VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

INFORMATICS STUDY PROGRAMME

MASTER'S THESIS

Artificial Intelligence for Time Series

Dirbtiniai neuroniniai tinklai laiko eilutėms

Author:	Md Jakaria Mashud Shahria	(parašas)
Supervisor:	Assist. prof. dr. Linas Litvinas	(parašas)
Reviewer:	Prof. Linas Laibinis	(parašas)

Vilnius, 2026

Acknowledgements

The day I was accepted into Vilnius University was a deeply proud moment for me. At that moment, my dreams of pursuing higher studies finally began to take shape. My journey as a student has not always been easy, yet the time has flown by so fast. Throughout this journey, especially during times of hardship, my mentor, guardian, and supervisor, Assistant Professor Linas Litvinas, has shown me the way. His guidance has brought me to this final stage of my degree.

I consider myself especially lucky to have had his support through my most difficult times. In my culture, it is said that teachers are a student's second parents, acting as a mother and father who provide mentorship, emotional support, and a role model to look up to. I would like to deeply thank him for his generous time and the unwavering guidance he has given me throughout my studies.

I would also like to thank all of my professors for sharing their valuable knowledge with me throughout my courses. Furthermore, I want to extend my gratitude to all the faculty, staff, and members of the Faculty of Mathematics and Informatics who manage this institute with such great care.

A special thank to my friends, Afjal and Niaz, whose own journeys to Lithuania for higher studies inspired and guided me here. Last but certainly not least, I want to thank my parents and my family for their unwavering understanding, love, and support.

This journey has taught me resilience, patience, and the value of hard work. As I stand on the horizon of completing this chapter, I carry forward not only academic knowledge but lifelong memories of my time in Vilnius.

Santrauka

Šiame darbe nagrinėjama, ar vieno paslėpto sluoksnio tiesioginio perdavimo neuroninis tinklas (FFNN), pagrįstas Kibenko universaliąja aproksimacijos teorema, gali pasiekti praktiškai realų kryptinį EUR/USD uždarymo kainos prognozavimo tikslumą, kai jis apmokytas naudojant iš anksto filtruotą istoriškai panašių rinkos modelių duomenų rinkinį.

Daugiau nei dvidešimties metų kasdieniai EUR/USD duomenys buvo konvertuoti į procentinę grąžą ir suskirstyti į penkių dienų slenkančio lango požymių vektorius. Euklido atstumo algoritmas filtravo panašius modelius naudodamas slenkstį $\tau = 1{,}4$, gaudamas 145 mokymosi fazės ir 11 bandymo fazės modelių. FFNN buvo apmokyti šiame filtruotame poaibyje naudojant sigmoidines ir ReLU aktyvinimo funkcijas, įvertinti pagal 100 atsitiktinių sėklų ir išplėsti naudojant K artimiausių kaimynų konsensuso ansamblį.

Sigmoidinio modelio testo tikslumas svyravo nuo 27,27% iki 81,82%, o ReLU modelio nuo 9,09% iki 81,82%, abiem 100 sėklų medianoms stabilizuojantis ties 54,55%. Geriausia vienos sėklos ReLU konfigūracija pasiekė 72,73% testo tikslumą, teisingai klasifikuodama visus neigiamus rezultatus. ReLU-KNN konsensuso ansamblis pateikė 81,82% atitikimo rodiklį, o konsensuso tikslumas 55,56%.

Tyrimo išvada yra ta, kad triukšmo mažinimas naudojant Euklido atstumo filtravimą yra lemiamas veiksnys siekiant patikimų kryptinių prognozių, leidžiantis minimaliai neuroninio tinklo architektūrai prasmingai atlikti nematomų finansinių duomenų analizę.

Keywords: tiesioginio ryšio neuroninis tinklas, EUR/USD prognozavimas, Euklido atstumo filtravimas, universali aproksimacijos teorema, krypties tikslumas, ReLU aktyvinimas, ansamblis, konsensusas.

Summary in English

This thesis investigates whether a single-hidden-layer Feedforward Neural Network (FFNN), based on Cybenko's Universal Approximation Theorem, can achieve practically viable directional prediction accuracy for the EUR/USD closing price when trained on a pre-filtered dataset of historically similar market patterns.

Over twenty years of daily EUR/USD data were converted to percentage returns and organized into five-day sliding window feature vectors. A Euclidean distance algorithm filtered similar patterns using threshold $\tau = 1.4$, yielding 145 learning-phase and 11 test-phase patterns. FFNNs were trained on this filtered subset using sigmoid and ReLU activation functions, evaluated across 100 random seeds, and extended with a K-Nearest Neighbours consensus ensemble.

The sigmoid model ranged from 27.27% to 81.82% test accuracy, while the ReLU model ranged from 9.09% to 81.82%, with both 100-seed medians stabilizing at 54.55%. The best single-seed ReLU configuration achieved 72.73% test accuracy, correctly classifying all negative outcomes. The ReLU-KNN consensus ensemble produced an agreement rate of 81.82% with a consensus accuracy of 55.56%.

The study concludes that noise reduction through Euclidean distance filtering is a decisive factor in achieving reliable directional predictions, enabling a minimal neural network architecture to perform meaningfully on unseen financial data.

Keywords: feedforward neural network, EUR/USD forecasting, Euclidean distance filtering, Universal Approximation Theorem, directional accuracy, ReLU activation, ensemble, consensus.

Contents

1. Introduction

The foreign exchange market is an important financial system globally. Forex or Foreign exchange makes the exchange of currencies between nations by enabling international trade and investment. In this world, every nation has their own currency but among these currencies EUR/USD pair traded in this dynamic market. EUR/USD holds particular importance because of its high trading and representation of the economic relationship between the European Union and United States. Predicting the closing price and opening price is a difficult task for the traders, financial analysts, and policy makers. The fundamental volatility and complex nature of Forex markets accurate forecasting is a considerable challenge.

Statistical models which are traditional methods have been commonly used for analyzing time series data for Forex markets. These models often fall short when they faced with highly dynamic and nonlinear patterns of financial data though they have proven effective in stable and predictable environments. This limitation has pushed researchers and professionals such they explore advanced computational techniques in the field of artificial intelligence. Artificial Neural Networks(ANNs) have appeared as a powerful platform for capturing complex patterns in data and providing reliable predictions. Single-layer neural networks with a sigmoidal activation function are universal approximations capable of modeling any continuous function [Cyb89]. This theoretical foundation justifies the application of ANNs in forecasting complex and nonlinear datasets.

While the application of ANNs to financial forecasting is a well-established field, the existing literature often gravitates towards increasingly complex architectures. This pursuit of complexity can sometimes overshadow the potential of simpler, more computationally efficient models. A rigorous, systematic investigation into the performance ceiling of a foundational architecture like the Feed-Forward Neural Network (FFNN), when subjected to advanced optimization and feature engineering techniques, remains an area requiring deeper exploration. The research problem is therefore defined, to determine if a systematically optimized FFNN, augmented with a comprehensive set of engineered features, can provide a practically viable and high-performing solution for forecasting the EUR/USD closing price, thereby offering a compelling balance between predictive accuracy, computational efficiency, and interpretability.

1.1. Relevance and Novelty

The importance of this research lies in its potential to address pressing challenges in financial forecasting. The Forex market's volatility and sensitivity to external factors demand advanced tools capable of adapting to dynamic environments. Traditional methods rely on assumptions of linearity and stationarity which often fail to capture the complexities of financial time series. This limitation has been highlighted in previous studies [BJR08] who emphasized the need for more flexible approaches to time series analysis.

The original contribution of this research is not the general use of ANNs in finance, as this is already a well-established field. Instead, the focus is on a specific methodological approach. The novelty therefore lies not in applying ANNs to finance, but in the specific combination of Euclidean distance filtering with a minimally complex, theoretically grounded architecture. Only historically similar patterns are kept instead of using the full raw market history. Cybenko [Cyb89] proved that a single hidden layer with a sigmoidal activation can approximate any continuous function. Therefore, the main research question is not whether the architecture has enough capacity, but whether gradient-based training on a small filtered dataset can achieve that theoretical capacity in practice.

1.2. Aim and Objective

The primary aim of this research is to investigate whether a systematically optimized single-hidden-layer Feedforward Neural Network (FFNN), based on Cybenko's Universal Approximation Theorem, can achieve practically viable directional prediction accuracy for the EUR/USD closing price when trained on a pre-filtered dataset of historically similar market patterns. The central hypothesis is that Euclidean distance-based pattern filtering can bridge the gap between the UAT's theoretical guarantees and practical predictive performance on noisy financial time series.

1.2.1. Research Tasks and Objectives

- Conduct a systematic literature review of traditional (ARIMA), hybrid, and deep learning (LSTM, Transformer) models for EUR/USD forecasting, and establish theoretical justification for FFNN use through the UAT.
- Collect daily EUR/USD closing price data (December 2003–December 2025), convert to percentage returns, and construct five-day sliding window feature vectors.
- Apply Euclidean distance filtering across base vectors $\boldsymbol{B}$ and threshold values $\tau = 0.5$ to 1.5

to identify historically consistent, directionally imbalanced pattern subsets, validated across both learning and testing phases.

- Design, train, and evaluate single-hidden-layer FFNNs using sigmoid and ReLU activations, including a 100-seed median analysis and a KNN consensus ensemble.
- Evaluate all configurations using directional accuracy, analyze the theory-to-practice gap, and propose directions for future research.

1.3. Expected Outcomes

The expected outcome is to find a reliable neural network model where initial starting with the feed-forward neural network that accurately predicts the closing prices of the EUR/USD currency pair. The model is expected to handle the nonlinear and dynamic patterns of Forex time series data more effectively than traditional methods. This research is expected to get the following outcomes:

- Find the best pattern from the dataset for training and test.
- Creating neural network model and evaluating including the feedforward neural network (FFNN), can accurately predict EUR/USD closing prices direction.
- Identify best practices for designing and training FFNNs in time series forecasting such as optimal hyperparameters and data preparation techniques.

2. Literature Review

2.1. Background

Trying to predict the direction of foreign exchange rates has long been one of the toughest puzzles in finance, mostly because the markets themselves are chaotic, unpredictable, and full of noise. The EUR/USD pair, being the most traded in the world, has naturally become the gold standard for testing new ideas. Lately, all the attention has shifted to highly complex 'deep learning' models like LSTMs and Transformers. These newer tools have become so popular that they've pushed the simpler, foundational models to the sidelines, treating them almost as historical footnotes. This review challenges that trend. Making a strong, clear case for why the original Feedforward Neural Network (FFNN) is not just relevant, but essential. It is not about seeing it as an old-fashioned competitor, but as a vital tool for understanding the real-world trade-offs between a model's complexity and its actual performance. The FFNN's value is four-fold: it is powerful enough to model complex patterns, it is far more efficient with data and computing power, it works brilliantly as a building block in larger systems, and most importantly it serves as the perfect, honest baseline to measure if those fancier models are truly earning their keep. To build this argument, first exploring from the modern EUR/USD market, then walk through the history of forecasting models from the traditional to the complex. By comparing the strengths and weaknesses of the FFNN, LSTM, and Transformer architectures, pinpointing the gaps in today's research and provide clear evidence for why this simple, effective model still deserves a central place in financial forecasting.

2.2. The EUR/USD Market

Choosing which financial instrument to test a forecasting model on is a critical first step, because the market's own personality shapes the entire challenge. For decades, the EUR/USD exchange rate has been the go-to benchmark for research, and this choice is no accident. It stems from the pair's unique position at the very heart of the global financial system. To understand why it is the standard and why it is so difficult to predict to appreciate its enormous size, its complex structure, and the diverse motivations of everyone trading it. The EUR/USD is not just another large market; it is a dynamic system where the sheer volume and liquidity create a 'worst-case scenario' of noise and complexity. This makes it the perfect crucible for testing the true limits of any forecasting model.

2.2.1. Market Scale and Dominance

The scale of the global Foreign Exchange (Forex) market is difficult to overstate, with the Bank for International Settlements (BIS) reporting a daily trading volume of $7.5 trillion in its 2022 survey [BIS22]. This enormous figure, which grew 14% from 2019, is a clear indicator of the market's intense activity, particularly during times of economic instability. Within this massive system, one currency pair stands above all others: the EUR/USD. It alone accounts for 22.7% of all daily trades [BIS22]. The reason for its central role is straightforward: it links the world's two largest economic areas, the Eurozone and the United States, which creates unmatched liquidity and makes it the primary trading hub for everyone from central banks to individual investors.

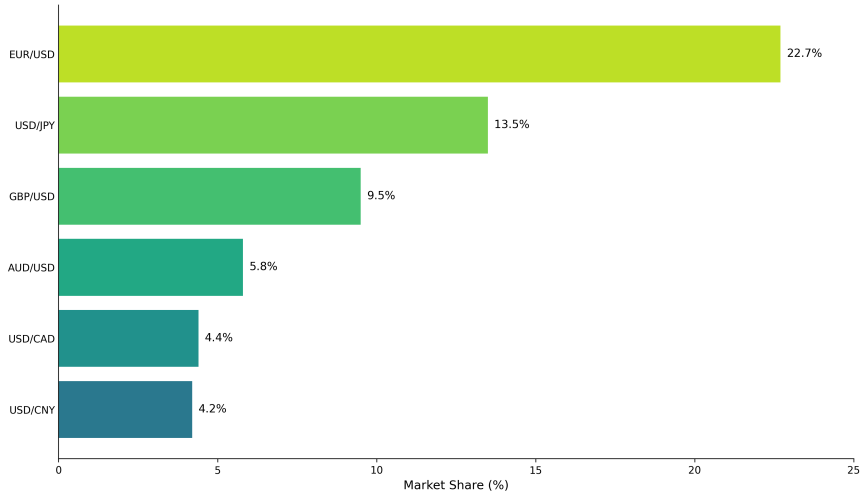

Figure 1: Forex Market Share by Currency Pair [BIS22].

The importance of the EUR/USD is not just a historical fact; the latest data from 2024 shows its trading activity is still expanding. For instance, the New York Foreign Exchange Committee's survey from October 2024 reported a regional daily volume of $1.196 trillion, marking a significant 17.2% jump in just one year [NFX24]. Most tellingly, the report pinpointed the EUR/USD as the primary engine of this growth, with its daily trading value surging by $89.3 billion. This is not just a local trend that we see the same pattern in other global hubs. Australia's market survey for April 2024 also listed EUR/USD as its second most traded pair, with a daily volume of $17.6 billion [AFX24]. Taken together, these figures make one thing clear: even if its slice of the overall market fluctuates, the sheer amount of EUR/USD being traded is growing, cementing its role as a critical subject for financial research.

2.2.2. Global Dynamics and Enduring

While it is true that the global currency landscape is shifting, with emerging currencies like the Chinese Renminbi gaining ground, this trend does not diminish the central role of the EUR/USD.

At first, the rise of new currencies might suggest that research focused on the Euro-Dollar is becoming less critical. However, a closer look reveals the opposite. The key is to distinguish between a currency's share of the market and its absolute trading volume. While the EUR/USD's relative share saw a slight dip between 2019 and 2022 [BIS22], recent 2024 regional data confirms its absolute daily trading volume has continued to climb. In other words, the EUR/USD market is larger and more liquid today than ever before, which transforms the challenge of forecasting its movements from a legacy issue into a problem of ever-increasing scale. As a result, research on the EUR/USD remains at the very forefront of finance, not because of habit, but because it continues to be the ultimate and most complex benchmark for testing the true power of any predictive model.

2.3. The Limits of Linearity

Deep learning did not just appear in finance out of nowhere; its adoption was a direct response to the shortcomings of traditional approaches. For years, standard statistical models like ARIMA were the primary tools for forecasting, but they consistently proved unable to fully capture the complex and often chaotic behavior of financial markets. This persistent gap between the models and reality signaled that a fundamentally new approach was required. The real turning point arrived with the development of hybrid models, which successfully blended the linear components of older methods with new, non-linear techniques. The success of these systems offered definitive proof of a core concept: that financial data is not simply linear or non-linear, but a mixture of both. This crucial insight was the breakthrough that paved the way for neural networks, establishing them as an essential tool for modern forecasting.

2.3.1. The Inadequacy of Linear Models: The Case of ARIMA

For decades, the forecasting world relied heavily on the Autoregressive Integrated Moving Average (ARIMA) model, famously systematized by Box and Jenkins [BJR08]. At its core, the model operates on an elegant principle: that a future value can be estimated from a linear combination of its own past values and its past forecast errors. These two elements are its Autoregressive (AR) and Moving Average (MA) components. The model's 'Integrated' dimension refers to the crucial pre-processing step of differencing the data to achieve stationarity. Yet, the enduring power of ARIMA came not just from the model itself, but from the structured, iterative methodology that Box and Jenkins developed alongside it. This roadmap of model identification, parameter estimation, and diagnostic checking, is what cemented ARIMA's status as the indispensable workhorse in forecasting for generations.

While ARIMA models, well-regarded tool in statistics, and common, they face a fundamental problem when applied to the financial world. The heart of the issue is that these models are designed to see the world in straight lines, assuming that patterns in the data follow a predictable, linear path. Financial markets, however, are far more complex and messy; they are shaped by sudden structural changes, chaotic behavior. Because of this basic mismatch, a great deal of research shows that simple linear models struggle to effectively predict short-term exchange rate movements. Consequently, when a tool like ARIMA is used to forecast a turbulent and non-linear currency pair, like the EUR/USD, the results are often inaccurate disappointingly, as the model's rigid assumptions are a poor fit for the complex reality of financial markets. Another key weakness is the model's static design. An ARIMA model is built using a specific portion of historical data, and once its parameters are set, they remain fixed. This becomes a major drawback because the model cannot adapt on its own when the fundamental behavior of the market shifts a constant reality in finance. As the model's predictions tend to quickly flatten out and just point to the historical average after a few periods, this inflexibility is particularly damaging for long-range forecasting. This failure to adjust to the dynamic and often unstable conditions of financial markets is a significant flaw in the traditional ARIMA approach and was a key reason that pushed researchers to explore more responsive and adaptive methods.

2.3.2. Hybrid Models: A Bridge to Non-Linearity

In response to the clear limitations of standalone linear models, researchers began to explore a new paradigm: hybrid modeling. The core idea behind the hybrid approach is intuitive yet powerful. It acknowledges that a financial time series is unlikely to be purely linear or purely non-linear, but rather a complex composite of both. Therefore, instead of trying to fit the entire series with a single type of model, the hybrid approach seeks to decompose the series into its linear and non-linear components and model each part with the most appropriate tool. This shift was part of a broader recognition of the power of Artificial Neural Networks (ANNs) for forecasting that gained traction in the late 1990s [ZPH98].

This led to the development of the now-classic ARIMA-ANN hybrid model. The methodology, though implemented with minor variations across studies, generally follows a consistent two-stage process:

1. **Linear Modeling:** First, an ARIMA model is fitted to the original time series data (X_t). This step is designed to capture the linear structure present in the data. The ARIMA model produces a series of one-step-ahead forecasts (F_t).

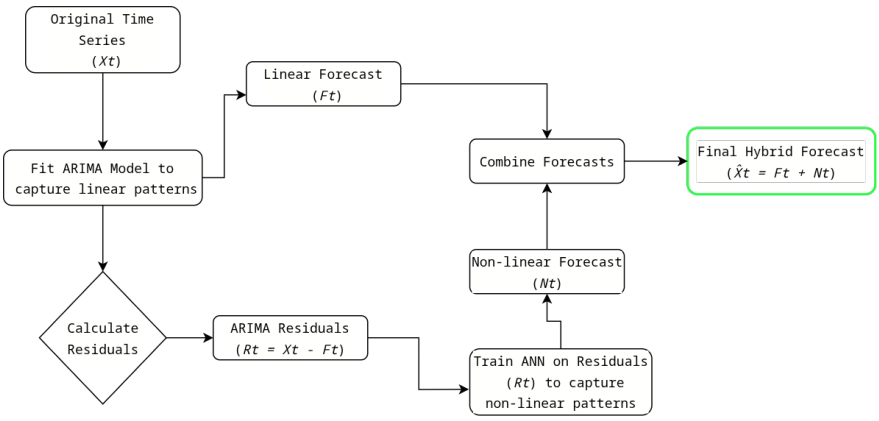

Figure 2: The workflow architecture of ARIMA-ANN.

2. **Non-Linear Modeling of Residuals:** The residuals from the ARIMA model are then calculated as the difference between the actual values and the linear forecasts ($R_t = X_t - F_t$). These residuals are not treated as random white noise. Instead, the crucial assumption is that they contain the systematic non-linear patterns that the ARIMA model was unable to capture. An Artificial Neural Network (ANN), with its ability to approximate arbitrary non-linear functions, is then trained to model these residuals. The ANN learns to predict the non-linear component of the series by learning the function that maps past residuals to the next residual.

3. **Combination of Forecasts:** The final forecast is an aggregation of the forecasts from the two stages. The linear forecast from the ARIMA model is combined with the non-linear forecast from the ANN (which is forecasting the error of the linear model) to produce a single, composite prediction.

A large and growing stack of studies backs up the value of mixing models, repeatedly showing that combining a linear approach with a neural net produces forecasts that are far stronger than either method alone. For example, a well-cited 2003 paper [Zha03] on the GBP/USD exchange rate reported that an ARIMA-ANN blend outperformed each single model, a finding later mirrored in a 2025 analysis of the volatile USD/NGN pair [BQ25]. The key reason is straightforward: ARIMA sketches the obvious linear trend, while the neural net is left to pick up the messiest, non-linear wiggles. So, when that leftover residue is neatly captured by the network, it tells us those remnants arent mere random scatter-they actually follow a real, predictable pattern. In short, forecasting is not a cruel choice between linear and non-linear; it is about accepting that market data is both at once and building a model smart enough to handle both halves together.

2.4. Feedforward Neural Networks as Universal Function Approximators

With the limitations of linear models established and the value of non-linear modeling demonstrated by hybrid systems, the focus of advanced forecasting research shifted decisively towards Artificial Neural Networks (ANNs). Among the diverse family of neural network architectures, the Feedforward Neural Network (FFNN), or Multi-Layer Perceptron (MLP), is the most foundational. While it lacks the explicit mechanisms for handling sequential data found in more modern architectures, its application to forecasting is not merely an ad-hoc heuristic. It is grounded in a powerful and elegant piece of mathematical theory: the Universal Approximation Theorem (UAT). This theorem provides the theoretical justification for using FFNNs as general-purpose function approximators, capable, in principle, of learning the complex, non-linear relationships that govern financial markets. Understanding the power and the practical limitations of the UAT is essential for situating the FFNN within the broader landscape of forecasting models.

2.4.1. The Architecture and the Theorem

An FFNN is characterized by its simple, directed structure. It is composed of an input layer, which receives the feature vector; one or more hidden layers, which perform the bulk of the computation; and an output layer, which produces the final prediction. Information flows in one direction only, from input to output, without any loops or cycles, distinguishing it from recurrent architectures. Within each hidden layer, every neuron receives inputs from all the neurons in the preceding layer. It computes a weighted sum of these inputs, adds a bias term, and then passes this result through a non-linear activation function (such as the sigmoid, hyperbolic tangent, or ReLU function). It is this combination of weighted sums and non-linear activations, repeated across multiple layers, that gives the network its computational power.

The primary theoretical pillar supporting the use of FFNNs for complex tasks like financial forecasting is the **Universal Approximation Theorem (UAT)**. In its most widely cited form, the theorem states that a standard FFNN with just a single hidden layer containing a finite number of neurons can approximate any continuous function on a compact (i.e., closed and bounded) subset of R^n to any desired degree of accuracy. The only key requirement is that the activation function used by the neurons must be non-polynomial. This remarkable result, first proven for sigmoid activation functions by [Cyb89] and then generalized by [Hor91] and others, has several profound implications:

Power of the Architecture: [Hor91] demonstrated that it is the multi-layer, feed-forward

architecture itself that endows the network with its universal approximation capabilities, not the specific choice of a particular activation function. As long as the activation function introduces non-linearity (i.e., is not a polynomial), the network can, in theory, approximate any continuous function.

Leshno et al. [LLP+93] proved that any multilayer feedforward network is a universal approximator if its activation function is locally bounded, piecewise continuous, and not a polynomial. Since ReLU meets all three conditions, it is covered by this theorem. Park et al. [PYL+21] further proved that the minimum width required for ReLU networks to universally approximate L^p functions is exactly $\max\{d_x + 1, d_y\}$, where d_x and d_y are the input and output dimensions. For a network with five inputs and one output, a minimum of six neurons in the hidden layer is both necessary and sufficient. Kim et al. [KMP24] refined this bound further for ReLU-like activations on compact domains, confirming that $\max\{d_x, d_y, 2\}$ is sufficient. Together, these results establish that ReLU is not only a practical improvement over sigmoid, but an equally sound theoretical choice within the universal approximation framework.

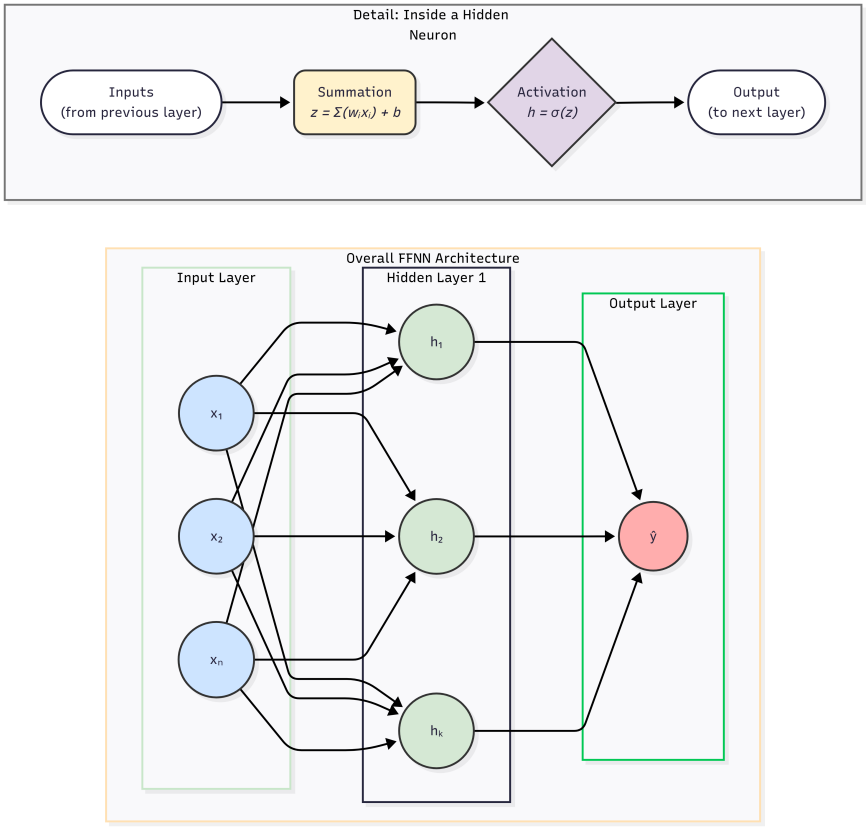

Figure 3: A Feedforward Neural Network (FFNN) architecture with a single hidden layer.

Arbitrary Width vs. Arbitrary Depth: The classical theorems focused on the 'arbitrary width' case, proving that a single, sufficiently wide hidden layer is enough. More recent research

has extended these results to the 'arbitrary depth' case, showing that deep networks with multiple, narrower hidden layers also possess universal approximation properties. These results are particularly relevant for modern deep learning, which favors depth, and have been proven for networks using the popular ReLU activation function [Han19; Hua20; PYL+21; KMP24].

Application to Forecasting: The task of time series forecasting can be mathematically framed as a problem of function approximation. Seeking to find a function, f, that maps a set of past observations and other relevant features to a future value: $y_t = f(y_{t-1}, y_{t-2}, ..., x_t, x_{t-1}, ...)$. The UAT provides the crucial theoretical guarantee that a sufficiently large FFNN is capable, in principle, of learning this complex, non-linear function f directly from the data.

2.4.2. Task Specification

To predict the closing price direction, a single hidden layer Feedforward Neural Network (FFNN) will be used. The model will have a specific and simple architecture: an input layer which takes past days' closing prices, a single hidden layer that uses the sigmoid activation function, and an output layer that produces the final price prediction. This structure is supported by the theorem, which states that even a network with a single hidden layer is powerful enough to approximate the complex, non-linear functions found in financial data. The input for the network will be past days' financial data. For example, giving the past five days, including the current day's closing price, the model will predict the closing price of tomorrow(the sixth day). The network will then be trained.

It is important to note that the UAT does not require the input data itself to be continuous. Rather, it requires that the target function being approximated is continuous on a compact domain. The assumption here is that there exists some underlying continuous function mapping five-day return patterns to the next day's return, even if the observed data contains noise and discrete sampling. The network's task is therefore to approximate this underlying function, and the linear output layer ensures the prediction remains a continuous real-valued output.

2.4.3. The Gap Between Theory and Practice

First and foremost, the UAT is an existence theorem, not a constructive one. It guarantees that a network with weights and biases capable of achieving the desired approximation exists. It does not, however, provide an algorithm for finding those optimal parameters. The standard method for training neural networks, backpropagation with gradient descent, is an optimization algorithm that seeks to minimize a loss function. There is no guarantee that this process will converge to the global minimum and find the ideal set of weights; it can easily get stuck in one of the numerous local minima that populate the complex loss landscapes of neural networks.

Second, while the Universal Approximation Theorem (UAT) guarantees expressive power under idealized conditions of infinite noise-free data, financial time series are finite and extremely noisy. An FFNN large enough to approximate the true signal also has sufficient capacity to memorize training noise, leading to severe overfitting and poor out-of-sample performance. Empirical studies on forex forecasting confirm that high-capacity networks are highly sensitive to regularization and often fail to deliver consistent economic gains despite competitive statistical accuracy [DHL+20]. Thus, the real challenge lies not in expressive power, but in effective regularization, optimization, and feature engineering. This difficulty has motivated the development of structured architectures like LSTMs that incorporate inductive biases for temporal data.

2.5. The State-of-the-Art: Attention, Transformers, and the Efficiency Frontier

While LSTM networks represented a major breakthrough in modeling sequential data, the quest for more powerful and efficient architectures has continued unabated. In recent years, the deep learning community's attention has shifted dramatically towards a new class of models: Transformers. Originally developed for Natural Language Processing (NLP) tasks, where they achieved revolutionary success, Transformers have been increasingly adapted for time series forecasting. Their core innovation, the self-attention mechanism, offers a different approach to handling long-range dependencies and opens up new possibilities for parallelization. However, the application of Transformers to finance is a more nascent and nuanced field than their application in NLP or the use of LSTMs in finance. A critical review of the latest literature reveals a complex picture, where the purported superiority of Transformers is not absolute but is highly dependent on the specific task, the chosen model variant, and the evaluation criteria [ZY24; BQ25]. This lack of a settled consensus on a single "best" model creates a crucial opening for re-evaluating simpler models like FFNNs, particularly when considering metrics beyond raw predictive accuracy, such as computational efficiency and interpretability.

2.5.1. The Transformer Architecture and its Variants

The Transformer architecture, introduced by Vaswani [VSP+17], eschews the sequential, recurrent structure of RNNs and LSTMs entirely. Instead, it relies on a mechanism called self-attention. In essence, self-attention allows the model to weigh the importance of all other data points in the input sequence when processing a single data point. It can dynamically learn which parts of the sequence are most relevant to the current context, regardless of their distance. This allows Transformers to capture long-range dependencies potentially more effectively than LSTMs, which can still struggle with extremely long sequences. A key advantage of this design is that it allows for massive

parallelization. Since the processing of each data point does not depend on the completion of the previous one (as it does in an RNN), the entire sequence can be processed simultaneously, leading to significantly faster training times on hardware that supports parallel computation, such as GPUs.

However, the vanilla Transformer architecture has its own limitations when applied directly to time series forecasting. For example, its self-attention mechanism has a computational and memory complexity that scales quadratically with the length of the input sequence, making it inefficient for very long time series. This has spurred the development of a new generation of Transformer-based models specifically designed to address the challenges of time series analysis. Among the most prominent are:

- **Informer:** This variant was designed specifically to improve the efficiency of Transformers for long sequence forecasting. It introduces mechanisms like the *ProbSparse* self-attention and a distilling operation to reduce the computational complexity from quadratic to log-linear, resulting in significantly faster training and convergence speeds without a major sacrifice in performance.
- **Temporal Fusion Transformer (TFT):** The TFT is a highly complex, hybrid architecture designed for both high performance and interpretability in time series forecasting. It integrates multiple components from other successful architectures, including gated residual networks, variable selection networks to identify relevant inputs, and interpretable multi-head attention to visualize the model's focus. It is designed to handle a wide variety of inputs, including static metadata, known future inputs, and historical time series data.

The development of these specialized variants is, in itself, a crucial finding. It is a direct admission by the research community that the original Transformer model is not a perfect, out-of-the-box solution for time series forecasting. The state-of-the-art is not simply 'Transformers,' but rather a collection of 'highly-specialized, Transformer-inspired hybrid architectures.' This ever-increasing complexity further strengthens the case for maintaining and evaluating simpler, more fundamental models like FFNNs as a necessary baseline. Without such a baseline, it becomes difficult to assess what is truly gained by adding successive layers of architectural complexity and the associated computational and implementation costs.

2.5.2. A Contested Frontier: Transformer vs. LSTM Performance

Given the excitement surrounding Transformers, a key question is whether they have definitively surpassed LSTMs as the premier architecture for financial forecasting. A review of the most recent

literature reveals that the evidence is mixed, nuanced, and highly dependent on the specifics of the study. There is no simple, universal answer.

On one hand, there is strong evidence supporting the high performance of Transformer variants. A comprehensive 2024 study [ZY24] evaluated the performance of the original Transformer, Informer, and TFT on four different NZD-based exchange rate datasets. The results were clear: the TFT model was the unambiguous winner in terms of predictive accuracy, achieving the lowest RMSE and MAE and the highest R-squared (R^2) value across the board. For the NZD/USD pair, the TFT achieved an impressive R^2 of 0.9892 when augmented with the VIX volatility index as an additional input. The Informer model offered the best balance of performance and efficiency, with faster training speeds than the TFT and better accuracy than the original Transformer. This study suggests that for achieving the absolute highest accuracy, a sophisticated Transformer variant like TFT is the current state-of-the-art. Another recent study on the RMB/USD exchange rate also identified a Transformer-based model, TSMixer, as the most effective architecture for their specific prediction task. Similarly, studies focusing on high-frequency and intraday data have shown that Transformer-based models can be highly effective in volatile market conditions and, in several cases, outperform LSTMs when provided with multivariate inputs or time embeddings. However, this performance advantage typically comes with substantially higher computational complexity [FSL24].

On the other hand, there is compelling evidence that suggests LSTMs remain superior for certain crucial financial forecasting tasks. A very recent [BQ25] (February 2025 pre-print) and direct comparative analysis was conducted on high-frequency limit order book data, a domain characterized by extremely noisy, high-velocity data. The study's key finding was a critical distinction in performance based on the prediction target. While Transformer-based models showed a marginal advantage when predicting the absolute price sequence, the LSTM-based models demonstrated superior and more consistent performance when predicting differential sequences, such as price differences and directional movements. This is a profoundly important distinction for practical finance. Many, if not most, trading strategies do not depend on predicting the exact future price of an asset to the fourth decimal place. Instead, they depend on correctly predicting the direction of the next price movement (up or down). The finding that LSTMs are more reliable for this specific, and arguably more practical, task suggests that their purported obsolescence is premature.

2.5.3. Open Questions and the Need for Rigorous Benchmarking

The application of Transformers to financial forecasting is still a field with many open questions and challenges, as acknowledged by researchers working at the frontier. One major issue is the lack of reproducibility, with many papers not providing the source code or detailed implementation information needed to validate their findings. Furthermore, many comparative studies are incomplete. They often fail to compare against the full suite of state-of-the-art Transformer models (such as PatchTST, Crossformer, or iTransformer) and may not conduct entirely fair comparisons. For example, a study might apply a performance-enhancing technique like time series decomposition to their proposed LSTM model but not to the baseline Transformer model, potentially biasing the results. These methodological gaps highlight the need for more rigorous, transparent, and comprehensive benchmarking in the field, a role that a well-designed FFNN baseline is perfectly suited to fill.

Table 1: Comparison of Neural Network Architectures for Forex Time Series Forecasting.

Feature	**Feedforward Neural Network (FFNN)**	**Long Short-Term Memory (LSTM)**	**Transformer**
Temporal Data Handling	Requires manual engineering (lagged inputs)	Native sequential processing via memory gates	Global context via self-attention
Core Mechanism	Weighted sums + activation functions	Forget/input/output gates	Multi-head attention layers
Training Complexity	Low (O(n) for n samples)	High (sequential processing)	Very high ($O(n^2)$ attention)
Parallelization	Fully parallelizable	Limited to layer-level	Fully parallelizable
Interpretability	Low (black box)	Moderate (gate analysis)	High (attention maps)
Memory Efficiency	High (fixed parameters)	Moderate (cell states)	Low (large matrices)
Best For EUR/USD	Short-term patterns	Medium-term trends	Long-term dependencies
Key Advantage	Fast execution	Temporal awareness	Context modeling
Key Limitation	No time modeling	Vanishing gradients	Compute intensive

2.5.4. Synthesis, Research Gaps, and the Contemporary Relevance of FFNNs

The preceding review has traced the intellectual trajectory of financial forecasting, moving from the rigid linearity of classical econometrics to the dynamic, non-linear world of deep learning. This journey reveals a field in constant pursuit of models with greater expressive power and predictive accuracy. The EUR/USD market, with its immense scale and structural complexity, serves as the ultimate testing ground for these methodologies. The failure of linear models like ARIMA gave

rise to hybrid systems, which in turn demonstrated the indispensable value of non-linear modeling and paved the way for end-to-end neural network approaches. The Long Short-Term Memory (LSTM) network emerged as a powerful solution, providing an effective architectural framework for capturing the crucial temporal dependencies inherent in financial time series. The current state-of-the-art is represented by the Transformer architecture and its sophisticated variants, which offer a new paradigm for handling long-range dependencies but whose universal superiority over LSTMs remains a subject of active and nuanced debate.

This relentless march towards greater complexity, however, has created its own set of challenges and, critically, has left important research questions underexplored. The literature is now saturated with studies aiming to prove that a new, more complex model can achieve a marginal improvement in a specific error metric. What is conspicuously less common is research that critically evaluates the return on complexity that is, a rigorous analysis of what is truly gained by adding layers of architectural sophistication and computational cost, a critique echoed by researchers who warn against an over-reliance on complex models when simpler alternatives may suffice [MSA+22]. It is within this context that a compelling and defensible research niche for the foundational Feedforward Neural Network (FFNN) emerges. Far from being an obsolete artifact, the FFNN, when viewed through a modern lens, serves as an essential tool for promoting rigor, efficiency, and a more holistic understanding of the forecasting landscape.

2.5.5. Identifying the Research Gap: The Need for Principled Baselines

The primary research gap that FFNN-centric studies can fill is the need for a principled, powerful, and computationally efficient baseline against which more complex models can be fairly evaluated. Comparing new complex models only against other complex models risks creating an insular research field. Rigorous scientific evaluation requires strong, well-understood simpler baselines to isolate the true source of performance gains. The feedforward neural network (FFNN) is uniquely suited for this role.

First, an FFNN provides an ideal instrument for disentangling the benefits of nonlinear function approximation from architectural temporal modeling. Thanks to the Universal Approximation Theorem, an FFNN is a theoretically powerful nonlinear approximator. By supplying it with engineered temporal features (e.g., lagged prices, moving averages, momentum indicators), all temporal reasoning is embedded in the features rather than the architecture. Comparing such an FFNN against LSTM or Transformer models fed with raw time series directly answers a key question: how much of the predictive power stems from the model's ability to learn temporal patterns versus

its capacity to fit complex nonlinear functions to well-engineered features? When a well-tuned FFNN performs comparably to more sophisticated recurrent models, it indicates that feature engineering may be the dominant driver of performance. This approach also aligns with established methodological warnings on overfitting in financial markets, which emphasize rigorous out-of-sample testing, conservative model complexity, and economic significance over purely statistical metrics [DHL+20].

Second, the FFNN serves as a strong benchmark for computational and data efficiency. LSTMs and especially Transformers are parameter-heavy, data-hungry, and computationally expensive. In contrast, FFNNs are simpler, train faster, and incur lower overfitting risk on smaller datasets. Recent empirical evidence on forex data supports this: Zafeiriou and Kalles [ZK24] demonstrated that a custom feedforward architecture not only matched or exceeded LSTM performance but did so with up to 27 times less processing time.

Third, the FFNN remains highly relevant as a modular component in modern hybrid systems. Following the success of early ARIMA-ANN hybrids, an FFNN can act as a flexible nonlinear engine within multi-stage frameworks for instance, modeling residuals from a GARCH volatility forecast or learning optimal nonlinear weights in an ensemble of LSTM, Transformer, and statistical models. In this role, the FFNN functions not as a standalone competitor but as a versatile building block for more robust forecasting systems.

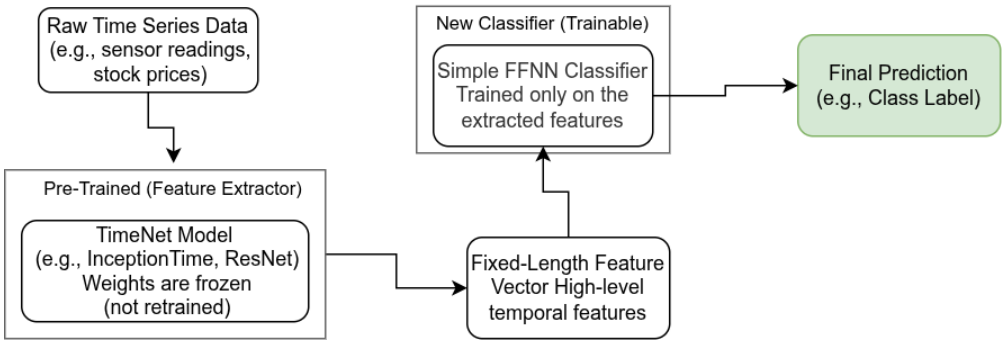

Figure 4: TimeNet-based transfer learning workflow.

Finally, an under-explored avenue of research is the combination of FFNNs with transfer learning [MVV+17]. The concept of transfer learning involves pre-training a deep model on a large, general dataset and then fine-tuning it for a specific task, which is particularly useful when labeled data for the target task is scarce. While this is often done with complex architectures, a highly efficient alternative exists. One could use a powerful, pre-trained feature extractor for time series, such as TimeNet [MVV+17], a deep RNN pre-trained on a diverse archive of time series

data to learn a generic representation. This pre-trained model can be used as an "off-the-shelf" tool to convert a raw time series into a rich, fixed-length feature vector. This vector, which now encodes complex temporal characteristics, can then be fed into a simple, fast, and easy-to-train FFNN for the final prediction task.

2.5.6. Final Reflections

This review has navigated the evolutionary path of EUR/USD forecasting, from the demonstrably limited linear models of the past to the complex deep learning architectures that define the present. The table 2 below provides a quantitative summary of the performance reported in recent literature, illustrating the general hierarchy where more complex, data-rich models tend to achieve lower error metrics.

Table 2: Reported Performance of Forecasting Models for EUR/USD.

Model	Features	Performance	Source
ARIMA	Closing prices only	MAE: 0.0164, MSE: 0.00046	[SGO19]
LSTM (Basic)	Closing prices only	MAE: 0.0179, MSE: 0.00033	[SGO19]
Dual-Input LSTM	Prices, Economic fundamentals	24% MAE reduction vs baseline	[SGO19]
Bi-LSTM (Optuna)	Multi-source data fusion	10.7% MAE improvement	[SGO19]

While the table 2 shows a clear trend towards superior performance with models like LSTMs and TFTs, especially when augmented with diverse data, it does not tell the whole story. The dominance of these complex models is nuanced and comes with significant costs in terms of computational resources, implementation complexity, and data requirements. The pursuit of marginal gains in accuracy has overshadowed other critical dimensions of model evaluation.

This research concludes that financial forecasting has become a ‘relentless march towards greater complexity,’ where models like LSTMs and Transformers are pursued for marginal accuracy gains without a critical analysis of their high computational and data costs. This creates a significant research gap: a failure to evaluate the practical ‘return on complexity’.

In response, it is mentioned that the foundational Feedforward Neural Network (FFNN) is not obsolete but remains an essential tool for more rigorous and efficient analysis. The FFNN serves as the perfect principled benchmark, allow to determine if performance gains come from a model’s sophisticated architecture or from intelligent feature engineering. Due to its architectural simplicity, the FFNN is significantly more efficient, offering faster training and prediction times,

which are critical for the viability of real-world trading systems. Ultimately, the goal is not to declare a single 'best' model but to re-establish that the FFNN can perform better. This encourages a more mature approach to forecasting that balances the quest for accuracy with the practical realities.

3. Methodology

This research introduces a dual approach to predicting the EUR/USD exchange rate, combining a pattern-matching method with a standard feed-forward neural network. The core strategy involves breaking the forecasting challenge down into two specific phases. First, the system identifies similar historical market situations to provide a stable baseline, acting like a memory bank. Secondly, the neural network analyses the complex differences to understand how the current market might deviate from those past patterns. This allows for a balance between historical reliability and modern adaptability. Currently, many financial AI projects involve feeding raw, disorganized data into complex systems. This often causes neural network models to fail because they memorize random noise instead of recognizing real, reliable patterns. It should be noted that successful models almost always require data to be pre-processed and denoised rather than using raw inputs [SGO19]. To avoid this issue, this work takes a step-by-step approach.

The decision to adopt this approach originates from the limitations of traditional models that were mentioned in the Literature Review. As [ZPH98] demonstrated, financial data contains a combination of simple, predictable trends and complex, unpredictable behaviors. Although standard statistical models such as ARIMA are useful, they are often deficient for handling the unstable nature of modern financial markets as they frequently fail to account for sudden shifts or periods of high instability [BJR08]. However, recently, the field has become increasingly complex. The introduction of Long Short-Term Memory (LSTM) networks [HS97] and Transformer [VSP+17] architectures has shifted the focus of research towards capturing long-term connections in data. Despite the power of these models, it is argued that this increased complexity is not always beneficial in fast-paced currency markets. They observe that due to the high level of random noise in financial data, these sophisticated systems often struggle to function effectively, instead memorizing errors rather than learning reliable patterns [ZK24; MSA+22].

Therefore, this methodology prioritizes an efficient approach, as recommended, emphasize the practical benefits of low-latency ANNs. They demonstrate that simpler, optimized networks often outperform heavier than more complex architectures in live trading scenarios, where speed and adaptability are essential [ZK24].

3.1. Analysis of Time-Series

This work begins with a careful preparation step involving the use of Euclidean distance prior to any neural network training. The aim is to demonstrate the usefulness of the data by comparing random market scenarios with real historical records to identify meaningful matches. During this phase, the neural network (FFNN) is not training but rather testing the data to ensure that only high-quality patterns are retained. This guarantees that the feed-forward neural network (FFNN) is built on the best possible foundation of information to model complex market relationships, using standard tools such as Python and NumPy for all calculations.

3.2. Data Acquisition

3.2.1. Dataset

The dataset consists of daily closing prices for the EUR/USD currency pair, sourced from historical financial records spanning December 2003 to December 2025 [Yah25; Ran25]. The dataset has approximately 5,730 day's data, comes with raw price data (Open, High, Low, Close). All others data has been cleaned and just kept 'Date' and 'Close'.

3.2.2. Data Preparation

Raw price levels (such as '1.1500') cannot be used directly for distance calculations. To ensure the Euclidean distance metric works correctly, the closing price data was used to get percentage returns and kept in a new column named 'difference' by this formula :

$$difference(\%) = \left(\frac{close_t - close_{t-1}}{close_{t-1}} \right) \times 100 \tag{3.1}$$

where $close_t$ is the closing price on day t.

Table 3: Sample of EUR/USD Closing Prices and % Differences (December 2003).

Date	Close	Difference (%)
2003-12-01	1.19650137424469	
2003-12-02	1.20889747142792	1.036028662404
2003-12-03	1.21229755878448	0.281255229407
2003-12-04	1.20809423923492	-0.34672341944
2003-12-05	1.21869468688965	0.877452048893
2003-12-08	1.22200095653534	0.271295976035
2003-12-09	1.22499477863312	0.244993433251
2003-12-10	1.21909594535828	-0.48153946268

This transformation serves an important theoretical purpose. Raw closing prices are non-

stationary and potentially discontinuous, which would violate the assumptions required for function approximation. By converting to daily percentage returns, the input feature vectors are bounded within a realistic range of approximately $\pm 2\%$, forming a compact subset of $\mathbb{R}^5$. This bounded and more stationary representation makes the input space more suitable for the continuity assumptions underlying the theoretical framework used in this study.

3.3. The Euclidean Distance

3.3.1. The 'Main Data-Table'

This process produces a timeline of daily percentage changes, which captures both market volatility and direction. A sliding window method was then used to organize the data into specific groups. Each group consists of six numbers: the first five represent the price changes over five days (feature vector $\boldsymbol{V}$), while the sixth number is the true value the next day, which is called the target (v_{target}), showing the price change on the following day. This collection, known as the 'Main Data-table,' contains approximately 5,730 of these patterns derived from the full 2003 to 2025 dataset. A sample is provided below (values are truncated):

Table 4: Vector values and true value next day (We call it the 'Main Data-table').

$$[1.036, 0.281, -0.347, 0.877, 0.271, 0.245]$$
$$[0.281, -0.347, 0.877, 0.271, 0.245, -0.481]$$
$$\dots$$
$$\dots$$
$$[-0.015, 0.144, -0.044, -0.061, -0.142, 0.274]$$

These data points represent five-day market trends paired with the outcome of the following day. This structure allows the system to identify similar patterns from the past to help predict future movements.

3.3.2. Generate Base Vectors

To simulate a wide variety of possible five-day market movements, 50 random base vectors (labeled as $\boldsymbol{B}$) were generated for one time. Each vector consists of five numbers randomly selected between -2.0 and 2.0. This range was chosen to reflect realistic daily price changes for the EUR/USD pair, which typically stay within $\pm 2\%$ but can sometimes show larger swings. These vectors function as pattern matching. They serve as tools designed to search the historical data for similar real-world patterns. Table 5 represents the first 4 base vectors out of 50.

Table 5: First four base vectors.

Vector	Values
$\mathbf{B_1}$	$[-1.010289286111500484, 1.833463084878135696, 0.8586831693146397271, 1.243189917255115517, -1.449313434074988027]$
$\mathbf{B_2}$	$[0.8428329116797472764, 1.527002299871566660, -1.251695816897025360, 0.5282498201246568215, 0.1509024899660227348]$
$\mathbf{B_3}$	$[-1.455915417154833413, -1.558242332923744033, -0.9607966667081542234, 0.4276299847022344558, -1.925764397485453028]$
$\mathbf{B_4}$	$[1.037132907656614922, -1.981972905967844145, -1.027229908064017039, -0.871805917273560501, -1.868143968475285632]$

3.3.3. The Threshold (τ) as a Filter

The threshold value (τ) acts as a sensitivity filter for the pattern-matching process. It determines how strictly historical data must match the generated base patterns. The threshold was adjusted from 0.5 to 1.5 in steps of 0.1 to control which matches were accepted. Lower values create stricter rules, resulting in fewer, but highly similar patterns. These close matches often reveal a clear trend for the next day's price change, showing a strong tendency toward either positive or negative results.

3.3.4. Data Selection Phases

The 'Main Data-table' has been created from the financial time series data from December 2003 to December 2025, is separated into two phases.

3.3.4.1. Learning Phase

It is almost 20 years of data from December 2003 to December 2023 has been used to calculate the Euclidean distance based on one time generated base vectors (***B***) and threshold values ***T*** = 0.5 to 1.5. Each threshold value is represented by a symbol called 'tau' (τ). The selected τ and ***B*** has been finalized based on considerable accepted number of ***L*** set, and percentage of positive and negative outcomes. This phase produced a large number of results.

3.3.4.2. Testing Phase

A specific threshold value τ and a base vector (***B***) from the learning phase result has been selected to calculate the Euclidean distance for the rest of the data from January 2024 to December 2025. The selected τ and ***B*** has been finalized based on considerable accepted number of ***L*** set, and

percentage of positive and negative outcomes.

3.3.5. Euclidean Distance Calculation

Euclidean distance was chosen because it is simple and effective. It provides a straightforward method of measuring the similarity of 5-day market patterns without the need for complex assumptions. This makes it ideal for identifying repeating trends within unpredictable financial data.

Unlike more complicated methods, this metric can be calculated quickly and interpreted easily, which aligns with the research goal of creating clear models for fundamental concepts such as the feed-forward neural network (FFNN). This preparation step identifies the most useful patterns to input into the AI for forecasting, effectively bridging the gap between traditional analysis and neural networks. Finally, the threshold acts as a control switch, balancing the need for exact matches (high precision) with the need for sufficient data. The formula for n-dimensional Euclidean distance for this work is:

$$d(\boldsymbol{B}, \boldsymbol{V}) = \sqrt{\sum_{i=1}^{n} (V_i - B_i)^2} \tag{3.2}$$

where:

$d(\boldsymbol{B}, \boldsymbol{V})$	is the Euclidean distance between vectors $\boldsymbol{B}$ and $\boldsymbol{V}$.
$\boldsymbol{B}, \boldsymbol{V}$	are the Base vectors and Feature vectors (first 5 data points from each vector mentioned in Table 4).
B_i, V_i	represent the individual data points in the vectors.
n	is the total number of dimensions in the vector space.

Calculation Algorithm

Let,
$\boldsymbol{D}$: "Main Data-table", feature vectors $\boldsymbol{V}$ and target values v_{target}.

N: Number of base vectors ($\boldsymbol{B}$) to generate.

$\boldsymbol{R}_{\text{range}}$: Range for random generation $[-2.0, 2.0]$.

$\boldsymbol{T}$: Set of threshold values $\{0.5, 0.6, \ldots, 1.5\}$. Step $= 0.1$.

$\boldsymbol{B}_{\text{set}}$ size $=$ 50 vectors

1. Generate multiple (N) vectors $\boldsymbol{B}$ of size 5 with uniform distribution from $\boldsymbol{R}_{\text{range}}$, we call it $\boldsymbol{B}_{\text{set}}$.
2. **For** each threshold $\boldsymbol{\tau}$ in $\boldsymbol{T}$:
    - **For** each base vector $\boldsymbol{B}$ in $\boldsymbol{B}_{\text{set}}$:
        - Initialize an empty result set $\boldsymbol{L}$
        - **For** each row $\boldsymbol{r}$ in data-table $\boldsymbol{D}$:
            - $\boldsymbol{V}$ = first 5 numbers of row $\boldsymbol{r}$
            - Calculate Euclidean distance $d(\boldsymbol{B}, \boldsymbol{V})$
            - **If** $d(\boldsymbol{B}, \boldsymbol{V}) < \boldsymbol{\tau}$:
                - Append $\boldsymbol{r}$ to $\boldsymbol{L}$, where $\boldsymbol{r}$ contains $\boldsymbol{V}$ and v_{target}.
            - **End If**
        - **End For**
        - **If** $\boldsymbol{L}$ set not empty:
            - Compute statistics for $\boldsymbol{L}$ (% of positive & negative v_{target})
            - Export results to a formatted .txt file
        - **End If**
    - **End For**
- **End For**

The results are saved in a .txt file following the format:

```
L set:
[first row of L]
[Second row of L]
['N' row of L]
B vector:
[B vector values]
tau value:
[tau value]
% of positive vtarget:
[percentage]
% of negative vtarget:
[percentage]


----------------------------------------
```

The full implementation of the Euclidean distance calculation, including the base vector generation and threshold filtering algorithm, is publicly available [Sha26a].

When a specific base vector ($\boldsymbol{B}$) and threshold (τ) produce a set of historical matches ($\boldsymbol{L}$), the value of $\boldsymbol{B}$ and τ are determined by the ratio of positive to negative outcomes:

A 50/50 split: is the least useful outcome, as it indicates that the results are 50% positive and 50% negative. It suggests that the pattern is random, similar to flipping a coin, and provides no predictive value.

A positive classifier: (more than 50% positive / less than 50% negative) indicates a predictive edge. It implies that this specific pattern and threshold can be used to predict a price increase with greater accuracy than would be expected by chance. The higher the percentage exceeds 50%, the stronger the buy signal.

A negative classifier: (more than 50% negative / less than 50% positive) is equally valuable as a positive classifier. It implies that the pattern can be used to predict a price decrease with greater than chance accuracy. A result significantly over 50% suggests a strong sell signal.

3.4. Results: Euclidean distance Analysis

3.4.1. Learning Phase

The experiment tested threshold values ranging from 0.5 to 1.5 with 50 ($\boldsymbol{B}_{set}$) base vectors ($\boldsymbol{B}$). As expected, there was a direct link between the threshold level (τ), base vector ($\boldsymbol{B}$) and the amount of data accepted. At the maximum level of τ= 1.5, the system captured the highest volume of data resulting in 2,862 matches for the base vector $\boldsymbol{B}$= [0.6673177879279355, -0.399743965085662, 0.1166337339292971, -0.4774067929209176, -0.38544785593128283] but the outcomes are not promising. It gives 50.98% positive and 49.02% negative. This slightly shows the positive classifier which indicates that the price maybe increase.

As the threshold was gradually lowered to 1.4 and 1.3, the count declined steadily, dropping by approximately 400 to 500 matches for every 0.1 adjustment.

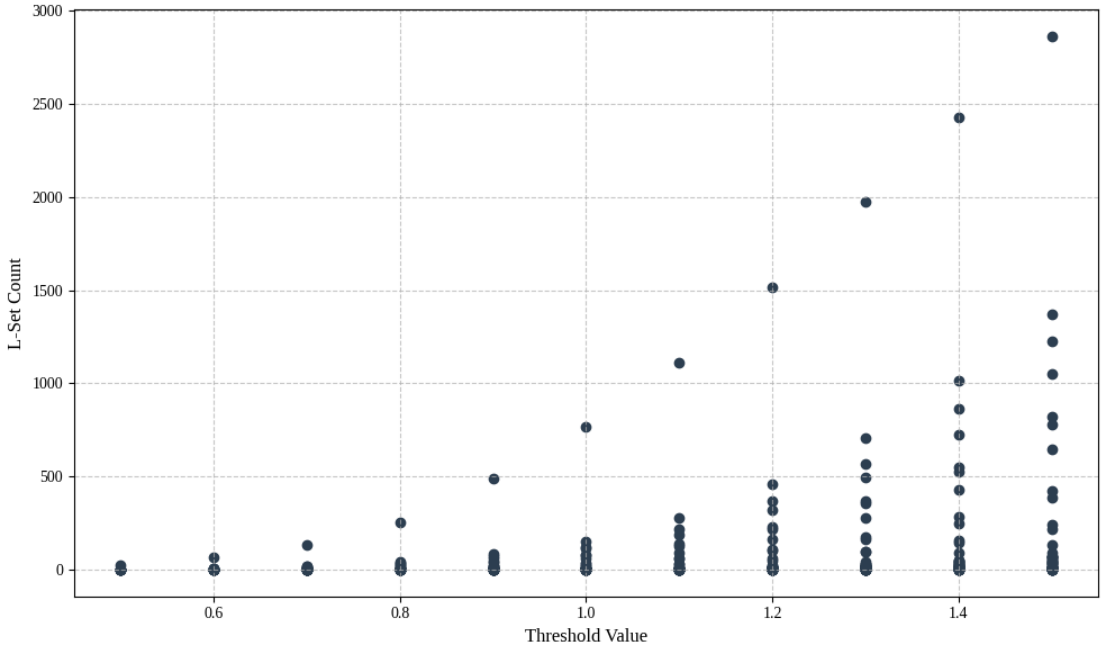

Figure 5: Effect of threshold with specific base vectors on $\boldsymbol{L}$ set count.

However, the results do not only depend on the specific threshold value (τ). They mostly depend on the combination of a specific threshold value (τ) and a specific base vector ($\boldsymbol{B}$). It is also not the threshold value (τ) that accepts the largest $\boldsymbol{L}$ set. The observation and selection criteria are based on the base vector ($\boldsymbol{B}$), the threshold value (τ) and the outcomes (percentage of positive and negative), which accept a considerable number of the $\boldsymbol{L}$ set.

3.4.2. Testing Phase

While the learning phase uses a large amount of data, the testing phase uses a small amount of data from the recent years January 2024 to December 2025. Base vector ($\boldsymbol{B}$) and threshold value (τ) was selected as compromise solution from the learning phase's results based on a large number of accepted $\boldsymbol{L}$ sets and outcomes of more than 50% in either the positive or negative case.

Table 6: Finalized base vector $\boldsymbol{B}$ and threshold value $\boldsymbol{\tau}$.

$$\boldsymbol{B} = [1.6085788331083064, -0.0518004411331483, -0.6413941226817417, -0.8585278694831779, 0.6172863137371993]$$

threshold value $\boldsymbol{\tau} = 1.4$

After observing the calculation with these $\boldsymbol{B}$, and τ, a base vector and a threshold value τ have been finalized which will be used for finding the future testing sets of the feed-forward neural network. The mentioned base vector $\boldsymbol{B}$, $\boldsymbol{\tau}$ have accepted 145 $\boldsymbol{L}$ set. Its outcomes also considerable

as it gives 53.1% positive and 46.9% negative outcomes in the learning phase experiment results, and gives 54.55% positive and 45.45% negative outcomes in the testing phase experiment results, though the accepted L set from the recent data is a few. It is an 11 $\boldsymbol{L}$ set size.

3.5. Future Integration with FFNN

3.5.1. Theoretical Foundation

The theoretical foundation for the use of feedforward neural networks in financial forecasting was established in the literature review. A single-hidden-layer FFNN with a non-linear activation function is theoretically capable of approximating any continuous function, which justifies its application to the complex, non-linear patterns found in financial time series.

3.5.2. FFNN Architecture Details

The feed-forward neural network is designed as a multi-layer perceptron (MLP). The multi-layer perceptron is the foundational classic neural network. It consists of an input layer, at least one hidden layer with non-linear activation function, and output layer. Hence, the architecture is like *input* $\rightarrow$ *hidden layer (finite number of neurons)* $\rightarrow$ *output layer* which can solve non-linear problems by adding hidden layer with an activation function like sigmoid. This architecture is theoretically capable of approximating any continuous function, as established in the literature review, provided it uses a non-linear activation function.

The specific parameters for the FFNN, including layer configurations and training details, will be determined in future work. Adjustments to the interaction between these parameters may be made based on empirical testing.

The use of a single hidden layer was based on the theorem. It was also necessary because the training data was limited. Euclidean distance filtering reduced the dataset to 115 training samples. A deeper architecture would have too many parameters for this small dataset. This would make overfitting very likely. Therefore, the simplest architecture was the most appropriate choice. This included a single hidden layer with a limited number of neurons.

Input Layer will be corresponding to one element of the five-day feature vector $\boldsymbol{V}$ = [V1, V2, V3, V4, V5], representing the percentage changes in EUR/USD closing prices over the past five days. It serves solely to pass the input data to the hidden layer.

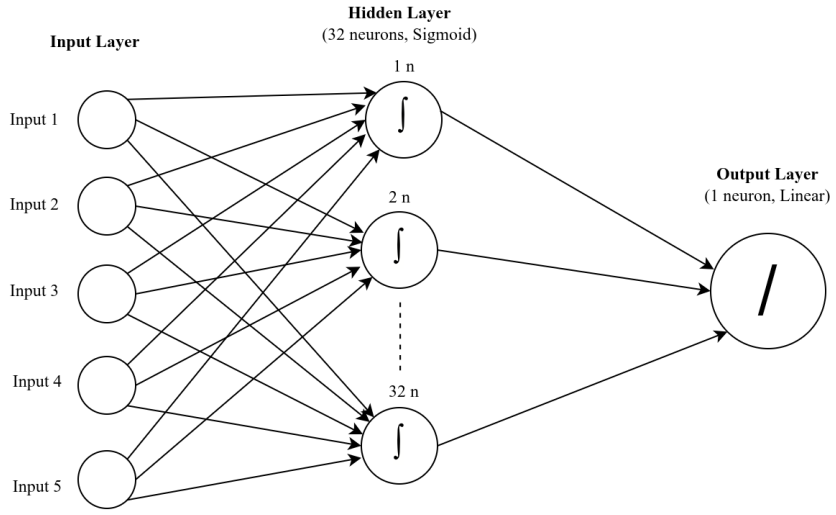

Figure 6: A graphical representation of FFNN.

Hidden Layer will use the sigmoid activation function, defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. The sigmoid function introduces nonlinearity, enabling the network to capture complex patterns and relationships within the filtered historical data ($\boldsymbol{L}$ set). The neuron size for this layer will decide in future work

Output Layer will be configured with a single neuron using a linear activation function (identity function, $f(z) = z$). This setup will be designed to produce a continuous output, allowing the model to predict the specific magnitude of the next day's percentage change ($\boldsymbol{V}_{target}$) in the EUR/USD exchange rate.

3.5.3. A Pattern Specific Training

Standard neural networks in finance often fail because they attempt to model the entire history of the market, which contains a high level of noise. Rather than training the FFNN on the entire dataset ($\boldsymbol{D}$), the network will be trained exclusively on the $\boldsymbol{L}$ set which is the subset of historical vectors identified by the Euclidean distance metric as being mathematically similar to the finalized base vector ($\boldsymbol{B}$). By limiting the training data to this pre-filtered $\boldsymbol{L}$ set, the FFNN does not need to learn 'general' market behavior. Its sole objective is to learn the specific, nonlinear variations of the single market pattern identified in section 3.4.

3.5.4. Data Splitting for the FFNN

To ensure robust model evaluation and prevent over-fitting, the data is split into three distinct sets: training, validation, and test sets. This follows standard machine learning practices, where the training set is used to fit the model, the validation set for hyper-parameter tuning and early

stopping, and the test set for final, unbiased performance assessment.

Training set: Derived from the $\boldsymbol{L}$ set accepted during the learning phase (December 2003 to December 2023). This consists of approximately 145 matched vectors identified via Euclidean distance. To create the training and validation splits, 115 $\boldsymbol{L}$ set will be allocated to training. The training process uses these samples to adjust the network's weights through back-propagation.

Validation set: The remaining 30 $\boldsymbol{L}$ set of the learning phase will serve as the validation set. This subset remains unseen during the initial weight updates and is used to monitor performance during training and implement early stopping to avoid over-fitting.

Test set: The $\boldsymbol{L}$ set generated during the testing phase (January 2024 to December 2025) will be used as test set. This consists of 11 matched vectors from recent data, providing a realistic evaluation of the model's generalization to unseen market conditions.

The investigation revealed a clear trade-off between the quantity of data collected and the accuracy of the patterns identified. While the threshold τ= 1.5 captured a large amount of data, their predictive ability was low with an accuracy rate of around 50%. By contrast, threshold τ= 1.4 identified a stable pattern that remained consistent during both the learning and unseen testing phases. These consistent results confirm that specific, non-random patterns exist within market noise that can be identified.

The next step is to implement the feed-forward neural network (FFNN). The filtered $\boldsymbol{L}$ set identified in this study will be used to train the neural network (FFNN). The aim is to transform the neural network from a general predictor into a good predicting model by limiting the input to these pre-validated patterns. Furthermore, future research will seek more diverse base patterns to expand the range of trading opportunities while maintaining the statistical advantages identified in this study.

4. Experimental Results and Analysis

This chapter details the results of applying feedforward neural networks (FFNNs) to predict the direction of EUR/USD price movements. The experiments follow a step-by-step process. First, a classical sigmoid activation function is used to create a baseline based on Cybenko's Universal Approximation Theorem [Cyb89]. Second, the ReLU activation function is tested to examine its practical benefits. Finally, a consensus ensemble approach is used, combining ReLU-based FFNN median predictions with a K-Nearest Neighbours (KNN) regressor. The main evaluation metric is directional accuracy, which is the percentage of test samples where the predicted price direction matches the actual direction. A prediction is marked as 1 if correct and 0 if incorrect.

The dataset and filtering methods were explained in Chapter 3. Daily closing prices for EUR/USD from December 2003 to December 2025 were converted to daily percentage returns, and a five-day sliding window was used to create feature-target pairs. A Euclidean distance filter, using a base vector $\boldsymbol{B}$ and a threshold $\tau = 1.4$, identified 145 similar patterns from the learning phase (December 2003 to December 2023). From these, 115 were used for training and 30 for validation. The testing phase from January 2024 to December 2025 produced 11 matched patterns for the test set. Features were standardized using a StandardScaler fitted only on the training data and then applied to the validation and test sets.

4.1. FFNN Experimental Design

The FFNN architecture includes an input layer of five neurons, one hidden layer with a specific activation function, and one output neuron with a linear activation function. This design follows the single-hidden-layer structure established in the literature review. The experiments start with the following:

FFNN with Sigmoid Activation: A standard single-run FFNN using a Sigmoid activation function is used to set a baseline performance level.

FFNN with ReLU Activation: The activation function is replaced with a Rectified Linear Unit (ReLU) to see how it affects convergence speed, gradient preservation, and predictive accuracy using the same hyperparameters.

FFNN with ReLU over 100 seed analysis: To reduce the noise caused by random weight initialization, 100 independent training runs (seeds 0 to 99) are performed, and the median

predictions are calculated.

FFNN and KNN-Ensemble Consensus: The classification strength of K-Nearest Neighbors (KNN) is combined with the 100-seed ReLU-FFNN median model to create a consensus-based ensemble.

Table 7: Architecture Configuration and Goal of the Network.

Layer	**Configuration / Description**	**Activation Function**
Input	Takes 5 input features (past values)	None
Hidden	Processes non-linear transformations	Sigmoid / ReLU
Output	Consists of 1 neuron producing predicted value	Linear
Common Goal: Predict the direction of the currency.		

The full implementation of all neural network experiments, including the sigmoid, ReLU, and KNN ensemble configurations, is publicly available [Sha26b].

4.2. Baseline FFNN with Sigmoid

The first set of experiments uses the sigmoid activation function in the hidden layer. The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ provides a smooth, bounded nonlinearity and serves as the classical baseline activation for single-hidden-layer networks. This experiment tests whether a theoretically grounded architecture can produce useful predictions on a small and filtered financial dataset.

4.2.1. Training and Evaluation

To show how the sigmoid model behaves in a reproducible setup, a single run was performed using a fixed random seed of 10, 16 hidden neurons, a learning rate of 0.00183, a batch size of 8, and a maximum of 30 training epochs. Early stopping was used with a patience of 5 epochs, based on validation directional accuracy. The model used the Adam optimizer and Mean Squared Error (MSE) as the loss function.

After training, the model evaluated on validation directional accuracy showed little and unstable improvement. The highest validation accuracy was 54.17% at Epoch 1, after which performance dropped steadily. Early stopping occurred after Epoch 6 when validation accuracy fell to 44.62%, and the best weights from Epoch 1 were restored. The tendency of the model to peak early and then decline indicates an interaction between the saturation behavior of the sigmoid function and the small training set. As neurons reach the tails of the sigmoid function, gradients decrease,

which slows learning and makes the model sensitive to the random initialization of the seed.

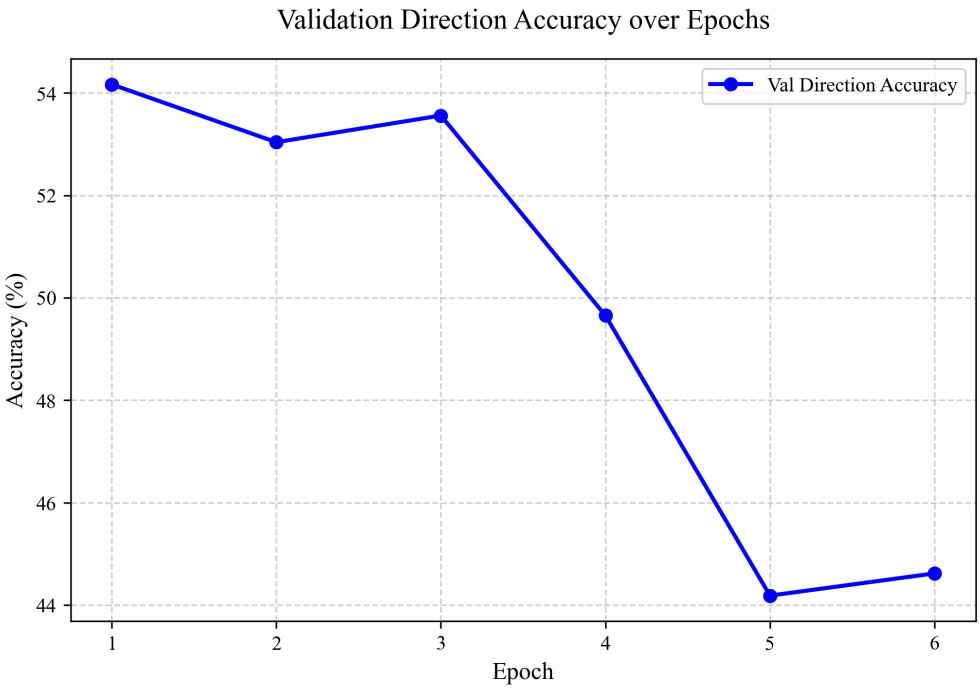

Figure 7: Sigmoid evaluated validation direction accuracy over epochs.

The best saved model was then evaluated on the validation set of 30 samples, where it achieved a directional accuracy of 56.67%. An inspection of the validation predictions revealed a consistent bias: the model predicted negative values for all 30 samples. This allowed the model to correctly identify the 17 negative cases in the validation set, but it failed to identify any of the 13 positive cases. This pattern shows that the model settled on a biased negative result instead of learning a balanced function for directional prediction.

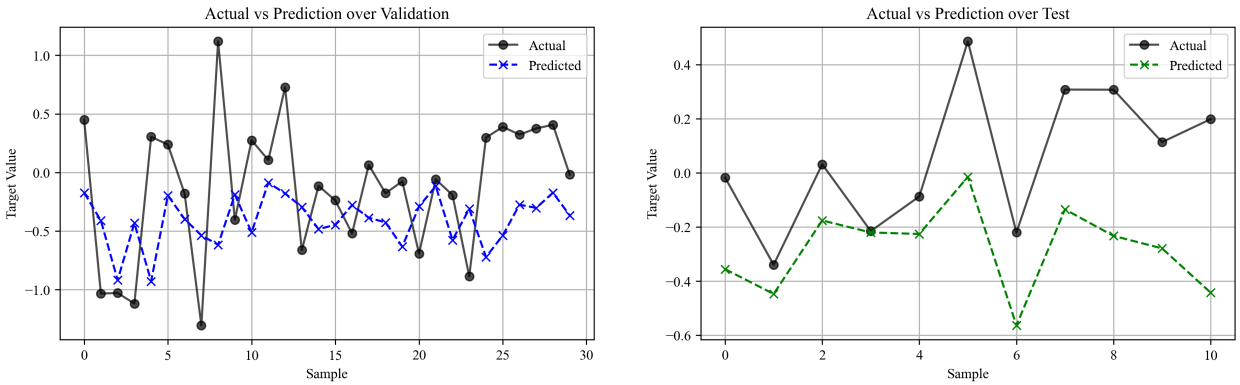

Figure 8: FFNN with sigmoid, Actual vs Prediction on validation and test.

On the test set of 11 samples from January 2024 to December 2025, the sigmoid model achieved a directional accuracy of 45.45%. The test predictions again showed a strong negative bias, with the model predicting negative values for almost every sample. The test set contained

Table 8: Sigmoid FFNN performance summary (Seed 10, 16 neurons, $lr = 0.00183$).

Metric	**Validation Set**	**Test Set (2024-2025)**
Directional Accuracy	56.67%	45.45%
Correct Positive Predictions	0 / 13	1 / 7
Correct Negative Predictions	17 / 17	4 / 4
Test MSE	—	0.1461

4 negative and 7 positive outcomes. The model correctly classified all 4 negatives but only 1 of the 7 positives, resulting in accuracy below the 50% random chance baseline. The mean squared error on the test set was 0.1461. These results suggest that while the sigmoid model captured a directional trend from the training data, it failed to correctly predict both directions on unseen data.

4.2.2. Median: Sigmoid over 0 to 99 seeds

To create a more reliable assessment of the sigmoid architecture, the experiment was repeated using 100 independent random seeds (seeds 0 to 99). For each seed, a new model was initialized with 6 hidden neurons, a learning rate of 0.00379, a batch size of 8, and a maximum of 30 epochs using the same early stopping rules. Each model was trained independently, the best version was saved, and predictions for the test set were collected. These 100 sets of predictions were combined by calculating the median for each of the 11 test samples.

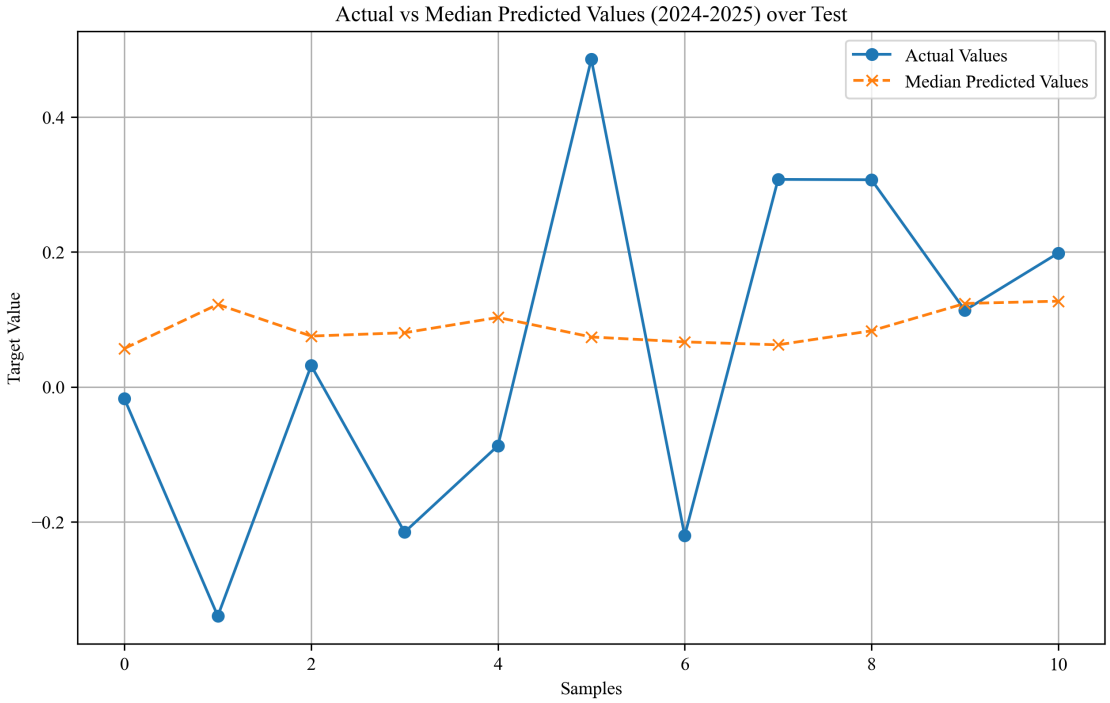

Figure 9: Sigmoid Actual vs Median Predicted values over Test.

The median was used instead of the mean because the median is less affected by extreme predictions from individual seeds where a model might have settled on an extreme value. This provides a more stable estimate of the overall directional signal. Using the median also helped reduce the strong negative bias seen in the single seed experiment. However, the final result showed a different systematic bias: the median of 100 sigmoid models predicted only positive values for all 11 test samples, with values ranging from approximately 0.057 to 0.128. Across all 100 seeds, the sigmoid model ranged from a minimum of 27.27% to a maximum of 81.82%, indicating that the architecture is capable of stronger predictions than the median alone suggests.

The final median directional accuracy on the test set was 54.55%, with 6 correct predictions out of 11. Out of the 4 actual negative outcomes in the test set, none were correctly identified, while 6 of the 7 positive outcomes were correct. This result is slightly above the random baseline of 50%, but the failure to predict any negative outcomes suggests that the sigmoid ensemble, when combined using the median, develops a consistent positive bias. This matches the known saturation behavior of sigmoid neurons, which tend to group outputs near the center of their range when working with small datasets. When the learning phase has a slight positive imbalance (53.1% positive outcomes), the median of these outputs converges to a stable positive value.

Table 9: Sigmoid FFNN median performance (seeds 0-99).

Metric	**Value**
Number of seeds	100 (seeds 0-99)
Hidden neurons	6
Learning rate	0.00379
Median directional accuracy (test)	54.55%
Correct positive predictions	6 / 7
Maximum directional accuracy (test)	81.82%
Minimum directional accuracy (test)	27.27%
Correct negative predictions	0 / 4

Together, the sigmoid experiments establish two important baselines. First, individual sigmoid runs with specific hyperparameters can achieve moderate accuracy on the validation set, but this does not consistently transfer to the test set. In the single seed case, performance dropped below the 50% baseline. Second, using the median of 100 seeds produces a slightly more stable signal (54.55%) but introduces a consistent positive bias that removes all negative predictions. These limitations show why the ReLU activation function is used in the next experiment.

4.3. FFNN with ReLU Activation

The Rectified Linear Unit (ReLU) activation function, defined as $f(z) = max(0, z)$, is widely used in modern neural networks. Unlike the sigmoid function, ReLU does not saturate for positive inputs. This reduces the vanishing gradient problem and generally allows for faster and more stable convergence. While sigmoid was used as the classical baseline, ReLU has since been shown to possess equivalent approximation properties through various theoretical studies. As established in the literature review, ReLU is a theoretically grounded activation function and a more practical choice than sigmoid for this task.

This research shows that ReLU networks are theoretically similar to sigmoid networks in their approximation capacity, but they are more efficient in practice. The following experiments test whether this theoretical advantage leads to better directional accuracy on the EUR/USD test set.

4.3.1. Training and Evaluation

The activation function in the hidden layer was changed to ReLU while keeping all other hyper-parameters identical to the sigmoid run (seed 10, 16 neurons, learning rate 0.00183, batch size 8, and 30 maximum epochs). This approach allows for a direct comparison of the two activation functions under the same experimental conditions.

The training patterns for the ReLU model were quite different from those of the sigmoid model. The highest validation directional accuracy was 57.8125% at Epoch 1. After this, accuracy decreased and the model did not improve, which triggered early stopping at Epoch 6. Although the peak validation accuracy was slighlty improvement over the sigmoid case (57.81% compared to 54.55%), the final evaluation on the validation set showed a different result. The ReLU model's prediction reached a validation directional accuracy of 70.00% across the 30 samples.

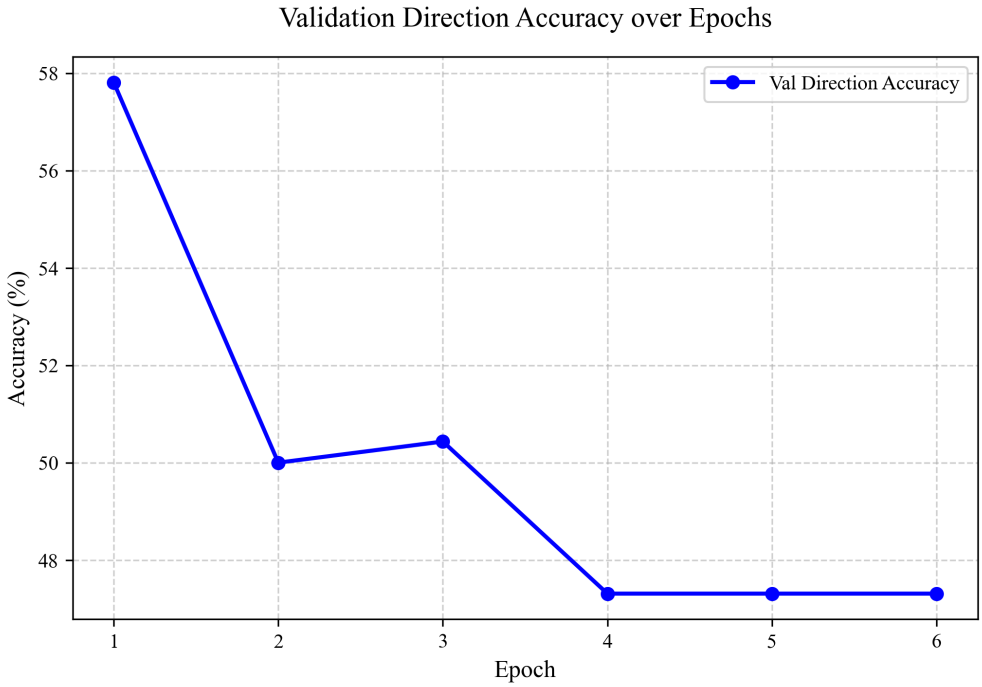

Figure 10: ReLU evaluated validation direction accuracy over epochs.

Unlike the sigmoid model, the ReLU model predicted both positive and negative directions correctly. An inspection of the 30 validation predictions shows that the model correctly predicted the direction for 21 samples. This included several large positive movements, such as an actual value of 1.118788 predicted as 0.493624, and large negative movements, such as an actual value of -1.028040 predicted as -2.512747.

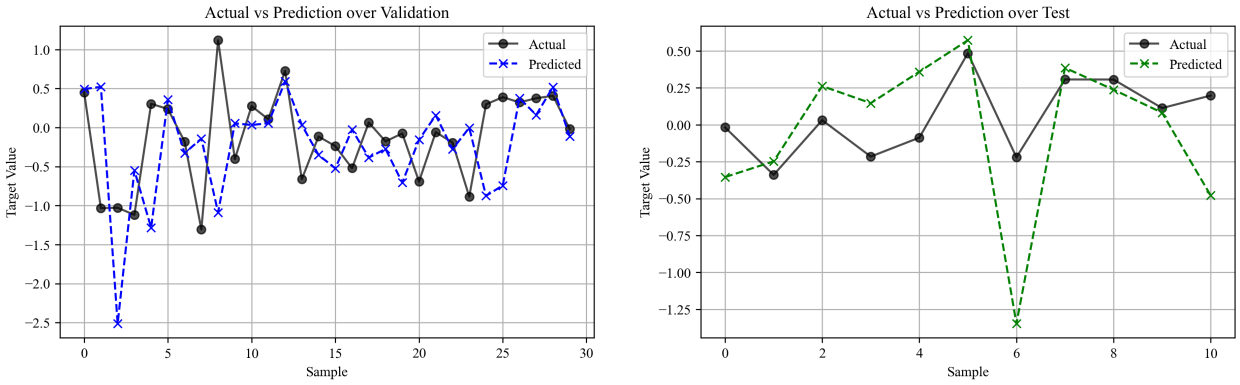

Figure 11: FFNN with ReLU, Actual vs Prediction on validation and test.

On the test set, the ReLU model achieved a directional accuracy of 72.73%, correctly classifying 8 out of 11 samples. This is a significant improvement over the sigmoid model's test accuracy of 45.45% using the same hyperparameters. The ReLU model correctly identified 4 of the 7 positive outcomes and 4 of the 4 negative outcomes, showing more balanced predictions compared to the biased sigmoid results. The test MSE was 0.2048, which is higher than the sigmoid's 0.1461. This indicates that while the ReLU model predicts direction more accurately, the magnitude of

its predictions can be further from the actual values. This trade-off is expected because MSE penalizes large prediction errors regardless of direction, whereas directional accuracy, the primary objective of this study, rewards the correct sign even when the magnitude is off.

Table 10: ReLU FFNN performance summary (Seed 10, 16 neurons, $lr = 0.00183$).

Metric	**Validation Set**	**Test Set (2024-2025)**
Directional Accuracy	70%	72.73%
Correct Positive Predictions	$8 \ / \ 13$	$4 \ / \ 7$
Correct Negative Predictions	$13 \ / \ 17$	$4 \ / \ 4$
Test MSE	—	0.2048

4.3.2. Median: ReLU over 0 to 99 seeds

Similar to the sigmoid median experiment, the ReLU model was trained using 100 independent seeds (seeds 0 to 99) with 6 hidden neurons, a learning rate of 0.00379, a batch size of 8, and up to 30 epochs with early stopping. Predictions from the best version of each of the 100 models were collected, and the median was calculated for each of the 11 test samples. Across all 100 seeds, the ReLU model ranged from a minimum of 9.09% to a maximum of 81.82%, confirming that weight initialization is the critical factor in determining performance on this small filtered dataset.

Table 11: ReLU FFNN median performance (seeds 0-99).

Metric	**Value**
Number of seeds	100 (seeds 0-99)
Hidden neurons	6
Learning rate	0.00379
Median directional accuracy (test)	54.55%
Maximum directional accuracy (test)	81.82%
Minimum directional accuracy (test)	0.09%
Correct positive predictions	$6 \ / \ 7$
Correct negative predictions	$0 \ / \ 4$

The resulting median directional accuracy was 54.55%, which is the same as the sigmoid median. However, an examination of the prediction values shows a different pattern. The ReLU median predictions were positive for all 11 samples, ranging from about 0.030 to 0.128. This sign distribution is similar to the sigmoid median results, although the magnitudes differ slightly. The

ReLU ensemble correctly predicted 6 of the 7 positive outcomes and 0 of the 4 negative outcomes, matching the pattern seen with the sigmoid median.

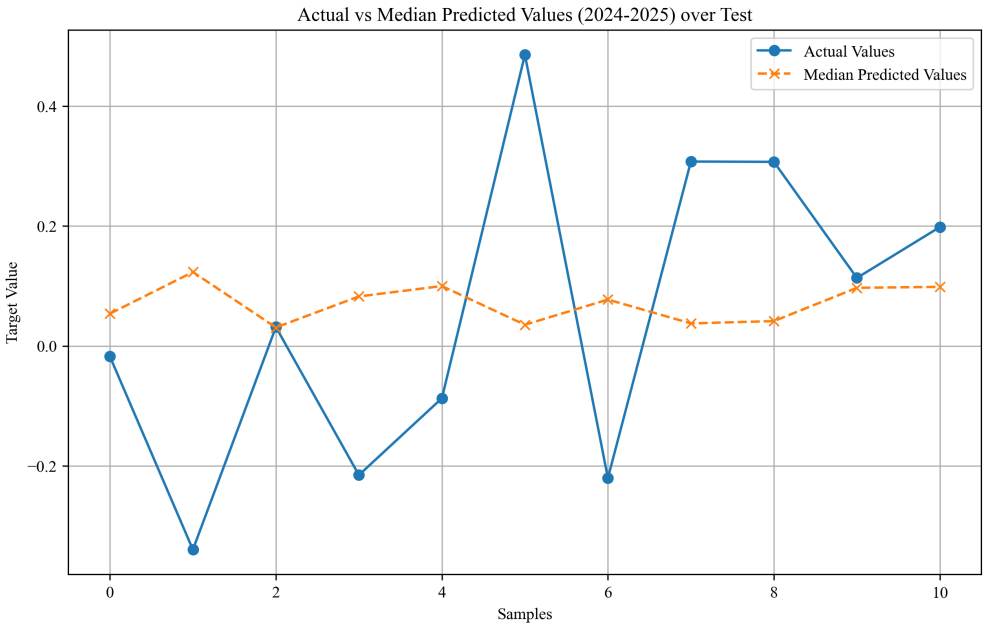

Figure 12: ReLU Actual vs Median Predicted values over Test.

The fact that both activation functions reached the same aggregate accuracy in the median is significant. This suggests that the 54.55% limit is not mainly caused by the choice of activation function, but is instead a basic feature of the filtered dataset and the sign distribution of the test set. Though both activation functions have the same level of accuracy, the number actually did not tell the whole story. In figure 9, it shows almost flat line while figure 12 show slightly cuvry line. The learning phase $\boldsymbol{L}$ set has a slight positive imbalance (53.1% positive), and this property seems to affect both activation functions during median aggregation. This leads the ensemble toward a mostly positive directional signal. This explains the need for the following ensemble method, which aims to restore predictive diversity by adding a KNN regressor.

4.4. Median ReLU FFNN and KNN

The final experiment adds a K-Nearest Neighbours (KNN) regressor to the ReLU median ensemble, combining them using a consensus mechanism. This approach was chosen because both the sigmoid and ReLU median ensembles showed a consistent positive bias when aggregated, even though individual runs performed well. A KNN regressor is a non-parametric, instance-based method that makes predictions by averaging the target values of the k nearest training points. It does not use gradient-based optimization, which is what often pushes neural networks toward biased results. Because of this, it may capture different directional signals from various parts of the feature space.

4.4.1. KNN Configuration and Individual Performance

The KNN regressor was trained on the same standardized training set of 115 samples used for the FFNN. It used k = 2 nearest neighbors, Euclidean distance, and uniform weights. Using k = 2 was a deliberate choice. Because the training set is small, a larger k might average across patterns that are too different, causing the model to lose the local detail that makes KNN useful in this setting.

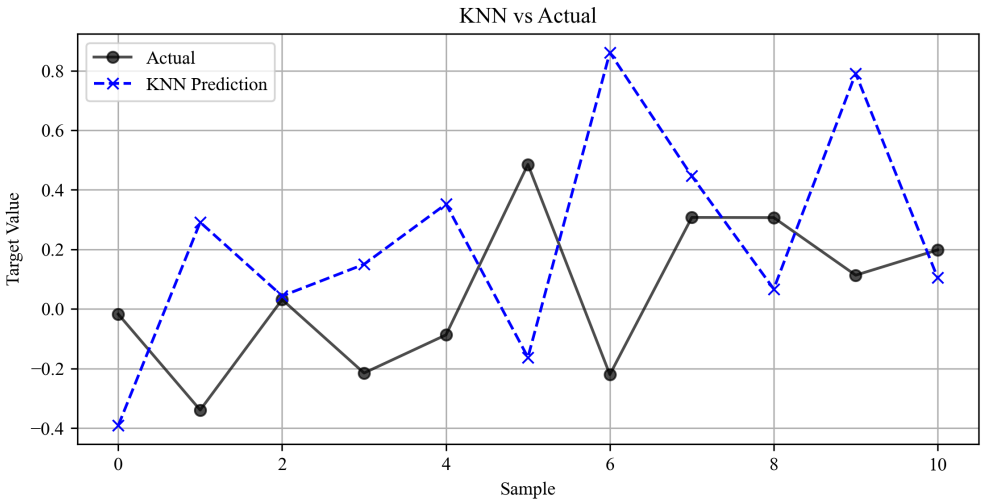

Figure 13: KNN Prediction vs Actual on Test.

When tested independently on the 11 samples, the KNN reached a directional accuracy of 54.55%, which is the same as the ReLU median FFNN. However, the signs of the KNN predictions were different. While the FFNN median predicted all 11 samples as positive, the KNN predicted 9 positive and 2 negative (samples 0 and 5). For example, sample 0 has an actual value of -0.017473. The KNN correctly predicted a negative value of -0.391660, while the FFNN median predicted a positive value of 0.053863. This difference in prediction signs provides the basis for the consensus ensemble approach.

4.4.2. Consensus Ensemble Mechanism

The consensus mechanism works as follows. For each test sample, the directional signals of the KNN and FFNN median predictions are compared. If both models predict the same sign, either positive or negative, a consensus is reached, and the prediction is considered a 'confident' signal. The ensemble prediction value is the average of the KNN and FFNN median predictions. If the two models disagree on the sign, the sample is excluded from the consensus accuracy calculation, although it is still included in the standard ensemble accuracy.

Out of the 11 test samples, the KNN and FFNN median agreed on the sign for 9 samples, re-

Table 12: KNN and FFNN Median predictions with consensus agreement on test set.

Sample	Actual	KNN Pred.	FFNN Median	Ensemble	Agree
0	−0.0175	−0.3917	0.0539	−0.1689	No
1	−0.3394	0.2912	0.1233	0.2072	Yes
2	0.0315	0.0435	0.0308	0.0372	Yes*
3	−0.2152	0.1500	0.0826	0.1163	Yes
4	−0.0871	0.3524	0.0999	0.2261	Yes
5	0.4854	−0.1625	0.0354	−0.0636	No
6	−0.2203	0.8614	0.0774	0.4694	Yes
7	0.3074	0.4465	0.0376	0.2420	Yes*
8	0.3069	0.0670	0.0415	0.0542	Yes*
9	0.1134	0.7910	0.0969	0.4440	Yes*
10	0.1983	0.1049	0.0985	0.1017	Yes*
* Indicates a correct direction prediction match under consensus agreement.					

sulting in a confidence rate of 81.82%. Of these 9 agreed samples, the models were correct 5 times, which gives a consensus directional accuracy of 55.56%. Although this is only slightly higher than the 54.55% achieved by the individual models, the main value of the consensus mechanism is its role as a confidence filter. The two samples where the models disagreed, samples 0 and 5, were both prediction errors for at least one of the models. By highlighting these disagreements, the consensus mechanism helps a trader avoid a directional trade when the two models are not aligned, which saves capital for higher-confidence opportunities.

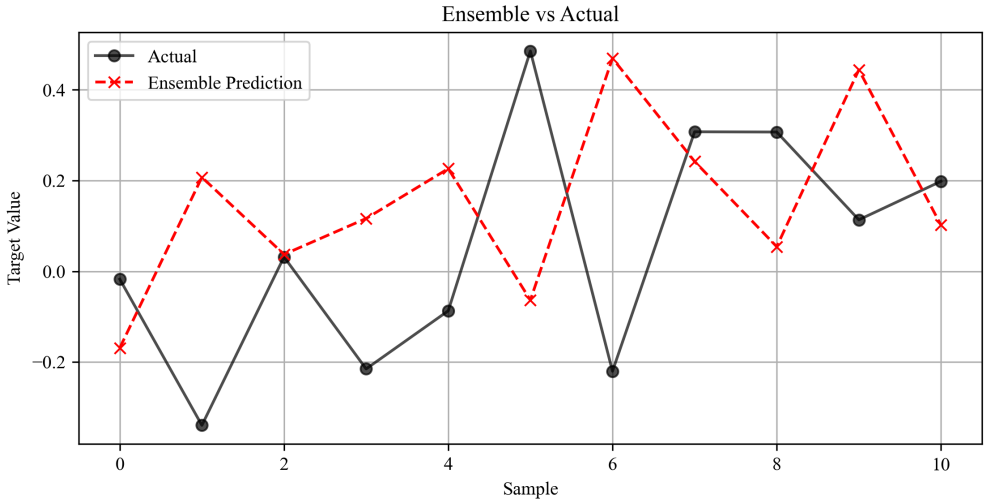

Figure 14: Ensembling the ReLU-FFNN and KNN predictions.

The arithmetic mean ensemble, which averages the KNN and FFNN median predictions for all 11 samples, achieved a directional accuracy of 54.55%. This is the same result as each individual model. This confirms that the mean ensemble does not increase accuracy over the individual models in this case, but shows the curve instead of the flat line sigmoid-ffnn shows. Therefore, the value of adding the KNN is qualitative. It provides a second, independent source

of directional prediction, and the agreement between the two models acts as a signal-to-noise filter rather than a direct boost to accuracy.

Table 13: KNN and ReLU-FFNN Median consensus ensemble results on test set.

Metric	**Value**
KNN individual accuracy	54.55%
FFNN Median (ReLU) accuracy	54.55%
Ensemble (mean) accuracy	54.55%
Samples with consensus agreement	9 / 11 (81.82%)
Consensus accuracy (when agreed)	55.56%
Correct matches in consensus	5 / 9

4.5. Comparative Summary of All Experiments

Table 14 summarizes the directional accuracy results for all the experiments in this chapter. The move from the sigmoid baseline to the ReLU single run and finally to the ensemble approach shows how the predictive system was improved over time. The single seed ReLU model with seed 10 achieved the highest test accuracy at 72.73%, showing that with favorable initial settings, the ReLU architecture can provide strong directional predictions on this filtered dataset.

Table 14: Comparative directional accuracy across all experimental configurations.

Configuration	**Val. Accuracy**	**Test Accuracy**	**Min**	**Max**
Sigmoid (Seed 10, 16n, lr=0.00183)	56.67%	45.45%	–	–
Sigmoid Median (Seeds 0-99, 6n)	—	54.55%	27.27%	81.82%
ReLU (Seed 10, 16n, lr=0.00183)	70.00%	72.73%	–	–
ReLU Median (Seeds 0-99, 6n)	—	54.55%	9.09%	81.82%
KNN (k=2, Euclidean)	—	54.55%	–	–
ReLU Median and KNN Consensus	—	55.56%	–	–

In contrast, the median aggregation approach gives up peak performance for more stability, resulting in 54.55% for both the sigmoid and ReLU models. The KNN consensus ensemble maintains this accuracy level while providing a confidence filter that is useful for creating trading signals.

5. Discussion

The experimental results in Chapter 4 require a detailed interpretation. The findings range from a low test accuracy of 45.45% for the sigmoid single seed configuration to a high of 72.73% for the best ReLU run, while the median and ensemble methods settled between 54% and 56%. This section examines these results from both theoretical and practical perspectives. It explores the relationship between Cybenko's Universal Approximation Theorem [Cyb89] and real world financial forecasting and discusses the methodological choices that led to these different results

5.1. Interpretation of the Results

The most notable finding of this study is the strong performance of the single seed ReLU model. With a test accuracy of 72.73% on 11 test samples, this configuration correctly identified the direction of EUR/USD daily movement for 8 out of 11 instances, including all 4 negative outcomes. This result is important not because it solves the problem, but because it shows that a single hidden layer FFNN, which is the simplest architecture supported by Cybenko's theorem, can produce directional predictions much better than random chance on unseen recent financial data under the right conditions. The phrase 'under the right conditions' is key. The performance of the seed 10 configuration may not be reproducible if parameters are changed, and it should not be viewed as a general result without broader validation.

A limitation of applying this theoretical framework to financial time series is that the data is discrete and non-stationary by nature. The UAT's guarantee applies to continuous functions on compact domains, whereas financial returns are discrete observations of a potentially discontinuous and time-varying process. Therefore, the theoretical framework provides motivation for the network's approximation capacity, but cannot guarantee convergence to the true underlying function given the nature of financial data. The results of this study should be interpreted with this limitation in mind.

The behavior of the sigmoid and ReLU models under median aggregation is very similar, as both reached 54.55% accuracy and predicted only positive values. However, their individual behaviors differ significantly. The single seed sigmoid model predicted only negative values on the test set, while the single seed ReLU model achieved the highest accuracy with a more balanced mix of positive and negative predictions. This contrast matches what is known about these two

activation functions. The limited output range and saturation of the sigmoid function make small networks more likely to develop strong directional biases when data is limited. Because ReLU does not saturate for positive inputs, it allows the network to maintain a consistent gradient flow across a wider range of weight settings. This allows for better directional learning when the model is initialized correctly.

As discussed in the literature review, ReLU is theoretically equivalent to sigmoid in approximation capacity but avoids the saturation problem, which explains its stronger directional performance observed in these experiments.

The KNN consensus mechanism provides a structurally independent perspective to the ensemble. The fact that the KNN and FFNN median disagreed on 2 out of 11 samples, and that both disagreements occurred where at least one model was wrong, suggests that these disagreements provide useful information. In a trading context, the 81.82% confidence rate is practically meaningful. For roughly four out of five trading opportunities identified by the Euclidean distance filter, the two models agree, and in those cases, the consensus accuracy is 55.56%. A trader using this system could avoid the 18.18% of signals where the models conflict, potentially preventing losses during uncertain market conditions.

Overall, the results align with the theoretical framework of this research. Cybenko's theorem guarantees that a single-hidden-layer sigmoid FFNN can approximate any continuous function. However, this guarantee is existential, meaning it asserts that the correct weights exist, but it does not guarantee that gradient-based training on a finite, noisy dataset will find them. The difference between theoretical capacity and actual performance is the theory-to-practice gap discussed in section 2.4.3 of the literature review. The experiments show that this gap can be overcome when using a filtered, noise-reduced dataset. Specifically, the ReLU model with seed 10 found a set of weights that turned theoretical capacity into practical directional accuracy above 70%.

5.2. Challenges

Several challenges appeared during this research that influenced both the experimental design and how the results were interpreted.

The most significant challenge was the very small amount of training data caused by the Euclidean distance filtering. Although this filtering was a deliberate and theoretical choice to remove noise and focus the network on a single recurring pattern, it reduced the training set to 115

samples and the test set to 11. In such a small dataset, neural network training is very sensitive to weight initialization. The loss landscape is inconsistent, there are many local minima, and the gradient signal from only 115 examples is not enough to reach a globally optimal solution. The wide range of accuracy observed across 100 random seeds, from near-random to 72.73%, is a direct result of this small sample size.

A second challenge was the instability of median aggregation when model results were diverse. When 100 independently trained models produce different predictions, such as some being all-positive, some all-negative, and some mixed, the median tends to shift toward the most common sign. In this case, the result was positive because of the slight positive imbalance in the $\boldsymbol{L}$ set. This means the median ensemble is robust because it is not affected by a single outlier, but it is not very useful because it loses the directional diversity found in the best individual seeds. Fixing this would require a more selective way to combine the results, which was not explored in this study.

A third challenge is the non-stationarity of the EUR/USD market. The base vector $\boldsymbol{B}$ and threshold τ were chosen using the learning phase data from December 2003 to December 2023. The test phase covers January 2024 to December 2025, a period with different macroeconomic conditions, such as post-pandemic monetary policy changes and geopolitical instability. It is not guaranteed that a pattern found to be recurring over a 20 year historical window will remain predictive in the last two years. Additionally, the small size of the test set makes it hard to determine if the results reflect a genuinely stable pattern or a period-specific coincidence.

Finally, the need for deterministic seeds creates a constraint on reproducibility. Since the results depend heavily on the initial weight configuration, any change to the software environment, framework version, or hardware could lead to numerical differences that change the training process. This fragility is a standard part of training neural networks with stochastic gradient descent. It is especially true in cases with small datasets where the training signal is weak.

5.3. Limitations

In addition to the challenges during the process, this research has several structural limitations that affect the scope and generalizability of the results.

The most significant limitation is the small size of the test set. With only 11 test samples, accuracy estimates have wide confidence intervals. Although the difference between 45.45% and 72.73% seems large, it is not statistically significant due to the small sample size. At least 50 to

100 test samples would be needed to make strong conclusions about accuracy differences between configurations. This limitation is a direct result of the Euclidean distance filtering method. While this method improves data quality, a stricter filter leads to cleaner patterns but fewer samples.

The study is also limited to a single base vector and threshold combination. The analysis of the learning phase identified one specific $\boldsymbol{B}$ and $\tau = 1.4$ configuration as the most promising, but this is only one possibility among many. Other base vectors might identify different patterns with better predictive value, more balanced classes, or larger $\boldsymbol{L}$ sets. Because the study is restricted to one pattern, the FFNN acts as a specialized predictor for one specific market configuration rather than a general directional forecaster. Selecting multiple base vectors $\boldsymbol{B}$ and thresholds τ would have provided a more complete picture of the market's behavior, covering a wider range of recurring patterns and allowing for a fuller evaluation of the forecasting system.

The range of hyperparameters tested in this study is broad, but it necessarily mentioned some specific hyperparameters in this research. Though tested over 100 of seed and different parameters, only two configurations were used in the single seed experiments: 16 neurons with a learning rate of 0.00183 for seed 10, and 6 neurons with a learning rate of 0.00379 for the median tests. A full grid search or Bayesian optimization of the neurons, learning rate, batch size, and regularization could find better settings, but this was not done due to time and computational limits.

Additionally, the FFNN was not compared against other forecasting models using the same filtered $\boldsymbol{L}$ set. While the comparisons between activation functions (sigmoid vs. ReLU) and model types (FFNN vs. KNN) are useful, the lack of a standard baseline, such as a majority-class predictor or a logistic regression, makes it hard to judge the absolute accuracy levels. Future research should include these baselines to better understand the specific contribution of the neural network architecture.

5.4. Future Directions

The findings and limitations of this study suggest several ways to improve future research. The first priority is to expand the search for base vectors($\boldsymbol{B}$) and threshold values($\boldsymbol{\tau}$). Instead of using only one $\boldsymbol{B}$ and $\boldsymbol{\tau}$ pair, future work should examine a wider range of base vectors($\boldsymbol{B}$) and threshold values ($\boldsymbol{\tau}$). Automated search strategies could be used to find configurations that produce larger $\boldsymbol{L}$ sets with strong directional imbalances. Each pattern found could then be used to create its own FFNN-KNN ensemble. This would turn the full system into a portfolio of predictors for specific

patterns, which would increase the number of trading signals and improve the statistical reliability of the evaluation.

A second area for improvement is the ensemble aggregation strategy. Instead of simply calculating the median for all 100 seeds, a validation-weighted method could be used. For example, only models that reach a certain validation accuracy threshold would be included in the final aggregation. This would reduce the impact of poor solutions and better preserve the directional diversity found in the best individual runs.

Finally, a walk-forward validation framework should replace the static train and test split used in this study. In a walk-forward design, the model is retrained periodically as new data arrives, allowing the system to adapt to changes in the market. This would address the concerns about non-stationarity and provide a more realistic look at how the system performs in live trading.

6. Conclusion

This research examined the use of single-hidden-layer feedforward neural networks (FFNNs) to predict the daily closing price direction of EUR/USD, combining Euclidean distance-based pattern filtering with neural network training on historically similar market patterns. The research examined the potential of a fundamental FFNN architecture to attain substantial accuracy when trained on a pre-filtered dataset. This investigation addressed the challenge of high noise in liquid financial instruments.

The methodology consisted of two stages. First, a Euclidean distance-based algorithm was applied to over 20 years of EUR/USD daily return data to identify similar five-day patterns. Using a base vector $\boldsymbol{B}$ and a threshold $\tau = 1.4$, a set of 145 learning-phase and 11 test-phase patterns was created. Second, single-hidden-layer FFNNs were trained exclusively on these filtered patterns. This approach focused the learning capacity on specific recurring market configurations instead of the entire noisy market history.

Three experimental approaches were tested, each evaluated across multiple configurations. The sigmoid activation function achieved limited test accuracy, with a single-seed run reaching 45.45% and the 100-seed median stabilizing at 54.55% with a strong positive bias. Across all 100 seeds, the sigmoid model ranged from a minimum of 27.27% to a maximum of 81.82%, demonstrating that the architecture is capable of strong predictions under favorable initialization conditions. The ReLU activation function produced a different result: the single-seed run reached 72.73% test accuracy, including all negative outcomes, although the ReLU median ensemble also converged to 54.55%. Across all 100 ReLU seeds, the range was wider, from a minimum of 9.09% to a maximum of 81.82%, further confirming that weight initialization is the critical factor on this small filtered dataset. Finally, combining the ReLU median FFNN with a K-Nearest Neighbours (KNN) regressor created a consensus ensemble. This system achieved a confidence rate of 81.82% and a consensus accuracy of 55.56%, acting as a useful confidence filter for trading decisions.

Through careful data pre-processing and the selection of the ReLU activation function, a simple FFNN can exceed random chance by a meaningful margin. This suggests that pattern-specific training is a promising way to avoid the problem of models memorizing noise. However, the small test set of 11 samples means the results have wide confidence intervals and may not

be fully generalizable. Additionally, the study was limited to a single base vector and threshold configuration. Selecting multiple $\boldsymbol{B}$ and $\boldsymbol{\tau}$ combinations would have provided a more complete picture of the market's recurring patterns and a fuller evaluation of the system's potential.

Future research should expand the search for base vectors($\boldsymbol{B}$) and threshold values($\boldsymbol{\tau}$) to create a portfolio of pattern-specific predictors for various market conditions. Improving the ensemble strategy by selecting only the top-k performing seeds could also reduce biased results. Finally, testing the system on other currency pairs and timeframes would help verify if this approach works beyond the EUR/USD daily setting.

In summary, this research demonstrates that a basic FFNN remains a capable tool for financial forecasting when supported by principled data filtering and the ReLU activation function. The peak accuracy of 72.73% provides evidence that a simple single-hidden-layer architecture can translate theoretical capacity into practical predictive performance under the right conditions.

References

[AFX24] Australian Foreign Exchange Committee. *Foreign Exchange Turnover Report - April 2024*. Accessed: 2025-06-15. Reserve Bank of Australia, 2024. URL: https://afxc.rba.gov.au/statistics/fx-turnover-reports/2024/apr-2024/list-of-tables.html.

[BIS22] Bank for International Settlements (BIS). *Triennial Central Bank Survey: Foreign exchange turnover in April 2022*. Accessed: 2025-06-05. Bank for International Settlements, 2022. URL: https://www.bis.org/statistics/rpfx22_fx.htm.

[BJR08] George E. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. "Time Series Analysis: Forecasting and Control". In: *Wiley* 4 (2008). 4th edition.

[BQ25] Paul Alexander Bilokon and Yitao Qiu. "Transformers versus LSTMs for Electronic Trading". In: (Feb. 2025). Submitted to ICLR 2025. URL: https://openreview.net/forum?id=2L1OxhQCwS.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[DHL+20] Alexander Jakob Dautel, Wolfgang Karl Hardle, Stefan Lessmann, and Hsin-Vonn Seow. "Forex Exchange Rate Forecasting Using Deep Recurrent Neural Networks". In: *Digital Finance* 2.1 (2020), pp. 69–96. DOI: 10.1007/s42521-020-00019-x.

[FSL24] T. Fischer, M. Sterling, and S. Lessmann. "Fx-spot predictions with state-of-the-art transformer and time embeddings". In: *Expert Systems with Applications* 249 (2024), p. 123538. DOI: 10.1016/j.eswa.2024.123538.

[Han19] Boris Hanin. "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations". In: *arXiv preprint arXiv:1708.02691* (2019). URL: https://arxiv.org/abs/1708.02691.

[Hor91] Kurt Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural Networks* 4.2 (1991). Expands on Cybenko's universal approximation

theorem, supporting FFNNs' theoretical strength for financial forecasting., pp. 251–257. DOI: 10.1016/0893-6080(91)90009-T.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997). Explains LSTM networks, widely used in time series forecasting. Useful for contrasting FFNNs with more complex architectures. Key quote: *"LSTM solves the vanishing gradient problem, enabling long-term dependency learning."*, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[Hua20] Chao Huang. "ReLU Networks Are Universal Approximators via Piecewise Linear or Constant Functions". In: *Neural Computation* 32.11 (2020), pp. 2249–2278. DOI: 10.1162/neco_a_01316.

[KMP24] Namjoon Kim, Chulhee Min, and Sejun Park. "Minimum Width for Universal Approximation using ReLU Networks on Compact Domain". In: *International Conference on Learning Representations (ICLR)*. 2024. URL: https://arxiv.org/abs/2309.10402.

[LLP+93] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. "Multilayer Feedforward Networks with a Nonpolynomial Activation Function Can Approximate Any Function". In: *Neural Networks* 6.6 (1993), pp. 861–867. DOI: 10.1016/S0893-6080(05)80131-5.

[MSA+22] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Artemios-Anargyros Semenoglou, Gary Mulder, and Konstantinos Nikolopoulos. "Statistical, machine learning, and deep learning forecasting methods: Comparisons and ways forward". In: *Journal of the Operational Research Society* 74 (Sept. 2022), pp. 1–20. DOI: 10.1080/01605682.2022.2118629.

[MVV+17] Pankaj Malhotra, T. V. Vishnu, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. "Timenet: Pre-trained deep recurrent neural network for time series classification". In: *Proceedings of the 25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. 2017.

[NFX24] New York Foreign Exchange Committee. *Foreign Exchange Volume Survey Results - October 2024*. Accessed: 2025-06-15. Federal Reserve Bank of New York, 2024. URL: https://www.newyorkfed.org/fxc/volumesurvey.

[Par+21] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. “Minimum Width for Universal Approximation”. In: *International Conference on Learning Representations (ICLR)*. 2021. URL: https://arxiv.org/abs/2006.08859.

[Ran25] A. Ranaroussi. *yfinance: A Python library for accessing Yahoo Finance data*. PyPi. 2025. URL: https://pypi.org/project/yfinance/.

[SGO19] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. *Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005-2019*. 2019. arXiv: 1911.13288 [cs.LG]. URL: https://arxiv.org/abs/1911.13288.

[Sha26a] Md Jakaria Mashud Shahria. *Euclidean Distance Calculation — Artificial Neural Network for Time Series*. GitHub repository. Accessed: 2026. URL: https://github.com/shahria-codes/artificial-neural-network-for-time-series/tree/main/ipynb_files/ED_Calculation.

[Sha26b] Md Jakaria Mashud Shahria. *Neural Network Experiments — Artificial Neural Network for Time Series*. GitHub repository. Accessed: 2026. URL: https://github.com/shahria-codes/artificial-neural-network-for-time-series/tree/tensorflow/stop/ffnn-knn/ipynb_files/Neural_Network.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017), pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[Yah25] Yahoo. *Yahoo Finance*. Online. 2025. URL: https://finance.yahoo.com.

[Zha03] G. Peter Zhang. “Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model”. In: *Neurocomputing* 50 (2003). Compares hybrid models (ARIMA + ANN) vs. standalone methods. Demonstrates ANN’s superiority in nonlinear scenarios., pp. 159–175. DOI: 10.1016/S0925-2312(01)00702-0.

[ZK24] T. Zafeiriou and D. Kalles. “Off-the-Shelf Neural Network Architectures for Forex Time Series Prediction come at a Cost”. In: *arXiv preprint arXiv:2405.10679* (2024). URL: https://arxiv.org/abs/2405.10679.

[ZPH98] G. Zhang, B. Eddy Patuwo, and M. Y. Hu. “Forecasting with artificial neural networks: The state of the art”. In: *International Journal of Forecasting* 14.1 (1998), pp. 35–62. DOI: 10.1016/S0169-2070(97)00044-7.

[ZY24] Lu Zhao and Wei Qi Yan. “Prediction of Currency Exchange Rate Based on Transformers”. In: *Journal of Risk and Financial Management* 17.8 (2024). ISSN: 1911-8074. DOI: 10.3390/jrfm17080332. URL: https://www.mdpi.com/1911-8074/17/8/332.