



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING
INGEGNERIA INFORMATICA

A comparative study of Deep Learning and Streaming Machine Learning approaches for stock market forecasting

Author:

Tommaso Bucaioni

Student ID:

994950

Advisor:

Prof. Emanuele Della Valle

Coadvisor:

Lorenzo Iovine

Academic Year:

2024-2025

Dedicated to my family.

Abstract

In recent years, Streaming Machine Learning (SML) has gained increasing attention due to its ability to process data incrementally and adapt models in real time. However, its application to financial time series forecasting remains less explored compared to traditional batch-based Deep Learning approaches. Financial markets are inherently non-stationary and characterized by volatility and distributional shifts, which can significantly affect predictive performance. This thesis presents a comparative study between a state-of-the-art Deep Learning model, the Time-series Dense Encoder (TiDE), and adaptive Streaming Machine Learning methods for short-term stock market prediction. The objective is to evaluate whether streaming approaches can provide competitive predictive accuracy while offering advantages in adaptability and computational efficiency. Experiments are conducted on the S&P 500 index and the Brazilian ETF EWZ, considering both daily and hourly data. The results show that Deep Learning achieves slightly higher aggregate accuracy. Nevertheless, the streaming models consistently perform above the random baseline and demonstrate structural advantages, including real-time prediction capability and incremental adaptation to evolving data without full retraining. These findings highlight Streaming Machine Learning as a viable and efficient alternative for financial forecasting in dynamic environments.

Keywords: Streaming Machine Learning, Deep Learning, Financial Time Series Forecasting, Concept Drift.

Abstract in lingua italiana

Negli ultimi anni, lo Streaming Machine Learning (SML) ha attirato crescente attenzione grazie alla sua capacità di elaborare dati in modo incrementale e di adattare i modelli in tempo reale. Tuttavia, la sua applicazione alla previsione di serie temporali finanziarie rimane meno esplorata rispetto ai tradizionali approcci di Deep Learning basati su addestramento batch. I mercati finanziari sono intrinsecamente non stazionari e caratterizzati da volatilità e variazioni nella distribuzione dei dati, che possono influenzare significativamente le prestazioni predittive. Questa tesi presenta uno studio comparativo tra un modello di Deep Learning all'avanguardia, il Time-series Dense Encoder (TiDE), e metodi adattivi di Streaming Machine Learning per la previsione a breve termine dei prezzi azionari. L'obiettivo è valutare se gli approcci di tipo streaming possano fornire un'accuratezza predittiva competitiva offrendo al contempo vantaggi in termini di adattabilità ed efficienza computazionale. Gli esperimenti sono condotti sull'indice S&P 500 e sull'ETF brasiliano EWZ, considerando sia dati giornalieri sia orari. I risultati mostrano che il Deep Learning raggiunge un'accuratezza aggregata leggermente superiore. Tuttavia, i modelli di streaming mantengono prestazioni costantemente superiori alla baseline casuale e dimostrano vantaggi strutturali, tra cui la capacità di effettuare previsioni in tempo reale e di adattarsi in modo incrementale ai dati in evoluzione senza richiedere un riaddestramento completo. Questi risultati evidenziano lo Streaming Machine Learning come un'alternativa valida ed efficiente per la previsione finanziaria in ambienti dinamici.

Parole chiave: Streaming Machine Learning, Deep Learning, Financial Time Series Forecasting, Concept Drift

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Background and motivation	1
1.2 Contributions	2
1.3 Outline of the thesis	2
2 State Of The Art	5
2.1 Deep Learning	6
2.1.1 Neural Network Architectures	7
2.1.2 Sequence Modeling with Deep Learning	9
2.1.3 TiDE Model	10
2.2 Streaming Machine Learning	11
2.2.1 Concept Drift	12
2.2.2 Adaptive Models in Streaming Machine Learning	12
2.3 Preprocessing for Financial Time Series	14
2.3.1 Preprocessing for Deep Learning Models	14
2.3.2 Preprocessing for Streaming Machine Learning	14
3 Problem Setting	17
3.1 Case Study	17
3.2 Requirement Analysis	18
4 Theoretical Analysis and Proposed Solution	21
4.1 Data Retrieval	21
4.2 Deep Learning Implementation	22

4.2.1	Data Preprocessing and Denoising	22
4.2.2	Deep Learning Model Implementation	24
4.2.3	Regression-to-Classification Conversion	26
4.2.4	Preprocessing and Transformation	29
4.2.5	Streaming Machine Learning Model Implementation	30
5	Experimental Results and Discussion	31
5.1	Proposed Environment	32
5.2	Experimental Results	33
5.2.1	Baseline Results from the Reference Study	33
5.2.2	Reproduced Deep Learning Results (TiDE)	34
5.2.3	Streaming Machine Learning results	36
5.3	Trade-off Analysis Between Deep Learning and Streaming Approaches . . .	38
5.3.1	Evaluation Protocol and Predictive Scope	39
5.3.2	Adaptability to New Data	39
5.3.3	Computational Efficiency and Time Complexity	39
5.3.4	Overall Trade-off Perspective	40
6	Conclusion and Future Work	41
6.1	Conclusion	41
6.2	Future Work	42
	Bibliography	43
	List of Figures	47
	List of Tables	49
	Acknowledgements	51

1 | Introduction

Forecasting financial markets has long been a central challenge for both researchers and practitioners. According to Eugene Fama's Efficient Market Hypothesis (EMH), stock prices fully reflect all available information, implying that consistent outperformance through prediction is nearly impossible [9]. Nevertheless, researchers have long attempted to model and anticipate price movements using a wide range of quantitative and statistical approaches, such as autoregressive models, moving averages, and regression-based techniques. While these models have provided valuable insights into market trends and cyclical behaviors, they often struggle to capture the complex, non-linear dependencies and abrupt regime shifts that characterize real-world financial systems.

1.1. Background and motivation

The rapid expansion of computational power, coupled with advances in machine learning, has opened new opportunities for modeling financial time series beyond traditional frameworks. Deep learning architectures such as recurrent neural networks, temporal convolutional networks, and transformers can learn intricate temporal patterns and dependencies directly from raw data, offering potential improvements in predictive performance. However, financial markets are inherently dynamic, with conditions evolving continuously and patterns that quickly lose relevance.

To address this, streaming and adaptive machine learning methods have gained attention as they allow models to learn incrementally from incoming data and adjust predictions in real time. These approaches are especially valuable in short-term forecasting scenarios, where even minimal latency can undermine practical applicability. The goal of this thesis is to answer the following question: Do Streaming Machine Learning algorithms outperform Deep Learning models in short-term stock price prediction in terms of predictive accuracy, adaptability to non-stationary market conditions, and computational efficiency?

By integrating insights from both deep learning and adaptive online learning paradigms,

this work aims to contribute to the ongoing effort of developing resilient, responsive, and interpretable models for financial forecasting in an ever-changing environment.

1.2. Contributions

The contributions of this thesis can be summarized as follows:

- Reproduction and validation of a state-of-the-art deep learning model for financial time series forecasting, based on the methodology and experimental setting described in the reference work. The reproduced results provide a reliable baseline for subsequent comparative analysis.
- Design and implementation of a unified experimental framework that enables a fair benchmarking between batch deep learning models and streaming machine learning approaches on financial stock data, including consistent preprocessing, evaluation metrics, and experimental conditions.
- Comprehensive empirical comparison between deep learning and streaming machine learning models for short-term stock prediction, including performance evaluation, analysis of adaptability to evolving data distributions, and a critical discussion of the trade-offs between the two paradigms, highlighting their respective strengths and limitations in practical financial forecasting scenarios.

1.3. Outline of the thesis

This thesis is organized into five main chapters, each addressing a distinct aspect of the research and providing a coherent progression from theoretical foundations to practical implementation and conclusions.

- **Chapter 2** includes the State of the Art and presents an overview of the existing literature, methods, and technologies relevant to the research problem. It introduces the main concepts of time series forecasting, deep learning models, and streaming machine learning approaches, highlighting their strengths, limitations, and typical applications.
- **Chapter 3** defines and contextualized the research problem. The motivation for the study is presented, along with the challenges involved in applying different machine learning paradigms to financial time series. The chapter also outlines the evaluation criteria and requirements used to compare the models.

- **Chapter 4** describes the methodology and strategies employed to address the research problem. It details the experimental design, model selection, data preprocessing, and the framework used for benchmarking deep learning and streaming machine learning models. Key difficulties encountered during the study and how they were mitigated are also discussed.
- **Chapter 5** focuses on the practical implementation of the proposed solutions. It covers the setup of the computational environment, deployment of models, and execution of experiments. Observations and intermediate results obtained during implementation are also presented.
- **Chapter 6** summarizes the findings of the research, highlighting the main contributions and insights gained. Limitations of the study are discussed, and potential directions for future work are suggested, including possible improvements to models and new avenues of research in time series forecasting.

2 | State Of The Art

This chapter presents a review of the main methodologies and technologies that are relevant to the work carried out in this thesis. The focus is placed on two key approaches for analyzing time series data: Deep Learning models, which can automatically extract complex hierarchical patterns from historical datasets, and Streaming Machine Learning models, which are designed to process data incrementally and adapt to continuously changing environments.

The chapter begins with an overview of Deep Learning concepts in Section 2.1, describing fundamental neural network architectures, techniques for sequence modeling, and specialized models such as TiDE. It then moves on to Streaming Machine Learning in Section 2.2, examining the challenges posed by concept drift, approaches for adaptive learning, and strategies for handling non-stationary data streams efficiently.

Finally, Section 2.3 addresses preprocessing techniques for financial time series, highlighting the differences between batch-oriented transformations suitable for Deep Learning and online-compatible methods tailored for streaming models. Overall, this review establishes the theoretical and practical foundation that supports the experiments and comparisons conducted later in the thesis.

The chapter begins with an overview of Deep Learning concepts in Section 2.1, describing fundamental neural network architectures, techniques for sequence modeling, and specialized models such as TiDE. It then moves on to Streaming Machine Learning in Section 2.2, examining the challenges posed by concept drift, approaches for adaptive learning, and strategies for handling non-stationary data streams efficiently.

Finally, Section 2.3 addresses preprocessing techniques for financial time series, highlighting the differences between batch-oriented transformations suitable for Deep Learning and online-compatible methods tailored for streaming models. Overall, this review establishes the theoretical and practical foundation that supports the experiments and comparisons conducted later in the thesis.

2.1. Deep Learning

Deep Learning (DL) has emerged as a cornerstone of modern artificial intelligence, fundamentally transforming the way complex problems are approached in domains such as computer vision, natural language processing, speech recognition, and predictive analytics. At its core, Deep Learning relies on artificial neural networks—computational models inspired by the structure and functioning of the human brain—which are capable of learning hierarchical representations of data. This capability allows DL models to automatically discover intricate patterns and relationships, often surpassing traditional machine learning methods that depend heavily on manual feature engineering.

The primary characteristic that distinguishes Deep Learning from classical machine learning approaches is the use of multiple layers of interconnected neurons, commonly referred to as *deep architectures*. Each layer of the network extracts progressively more abstract features from the input data: lower layers may capture simple patterns, while higher layers encode more complex, high-level representations. This hierarchical feature learning enables Deep Learning models to effectively process large-scale and high-dimensional data without the need for explicitly defined rules or heuristics.

The training of deep neural networks is based on data-driven optimization techniques, where the model iteratively adjusts its internal parameters to minimize a predefined loss function. Gradient-based algorithms, such as stochastic gradient descent and its variants, propagate error signals backward through the network using the backpropagation algorithm. The combination of deep architectures, large datasets, and specialized hardware accelerators such as GPUs or TPUs has made it possible to train models with millions or even billions of parameters, achieving high levels of accuracy and strong generalization capabilities.

Another defining feature of Deep Learning is its flexibility and adaptability. Different network architectures can be designed to accommodate specific data modalities and tasks, allowing the same foundational principles to be applied to images, text, audio signals, and time series data. Moreover, the field is continuously evolving, with new models and training strategies being proposed to improve performance, efficiency, and interpretability.

Overall, Deep Learning represents a paradigm shift in artificial intelligence, characterized by its ability to learn complex representations directly from raw data, its hierarchical structure, and its adaptability across a wide range of application domains. Building on these foundational concepts, the following sections will examine the structure of neural networks in more detail, explore architectures designed for sequential data, and discuss

specialized models that illustrate the ongoing evolution of Deep Learning techniques.

2.1.1. Neural Network Architectures

Artificial Neural Networks (ANNs) form the foundational building blocks of Deep Learning, serving as computational models inspired by the structure and function of the human brain. Their strength lies in their ability to approximate complex nonlinear mappings between input and output data through the composition of multiple layers of simple processing units called neurons. This layered design enables neural networks to learn hierarchical feature representations, making them powerful tools for tasks such as classification, regression, and prediction [21].

Fundamental Structure of Neural Networks

A typical feedforward neural network is organized into three primary types of layers: an input layer, one or more hidden layers, and an output layer. Each layer is composed of neurons that perform a weighted sum of their inputs followed by the application of a nonlinear activation function. Mathematically, for a neuron j in layer l , the output (also called activation) is defined as:

$$a_j^{(l)} = \sigma \left(\sum_i w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)} \right)$$

where $a_i^{(l-1)}$ represents the activations from the previous layer, $w_{ij}^{(l)}$ are the weights connecting neuron i to neuron j , $b_j^{(l)}$ is the bias term, and $\sigma(\cdot)$ denotes the activation function (such as ReLU, sigmoid, or tanh). The input layer receives raw data—e.g., pixel intensities for an image, or time steps for a sequence—while the hidden layers successively transform this input into higher-level representations. Finally, the output layer produces the network's prediction, such as a class label or a continuous value. The key strength of this structure lies in its ability to learn the parameters $W = \{w_{ij}\}$ and $B = \{b_j\}$ directly from data through optimization [29].

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}$$

where η is the learning rate controlling the step size of the update. To compute the gradients efficiently, the algorithm known as backpropagation—first popularized by Rumelhart, Hinton, and Williams in the 1980s—applies the chain rule of calculus to propagate the loss derivatives from the output layer back through the network. This process systemati-

cally adjusts each weight to minimize the overall error [24]. Schmidhuber’s comprehensive review highlights that the success of deep learning architectures stems from this iterative refinement mechanism combined with large-scale data and high-performance computing [25].

Activation Functions and Nonlinearity

Nonlinearity is a crucial element that enables neural networks to model complex functions. Without nonlinear activation functions, a deep network would reduce to a single linear transformation, regardless of its depth. In practice, modern architectures typically employ activation functions such as the Rectified Linear Unit (ReLU) and its variants, which improve convergence behavior and help mitigate issues such as vanishing gradients [20]. In this thesis, activation functions are used as defined within the respective model implementations, without additional modification.

Depth, Capacity, and Generalization

The term deep in Deep Learning refers to the number of hidden layers within a neural network. Increasing network depth allows the model to capture more abstract and hierarchical features, thereby improving its expressive power. The Universal Approximation Theorem states that a feedforward network with a single hidden layer containing a sufficient number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$. However, deeper networks often achieve this approximation with far fewer parameters, providing better generalization and representational efficiency [18]. Nevertheless, deeper networks also introduce challenges, such as overfitting and optimization difficulties. To address these, various regularization techniques have been developed, including dropout, batch normalization, and weight decay, which help stabilize training and improve performance [27]. Recent research has also investigated the complexity of deep networks and their generalization behavior, showing that depth alone does not guarantee superior performance but must be balanced with architecture design and training strategy [19].

Modern Architectural Trends

While the foundational principles of neural networks remain consistent, modern architectures have evolved considerably to address efficiency, scalability, and long-range dependency modeling. Early models relied primarily on fully connected (dense) layers, where every neuron in one layer connects to every neuron in the next. However, this approach is computationally expensive and prone to redundancy. Contemporary designs introduce structural innovations that enhance performance:

- **Convolutional Neural Networks (CNNs):** exploit local connectivity and weight sharing to process spatial data efficiently, revolutionizing image recognition tasks [21, 29].
- **Residual Networks (ResNets) and Inception architectures:** incorporate skip connections and multi-scale processing paths, mitigating degradation problems in very deep networks [20].
- **Recurrent Neural Networks (RNNs), LSTM, and GRU:** extend neural connectivity across time, allowing information to persist across sequential inputs, which is essential for time-series and natural language data [16].
- **Attention-based architectures:** the Transformer model introduced by Vaswani et al. [30] relies entirely on self-attention mechanisms, completely removing recurrence. This allows the network to weigh the importance of different positions in the input sequence when generating each output, improving parallelization and achieving state-of-the-art performance in NLP and sequence modeling tasks.

Formally, the self-attention mechanism in Transformers is computed as:

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

where Q , K , and V are the query, key, and value matrices derived from the input, and d_k is the dimensionality of the key vectors. These trends highlight the continuous evolution of neural network architectures toward greater efficiency, interpretability, and applicability across diverse domains, from computer vision to time-series forecasting and natural language processing.

2.1.2. Sequence Modeling with Deep Learning

Many real-world problems involve data that are inherently sequential, where observations are ordered in time or space and strong dependencies exist between successive elements. Examples include natural language, speech signals, sensor measurements, and financial time series. In such contexts, modeling the temporal structure of the data is crucial, as the meaning or relevance of each observation often depends on previous inputs. Deep Learning provides a set of specialized architectures designed to explicitly capture these dependencies, enabling effective learning from sequential data.

Overview of Sequential Data Models

Sequence modeling is a core problem in machine learning, concerned with predicting or generating data where the order of elements matters, such as text, audio, or time series [1]. Unlike traditional deep learning models that assume independent and identically distributed (i.i.d.) inputs, sequence models maintain an internal state that summarizes past information, allowing predictions to depend on previous elements in the sequence. At each time step t , a sequence model updates its hidden state h_t based on the current input x_t and the previous state h_{t-1} :

$$h_t = f(h_{t-1}, x_t),$$

where f is a learnable non-linear function. This recurrent mechanism enables the network to capture temporal dependencies of varying length, a capability that standard feedforward networks lack. Advanced architectures improve the retention and selective propagation of information through gating mechanisms or memory cells. This allows the model to focus on relevant portions of the sequence while mitigating the influence of less informative data [16]. Sequence-to-sequence models extend this approach by encoding an input sequence into a context representation and generating an output sequence step by step, supporting applications such as machine translation and time series forecasting [28]. Recent models based on structured state spaces, such as S4, combine the advantages of recurrent networks and attention mechanisms to efficiently handle long sequences while capturing long-range dependencies [26]. Overall, deep learning for sequences generalizes standard feedforward architectures by integrating memory and temporal dynamics, enabling models to learn rich patterns that depend on both local and global context. This makes sequence modeling essential in domains where the temporal order of data critically affects the task outcome.

2.1.3. TiDE Model

The Time-series Dense Encoder (TiDE) model is a deep learning architecture specifically designed for long-term forecasting of sequential data [5, 6]. Traditional deep learning models for sequences, such as recurrent networks or Transformers, face limitations when handling very long input sequences due to either the vanishing gradient problem or quadratic scaling in computation and memory. TiDE addresses these challenges by leveraging a fully feedforward architecture that encodes the temporal context of a sequence into a dense representation, enabling accurate predictions across extended horizons without relying on recurrence or attention mechanisms. Conceptually, TiDE operates by trans-

forming the observed historical values of a time series, along with optional covariates, into a fixed-size context vector using a stack of Multi-Layer Perceptrons (MLPs). This dense representation effectively summarizes the relevant temporal information, serving as a memory of past events. Formally, given an input sequence $x_{1:T}$ and covariates $c_{1:T}$, the encoder produces a context vector z through learnable transformations:

$$z = \text{MLP}(x_{1:T}, c_{1:T}),$$

where $\text{MLP}(\cdot)$ denotes a composition of fully connected layers with non-linear activations. The decoder then maps z to future predictions $\hat{y}_{T+1:T+H}$, enabling long-term forecasting over the horizon H . A key advantage of TiDE is its linear computational complexity with respect to both the input sequence length and forecast horizon. By avoiding recurrence and attention, the model can efficiently process very long sequences while mitigating common issues such as vanishing gradients or high memory consumption. The architecture’s dense encoding captures complex non-linear dependencies between past observations and covariates, allowing it to outperform many traditional sequence models in benchmarks involving long-term forecasting [5, 6]. Empirical studies demonstrate that TiDE achieves superior accuracy on a variety of datasets while maintaining scalability. Its simplicity, combined with the ability to encode temporal dependencies effectively, makes it a compelling alternative to recurrent and attention-based sequence models. In practical terms, TiDE allows researchers and practitioners to perform accurate long-term forecasting without the computational overhead associated with more complex sequential architectures, making it suitable for a wide range of real-world applications, including financial time series prediction, energy demand forecasting, and other domains where modeling long-range dependencies is critical. In summary, TiDE represents an innovative approach to sequence modeling, combining the efficiency of feedforward networks with the capacity to capture temporal patterns. By encoding historical context into a dense representation and leveraging a linear, scalable architecture, TiDE overcomes many limitations of traditional deep learning models for sequences, providing both high predictive performance and computational efficiency [5, 6].

2.2. Streaming Machine Learning

Data stream prediction is an emerging area of machine learning focused on continuously processing and learning from data that arrive in a sequential and potentially unbounded fashion. Unlike traditional batch learning, where the full dataset is available before training, data stream learning assumes that examples appear only once and must be processed

under strict time and memory constraints. As Gama notes in his seminal work *Knowledge Discovery from Data Streams* [10], such environments are inherently dynamic, requiring models to update incrementally and adapt to evolving patterns over time. The motivation for streaming learning arises from real-world applications such as financial transaction monitoring, intrusion detection, sensor networks, and online recommendation systems. In these domains, data evolve continuously and the statistical properties of the target concept may shift—a phenomenon known as *concept drift*. The ability to adapt efficiently to these changes while preserving past knowledge is what distinguishes streaming machine learning from conventional static approaches.

2.2.1. Concept Drift

In streaming supervised learning, *concept drift* denotes a change over time in the predictive concept, i.e., in the conditional distribution $P(Y | X)$ [8]. In stationary environments this relationship is assumed constant, whereas in real-world data streams it may evolve due to seasonal effects, structural market changes, or shifts in underlying economic conditions.

Elwell and Polikar [8] distinguish between two types of drift. *Real drift* occurs when the conditional distribution $P(Y | X)$ changes, directly altering the predictive relationship. In contrast, *virtual drift* arises when only the marginal distribution $P(X)$ shifts, a phenomenon commonly referred to as data drift or covariate shift, which does not necessarily modify the underlying concept.

Concept drift poses a significant challenge because models trained on outdated distributions can rapidly lose predictive performance. The problem becomes more complex in the presence of class imbalance, where rare events may be masked by majority classes, requiring adaptive learning strategies [17].

To address concept drift, the literature proposes two main categories of approaches: detection-based methods, which explicitly identify changes and trigger model updates, and continuous adaptive methods, which incrementally update the model as new data arrives. In this thesis, the focus is placed on the adaptive learning capabilities inherent to streaming algorithms, rather than on explicit drift detection mechanisms.[10–12]

2.2.2. Adaptive Models in Streaming Machine Learning

In streaming environments, where data arrive continuously and the underlying distribution may evolve, static models trained on a fixed dataset quickly become obsolete. To

maintain accuracy over time, models must be capable of adapting to changes in the data-generating process. These *adaptive models* constitute a cornerstone of Streaming Machine Learning, allowing systems to evolve incrementally while handling the challenges of non-stationarity and limited computational resources [4].

According to Bifet et al. [4], adaptive models address change in data streams through three main strategies: (1) adaptive estimators, (2) explicit change detection mechanisms, and (3) ensemble-based adaptation. Adaptive estimators dynamically update internal statistics such as probabilities or counts as new data arrive. These updates occur incrementally and continuously, without revisiting previous data, enabling models like Naive Bayes and the Perceptron to remain up to date with minimal computational overhead.

The second strategy involves explicit *change detection*, where algorithms monitor model error or data distribution and trigger adaptation when statistically significant deviations are observed. One of the most widely used examples is ADWIN (Adaptive Windowing) [2], which maintains a variable-length window of recent instances and detects changes when the distribution of the data inside the window shifts. When drift is detected, the model can either be retrained from recent data or locally adjusted, depending on the magnitude and nature of the change.

The third and most widely used category of adaptive models relies on *ensemble learning*. In this approach, multiple classifiers are maintained simultaneously, each specializing in a different part of the stream. Poorly performing models are discarded, and new ones are added as the data evolve. Techniques such as Adaptive Random Forests [15] and ADWIN Bagging [3] exemplify this paradigm, providing resilience against both abrupt and gradual concept drift. Ensembles allow fast adaptation without fully discarding previous knowledge, balancing stability and plasticity.

Among the most influential adaptive models for data streams is the *Hoeffding Adaptive Tree* (HAT), an evolution of the Hoeffding Tree introduced by Domingos and Hulten [7]. HAT integrates change detection (often via ADWIN) within its decision nodes, enabling the tree to replace subtrees when the local distribution changes. This adaptive mechanism allows the model to remain consistent with evolving concepts while maintaining computational efficiency. Such methods are implemented in large-scale stream mining frameworks like MOA (Massive Online Analysis) and SAMOA, forming the foundation of practical adaptive systems [4].

In summary, adaptive models enable continuous learning under nonstationary conditions by dynamically updating their parameters, detecting drift, and combining multiple learners. These strategies form the backbone of modern Streaming Machine Learning, en-

sureing that models remain both responsive and robust in environments where data evolve over time.

2.3. Preprocessing for Financial Time Series

Preprocessing is a fundamental component of machine learning pipelines, particularly in financial time series forecasting, where raw data are affected by noise, non-stationarity, and evolving statistical properties. The choice of preprocessing techniques is closely related to the learning paradigm adopted, as batch-oriented Deep Learning models and Streaming Machine Learning algorithms impose different constraints in terms of data availability, computational efficiency, and adaptivity.

2.3.1. Preprocessing for Deep Learning Models

In batch-oriented Deep Learning settings, preprocessing techniques can leverage access to historical data to construct richer and more structured representations of the input signal. Financial time series, such as stock prices, exhibit multi-scale temporal dynamics in which relevant patterns may occur simultaneously at different frequencies. Traditional preprocessing methods operating at a single temporal resolution are often insufficient to capture this complexity.

To address this limitation, this thesis adopts the Discrete Wavelet Transform (DWT) as a preprocessing step for Deep Learning models. DWT enables a time series to be decomposed into approximation and detail coefficients across multiple resolution levels, providing localized representations in both time and frequency domains. This multi-resolution analysis is particularly well suited for non-stationary signals, as it allows long-term trends and short-term fluctuations to be modeled separately [22, 23].

By transforming raw stock price series into wavelet-based components, the preprocessing step aims to reduce noise and enhance the signal-to-noise ratio while preserving meaningful temporal structures. This representation aligns naturally with the hierarchical feature extraction mechanisms of Deep Learning architectures, allowing models to focus on learning predictive patterns rather than compensating for raw data irregularities.

2.3.2. Preprocessing for Streaming Machine Learning

In contrast, Streaming Machine Learning operates under strict online constraints, where data arrive sequentially and preprocessing must be performed incrementally, without access to the full historical dataset. As a result, preprocessing techniques for data streams

prioritize computational efficiency, low memory usage, and adaptability to concept drift.

Common preprocessing strategies in streaming settings include incremental normalization, standardization, and transformation-based techniques designed to stabilize learning dynamics over time. Incremental standardization methods update summary statistics, such as mean and variance, on the fly, ensuring that input features remain appropriately scaled as the data distribution evolves [13].

Additionally, transformation-based approaches, such as add-noise transformations, have been proposed to improve robustness and generalization in non-stationary environments. By injecting controlled noise into the input stream, these methods act as a form of regularization, reducing sensitivity to short-term fluctuations and helping models adapt more smoothly to distributional changes [17]. Such techniques are particularly relevant in financial data streams, where abrupt changes and noisy observations are common.

Overall, preprocessing in Streaming Machine Learning emphasizes simplicity and adaptability over complex signal decomposition. This distinction reflects the fundamental differences between batch and online learning paradigms and motivates the use of different preprocessing strategies for Deep Learning and Streaming Machine Learning models within this thesis.

3 | Problem Setting

The primary objective of this thesis is to evaluate and compare the performance of Deep Learning and Streaming Machine Learning models for univariate stock price prediction. The starting point for this work is a reference paper that analyzes the results of various Deep Learning models for time series forecasting [14]. The first step is to reproduce the predictive performance reported in that paper using the same models and datasets. Building upon this foundation, Streaming Machine Learning models are then applied to the same datasets to assess whether incremental learning can provide improvements in predictive accuracy and adaptability.

The research question guiding this thesis is:

To what extent can Streaming Machine Learning models provide competitive or superior predictive performance compared to batch-trained Deep Learning models on financial time series, and under which conditions are the advantages of streaming approaches most pronounced?

To address this question, the following objectives are defined:

- Reproduce the results of the reference paper using batch-trained Deep Learning models.
- Evaluate the performance of Streaming Machine Learning models on the same datasets, focusing on online learning and concept drift handling.
- Compare the two paradigms with respect to forecast accuracy, resilience to non-stationarity, and responsiveness to abrupt market changes.

3.1. Case Study

The case study is based on publicly available historical stock price datasets, including daily prices, trading volumes, and relevant market indicators. The data exhibit several characteristics that are typical of financial time series:

- Daily frequency over multiple years, enabling short- and medium-term forecasting analysis.
- Non-stationary behavior, including trends, seasonal effects, and structural breaks.
- High volatility and noise, with occasional outliers that require robust modeling strategies.

To evaluate the proposed approaches, the experimental framework considers three main scenarios. First, a one-step-ahead forecasting setting is adopted, where models predict the next stock price using historical observations. Second, the incremental arrival of new data is simulated to assess the online learning capabilities of streaming models. Finally, robustness is evaluated under sudden market shifts, approximating concept drift conditions and testing model adaptability.

3.2. Requirement Analysis

Based on the objectives of the thesis and on the characteristics of the considered financial time series, the following requirements have been defined in order to ensure a fair and meaningful comparison between Deep Learning and Streaming Machine Learning approaches.

- R1. Preprocessing:** Data preprocessing must follow the methodology adopted in the reference paper. In particular, Discrete Wavelet Transform (DWT) must be applied as a batch preprocessing step, using the same configuration and implementation described in the original study.
- R2. Deep Learning Implementation:** The Deep Learning baseline must be implemented using the TiDE model, adopting the same architecture and hyperparameters specified in the reference paper, in order to faithfully reproduce the reported results.
- R3. Streaming Machine Learning Implementation:** A set of Streaming Machine Learning models must be implemented and evaluated on the same datasets, with the objective of comparing their predictive performance against the Deep Learning baseline. The streaming models must support incremental learning and adaptation to concept drift.
- R4. Evaluation Metrics:** All models must be evaluated using the same classification metrics, namely accuracy, precision, recall, and F1-score, in order to ensure a consistent and comparable performance assessment.

R5. Scenario Coverage: The experimental evaluation must be conducted on the same financial instruments and time range considered in the reference paper. In particular, the analysis must include the S&P 500 index and the Brazilian ETF EWZ, covering the period from July 14, 2000 to December 29, 2023. Furthermore, experiments must be performed at different temporal granularities, namely daily and hourly data, as specified in the original study.

4 | Theoretical Analysis and Proposed Solution

This chapter describes the theoretical framework and the methodological choices adopted to address the research problem defined in Chapter 3. In particular, it outlines how the comparison between Deep Learning and Streaming Machine Learning models for stock market trend prediction has been conducted. The chapter details the dataset preparation, preprocessing strategies, learning paradigms, and model configurations, highlighting the differences between batch-based and streaming-based approaches.

The proposed solution is implemented through two distinct and independent learning pipelines, each designed to reflect the computational and operational constraints of its respective paradigm. While both approaches operate on the same financial time series and share a common target definition, their data handling, preprocessing steps, training procedures, and update mechanisms differ substantially. For this reason, the Deep Learning and Streaming Machine Learning solutions are presented separately, with a focus on their specific implementation choices. This structure enables a fair and consistent comparison while preserving the intrinsic characteristics of each learning paradigm.

4.1. Data Retrieval

This section describes the data acquisition process adopted in this thesis. It represents the only implementation stage that is shared between the Deep Learning and Streaming Machine Learning pipelines, as both approaches rely on the same raw financial time series before diverging in their preprocessing strategies, learning procedures, and model update mechanisms. The datasets used in this thesis are constructed in accordance with the reference paper and include the S&P 500 index and the Brazilian ETF EWZ over the period from July 14, 2000 to December 29, 2023, considering both daily and hourly granularities as specified in the original study. For each financial instrument and temporal resolution, raw market data are collected using the standard OHLCV representation, where each time interval is summarized by its opening price, highest price, lowest price,

closing price, and trading volume. The data retrieval process is performed through the Yahoo Finance API, which provides historical market information in a structured and reliable format. The selected time range ensures the availability of long-term price dynamics, including multiple market regimes, volatility phases, and macroeconomic events. Both daily and hourly resolutions are considered in order to evaluate the robustness of the learning models under different temporal granularities. In order to align the feature set with the experimental design of the reference study and reduce the dimensionality of the input space, the original OHLCV representation is simplified by retaining only the *Close* price and the *Volume*. The remaining price-related attributes (Open, High, Low) as well as auxiliary fields are removed. This choice reflects the goal of focusing the learning process on the most informative signals while minimizing redundancy and noise in the input data.

4.2. Deep Learning Implementation

This section describes the implementation of the Deep Learning pipeline adopted in this work. The approach follows a batch-oriented workflow, where historical financial data are fully available prior to training and are processed through a sequential pipeline comprising data acquisition, cleaning, signal denoising, preprocessing, model training, and output transformation.

The implementation is designed to remain consistent with the experimental setup and data processing choices reported in the reference study, ensuring comparability of results while allowing the proposed extensions to be evaluated under controlled conditions. The following subsection details the data acquisition process, the cleaning steps applied to the raw market data, and the denoising procedure adopted prior to model training.

4.2.1. Data Preprocessing and Denoising

Prior to model training, the raw financial time series are processed through a dedicated preprocessing pipeline designed to improve data quality, reduce noise, and prepare the inputs for batch-based Deep Learning models. Financial market data are inherently affected by high-frequency fluctuations, short-term volatility, and measurement noise, which can obscure meaningful temporal patterns and negatively impact the learning process of neural networks. For this reason, a signal denoising stage is introduced before feature scaling and dataset splitting.

Noise reduction is performed using the Discrete Wavelet Transform (DWT), which en-

ables the decomposition of the original stock price signal into multiple frequency components. In this implementation, the Daubechies wavelet of order 4 (db4) is selected due to its good time–frequency localization properties and its suitability for representing non-stationary financial signals. The decomposition level is set to 4, allowing the signal to be separated into approximation coefficients, which capture the low-frequency trend, and detail coefficients, which represent higher-frequency variations typically associated with noise and short-term fluctuations.

To reduce noise while preserving the underlying structure of the signal, a soft thresholding procedure is applied to the corresponding detail coefficients. The threshold value is computed using the universal threshold method, where the noise standard deviation is estimated from the median absolute deviation of the finest-scale wavelet coefficients. This approach provides an adaptive and data-driven threshold that scales with the noise level of the signal. The approximation coefficients are left unchanged in order to retain the long-term price dynamics, while the detail coefficients are shrunk using soft thresholding, which reduces their magnitude without introducing abrupt discontinuities. The denoised signal is then reconstructed through the inverse wavelet transform, yielding a smoother time series that emphasizes trend information over high-frequency noise.

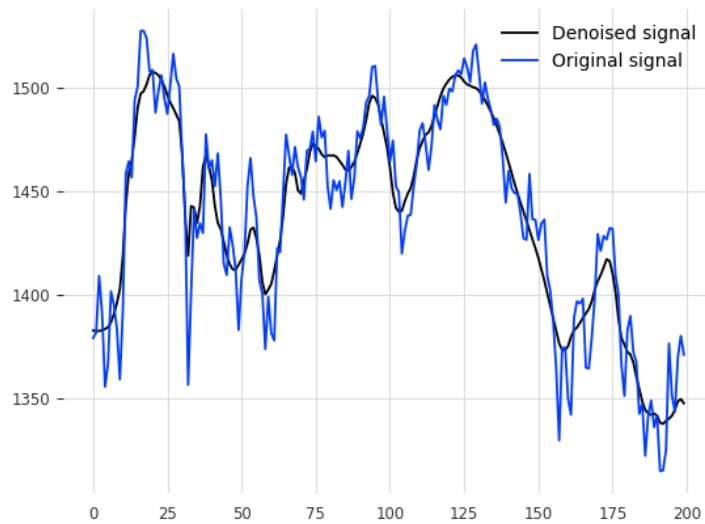


Figure 4.1: Comparison between a segment of the original GSPC (S&P 500) time series and its corresponding wavelet-denoised signal.

After denoising, the time series is normalized using a scaling transformation to ensure numerical stability during neural network training. Feature scaling is a crucial step for Deep Learning models, as it prevents large input values from dominating the optimization process and improves convergence behavior. The scaling operation maps the signal values to a standardized range, enabling consistent gradient updates and reducing the risk of

training instability.

The scaled and denoised time series is subsequently divided into training, validation, and test sets using a chronological split. For the daily dataset, 86% of the available data is assigned to the training set, while the remaining 14% is evenly split between validation and test sets. For the hourly dataset, a different partitioning strategy is adopted, with 75% of the data used for training and the remaining 25% equally divided between validation and test sets (12.5% each).

This temporal partitioning preserves the sequential nature of the financial data and ensures that the models are evaluated on future, unseen observations, reflecting a realistic forecasting scenario. The validation set is used to tune model hyperparameters and prevent overfitting, while the test set is reserved for the final performance assessment.

Table 4.1: Start and End Dates for Dataset Splits

Dataset	Split	Start Date	End Date	Percentage
Daily	Training	01/01/2000	31/12/2020	86%
	Validation	01/01/2021	30/06/2022	7%
	Test	01/07/2022	31/12/2023	7%
Hourly	Training	13/07/2020	30/06/2023	75%
	Validation	01/07/2023	31/12/2023	12.5%
	Test	01/01/2024	11/07/2024	12.5%

This preprocessing pipeline provides the Deep Learning models with cleaner, normalized, and temporally consistent input data, facilitating more stable training and improving the ability of the networks to capture meaningful patterns in stock market dynamics.

4.2.2. Deep Learning Model Implementation

The Deep Learning forecasting component is implemented using the *Time-series Dense Encoder* (TiDE) architecture provided by the **Darts** library. TiDE is a fully-connected encoder-decoder model specifically designed for long-term time series forecasting, which replaces the attention mechanism of Transformer-based models with dense layers in order to reduce computational complexity while preserving strong predictive performance.

In this work, the model is applied to a univariate financial time series forecasting task formulated as a regression problem. The objective is to predict the next value of the target series using a fixed-length window of past observations. The input sequence length is set to `input_chunk_length = 100`, meaning that the model receives the previous 100 time steps as input. The forecasting horizon is defined by `output_chunk_length = 1`, corresponding to a one-step-ahead prediction.

To improve robustness to distributional shifts and stabilize training, the model is configured with *Reversible Instance Normalization* (RIN) enabled. The corresponding option is set to (`use_reversible_instance_norm = True`). This normalization technique applies instance-wise normalization to the input series and reverses the transformation on the model output, allowing the network to focus on learning temporal patterns while preserving the original data scale.

Model training is performed using the PyTorch Lightning backend integrated into the Darts framework. Optimization is carried out with the Adam optimizer using a learning rate of 10^{-4} . To ensure a gradual and stable learning process, an exponential learning rate scheduler is applied, with a decay factor of $\gamma = 0.999$ at each epoch. The training procedure is configured to run for a maximum of 200 epochs, with a batch size of 256 samples.

To improve training stability, gradient clipping is applied with a maximum norm of 1. In addition, an early stopping strategy is employed to prevent overfitting and reduce unnecessary computation. The validation loss (`val_loss`) is monitored during training, and the process is stopped if no improvement greater than 10^{-3} is observed for 10 consecutive epochs. Model checkpoints are automatically saved, and the best-performing model state is restored at the end of training.

Table 4.2: TiDE Model Configuration and Training Parameters

Parameter	Value
Model architecture	TiDE (Time-series Dense Encoder)
Task type	Regression
Input window length	100
Forecast horizon	1
Batch size	256
Maximum epochs	200
Optimizer	Adam
Learning rate	1×10^{-4}
Learning rate scheduler	ExponentialLR ($\gamma = 0.999$)
Gradient clipping	1
Early stopping patience	10 epochs
Early stopping min_delta	1×10^{-3}
Checkpoint saving	Enabled
Reversible Instance Normalization (RIN)	Enabled
Random seed	42

The model is trained on the training split and validated on the validation split of the dataset. After training, predictions are generated on the test set using an autoregressive

rollout strategy. Since the forecasting horizon exceeds the output chunk length, the model iteratively uses its own previous predictions as input to generate a sequence of future values. In this work, a total of 100 future time steps are predicted for each test segment.

This implementation enables a consistent and controlled evaluation of the TiDE architecture for one-step-ahead financial time series forecasting, providing a strong Deep Learning baseline for comparison with Streaming Machine Learning approaches.

4.2.3. Regression-to-Classification Conversion

The TiDE model outputs continuous predictions of the stock closing price, thus formulating the original task as a regression problem. However, in order to perform directional forecasting, the objective is reformulated as a classification task aimed at predicting whether the stock price will move upward or downward relative to the previous trading day. This transformation is achieved through a post-processing step applied to the regression outputs.

Financial return series are typically characterized by a strong concentration of values around zero, with relatively infrequent large variations. Figure 4.2 illustrates the empirical distribution of daily price variations for the S&P 500 dataset prior to directional labeling. The distribution is sharply centered around zero, indicating that most observations correspond to small price changes, while larger movements form the distribution tails. This characteristic highlights the presence of noise and the potential for class imbalance when discretizing continuous returns, motivating a careful definition of the classification rule.

First, the predicted closing prices are converted into daily percentage returns, computed as the relative change between two consecutive predicted time steps. Given the predicted closing price $\hat{P}_t$ at time t and the predicted price $\hat{P}_{t+1}$ at time $t + 1$, the percentage return is defined as:

$$r_t = \frac{\hat{P}_{t+1} - \hat{P}_t}{\hat{P}_t} \times 100$$

This representation removes the dependency on absolute price levels and provides a scale-invariant measure of price movement, which is more suitable for directional analysis.

Subsequently, the continuous percentage returns are discretized into categorical classes using a threshold-based rule. If the return r_t is greater than or equal to a predefined threshold, the movement is labeled as *up*; if r_t is lower than the negative threshold, it is labeled as *down*. Optionally, returns within the threshold interval can be assigned to a *no-change* class to account for negligible fluctuations. In this work, a zero threshold is adopted, resulting in a binary classification setting where any positive return is classified

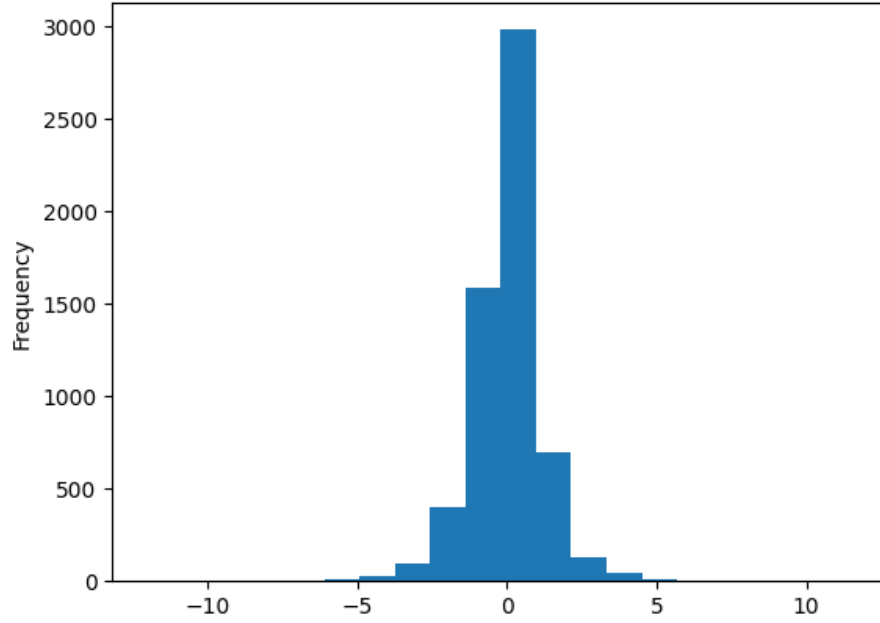


Figure 4.2: Distribution of daily price variations for the S&P 500 dataset prior to regression-to-classification conversion.

as *up* and any negative return as *down*.

Figure 4.3 shows the resulting class distribution obtained after applying the regression-to-classification conversion to the dataset. The distribution exhibits a moderate predominance of upward movements over downward movements, reflecting typical market behavior. Although a slight class imbalance is present, the dataset remains reasonably balanced, and no explicit resampling or class reweighting strategies are required.

The same regression-to-classification procedure is applied to the EWZ ETF dataset. Figure 4.4 illustrates the distribution of daily price variations prior to directional labeling, while Figure 4.5 reports the resulting class balance obtained after threshold-based discretization.

The adoption of a zero threshold simplifies the labeling strategy and avoids introducing additional hyperparameters, while remaining consistent with the empirical distribution of returns observed in the dataset. This regression-to-classification conversion enables the evaluation of the TiDE model from a directional perspective and allows a fair comparison with classification-based and streaming machine learning approaches, while still leveraging the information learned by the original regression model.

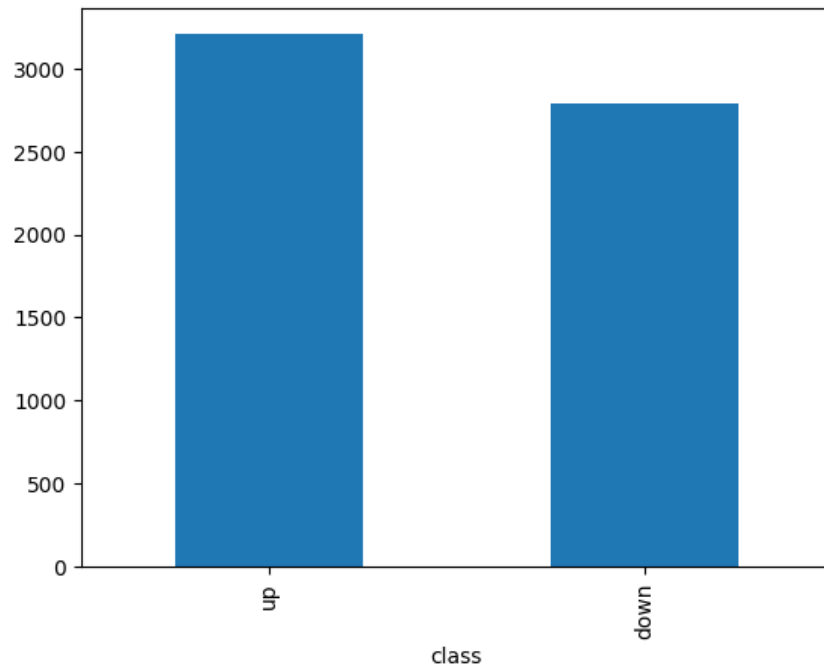


Figure 4.3: Class distribution of directional labels (*up* and *down*) obtained after regression-to-classification conversion for the S&P 500 dataset.

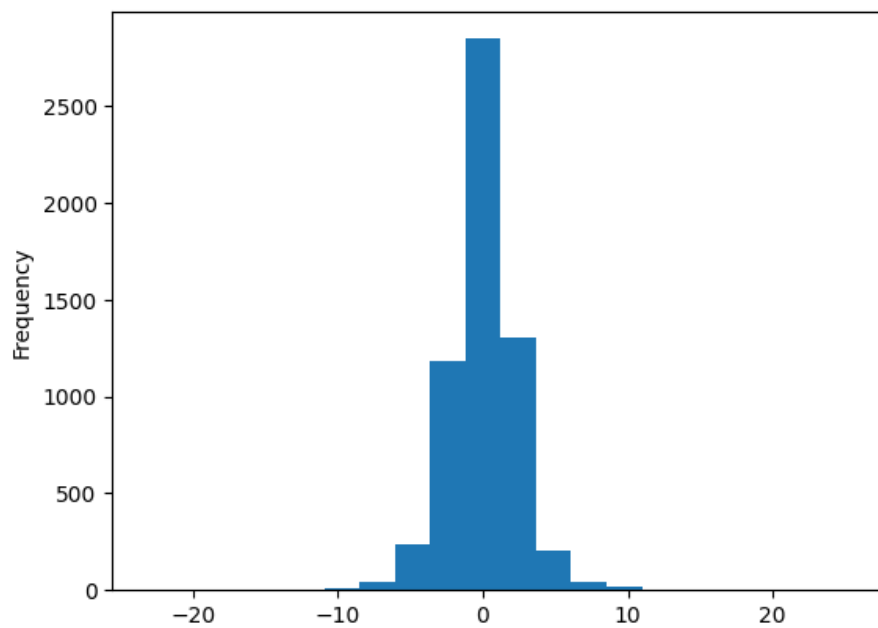


Figure 4.4: Distribution of daily price variations for the EWZ ETF dataset prior to regression-to-classification conversion.

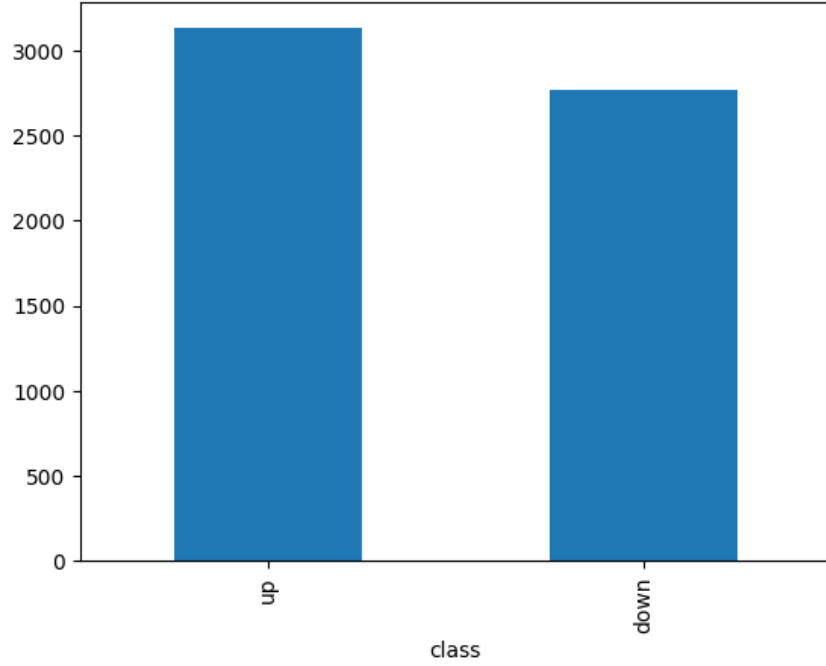


Figure 4.5: Class distribution of directional labels (*up* and *down*) obtained after regression-to-classification conversion for the EWZ ETF dataset.

4.2.4. Preprocessing and Transformation

The preprocessing stage transforms the raw stock price time series into a stream of instances suitable for online classification, with particular emphasis on normalization and denoising. Since streaming models do not benefit from learned feature representations, careful preprocessing is required to mitigate the effects of scale variability and market noise.

The target variable is derived by computing the daily percentage return from consecutive closing prices and discretizing it into directional classes. A binary labeling scheme is adopted, where positive returns are labeled as *up* and negative returns as *down*. This formulation aligns the streaming task with directional prediction objectives.

To model short-term temporal dependencies, lag-based features are introduced by including the closing prices of the previous five trading days. Additionally, the class labels observed in the previous five time steps are incorporated as categorical features, enabling the model to capture recent momentum and trend persistence.

Before classification, each incoming instance undergoes online normalization using an incremental standardization filter. This step ensures that all numerical features are maintained on a comparable scale throughout the stream, preventing dominance by high-

magnitude attributes and improving numerical stability. Following normalization, a denoising transformation is applied by injecting small random noise into the feature values. This procedure reduces sensitivity to minor fluctuations and high-frequency noise, which are common in financial time series, and promotes better generalization in the absence of deep hierarchical representations.

All transformations are applied sequentially and incrementally, preserving the temporal ordering of the data and avoiding any form of data leakage from future observations.

4.2.5. Streaming Machine Learning Model Implementation

The streaming classification task is addressed using a Hoeffding Adaptive Tree (HAT), a decision tree model specifically designed for incremental learning on evolving data streams. The classifier processes one instance at a time and updates its structure without retaining historical data, making it suitable for real-time financial applications. The learning process is governed by the Hoeffding bound, which provides a statistically sound criterion for deciding when a node should be split based on a finite number of observations. Candidate split attributes are evaluated incrementally, and a split is performed only when sufficient evidence indicates a clear advantage of one attribute over the others. The adaptive variant of the Hoeffding Tree is particularly relevant in the financial domain, as it incorporates mechanisms to detect and react to concept drift. By maintaining alternate subtrees and monitoring their predictive performance, the model can replace outdated branches when changes in the data distribution are detected, enabling continuous adaptation to evolving market conditions. In this implementation, the classifier is embedded within an online pipeline that explicitly separates preprocessing and learning. Each incoming instance is first normalized and denoised before being passed to the classifier for prediction. Once the true label becomes available, the same instance is used to update the model parameters, following a prequential (test-then-train) evaluation protocol. This explicit preprocessing pipeline compensates for the absence of deep representation learning and is a key distinguishing factor with respect to the deep learning approach. Model performance is evaluated using an online classification evaluator that updates accuracy incrementally after each prediction, ensuring a realistic assessment of the model under streaming constraints.

5 | Experimental Results and Discussion

This chapter presents the implementation and empirical evaluation of the methodologies introduced in Chapter 4. While the previous chapter detailed the experimental design, model configurations, and learning strategies adopted for both deep learning and streaming machine learning approaches, the focus here shifts to their execution and observed behavior in a real-world financial setting.

The primary objective is to assess how the proposed models perform when applied to real financial time series data. To ensure transparency and reproducibility, the computational environment and software stack used for the experiments are first described, with particular attention to hardware limitations and resource constraints. The experiments were intentionally conducted on consumer-grade hardware in order to evaluate the feasibility of the proposed approaches under realistic conditions, without relying on specialized high-performance computing infrastructure.

The chapter then reports the empirical results obtained from both deep learning and streaming machine learning models. Quantitative performance metrics are presented through tables and graphical visualizations to provide a structured and objective comparison between the two paradigms. Beyond final performance scores, intermediate observations gathered during execution are analyzed, including aspects such as training stability, sensitivity to noise, and responsiveness to evolving data distributions.

Finally, a comprehensive comparative discussion examines the trade-offs between deep learning and streaming machine learning. This analysis extends beyond raw predictive accuracy and considers computational efficiency, memory consumption, adaptability to non-stationary environments, and reliance on preprocessing and denoising techniques. The purpose is to provide a holistic evaluation of the two paradigms, thereby directly addressing the research question and laying the groundwork for the conclusions drawn in the final chapter.

5.1. Proposed Environment

All experiments were conducted on a *MacBook Pro* equipped with an *Apple M2* CPU and *8 GB* of unified memory. The experimental setup intentionally relies on a consumer-grade machine to demonstrate the feasibility of both deep learning and streaming machine learning approaches without requiring specialized hardware or high-performance computing resources.

The entire implementation was developed in **Python**, which was selected for its extensive ecosystem of libraries supporting time series analysis, machine learning, and deep learning. To ensure reproducibility and dependency isolation, the execution environment was containerized using **Docker**. A dedicated base image (`tachyonic/jupyter-capymoa`) was adopted to provide native support for streaming machine learning with **CapyMOA**. Additional project dependencies were installed via a `requirements.txt` file, and **PyTorch** was explicitly installed in its CPU-only configuration to avoid GPU-specific dependencies and ensure consistency across executions.

The software stack integrates libraries addressing data acquisition, preprocessing, modeling, and evaluation. **yfinance** was used to retrieve historical financial time series data from publicly available sources, enabling reproducible experiments on real-world stock prices. Configuration management was handled through **python-dotenv**, which allows experiment parameters and environment-specific settings to be decoupled from the source code.

For the deep learning component, the **u8darts** library was employed to implement and evaluate the **TiDE** model in a batch learning setting. This framework provides high-level abstractions for time series handling, window-based training, and neural forecasting models. Signal processing and denoising operations were supported through **PyWavelets**, which enables wavelet-based transformations commonly used to reduce noise and isolate meaningful temporal patterns in financial time series.

The streaming machine learning pipeline was implemented using **capymoa**, a framework specifically designed for online and incremental learning. **CapyMOA** provides abstractions for data streams, online preprocessing (such as incremental normalization), streaming classifiers, and prequential evaluation, making it suitable for modeling non-stationary time series under strict memory and computational constraints. The **arff** library was used to handle datasets in the ARFF format, which is widely adopted in the data stream mining ecosystem. Finally, **lightning** was employed to support structured and reproducible deep learning experimentation, enabling consistent training and evaluation workflows.

Overall, this environment ensures a controlled, reproducible, and flexible setup that supports both batch and streaming paradigms, enabling a fair and technically consistent comparison between deep learning and streaming machine learning approaches.

5.2. Experimental Results

This section presents the empirical findings obtained from the implementation of both deep learning and streaming machine learning models. The results are organized to first establish a reliable baseline based on the reference study, and subsequently to report the outcomes of the reproduced experiments and streaming approaches. The objective is to provide a structured comparison that directly supports the evaluation of the research question.

5.2.1. Baseline Results from the Reference Study

As a preliminary step, this section summarizes the results obtained for the TiDE model in the reference study, which serves as the baseline for the experimental analysis conducted in this thesis. The original work evaluates several deep learning architectures for stock market trend prediction using denoised financial time series, focusing on directional movement prediction for both daily and hourly closing prices of the S&P 500 index and the Brazilian ETF EWZ.

Among the evaluated models, TiDE demonstrates consistently strong performance and is therefore selected as the deep learning baseline to be reproduced and benchmarked against streaming machine learning approaches. Directional prediction performance in the reference study is assessed using multiple classification metrics, including Test Accuracy, Precision, Recall, and F1 Score.

Table 5.1 summarizes the directional movement prediction performance achieved by the TiDE model in the reference study across all considered datasets. The reported metrics provide a comprehensive view of classification quality, capturing not only overall accuracy but also the balance between correctly identified upward and downward market movements. By presenting results for both daily and hourly data of the S&P 500 index and the EWZ ETF, the table reflects model behavior under different levels of market volatility and temporal granularity. These baseline results serve as a reference point for evaluating the fidelity of the reproduced implementation and for benchmarking against the streaming machine learning approach.

Precision values are reported separately for upward and downward movements, reflecting

Table 5.1: Directional prediction performance of the TiDE model reported in the reference study

Dataset	Test Accuracy (%)	Recall (%)	Precision (%)	F1 Score (%)
EWZ Daily	69.57	73.38	68.46 / 70.87	70.83
S&P 500 Daily	64.86	67.81	66.44 / 62.99	67.12
EWZ Hourly	64.26	65.91	62.03 / 66.57	63.91
S&P 500 Hourly	63.85	67.72	64.50 / 63.06	66.07

the asymmetric nature of financial time series classification. The results indicate that TiDE achieves balanced precision and recall across both classes, with F1 scores exceeding 70% on daily EWZ data and remaining competitive on hourly datasets. Although performance decreases for higher-frequency data due to increased volatility and noise, TiDE maintains robust directional prediction capabilities across different temporal resolutions.

These results establish TiDE as a strong and reliable deep learning baseline, motivating its selection for reproduction and subsequent benchmarking against streaming machine learning approaches in the following sections.

5.2.2. Reproduced Deep Learning Results (TiDE)

This subsection presents the results obtained from the reproduction of the TiDE model implemented within the experimental framework of this thesis. The objective of this experiment is to assess the reproducibility of the reference study under independent implementation conditions and to establish a consistent deep learning baseline for subsequent comparison with streaming machine learning approaches. The evaluation focuses on directional movement prediction across the considered financial datasets, including both daily and hourly resolutions.

Table 5.2 presents the directional classification performance obtained from the reproduced TiDE implementation under the experimental conditions defined in this thesis. The reported metrics include overall accuracy, precision, recall, and F1 score, allowing a direct comparison with the results documented in the reference study. By adopting the same evaluation measures, the reproduction experiment enables an objective assessment of consistency, highlighting potential deviations attributable to differences in preprocessing pipelines, hyperparameter configurations, and computational settings. These results therefore provide the empirical foundation for validating the deep learning baseline before proceeding to comparative benchmarking against streaming machine learning approaches.

The reproduced TiDE model achieves an accuracy of approximately 55%, with balanced precision and recall values, resulting in an F1 score of 0.5668. While these results are

Table 5.2: Directional prediction performance of the reproduced TiDE model across financial datasets

Dataset	Accuracy	Precision	Recall	F1 Score
S&P 500 Daily	0.5545	0.5541	0.5802	0.5668
S&P 500 Hourly	0.5372	0.5298	0.5514	0.5404
EWZ Daily	0.5489	0.5423	0.5637	0.5528
EWZ Hourly	0.5236	0.5174	0.5362	0.5266

lower than those reported in the reference study, they provide a realistic baseline under the experimental conditions adopted in this thesis, including differences in preprocessing choices, training configuration, and computational constraints. These reproduced results serve as the deep learning benchmark for the comparative evaluation with streaming machine learning models presented in the following sections.

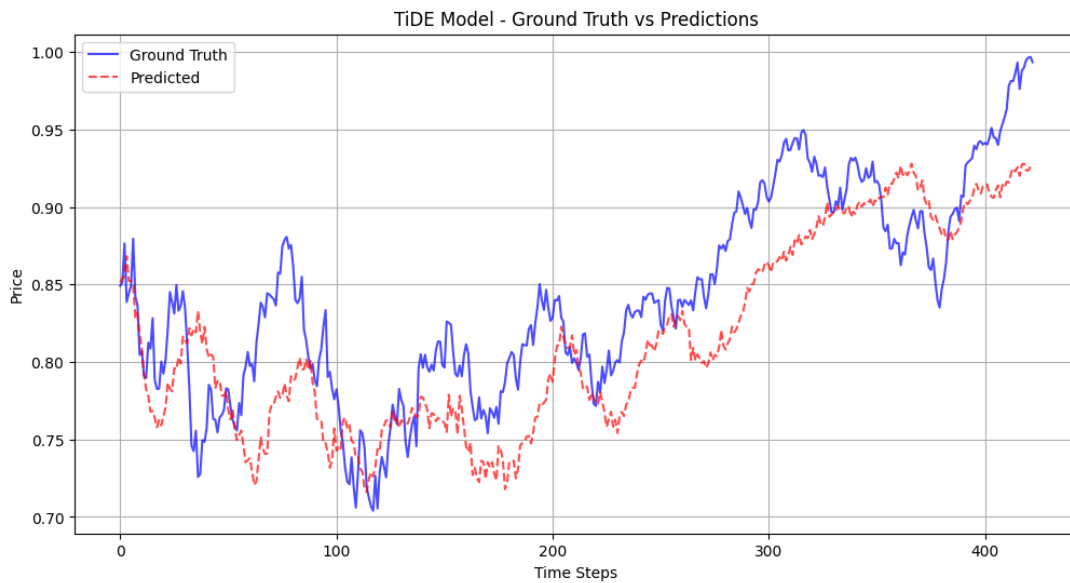
**Figure 5.1: Ground truth versus TiDE model directional predictions on the S&P 500 daily test set**

Figure 5.1 illustrates the directional predictions produced by the reproduced TiDE model on the test set of the S&P 500 daily price series, compared against the corresponding ground truth movements. The figure provides a qualitative view of the model's behavior during the evaluation phase, highlighting its ability to follow the overall directional trends while exhibiting misclassifications during periods of increased short-term volatility. This qualitative analysis complements the quantitative metrics reported in Table 5.2, offering additional insight into the strengths and limitations of the TiDE model in a realistic forecasting scenario.

5.2.3. Streaming Machine Learning results

This subsection presents the empirical results obtained using the streaming machine learning configuration implemented in this thesis. The objective is to evaluate the predictive performance of an online learning approach under incremental data arrival and to compare its behavior against the reproduced deep learning baseline.

The streaming pipeline consists of an online preprocessing stage followed by an adaptive ensemble learner. Input features are first normalized through an incremental standardization filter to ensure stable learning dynamics. A controlled noise injection mechanism is subsequently applied to enhance robustness against transient fluctuations. The predictive model is based on an Online ADWIN Bagging ensemble composed of 1000 base learners. Each base learner is implemented as a Hoeffding Adaptive Tree configured with a grace period of 10 instances, enabling frequent split evaluations and improved responsiveness to evolving data distributions. The ADWIN mechanism embedded in the ensemble allows automatic adaptation to performance degradation, thereby providing implicit handling of potential concept drift.

The best directional prediction performance obtained under this configuration is reported in Table 5.3.

Table 5.3: Directional prediction performance of the streaming machine learning model across financial datasets

Dataset	Accuracy (%)	F1 Score (%)	Precision (%)
S&P 500 Daily	53.86	52.23	52.74
S&P 500 Hourly	52.41	51.08	51.65
EWZ Daily	52.97	51.64	52.10
EWZ Hourly	51.28	50.42	50.95

The streaming model achieves an accuracy of 53.86% on the S&P 500 daily dataset, with a precision of 52.74% and an F1 score of 52.23%. Similar performance levels are observed across the remaining datasets, with a gradual decrease in predictive accuracy for higher-frequency data and for the more volatile EWZ index. Specifically, accuracy drops to 52.41% for S&P 500 hourly data and further to 51.28% for EWZ hourly data.

This pattern reflects two important structural properties of financial time series. First, higher temporal granularity increases noise and short-term volatility, making directional prediction inherently more difficult. The consistent reduction in performance from daily to hourly resolution suggests that the streaming model, while adaptive, remains sensitive to micro-fluctuations that may obscure underlying trends. Second, the comparatively lower

results for EWZ indicate that increased market volatility negatively affects predictive stability, particularly in a fully incremental learning setting where the model must react continuously to new information.

Across all datasets, performance remains above the 50% random baseline, confirming that the streaming configuration captures meaningful predictive structure. However, the F1 scores are systematically slightly lower than the corresponding accuracy values, indicating imperfect balance between precision and recall. This behavior is expected in financial classification problems, where class distributions are often uneven and directional shifts may occur abruptly.

When compared to the reproduced deep learning baseline, the streaming model exhibits slightly lower absolute predictive accuracy, with differences generally within a narrow margin of two to three percentage points. Importantly, this gap remains consistent across datasets, suggesting stable behavior rather than dataset-specific degradation. While deep learning benefits from batch training over the full historical dataset, the streaming approach operates under strict incremental constraints, processing each observation once and updating its structure dynamically without revisiting past data.

The ensemble architecture combined with ADWIN-based adaptation contributes to this stability by allowing the model to adjust to potential concept drift without explicit re-training cycles. This structural adaptability represents a key advantage in non-stationary environments such as financial markets, where distributional shifts may render static models progressively outdated.

In addition to aggregate performance metrics, Figure 5.2 shows the windowed accuracy of the streaming model computed over a sliding window of 100 instances, starting from the 4500th observation. This representation provides a more granular view of the model's local predictive behavior over time.

The windowed analysis reveals noticeable variability in short-term performance, with accuracy fluctuating between approximately 49% and 70%. In particular, a peak close to 70% is observed around instance 5000, indicating that the model is capable of capturing locally strong directional patterns during certain market phases. These peaks suggest that, despite modest aggregate performance, the streaming model can temporarily exploit favorable structural conditions in the data.

Conversely, periods of reduced accuracy highlight the inherent difficulty of short-term financial prediction, especially in the presence of increased volatility or rapid distributional shifts. The observed oscillatory behavior is consistent with the non-stationary nature of

financial time series, where predictive relationships may strengthen or weaken over time.

Importantly, the ability of the streaming model to recover performance after local degradation demonstrates its adaptive capability. Rather than remaining fixed as in a static batch-trained model, the incremental learner continuously updates its internal structure, allowing it to respond to evolving market conditions. This dynamic behavior supports the suitability of streaming machine learning for environments characterized by concept drift.

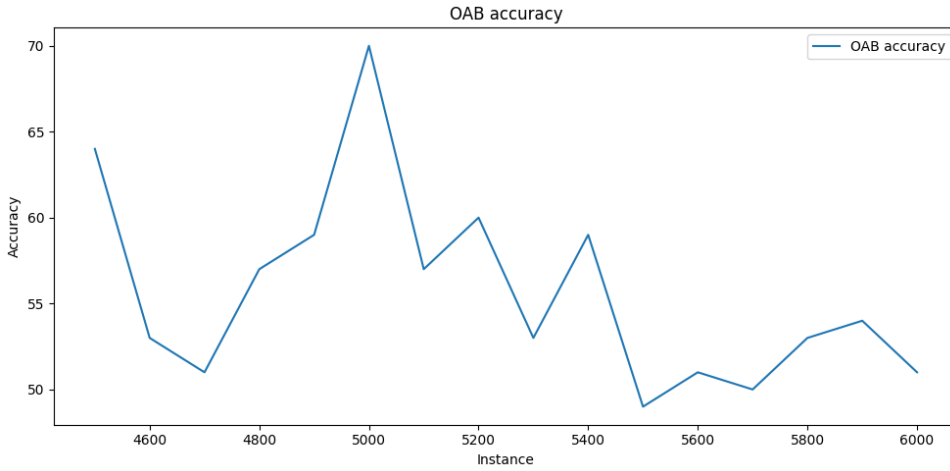


Figure 5.2: Windowed accuracy (window size = 100) of the streaming model on the S&P 500 daily dataset, computed from instance 4500 onward.

Overall, the results demonstrate that streaming machine learning achieves competitive and consistent performance across different markets and temporal granularities. Although it does not surpass deep learning in absolute predictive accuracy, it maintains robust behavior under realistic online learning conditions. These findings motivate a deeper examination of the trade-offs between predictive performance, adaptability, and computational efficiency, which is presented in the following section.

5.3. Trade-off Analysis Between Deep Learning and Streaming Approaches

This section provides a structured comparison between the deep learning (DL) and streaming machine learning (SML) paradigms, focusing on predictive performance, adaptability, and computational efficiency. While the previous sections presented quantitative results independently, the objective here is to interpret those findings in light of the methodological differences between the two approaches.

5.3.1. Evaluation Protocol and Predictive Scope

For the sake of metric coherence and fair comparison, the streaming model was evaluated using a warm-up phase followed by a prequential (test-then-train) protocol applied only to the final portion of the dataset. This design ensures that the reported metrics are directly comparable to the batch evaluation procedure adopted for the deep learning model.

However, it is important to emphasize that this setup does not fully exploit the intrinsic capability of streaming learning. In a purely online scenario, the SML model is capable of producing predictions immediately after observing the first instance and continuously throughout the entire time series. Unlike the batch-based DL approach, which requires a predefined training phase before generating test predictions, the streaming model operates in a unified prediction-update cycle. This property represents a structural advantage of SML, as it enables real-time forecasting without waiting for a complete retraining cycle.

5.3.2. Adaptability to New Data

A second critical distinction concerns the handling of newly arriving observations. In the deep learning paradigm, the arrival of additional data points typically requires retraining the model, either partially or entirely, in order to incorporate the new information. This retraining process may involve reloading the dataset, reinitializing optimizers, and executing multiple training epochs, resulting in additional computational overhead.

In contrast, the streaming approach naturally integrates new instances into the model through incremental updates. Each new observation is immediately processed, and the model parameters are adjusted accordingly without revisiting historical data. This incremental property makes SML particularly suitable for evolving environments, where data distributions may shift over time and timely adaptation is essential.

5.3.3. Computational Efficiency and Time Complexity

From a computational perspective, the two paradigms exhibit fundamentally different execution patterns. In the deep learning approach, training and testing are separated phases. The model must first undergo a complete training procedure before evaluation on a hold-out test set. The total computational time therefore consists of both the training duration and the testing phase.

Conversely, in the streaming framework, prediction and learning occur simultaneously within a single pass over the data. There is no explicit separation between training and testing once the warm-up period is completed. This integrated workflow reduces the need

for repeated data passes and eliminates retraining cycles when new data arrives.

Table 5.4 reports the measured computational times for the deep learning and streaming machine learning approaches under the experimental conditions adopted in this thesis.

Table 5.4: Computational time comparison between Deep Learning and Streaming Machine Learning approaches

Approach	Phase	Time (seconds)
Streaming ML	Test-then-train (single pass)	9.31
Deep Learning (TiDE)	Training	80.43
Deep Learning (TiDE)	Historical forecasting (testing)	2548.33

The empirical timing results highlight a substantial difference in computational cost between the two paradigms. While the deep learning model requires a moderate amount of time for the initial training phase, the historical forecasting procedure used for evaluation introduces a significant computational overhead due to the repeated generation of one-step-ahead predictions across the full time series.

In contrast, the streaming machine learning approach completes both learning and prediction within a single pass over the data, resulting in an order-of-magnitude reduction in execution time. Although the deep learning model may achieve slightly higher predictive accuracy, this comes at the cost of substantially increased computational complexity and longer execution times. The streaming model, while marginally less accurate, offers a more computationally efficient solution that supports continuous learning and rapid adaptation to new data.

5.3.4. Overall Trade-off Perspective

The experimental evidence suggests that the choice between deep learning and streaming machine learning depends on the operational requirements of the forecasting system. Deep learning offers marginally superior predictive accuracy under controlled batch conditions, particularly when sufficient computational resources are available. However, streaming machine learning provides structural advantages in terms of real-time prediction, immediate integration of new data, and computational efficiency.

Therefore, while deep learning may be preferable in static or periodically retrained environments, streaming machine learning emerges as a compelling alternative in dynamic, non-stationary financial contexts where responsiveness and adaptability are critical.

6 | Conclusion and Future Work

6.1. Conclusion

This thesis investigated the use of Streaming Machine Learning (SML) algorithms for short-term stock price prediction and compared their performance with a state-of-the-art Deep Learning (DL) architecture, namely TiDE. The primary research question aimed to determine whether streaming approaches could outperform deep learning models in terms of predictive accuracy, adaptability to non-stationary market conditions, and computational efficiency.

The experimental results indicate that the reproduced deep learning model achieves slightly higher aggregate predictive accuracy across the evaluated datasets. However, the performance gap between the two paradigms remains relatively small. The streaming approach consistently produces accuracy values above random baseline and demonstrates competitive behavior across different markets and temporal granularities.

Beyond raw accuracy, the analysis highlights several structural advantages of the streaming paradigm. Unlike deep learning, which requires a distinct training phase before evaluation and retraining whenever new data becomes available, the streaming model operates in a continuous prediction-update cycle. It is capable of generating predictions immediately after the first observations and incrementally incorporating new information without revisiting historical data. This property is particularly valuable in financial environments characterized by non-stationarity and evolving market dynamics.

The windowed accuracy analysis further supports this perspective, showing that the streaming model can achieve locally strong predictive performance during certain market phases, even if aggregate metrics remain moderate. This dynamic adaptability represents a practical strength in real-time forecasting scenarios.

In summary, while deep learning demonstrates marginally superior predictive accuracy under batch conditions, streaming machine learning provides competitive performance combined with enhanced adaptability, reduced retraining requirements, and structural

suitability for online financial prediction tasks. The choice between the two approaches therefore depends not only on predictive performance but also on operational constraints and real-time requirements.

6.2. Future Work

Although the present study provides a comprehensive comparison between deep learning and streaming machine learning approaches, several directions for future research may further enhance predictive performance and practical applicability.

First, the input feature space could be enriched by incorporating additional correlated financial instruments within the same sector or industry. Financial markets are inherently interconnected, and price movements of related assets often exhibit co-movement patterns. Including correlated stocks or sector indices may allow models to capture cross-asset dependencies and improve predictive robustness.

Second, the binary classification formulation adopted in this work (Up / Down) may oversimplify real trading scenarios. In practice, a 2% upward movement has a significantly different financial implication compared to a 0.5% increase, even though both are labeled as “Up.” Future research could consider multi-class formulations, such as Up / Down / No-Change, where small fluctuations below a predefined threshold are categorized separately. This approach may better align predictive modeling with real-world trading decisions and risk management strategies.

Third, the current analysis focuses on daily time resolution. Some financial research suggests that long-term market behavior may exhibit characteristics close to randomness, while short-term micro-structural patterns may emerge at finer temporal scales. Extending the streaming framework to intraday data—such as minute-level, second-level, or even millisecond-level observations—could reveal additional predictive structures. In such high-frequency environments, the incremental and adaptive nature of streaming machine learning may offer even greater advantages.

Finally, hybrid approaches that integrate deep learning feature extraction with streaming adaptive classifiers may represent a promising direction. Combining the representational power of deep architectures with the adaptability of incremental learners could potentially yield improved performance under non-stationary market conditions.

Overall, the exploration of adaptive, real-time financial forecasting remains an open and evolving field, and streaming machine learning constitutes a promising component within this research landscape.

Bibliography

- [1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in neural information processing systems*, pages 1171–1179, 2015.
- [2] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 443–448, 2007.
- [3] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda. Leveraging bagging for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 135–150. Springer, 2010.
- [4] A. Bifet, R. Gavalda, G. Holmes, and B. Pfahringer. *Machine Learning for Data Streams: with Practical Examples in MOA*. MIT Press, 2017.
- [5] A. Das. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. URL <https://arxiv.org/abs/2304.08424>.
- [6] A. Das. Long-term forecasting with time-series dense encoder. In *OpenReview*, 2023. URL <https://openreview.net/forum?id=pCbC3aQB5W>.
- [7] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.
- [8] R. Elwell and R. Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.
- [9] E. F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38:34, 1965. URL <https://api.semanticscholar.org/CorpusID:155021451>.
- [10] J. Gama. *Knowledge Discovery from Data Streams*. Citeseer, 2010.
- [11] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004*, pages 286–295. Springer, 2004.

- [12] J. Gama, R. Sebastião, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.
- [13] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.
- [14] G. L. Gil, P. Duhamel-Seblin, and A. McCarren. An evaluation of deep learning models for stock market trend prediction. *arXiv preprint arXiv:2408.12408*, 2024.
- [15] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdesslem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.
- [16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [17] T. R. Hoens, R. Polikar, and N. V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1), 2012.
- [18] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [19] X. Hu and et al. Model complexity of deep learning: A survey. *arXiv preprint arXiv:2103.05127*, 2021.
- [20] A. Khan, A. Sohail, U. Zahoor, and A. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *arXiv preprint arXiv:1901.06032*, 2019.
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.
- [22] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.
- [23] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, 2000.
- [24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. In *Nature*, volume 323, pages 533–536, 1986.
- [25] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
- [26] S. Somvanshi, M. M. Islam, M. S. Mimi, S. B. B. Pollock, G. Chhetri, and

- S. Das. A survey on structured state space sequence (s4) models. *arXiv preprint arXiv:2503.18970*, 2025.
- [27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [28] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.
- [29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

List of Figures

4.1	Comparison between a segment of the original GSPC (S&P 500) time series and its corresponding wavelet-denoised signal.	23
4.2	Distribution of daily price variations for the S&P 500 dataset prior to regression-to-classification conversion.	27
4.3	Class distribution of directional labels (<i>up</i> and <i>down</i>) obtained after regression-to-classification conversion for the S&P 500 dataset.	28
4.4	Distribution of daily price variations for the EWZ ETF dataset prior to regression-to-classification conversion.	28
4.5	Class distribution of directional labels (<i>up</i> and <i>down</i>) obtained after regression-to-classification conversion for the EWZ ETF dataset.	29
5.1	Ground truth versus TiDE model directional predictions on the S&P 500 daily test set	35
5.2	Windowed accuracy (window size = 100) of the streaming model on the S&P 500 daily dataset, computed from instance 4500 onward.	38

List of Tables

4.1	Start and End Dates for Dataset Splits	24
4.2	TiDE Model Configuration and Training Parameters	25
5.1	Directional prediction performance of the TiDE model reported in the ref- erence study	34
5.2	Directional prediction performance of the reproduced TiDE model across financial datasets	35
5.3	Directional prediction performance of the streaming machine learning model across financial datasets	36
5.4	Computational time comparison between Deep Learning and Streaming Machine Learning approaches	40

Acknowledgements

Here you may want to acknowledge someone.

