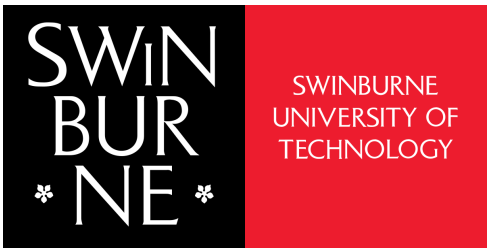


# On Forecasting, Deep Learning and and Kolmogorov Arnold Networks

NPS30003 Grand Challenges in Science

**by**
**Tabsish Ali Rather**
**S M Mahmudul Hasan Joy**
**and**
**Jack Ruzeu**

Word count = 4120

**Abstract**

This project investigates forecasting methods with a focus on stock market data due to its stochastic nature. The goal is to evaluate models that balance accuracy and interpretability—an often challenging combination in current forecasting techniques.

The ability to forecast stochastic data is critical across fields like finance, climate modeling, and energy management, where accurate predictions influence decisions in investment, resource allocation, and sustainability planning. This study examines traditional models such as random walks, advanced deep learning models like Long Short-Term Memory networks (LSTMs), and newer architectures, specifically Kolmogorov-Arnold Networks (KANs). While LSTMs excel in accuracy, their low interpretability limits their utility in contexts requiring transparency. In contrast, KANs hold promise for achieving both accuracy and interpretability, though their performance on highly stochastic data remains underexplored.

The methodology includes collecting historical stock data, processing it for forecasting, and evaluating model performance using RMSE and MSE. Initial findings indicate that while traditional forecasting models perform poorly in accuracy, LSTMs offer high predictive accuracy but lack transparency. KANs, though less accurate than LSTMs are highly interpretable by their very design, with potential for accuracy improvements through specialised architectures.

These results suggest that KANs could be a viable option for stochastic forecasting with further development, offering a pathway to models that are both precise and interpretable in fields where understanding model decisions is as important as the predictions themselves.

# Declaration

'We hereby declare that this submission is the authors own work and to the best of our knowledge it contains no materials previously published or written by another person, or substantial proportions of material, except where due acknowledgment is made in the manuscript. Any contribution made to the research by others, with whom we have worked at Swinburne or elsewhere, is explicitly acknowledged in the report. We also declare that the intellectual content of this report is the product of our own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Tabish Ali Rather, S M Mahmudul Hasan Joy and Jack Ruzeu
November 3, 2024

**Cover sheet for submission of work for assessment**

SWINBURNE UNIVERSITY OF TECHNOLOGY

**UNIT DETAILS**

| | | | | |
|---|---|---|---|---|
| Unit name | Grand Challenges | Class day/time | Mondays, 1:30pm | Office use only |
| Unit code | NPS30004 Assignment no. | Due date | 3/11/2014 | |
| Name of lecturer/teacher | Nadezda Sukhorukova | | | |
| Tutor/marker's name | Nadezda Sukhorukova | | | Faculty or school date stamp |

**STUDENT(S)**

| | Family Name(s) | Given Name(s) | Student ID Number(s) |
|---|---|---|---|
| (1) | Ruzeu | Jack | 103051551 |
| (2) | Rather | Tabish Ali | 103534434 |
| (3) | Joy | S M Mahmudul Hasan | 103831012 |
| (4) | | | |
| (5) | | | |
| (6) | | | |

**DECLARATION AND STATEMENT OF AUTHORSHIP**

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

**Student signature/s**

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1) Ruzeu (4)
(2) Tabish rather (5)
(3) (6)

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/** | PAGE 1 OF 1

# Acknowledgement

We would like to extend our heartfelt gratitude to Dr. Nadezda Sukhorukova (Nadia) for her invaluable support, guidance, and kindness throughout this project. Her expertise and encouragement have been instrumental in helping us navigate challenges and refine our work.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

Uncertainty is an inherent part of financial markets, where stochastic processes model the random movements of asset prices and interest rates. These processes have long been used to understand financial unpredictability. As financial systems become more complex, the application of advanced tools like neural networks has become vital, enabling the analysis of massive datasets to extract hidden patterns. Traditional deep learning models, such as Long Short-Term Memory (LSTM) models [3], are widely used for time series forecasting. However, they suffer from poor interpretability. Recent developments in neural network architectures, particularly Kolmogorov-Arnold Networks (KANs) [1], offer a structured approach to represent multivariate functions, potentially leading to more efficient and interpretable models.

The primary purpose of this study is to evaluate the effectiveness of KANs on stochastic data compared to traditional deep learning models like LSTMs and statistical methods like traditional forecasting models [4]. This comparison is significant for advancing the theoretical understanding of neural networks in the context of forecasting and decision making, particularly in terms of achieving both accuracy and interpretability. By exploring these different approaches, the study aims to contribute to the growing field of financial time series forecasting and to the emerging research on KANs.

The contribution of this project is twofold. First, it adds to the theoretical body of knowledge by exploring the capabilities of KANs, addressing a gap in understanding their performance on highly stochastic data. Second, it holds practical significance for the global scientific community, as enhanced accuracy and interpretability in forecasting models can

improve informed decision-making across industries such as finance, risk management, and resource planning. This work provides insights that could lead to more reliable, interpretable models for forecasting financial data and other stochastic processes.

# Chapter 2

# Literature Review and Theoretical Framework

## 2.1 Stochastic Processes and the Origins of Mathematical Finance

### 2.1.1 What is a stochastic process?

A stochastic process (also called a random process) is defined mathematically as a sequence of random variables $X_t$, for $t \in T$, with probability space $(\Omega, F, P)$ and state space $(S, \mathcal{S})$ [5]. Here, $\Omega$ represents the sample space, which is the set of all possible outcomes the random variable $X$ might take – while $F$ represents the event space, which comprises certain individual outcomes or other outcome subsets in the sample space $\Omega$. $P$ denotes a probability set function, which maps events in the event space to probabilities consistent with event space outcomes, $P : F \to [0, 1]$. The state space, $(S, \mathcal{S})$, refers to all possible configurations (be it position or velocity, or in the case of financial analysis, it may refer to credit ratings) that a stochastic system may inhabit.

Typically, the key difference between state space and probability space is that the probability space associates a likelihood to values in the event space (and hence sample space) via a probability function. For example, a credit rating may place any particular individuals into categories (states) ranging from the best credit rating to the poorest, only after the probability of events like default have been determined as part of the probability

space [6]. Mathematically, a stochastic process is defined as,

$$\{X_t \mid t \in T\}, t_1, \ldots, t_n \in T$$

Alternatively, we can express it as:

$$X(t, \xi),$$

where, for any particular $t$, the variable $X_t$ (for example, $t = t_1$) represents a single random variable in the stochastic process. Here, $\xi$ represents a random outcome and is indexed accordingly to correspond to each random variable, $X_{t_i}$. The outcome $\xi$ can be thought of as a random jump from a previous state to an unpredictable future state. Consider the generalised random walk process as a sum of a sequence of independent and identically distributed random variables:

$$X_n = X_0 + \sum_{i=1}^{n} \xi_i.$$

The dimensions of the integer lattice system are specified by:

$$\mathbf{Z}^d = \{(x_1, \ldots, x_d) \mid x_j \in \mathbf{Z}\}.$$

The variable $X_0$ is given as a starting point for the model in the d-dimensional space and $\xi_i$ are the random movements associated with each random variable in the sequence. The random walk process is extensively utilised in mathematical financial forecasting and is the basis for many of the following topics that will be discussed [7]. Consequently, it is necessary to briefly describe it. Furthermore, when defining the distribution of a stochastic process, one must consider the joint probability function, typically the Cumulative Distribution Function (CDF), from which the Probability Density Function (PDF) can be obtained through partial differentiation:

$$F_{X_{t_1},\ldots,X_{t_n}}(x_1, \ldots, x_n) = P(\{X_{t_1} \leq x_1\} \cap \ldots \cap \{X_{t_n} \leq x_n\}).$$

### 2.1.2 A Brief History of Mathematical Finance

What is described as mathematical finance is generally thought to have begun with French mathematician Louis Bachelier, when he postulated his thesis in 1900 [8]. In his paper, "Theory of Speculation," Bachelier sought to describe day-to-day fluctuations in the Paris stock exchange forward contracts and options market, with respect to government-issued perpetual bonds (having no maturity date). Bachelier's basis for the determination of the probability of a particular current quoted price follows a list of derivations for various methods of predictive pricing. He begins first with the law of compound probability for independent events, in his own notation:

$$p_{z,t_1+t_2}dz = \int_{-\infty}^{\infty} p_{x,t_1}p_{z-x,t_2}dx,$$

where $p_{x,t_1}$ represents the probability of $x$ being the quoted price at time $t_1$ and $p_{z-x,t_2}$ represents the probability of the next quoted price $z-x$ taking some value before the eventual current quoted price at $t_1+t_2$. Defining a suitable function for $p$ and integrating, Bachelier asserts that the following function defines the probability of some quoted price of securities:

$$p = p_0 e^{-\pi^2 p_0 x^2}.$$

Bachelier goes on to derive probability as a function of time and many other relationships.

Following on from Bachelier, other notable names include French mathematician Henri Lebesgue [9], Johann Radon, and Percy Daniell. The combination of these preceding ideas led Norbert Wiener to formulate a complete description of Brownian motion in 1923 [7]. Brownian motion, named after Robert Brown – a Scottish botanist studying pollen grains submerged in water, refers to any random process where increments in motion or fluctuation are seemingly arbitrary and unpredictable. The Wiener process resembles the description of a random walk where a sequence of stationary, independent increments that, when summed, will converge to a Wiener process:

$$W_{t_n} = \sum_{i=1}^{n} \Delta W_{t_i}.$$

In 1934, French engineer and mathematician Paul Levy introduced the theory of martingale [10], and this has become one of the principal methods of analysis in modern finance. The name martingale has its origins in early 18th century France, coined after a betting strategy where gamblers doubled down their bets after each successive loss in the hopes that eventually, one must win. Given a family of information sets $\{I_t, t \in [0, \infty]\}$ (interpreted as available historical information regarding the stochastic process $S_t$), then a stochastic process $\{S_t, t \in [0, \infty]\}$ is called a martingale if:

1. $S_t$ is adapted to $I_t$ (given historical data, $S_t$ is formulated).
2. Unconditional forecasts are finite, $E[S_t] < \infty$ - bets that can be placed, or predictions made, are not infinite with respect to resources.
3. $E_t[S_T] = S_t$, with probability equal to 1, states that the best possible forecast of some future outcome is equal to the last outcome, $S_t$.

In other words,

$$E_t[S_{t+u} - S_t] = E_t[S_{t+u}] - E_t[S_t] = 0.$$

The following decade, in 1944, saw Kiyosi Ito's work surrounding stochastic integration [5]. Given the form of a general stochastic differential equation:

$$dx = f(x)dt + g(x)dW(t),$$

the term $dW(t)$ is a Wiener process, and $f(x)$ and $g(x)$ are some functions. Through the use of a truncated Taylor series expansion, Ito's formula is obtained:

$$\frac{dF}{dt} = F'(x)f(x) + \frac{F''(x)}{2}g^2 + F'(x)g(x)\xi(t).$$

Later, in his 1953 book, Joseph L. Doob expanded on Ito's stochastic integration method to processes with conditionally orthogonal increments (successive increments between steps in the process are uncorrelated) [11]. From here, the Doob decomposition method for sub-martingales was established. Following on, in 1973, Robert Merton, Myron S. Scholes, and Fisher Black introduced what is likely the most well-known model in stochastic

finance – the Black-Merton-Scholes model [12]. While it is known predominantly as the Black-Scholes model, Merton's formulas expanded their work by considering dividend payments and contributed to the field of economics in many other areas. Together, Myron Scholes and Robert Merton, along with Fisher Black posthumously, in 1997, received the Nobel prize for their contributions to the world of finance and economics.

## 2.2 Deep Learning and its Mathematical Foundations

### 2.2.1 Origins of Deep Learning

The concept of intelligent machines has its roots in the pioneering work of Alan Turing, particularly in his seminal paper "Computing Machinery and Intelligence" (1950), which laid the groundwork for the field of artificial intelligence (AI) [13]. However, the true potential of AI was not fully realised until the fields of neuroscience and mathematics were intertwined. Even before Turing's influential work, two scientists, neurologist Warren S. McCulloch and logician Walter Pitts, had already begun to bridge the gap between biology and logic. In their groundbreaking paper, "A Logical Calculus of the Ideas Immanent in Nervous Activity" (1943), McCulloch and Pitts proposed a model of the neuron based on logical functions, which provided a mathematical framework for understanding the brain's functioning [14].

### 2.2.2 Mathematics of Deep Learning

The fusion of McCulloch and Pitts' ideas with mathematical theory led to a significant breakthrough in AI. Frank Rosenblatt, a psychologist and logician, advanced their work by developing the perceptron—a mathematical model of a neuron that served as the foundation for artificial neural networks. The perceptron operates using a simple decision rule [15]:

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$

Where $w_i$ is the weight for the $i$-th input $x_i$, $x_i$ is the $i$-th input feature, and $T$ is the threshold that determines the output based on the weighted sum of inputs.

While perceptrons demonstrated the ability to perform various computations, they were limited in their ability to solve certain logical problems, such as the XNOR gate computation. This limitation was highlighted by Marvin Minsky and Seymour Papert in their book "Perceptrons: An Introduction to Computational Geometry" (1969), where they showed that single-layer perceptrons could not solve non-linearly separable problems [16]. However, they also demonstrated that multi-layer perceptrons (MLPs) could overcome this limitation, thus laying the foundation for deep learning.

### 2.2.3 Universal Approximation Theorem

The realisation that multi-layer architectures could perform complex computations led to significant theoretical developments in neural networks. George Cybenko, in his influential work, formulated the Universal Approximation Theorem, which asserts that a feedforward neural network with a single hidden layer can approximate any continuous function on a compact subset of $\mathbf{R}^n$, given a sufficient number of neurons [17]. Mathematically, the theorem can be expressed as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \sigma(\mathbf{w}_i^\top \mathbf{x} + b_i)$$

Where $\mathbf{x}$ is the input vector, $\mathbf{w}_i$ are the weight vectors, $b_i$ are the biases, and $\alpha_i$ are the output weights. The theorem concludes that for any continuous function $g : K \to \mathbf{R}$ defined on a compact subset $K \subset \mathbf{R}^m$ and for any $\epsilon > 0$, there exists a neural network such that:

$$\sup_{\mathbf{x} \in K} |g(\mathbf{x}) - f(\mathbf{x})| < \epsilon$$

This result provided a rigorous mathematical foundation for neural networks, establishing their capability to approximate complex functions and thus perform a wide range of tasks.

### 2.2.4 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are a type of computer model/program designed to mimic how the human brain works. They are a broader class of models that generalise Multi-Layer Perceptrons (MLPs), both are based on the same principles[15]. Like our

brains are made up of billions of interconnected neurons, ANNs are made up of layers of connected units called "nodes" or "neurons". These nodes work together to process information and learn patterns from data [18]. Importantly, neural networks are grounded in the Universal Approximation Theorem, which guarantees that a sufficiently large network can approximate any continuous function, providing a theoretical foundation for their ability to learn complex mappings.[17]

When we train an ANN on some data, like pictures, text, or numbers, it learns to recognise patterns or relationships within that data. For example, if we train an ANN on many pictures of cats and dogs, it can learn to tell the difference between them. This learning process is done through "training," where the network adjusts its internal connections, called "weights," based on the data it sees. The weights are adjusted usually using back-propagation[19].

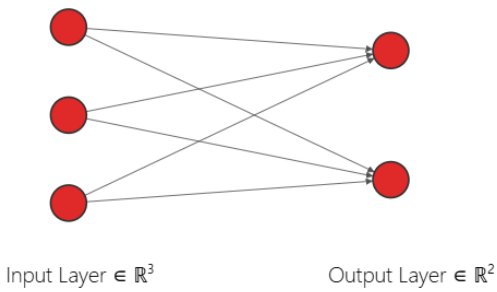


**Figure 2.1:** ANN with 0 hidden layers

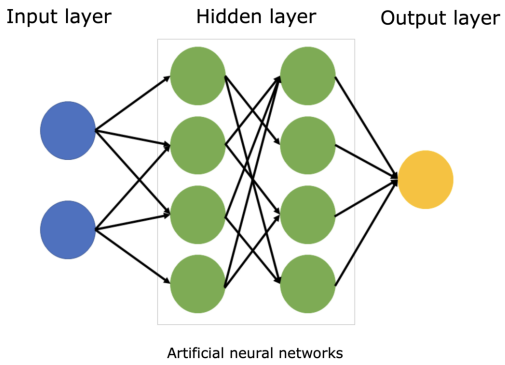


**Figure 2.2:** ANN with multiple hidden layers

Although1 ANNs can be quite useful, they require a lot of data and computational power to learn effectively. The more layers[20] and nodes an ANN has, the better it can understand complex patterns, but it also becomes more difficult to train. This is why ANNs are often used in tasks where traditional methods struggle, such as image recognition, natural language processing, and financial forecasting.

### 2.2.5 Kolmogorov-Arnold Theorem (KAT)

While MLPs have dominated the landscape of deep learning, other developments have introduced alternative approaches based on deeper mathematical insights. One such approach is the Kolmogorov-Arnold Theorem, which emerged from a problem posed by David Hilbert in 1900 [21]. Hilbert's 13th problem questioned whether a multivariate

function could be represented as a composition of univariate functions. In 1957, Andrey Kolmogorov and his PhD student Vladimir Arnold provided a partial solution to this problem, demonstrating that any multivariate continuous function can be expressed as a superposition of continuous functions of one variable [22]. The theorem is mathematically represented as:

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left( \sum_{p=1}^{n} \varphi_{q,p}(x_p) \right).$$

### 2.2.6 Kolmogorov-Arnold Network (KAN)

In 90s of the last century, despite the success of the Kolmogorov-Arnold Theorem, its potential use in machine learning was questioned. Girosi et al. (1989) [23] claimed that the superposition of functions in the first layer resulted in non-smooth functions, and therefore could not be applied to machine learning. However, Kůrková (1991) [24] rebutted this argument by stating that the inner and outer functions of the Kolmogorov-Arnold Theorem do not need to be exact. She suggested that locally smooth approximations could be used to create multilayer networks similar to modern MLPs.

Building on the Kolmogorov-Arnold Theorem, Liu et al. (2024) introduced the Kolmogorov-Arnold Network (KAN), which provides an alternative to traditional MLPs. Unlike MLPs that use fixed and non-learnable activation functions for each neuron, KANs utilise learnable activation functions organised in multiple layers, enhancing their interpretability and efficiency for certain tasks [1]. The introduction of KANs represents a significant shift in deep neural network design, driven by fundamental mathematical principles.

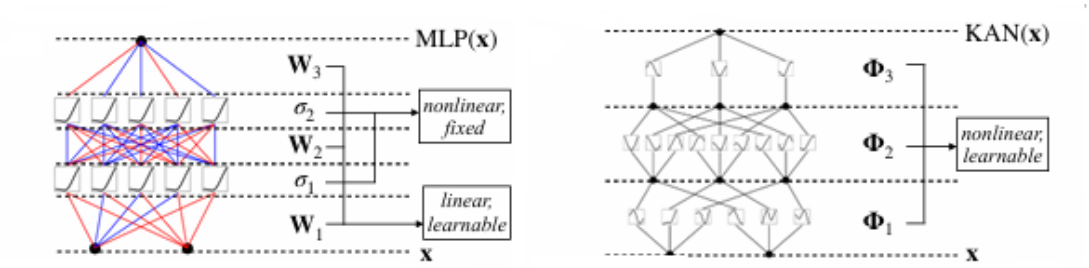


**Figure 2.3:** Multi-Layer Perceptrons (MLPs) vs Kolmogorov-Arnold Networks (KANs)[1]

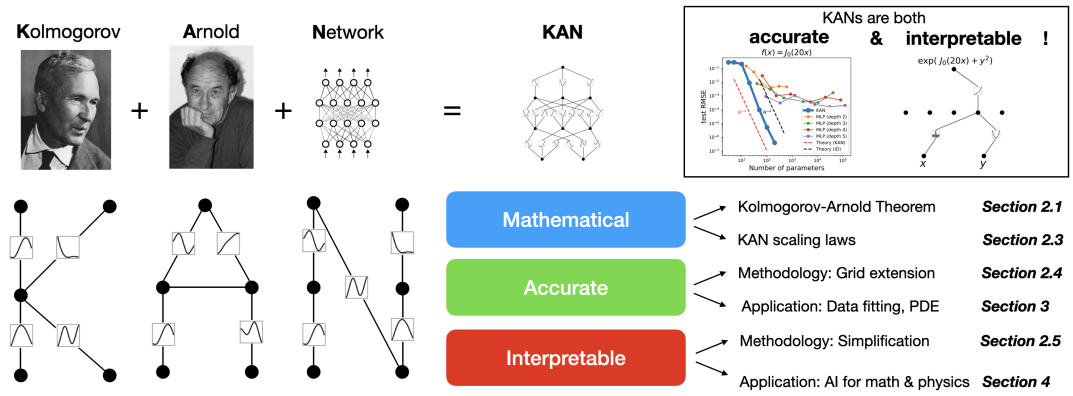


**Figure 2.4:** Kolmogorov-Arnold Networks (KANs) combine mathematical rigor with accuracy and interpretability, grounded in Kolmogorov-Arnold theorem, [1]

## 2.3 Introduction to Deep Learning for Financial Data

Deep Learning (DL) has become a powerful tool for analysing complex datasets; in recent times, efforts are being made to apply Deep Learning to financial markets, particularly the stock market. The mathematics behind DL models involves optimising high-dimensional functions and learning patterns from data. Stock market data, known to be unpredictable and noisy, require advanced methods to extract reliable insights. DL methods have been particularly effective in this domain because they can model non-linear relationships and extract hidden patterns that traditional statistical methods might miss [25]. This section reviews key deep learning models and techniques, as well as their applications in financial data, while explaining their important mathematical foundations.

### 2.3.1 Data Transformation and Feature Engineering

One of the main challenges in applying deep learning to financial data is the high dimensionality of the data sets, or the so-called "Curse of Dimensionality" [26]. Techniques like auto-encoders and Restricted Boltzmann Machines (RBMs) help reduce dimensionality while preserving important features.

An **auto-encoder** is a type of neural network that learns to compress the data (encoding) and then reconstruct it (decoding). Mathematically, an auto-encoder consists of two main functions and encoder and a decoder.

In financial applications, auto-encoders compress stock market data, thereby reducing noise and making it easier to analyse the data and make reliable predictions [27].

### 2.3.2 Deep Learning Models for Stock Market Prediction

Financial markets are complex, requiring models that can capture both time-based dependencies and non-linear relationships. **Recurrent Neural Networks (RNNs)** and their variant, **Long Short-Term Memory (LSTM)** networks, are particularly well suited for this task.

An RNN processes sequences of data by maintaining a hidden state that updates at each time step:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t)$$

where $h_t$ is the hidden state at time $t$, $x_t$ is the input at time $t$, and $W_h$ and $W_x$ are weight matrices. The function $\sigma$ represents a non-linear activation function, often the **sigmoid** function [2]. The sigmoid function maps any real-valued number into the range (0, 1), defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. This property makes it useful for squashing values to a bounded range which helps in gradient-based learning. However, RNNs can struggle with the **vanishing gradient problem** [28], where gradients (used in optimisation) become very small and stop the model from learning effectively over long sequences.

LSTMs solve this issue using gates to control the flow of information. The **input gate**, **forget gate**, and **output gate** regulate the information passed through the network. These gates are defined as follows:

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input gate)} \\ f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(Forget gate)} \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output gate)} \end{aligned}$$

LSTMs are widely used to predict stock prices and trends because they effectively learn patterns in time series data[29].

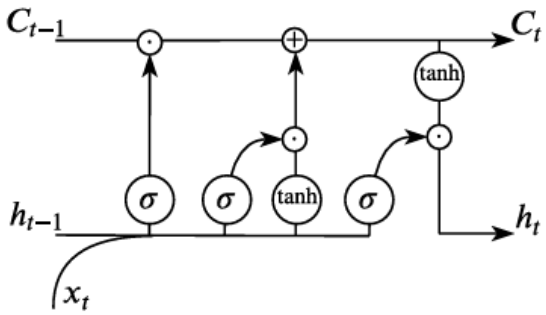


**Figure 2.5:** LSTM representation (Goodfellow et al 2016)[2]

# Chapter 3

# Methodology

## 3.1 Problem Formulation

This project addresses the problem of forecasting on stochastic data, specifically stock prices, by exploring a two-fold approach: statistical forecasting and function approximation with deep learning, and attempts to make a comparative analysis between these two methods to enable a deeper understanding of each method's nuances and trade-offs. Statistical forecasting methods, such as random walks, ARIMA and more interpretable but struggle with capturing complex patterns, while deep learning methods, such as LSTMs, can model complicated dependencies and nonlinear relationships but often lack interpretability and require substantial data and computational resources.
Statistical forecasting methods assume that past values, trends, seasonality, and residual errors contain information about future outcomes. By modelling these components, they can capture the temporal dependencies in data, making it possible to generate **forecasts** by extrapolating these patterns forward.

On the other hand, the deep learning component leverages neural networks for **function approximation**. Here, the goal is not merely to predict specific future values but to approximate the underlying functional relationships between different variables within the dataset.

## 3.2 Data Collection and Preparation

The historical stock market data was obtained using the `yfinance` API [30], allowing users to enter the company ticker as well as the duration for the data. Key variables extracted from the dataset included *open*, *close*, *high*, *low* prices, *trading volume*, and *adjusted trading volume*.

**Handling Missing Values:** To maintain data integrity, any missing values in the dataset were removed. This approach was chosen to prevent potential inaccuracies during model training.

**Normalisation:** Normalisation was achieved using Min-Max scaling, which transformed all features to a consistent range between 0 and 1. This step helps models converge faster.

**Feature Engineering:** To create target variables for the model, future values of the *Close* price were generated based on a user-defined prediction window.

**Data Splitting:** The processed data was then split into sequences of a fixed length, that users can decide, with each sequence containing historical observations used to predict future values.

### 3.2.1 LSTM Model Development

The LSTM model was chosen for its ability to capture temporal dependencies in sequential data. The model architecture was defined based on several adjustable hyperparameters:

- **Number of Layers:** The model consisted of multiple stacked LSTM layers, determined based on performance during experimentation.
- **Units per Layer:** The number of units in each LSTM layer was set through trial-and-error, balancing complexity and model accuracy.
- **Other Hyperparameters:** The activation function, loss function, and optimiser were configured based on the model's convergence during training. Specific metrics, such as Mean Squared Error (MSE), were used to evaluate model performance.

**Hyperparameter Tuning:** Tuning was conducted manually, adjusting the hyper-parameters iteratively based on performance on the training data.

**Evaluation Metrics:** Model performance was compared using Mean Squared Error

(MSE) and Root Mean Squared Error (RMSE).

### 3.2.2 KAN Model Development

The Kolmogorov-Arnold Network (KAN) [1] model was implemented in this project to test its feasibility to model stochastic time series data.

**Model Architecture**: The KAN model was designed with an input layer, hidden layers based on the Kolmogorov-Arnold representation, and a single output neuron for predictions. Key elements of the model architecture included:

- **Input Size**: The input size was configured to match the flattened dimensions of the input features, which contained stock market data.
- **Hidden Layer Width**: The model consisted of multiple hidden layers determined based on performance during experimentation.
- **Grid and k-Value Parameters**: The grid size in the KAN model defines the level of granularity in dividing the input space, while the parameter k controls the number of basis functions used to approximate complex relationships in the data. Again, these are chosen by experimentation.
- **Optimizer**: The KAN model was trained using the Limited-memory Broyden–Fletcher–Goldfarb– (LBFGS) [31] optimisation algorithm. It was used because it is effective for high-dimensional time series data.

**Hyperparameter Tuning**: These were manually tuned to optimise the model's performance. These adjustments were made based on performance observed during training.

# Chapter 4

# Results and Discussions

## 4.1 Results

### 4.1.1 LSTM Results

The results, summarised in the table below, show the training and testing MSE and RMSE for various configurations

| Model Configuration | Train MSE | Test RMSE | Test MSE | Test RMSE |
|---|---|---|---|---|
| LSTM (4 layers, 100 units, activation=linear) | 0.0071 | 0.0841 | 0.0069 | 0.0829 |
| LSTM (5 layers, 100 units, activation=linear) | 0.0089 | 0.0942 | 0.0102 | 0.1010 |
| LSTM (6 layers, 100 units, activation=linear) | 0.0112 | 0.1059 | 0.0113 | 0.1062 |
| LSTM (6 layers, 50 units, activation=linear) | 0.0115 | 0.1073 | 0.0115 | 0.1074 |
| LSTM (6 layers, 20 units, activation=linear) | 0.0104 | 0.1018 | 0.0117 | 0.1080 |
| LSTM (6 layers, 20 units, activation=tanh) | 0.0140 | 0.1183 | 0.0139 | 0.1179 |
| LSTM (3 layers, 20 units, activation=tanh) | 0.0064 | 0.0802 | 0.0070 | 0.0835 |
| LSTM (2 layers, 20 units, activation=tanh) | 0.0065 | 0.0804 | 0.0063 | 0.0792 |
| LSTM (2 layers, 10 units, activation=tanh) | 0.0065 | 0.0807 | 0.0055 | 0.0745 |

**Table 4.1:** Performance metrics for different LSTM model configurations, showing training and testing MSE and RMSE values. Highlighted cells indicate the lowest values.

The configurations demonstrate how the LSTM model's performance varies with difference hyper-parameters:

- **Number of Layers and Units**: Increasing the number of layers and units generally leads to higher training and testing errors, as seen in configurations with 5 or 6 layers and higher units per layer. This indicates overfitting as the model complexity increases, especially with larger units.
- **Optimal Configuration**: The configuration with 2 layers and 10 units (tanh activation) achieved the lowest test RMSE (0.0745), suggesting this setup may provide

the best balance of model simplicity and predictive accuracy on this dataset.

In summary, these LSTM configurations illustrate the trade-offs between model complexity and accuracy. While deeper models with more units tend to have higher errors, simpler configurations such as 2 layers with 10 units yield better generalisation on unseen data.

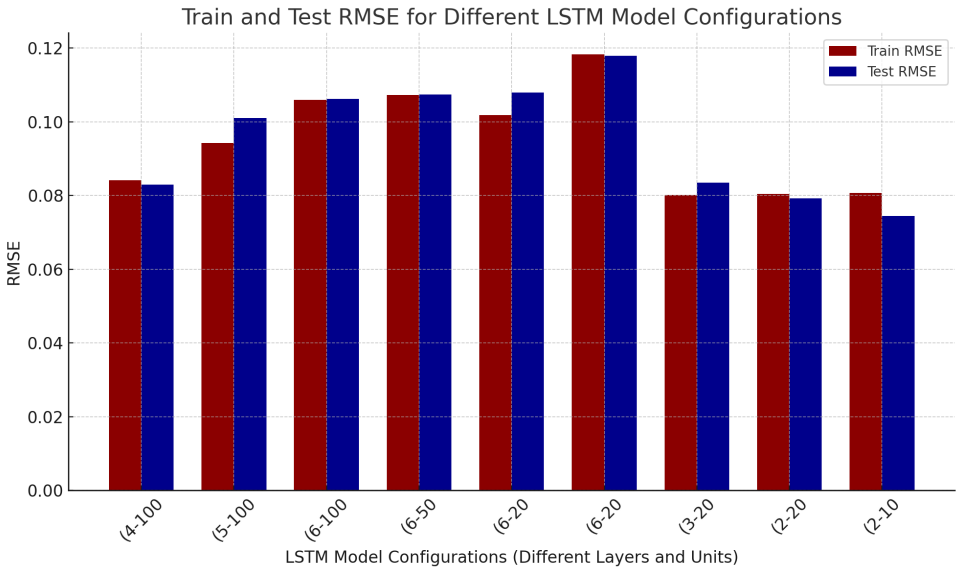


**Figure 4.1:** Train and Test RMSE for Different LSTM Model Configurations

### 4.1.2 KAN Results

The results, summarised in the table below, show the performance of various configurations of the KAN models specific hyperparameters

| Configuration | Grid | k | Train MSE | Train RMSE | Test MSE | Test RMSE | Num Neurons |
|---|---|---|---|---|---|---|---|
| KAN Config 1 | 3 | 6 | 0.0751 | 0.274 | 0.03534 | 0.188 | `len(dataset['train_input']) // 10` |
| KAN Config 2 | 3 | 2 | 0.0751 | 0.274 | 0.03534 | 0.188 | `len(dataset['train_input']) // 10` |
| KAN Config 3 | 7 | 2 | 0.271 | 0.5202 | 0.1096 | 0.331 | `len(dataset['train_input']) // 10` |
| KAN Config 4 | 3 | 2 | 0.074 | 0.272 | 0.0566 | 0.238 | `len(dataset['train_input']) // 4` |
| KAN Config 5 | 3 | 2 | 0.0751 | 0.274 | 0.0231 | 0.152 | `len(dataset['train_input']) // 10` |
| KAN Config 6 | 3 | 2 | 0.0751 | 0.274 | 0.0231 | 0.152 | `len(dataset['train_input']) // 5` |

**Table 4.2:** Comparison of KAN Configurations with Train and Test MSE and RMSE values

Overall, the table demonstrates that careful tuning of grid size, $k$, and the number of neurons can significantly affect the KAN model's ability to generalise, with Configs 5 and 6 achieving the best performance.

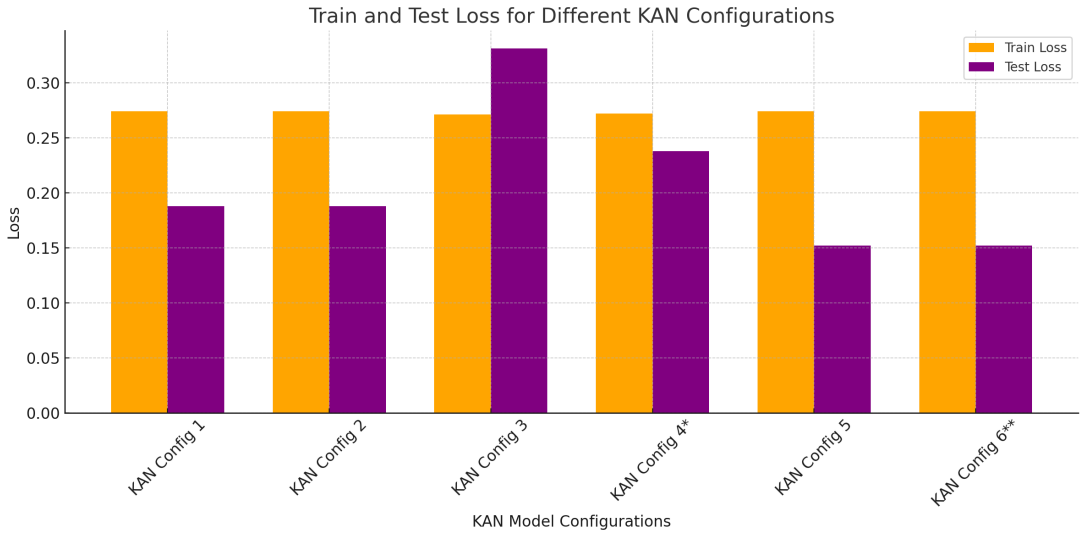


**Figure 4.2:** Train and Test Loss for Different KAN Configurations

## 4.2 Discussion

Following the data collection, Long Short-Term Memory (LSTM) networks were initially applied to raw stock market data due to their effectiveness with sequential data. While LSTM demonstrated strong predictive performance, its "black-box" nature limits interpretability, making it challenging to understand how data processing and weight adjustments contribute to predictions [32]. This lack of transparency presents challenges in applications where the decision-making process needs to be as understandable as the output [33].

To address these limitations, Kolmogorov-Arnold Networks (KANs) were explored as an alternative. KANs, grounded in the Kolmogorov-Arnold representation theorem, offer a structured approach to representing multivariate functions, allowing for model transparency and interpretability [22]. Unlike LSTM, KANs allow insight into how the model realises the predictions, providing a clearer model structure suitable for forecasting, where both accuracy and transparency are critical [34].

### 4.2.1 Comparative Analysis: Test Loss Performance of LSTM vs. KAN Models

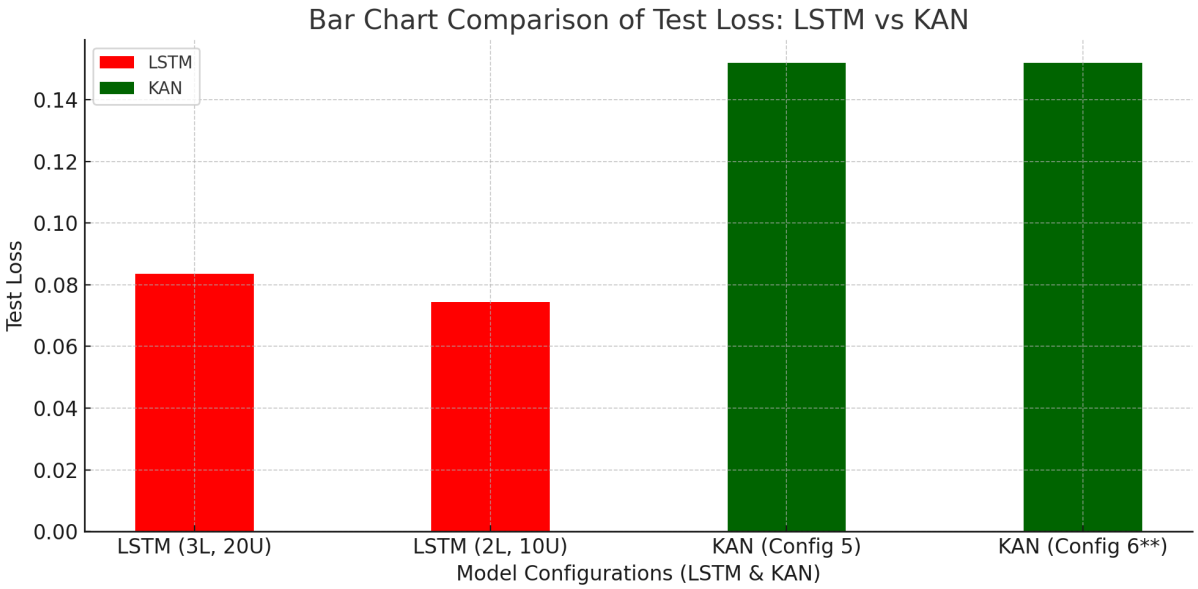


**Figure 4.3:** Bar Chart Comparison of Test Loss: LSTM vs. The ratio of minimum test RMSE KAN vs LSTM is $\frac{0.152}{0.0745} \approx 2.04$

The bar chart (Figure 4.3) shows that LSTM models significantly outperformed KAN models in minimising test loss. The *LSTM (3 layers, 20 units)* and *LSTM (2 layers, 10 units)* configurations demonstrated low test losses of approximately 0.0835 and 0.0745, respectively, with the simpler 2-layer model achieving the best results. This indicates that LSTM models effectively balance complexity and generalisation, even with fewer layers and units.

In contrast, the *KAN Config 5* and *KAN Config 6* models had higher test losses around 0.152, suggesting potential challenges with overfitting or suboptimal configurations. This highlights that LSTM models, known for handling sequential data well, are better suited for minimising test loss in this context.

Overall, LSTM models, especially simpler configurations, are preferable for tasks requiring lower test loss. However, it is worth noting that LSTM is a specialised version of RNN designed to work with time series data, while the KAN used here is a simple and general use KAN, even then the Test loss is not too terrible, highlight it's potential for use in forecasting, given specialised models are developed.

Although overall LSTM models, particularly in simpler configurations, are preferable for

tasks that prioritise achieving lower test loss. However, it is important to note that LSTM is a specialised variant of RNN, specifically designed for time series data and the KAN used here is a more general-purpose model. Despite this, the KAN's test loss remains reasonably low, highlighting its potential for forecasting applications, especially if specialised KAN models are developed for this purpose

# Chapter 5

# Conclusion and Future Directions

## 5.1 Conclusion

This study investigated the efficacy of various forecasting models on stochastic data, specifically focusing on the stock market. The primary aim was to assess the predictive capabilities and interpretability of traditional statistical methods like random walks, naive forecasting, ARIMA etc, and deep learning models like LSTM, and emerging architectures like the Kolmogorov-Arnold Network (KAN). Each model provided unique insights into stock price movements, balancing between accuracy and interpretability.

Statistical forecasting demonstrated its limitations with stochastic data due to its rigid structure, resulting in lower accuracy compared to deep learning models. LSTMs excelled in capturing sequential patterns, offering high predictive performance; however, their "black-box" nature posed challenges in understanding model behaviour, impacting interpretability. KANs, built on the Kolmogorov-Arnold Theorem, provided a promising alternative by balancing accuracy with a structured and interpretable model framework, though their accuracy was somewhat lower than LSTM in this context.

## 5.2 Future Research Directions

For future research, we offer the following two proposals:

- To use autoencoders [35] to reduce the dimensionality of time series data [36] which than then be fed into KAN-based models as per [1], potentially improving their efficiency and performance in stochastic time series data. Autoencoders compress the time series data into lower dimension for representation, essentially it captures the important and meaningful features while disregarding the noise in the data.
- To develop specialised models which are based on KAN architecture for stochastic time series data, similar to how LSTM, RNN models etc are based on the MLP architecture. This could include KAN architecture based LSTMs, transformers(so called KANSFORMERS [1]) etc.

# Appendix

You can access the project repository on GitHub by following this link:

https://github.com/tabishalirather/grand_challenges_2024/tree/master

**Note on Statistical Forecasting Results**

This report references statistical forecasting methods but does not include any results from these methods. Unfortunately, this is due to the lack of response and submission from a team member, Jack Ruzeu. Despite multiple attempts to reach out—both by our team and by our supervisor, Dr. Nadezda Sukhorukova—Jack did not respond to our repeated requests for his contributions.

Jack only provided his report in PDF format around 1:13 p.m. on the day of submission, 3rd November 2024. When requested to submit his work in LaTeX format to maintain consistency. However, he indicated that he would not be able to do so. Our supervisor, Dr. Nadezda, is fully aware of this situation. We acknowledge Jack's contribution to the literature review of this report. Additionally, Jack has indicated that he will make a separate submission for his part.

We sincerely reg hope that my teammate and I are not penalised for this unfortunate issue, which was beyond our control. We have made every effort to ensure that the rest of the report meets the required standards, and we appreciate your understanding regarding this matter.

# Bibliography


[1] Z Liu, Y Wang, S Vaidya, F Ruehle, J Halverson, M Soljačić, TY Hou, and M Tegmark. Kan: Kolmogorov–arnold networks, 2024. arXiv preprint arXiv:2404.19756v4 [cs.LG].

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[4] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2nd edition, 2018. https://otexts.com/fpp2/.

[5] A Hirsa and S.N. Neftci. *An Introduction to the Mathematics of Financial Derivatives*. Academic Press, third edition edition, 2013.

[6] K. Siegrist. *Stochastic Processes*. LibreTexts, 2022.

[7] G.F. Lawler and V. Limic. *Random Walk: A Modern Introduction*. Cambridge University Press, 2010. Online publication 2012, accessed online August 29.

[8] Louis Bachelier. *The Theory of Speculation*. Annales scientifiques de l'École Normale Supérieure, 1900. Translated by D. May from Annales scientifiques de l' École Normale Supérieure.

[9] H. Lebesgue. Intégrale, longueur, aire. *Annali di Matematica*, pages 231–359, 1902.

[10] N. Freeman. Martingale theory and applications, 2015. University of Bristol, available at:.

[11] R. Getoor. J. l. doob: Foundations of stochastic processes and probabilistic potential theory. *Annals of Probability*, 37(5):1647–1663, 2009.

[12] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[13] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.

[14] WS McCulloch and WH Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–137, 1943.

[15] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[16] ML Minsky and SA Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, Cambridge, MA, 1969.

[17] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[18] Carnegie Mellon University. Introduction to deep learning lecture 1, 2020. Deep Learning Lecture, [online] YouTube.

[19] R Hecht-Nielsen. Theory of the backpropagation neural network. In R. Hecht-Nielsen, editor, *Neural Networks for Perception*, pages 65–93. IEEE, 1992. Based on "non-indent" by Robert Hecht-Nielsen, Proceedings of the International Joint Conference on Neural Networks, vol. 1, pp. 593–611, June 1989, © 1989 IEEE.

[20] P. Raut and A. Dani. Correlation between number of hidden layers and accuracy of artificial neural network. In *Algorithms for Intelligent Systems*, pages 513–521. Springer, Singapore, 2020.

[21] D Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8:437–479, 1902.

[22] AN Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR (Proceedings of the USSR Academy of Sciences)*, 114(5):953–956, 1957.

[23] F Girosi and T Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.

[24] V Kůrková. Kolmogorov's theorem is relevant. *Neural Computation*, 3(4):617–622, 1991.

[25] J. Wang et al. A deep learning approach for daily stock movement prediction using wavelet and attention-based lstm. *Quantitative Finance*, 19(9):1467–1487, 2019.

[26] A. Soleymani and U. Paquet. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder-deepbreath. *Expert Systems with Applications*, 156:113456, 2020.

[27] A. Koshiyama et al. Learning from the market: Data-driven market making via deep reinforcement learning. *Expert Systems with Applications*, 151:113343, 2020.

[28] S. Das, A. Tariq, T. Santos, S.S. Kantareddy, and I. Banerjee. Recurrent neural networks (rnns): Architectures, training tricks, and introduction to influential research. In *Neuromethods*, pages 117–138. Humana, New York, NY, 2023.

[29] X. Zhang et al. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(2):300–313, 2019.

[30] Ran Aroussi. Github - ranaroussi/yfinance: Yahoo! finance market data downloader (+faster pandas datareader). https://github.com/ranaroussi/yfinance, n.d. Accessed: October 29, 2024.

[31] Optax. L-bfgs example. https://optax.readthedocs.io/en/stable/_collections/examples/lbfgs.html. Accessed: 2024-11-03.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[33] Mohammad Moradi, Shahram Panahi, Erik Bollt, and Ying-Cheng Lai. Kolmogorov-arnold network autoencoders. Technical report, Arizona State University, Tempe, AZ, 2024.

[34] Johannes Schmidt-Hieber. The kolmogorov–arnold representation theorem revisited. *Neural Networks*, 137:119–126, 2021.

[35] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[36] William Todo, Beatrice Laurent, Jean-Michel Loubes, and Merwann Selmani. Dimension reduction for time series with variational autoencoders. *arXiv preprint arXiv:2204.11060*, 2022.