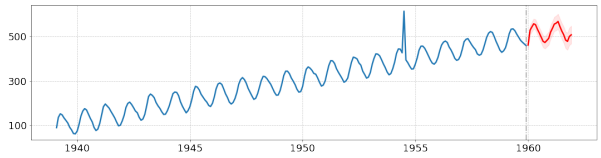
Time series data is everywhere. It is the kind of data that is acquired when something is recorded over time; whether it is the daily temperature, stock prices or blood sugar levels. What makes time series data unique is that the order of data points matters. Each point is connected to the one before and the one after, creating a sequence that captures the way something changes over time.

2.1.1 Time Series Analysis

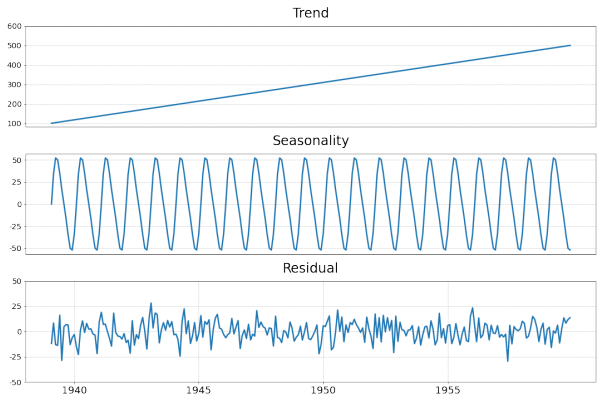
Time series analysis is the process of studying and understanding these sequences of data. The goal is to recognize and model underlying patterns that reveal themselves within the sequence. These patterns, often hidden within the data, help explain how the system under study evolves. By analyzing series, it is possible to detect trends, seasonal changes or irregular events that may not be immediately visible but are important to understanding the behavior of the data.



(a) Anomaly Detection



(b) Forecasting



(c) Decomposition

Figure 1: An illustration demonstrating the different types of time series analyses. The data shows the amount of airline passengers in the United States during the middle of the 20th century [7].

An important application of time series analysis is anomaly detection (see Figure 1a), where the aim is to identify data points that deviate from the expected pattern. These anomalies can provide significant insights, such as spotting fraudulent transactions in finance or

detecting unusual health readings in medical monitoring [24]. Anomalies are often outliers that break the regular trends or cycles, and identifying them early can prevent larger problems down the line.

Another key technique in time series analysis is decomposition (Figure 1c), which involves breaking the data down into different components: a trend, which shows the overall direction of change over time; seasonality, which captures recurring patterns at regular intervals; and residuals, which represent the random noise or anomalies in the data. By separating these components, the underlying structure of the data becomes clearer, making it easier to interpret past behavior and plan for future trends.

All of these techniques serve a common goal: to understand the past and make better predictions about the future. This leads naturally to the concept of *forecasting* (Figure 1b) where the patterns learned from historical data are used to make informed predictions about what's coming next.

2.1.2 Time Series Forecasting

Time series forecasting is the process of using patterns in past data to predict future values. More formally, given a time series of observed data points $\{x_1, x_2, ..., x_T\}$, the goal of forecasting is to estimate future values $\{x_{T+1}, x_{T+2}, ..., x_{T+h}\}$ based on the past values up to time T. Here h indicates the forecasting *horizon*, or the amount of time steps we want to forecast ahead.

In 1927, British statistician Sir Udny Yule introduced one of the earliest forecasting models: the autoregressive (AR) model [25]. The AR model works by assuming that the current value of a sequence can be explained by a combination of its previous values, also called *lag terms*. In the simplest version of the model $AR(1)$, a value x_{T+1} only depends on the value immediately preceding it, x_T.

The AR model assigns a set of weights, or *coefficients* (ϕ), to determine how much influence each past value has on the current prediction. It also contains a randomness term to account for elements that are not predictable. This model formed the foundation of the more complex family of statistical forecasting models known as Auto Regressive Moving Average (ARMA) models.

This family of statistical approaches towards forecasting, was popularized by George Box and Gwilym Jenkins in their 1970 book *"Time Series Analysis: Forecasting and Control"* [7]. The ARMA models combine autoregressive approaches with *moving averages*, which help to smooth out fluctuations by averaging a fixed number of past observations. This smoothing process helps reveal underlying trends and patterns that the AR model alone may miss.

Around the same time, another significant contribution to statistical forecasting came from Charles Holt and Peter Winters with their development of Exponential Smoothing (ETS) models [8, 26]. Unlike moving averages, ETS models apply a smoothing factor that exponentially decreases for older data points, giving more weight to recent observations. This allows the models to quickly respond to changes in trends, while still considering the broader trend of the sequence.

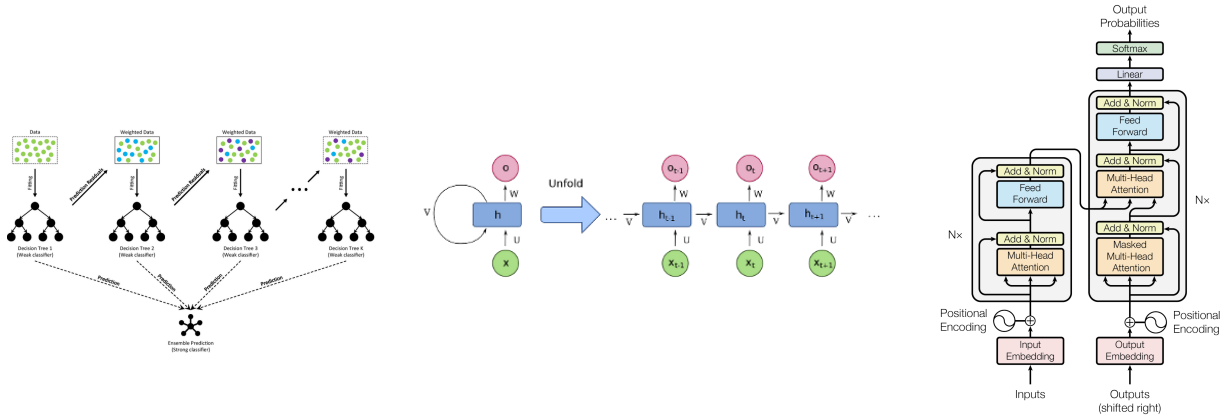
What makes ETS models stand out is their ability to decompose a time series into three key components: ***E***rror, the randomness of a sequence; ***T***rend, the long-term direction of a sequence; and ***S***easonality, the repeating patterns that occur in the sequence (Figure 1c). By smoothing each component separately, ETS models are able to handle time series with complex patterns.

Both ETS and ARMA models have motivated some of the most widely used forecasting methods today [27]. In this thesis research two variations, AutoETS and AutoARIMA, are employed to set a statistical baseline for financial time series forecasting. Further theoretical background of these models and implementation details can be found in Section 2.2.1 and 3.3.1 respectively.

Throughout the next decades statistical models remained dominant in forecasting. Their performance was repeatedly evaluated in the M-competitions [28, 29, 30], a series of forecasting challenges designed to compare the performance of a large number of major time series methods. In the first three competitions, held between 1982 and 2000, the results consistently showed that simpler statistical models were often the most accurate. Spyros Makridakis, the organizer of the challenges, wrote in his conclusion of the second M-competition that *"The best methods were found to be the simplest"*.

It is only in the past decade that models with more complex architectures started to

outperform classic statistical methods [10]. Machine learning techniques such as Gradient Boosting Decision Trees (GBDTs) [31] offer more flexibility when modeling time series data. Unlike statistical models they do not rely on strict assumptions about the data; instead, they can adapt to the inherent complexity of specific sequences. This makes them particularly useful in scenarios where the data exhibits non-linear relationships or irregular patterns.



(a) Gradient Boosting Decision Tree (b) Recurrent Neural Network (c) Transformer

Figure 2: Examples of the main forecasting architectures that have outperformed statistical methods over the past two decades: (a) A Gradient Boosting Decision Tree, a traditional machine learning method based on an ensemble of sequential decision trees [32] (b) A Recurrent Neural Network, a deep learning method with the ability to retain information over time [33] (c) The Transformer architecture, a deep learning architecture efficiently models long-term relationships in data using self-attention mechanisms [14].

Gradient Boosting Decision Trees combine several decision trees, where each tree contributes to improving the overall prediction (Figure 2a). A decision tree works by splitting data based on certain conditions to make predictions. However, a single decision tree is often considered a "weak learner" that struggles to capture complex patterns. GBDTs address this by building one tree at a time, where each new tree focuses on correcting the errors made by the previous trees [31, 9]. This process of sequential improvement, known as *boosting*, allows a model to learn from its mistakes.

In the context of forecasting, this process allows individual trees to adjust for trends or seasonal effects in a time series, that earlier decision trees might have missed. By progressively refining the model, GBDTs can capture complex, non-linear relationships in sequences, leading to more accurate predictions. This was particularly proven when Light-

GBM [9], a GBDT designed by Microsoft, outperformed all other statistical and deep learning models during the fifth edition of the M-competition in 2020 [10].

The work of British computer scientist Geoffrey Hinton on *"Deep Belief Nets"* [11], introduced a new class of machine learning models, contributing to the development of what we now call *deep learning* models. These models are based on *neural networks*, architectures inspired by the structure of the human brain, where layers of artificial neurons process information step by step. Different deep learning architectures have led to significant advancements in fields such as computer vision [12, 34, 35], natural language processing [36, 14] and, in 2024, Hinton's work was recognized with the Nobel Prize for physics [37].

Recurrent Neural Networks (RNNs, Figure 2b) [38] introduced an innovative way to handle sequential data, by incorporating loops in their architecture. Unlike traditional "feedforward" networks, where the information flows only in one direction, RNNs allow connections to cycle back on themselves, giving the network a form of memory. In other words, past information can influence current predictions, making RNNs particularly useful for time series tasks.

The main limitation with RNNs is called *the vanishing gradient problem*; the influence of earlier time steps in the sequence diminishes as information is passed through the network, causing the network to "forget" information that happened earlier in the sequence [39].

The Long Short-Term Memory (LSTM) architecture [13] is a specific RNN architecture designed to mitigate this issue, and capture the relationships within data over longer periods of time. LSTMs work with *memory cells*, that can store information over long periods of time. The flow of information through these memory cells is controlled by three types of gates:

1. The *input gate*, which decides what new information should be stored in the memory.
2. The *forget gate*, which determines what part of the memory needs to be discarded.
3. The *output gate*, which controls how much information stored in the cell is used for the current prediction and passed on to the next step in the sequence.

By carefully controlling the information that passes through the network, LSTMs can memorize information over longer sequences, and capture long-term patterns in the data.

LSTMs often outperform statistical models in fields as financial time series forecasting [40, 41, 42], but can still struggle with very long sequences [43]. Additionally, due to their nature of processing information one step at a time, LSTMs are computationally slow and expensive.

In 2017, researchers at Google Brain created the Transformer architecture [14], shown in Figure 2c. Transformers use a mechanism introduced as *self-attention*, which combines the ability to capture long-term dependencies and process data in a parallel manner, rather than sequentially like RNNs.

The self-attention mechanism works by assigning different levels of importance, or *attention*, to each data point in a sequence. For example, in time series forecasting, a Transformer can weigh certain past data points more heavily when predicting future values, even if these points occurred long ago. Being able to efficiently model both short-term fluctuations and long-term dependencies makes Transformers based models competitive forecasters [44, 45, 46, 47].

However, despite achieving unparalleled performances in Natural Language Processing [15, 36] and Computer Vision [35] tasks, the application of Transformers to time series forecasting has not been as dominant. In their 2022 paper *"Are Transformers Effective for Time Series Forecasting?"*, Zeng et al. claim that due to the permutation-invariant nature of the self-attention mechanism, temporal information will inevitably get lost in the Transformer architecture [48]. The authors further demonstrate this, by introducing a set of "embarrasingly" simple one-layer models that outperform existing sophisticated Transformer models.

Ultimately, at least currently, there does not exist a one-size-fits-all model for time series forecasting. Different architectures have their own strengths and limitations, and each works better or worse depending on the characteristics of the data and the forecasting task at hand.

This thesis focuses on the performance of six forecasting models, each based on a different architecture, in predicting transaction data. These models serve as a benchmark to compare the performance of a new class of models, known as *Large Language Models*. In Section 2.2 a deeper theoretical background of the benchmark models is given; consequently, in

Section 2.3, the concept of the novel large language models is explained.

2.2 Forecasting Models

There is no single dominant model in the field of time series forecasting, resulting in a wide range of architectures that remain competitive. In this thesis, the focus shifts to nine forecasting models, each built on a different underlying architecture, summarized in table 1.

The collection of models includes statistical baselines, the AutoARIMA and AutoETS model; and more advanced, state-of-the-art, deep learning models: PatchTST, NHITS, DeepAR and TimesNet. In this section, each model's theoretical background is discussed in more detail, explaining how they are designed and how they approach time series forecasting.

Class	**Model**	**Architecture**	**Parameters**
Statistical	`AutoARIMA` [7]	ARMA	variable
	`AutoETS` [8, 26]	ES	variable
Deep Learning	`PatchTST` [47]	Transformer	604,000
	`NHITS` [49]	MLP	3,900,000
	`DeepAR` [50]	RNN	199,258
	`TimesNet` [51]	CNN	4,900,000
Foundation	`Chronos-small` [19]	LLM	46,000,000
	`Chronos-large` [19]	LLM	710,000,000
	`Chronos-FT` [19]	LLM	46,000,000

Table 1: Overview of the models used in this thesis to create both a statistical baseline, and offer insights into the performance of more advanced deep learning based forecasters.

2.2.1 Statistical Models

Using classic statistical models for time series forecasting offers the advantage that these models do not require to be trained on large datasets. Instead, they base predictions solely on historical patterns within each individual time series, providing robust baseline forecasts without needing complex model training.

AutoARIMA

The AutoARIMA model [52] is an extension of the traditional ARIMA model [7], designed to automatically select the optimal parameters for forecasting given a time series. To understand AutoARIMA, it is useful to first understand the ARIMA model.

The ARIMA model combines three main components: autoregressive (AR), integration (I) and moving averages (MA). These components are defined by three parameters p, d and q, representing the order of each component:

- **Autoregressive (AR)**: The AR component, indicated by the parameter p, determines how many past values, or *lag terms*, the model will use to predict a future value. For example, if $p = 1$, the model only considers a direct predecessor when forecasting.

  For example, an AR(1) model with $p = 1$ would look like:

  $$y_t = \mu + \phi_1 y_{t-1} + \epsilon_t$$

  where y_t is the current value, μ is the level or intercept of the series, ϕ_1 is the coefficient for the last value y_{t-1}, and ϵ_t is the error term.

- **Integration (I)**: The differencing component, indicated by d, is applied to make a sequence stationary by subtracting the previous value for each element. This results in a sequence with statistical properties, such as mean and variance, that remain constant over time.

  To handle non-stationary time series, the ARIMA model applies differencing d times to the series, making it stationary. For instance, if $d = 1$, the differenced series y'_t is:

  $$y'_t = y_t - y_{t-1}$$

- **Moving Average (MA)**: The MA part, controlled by q, allows the model to incorporate past errors it made, into its forecasts. This moving average component can capture patterns in the noise of the time series and use them to forecast more accurately. For example, an MA(1) model with $q = 1$ would look like:

  $$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1}$$

  where θ_1 is the coefficient applied to the previous error term ϵ_{t-1}.

An ARIMA(p, d, q) model combines these components on the differenced series y'_t. For example, an ARIMA(1,1,1) model would look like:

$$y't = \mu + \phi_1 y't - 1 + \theta_1 \epsilon_{t-1} + \epsilon_t$$

where y'_t is the differenced series, ϕ_1 is the autoregressive coefficient, and θ_1 is the moving average coefficient.

In summary, the ARIMA model combines these components to predict the next value based on past values, differenced data, and past errors. The parameters p, d, and q determine how much influence each component has, allowing the model to adapt its forecasts to the unique patterns of each time series.

Selecting the appropriate parameters p, d and q for an ARIMA model typically requires careful analysis of the time series, as each series may have unique statistical properties. This task becomes increasingly challenging when working with datasets that contain large amounts of time series, each with different patterns. Manually choosing parameters for each individual series is not feasible in such cases.

This is where AutoARIMA, as implemented in the `statsforecast` library [52], becomes valuable. AutoARIMA leverages a model selection method called the Akaike Information Criterion (AIC) [53] to identify the most suitable values for p, d and q automatically. The AIC evaluates how well a model fits while penalizing for complexity, helping to avoid overfitting. Formally, the AIC is calculated as:

$$\text{AIC} = 2k - 2\ln(L)$$

here, k is the number of parameters in a model, and L is the likelihood of observing the model given the data, which is calculated using maximum likelihood estimation. Lower AIC values indicate a better balance between model complexity and fit.

With this automated selection process, AutoARIMA enables efficient and robust forecasting across a large amount of time series, without the need to manually set the parameters for each individual time series.

AutoETS

The AutoETS model [52] is an automated implementation of the ETS model family [8, 26, 54]; a family of statistical forecasting models, that are able to explicitly decompose a time series into its **E**rror, **T**rend and **S**easonality components (Figure 1c). ETS models leverage this decomposition to produce forecasts by modeling each component independently and combining their estimated future values [27]:

- **Error (E)**: The error component, E, represents the irregular fluctuations in the time series; the residuals that can not be explained through the seasonality or trend components. This component can be either *additive* (A) or *multiplicative* (M).

  In an additive model, the error term is simply added to the seasonality and trend components, implying the error remains constant despite the overall level of the time series. In a multiplicative model, the error term is multiplied with the seasonality and trend; meaning that as the overall level of a series increases, the errors increases with it. These cases are visualized in Figure 3.

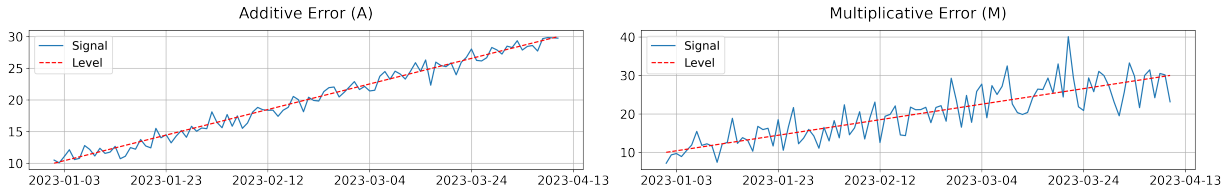


Figure 3: Error components in ETS models showing additive (left) and multiplicative (right) errors, which show the errors scale with the level. The dashed red line shows the base signal, while the blue line shows the signal with error.

- **Trend (T)**: The trend component, T, captures the underlying direction of the time series. It represents whether the series is generally increasing, decreasing or constant.

  The trend could be either *none* (N), indicating no observed underlying direction; *additive*, implying the series level increases or decreases by a fixed amount at each time step, a linear trend; *multiplicative*, meaning the series level changes with a constant percentage rather than a fixed amount, an exponential trend; or lastly, *additive damped* (Ad) or *multiplicative damped* (Md), which are variations of additive and multiplicative trends that gradually reduce the impact of the trend over time, see Figure 4.

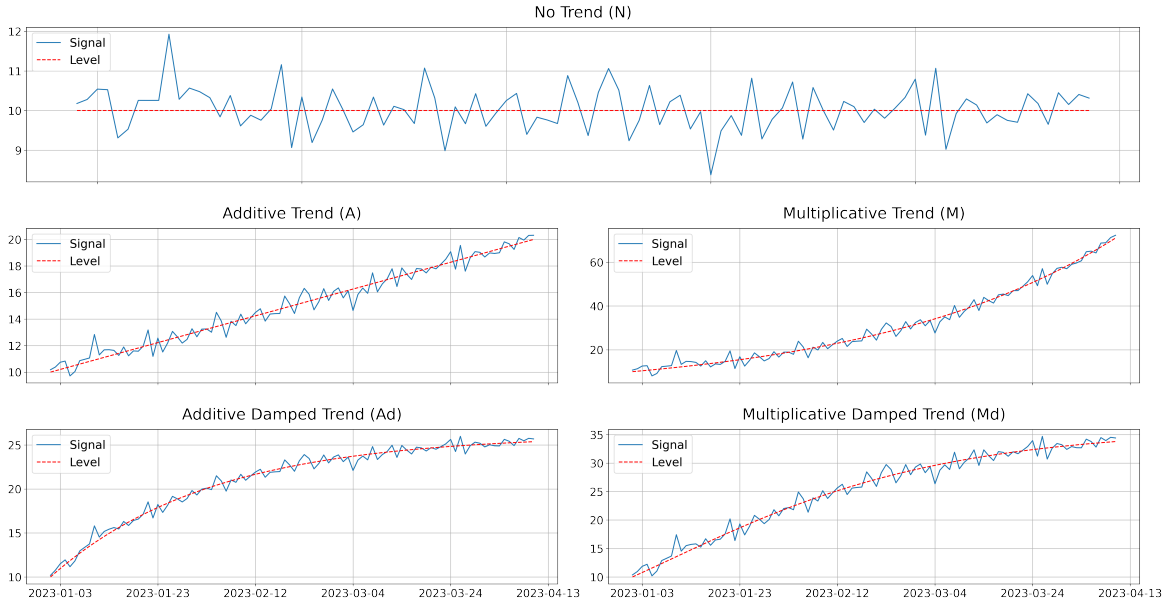


Figure 4: Trend components in ETS models showing no trend (top), and four types of trends (below): additive (top left), multiplicative (top right), additive damped (bottom left), and multiplicative damped (bottom right). The dashed red line shows the underlying trend, while the blue line shows the sequence values.

- **Seasonal (S)**: The seasonal component, S, represents the seasonality of the time series; the repeated patterns in a sequence that can be described as predictable fluctuations in the data.

  Seasonality in an ETS model can be either *none* (N), *additive* (A) or *multiplicative* (M). An ETS model where the seasonality component is none, assumes there is no seasonal effect in the data. Similar to the error component, an additive seasonality component indicates a consistent impact of the seasonality over time; where a multiplicative seasonal component indicates a seasonality effect that scales with the overall level of the series, both demonstrated in Figure 5.

An ETS(E,T,S) model uses smoothing to decompose a time series in its error, trend and seasonality component. Based on the chosen parameters, it combines these components in different ways to predict the future values of a sequence [27].

However, similar to ARIMA, selecting the optimal parameters for the ETS model requires a deeper analysis of an individual series. This becomes unfeasable when working with a

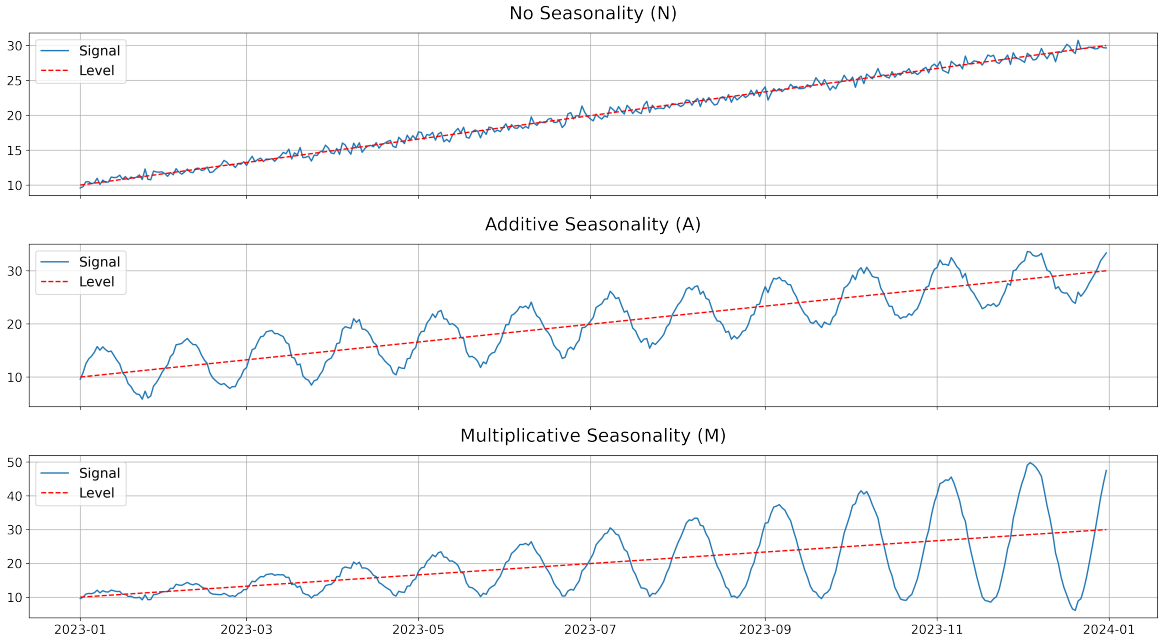


Figure 5: Seasonal components in ETS models showing no seasonality (top), additive seasonality (middle), and multiplicative seasonality (bottom). The dashed red line shows the underlying trend, while the blue line shows the sequence values with seasonal patterns.

large dataset of time series, each with different characteristics. As implemented in the `statsforecast` library [52], the AutoETS model uses the same model selection criterion as AutoARIMA: the AIC [53], automatically identifying the best-fitting parameters for each series.

In summary, AutoARIMA and AutoETS provide a set of flexible statistical models that can adapt to individual time series, without the need to manually tune them. These models offer robust baseline forecasts, setting a solid benchmark, before exploring more advanced deep learning methods.

2.2.2 Deep Learning Architectures

Unlike statistical models, deep learning architectures can learn complex patterns across multiple time series, capturing non-linear dependencies that traditional methods may miss. This ability to learn directly from data without predefined assumptions makes deep learning valuable in forecasting, especially for volatile, irregular patterns often seen in financial data [42, 55].

This thesis focuses on four distinct deep learning architectures: PatchTST [47], NHITS [49], DeepAR [50], and TimesNet [51]. These models, each with their own strengths and weaknesses, provide a more flexible alternative to statistical baselines, especially when dealing with the highly complex and inherently noisy transaction data.

PatchTST - *Transformer*

The PatchTST model is introduced in a paper titled: *"A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers"* [47], a nod to the influential paper *"An Image is Worth 16×16 Words"* [35]. This earlier work introduced the Vision Transformer (ViT), a model that outperformed the state-of-the-art Convolutional Neural Networks in image recognition, by using a novel approach of dividing the images into smaller *patches*.

Inspired by this success, with PatchTST, Nie et al. [47] introduce a similar concept to time series forecasting. Rather than evaluating time steps in a sequence individually, the series is divided into short, overlapping sub-sequences, described as patches. This allows the model to preserve local information and reduce computational complexity, since time steps are now processed in groups rather than individually.

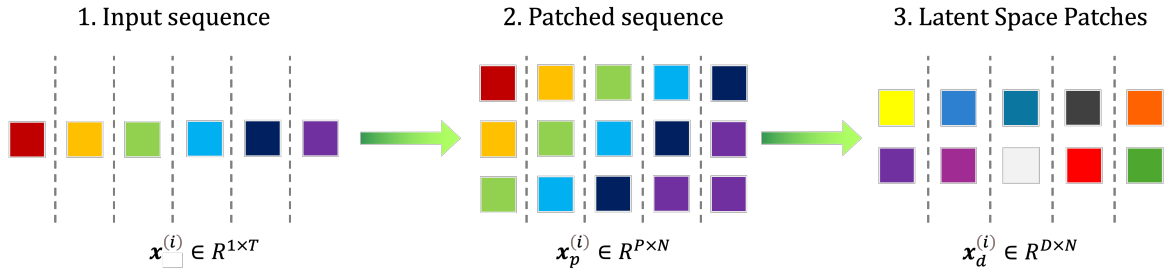


Figure 6: The patching process in PatchTST, where an input time series $x^{(i)}$ is divided into P overlapping patches of size N, with a stride S of 1. Each patch is then mapped into a latent space representation, resulting in output matrix $x_d^{(i)}$. This process enables PatchTST to learn both local and global dependencies in the time series efficiently.

The input to PatchTST is a univariate time series represented as a vector $x^{(i)} \in \mathbb{R}^{1\times T}$, where T denotes the length of the sequence. In the first step, illustrated in Figure 6, the series is divided into patches of size N, creating a sequence of patched segments $x_p^{(i)} \in \mathbb{R}^{P\times N}$, where P is the number of patches. The stride S, denotes the size of the non-overlapping region between patches.

Through a trainable matrix $\mathbf{W}_p \in \mathbb{R}^{D \times P}$ each patch is then mapped into a latent space, illustrated by step 2 and 3 in Figure 6, resulting in $x_d^{(i)} \in \mathbb{R}^{D \times N}$, a sequence of patches, where D represents the dimension of the latent space. Then, PatchTST applies the transformer architecture to process this sequence of latent space patches. Each patch in the latent representation $x_d^{(i)} \in \mathbb{R}^{D \times N}$ is treated as a token, similar to how words are treated in natural language processing.

The self-attention mechanism in the transformer block, enables the model to adjust the importance of each patch relative to others. This mechanism allows the model to focus -hence the term self *attention*-, on specific patterns that are most relevant for forecasting. For instance, when a time series has a strong weekly pattern, the model is able to pay more attention to patches from previous weeks.

Through its ability to learn both short-term fluctuations within each patch, and long-term dependencies over a sequence, PatchTST has proven successful over widely used benchmarking datasets [23]. However, its performance in financial forecasting remains relatively unexplored.

2.3 Foundation Models

The field of Artificial Intelligence is seeing a shift in how models are designed and applied, driven by the emergence of a new class of models, known as *foundation models* [56]. Although built using existing architectures, foundation models stand out due to their gigantic scale, both in model size and amount of training data. Unlike traditional models, which are often trained and specialized for single applications, foundation models are able to generalize across a wide range of tasks [15, 57, 58, 59, 60], see Figure 7.

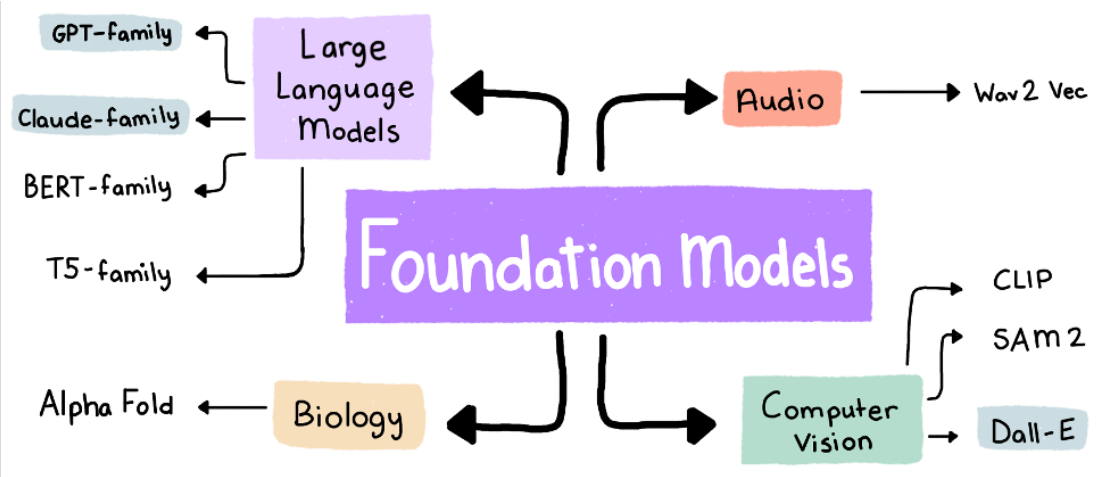


Figure 7: Simplified taxonomy of the classes of Foundation Models with generative models highlighted. This visualization serves to clarify the differences between terms such as Foundation Models, Large Language Models and Generative Pre-Trained Transformers [61].

2.3.1 Large Language Models

The most well known sub-class of these foundation models, focused on language tasks such as generating, summarizing and translating text, is referred to as *Large Language Models* (LLMs). LLMs are designed to interpret and generate human language by learning patterns within massive amounts of text data. Within this group, some models are generative, meaning they are, for example, capable of constructing coherent language or images. Examples include the GPT-model family [15], the driving force behind famous chat-bot ChatGPT; and the Claude model family, such as Sonnet 3.5 [62], which excels in generating code.

Not all LLMs are generative. Models like those in the BERT [36] and T5 [63] families are designed primarily for understanding and processing language, rather than generating it. The T5 model, for example, treats each task as a text-to-text problem; taking text as the problem input and producing text as solution output.

2.3.2 LLMs for Time Series Analysis

While foundation models like LLMs are initially designed for language-based tasks, their success has inspired research into the adaption of LLMs in new domains, such as time series analysis [16, 18]. Here, the challenge lies in bridging the gap between the LLM's original

text data and the numerical nature of the time series data [17]. This requires transforming the time series data in such a manner that it can be interpreted correctly by an LLM.

The most straightforward method to do this is by encoding a time series as a string of numerical digits and directly prompting the large language model [64, 65], visualized in Figure 8. For example, a temperature series could be converted into a sentence: *"The temperature for the last 30 days was* $x_1, x_2, \cdots, x_{30}$*; predict tomorrow's temperature."*

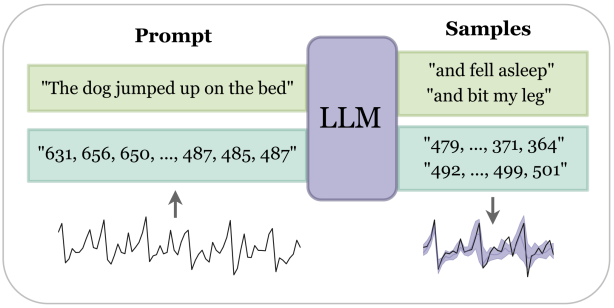


Figure 8: Visualization of directly prompting an LLM and by encoding a time series as a string of numerical digits, and leveraging the LLM's next-token-prediction capabilities [65].

This effectively transforms the time series forecasting task into a text generation problem, and attempts to leverage the LLM's inherent next-word prediction capabilities. However, LLMs represent their input data as *tokens* through a process known as *tokenization*. These tokenizers are generally designed for text, meaning they struggle to handle numerical data correctly [66]. Especially floating point numbers are challenging to tokenize consistently; consider the number 3.14159, which might be tokenized into multiple different chunks.

Tokenizers in LLMs work well for language because language has a finite vocabulary, allowing for a finite set of tokens. However, for continuous numerical data, like time series values, there is no practical way to directly create a token for every possible value. The technique of quantization [67] is used to tackle this problem by converting the continuous values into a fixed set of discrete categories; a vocabulary for continuous values, if you will. Quantization involves mapping [68] each value in a time series to its closest match in the vocabulary.

This approach allows LLMs to handle time series data without needing a unique token for every possible value. Despite quantization improving the consistency of numerical tok-

enization, it can introduce information loss, especially in time series with subtle variations [67].

To more carefully align the modalities, a specialized *encoder* can be trained to represent time series data in a format that matches the LLM's representations; a process known as *aligning* [17, 69]. The approach involves transforming the time series into embeddings, a vector representation, and aligning them with the LLM's own language-based embeddings using contrastive learning [70].

In contrastive learning, the model is trained to bring similar representations closer together, while pushing dissimilar ones apart. This technique, also used in the CLIP [60] foundation model to align the image and text modalities, helps to align time series data with the language model's structure. However, the creation of an effective alignment between time series and the LLM's embeddings requires complex contrastive learning set ups that can produce inconsistent results [17, 66].

2.3.3 Foundation Models for Time Series

Despite the progress in adapting LLMs to numerical data, it remains challenging to align numerical time series data with existing pre-trained large language models. This gap in aligning the modalities led to research into development of a dedicated time series foundation model [19, 20, 21, 71, 22]. A model pre-trained on large amounts of data, that can forecast unseen sequences, without needing to be trained on them; known as *zero-shotting*.

Over the past year, multiple first generation foundation models have demonstrated zero-shot forecasting capabilities comparable to dedicated deep learning forecasters specifically trained for individual tasks [72]. The models are trained on extensive and diverse time series datasets, covering domains such as climate, healthcare, finance and more. The TimeGPT model, for example, was trained on the largest publicly available collection of time series, containing over a billion training points [20].

This thesis focuses on the ability of leveraging an existing large language model architecture, by evaluating a family of forecasting foundation models developed by AWS, called *Chronos* [19].

Chronos - *Learning the Language of Time Series*

Chronos is a framework of pre-trained probabilistic forecasting models that are trained using the existing architectures of the T5 large language model family [63]. It is trained on a large amount of public datasets, supplemented with synthetic data, generated by augmenting the original data using a newly proposed augmentation scheme, *TSMixup*; and a novel data generation scheme using Gaussian Processes [73], *KernelSynth*.

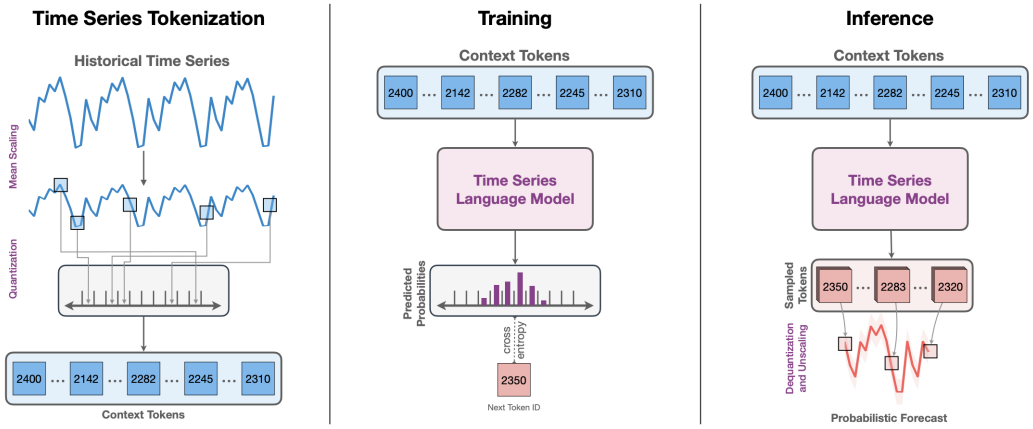


Figure 9: High-level overview of the Chronos framework; on the left, illustrating the process of tokenizing the continuous time series data through mean scaling and quantization; the middle, demonstrating how patterns in the tokenized data are learned using the T5 architecture; on the right, depicting the inference stage, where probabilistic forecasts are generated by sampling multiple tokens for each future time step. [19].

The Chronos framework (Figure 9) begins with transforming the continuous time series data into a language-like format using *scaling* and *quantization*. First, *mean scaling* is applied to standardize the range of the different time series. For a given time series $\{x_1, x_2, \ldots, x_n\}$, mean scaling normalizes each value by dividing it by the mean absolute value of the series, defined as:

$$\tilde{x}_i = \frac{x_i}{\frac{1}{n}\sum_{j=1}^{n} |x_j|}$$

This scaled data is then quantized, mapping each continuous $\tilde{x}_i$ to a discrete category or token. Through this step of scaling and quantization, the infinite range of continuous values is reduced to a finite set of tokens, similar to the vocabulary of words in a language model. This transformation allows Chronos to interpret the time series data as a sequence of tokens, similar to how large language models process text.

Once the continuous time series has been scaled and quantized into a sequence of tokens, these tokens are fed into the T5 large language model architecture. Chronos leverages the full T5 family; a transformer-based family of models, varying in number of layers and parameters. In general, the smaller Chronos models will be suitable for faster, less specific forecasting, where the larger models are useful for more precise forecasting tasks.

During inference, Chronos generates probabilistic forecasts by predicting the next token in the sequence, one step at a time. Starting with a given sequence of tokens, the model samples multiple tokens for each future time step, essentially creating multiple possible forecasting trajectories.

Once the tokens for each trajectory are generated, they are converted back to continuous numerical values through a process of dequantization and unscaling, reversing the initial quantization and scaling transformations. This results in a distribution of possible outcomes for each time step, where the range of values reflects the model's confidence in a forecast. This probabilistic approach is especially valuable in finance applications such as risk management.

2.4 Foundation Models for Financial Forecasting

In finance, the ability to forecast accurately without extensive task-specific training, could be highly useful. Foundation models enable institutions to quickly generate predictions on new datasets, reducing both the time and computational resources needed for model training. Especially in finance, where data is often inherently noisy and unpredictable, the adaptability of foundation models is promising.

Although foundation models do incorporate some financial data in their pre-training [74], many commonly used benchmarking datasets are not focused on time series from the finance domain; the often used Darts [75] and Autoformer [45] datasets even contain no financial time series. Additionally, the financial data used during training and evaluation is often limited to publicly available sources, such as stock prices or exchange rates [74].

This raises the question how well foundation models can adapt to other, more private and highly protected financial datasets. This private financial data, such as transaction records, can contain unique patterns and irregularities that are specific to individual finan-

cial behaviors; patterns that a foundation model has not been able to encounter during its pre-training. Testing a foundation model's performance on this new, unseen class of data, can reveal whether the pre-trained architecture can generalize effectively, and learn to recognize unfamiliar patterns.

2.5 Data Science at ING Bank

At ING, the largest bank in the Netherlands, the Data Science group within the Wholesale Banking Advanced Analytics (WBAA) team, specializes in building AI-driven solutions for financial decision-making. The team tackles challenges from forecasting excess cash and detecting recurring payments to applying large language models for environmental, social and governance analysis.

Processing millions of transactions daily, ING generates huge, private datasets, providing a unique environment for data analysis. This data offers insights into financial patterns and behaviors, different from those found in public data; creating an ideal setting to explore advanced approaches in banking analytics.

3 Research and Methodology

The focus in this chapter shifts from the more general theoretical background to the specific research that is addressed in this thesis. First, the research question is defined, after which the data collection, model training and evaluation methods are presented.

3.1 Research question

From the theoretical background, it can be concluded that foundation models are promising time series forecasters. However, their ability to recognize patterns within the class of private and inherently noisy transaction data is yet to be explored.

In this thesis, the forecasting capabilities of large language model architecture Chronos, pre-trained on time series data, is evaluated in the domain of transaction data. The goal of this thesis is to better understand the potential and limitations of LLM architectures as time series forecasters. The research question is as follows:

"To what extent can large language model architectures be applied to financial time series forecasting, in comparison to traditional statistical and deep learning models?"

To answer this, four sub-questions are formulated, which together will provide an answer to the above question.

1. How well does a large language model architecture, pre-trained on time series data, forecast *unseen transaction data*, compared to state-of-the-art statistical and deep learning models?

This question is investigated by evaluating the Chronos model and six statistical and deep learning baselines on financial data, specifically private transaction data from the largest bank in the Netherlands, ING. The models are tasked with forecasting account balances over 1-day, 7-day, 14-day and 30-day forecasting horizons, providing insight into the zero-shot capabilities of a foundation model on financial forecasting.

2. How well does a large language model architecture forecast transaction data, when it is pre-trained on time series data *and fine-tuned on transaction data*?

To answer this question, the Chronos model is fine-tuned on a subset of the transaction data, updating its parameters to better align with specific patterns within this dataset. The evaluation then compares the fine-tuned predictions with the zero-shot and baseline predictions.

3. How do *intrinsic characteristics* of financial time series affect the performance of a large language model architecture and other forecasting models?

This question explores how time series characteristics, specifically approximate entropy, seasonality and trend, impact model performance. By comparing how models perform on time series with different properties, this question aims to provide insight into whether specific sequence characteristics influence the accuracy of the models.

4. How accurate is the *probabilistic output* produced by a large language model, and how do they compare to those generated by traditional models?

Each model outputs a confidence interval around its predictions, providing an estimate of their forecasting uncertainty. This question assesses how well calibrated these intervals are by inspecting whether predictions fall within a model's confidence range at expected rates.

The next sections include a detailed explanation of how the experiments answering these questions have been conducted.

3.2 Data Collection and Processing

The transactional data analyzed originates from ING's booking system, Profile, which records the balance of an account at the close of each day. The dataset focuses on all ING's Dutch *Transaction Services* Wholesale Banking clients, who maintained an active account from January 1, 2022 to October 10, 2024. This results in a time series with 1,014 daily steps per account, providing information of account activity over this period. In this section the steps taken in order to transform the data into model input are described.

3.2.1 Data Preprocessing

The dataset includes $10,628$ accounts associated with $4,812$ clients, with some clients having multiple accounts. Specifically, the clients can be grouped under 502 *ultimate parents*; which can be described as overarching entities or companies under which multiple

accounts operate. To minimize noise in the data, the accounts are aggregated by their ultimate parent. This aggregation effectively neutralizes internal transactions, where money is transferred between different accounts of the same client.

To prepare the data for forecasting, the remaining missing values are *forward-filled*, meaning each gap is filled with the last known balance, to create a continuous sequence. Following this step, static accounts, those with little to no transactional activity, were excluded. An account is considered static if its balance changed less than half of the time, ensuring that the forecasting models can focus on active patterns.

The accounts in this dataset span different scales, making it hard to compare them directly. To address this, each time series is scaled using *min-max normalization*. This scaling technique preserves the original shape and trends of the data, while representing each time series on a common $[0, 1]$ scale. Figure 10 shows this process, with the original time series on the left and the normalized sequence on the right.

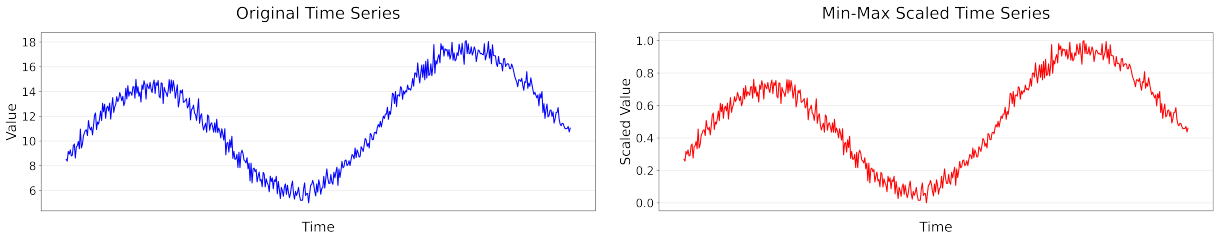


Figure 10: Illustration of min-max normalization applied to account balance time series. Each time series is independently scaled to the [0, 1] range, while preserving the patterns and trends of the original sequence.

The resulting dataset is a table of size 1014×278, where the first column represents the date and each other column represents a time series sequence of 1,014 days.

3.2.2 Train and Test Splitting

To evaluate model performance accurately, the data needs to be separated into sets of training- and test data. In time series forecasting, this separation is usually done by "cutting off" the most recent period of data, which simulates the horizon that is being forecasted.

For this research, the objective is to evaluate forecasting models across four different fore-

casting horizons: 1 day, 7 days, 14 days and 30 days. However, simply assigning the last 30 days as the test period can introduce bias into the evaluation. For instance, if unusual economic activity impacted that specific month, the model's performance might be under- or overestimated by the evaluation.

To address this issue, the last 360 days of data are divided into twelve consecutive 30-day intervals and separated from the training data; which is the nearly two years of data preceding this 360-day testing period. From these twelve intervals, six periods, spaced one month apart across the year, are selected for testing. By testing across multiple periods, the evaluation covers a range of seasonal and economic conditions, reducing the impact of any specific trends or anomalies, as seen in the third testing period in Figure 11.

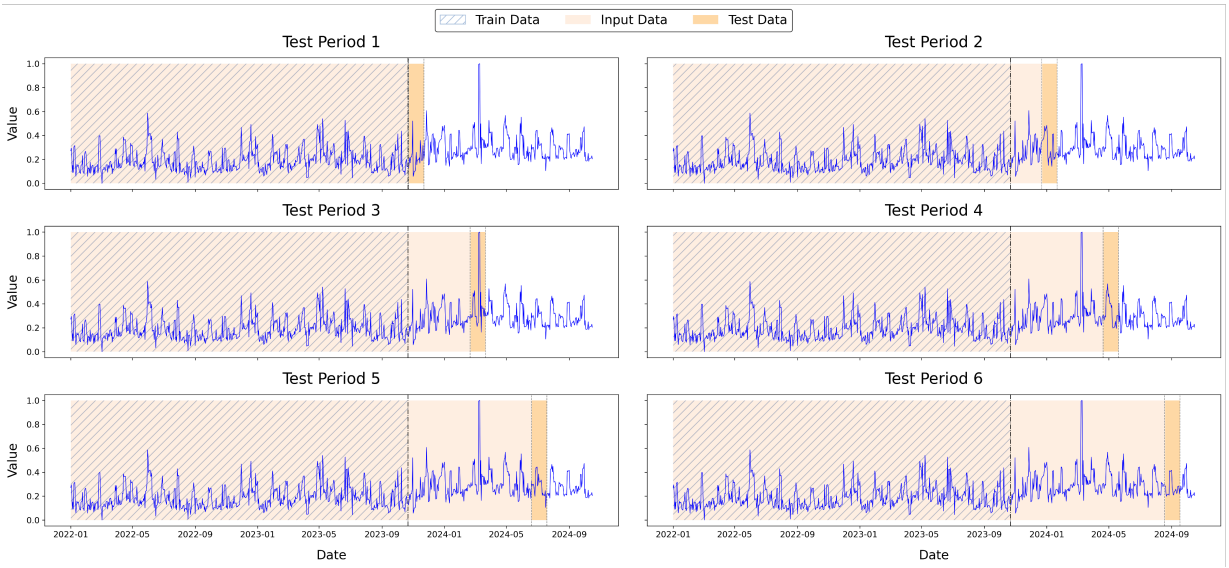


Figure 11: Visualization of the six different testing periods used to evaluate model performance. Each model is trained once on the training data, however the input data is extended up to the beginning of each test period.

Ideally, each model would be retrained up to the start of each testing period, to capture the most recent trends in the data before forecasting. However, training six separate models, one for each of the forecast intervals, requires significant computational resources. As a solution, the model is trained once on the full training period, the striped area in Figure 11. The input data, used during inference, is extended up to the beginning of each test period. This allows forecasts to be made using the most recent data, despite not being

trained on it.

In addition to creating these test and train periods, the time series are split into *in-sample* and *out-of-sample* subsets. Specifically, 20% of the sequences are removed entirely from the training dataset, creating a subset of unseen data, the out-of-sample set. This allows for two types of evaluation: in-sample, which measures how well the model learned the patterns within the training data; and out-of-sample, which estimates the model's ability to generalize to new, unseen data. Together, these methods provide an insight into the model's pattern recognition capabilities and adaptability to new tasks.

3.3 Model Training

All models are trained within ING's Data Analytics Platform (DAP), a secure environment developed for data scientists at ING. By providing centralized access to datasets and computational resources, DAP enables efficient analysis of large-scale, private financial data.

In order to create a straightforward and reproducible workflow, a series of Jupyter notebooks is created on DAP, covering each stage of the training pipeline. These notebooks, available in the thesis GitHub repository[1], allow a flexible way to make adjustments to the pipeline, such as model selection, parameter tuning and visualizing.

Following the data preprocessing and train-test splitting steps, outlined in Section 3.2.1 and 3.2.2 respectively, this section details the training processes used for the forecasting models used in this research.

3.3.1 Statistical Models

In line with the theoretical background discussed in Section 2.2.1, the statistical forecasting models used in this research are AutoARIMA and AutoETS from the `statsforecast` [52] library. Both of these models have built-in mechanisms to select the optimal parameters related to autoregression, integration and moving average (AutoARIMA); or error, trend and seasonality (AutoETS).

[1] `https://github.com/didiermerk/forecasting`

However, the models do not automatically detect or handle the length of the seasonal component, which is required as an input parameter for the model to train. Additionally, traditional statistical models, including ARIMA and ETS, can not handle time series with multiple seasonal patterns, such as combinations of weekly, monthly or yearly trends, that are common in financial data.

In this study, the *Multiple Seasonal-Trend decomposition using Loess* (MSTL) model [52, 76] is used to address these limitations. The MSTL model enables a decomposition of each time series into an error component, a trend component and, possibly, a component consisting of multiple seasonalities. This approach can handle more nuanced seasonal patterns in the data, while AutoARIMA and AutoETS serve as the core forecasters for the error and trend components, shown in the code below:

```
# Initialize the MSTL model with AutoARIMA backbone
mstl_model = [MSTL(
                season_length=[7, 30, 365], # weekly, monthly and yearly pattern
                trend_forecaster=AutoARIMA(max_p=4, max_q=4)
            )]
```

To determine the best-fitting seasonality for each sequence, a notebook is created to evaluate all possible combinations of seasonal components (*none*, *weekly*, *monthly* or *yearly*) with the MSTL decomposition. By selecting the configuration with the lowest average error term, it is possible to automatically estimate the (multiple) seasonality of a time series.

This enables the AutoARIMA and AutoETS models to be used in a fully automated manner, while incorporating more complex seasonal patterns when relevant. To complete the set of statistical baselines, the straightforward, *Naive* model is included. This model simply forecasts each future value as the most recent known value of the time series, resulting in a constant value over the forecast horizon. Together, these three models set a robust statistical baseline for time series forecasting.

3.3.2 Deep Learning Models

The deep learning models in this study are implemented using the `neuralforecast` library [77]. Four architectures are investigated: PatchTST, NHITS, DeepAR and TimesNet. Each model is configured with specific hyperparameters, which are manually tuned to optimize performance for this forecasting task. Table 2 provides an overview of the key hyperparameters, including the loss that the models were trained on in order to output probabilistic forecasts, the amount of epochs and any model-specific settings.

Model	**Loss**	**Input Size**	**Max Steps**	**Model Specific Settings**
PatchTST	`MQLoss`	`15 * fh`	3500 (500 epochs)	`patch_len=64`
NHITS	`MQLoss`	`15 * fh`	3500	`n_freq_downsample=[2, 1, 1]`
DeepAR	`DistributionLoss`	`15 * fh`	3500	`distribution='Normal'`
TimesNet	`MQLoss`	`15 * fh`	3500	-

Table 2: Overview of the hyperparameters that were used to train the four deep learning models, after tuning them manually. The forecasting horizon, `fh`, in this thesis is 30 days. The right-most column indicates any model-specific settings.

3.3.3 Chronos

The foundation model framework used for forecasting in this research is Chronos, a family of pre-trained time series models built on the T5 language model architectures. The Chronos family includes several model sizes, allowing for different levels of computational efficiency and forecasting precision, based on the specific model variant used (see table 3). In this study, two versions of Chronos, *Chronos-small* and *Chronos-large*, are used to evaluate zero-shot performance across both a resource-efficient model variant and a high-capacity model.

Training and fine-tuning for Chronos is implemented using a combination of the official Chronos GitHub[2] package and the `AutoGluon` library [78]. The Chronos-small model architecture was fine-tuned on 80% of the dataset, using the HuggingFace `Trainer` class, taking six hours to complete 500 epochs using a single GPU.

[2] `https://github.com/amazon-science/chronos-forecasting.git`

Model	Parameters
`Chronos-tiny`	8M
`Chronos-mini`	20M
`Chronos-small*`†	46M
`Chronos-base`	200M
`Chronos-large*`	710M

Table 3: Parameter counts for the different models in the Chronos foundation model family. The models highlighted with an * are used for zero-shot evaluation, and those highlighted with a † are used for fine-tuning.

3.4 Evaluation

To ensure that the forecasting models are reliable and trustworthy, their performance needs to be evaluated. This section details the three key evaluation aspects: forecasting accuracy, approximate entropy and confidence interval reliability. Together, these aspects aim to determine not only how close a model's predictions are to the ground truth, but also how they handle specific characteristics within the data and quantify the reliability of their forecasts.

3.4.1 Forecasting Accuracy

Forecasting accuracy measures the closeness of a model's predictions to the actual observed values in the test set. Three metrics are used to evaluate this: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE); each providing different perspectives on the prediction errors:

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \qquad \text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

Here, y_t is the ground truth value at time step t, $\hat{y}_t$ is the forecasted value at time step t and n is the number of time steps that have been predicted; the forecasting horizons being either 1, 7, 14 or 30.

The MAE calculates the absolute differences between the prediction and ground truth values, $|y_t - \hat{y}_t|$, for each time step in a sequence, divided by the number of time steps.

This metric, provides an intuitive sense of how much, on average, your predictions differ from reality. However, when comparing series with different scales, the MAE can be skewed by the sequences with larger values.

The MAPE combats this problem by looking at the relative error of the predictions. For each time step, the MAPE calculates the error, $|y_t - \hat{y}_t|$, as a proportion of the ground truth value, y_t. Although in this case, when ground truth values approach 0, the MAPE can become very large and not accurately represent the errors of the model.

The RMSE presumes a slightly different approach of accuracy evaluation, by placing emphasis on larger errors through squaring them $(y_t - \hat{y}_t)^2$. As a result, the RMSE penalizes models that occasionally make larger errors, favoring models that keep large errors to a minimum.

For every model, these metrics are calculated for each individual time series and subsequently aggregated by taking the median value across all series. This results in three core evaluation metrics indicating a model's accuracy over multiple forecasting horizons.

3.4.2 Approximate Entropy

In financial time series, specific sequences may follow predictable patterns with little fluctuations, allowing most models to forecast them well. However, some series exhibit inherent unpredictability, noise or complexity, making them more difficult to forecast accurately.

To evaluate a model's capability of handling these more complex series, the concept of *Approximate Entropy* (ApEn) [79] is introduced; a statistical technique used to quantify the unpredictability of a time series sequence. The approximate entropy of a sequence of length N is calculated as follows:

$$\text{ApEn}(m, r, N) = \phi^m(r) - \phi^{m+1}(r)$$

It is best to consider this function intuitively: the ApEn attempts to capture the amount of similar patterns in the sequence. The length of the patterns that are compared is given by m. The next parameter, r, serves as a distance threshold, setting the sensitivity of deciding which patterns are considered similar.

First, the $\phi^m(r)$ value is calculated, which captures the frequency of similar patterns in a

time series of length N and is defined as:

$$\phi^m(r) = \frac{1}{N-m+1} \sum_{i=1}^{N-m+1} \ln C_i^m(r)$$

It captures the idea of going through each possible pattern of length m within a sequence, comparing them to all other patterns of the same length and deciding whether they are similar or not based on threshold r. $C_i^m(r)$ represents the fraction of patterns similar to the pattern at point i. If this value decreases when compared to slightly longer patterns, $\phi^m(r) - \phi^{m+1}(r)$, the sequence is considered predictable, see Figure 12.

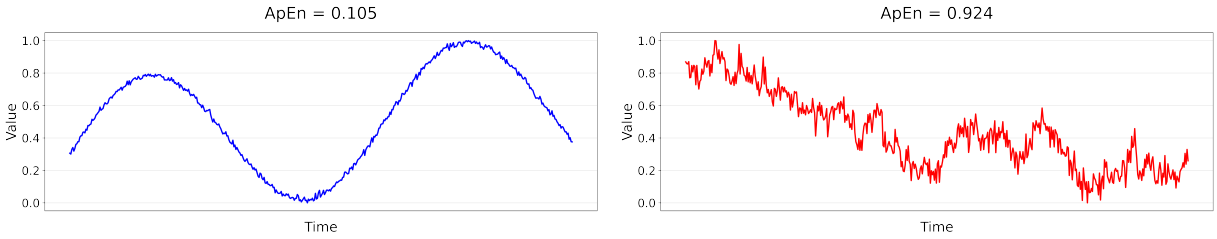


Figure 12: Illustration of the approximate entropy (ApEn) of a periodic (left) and noisy (right) sequence. The ApEn is a value between 0 and 1, where a higher value indicates an inherently more unpredictable sequence.

By evaluating model performance against the approximate entropy of a sequence, it is possible to determine whether certain models are better suited to forecast time series that are inherently more unpredictable.

3.4.3 Confidence Interval Reliability

Confidence intervals provide a probabilistic measure of the uncertainty associated with a model's predictions. In this study, each model outputs 60-, 70-, 80- and 90% confidence intervals around its forecasts. Ideally, for a 90% confidence interval, approximately 90% of the actual observations should lie within the predicted range. By comparing the expected coverage with the actual coverage, it is possible to assess how well-calibrated the models are in estimating their prediction uncertainty.

4 Results

In this section, the main results of the forecasting task on the balances dataset are presented. The analysis focuses on three main aspects: forecasting accuracy across multiple horizons, a model's ability to handle complex time series with inherent unpredictability and the reliability of a model's confidence intervals.

4.1 Forecasting Accuracy

The forecasting accuracy of the statistical, deep learning and foundation models is evaluated across four forecasting horizons: 1 day, 7 days, 14 days and 30 days. Three core metrics are calculated (Section 3.4.1), providing a detailed comparison of model performance across varying forecast lengths.

		Statistical			Deep Learning				Foundation Models		
Metric	**Horizon**	**Naive**	**ARIMA**	**ETS**	**NHITS**	**PatchTST**	**TimesNet**	**DeepAR**	**Chronos-S**	**Chronos-L**	**Chronos-FT**
MAE	1 day	0.0091	0.0237	0.0239	0.0158	0.0116	0.0258	0.0123	0.0083	**0.0077**	0.0102
	7 days	0.0323	0.0459	0.0456	0.0403	0.0320	0.0385	0.0346	0.0293	**0.0280**	0.0338
	14 days	0.0449	0.0571	0.0580	0.0485	0.0403	0.0473	0.0450	0.0397	**0.0389**	0.0429
	30 days	0.0517	0.0616	0.0636	0.0550	0.0446	0.0514	0.0524	0.0460	**0.0440**	0.0480
MAPE	1 day	3.5840	9.2107	8.9328	6.7932	5.0918	10.8874	5.2234	3.3121	**3.2282**	3.9072
	7 days	13.4556	18.2455	18.7574	16.2607	13.1425	16.6501	14.0777	12.2991	**12.1936**	14.4643
	14 days	19.8646	23.9450	24.3448	21.6301	17.7806	20.8744	19.2472	17.5705	**17.3186**	18.1345
	30 days	21.6435	24.8841	25.5652	22.6731	18.9188	21.5849	21.6900	19.1657	**18.3518**	19.6172
RMSE	1 day	0.0091	0.0237	0.0239	0.0158	0.0116	0.0258	0.0123	0.0083	**0.0077**	0.0102
	7 days	0.0415	0.0571	0.0584	0.0491	0.0398	0.0467	0.0433	0.0368	**0.0365**	0.0449
	14 days	0.0587	0.0739	0.0763	0.0616	**0.0520**	0.0588	0.0585	0.0548	0.0529	0.0581
	30 days	0.0704	0.0806	0.0830	0.0714	**0.0612**	0.0670	0.0685	0.0644	0.0625	0.0661

Table 4: This table presents the in-sample forecasting accuracy of various time series models, categorized into Statistical, Deep Learning and Foundation Models, across four forecasting horizons (1 day, 7 days, 14 days, and 30 days). The metrics used to evaluate model performance include the median Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE) over all time series, averaged over the six test periods. The values in **bold** indicate the best-performing model for a specific metric and horizon, with the second best models underlined.

Table 4 summarizes the in-sample accuracy results across horizons and metrics, with the best-performing models highlighted in bold for each metric and horizon. The foundation models, particularly Chronos-Large, consistently achieve the lowest errors across most

horizons and metrics, suggesting strong zero-shot performance on the unseen data. Surprisingly the fine-tuned Chronos model, does not achieve scores that are higher than his non-fine-tuned counterpart, Chronos-small.

Among the deep learning models -trained on this specific task-, PatchTST performs the best, achieving the second best results over a 30-day forecasting horizon. Remarkably, the Naive model outperforms the dedicated deep learning models NHITS and DeepAR over multiple metrics and horizons, this is further discussed in Section 5.1.

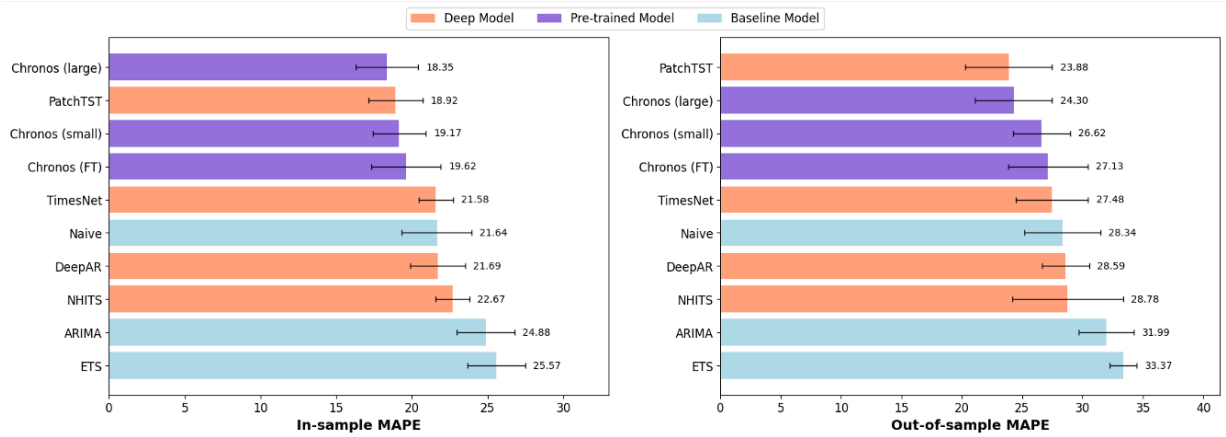


Figure 13: Comparison of in-sample and out-of-sample MAPE values across different models for the 30-day forecasting horizon. For each model, the median MAPE over all time series is calculated for six test periods (Section 3.2.2). The colored bars represent the average MAPE of a model over six test periods, with a lower MAPE indicating better performance. The error bars represent the variability in MAPE across the six test periods. The chart on the left displays the in-sample MAPE, showcasing the forecasting accuracy within the training data, while the chart on the right shows out-sample MAPE, highlighting the models' performance on unseen data. Chronos-large and PatchTST, both based on a transformer architecture, exhibit the best performance.

In Figure 13 the models' abilities to forecast 30 days ahead for both the in-sample and out-of-sample subsets are illustrated. This figure shows that the large variant of the Chronos architecture performs on par with the PatchTST model on both seen and unseen data.

It is worthy to note that the statistical models, which forecast each time series independently and the foundation models, all perform worse on the out-of-sample subset of data. Since there should not be a difference in performance between seen and unseen data for these models, this possibly indicates that the randomly chosen subset of out-of-sample time series is inherently harder to forecast.

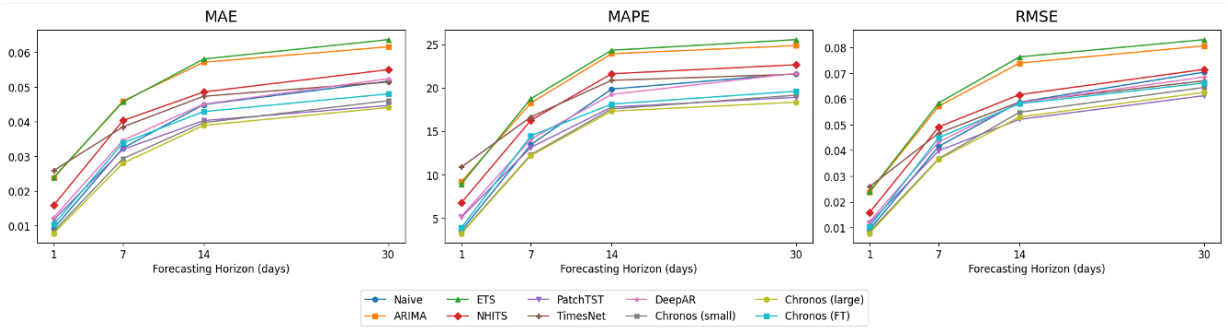


Figure 14: The model metrics illustrated over the forecasting horizons. Each sub-plot demonstrates how model accuracy degrades over longer forecasting horizons, with Chronos and deep learning models generally maintaining lower errors across metrics.

To understand how forecasting accuracy changes between short-term and long-term predictions, Figure 14 plots each metric across the forecasting horizons. As expected, prediction accuracy decreases for all models when the horizon length increases, reflecting the difficulty to make long-term predictions. Nevertheless, the Chronos-large model is consistently the best model across all forecasting horizons.

For the 1-day forecasting horizon the Naive model performs well, since it assumes that the most recent value will remain unchanged in the immediate future. However, as the horizon increases, the accuracy of the Naive model declines relatively faster than the other models. For example, TimesNet, which on the 1-day horizon has an MAE nearly three times that of the Naive model, outperforms the Naive model on the 30-day horizon.

4.2 Seasonality and Approximate Entropy

To better understand how model performance varies based on time series characteristics, this section examines the influence of seasonal patterns and approximate entropy on forecasting accuracy.

Figure 15 shows the performance of all models on different seasonalities. On average, models forecast most accurate on the group of sequences containing a single periodic pattern. The statistical models report roughly the same performance across different types of seasonality, meaning they can forecast time series with multiple seasonalities just as accurate as sequences with at most one periodic pattern. The foundation and deep learning models,

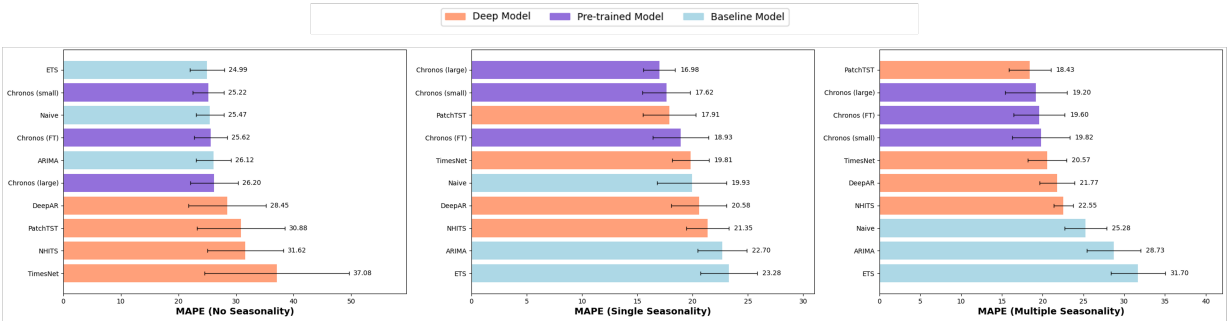


Figure 15: This figure illustrates the Mean Absolute Percentage Error (MAPE) across models for time series with different seasonal patterns: no seasonality, single seasonality and multiple seasonalities. Each bar represents the MAPE for a model within a specific seasonality group, with lower MAPE values indicating better performance. The error bars reflect the variability across test periods.

however, show a clear ability of being better forecasters when the time series contains some form of seasonality.

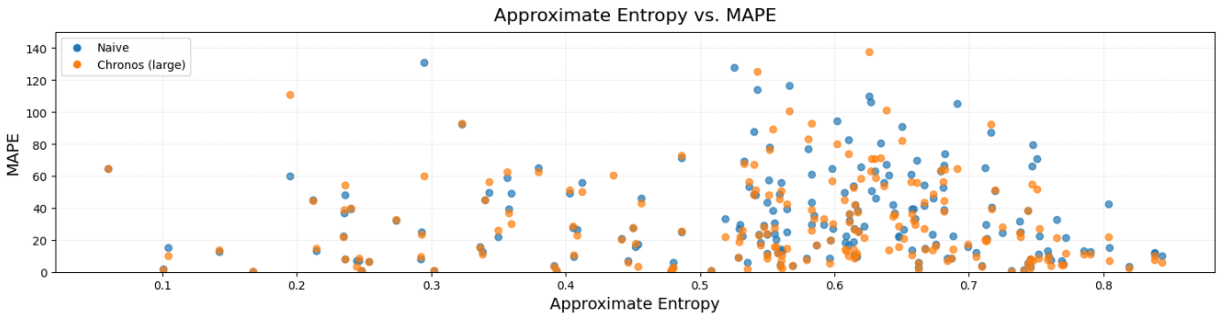


Figure 16: This figure shows the relationship between Approximate Entropy (ApEn) and Mean Absolute Percentage Error (MAPE) for two selected models, Chronos-Large and Naive. Each point represents a time series, with approximate entropy on the x-axis and the corresponding MAPE on the y-axis. Higher ApEn values indicate a sequence with a greater amount of irregular patterns in the data.

The scatter plot in Figure 16 illustrates the relationship between the approximate entropy and MAPE for two selected models, Naive and Chronos-large. Most time series in the dataset have an ApEn value between 0.5 and 0.8. Although a clear pattern can not be directly observed, there appears to be some tendency for higher MAPE values to occur further towards the right of the plot, suggesting that more unpredictable (higher entropy) sequences are generally harder to forecast. However, it would be too early to draw any conclusions about differences between the Naive and Chronos model based on this plot.

4.3 Confidence Interval Reliability

To evaluate the reliability of each model's confidence intervals, Figure 17 presents the calibration results at four confidence levels: 60%, 70%, 80% and 90%. The calibration percentage reflects the proportion of ground truth values that fall within the predicted confidence interval, with the dotted vertical line indicating the ideal calibration percentage for each interval.

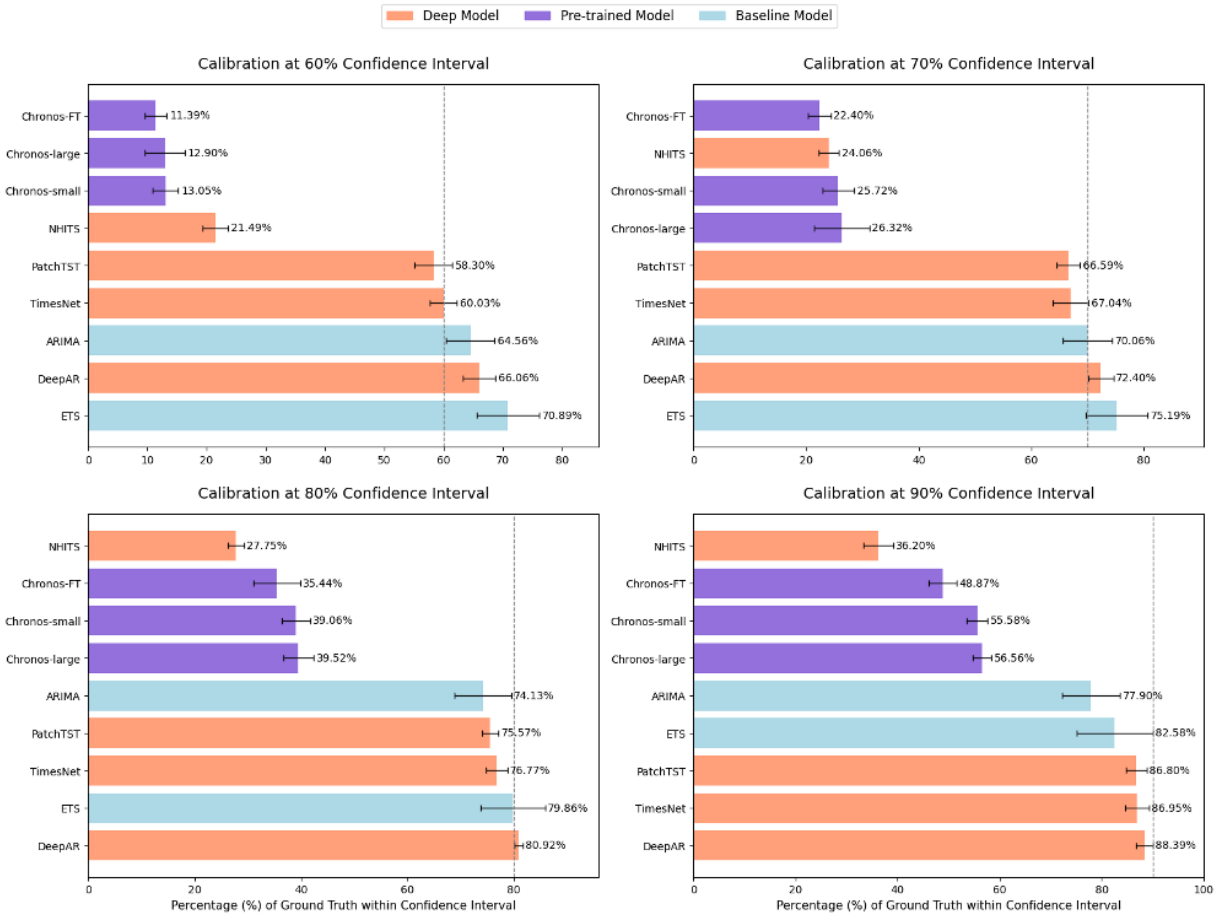


Figure 17: This figure demonstrates the calibration of the forecasting models at 60%, 70%, 80% and 90% confidence intervals. Each bar represents the percentage of ground truth values falling within the model's confidence intervals at each level. The dotted vertical line indicates the claimed confidence level. Pre-trained models (Chronos variants) display lower calibration percentages across all intervals, indicating over-confidence, while traditional and deep learning models generally achieve higher calibration percentages, closer to the ideal values. The error bars represent the variability in calibration accuracy across different test periods.

While most models are able to quantify their prediction confidence accurately, the foundation models consistently show lower calibration percentages across all confidence intervals. This outcome does not imply poor forecasting accuracy, as shown in table 4 and Figure 13, where the foundation models exhibit strong performances in terms of accuracy. It rather indicates that the models are often overconfident in their ability to forecast the unseen data, potentially underestimating the range of possible forecasts.

Interestingly, fine-tuning the foundation model on the data did not reduce its overconfidence; instead, it even increased the model's certainty in its predictions, making it the most poorly calibrated model of the Chronos variants. A similar pattern is seen in the NHITS model, which also demonstrates lower calibration across the confidence levels.

This result points to a possible trade-off between achieving high forecasting accuracy and maintaining reliable confidence intervals for foundation models. In some cases, such as predicting short-term sales, the forecasting ability of a model might be more important; in other areas, such as risk management, the ability to correctly assess prediction certainty is more crucial.

5 Discussion

This section discusses the main findings from the results, focusing on model performance in terms of forecasting accuracy, handling seasonal and unpredictable patterns, and producing reliable confidence intervals. These insights help clarify the strengths and limitations of each model within the context of financial forecasting.

5.1 Forecasting Accuracy

Chronos, the foundation model, demonstrated impressive zero-shot performance, outperforming all deep learning and statistical baselines on nearly every metric and forecasting horizons. When tasked with predicting account balances for the next 30 days, Chronos-large achieved an overall MAPE of **18.35%**, indicating that for half the time series in the dataset, Chronos achieved an average error less than this. Chronos even outperformed the Naive model on a 1-day horizon, showing greater accuracy in 1-day predictions than a model that assumes the previous value will persist. While this may appear straightforward, no other model achieved this level of short-term accuracy.

Surprisingly, fine tuning the Chronos architecture on a subset of the data, did not improve its accuracy. This may be due to the limited 50 epochs (25000 steps) used for fine-tuning, which is potentially not enough time for the model to adjust its parameters to the specific patterns within this dataset.

Another unexpected result is the relatively strong performance of the Naive model, which outperformed two deep learning models specifically trained on this task. However, further investigation into the data revealed that a significant amount of time series in the dataset show, aside from a few large jumps, very little change over time. A generated example of this can be seen in Figure 18. In these cases, the Naive model provides highly accurate forecasts, lowering its average error across the dataset. This is further supported by Figure 15, where the Naive model performed worse than all deep learning and foundation models on sequences with multiple seasonal patterns.

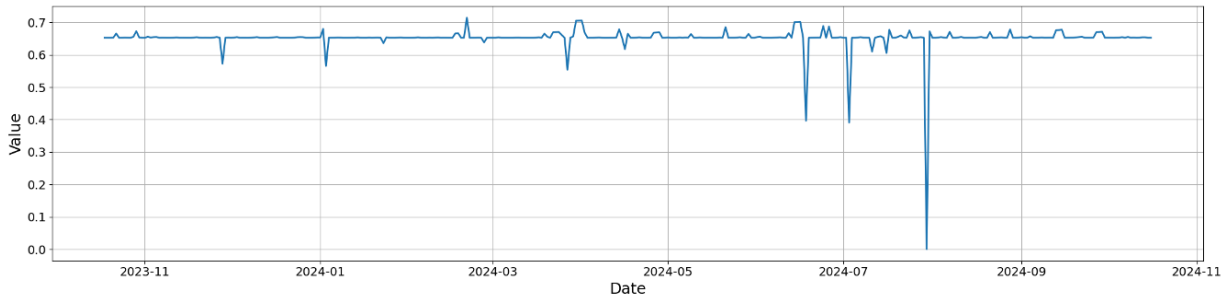


Figure 18: Visualization of a sequence representing a group of time series in the dataset that, aside from sporadic large jumps, remain largely inactive.

5.2 Seasonality and Entropy

In terms of seasonality, each statistical model outperformed every deep learning model when applied to time series with no seasonal patterns. This would suggest that statistical models are able to better capture patterns without explicit seasonality. However, this result could also be influenced by the method used for automatic seasonality detection, which used the same statistical models to detect the correct seasonality, as it used to forecast, possibly creating bias.

When examining the relationship between the approximate entropy, the inherent unpredictability of a sequence, and prediction accuracy, a slight trend seemed visible where series with a higher entropy seemed to have larger prediction errors. Future research into this topic is necessary, to deduce whether certain models can learn to recognize seemingly unpredictable patterns better than others.

5.3 Reliability

A key finding is the calibration of confidence intervals across models. While most models produce well-calibrated confidence intervals, the Chronos models (both in zero-shot and fine-tuned settings) and NHITS consistently overestimated their predictions. This overconfidence, however, does not correlate with poor forecasting performance, since Chronos is the best performing forecaster over all horizons.

To better understand this misalignment, the widths of the confidence intervals are investigated, shown in Figure 19. In this figure, it can be seen that the Chronos models and the

NHITS model indeed produce confidence intervals that are a lot smaller than those of the other models. This supports the idea the forecasts are not inaccurate, the model simply overestimates its confidence.

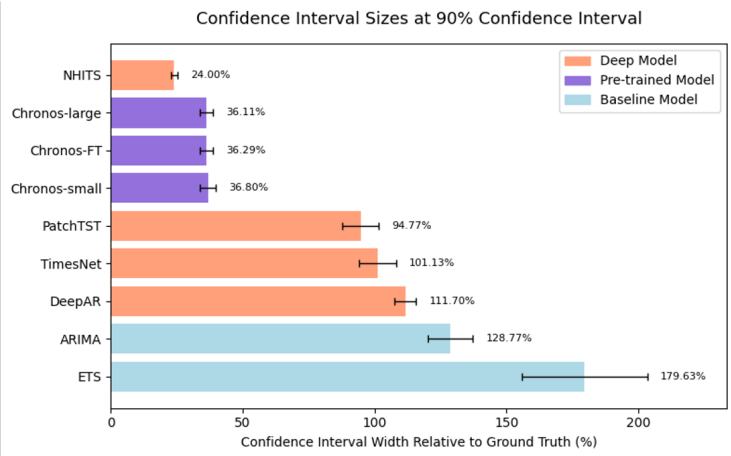


Figure 19: Plot showing the relative sizes of the 90% confidence interval width, compared to the ground truth value. For its 90% confidence intervals, Chronos typically assigns a width of 36% of the size of the real value, where ETS assigns a width of nearly 180%.

5.4 Future Research

This research demonstrates the diverse nature of financial transaction time series. Each time series comes with its own characteristics and each model with its own strengths and weaknesses. Rather than relying on a single model to perform optimally across all series, future research could explore using historical data to find the best suited model for a specific sequence. By looking at past performance, this approach could improve the model selection to best fit each individual time series.

Similarly, calibration of the confidence interval using past data could be explored. Using the historical confidence errors to adjust the interval widths for future forecasts may provide more reliable probabilistic forecasts.

Together, these approaches could make forecasting models both more adaptable and reliable.

6 Conclusion

This research has explored the potential for foundation models, specifically Chronos, to handle financial time series forecasting; comparing its performance against traditional statistical and deep learning baselines. Chronos demonstrated competitive accuracy, achieving zero-shot results that outperformed baseline models across multiple metrics and forecasting horizons. However, Chronos' probabilistic outputs showed inconsistencies, with confidence intervals often being overly confident. This invites further research into the calibration of probabilistic output in foundation models, to increase their ability of being suited for risk-sensitive applications.

A key takeaway is the relationship between seasonality patterns in the data and model performance. While both deep learning and foundation models tend to perform better on series with clear seasonal patterns, Chronos demonstrated an advantage over deep learning models when forecasting non-seasonal data. This suggests that Chronos can handle a more diverse set of time series, which shows great potential for its application in financial contexts, where traditional patterns are less predictable.

Looking ahead, future work could explore how a time series' inherent unpredictability, defined by its approximate entropy, affects model performance. A question that remains is whether Chronos performs consistently well across the full scope of predictable and unpredictable series, or if its strengths are limited to more regular patterns.

This thesis has explored time series forecasting in depth, focusing on both the theoretical background of many classic and state-of-the-art forecasting models and comparing their performance to a new class of foundation models. The results show encouraging accuracy, but future research is recommended before these models can be widely used. I hope that these insights add a small step forward in the research towards foundation models in forecasting.

References

[1] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2023. URL `https://arxiv.org/abs/2212.12794`.

[2] Renato Molina and Ivan Rudik. The social value of hurricane forecasts. Working Paper 32548, National Bureau of Economic Research, June 2024. URL `http://www.nber.org/papers/w32548`.

[3] Xue Yang, Xuejun Qi, and Xiaobo Zhou. Deep learning technologies for time series anomaly detection in healthcare: A review. *IEEE Access*, 11:117788–117799, 2023. doi: 10.1109/ACCESS.2023.3325896.

[4] Bader Aldughayfiq, Farzeen Ashfaq, N. Z. Jhanjhi, and Mamoona Humayun. A deep learning approach for atrial fibrillation classification using multi-feature time series data from ecg and ppg. *Diagnostics*, 13(14), 2023. ISSN 2075-4418. doi: 10.3390/diagnostics13142442. URL `https://www.mdpi.com/2075-4418/13/14/2442`.

[5] Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007, 1982. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/1912773`.

[6] Clive W. J. Granger. *Some Properties of Time Series Data and Their Use in Econometric Model Specification*, page 119–128. Econometric Society Monographs. Cambridge University Press, 2001.

[7] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN 9781118674925. URL `https://books.google.nl/books?id=rNt5CgAAQBAJ`.

[8] Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004. ISSN 0169-

2070. doi: https://doi.org/10.1016/j.ijforecast.2003.09.015. URL https://www.sciencedirect.com/science/article/pii/S0169207003001134.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[10] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL https://www.sciencedirect.com/science/article/pii/S0169207021001874. Special Issue: M5 competition.

[11] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[14] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

[16] Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position: What can large language models tell us about time series analysis, 2024. URL https://arxiv.org/abs/2402.02713.

[17] Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K. Gupta, and Jingbo Shang. Large language models for time series: A survey, 2024. URL https://arxiv.org/abs/2402.01801.

[18] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, Shirui Pan, Vincent S. Tseng, Yu Zheng, Lei Chen, and Hui Xiong. Large models for time series and spatio-temporal data: A survey and outlook, 2023. URL https://arxiv.org/abs/2310.10196.

[19] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

[20] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1, 2024. URL https://arxiv.org/abs/2310.03589.

[21] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

[22] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024. URL https://arxiv.org/abs/2310.08278.

[23] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *CoRR*, abs/2106.13008, 2021. URL https://arxiv.org/abs/2106.13008.

[24] Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, October 2024. ISSN 1557-7341. doi: 10.1145/3691338. URL http://dx.doi.org/10.1145/3691338.

[25] George Udny Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

[26] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[27] Rob Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 3rd edition, 2021.

[28] Spyros Makridakis, Allan Andersen, Robert Carbone, Robert Fildes, Michele Hibon, Rudolf Lewandowski, Joseph Newton, Emanuel Parzen, and Robert Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2):111–153, 1982.

[29] Spyros Makridakis, Chris Chatfield, Michèle Hibon, Michael Lawrence, Terence Mills, Keith Ord, and LeRoy F. Simmons. The m2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 9(1):5–22, 1993. ISSN 0169-2070. doi: https://doi.org/10.1016/0169-2070(93)90044-N. URL https://www.sciencedirect.com/science/article/pii/016920709390044N.

[30] Spyros Makridakis and Michèle Hibon. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(00)00057-1. URL https://www.sciencedirect.com/science/article/pii/S0169207000000571. The M3- Competition.

[31] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[32] Haowen Deng, Youyou Zhou, Lin Wang, and Cheng Zhang. Ensemble learning for the early prediction of neonatal jaundice with genetic features. *BMC Medical Informatics and Decision Making*, 21, 12 2021. doi: 10.1186/s12911-021-01701-9.

[33] João Monge, Gonçalo Ribeiro, António Raimundo, Octavian Postolache, and Joel Santos. Ai-based smart sensing and ar for gait rehabilitation assessment. *Information*, 14:355, 06 2023. doi: 10.3390/info14070355.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL `https://arxiv.org/abs/2010.11929`.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[37] Press release, October 2024. URL `https://www.nobelprize.org/prizes/physics/2024/press-release/`.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[39] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[40] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. Ieee, 2018.

[41] Md. Arif Istiake Sunny, Mirza Mohd Shahriar Maswood, and Abdullah G. Alharbi. Deep learning-based stock price prediction using lstm and bi-directional lstm model.

In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 87–92, 2020. doi: 10.1109/NILES50944.2020.9257950.

[42] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020.106181. URL https://www.sciencedirect.com/science/article/pii/S1568494620301216.

[43] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

[44] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021. URL https://arxiv.org/abs/2012.07436.

[45] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2022. URL https://arxiv.org/abs/2106.13008.

[46] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, 2022. URL https://arxiv.org/abs/2201.12740.

[47] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023. URL https://arxiv.org/abs/2211.14730.

[48] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

[49] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for

time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 6989–6997, 2023.

[50] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

[51] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[52] Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. Statsforecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL `https://github.com/Nixtla/statsforecast`.

[53] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[54] RG Brown. Statistical forecasting for inventory control. *McGraw-Hill google schola*, 2:443–473, 1959.

[55] Spyros Makridakis, Evangelos Spiliotis, Ross Hollyman, Fotios Petropoulos, Norman Swanson, and Anil Gaba. The m6 forecasting competition: Bridging the gap between forecasting and investment decisions, 2023. URL `https://arxiv.org/abs/2310.13357`.

[56] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[57] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. URL `https://arxiv.org/abs/2102.12092`.

[58] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020. URL https://arxiv.org/abs/2006.11477.

[59] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

[61] Hélène Merk. Drawing of foundation models taxonomy, 2024.

[62] Anthropic. Introducing claude 3.5 sonnet — anthropic.com. https://www.anthropic.com/news/claude-3-5-sonnet, 2024. [Accessed 23-10-2024].

[63] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67, 2020.

[64] Hao Xue and Flora D. Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting, 2023. URL https://arxiv.org/abs/2210.08964.

[65] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

[66] Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models, 2023. URL https://arxiv.org/abs/2309.06236.

[67] Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. The effectiveness of discretization in forecasting: An empirical study on neural time series models, 2020. URL https://arxiv.org/abs/2005.10111.

[68] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937.

[69] Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. Test: Text prototype aligned embedding to activate llm's ability for time series, 2024. URL https://arxiv.org/abs/2308.08241.

[70] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[71] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

[72] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Benchmarking foundation models for time series, 2024. URL https://github.com/Nixtla/nixtla/tree/main/experiments/foundation-time-series-arena.

[73] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search, 2013. URL https://arxiv.org/abs/1302.4922.

[74] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive, 2021. URL https://arxiv.org/abs/2105.06643.

[75] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Kościsz, Dennis Bader, Frédérick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gaël Grosch. Darts: User-friendly modern machine learning for time series, 2022. URL https://arxiv.org/abs/2110.03224.

[76] Kasun Bandara, Rob J Hyndman, and Christoph Bergmeir. Mstl: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns, 2021. URL `https://arxiv.org/abs/2107.13462`.

[77] Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022. URL `https://github.com/Nixtla/neuralforecast`.

[78] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[79] Steven M Pincus. Approximate entropy as a measure of system complexity. *Proceedings of the national academy of sciences*, 88(6):2297–2301, 1991.