AI-Driven Predictions of Industrial Metal Prices on the London Metal Exchange

Optimizing Echo State Networks with Genetic Algorithms

Master's thesis in Engineering Mathematics and Computational Science

MARKUS EDLUND

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

AI-Driven Predictions of Industrial Metal Prices on the London Metal Exchange

Optimizing Echo State Networks with Genetic Algorithms

MARKUS EDLUND

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2025

AI-Driven Predictions of Industrial Metal Prices on the London Metal Exchange
Optimizing Echo State Networks with Genetic Algorithms
MARKUS EDLUND

Supervisor: Zoran Konkoli, Department of Microtechnology and Nanoscience
Examiner: Zoran Konkoli, Department of Microtechnology and Nanoscience

Master's Thesis 2025
Department of Microtechnology and Nanoscience
Division of Division name
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in $\LaTeX$
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

AI-Driven Predictions of Industrial Metal Prices on the London Metal Exchange
Optimizing Echo State Networks with Genetic Algorithms

MARKUS EDLUND
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

The battery manufacturer Northvolt aims to reduce inventory risks. In an increasingly competitive industrial metals market, protecting inventory value by employing hedging strategies is a key component of lowering costs. Using artificial intelligence for reliable predictions of short term prices is an interesting prospect for improving such strategies.

This study uses data from the London Metal Exchange, including cash and three-month futures contracts, along with stock volume, to predict next-day cash prices for six industrial metals. These predictions are compared to a baseline random walk model using the metrics mean squared error (MSE), mean absolute error (MAE), R^2, and directional accuracy.

Predictions are made using an Echo State Network (ESN), with optimized parameters chosen by a Genetic Algorithm (GA). The network is trained offline with ridge regression and online with stochastic gradient descent. The performance of the ESN-GA setup is validated using chaotic systems (Mackey-Glass and Lorenz equations) before being applied to metals futures data. The data is split into four different sets: a warmup set necessary for ESNs, a training set used for offline training, a validation set to measure GA performance, and a test set of unseen data to ensure the network generalizes well.

The GA optimization significantly reduced prediction errors in the Mackey-Glass and Lorenz systems, with MSE values improving from 1.2E-8 to 5.3E-12 and from 3.3E-2 to 8.8E-7, respectively, on the validation set. For metals futures data, the GA enhanced ESN performance, outperforming the random walk across all metrics on the validation set for all six metals. However, slight modifications were necessary to achieve superior performance on the unseen test set. Superior results were achieved for four metals across all metrics, while the results for the remaining two metals were mixed.

The GA effectively optimizes ESN parameters for systems with clear, deterministic dynamics but encounters challenges when applied to stochastic futures data, particularly due to the risk of overfitting the validation set. While the ESN-GA method does not consistently outperform the random walk on unseen test data, it demonstrates significant potential. Further exploration with alternative configurations, additional data types, and more robust validation techniques is warranted to enhance its practical applicability.

Keywords: Artificial Intelligence, Echo State Networks, Genetic Algorithms, Northvolt, London Metal Exchange, Metal Futures, Financial Time Series, Random Walk

Acknowledgements

I would like to express my heartfelt gratitude to several individuals and organizations who have supported me throughout this journey.
First and foremost, I would like to thank my supervisor and examiner, Zoran Konkoli, for his patience, guidance, and encouragement in allowing me the freedom to explore reservoir computing according to my interests and aspirations. His support has been invaluable throughout this process.
I am deeply grateful to Denver Stretesky for serving as my opponent and for providing great feedback and insightful suggestions that have greatly improved the quality of this work.
A special thanks goes to Frederic Sese and Axel Myrberger at Northvolt for giving me the opportunity to work on such an engaging and meaningful problem. Despite challenging times, their enthusiasm has never wavered.
Finally, I extend my thanks to everyone who contributed directly or indirectly to the completion of this project.

Markus Edlund, Gothenburg, January 2025

List of Acronyms

DA	Directional Accuracy
ESP	Echo State Property
ESN	Echo State Network
GA	Genetic Algorithm
GD	Gradient Descent
LME	London Metal Exchange
MAE	Mean Absolute Error
MSE	Mean Squared Error
RC	Reservoir Computing

Contents

1 Introduction

In an increasingly competitive industrial metals market, protecting inventory value is critical for industrial actors like Northvolt, one of Europe's fastest-growing battery manufacturers [70]. Materials account for approximately 65% of the total production costs of electric vehicle batteries [45], therefore delivery contracts of batteries usually include a moving prices of said materials [80]. This exposes the whole supply chain to significant financial risk if prices are volatile, highlighting the need for effective mitigation strategies. To address this challenge, hedging strategies involving futures contracts traded on the London Metal Exchange (LME), are employed to stabilize costs and protect inventory value [55][46][75].

The LME is the world's primary commodities exchange for industrial metals, with $15 trillion traded in 2023 [53]. Futures contracts allow traders to lock in prices for metals at a future date, enabling them to offset potential losses caused by price fluctuations [55]. For instance, selling futures while holding an equivalent physical inventory can balance the financial impact of price changes. In this context, accurate short-term price predictions, particularly for LME cash contracts, could improve hedging strategies and inventory management.

Predicting financial markets, however, is notoriously challenging. The efficient market hypothesis suggests that market prices reflect all available information, causing price movements to behave like a random walk, that is inherently unpredictable [21]. Yet, skepticism remains. Some studies propose that financial time series exhibit chaotic dynamics rather than pure randomness, meaning there may be underlying patterns that can be identified and modeled [62]. If chaotic behavior exists, models adept at predicting chaotic systems might offer a competitive edge over other models.

To address this, this thesis explores the use of artificial intelligence for predicting short-term cash prices of industrial metals. Specifically, the study applies an Echo State Network (ESN) to forecast next-day cash prices of six metals traded on the LME. The ESN's parameters are optimized using a Genetic Algorithm (GA), a nature-inspired optimization technique [66]. The performance of the ESN-GA model is evaluated against a random walk baseline, using the metrics mean squared error (MSE), mean absolute error (MAE), R^2, and directional accuracy.

1.1 Research Questions

This thesis aims to answer the following primary research question:

- **Can Echo State Networks (ESNs), optimized using a Genetic Algo-**

rithm (GA), outperform a random walk model in predicting next-day metal futures prices?

To address this, the following secondary questions are considered:

- Is the GA effective at improving the ESN's predictive performance?
- What are the computational costs and time requirements for using a GA to optimize ESNs?
- How robust is the GA-optimized ESN model for different metals and across various datasets?
- What patterns or insights emerge from the ESN model's predictions that could inform trading strategies?

1.2 Overview of Methods

The study uses historical data from the LME, including cash contracts, three-month futures contracts, and stock volume for six industrial metals: aluminium, copper, lead, nickel, tin, and zinc accessed through Westmetall [86]. The data is split into warmup, training, validation, and test sets. The ESN, a type of recurrent neural network, is constructed to predict next-day cash prices. Its parameters are tuned using a GA, which iteratively evolves the network design to minimize prediction error. The ESN-GA model's performance is validated for chaotic systems using the Mackey-Glass and Lorenz Equations. Finally the performance of the ESN-GA model is tested on the industrial metals and benchmarked against a random walk model.

1.3 Significance of the Study

This study contributes to the fields of machine learning and financial time series forecasting by exploring the effectiveness of ESN-GA models for industrial metal price predictions. It also bridges the gap between theoretical AI models and practical financial applications, offering insights that could enhance hedging strategies for industrial actors like Northvolt. By evaluating the ESN-GA model against the random walk, this research provides a rigorous test of the model's predictive capabilities often lacking in many similar studies on using AI for financial time series predictions [73][83][13][58].

2

Background

This section established the practical context and briefly introduces the theoretical frameworks and machine learning techniques that will be applied in the thesis.

2.1 Northvolt and the London Metal Exchange

Northvolt is a Swedish battery manufacturer and was the fastest-growing producer in Europe 2023, as highlighted in their sustainability and annual report [70]. The company's value proposition is centered around producing the world's greenest batteries with industry-leading performance, enabling customers to transition to electric solutions. To manufacture these batteries a wide range of metals and materials are required. This exposes Northvolt to fluctuations in metal prices. As part of its strategy to manage this risk Northvolt incorporates recycling into its processes [71]. In 2023, 6% of the materials in their battery cells were recycled, with an ambitious target of 50% by 2030 [70].

Materials account for a significant portion of the costs in battery manufacturing, approximately 65% according to industry estimates for electric vehicle batteries [45]. While this estimate is not specific to Northvolt, material costs are critical for any battery manufacturer, emphasizing the importance of intelligent procurement strategies. While well structured supply chains are probably the most important part of driving costs down, an important complement is the use of financial tools such as futures [75] [46].

The London Metal Exchange (LME) is the world's primary commodities exchange for industrial metals. In 2023, it handled trading volumes of 3.5 billion tonnes, and a notional trading value of $15 trillion [20]. The LME facilitates the majority of global financial business related to non-ferrous metals, i.e. metals that do not contain iron, and prevalent in batteries.

One of the LME's key functions is enabling the trading of futures contracts [55]. Futures are agreements that obligate buyers and sellers to transact a specified quantity of a commodity at a predetermined price on a future date [31]. Notably, traders do not need to physically own the underlying metal to trade futures. They can offset contracts by entering equivalent contracts with matching terms on the opposite side. Upon contract expiration, the LME clearing house records a net-zero position, and the trader realizes a profit or loss based on the difference between the purchase and sale prices of the contracts and no metals have to be delivered or received [52].

Futures contracts are an effective tool for reducing inventory risk without the need to buy or sell additional metal. For example, buying a certain amount of a metal and

simultaneously selling an equivalent quantity in futures contracts offsets potential inventory losses due to declining metal prices, as these losses are balanced by gains in the futures contracts. Conversely, increases in inventory value are offset by losses in the futures position. Something similar to this approach is widely employed by industrial players, including Northvolt, to mitigate inventory risks [80].
This study uses two types of LME contracts as data: cash contracts and three-month contracts.

- **Cash** contracts are the shortest-duration contracts on the LME [51]. Once a cash contract is entered, it cannot be offset with another contract because no equivalent contract exists to take an opposing position. These contracts effectively determine the final cash value for traders who do not wish to settle their positions physically. The next-day cash price is the primary focus of the predictive model in this thesis.
- **Three-month** contracts are the longest contracts that are settled daily on the LME [51]. As the name implies, they mature three months from the purchase date. These contracts offer insight into market expectations of future price movements. This relationship may hold predictive power for next-day cash prices.

The third type of data used in this study is **stock** data. The LME authorizes a global network of warehouses for storing metals traded on the exchange [54]. When a futures contract is settled, sellers issue warrants for the agreed-upon quantity of metal, which buyers can redeem at these warehouses. Since prices are fundamentally driven by supply and demand, information about available metal stock may provide valuable insights for price predictions.

2.2 The Random Walk and Chaos in Finance

The dynamics of financial markets are complex. Unlike physical systems governed by constant fundamental laws, a major contributor to financial market outcomes are the anticipations of its participants [65]. Adding to this complexity is bad input data, extraordinary events and malevolent actors. This makes price prediction a daunting challenge. If financial market prices were predictable using readily available formulas, these formulas would create a self-correcting effect: market participants would exploit the predictions, and subsequent prices would quickly reflect the expected outcomes. This leads to the notion that prices fully incorporate available information, with price movements occurring as independent random events when new information emerges. This concept is encapsulated in the Efficient Market Hypothesis (EMH), which states that: *"A market in which prices always fully reflect available information is called efficient"* [21].
The EMH is not meant to be taken literally and recognizes varying degrees of market efficiency. Empirical tests of market efficiency are typically categorized as [21]:

1. **Weak**: Only historical prices are considered available information.
2. **Semi-strong**: All publicly available information is considered.
3. **Strong**: All information, including non-public data, is incorporated.

While the validity of the Efficient Market Hypothesis remains a subject of debate

[91], AI models often struggle to outperform a random walk in short-term predictions, particularly when relying solely on historical price data [58].

2.2.1 Is the LME an Efficient Market?

There is a case to argue that the LME may not fully align with the definition of an efficient market. The sufficient conditions for market efficiency include [21]:

1. No transaction costs in trading securities.
2. All available information is costlessly accessible to all market participants.
3. All participants agree on the implications of current information for prices.

The LME has transaction costs [48], and historical data is not freely available [49], posing barriers for some participants, such as the author of this thesis. However, in practice the only actors directly trading on the LME are institutional, making the economical barriers to information and transactional costs negligible [50]. Regarding the last condition, it would be incredibly difficult to show that every single participant agrees on anything and is entirely out of scope for this thesis. While not technically fulfilling the sufficient conditions for a market to be considered efficient, it is certainly possible that the LME is. A brief older study on the LME copper market concluded that it probably is, despite statistical evidence being inconclusive [30].

2.2.2 The Random Walk Model

The predictive model derived from the weak form of the EMH is the random walk. A time-discrete random walk has two key properties [21]:

1. Changes in price are independent of each other.
2. Changes in price are identically distributed.

The random walk (RW) model is represented as:

$$S_t = S_{t-1} + \epsilon_t$$

where $\S_t$ is the price at time t and ϵ_t is a noise term with $\mathbb{E}[\epsilon_t] = 0$.
This implies that the expected value of the next day's rate of change (r_{t+1}) given the available information (I_t) is:

$$\mathbb{E}[r_t \mid I_{t-1}] = 0$$

where I_{t-1} represents the information set containing historical data.
Extensions and variations of the RW assumption is widely used, with applications such as the Black-Scholes model for option pricing where log returns are assumed to be independent and normally distributed [6]. The RW model in this thesis only assumes that expected next day returns are zero, thus the best prediction of the next day price is the same price as today.

2.2.3 Beyond the Random Walk: Chaotic Dynamics

Despite its prevalence, the random walk model has its skeptics. Some research has found that financial time series are better described as chaotic rather than random

[62]. Unlike randomness, chaos implies deterministic yet complex patterns that can be modeled. Accurate modeling of chaotic systems requires precise measurements of all relevant variables, which may be infeasible in the case of financial time series. Regardless, using some of the relevant variables might produce more accurate predictions than a random walk.
Chaotic systems, such as the Mackey-Glass and Lorenz equations, are examples of differential equations describing the dynamics of complex systems [63] [56]. These models will be used in this thesis to demonstrate the ability to predict chaotic systems.
The reasoning is that if the LME futures market exhibits chaotic behavior, a model designed for chaotic systems should outperform the random walk. Instead of debating whether markets are inherently chaotic or efficient, this thesis adopts a pragmatic approach: constructing a model that excels at chaotic prediction and then comparing its performance with the random walk model.
With some of the inherent challenges of financial time series forecasting established, the next section introduces the model employed in this thesis.

2.3 The ESN-GA Model

Echo State Networks (ESNs) are a category of Reservoir Computing (RC) models that have shown excellent performance in predicting chaotic time series [19][41]. One of ESN's key advantages is its limited training requirements. This is achieved through the construction of the reservoir, a high-dimensional, non-linear space of sparsely connected nodes whose weights and connections remain fixed [38]. Instead of training the entire network, only a linear readout layer is trained to interpret the various states of the reservoir nodes. Since this readout layer is linear and non-recurrent, efficient regression techniques can be employed.
The primary challenge in constructing ESNs lies in selecting the parameters defining the reservoir [57]. Questions such as "How many nodes should be used?", "How should they be connected?", and "How should the input be scaled?" are critical to the network's performance. A common approach involves relying on expertise or intuition, essentially educated guessing. When such expertise is unavailable, a systematic and automated method becomes necessary. In this thesis, parameters are selected using a Genetic Algorithm (GA).
A GA simulates the process of evolution [66]. Central to evolution is the concept of survival of the fittest, which measures how well an individual adapts to its environment. In the context of GAs, fitness is quantified using a mathematical function that evaluates how well an individual performs a given task. Here, an individual represents a set of parameters defining an ESN model.
The GA process begins by randomly initializing a population of individuals, the first generation, within a defined search space. Each individual is evaluated based on its fitness, which, in this context measures the ESN's predictive performance on LME cash contracts. To improve the next generation, evolutionary operators such as selective breeding and random mutation are applied:

- **Selective breeding**: Individuals with higher fitness are more likely to be selected for reproduction, creating new offspring by combining the traits of

two selected parents.
- **Random mutation**: Introduces novel variations by randomly altering traits. While often disruptive, mutation is crucial for maintaining genetic diversity and discovering new, potentially superior solutions.

This process of selection, crossover, and mutation is repeated over multiple generations, allowing the population to converge toward optimal or near-optimal solutions. Compared to random or grid search methods, GAs often identify high-performing solutions more efficiently [85].
The nested optimization loop, where the GA optimizes the parameters of the ESN whose readout layer is then trained, form the ESN-GA model proposed in this thesis. This combination leverages the predictive power of ESNs and the optimization capabilities of GAs to address the challenges of parameter selection systematically.

2.4 Related Work

Having introduced the relevant topics of the thesis it is worth looking into whether anything similar has been done before. The concept of reservoir computing is not new. It was formally introduced by Herbert Jaeger in 2001 [38]. Similar models were known before but Jaeger's work is the foundation of what RC is today [40]. Since then a multitude of effective applications have been found [12]. There are also multiple RC libraries available for ease of implementation [68] [15] [16].
However, when pairing the RC model with a GA, using existing libraries can be cumbersome and clunky. Instead a low level custom implementation is favored allowing for control and larger understanding of network design decisions. Creating a low level implementation of RC can be challenging but the work by Lukoševičius, specifically *A Practical Guide to Applying Echo State Networks*, is a vast resource [57]. The guide is in fact cited in most of the existing libraries.
Others have used a genetic algorithm to optimize the parameters of an RC model with positive results [29] [11]. There are also implementations of RC networks specifically for financial time series [84] [89] [88], however none known to the author for predicting metal futures.
Using ESNs, or reservoir computing for that matter, for financial time series predictions is quite niche and there are seemingly no comprehensive literary reviews on the topic. Exemplifying this is the omission of RC models from reviews of deep learning for financial time series predictions [13] [83] [73]. It should also be noted that data from non-publicly traded metal futures, such as the LME data, is also never mentioned in any of the reviews. Further, using the random walk as a benchmark is not common. In the respective 72, 85 and 140 studies in these reviews, using the random walk as a comparative baseline only appears twice, even though the efficient market hypothesis is commonly cited. The explanation for this might be found in a study that replicated 15 best in class deep learning models found in other studies [58]. None of the models replicated in the study managed to outperform the *naive approach*, called random walk model in this thesis, based on mean squared error over multiple time periods.

3
Theory

This section handles the theory required to construct an ESN, optimize its parameters using a GA and produce predictions.

3.1 Network Topologies

A network topology refers to the arrangement or organization of nodes and connections within a network. It is particularly relevant to this subject due to its significant impact on network performance [36] [59] [60].
This section outlines the two different network topologies utilized in this study. The algorithms used create simple adjacency matrices. Such matrices describe in binary terms how the nodes in the network are connected. A one represents a connection, and subsequently a zero represents no connection. The matrices are directional and self-connecting, that is, node a can be connected to node b while node b does not necessarily need to be connected to node a, also node a can be connected to itself. Let A be an $n \times n$ adjacency matrix where the elements A_{ij} are defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ is adjacent to } j, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for all } i, j \in N.$$

After the simple adjacency matrix is generated, each of the non-zero elements are then replaced to fulfill the echo state property, this is handled later. The new matrix defines all the connections and corresponding weights in the reservoir, hence called the weight matrix.
Below are descriptions of the two algorithms used for generating the simple adjacency matrices.

3.1.1 Scale-Free Network

The reason for generating scale free networks in this context is to mimic the complex structure of networks occurring in nature, such as the neural network in brains [4]. Scale-free networks can be generated using the Albert-Barabasi model [5] and is achieved through so called preferential attachment. The process follows two general steps: first generate a small number of connected nodes, second add subsequent nodes with preferential attachment. That is, add connections with a proportional probability based on the degree of each node over the total degree of the network [5]. This approach leads to an undirected network without self connections so a slightly

edited approach inspired by [69] and [3] is taken to make the network directed and self connecting. The process follows:
The algorithm begins by initializing a clique C of size 2, consisting of two fully connected nodes. This is represented by an adjacency matrix $A \in \{0,1\}^{n \times n}$, where $A_{ij} = 1$ for $i, j \in \{0, 1\}$, and $A_{ij} = 0$ otherwise.
Define the set of all nodes as $N = \{0, 1, \ldots, n-1\}$, where n is the total number of nodes in the network. The initial clique $C \subseteq N$ contains two nodes, and the set $S = N \setminus C$ represents the nodes that are not initially part of the clique.
For each node $s_k \in S$, the algorithm sequentially adds s_k to the clique C, so that $C \leftarrow C \cup \{s_k\}$.
Once s_k is added to C, edges between s_k and the nodes in C are established based on preferential attachment. Specifically, for each node $s_i \in C$, the following steps are performed:

1. Sample two random values $r_1, r_2 \sim \mathcal{U}(0, 1)$, where $\mathcal{U}(0, 1)$ represents the uniform distribution on the interval $[0, 1]$.

2. Compute the degree $\deg(s_i)$ of node s_i, which is the sum of the entries in the i-th row of the adjacency matrix A:

$$\deg(s_i) = \sum_{j=0}^{n-1} A_{ij}$$

3. Let the total degree of the nodes in C be:

$$\deg(C) = \sum_{i \in C} \deg(s_i)$$

4. Define the probability of connecting s_k to s_i as:

$$P(s_k \to s_i) = \frac{\deg(s_i)}{\deg(C)}$$

If $r_1 < P(s_k \to s_i)$, then set $A_{ik} = 1$ (i.e., create a directed edge from s_k to s_i).

5. Similarly, define the probability of connecting s_i to s_k as:

$$P(s_i \to s_k) = \frac{\deg(s_i)}{\deg(C)}$$

If $r_2 < P(s_i \to s_k)$, then set $A_{ki} = 1$ (i.e., create a directed edge from s_i to s_k).

If, after iterating through all nodes in C, node s_k has not been connected to any other nodes, i.e. no entries in row $A_{k\cdot}$ or column $A_{\cdot k}$ are set to one, the process is repeated until s_k is connected to at least one node in C.
This process is repeated for each node $s_k \in S$ until all nodes are part of the clique C and A defines the scale-free simple adjacency matrix.

3.1.2 Uniformly Random Networks

Generating uniformly random networks requires less steps and is commonly used in ESNs [36]. For every entry in the adjacency matrix a uniformly random number between 0 and 1 is sampled. If the random number is below a specified probability P the entry in the adjacency matrix is put to one, else zero. That is:
Let A be an $n \times n$ adjacency matrix where the elements A_{ij} are defined as follows. For every $i, j \in N = 0, 1, ..., n-1$, let $p \sim U(0,1)$, where $U(0,1)$ represents the uniform distribution on the interval $(0,1)$. Then, the elements A_{ij} are defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } p < P, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for all } i, j \in N.$$

Thus a uniformly random topology requires a connectivity parameter P. A defines the uniformly random simple adjacency matrix.

3.2 Theory of Reservoir Computing Models

Reservoir computing is a recurrent neural network framework designed for , among other things, processing temporal data and solving complex dynamical problems [26]. It consists of a fixed, randomly initialized set of recurrent nodes the so-called reservoir, which captures and transforms input data into high-dimensional space. All nodes in the reservoir record states, which are then interpreted by a readout layer. A key feature is that only the readout layer is trained. This makes the approach computationally efficient, considering the training becomes a least squares problem, while still capable of learning complex patterns [79]. This architecture is particularly well-suited for tasks like forecasting, pattern recognition, and modelling chaotic systems [26][41], which financial markets could be claimed to be a part of [33]. Regardless, there are promising results for utilizing reservoir computing models within financial markets to make predictions [33][84].
Following is an abstract outline of how ESNs operate as described in [61]. In short, the process involves handling input to produce output and then optimize the output towards a goal:
Form two separate sequences:

$$D = \{x_t, y_t\}_{t=0}^{T-1}$$

where:

- The set D is the training data.
- x_t is the input signal at time t
- y_t is the output signal at time t
- There exists some relationship between the two signals.

The goal is to learn the relationship between x and y from D, and then, after the training period (for $t \geq T$), be able to give estimates $\hat{y}_t$ for an unknown y_t given some new input x_t.
When performing a time series prediction, the goal is often to predict $x_{t+\tau} = y_t$ given $\{x_i\}_{i \leq t}$, which is called a τ-step ahead prediction. Now that the purpose of

an ESN has been established, following is an abstract illustration of the network structure:

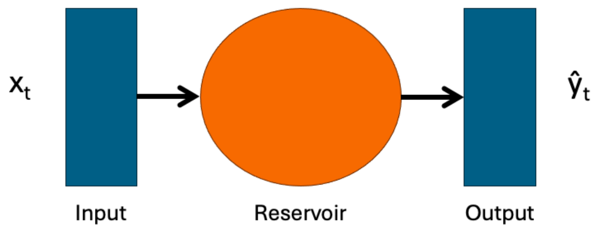


Figure 3.1: Illustration of a Reservoir Computing Model

A general reservoir computing model, as illustrated above, consists of the input and output layers (sometimes called the input and readout layer) connected to the n-dimensional dynamical space called the reservoir. The reservoir can be described as follows:

$$S_t = F(S_{t-1}, x_t)$$

where

- $S_t \in \mathbb{R}^n$ is the state of the reservoir at time t, which consists of $\{s_{i,t}\}_{i=0}^{n-1}$, the individual states of the reservoir nodes.
- $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is the map describing the dynamics of the reservoir.

This means that the state of the reservoir S_t evolves over time according to the dynamics F and the input x_t.
The approximation $\hat{y}_t$ of y_t is calculated from the reservoir as follows:

$$\hat{y}_t := \sum_{i=0}^{n-1} w_i s_{i,t}$$

That is, reading the states of the nodes in the reservoir and scaling them by the output weights $\{w_i\}_{i=0}^{n-1}$.

Choosing the output weights is done by making the approximation as close to the output signal as possible in the training data D. That is, $\hat{y}_t \simeq y_t$, for $t \in [0, T-1]$. Typically, this is done by a least squares method.

3.2.1 The Echo State Property

For the functionality of the ESN it is essential that the state of the network is uniquely determined by any left-infinite sequence of inputs [32]. This means that, regardless of the initial state of the network, the current state is fully determined by the infinite history of past inputs, ensuring that the influence of any arbitrary initial conditions vanishes over time. Guaranteeing this can be done through the echo state property (ESP). A few formal definitions for the ESP have been proposed depending on restrictiveness, slight differences in the ESN and practicality [32] [37][38][39]. Following is the definition offered by [32]: Being uniquely determined by any left-infinite sequence depends on the input sequence. Therefore, the definition of the ESP is stated through constraining the input sequence's range to a compact set, let's denote that $\mathbf{X}$. We also need to relate $\mathbf{X}$ to the set of all possible states of the nodes in the reservoir, denote that $\mathbf{S}$. This leads to the compactness condition which states:

Let F be the map defining the dynamics of the reservoir.
- F defined on $\mathbf{S} \times \mathbf{X}$
- $\mathbf{S} \subset \mathbb{R}^n$
- $\mathbf{X} \subset \mathbb{R}^t$
- $\mathbf{X}$ and $\mathbf{S}$ are compact sets
- $F(\mathbf{S}_{t-1}, x_t) \in \mathbf{S},\ x_t \in \mathbf{X}$ for any $t \in \mathbb{Z}$

Now, $\mathbf{S}$ being compact is guaranteed when the activation function of the reservoir nodes f is bounded. A common example of a bounded activation function is the hyperbolic tangent $tanh$.
Practically speaking the inputs x_t will also be bounded making $\mathbf{X}$ a compact set.
Now, let $\mathbf{X}^{-\infty}$ and $\mathbf{S}^{-\infty}$ denote the left infinite input and state vectors respectively defined as:

- $\mathbf{X}^{-\infty} := \{x^{-\infty} = \{x_i\}_{i=-\infty}^{0} \mid x_t \in \mathbf{X} \text{ for any } t \leq 0\}$
- $\mathbf{S}^{-\infty} := \{S^{-\infty} = \{S_i\}_{i=-\infty}^{0} \mid S_t \in \mathbf{S} \text{ for any } t \leq 0\}$

$S^{-\infty}$ is compatible with $x^{-\infty}$ when:

- $S_t = F(S_{t-1}, x_t),\ \forall t \leq 0$

Equipped with this, the definition of the ESP follows:

A network with dynamics:

- $F : \mathbf{S} \times \mathbf{X} - > \mathbf{S}$

satisfying the compactness condition has the echo state property for any input sequence:

$x^{-\infty} \in \mathbf{X}^{-\infty}$

and any two state vector sequences:

$S^{-\infty}, K^{-\infty} \in \mathbf{S}^{-\infty}$

Compatible with $x^{-\infty}$.

It holds that $S_0 = K_0$.

This definition implies that states that are nearby must be the result of similar input histories [8]. Subsequently the current state should depend more on recent inputs and states rather than historic ones.
In practice one way of achieving the echo state property is by assuming a compact set of input vectors, which almost always holds, and adjusting the reservoir weights

matrix accordingly.
So, for the standard ESN, construct the reservoir weights matrix W^{res} in the following manner to guarantee the ESP [32]:

1. Randomly initialize $W^{res} \in \mathbb{R}^{n \times n}$ such that all entries $w_{ij}^{res} \geq 0$, i.e., they are non-negative.
2. Scale W^{res} such that the spectral radius is less than 1, i.e., $\rho(W^{res}) < 1$.
3. Change the sign of a desired number of weights w_{ij}^{res} so negative connections are introduced.

This guarantees the ESP. A few ways to prove this can be found in [32][39][38].
It should be noted that as explicitly stated in [40] and as shown in [32] that randomly initializing W and then scaling the spectral radius below unit is not necessary for the ESP and spectral radii above unity might be optimal for the specific application.

3.2.2 Calculating Output

Previously the calculation of the output, the approximation $\hat{y}$, was described as multiplying the states of the reservoir by the weights of the output layer, that is a simplification. This section in turn outlines the full calculation from input to output. To start with, we define all the different vectors, matrices and activation functions used to define the dynamics of the reservoir F subsequent calculation of the output.

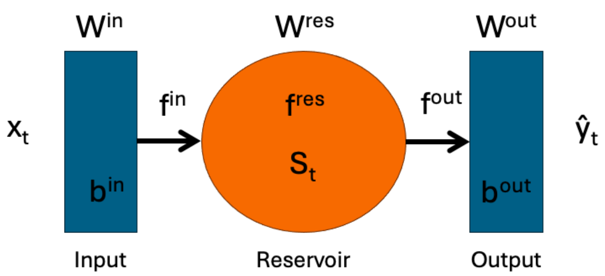


Figure 3.2: Illustration of Reservoir Computing Model, Complemented with Vectors, Matrices and Activation Functions used to Define the Dynamics of the Reservoir

First we have the matrices:
- $W^{in} \in \mathbb{R}^{k \times n}$, where $\dim(x_t) = k$, the number of dimensions to the input signal.
- $W^{res} \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the reservoir.
- $W^{out} \in \mathbb{R}^{n \times m}$, where $\dim(y_t) = \dim(\hat{y}_t) = m$, the number of dimensions of the output signal and approximation.

Second the bias vectors:
- $b^{in} \in \mathbb{R}^n$
- $b^{out} \in \mathbb{R}^m$

Note that not necessarily all the nodes in the reservoir are connected to the input or output layers. For the connected nodes the entries in the corresponding weight matrix and bias vector are non-zero, correspondingly for non-connected nodes the entries are zero.

Third the state vector of the reservoir

- $S_t \in \mathbb{R}^n$, the reservoir state at time t
- $S_{t,i}$ denotes the state of node i at time t

And fourth the three activation functions:

- f^{in}, the element-wise activation function of the input layer. Usually just the unity.
- f^{res}, the element-wise activation function of the reservoir. Usually the hyperbolic tangent.
- f^{out}, the element-wise activation function of the output layer. Usually unity or a rectifier.

Last, as suggested in [68], [27] and [10] we introduce a leakage rate, $\alpha \in (0, 1]$. The term helps control how fast the state of the reservoir changes with new input, allowing learning of slower dynamical systems and recognizing strongly time warped patterns [27].
With all terms defined, the first step is to calculate S_t using the dynamics of the reservoir F driven by x_t:

$$S_t = F(S_{t-1}, x_t)$$

Where the map describes:

$$S_t = (1-\alpha)S_{t-1} + \alpha f^{res}\left(W^{res}S_{t-1} + f^{in}\left(W^{in}x_t + b^{in}\right)\right)$$

With the updated states S_t, we can calculate the output $\hat{y}$:

$$\hat{y}_t = f^{out}(W^{out}S_t + b^{out})$$

Having calculated the output, it can be used to train the output weights and or make predictions, discussed in the following sections.

3.2.3 Offline Learning: Ridge-Regression

Offline training entails training the model on a fixed historical dataset.
The only coefficients in the model that are being optimized are W^{out} and b^{out}. Because f^{out} is typically linear the calculation of the output can be written as follows:

$$\hat{y}_t = \bar{W^{out}}\bar{S}_t$$

where $\bar{W^{out}} = [W^{out}; b^{out}]$ and $\bar{S}_t = [S_t; 1]$, allowing for simpler notation.
The purpose of the ESN is to match the input-output pair $[X, Y]$. This is done by feeding x_t into the reservoir, updating the states S_t and calculating the output $\hat{y}_t$ as above, where $\hat{y}_t$ should be as close to y_t as possible. To achieve this one can gather all the states produced by X and minimize the squared error with respect to the weights and bias. Assuming we have T time steps of training data we want to solve the following equation for $\bar{W}^{out}$:

$$Y = \bar{W}^{out}\bar{S}$$

where $Y \in \mathbb{R}^{m \times T}$, $\bar{S} \in \mathbb{R}^{(n+1) \times T}$, and $\bar{W}^{out} \in \mathbb{R}^{(n+1) \times m}$.
Since the system is usually overdetermined, $T >> (n+1)$, there are many known solutions. Using a ridge regression is the most universal and stable, which is the following equation [57] :

$$\bar{W}^{out} = Y\bar{S}\left(\bar{S}\bar{S}^T + \lambda I\right)^{-1}$$

where T denotes the transpose, I is the identity matrix, $(\cdot)^{-1}$ represents the matrix inverse, and λ is the regularization coefficient.
The inclusion of λ is made to penalize large absolute values of $\bar{W}^{out}$. If $\bar{W}^{out}$ has very large absolute values this might amplify tiny differences in $\bar{S}$. This means possibly worse generalization for out of sample inputs that produce never before seen states and potentially massive errors. Picking λ such that this does not happen is important and therefore included as a parameter that needs optimization.

3.2.4 Online Learning: Stochastic Gradient Descent

Online learning enables the model to adapt to new data. This is desirable in cases where the time series X is non-stationary [72] [90], which financial time series can be. Online learning through stochastic gradient descent (SGD) enables this by, at each time step, shifting the weights and bias such that the squared error becomes smaller.
SGD is formulated as follows [67]:
The objective of SGD is to minimize the average value of a function L:

$$L(\theta) = \mathbb{E}_{q(p)}[L(\theta, p)]$$

Where θ is the parameters we want to optimize and p is a random input to L. For example, an approximation from a training set.
At each step of the optimization, we assume that we observe:

$$L_t(\theta) = L(\theta, p_t)$$

And $p_t \sim q$.
We also assume that an unbiased estimate for the gradient of L can be computed. That holds if the distribution of $q(p)$ is independent of θ and we can use:

$$g_t = \nabla_\theta L_t(\theta_t)$$

With that we can follow the negative gradient and iteratively update the parameters:

$$\theta_{t+1} = \theta_t - \eta_t g_t$$

Where η is the learning rate determining how far the step in the negative gradient direction the update should be. Because the method approximates the error space by a hyper plane and traverses it in the steepest instantaneous direction, η needs to be optimized. If picked too large the new weights will overshoot the optimal solution and too small never adapt to a non-stationary time series. The choice of η is handled in the methods section.

Now, θ represents the readout layer, that is W^{out} and b^{out}. The calculation of the gradient of the loss function with respect to the two is also handled in the methods section.

3.3 Theory of Genetic Algorithms

Thus far a few parameters needed to construct and train ESNs have been introduced. This section outlines some theory of genetic algorithms (GAs), its usefulness in parameter selection and how they can be implemented.
GAs mimic Darwinistic evolution [66]. This is why most terminology is borrowed directly from the world of biology. A general genetic algorithm proceeds as follows [66] [85]:

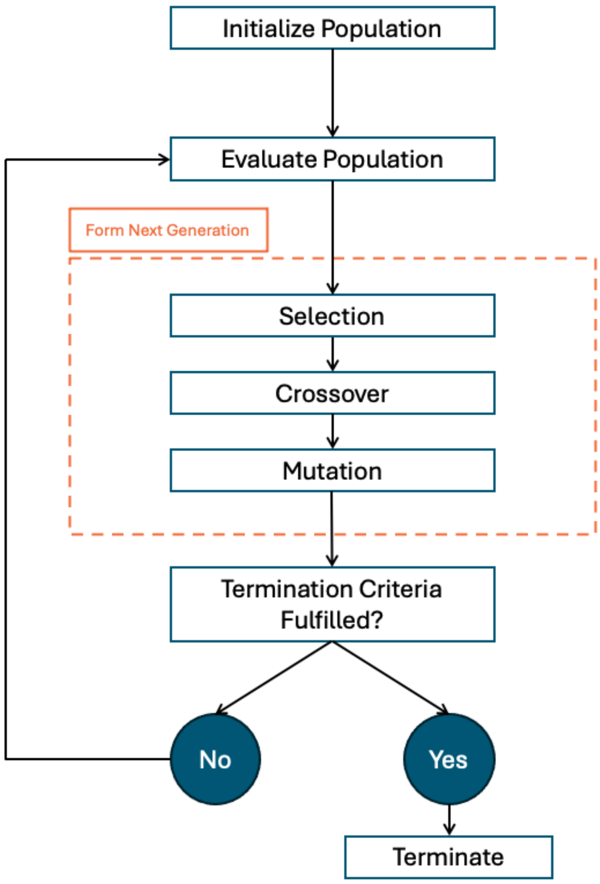


Figure 3.3: Schematic of a General Genetic Algorithm

Where a more detailed explanation follows:

1. Initialize the population by randomly generating N different chromosomes, c_i, where $i = 0, \ldots, N-1$, from the genome (the search space).
2. Evaluate the individuals
   (a) Decode the chromosomes into their corresponding parameters, p_i, where $i = 0, \ldots, N-1$.

(b) Evaluate the objective function g using the parameters. Assign a fitness value to each individual $G_i = g(p_i)$, where $i = 0, \dots, N-1$.

3. Form the next generation
   (a) Select two parents, i_1 and i_2, such that individuals with a better fitness value have a higher probability of being selected.
   (b) Generate two children, j_1 and j_2, by crossing the two parents' chromosomes c_{i_1} and c_{i_2}.
   (c) Mutate the two children j_1 and j_2.
   (d) Repeat until you have a new generation of N children.
4. Return to step 2 unless the termination criteria have been achieved.

Where the different terms are briefly explained as:

- **Population**: The set of individuals with size N.
- **Individuals**: The different sets of parameters being evaluated, encoded as chromosomes.
- **Decode**: Translating the numerical values stored in the chromosome into the parameters.
- **Objective function**: The function being optimized. In the case of this paper, it is the mean squared error of the Echo State Network (ESN) on the validation set.
- **Fitness value**: The value of the objective function for an individual. In the case of mean squared error, lower is better.
- **Generation**: The population during a specific iteration of the algorithm.
- **Parents**: The individuals selected in the current generation to create the next.
- **Children**: The product of the parents after crossing and mutation, constituting the next generation.
- **Termination Criteria**: The conditions that halt the algorithm. This could be a specified number of iterations, a certain fitness value achieved, or a maximum number of iterations without an improvement in the fitness value.

The three operators central to genetic algorithms are selection, crossover and mutation [66]. Different versions of the three exist and the ones utilized in this thesis are as follows:

Tournament Selection

1. Randomly select $n_{tournament}$ individuals from the population.
2. Order the selected individuals from best to worst.
3. Select the best individual with probability $p_{\text{tournament}}$. Continue down the ordered list until an individual has been selected, or there is only one individual left. In that case, select the last individual.

Single Point Crossover

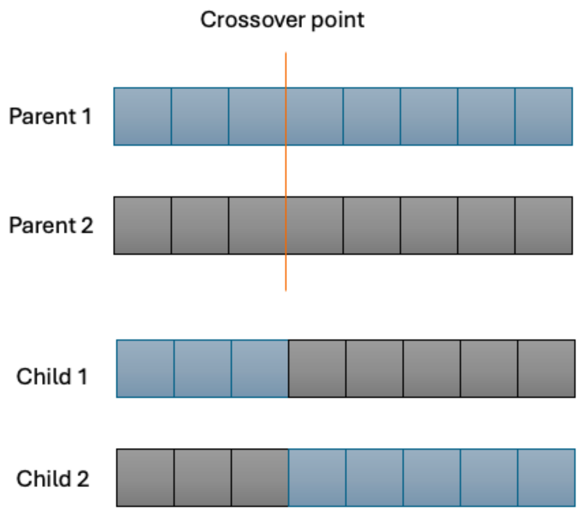


Figure 3.4: Schematic of a Single Point Crossover

1. Given two selected parents, cross them with probability p_{cross}. Otherwise, set the children to be exact copies of the parents.
2. Uniformly at random select a crossover point in the chromosome.
3. Set the first child to be the first parent's genes before the crossover point, and the second parent's genes after the crossover point. Vice versa for the second child.

Modified Bit-Flipping Mutation

Typically the chromosomes are encoded as strings of bits, ones and zeros. Then, during mutation every bit in each chromosome is flipped with a probability p_{mut}. This means that for a random 32-bit string representing a number in a specified range, you have the following distribution of absolute change for a mutation of $p_{mut} = 5/32$:

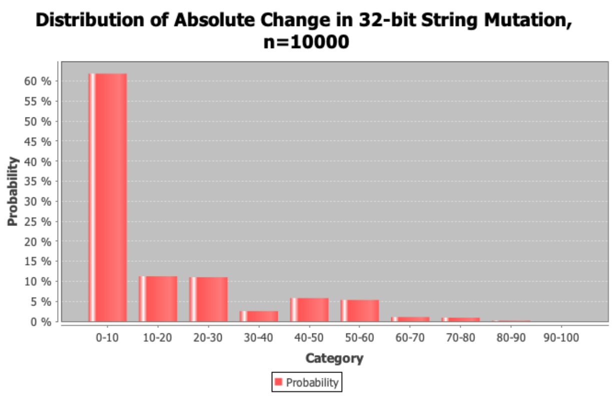


Figure 3.5: Probability Distribution of Absolute Change in a 32-bit String with $p_{mut} = 5/32$

The distribution reveals that smaller changes, relative to the allowed range, are significantly more frequent when flipping bits compared to mutations that alter 90-100% of the allowed range. In essence, bit flipping facilitates both small and large changes, but the likelihood of larger changes diminishes progressively as their magnitude increases.
Now, in the implementation in this thesis, instead of using bits, floating point numbers are used. One way of implementing mutation for floating point encoding is to uniformly pick a number in the allowed range. This loses the property described earlier of bit-flipping mutation. So, the method used in [7] is employed instead. The process follows:
Given an allowed range $R = [c_{Min}, c_{Max}]$ and a value c that should be mutated, randomly pick a direction up or down for c.
Without loss of generality we assume up was picked. This means the range the mutated value can fall into is $R = [c, c_{Max}]$.
Partition R into a equal parts of width w. Now, we want to pick a sub-range R_{sub}:

$$R_{\text{sub}} = [c, c + wi]$$
$$i \in \mathbb{Z}, \quad i \sim \mathbf{U}[1, a]$$

Then pick a number c_{mut} uniformly at random in R_{sub}. Let c_{mut} replace c in the chromosome.
This method of mutation leads to the following distribution of absolute change given a mutation occurs:

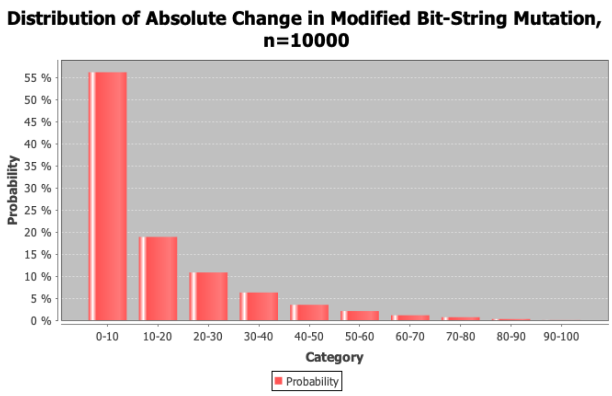


Figure 3.6: Probability Distribution of Absolute Change in Modified Bit-Flipping

The method enables mutations to result in both large and small changes but smaller changes are more likely, which is the property desired.

GA-Parameters

With the evolutionary operators defined, to run the genetic algorithm requires the following parameters:

1. N, Population size
2. M, Number of iterations

3. $n_{tournament}$, Tournament size
4. p_{cross}, Crossover probability
5. p_{mut}, Mutation probability
6. $[c_i^{min}, c_i^{max}]$, range of chromosome i
7. a_i accuracy of chromosome i

3.3.1 Premature Convergence

A common issue when implementing genetic algorithms is fitness values converging to a local optima and being unable to find children outperforming parents. This is called premature convergence [22]. There are several factors that play a role in premature convergence, but the main culprit is a lack of genetic diversity [64]. Various techniques have been developed to address this issue [28]. While choosing an appropriate approach one should strike a balance between exploration of the search space and exploitation of solutions already found.
Three measures are implemented in this thesis where the first two are inspired by [1] and the third by [25] [64]. They follow below.

1. Linearly Decreasing Crossover Rate

The idea is that, initially the population is diverse, and exploiting the best solutions to a higher degree will not cause premature convergence. Later on, when the population converges on a few solutions, a lower crossover rate should allow the algorithm to still explore new solutions. The linearly decreasing crossover rate is implemented as follows:

$$p_i^{\text{cross}} = p_{\max}^{\text{cross}} - \frac{p_{\max}^{\text{cross}} - p_{\min}^{\text{cross}}}{N-1} \cdot i$$

where:
- N is the number of iterations.
- p_i^{cross} is the crossover rate for iteration $i = 0, \ldots, N-1$.
- $p_{\min}^{\text{cross}}$ is the initial crossover rate.
- $p_{\max}^{\text{cross}}$ is the final crossover rate.

2. Linearly Increasing Mutation Rate

The idea is similar to the previous technique. Initially, the population is diverse, and a high mutation rate pushes the algorithm toward a random search. Later on, a higher mutation rate helps escape local optima. The linearly increasing mutation rate is implemented as follows:

$$p_i^{\text{mut}} = p_{\min}^{\text{mut}} + \frac{p_{\max}^{\text{mut}} - p_{\min}^{\text{mut}}}{N-1} \cdot i$$

where:
- N is the number of iterations.
- p_i^{mut} is the mutation rate for iteration $i = 0, \ldots, N-1$.
- $p_{\min}^{\text{mut}}$ is the initial mutation rate.

- $p^{\mathrm{mut}}_{\mathrm{max}}$ is the final mutation rate.

3. Stagnation Reset and Elitism

Since it is difficult to determine a satisfactory fitness level, the model is terminated after a certain number of iterations. In some cases, the model converges prematurely, despite the efforts mentioned above. One could decide to halt the algorithm after a certain number of stagnated iterations, that is iterations with very marginal or no progress.
The approach taken instead is to reintroduce genetic diversity by resetting the population after fitness progression has stagnated. This approach almost certainly worsens fitness at first as it's distribution is equivalent the fitness of the first generation. To prevent fitness from regressing, the individual with the best fitness previously is copied directly to the next generation. This is known as elitism. Elitism is applied every iteration, regardless of stagnation resets.
Elitism and stagnation resets are implemented as follows:

1. After crossing and mutating in each iteration:
   - If fitness improved, set the stagnation counter $c_{\text{stagnation}} = 0$, otherwise set $c_{\text{stagnation}} \leftarrow c_{\text{stagnation}} + 1$.
2. If $c_{\text{stagnation}} = N_{stagnation}$, set all children randomly.
3. After all operators (and possibly resetting), apply elitism:
   - Set one of the children to be an exact copy of the best parent.

Note that these techniques introduce additional parameters for the genetic algorithm, such as $c_{stagnation}$, and ranges for p_{cross} and p_{mut}.
It should also be noted that the selection method used affects premature convergence. Using a ranked system, such as tournament selection, instead of proportional fitness levels among individuals, can potentially mitigate premature convergence [85].

3.4 Overfitting

In most machine learning applications, over-fitting is a concern [34][85][77]. In general, the purpose of machine learning is to use some known input-output pairing to train a model such that it can generalize to unseen data, that is, produce predictions based on novel input. Under certain conditions, the model might fit very well to the seen data, but fail to make accurate predictions for unseen data; this is known as over-fitting. Usually, this occurs when the training methods over correct for the noise in the data.
Considering the methods used in this paper, where three machine learning algorithms are employed, over-fitting might be a concern. Two remedies for over-fitting are using large datasets and validating the results before testing. Fortunately, the data available is quite extensive. However, validating both GA and the training of the ESN requires splitting the dataset into validation and testing sets for both models. Another common approach is to use cross validation techniques, how this should be done for temporal data and reservoir states is unclear. Instead, the splitting is done as follows:

- **Warm-up Set** – washes out the initial random states.
- **Training Set** – used to optimize the readout layer.
- **Validation Set** – validates the ESN training and is used for evaluation in the GA.
- **Test Set** – tests the GA-optimized ESN.

With this setup, the issue of over-fitting could arise in the genetic algorithm on the validation set. However, the genetic algorithm is somewhat blunt in the sense that it only indirectly interacts with the noise in the data. To check if the setup is over-fitting the validation set, which is used to train the genetic algorithm, the training error is measured for the best performing individual (set of parameters). If the training error starts to consistently increase, that would be an indication of overfitting the validation set.

It is important to emphasize that the test set remains completely unseen by all optimization algorithms, ensuring it serves as an unbiased evaluation of the model's performance. The fitness on the validation set is exclusively used to guide the selection of the best network. If the genetic algorithm begins to overfit the validation set, this would likely manifest as both an increase in training error and poor performance on the test set. In such cases, alternative validation techniques or more robust data-splitting strategies should be explored to mitigate overfitting and improve the generalizability of the model.

4
Methods

When building an ESN and running a GA, there are plenty of design choices that needs to be made. There are also plenty of ways to process data and evaluate results. This chapter describes the approaches taken.

4.1 Data

The full historical data from the London Metal Exchange (LME) comes at a significant cost [48]. However, Westmetall provides access to a subset of historical data on their website [86]. The "cash settlement" data refers to the closing offer from the daily second ring round of cash futures, while the "three month" data corresponds to the closing offer from the second daily ring round of the three-month futures contract. It is important to note that these are not the official prices determined by the LME. Official prices typically fall between the final bid and offer of the second daily ring round, with final offers generally being slightly higher, though rarely exceeding a single trading tick ($0.5 or $0.1 depending on the metal). Additionally, the "stock" data provided by Westmetall represents the official stock levels recorded by LME-approved warehouses.

The metals Westmetall provide data for are aluminium, copper, lead, nickel, tin, zinc. All metals but tin are mentioned in the Northvolt 2023 annual report[70]. What ratios or volumes used in their manufacturing process is not public information.

Unfortunately, the data available on the Westmetall website is not provided in a readily downloadable format. As a result, it is necessary to scrape the data table directly from the HTML code. The algorithm employed for this purpose utilizes the Jsoup library [35] for parsing the HTML and the OpenCSV library [78] for handling CSV operations. The process operates as follows:

This produces a CSV file where each line consists of a date, cash settlement, three month and stock. The CSV files are then read into Java.

The data is then transformed to reflect the daily percentage change. This transformation results in the loss of the very first data point, as it is used as a reference. Various transformations were tested, including using the raw data or multiplicative changes, but the percentage change consistently yielded the best results. It is also an intuitive and easily interpretable format compared to other options, such as log changes. The transformation is applied to all data types (cash settlement, three month, and stock) as follows:

Algorithm 1 Extract Table Data from HTML and Write to CSV

```
Connect to the target URL
Select CSS query "table"
for each table in document do
    for each row in table (table.select("tr")) do
        for each cell in row (table.select("td")) do
            Write cell content to CSV
            Append comma "," after each cell
        end for
        Write new line in CSV after all cells in the row are processed
    end for
end for
```

$$\text{Let } i = 0, ..., T-1, \quad \text{percentageChange}[i] = \left(\frac{\text{data}[i+1] - \text{data}[i]}{\text{data}[i]} \right) \times 100$$

The percentage change of the different data types are then assembled into input and output matrices x and y to form paired sets for one-step predictions. This is done such that:

$$y_{i-1} = x_i, \quad \text{for } i = 1, \ldots, T$$

Where each pair $[x_i, y_i]$ is used for a one-step predictions, where x_i, along with it's history, aims to predict y_i.

The $[x, y]$ data is then split into a warmup, training, validation and test set. The warmup set should be large enough to wash out the initial states of the reservoir, 1000 data points of warmup and 2000 of training is suggested in [42]. Considering the finite amount of data in this application, such a ratio between warmup and training (1:2) might be too high. One also has to take into account that training the network with a ridge regression requires an overdetermined system of equations. How to split the data is a vast topic of discussion. Ultimately the split decided upon is:

Phase	**Percentage**
Warmup	10%
Training	85%
Validation	2.5%
Testing	2.5%

Because the number of data points available are slightly different among the metals, a percentage of the available is used instead of the number of data points.

With the data collected, transformed and split into different sets it is time to describe the specifics of the ESN, which is the topic of the next section.

4.2 The Echo State Network

This section describes how the ESN is initialized and how warmup, training, validation and testing is performed.
The activation functions used are: f^{in} - unity, f^{res} - tanh and f^{out} - unity. This results in the following formulations for updating states S_t and calculating output $\hat{y}_t$:

$$S_t = (1-\alpha)S_{t-1} + \alpha tanh\left(W^{res}S_{t-1} + \bar{W}^{in}\bar{x}_t\right)$$

$$\hat{y}_t = \bar{W}^{out}\bar{S}_t$$

Where the "bar" indicates the matrix or vector has been extended to accommodate a bias term and $tanh(\cdot)$ indicates the element wise hyperbolic tangent.

4.2.1 Initializing Networks

To initialize the network the initial state vector S^0 and the matrices W^{in}, W^{out} and W^{res} have to be defined and populated. There are a few legitimate choices that can be made. The approach taken follows:

$$S_i = 0 \ , \forall i \in [0, n-1]$$

where n is the reservoir size. Setting S to reasonable values is later handled by the warmup.

Having all of the nodes in the reservoir connected to both the input and output layers is not always optimal. To control this we need two sets of connections to the reservoir nodes for W^{in} and W^{out}:

$$\text{NodesIn} = \{s_i \in \mathcal{U}_d[0, n-1] \mid s_i \text{ drawn without replacement, for } i \in [0, inLength-1]\}$$

$$\text{NodesOut} = \{s_i \in \mathcal{U}_d[0, n-1] \mid s_i \text{ drawn without replacement, for } i \in [0, outLength-1]\}$$

Where $\mathcal{U}_d$ denotes the discrete uniform distribution and in- and outLength is the number of non zero connections to the reservoir.

Since W^{out} is determined during training with a ridge regression, which does not require calculation of output since the desired output is known, there is no need to randomly initialize the matrix. NodesOut is used to determine which nodes' state should be collected to form the design matrix. The entries of $\bar{W}_{out}$ are then calculated as seen in the theory section.

W^{in} on the other hand needs to be randomly initialized. Controlling the distribution of W^{in} in relation to the input types and reservoir is a desirable feature. Therefore an input scale for each of the data types and a bias scale is used. W^{in} is initialized as:

$$W^{in}_{j,i} \in \mathcal{U}[-inScale_j, inScale_j] \quad \forall i \in \text{NodesIn},\ j \in [0, k-1]$$
$$W^{in}_{j,n} \in \mathcal{U}[-inBiasScale, inBiasScale] \quad \forall j \in [0, k-1]$$

where k is the number of data types.

When it comes to W^{res} the process becomes more involved, considering the echo state property (ESP). The process follows:
Let A denote the $n \times n$ adjacency matrix initialized either uniformly random or scale free as described in the theory section. Initialize the weights as:

$$W^{res}_{i,j} \leftarrow A_{i,j}\mathcal{U}(0, 1] \quad \forall i, j \in [0, n-1]$$

Such that the non-zero elements are set uniformly random on $(0, 1]$. The next step is to scale the weights such that W^{res} has a specified spectral radius. This is done by calculating the current spectral radius of W^{res} and then multiplying all weights by a ratio of the wanted spectral radius and the current spectral radius. The spectral radius is approximated using power iteration. The process follows [87][9]:

Algorithm 2 Power Iteration for Largest Eigenvalue

```
randomly initialize b_k ∈ R^n, where ||b_k|| ≠ 0, b_k is the initial guess for the
dominant eigenvector.
for iter = 1 to MaxIterations do
    b_{k+1} ← W^{res} b_k
    ρ ← (b_{k+1}^T b_{k+1}) / ||b_{k+1}||
    b_{k+1} ← b_{k+1} / ||b_{k+1}||  {normalize b_{k+1}}
    if ||b_{k+1} − b_k|| < tolerance then
        break
    end if
    b_k ← b_{k+1}  {update for the next iteration}
end for
```

Here, b_k is a normalized vector that approximates the dominant eigenvector of W^{res}, which corresponds to the largest eigenvalue. The spectral radius ρ is approximated as the largest eigenvalue of W^{res}, estimated iteratively using the updated b_{k+1} vector. The power iteration is run with $MaxIterations = 1000$ and $tolerance = 1e-9$. Thus we have a close approximation of the spectral radius of W^{res} and we can rescale to the spectral radius wanted:

$$W^{res} \leftarrow \frac{\rho_{wanted}}{\rho} W^{res}$$

To make W^{res} include both positive and negative values approximately half of the non-zero values are randomly set to be negative.

This concludes the network initialization.

4.2.2 Warmup and Training

Warmup is simply passing input to the reservoir updating the states. This assures that the initial state, where all states are zero, does not affect the subsequent training process.
When the warmup set has been passed through, training begins. This is done by passing the training set through the reservoir and collecting the states of the output nodes creating the design matrix. Then the output weights and bias is calculated using ridge regression as described in the theory section.
There are a few alternatives to optimize λ since the computational expense is quite low considering the same design matrix is used in addition the network does not have to be reinitialized. However, employing for instance a logarithmic grid search for λ, in conjunction with the GA for the other parameters, adds up and ultimately slows down the optimization significantly. λ is there for also included in the genome fo the GA.

4.2.3 Validation and Testing

With the output weights and bias set from the offline training, the model can now produce predictions. To validate the model the validation set is passed through the reservoir and approximations $\hat{y}_t$ are calculated. Since the model is adaptive though online training, at each time step the error gradient is calculated and the weights slightly adjusted. As discussed in the theory section we need the gradient of the error with respect to the weights and bias. Defining the loss function L_t as follows, we have:

$$L_t = \frac{1}{2}||\hat{y}_t - y_t||^2$$

$$\hat{y}_t = f^{out}(z_t)$$

$$z_t = \bar{W}^{out}\bar{S}_t$$

That is, L_t is a function of $\hat{y}_t$ which is a function of the activation z_t which is in turn a function of $\bar{W}^{out}$. Using the chain rule, the gradient of L_t can be expressed as:

$$\frac{\partial L}{\partial \bar{W}^{out}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial \bar{W}^{out}}$$

Where:

$$\begin{aligned} \frac{\partial L_t}{\partial \hat{y}_t} &= \hat{y}_t - y_t \\ \frac{\partial \hat{y}_t}{\partial z_t} &= 1,\ f^{out} \text{ is unity} \\ \frac{\partial z_t}{\partial \bar{W}^{\text{out}}} &= \bar{S}_t \end{aligned}$$

and $\bar{W}^{out}$ is updated as:

$$\begin{aligned}\bar{W}^{out} \leftarrow & \bar{W}^{out} - \eta \nabla L_t \\ \bar{W}^{out} \leftarrow & \bar{W}^{out} - \eta \left((\hat{y}_t - y_t) \bar{S}_t^T \right)\end{aligned}$$

This is done at every step of both validation and testing. Now, η has to be set such that the model does not over correct for small errors nor fail to correct for changing dynamics. The optimization of η is handled by the GA.
When the model has produced predictions for the validation and test sets respectively the results are measured using a few test metrics. These are the topic of the following section.

4.2.4 Calculating Test Metrics

To measure how well the model can make predictions, the mean squared error (MSE) is used. Considering the output has multiple dimensions the mean squared error is the mean of the MSE of the different dimensions. Given T predictions and m dimensions this is calculated as follows:

$$\text{MSE} = \frac{1}{Tm} \sum_{i=1}^{T} \sum_{j=1}^{m} (y_{j,i} - \hat{y}_{j,i})^2$$

This is the metric used as fitness in the GA.
The primary concern of the model is to predict the next day cash. Therefore, the MSE of the cash dimension of y and $\hat{y}$ is metric of ultimate interest. These values are denoted as y_{cash} and $\hat{y}_{\text{cash}}$, respectively. Given T predictions, the MSE_{cash} is calculated as follows:

$$\text{MSE}_{cash} = \frac{1}{T} \sum_{i=1}^{T} (y_{\text{cash},i} - \hat{y}_{\text{cash},i})^2$$

In addition to MSE, three other evaluation metrics are calculated to assess the efficacy of the model: mean absolute error (MAE), coefficient of determination (R^2), and directional accuracy. Omitting the cash sub text for simplicity, these metrics are calculated as follows:

$$\text{MAE} = \frac{1}{T} \sum_{i=1}^{T} |y_i - \hat{y}_i|$$

$$R^2 = 1 - \frac{\sum_{i=1}^{T} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{T} (y_i - \bar{y})^2} \quad \text{where } \bar{y} \text{ denotes the mean of } y.$$

$$\text{Directional Accuracy} = \frac{1}{T} \sum_{i=0}^{T-1} 1_{y_i,\hat{y}_i}, \text{ where } 1_{y_i,\hat{y}_i} = \begin{cases} 1 & \text{if } sign(y_i) = sign(\hat{y}_i) \\ 0 & \text{otherwise} \end{cases}$$

Because the model takes the transformed version of the underlying data it might also of interest to take a look at what the predictions produced in the original, reverted form. Since the original values of y are already available the only thing one has to calculate is the reverted $\hat{y}$ which is done as follows:

$$\hat{y}_i^{reverted} = x_i^{original}\left(1 + \frac{\hat{y}_i^{percent}}{100}\right)$$

Note that the reverted and original super text is omitted in other cases for simplicity.

As outlined in the theory section the RW model produces the prediction that the expected value of the next price is the same as the current price. In terms of percentage change, the model predicts an expected 0% change every day. This makes predictions very convenient to calculate. The result metrics for the RW model are calculated in the exact same way as above, just replacing $\hat{y}_{cash}$ with 0 and $\hat{y}_{cash}^{reverted}$ with $x_{cash}^{original}$. The directional accuracy is not very useful, considering the RW model predicts no directional change, therefore instead the ratio of upward daily movements are calculated to give context to the results of the ESN's predictions. The directional accuracy is calculated as:

$$\text{Directional Accuracy} = \frac{1}{T}\sum_{i=0}^{T-1}\begin{cases} 1 & \text{if } y_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

4.3 Genetic Algorithm

This section outlines the approach taken for the genetic algorithm. Having established all the parameters needed for the initialization, offline and online training of the ESN as well as having a way of measuring how well the model makes predictions the setup for the GA can be established. First off is the search space for the parameters the Genome, the topic of the following section.

4.3.1 The Genome

The genome is what defines the search space for the parameters. These are the values used:

Parameter	Min Value	Max Value	Precision
Topology	0	1	1
Connectivity	0.01	0.15	5
LearningRate	0	0.01	5
Regularization	-5	5	5
SpectralRadius	0.5	1	7
LeakageRate	0	1	7
ReservoirSize	500	1200	7
inLength	50	1200*	7
outLength	50	1200*	7
inBiasScale	0	100	5
inFactorScales	0	3	5

The precision is the number of intervals used in the modified bit-flipping mutation, therefore a higher precision value makes smaller changes more likely. The topology parameter is the only non numerical parameter. Since the parameter has precision 1, initialization and mutation is uniformly random on the interval. The random value is then rounded to the closest integer, that is either 0 which indicates scale free or 1 which indicates a uniformly random topology. The regularization is on a 10 base logarithmic scale. That is if the gene is encoded as -5, the λ in the model is decoded as 10^{-5}. Since the in- and outLength cannot exceed the reservoirSize this has to be accounted for when decoding. Since the lengths and sizes represent a number of nodes, the gene values are rounded to the nearest integer.

Choosing the values for the genome is challenging and mostly done through trial and error. There are some general guidelines. The spectral radius, for instance, should be scaled below unity. The connectivity should make the reservoir sparse and the leakage rate is defined between 0 and 1. Reservoir size is limited by computing capacity, which in turn limits the in and out lengths since they cannot be larger than the number of nodes in the reservoir. The out length is also limited by the available training data, considering that ridge regression requires an overdetermined system.
The approach to defining the genome was to iteratively test using the metal data and inspecting if any parameter approached a boundary, then limits were adjusted accordingly.
Also note that the precision determines how likely mutation is to take small steps in the allowed range. Therefore a higher precision places more emphasis on finding a precise optimal value than exploring the whole range. A precision of 1 means, given the parameter is chosen the be mutated, that all the values in the range are equally likely during mutation.

4.3.2 Initialization

Initializing the population is done as follows:

$c_{i,j} = min_j + (max_j - min_j)r$ for every $i \in [0, P-1], \quad j \in [0, C-1]$, where $r \in \mathcal{U}[0,1]$

P indicates the population size, C which is the number of genes for each individual and min_j, max_j is the min and max values for each gene determined by the genome.

4.3.3 Evaluation

Given a set of individuals P_g, the population of generation g, each individual c_i defines an ESN when decoded. For each c_i an ESN is initialized, warmed up, trained and then validated with online training. The MSE, averaged for all dimensions of the output, on the validation set is then used as a fitness score for evaluation.
In the case of the metals, this means the parameters defining networks producing smaller MSE for predicting the next day cash, three month and stock is awarded a better fitness score.
While trying out parameters for the GA the error for the training set would sometimes explode while still improving on the validation set. With the setup ultimately

decided upon this does not seem to happen. Regardless, the MSE on the training set is tracked however not directly involved in the evaluation.
The GA is run for 60 generations after which the parameters corresponding to the best individual are stored in a CSV file. Also included in the file is the random seed used to construct the ESN. This is necessary for reproducing the ESN's performance seen during the GA which is sensitive to the random parts of initialization.

4.3.4 Evolution

Once all c_i have been evaluated they are ready to be evolved. To do this the cross and mutation probabilities for generation g are calculated as described in the theory section.
Using tournament selection, with a tournament size of 5, the parents of the next generation are chosen. With the current cross probability, they are crossed using one point crossover and the children of the next generation are formed.
Next, each of the genes in the children are mutated with the current mutation probability, using modified bit-flip mutation for the gene specific specified precision in the allowed range.
To make sure the performance does not regress elitism is applied. That is, a random child in the next generation is replaced with a direct copy of the best parent.
To make sure that the search space is adequately explored, should the performance of the best individual be the same for 10 generations, the population is re-initialized. This is done before elitism is applied meaning the performance does not regress despite resetting the population.

4.3.5 Summary of Genetic Algorithm Parameters

To summarize, the parameters used for each of the runs, that is both the benchmarking systems and all the metals, for the genetic algorithm are:

Parameter	Value
Iterations	60
Population Size	30
Tournament Probability	0.8
Tournament Size	5
Initial Cross P.	0.9
Final Cross P.	0.5
Initial Mutation P.	0.5/numGenes
Final Mutation P.	2.0/numGenes
Reset Duration	10

Table 4.1: Genetic Algorithm Parameters

Where numGenes is the number of genes in the respective genomes. This means that on average 0.5 genes in a chromosome are mutated initially, increasing to 2 genes mutated per chromosome towards the end of the run. The formulation is often more convenient than just using a raw number, especially when the number of genes shift which is the case of the Mackey-Glass Equations as seen in section 4.5.

4.4 Refinements to the Methods

During testing of the methods on the metal dataset, it became evident that a uniform approach across all metals was suboptimal. The primary challenge is ensuring the model's ability to generalize to unseen data in the test set. To address this, three refinements were introduced.

First, the fitness function was modified to focus solely on MSE_{cash}. Guiding the GA to prioritize models that optimize performance in the cash dimension allows it not to be constrained by trade-offs with other dimensions. The metric aligns with the primary objective of the report used to benchmark the model against the RW. The calculation of MSE_{cash} follows the procedure detailed in 4.2.4.

Second, the optimization of the regularization parameter (λ) was separated from the GA and handled through a grid search. This approach proceeded as follows:

Algorithm 3 Grid Search for Regularization Parameter λ in Echo State Network (ESN)

for each individual in the population **do**
  Initialize the Echo State Network (ESN)
  Wash out the initial states using the warmup set
  Gather the reservoir states for the first 90% of the training set, denoting the state matrix as $\bar{S}^{90}$
  for each $\lambda_i = 10^i, i = -3, -2, \ldots, 2, 3$ **do**
    Calculate the output weights and bias:

$$\bar{W}^{\text{out}} = Y\bar{S}^{90}\left(\bar{S}^{90}\bar{S}^{90^T} + \lambda_i I\right)^{-1}$$

    Use these weights to make predictions for the remaining 10% of the training set
    Compute MSE_{all}
    Reset the reservoir states to $\bar{S}^{90}_{\text{end}}$, the final state of the first 90% of the training set
  end for
  Choose the λ_i with the lowest MSE_{all}
  Reset the reservoir states to zero
  Proceed with the usual process: warmup, training on the entire training set, and compute MSE_{all} for predictions on the validation set
end for

Third, the stock dimension was removed from the dataset as it was identified as a possible source of issues.

These refinements all maintain the same data splits for the validation and test sets, ensuring that results remain directly comparable. While alternative methods, such as cross-validation, could potentially improve generalization, they are challenging to implement due to the temporal structure of the data.

4.5 Benchmarking Using Mackey-Glass and Lorenz Attractor

To assess whether the ESN-GA model is effective at predicting chaotic dynamic systems, the two examples Mackey-Glass and Lorenz are used. The two systems are described by the following equations:
Mackey-Glass Equations:

$$\frac{dx(t)}{dt} = \beta \frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t)$$

where x describes the state of the system, and the parameters used are:

$$n = 10 \quad \beta = 0.2 \quad \tau = 17 \quad \gamma = 0.1$$

Lorenz Attractor Equations:

$$\frac{dx}{dt} = \sigma(y-x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

where x, y, z determine the state of the system over time, and the parameters used are:

$$\sigma = 10 \quad \rho = 28 \quad \beta = \frac{8}{3}$$

Solving these ordinary differential equations, given starting points to the variables, is in short done by discretizing t and numerical integration. There are available methods for this in the apache commons library [2]. However, because training the ESN requires several thousand data points, issues arise. The solvers FirstOrderIntegrator, ClassicalRungeKuttaIntegrator and DormandPrince853Integrator all stop functioning before a satisfactory number of datapoints have been produced. This might be an issue of implementation.

Luckily there are readily available implementations that can produce several thousand datapoints. The code used for the Mackey-Glass is provided by [14] and the Lorenz Attractor provided by [82]. Both implementations are run with the desired parameters, with the results exported to CSV files and then imported into Java for further processing.
Note that the Mackey-Glass system has one dimension. This means a smaller genome is required considering there is only one input scale to optimize. Both the Lorenz Attractor and the metal data have three dimensions.

The same data split for the two benchmarking systems is used. It follows:

Phase	**Number of Data Points**
Warmup	1000
Training	2000
Validation	500
Testing	500

4.6 Software and Tools

All code, except for generating chaotic systems data, was written in the IntelliJ IDE [43], utilizing Java 13. The code is then run through a java virtual machine on a MacBook Air 2017 with MacOS Monterey 12.7.6.
The following libraries were used, all accessed through Maven [24]:

- **Apache Commons Math**: version 3.6.1 [2]
- **Efficient Java Matrix Library (EJML)**: version 0.43.1 [76]
- **JFreeChart**: version 1.0.17 [18]
- **Jsoup**: version 1.17.2 [35]
- **OpenCSV**: version 5.9 [78]

During the coding process, both OpenAI's ChatGPT 3.5 and 4.0 [74] and JetBrains AI Assistant 2024.2 and 2024.3 [44] were used.
The LaTeX template used for this document was provided by David Frisk [17], and ChatGPT is used to assist with formatting plots, tables, and equations.
The Mackey-Glass data was generated using MATLAB [81], with code provided by [14]. Meanwhile, the Lorenz Attractor data was generated in Python [23], with code sourced from [82].

4.7 Limitations of the Methods

The methods employed in this study have several limitations:

1. **Parameter Search:** The genetic algorithm used for parameter search is not guaranteed to find a global optimum but rather a satisfactory solution. Additionally, the complexity of patterns in the data might exceed the representational capacity of the network, constrained by both the number of nodes and available hardware resources.
2. **Data Complexity and Availability:**
   - The patterns in the data might lack sufficient predictive power to consistently outperform a random walk.
   - The methods are not restricted to the three data types used but could incorporate additional datasets, which are unavailable to the author.
3. **Price Discrepancy:** The data does not use the official LME closing price but instead relies on the last *offer* made on the trading ring. While this difference is often minimal, just a single trading tick, it may still influence results and practical application.

4. **Computational Constraints:** Limited computational capacity restricts the size of the search space and the ability to thoroughly evaluate all possible configurations.
5. **Model Construction Choices:** The numerous design decisions made during model construction are not rigorously verified for their individual contributions to performance. Comparing alternate paths or evaluating every choice is infeasible within the given time frame, particularly due to the long runtime required for GA evaluations and subsequent testing.
6. **Fitness Function:** The fitness function, which evaluates parameter combinations, is not guaranteed to be convex. Optimal solutions may lie outside the explored search space, even if parameters are far from a boundary.
7. **Inherent Data Randomness:** This setup cannot directly determine whether the data is inherently random. Additional tests would be required to evaluate this aspect.
8. **Statistical Significance:** The results do not include statistical significance measures, which limits the ability to provide confidence intervals for predictions.
9. **Data Integrity:** The data originates from a third-party source, so its accuracy cannot be independently verified.
10. **Prediction Horizon:** The model is limited to one-day-ahead predictions. Extending the prediction horizon would require modifications to the setup.
11. **Multi-Day Predictions:** The current setup does not support autonomous multi-day predictions. While implementing a feedback model that uses its own predictions as inputs is straightforward, it is incompatible with stochastic gradient descent, which requires true data for iterative weight updates.
12. **Simultaneous Time and Space Dependence:** The Mackey-Glass equations exhibit time dependence, while the Lorenz Attractor demonstrates spatial dependence. Although the setup is capable of handling these phenomena individually, there is no assurance that the model can accurately predict systems where both time and space dependencies occur simultaneously.

This list is by no means exhaustive.

4.8 Summary of Methodological Approach

This section summarizes the methodological approach employed for optimizing and testing the model.

Data Collection

- **For Benchmarking:**
  - Generate the data.
  - Split the data into warmup, training, validation, and test sets.
- **For Metals:**
  - Collect the data from Westmetall.
  - Transform the data into percentage changes.
  - Split the data into warmup, training, validation, and test sets.

Genetic Algorithm

- Initialize the population.
- **Loop:** For each generation:
  - **Evaluate Population:**
    - Initialize the ESN.
    - Perform warmup.
    - Train:
      - Train weights using Ridge Regression.
    - Validate:
      - Nudge weights using gradient descent.
      - Calculate Mean Squared Error (MSE).
  - **Evolve Population:**
    - Perform selection and crossover.
    - Mutation.
    - Stagnation reset if necessary.
    - Elitism.
- **Store optimized parameters and SEED.**

Testing Optimized Parameters

- Initialize the ESN with optimized parameters.
- Perform warmup.
- Train using Ridge Regression.
- Validate using gradient descent and store validation metrics.
- Test using gradient descent and store test metrics.

The four different setups MSE_{all}, MSE_{cash}, grid search, and no-stock are applied to six different metals. The benchmarking of the model for Mackey-Glass and Lorenz data is only done with the MSE_{all} setup.

5
Results

This section presents the results. First the ESN-GA model is tested for the known application of predicting Mackey-Glass and Lorenz Attractor data. Then an overview of the metal data is presented leading up to the results pertaining to parameter optimization and finally a performance comparison to a random walk model.

5.1 Mackey-Glass Equations and Lorenz Attractor

This section is included to validate the ESN-GA model on well-established applications, ensuring its functionality. Additionally, it examines whether the GA effectively selects parameters that enhance performance and evaluates the ESN's ability to handle chaotic data.
We begin by analyzing the performance progression of the GA on the Mackey-Glass dataset. Given that the Mackey-Glass system features a single spatial dimension with temporal dependencies, strong performance would indicate that the ESN-GA model successfully relates past values to predict future ones.

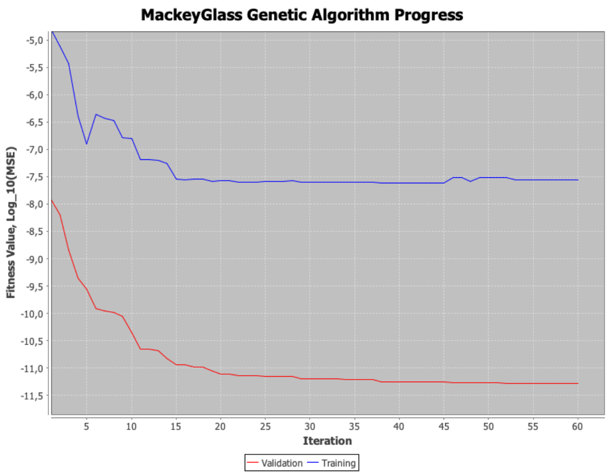


Figure 5.1: Genetic Algorithm Progress for Mackey-Glass Equations

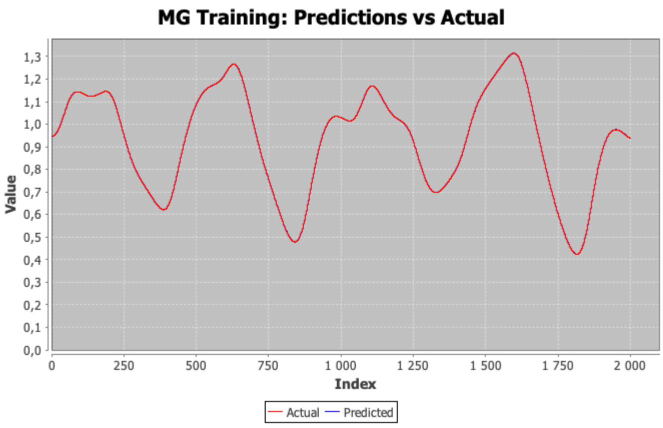
The training and validation MSE goes from 1.628E-5 and 1.167E-8 to 2.795E-8 and 5.253E-12 respectively. Note that the plot is on a logarithmic scale and that the algorithm is still improving upon termination, though progress slows down. There might be a case for running the GA longer, however this clearly indicates that the

GA is capable of selecting parameters superior to random ones. The parameters settled upon are:

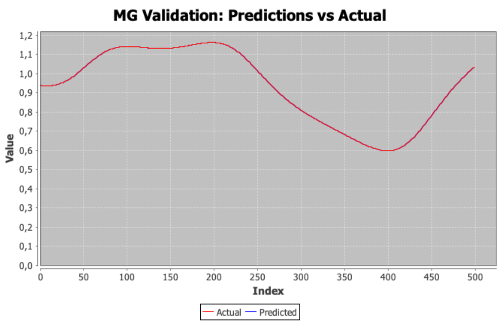
Parameter	Value
Topology	Scale-Free
Connectivity	0.1387
Spectral Radius	0.8399
Regularization	2.730e-4
Learning Rate	9.999e-3
Leakage Rate	0.998
Reservoir Size	764
Input Length	358
Output Length	358
Input Bias Scale	15.405
Input Factor Scale	2.994

Table 5.1: GA Optimized Parameters for Mackey-Glass

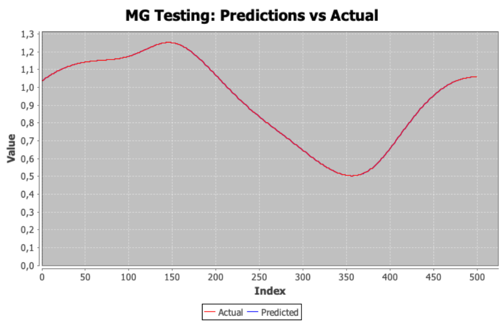
Re-initializing the network, using the same SEED as during the GA, results in the following predictions:



(a) Training



(b) Validation



(c) Testing

Figure 5.2: Mackey-Glass Model Predictions vs Actual for Training, Validation, and Testing

The predicted values closely align with the actual values to the point where the difference is not distinguishable at the resolution and line thickness used. This points to the model's ability to predict the behavior of the Mackey-Glass system. Following is the test metrics of the predictions:

Metric	MSE	MAE	R-Squared	Directional Accuracy
Training	2.795e-08	1.157e-04	1.000	0.993
Validation	5.250e-12	1.786e-06	1.000	0.998
Testing	1.966e-11	3.032e-06	1.000	0.996

Table 5.2: Mackey-Glass Model: Performance Metrics

The results demonstrate that the model clearly can understand the behavior of the Mackey-Glass system. There is still an error, although small, and possibly room for improvement. Considering the values are stored as doubles, which have a precision of around 10^{-16} this should not cause issues. The GA is still improving after 60 iterations indicating the model in its current form has more potential. Although, more iterations are not necessary considering the results clearly illustrate that the model is capable of predicting the Mackey-Glass system.
With that established we move on to the Lorenz system. The system has three spatial dimensions that interact making the complexity greater. If the ESN-GA model can perform well it establishes the model is capable of detecting interactions between input variables, not only temporal structures in the data. The Lorenz attractor also has a higher degree of chaos, measured in Lyapunov exponent, compared to Mackey-Glass for the respective parameters, thus making predictions slightly more difficult. Note that the predictions are one-step, so errors do not accumulate exponentially as they would do for multi-step predictions.
The GA progress follows:

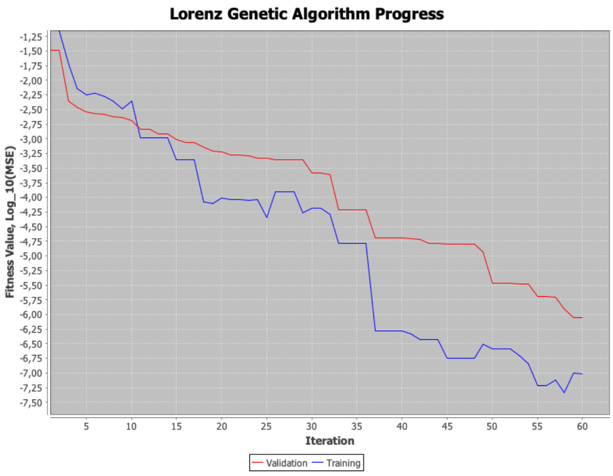


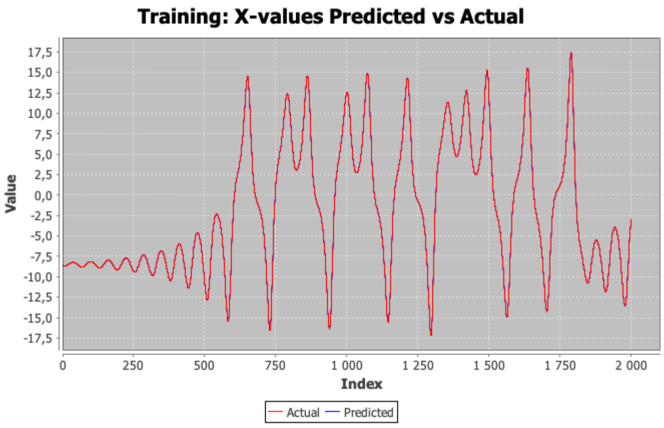
Figure 5.3: Genetic Algorithm Progress for Lorenz Attractor

The training and validation MSE go from 0.073 and 0.033 to 9.756E-8 and 8.782E-7 respectively. Yet again a marked improvement from the initial random ESN parameters. Here it is more clear that the GA could run for longer considering the steady improvements until termination. Because there are no obvious signs of premature convergence the population size and enhancements can be deemed satisfactory, at elast in this case. The following are the parameters the GA settled upon:

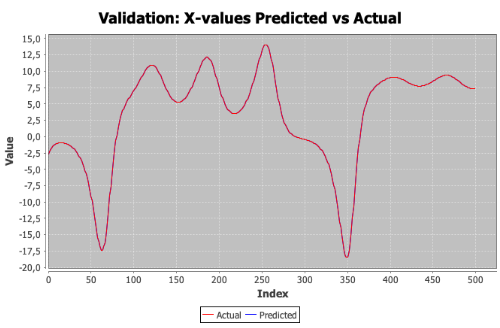
Parameter	**Value**
Topology	Scale-Free
Connectivity	0.1352
Spectral Radius	0.6985
Regularization	1.690e-5
Learning Rate	2.361e-3
Leakage Rate	0.1324
Reservoir Size	967
Input Length	967
Output Length	855
Input Bias Scale	13.029
Input Factor Scale	[0.2937, 0.0741, 0.0804]

Table 5.3: GA Optimized Parameters for Lorenz Attractor

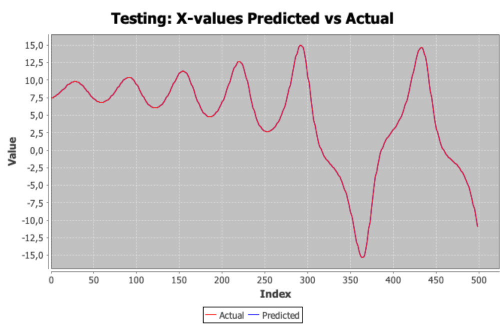
Re-initializing the network with the optimized parameters and seed produces the following predictions:



(a) Training

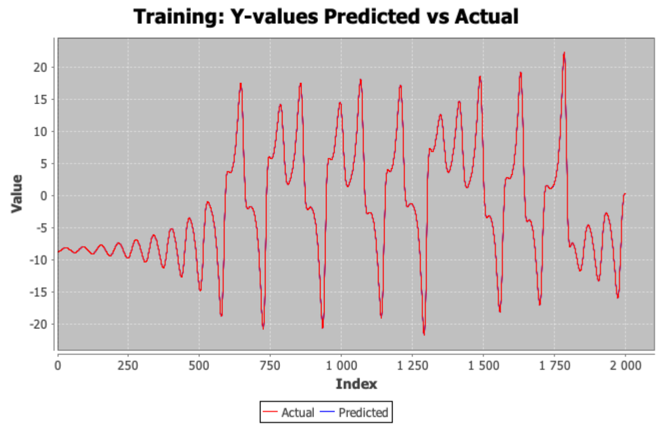


(b) Validation

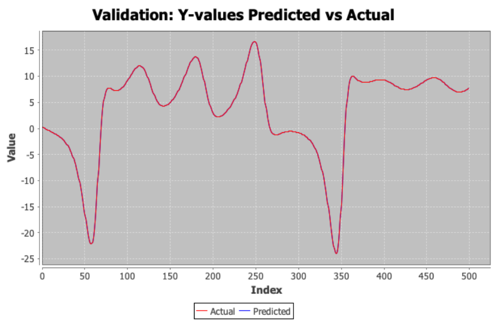


(c) Testing

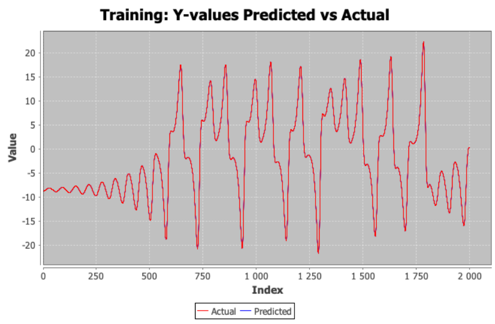
Figure 5.4: X Values of Lorenz Attractor Model Predictions vs Actual for Training, Validation, and Testing



(a) Training

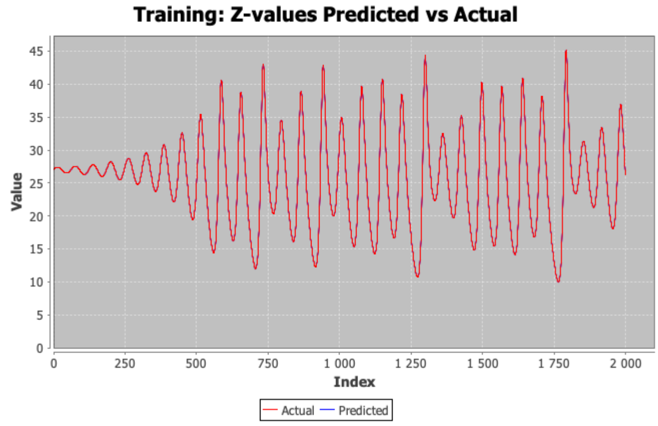


(b) Validation

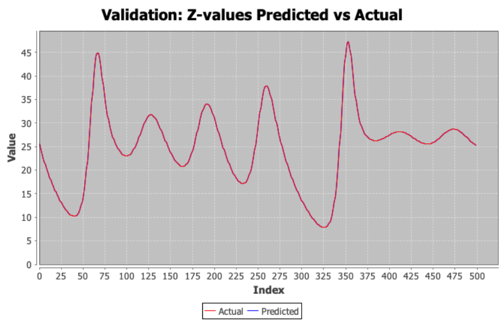


(c) Testing

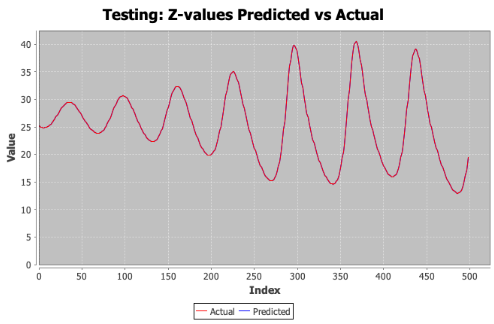
Figure 5.5: Y Values of Lorenz Attractor Model Predictions vs Actual for Training, Validation, and Testing



(a) Training



(b) Validation



(c) Testing

Figure 5.6: Z Values of Lorenz Attractor Model Predictions vs Actual for Training, Validation, and Testing

Again the difference between the actual and predicted values are hard to distinguish in the plots. Clearly the model is capable of making one-step predictions in multiple dimensions. The test metrics for the three dimensions of the Lorenz Attractor Model follows:

Variable	Phase	MSE	MAE	R-Squared	Directional Accuracy
X	**Training**	3.107e-08	1.310e-04	1.000	0.999
	Validation	1.560e-07	1.307e-04	1.000	0.998
	Testing	2.081e-09	3.206e-05	1.000	0.998
Y	**Training**	1.651e-07	2.962e-04	1.000	0.999
	Validation	1.766e-06	3.521e-04	1.000	0.996
	Testing	1.173e-08	7.243e-05	1.000	0.998
Z	**Training**	9.651e-08	2.302e-04	1.000	1.000
	Validation	7.124e-07	2.644e-04	1.000	0.998
	Testing	8.112e-09	6.431e-05	1.000	0.998

Table 5.4: Lorenz Attractor Model: Performance Metrics for X, Y, and Z Variables

The situation is similar for the Lorenz attractor. The model is clearly capable of predicting the behavior of the system with near perfect metrics for both R^2 and directional accuracy. The comparatively larger values for MSE and MAE could to some degree be attributed to larger absolute values. This might call for the use of a ratio based error metric, but the purpose is not to extensively analyze the results from these two benchmark data sets but to be able to conclude that the model works for known applications. It is clear that the model is capable of improving on a random selection of parameters and the ESN model initialized with said parameters is capable of predicting chaotic systems with temporal or spatial dependencies. The next step is to apply the model to the metal data. First we take a look at the LME data.

5.2 Data Overview

The following is the data from the LME provided by Westmetall from the 2nd of January 2008 to 2nd of September 2024 for the cash settlement, three month and stock of six metals. Due to trading being halted there are some gaps in the data. These missing data are summarized in the table below for the different metals.

Metal	**Missing Data Points**	**Usable Data Points**
Aluminium	87	4131
Copper	7	4211
Lead	85	4132
Nickel	99	4118
Tin	87	4130
Zinc	85	4132

Table 5.5: Number of Missing and Usable Data Points for the Metals

It should be noted that the missing data are distributed across the various data types. When one or more dimensions are missing for a given trading day, the entire day is discarded. Given the relatively small proportion of missing data, the remaining dataset is assumed to still provide an accurate representation of reality.

A potential issue arises when missing data points occur consecutively over extended periods. For example, the tin three-month data is missing from February 20 to June 16, 2008, and the nickel three-month data is missing from March 8 to March 25, 2022. In such cases, these gaps are simply skipped, resulting in discontinuities in the time series. For instance, in the case of nickel, the time series entry following March 7, 2022, directly after is March 26, 2022. This requires the assumption that such gaps will not significantly impact predictive performance.

The effect of missing data is visibly evident as straight-line segments in the following plots of the datasets, which highlight these discontinuities.

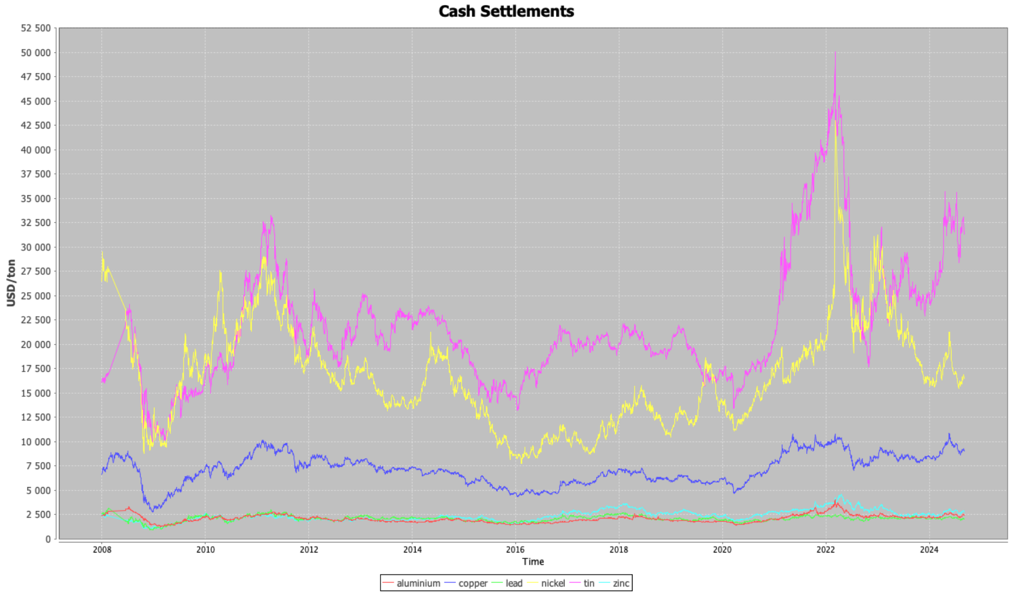


Figure 5.7: Cash Settlements of the Metals

Metal	Mean	Median	Variance
Aluminium	2.045E+3	1.997E+3	1.511E+5
Copper	7.107E+3	7.049E+3	2.486E+6
Lead	2.064E+3	2.079E+3	8.698E+4
Nickel	1.650E+4	1.614E+4	2.563E+7
Tin	2.183E+4	2.059E+4	4.008E+7
Zinc	2.369E+3	2.306E+3	3.264E+5

Table 5.6: Statistics for the Cash

As the plots illustrate, the cash settlements for the different metals are quite different in absolute terms. Lead, zinc, and aluminium appear to have similar behavior being quite stable during the period. Contrasted with nickel and tin which are more volatile, exemplified by the price level in early 2009 then to be almost five times higher in early 2022 and then almost completely recovered by 2024.

Next, we inspect the 3 three month for the metals.

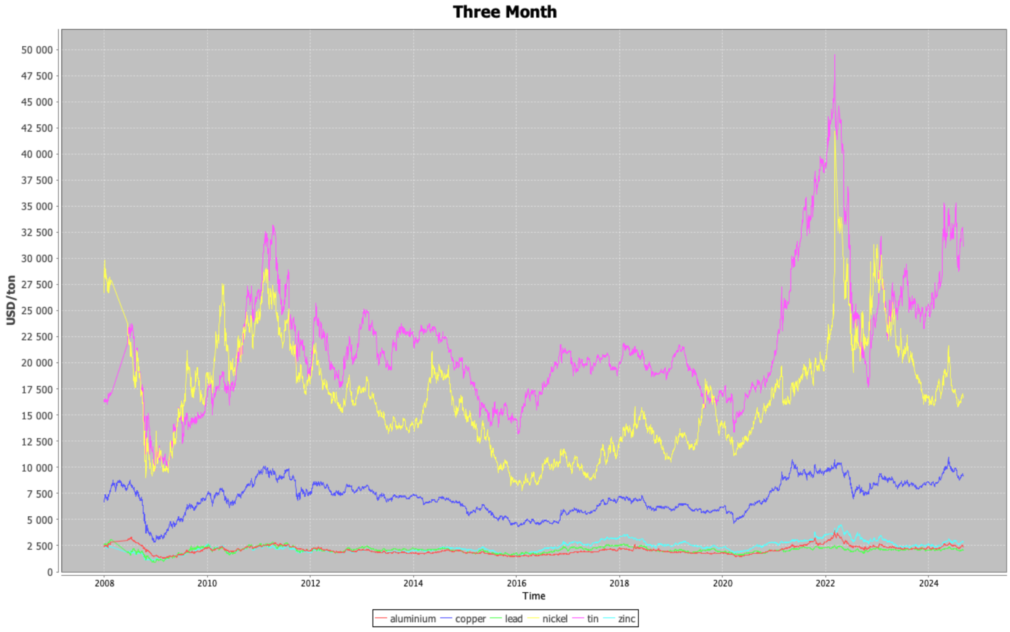


Figure 5.8: Three Month of the Metals

Metal	Mean	Median	Variance
Aluminium	2.069E+3	2.018E+3	1.501E+5
Copper	7.115E+3	7.062E+3	2.479E+6
Lead	2.072E+3	2.090E+3	8.470E+4
Nickel	1.657E+4	1.625E+4	2.565E+7
Tin	2.169E+4	2.050E+4	3.829E+7
Zinc	2.374E+3	2.315E+3	3.095E+5

Table 5.7: Statistics for the Three Month

As expected, the series look similar. To get a sense of the possible predictive power of utilizing the three month one can plot the difference between the three month and the cash settlement where a large deviation indicates a strong conviction in the market of upcoming changes.

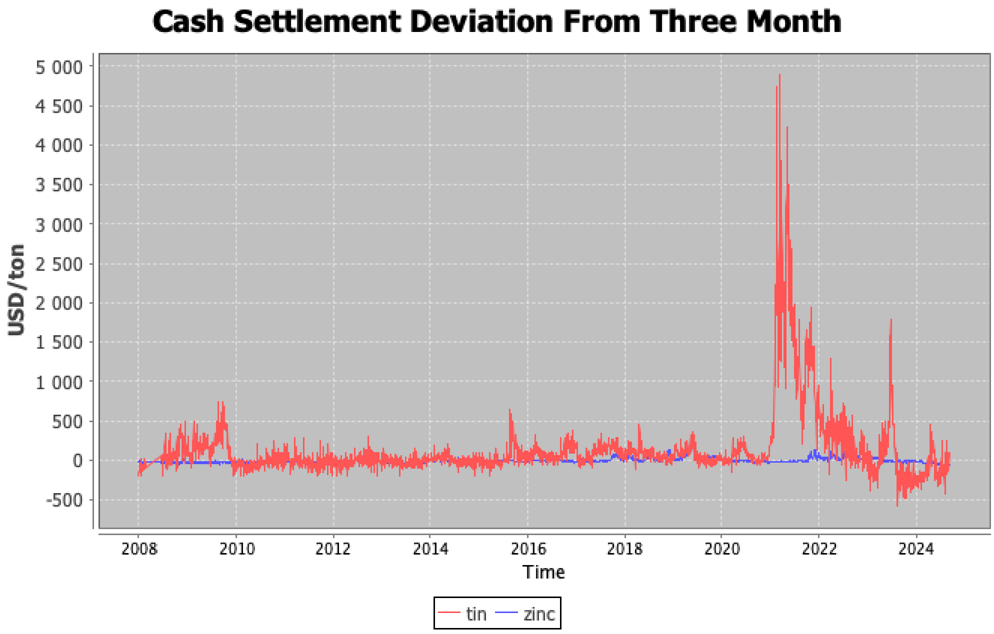


Figure 5.9: Cash Settlement Deviation from Three Month

The plot illuminates the notion that the different metals behave slightly different. Zinc barely has any deviation at all while tin a times have massive differences between cash and three month. There might also be some indication that the developments of tin during 2021-2022 are exceptional. This makes predictions using machine learning in general, and reservoir computing models in particular, difficult considering there might not enough data the model can learn from. It should be noted that the plots so far depict absolute terms. Later relative price changes will be considered hopefully revealing a more stationary behavior, which is more suited for machine learning despite the online learning scheme.

Below is the final type of data used in the study, the stock of the different metals.

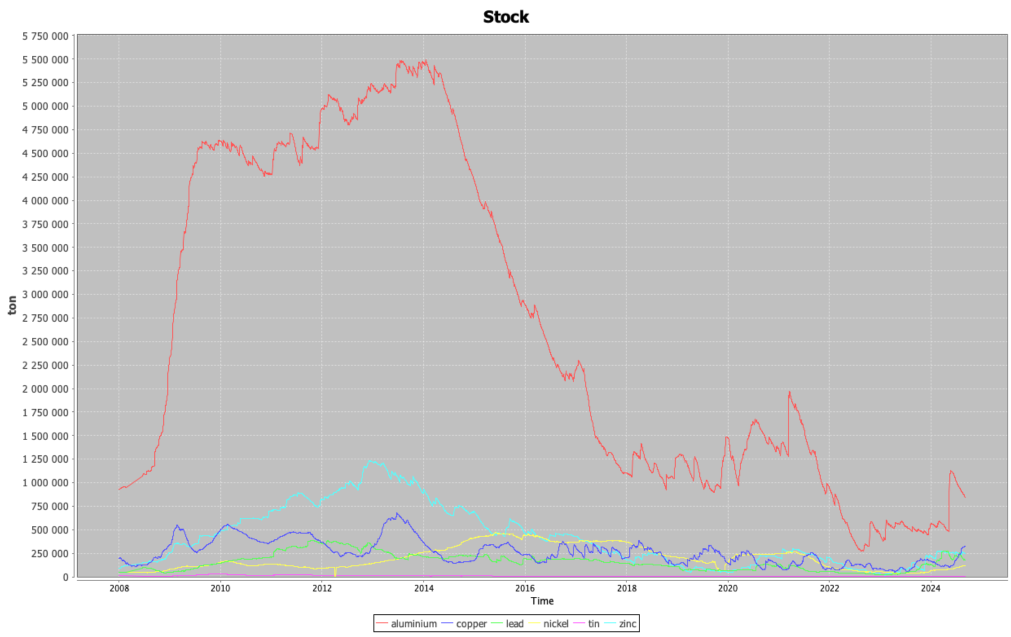


Figure 5.10: Stock of the Metals

Metal	Mean	Median	Variance
Aluminium	2.653E+6	2.099E+6	3.099E+12
Copper	2.619E+5	2.429E+5	1.802E+10
Lead	1.574E+5	1.488E+5	8.350E+9
Nickel	2.000E+5	1.615E+5	1.577E+10
Tin	8.290E+3	6.283E+3	4.056E+7
Zinc	4.247E+5	3.358E+5	1.018E+11

Table 5.8: Statistics for the Stock

The plot perfectly illustrates the need for using relative terms. The absolute difference in aluminium stock and tin stock is massive. The graph should be viewed in the light of aluminium volumes sold at the LME for 2023 being around 57 million tons and tin volumes being around 1.2 million tons [53], which naturally will be reflected in the absolute stock kept.

There is also an apparent issue with the way Westmetall collects data from the LME. The nickel stock in the beginning of April 2012 is completely wrong. On March 30 the nickel stock presented is 99882 ton which drops to 542 ton on April 2 and stays at this level until April 11 when the stock goes back to more reasonable levels of 99330 ton. Since there is no information on there being something exceptional happening during this period the data is suspicious. Luckily there is a snapshot of the LME website, which posts daily data, indicating that the nickel stock on April 3 2012 should be 100542 ton [47]. It is probably safe to assume that the software used to collect the data from the LME by Westmetall has somehow removed 100 000 from the actual stock kept. Therefore 100 000 is added back in the data used for the suspicious stock values between March 30 and April 11 2012. No similar cases are found throughout the dataset but other errors cannot be ruled out possibly affecting the effectiveness of the training and validity of the results.

Extrapolating from basic economic theory on supply and demand there should be a relationship between the stock kept and the prices. This might have predictive power in the model. Therefore, it is of interest to see the cash settlement and stock in the same graph. Some examples are presented below.

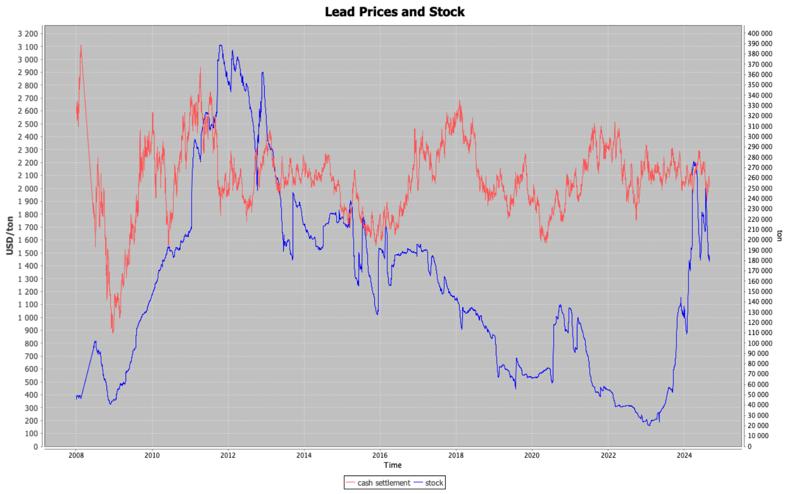


Figure 5.11: Lead Cash Settlement and Stock

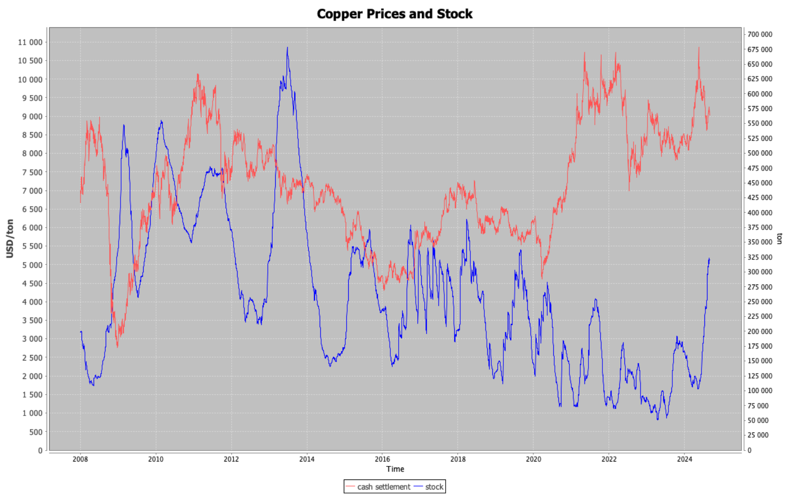


Figure 5.12: Copper Cash Settlement and Stock

No strikingly obvious relationship is observed for either of the examples. Further speculation is pointless, and the usefulness of the stock will hopefully be evident in the subsequent optimization of the ESN.

Next the percentage changes of the data is presented.

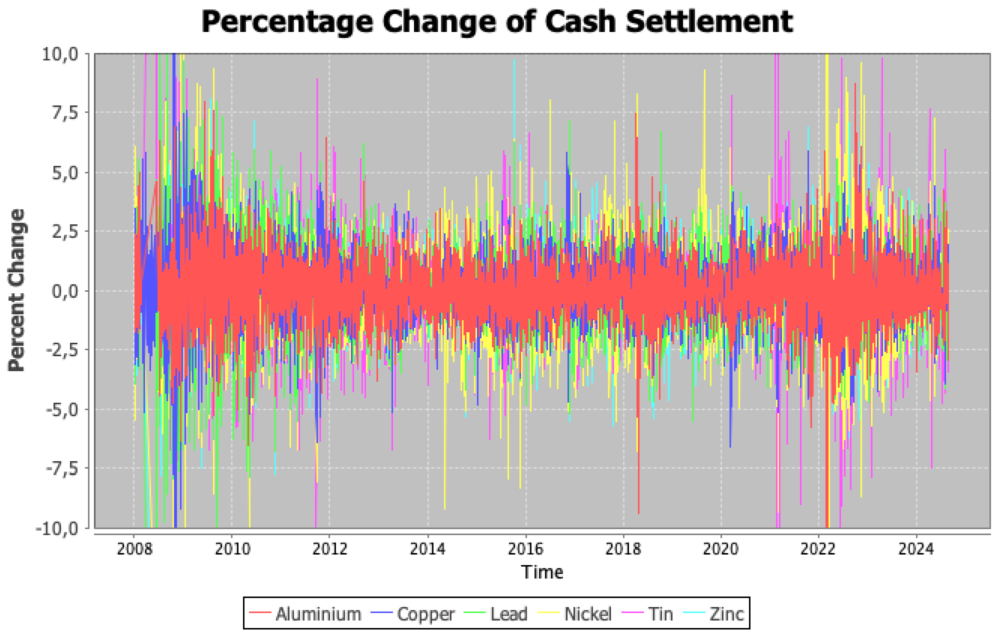


Figure 5.13: Percent Change of Cash Settlement of the Metals

Metal	Mean	Median	Variance
Aluminium	0.0107	0.0000	2.0825
Copper	0.0193	0.0025	2.4216
Lead	0.0146	0.0000	3.9414
Nickel	0.0136	0.0000	5.1395
Tin	0.0353	0.0577	3.8550
Zinc	0.0203	0.0210	3.2898

Table 5.9: Statistics for the Percentage Change of Cash

The cash percentage changes appear to be quite noisy, essentially centered around zero. This is in accordance with a random walk model. There are however some larger outliers, not indicative of normally distributed data. This is reflected in the slight differences in mean and median. Also, the variance appears to not be constant indicating some heteroskedasticity.

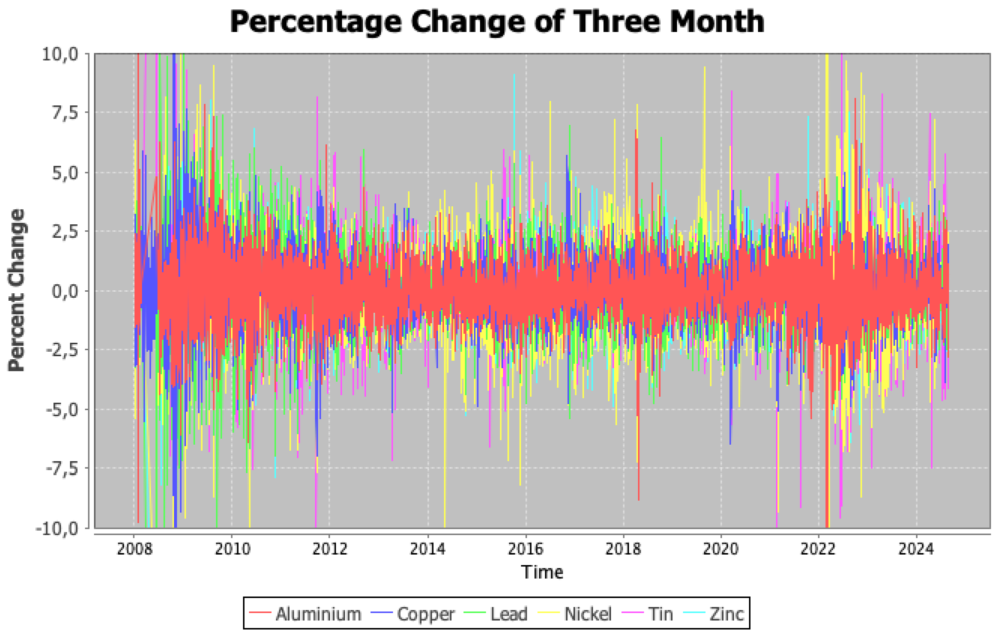


Figure 5.14: Percent Change of Three Month of the Metals

Metal	Mean	Median	Variance
Aluminium	0.0098	0.0000	1.9617
Copper	0.0193	0.0143	2.3787
Lead	0.0135	0.0000	3.6631
Nickel	0.0135	0.0000	5.1202
Tin	0.0328	0.0454	3.4251
Zinc	0.0194	0.0190	3.0547

Table 5.10: Statistics for the Percentage Change of Three Month

The data of the three month is similar to the cash. Though the data can be said to be slightly less skewed in distribution considering the slightly smaller differences in median and mean.

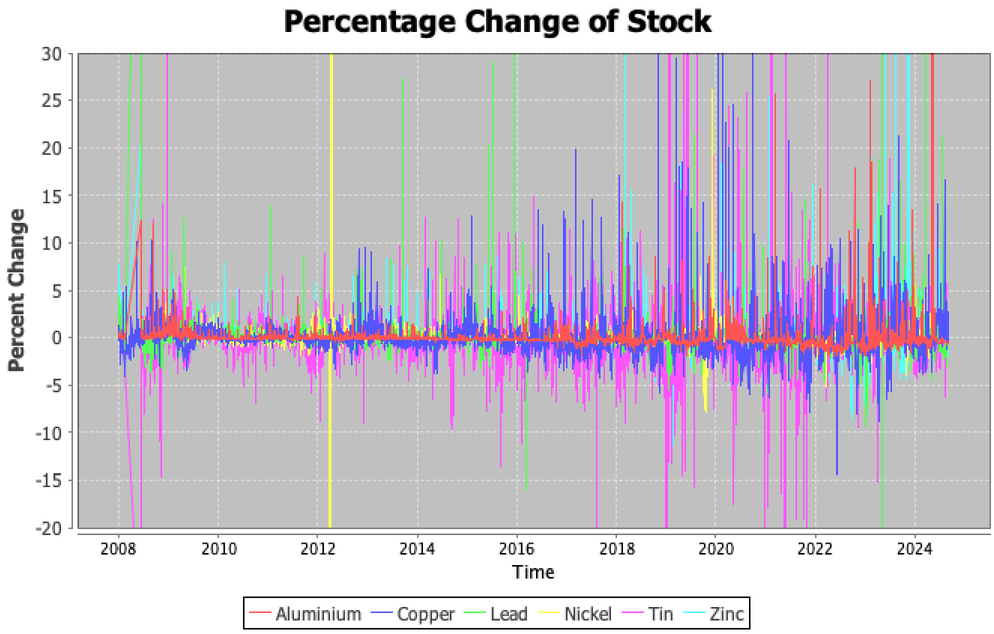


Figure 5.15: Percent Change of Stock of the Metals

Metal	Mean	Median	Variance
Aluminium	0.0118	-0.1611	3.6109
Copper	0.0446	-0.2633	7.2488
Lead	0.0596	-0.0743	6.6263
Nickel	0.0267	-0.0495	0.9469
Tin	0.0456	0.0000	15.8497
Zinc	0.0566	-0.1938	7.9666

Table 5.11: Statistics for the Percentage Change of Stock

The stock percentage changes do not appear to be normally distributed. There is also an apparent skewness, where slightly negative changes are more common reflected in the median. A probable explanation is that the warehouses are continuously emptied and intermittently get large deliveries.

To summarize the transformed data appears more suitable for usage in an ESN. This was confirmed while testing different setups, where using the raw data did not produce any meaningful results at all.

5.3 Parameter Optimization

The genetic algorithm was run with the parameters described in the methods section for the specified genome. The progression of the MSE on the testing and validation sets follow:

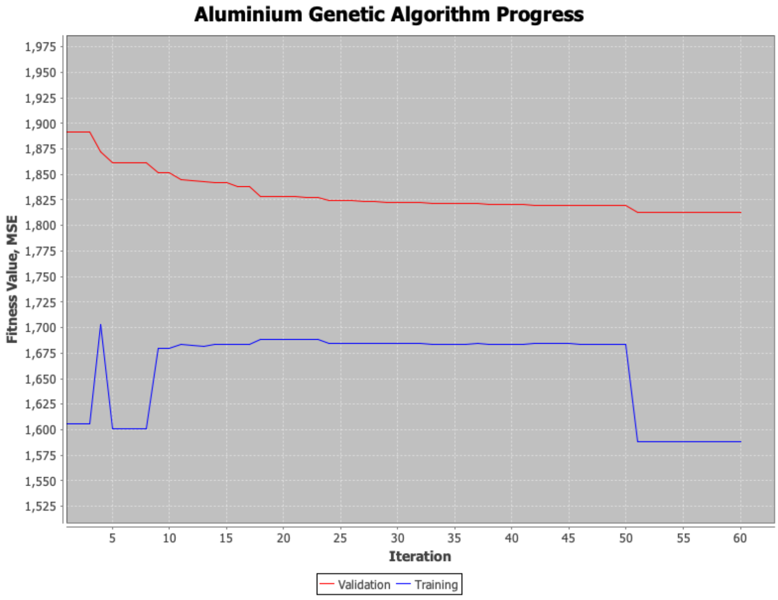


Figure 5.16: Aluminium Genetic Algorithm Progress for Training and Validation

The GA for aluminium is still improving at the end of the run. Probably more iterations would be beneficial. Considering the performance gains at the very end are marginal there might be some premature convergence present.

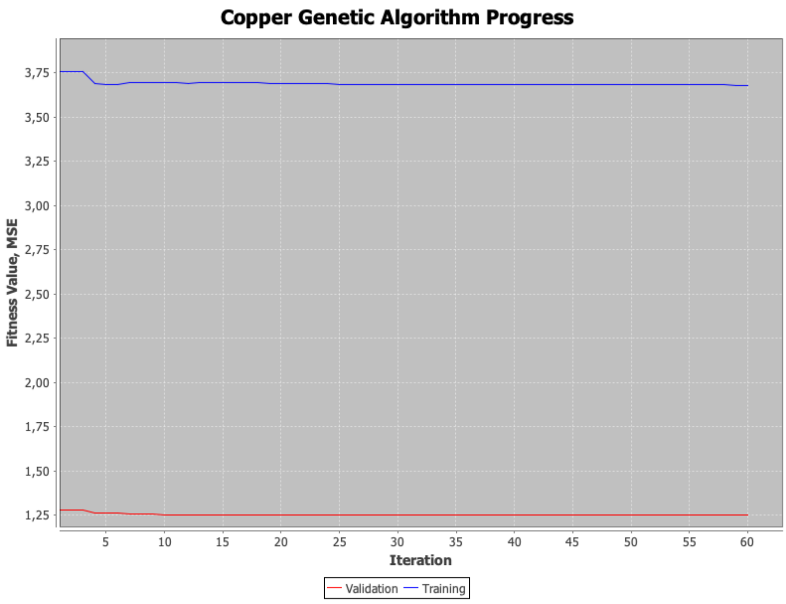


Figure 5.17: Copper Genetic Algorithm Progress for Training and Validation

The GA performance for copper barely improves. Either a lucky initial state is found, premature convergence or there is not much room to improve at all. This can occur if the data is inherently random.

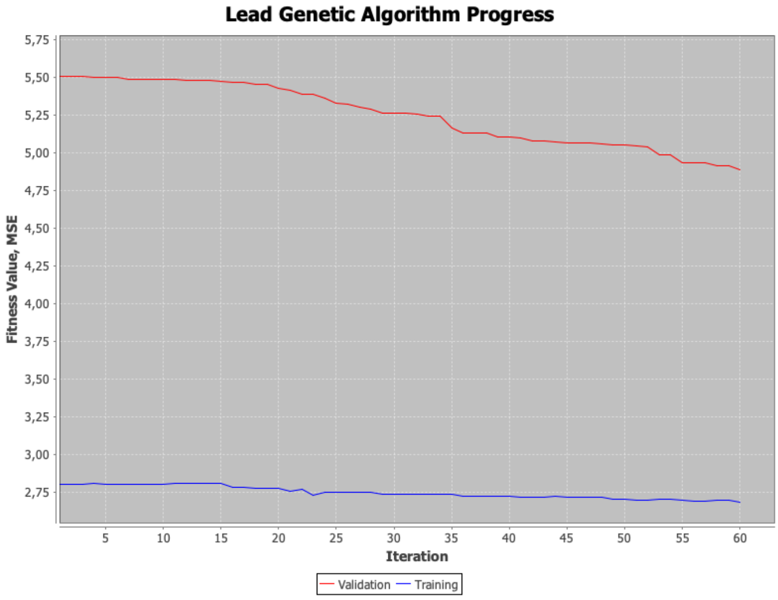


Figure 5.18: Lead Genetic Algorithm Progress for Training and Validation

Lead is seemingly still improving when the GA terminates. There might be reason to do a re-run with longer duration.

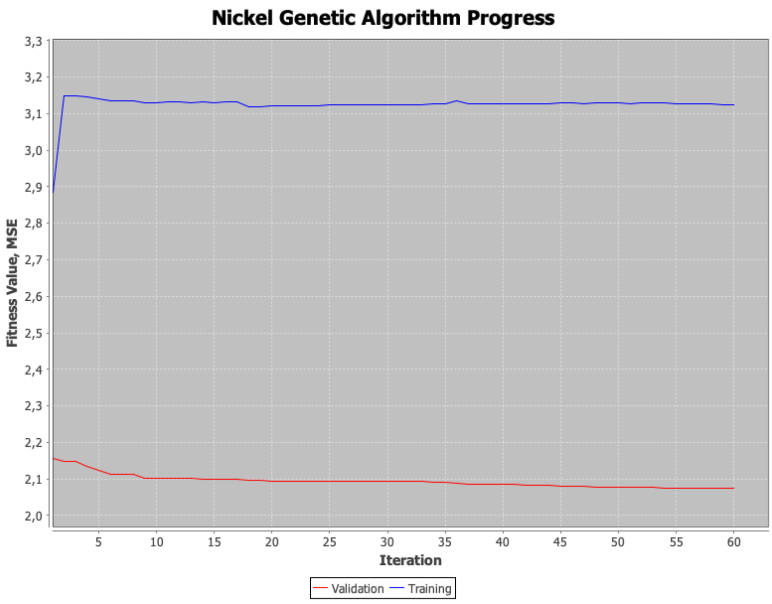


Figure 5.19: Nickel Genetic Algorithm Progress for Training and Validation

Seems to have found satisfactory parameters for the validation set. The algorithm never recovers the initial training error, this might indicate overfitting. However the error stabilizes and is not increasing towards the end.

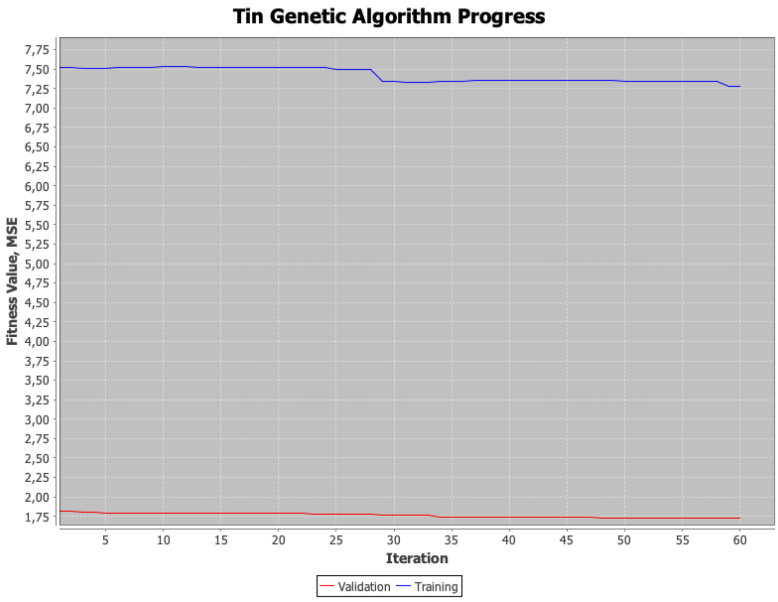


Figure 5.20: Tin Genetic Algorithm Progress for Training and Validation

The validation does not progress much. Possibly a premature convergence or maybe lucky initial parameters. Considering the training error is improving during the period, the seemingly minimal progress on the validation set the GA has some positive effect. The Tin parameters should be good enough.

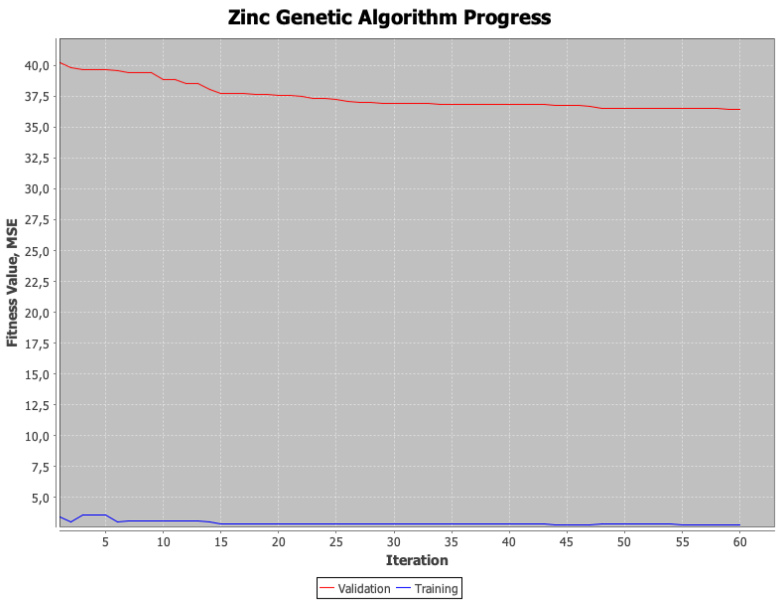


Figure 5.21: Zinc Genetic Algorithm Progress for Training and Validation

The MSE on the validation set is still improving. Possibly more generations would benefit performance. There is also a relatively large error present. Perhaps some outlier is present skewing the results.
The progression for the metals behaves differently from the benchmarking functions. There are some improvements but only marginal. Note that all the GA plots for the

metals are on a linear scale while the benchmarking functions are on a logarithmic. The improvements for the benchmarking functions are more indicative of other applications of genetic algorithms. There is a possibility that there is not much rooms for improvement in the case of the metals, which would be an indication that the data is more random than chaotic and the model cannot make accurate predictions in the short term.

Regardless, here are the parameters found and used for subsequent testing.

Parameter	Aluminium	Copper	Lead	Nickel	Tin	Zinc
Topology	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free
Connectivity	0.033	0.023	0.053	0.057	0.037	0.095
Spectral Radius	0.614	0.968	0.544	0.515	0.657	0.562
Regularization	1.135E-5	34.088	7.844E-2	185.198	4.410	2.653E-3
Learning Rate	1.074E-3	2.487E-3	4.832E-3	4.249E-5	1.416E-6	3.106E-5
Leakage Rate	0.170	0.706	0.971	0.380	0.234	0.508
Reservoir Size	687	861	632	604	761	1174
In Length	134	115	128	141	415	514
Out Length	134	115	128	141	415	514
In Bias Scale	19.316	15.825	0.197	7.064	92.472	25.906
In Factor Scale	[1.175, 2.475, 1.939]	[2.010, 0.719, 0.939]	[0.829, 0.540, 0.321]	[2.874, 2.469, 0.021]	[2.056, 1.157, 2.967]	[0.206, 2.797, 2.993]

Table 5.12: GA Optimized Parameters for the Metals

Note that the parameters are stored as doubles, here they are rounded to three decimal points. The parameters are usually specific for each application but some general observations can be made. For all the optimized parameters found, the scale free topology is preferred. Since the scale free topology does not require optimization of the connectivity parameter, the values presented are redundant. This might be a flaw in the set up, considering a uniformly random network requires an extra parameter to be optimized, it might be harder to find good solutions making the GA settle on the scale free topology.

Another notable trend is that the input and output lengths are the same in every instance. Note that the algorithm allows them to be different although not larger than the reservoir size. This is adjusted in the decoding, not in the individual chromosomes, such that a relatively small reservoir would not force the in and out lengths to the same smaller value and then in subsequent generations increase the reservoir size. An explanation could possibly be found in an analysis of the actual connections in the network, which is beyond the scope of this thesis.

5.4 Testing Results

When testing the optimized parameters the networks are recreated using the seed stored during the GA such that all random choices are the same as in the GA. The results are presented with a plot of the predictions during validation and testing for both the percentage change and the reverted values accompanied with a set of metrics which are compared to a random walk model. First the metals are handled individually and then summarized. Note that metrics in bold denote better performance for the specific data set and metric.

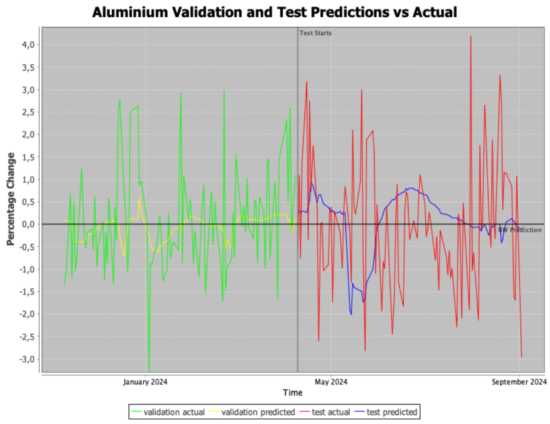


Figure 5.22: Aluminium Predictions Percentage Change

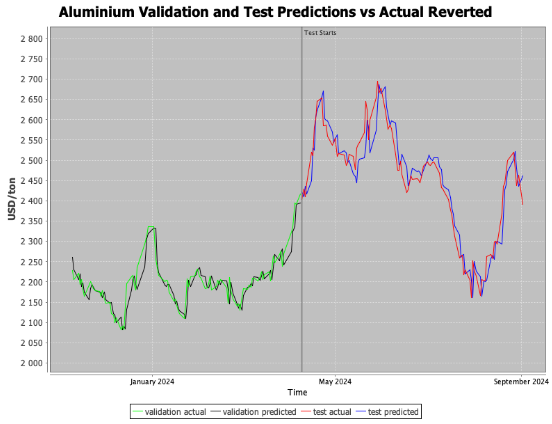


Figure 5.23: Aluminium Predictions Reverted

Table 5.13: Model Metrics Compared With Random Walk (RW) for Aluminium

Aluminium	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Percentage Change	MSE	**1.219e+00**	1.289e+00	2.534e+00	**1.925e+00**
	MAE	**8.303e-01**	8.742e-01	1.230e+00	**1.053e+00**
	R^2	**0.051**	-0.004	-0.316	**-0.000**
	Directional Accuracy	**0.660**	0.485	0.422	**0.510**
Reverted	MSE	**5.918e+02**	6.286e+02	1.525e+03	**1.137e+03**
	MAE	**1.827e+01**	1.926e+01	3.019e+01	**2.574e+01**
	R^2	**0.842**	0.832	0.909	**0.932**

The model demonstrates some ability to identify patterns in the validation set, as it outperforms the Random Walk (RW) benchmark across all metrics, including lower MSE and MAE, higher R^2, and better directional accuracy. This suggests that the model successfully captures underlying signals in the aluminium validation data.

However, the reverse is true for the test set, where the RW performs better in all metrics. Specifically, the model's significantly higher MSE and MAE, coupled with poorer directional accuracy, indicate it may be overfitting to the noise or specific characteristics of the validation set. This overfitting results in a lack of generalization to unseen data, as evidenced by its inferior performance on the test set.

While the model shows promise during validation, its inability to outperform the RW on the test set raises concerns about overfitting, necessitating further refinement or regularization to improve generalization.

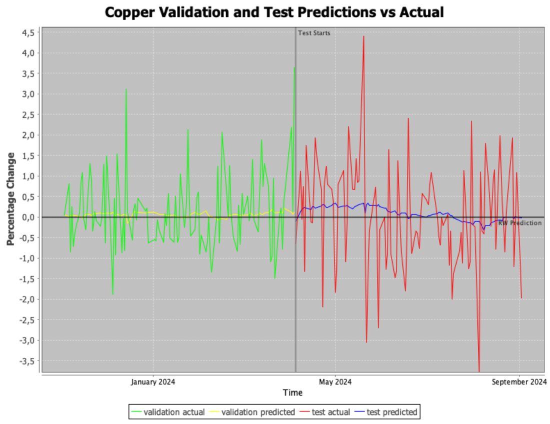


Figure 5.24: Copper Predictions Percentage Change

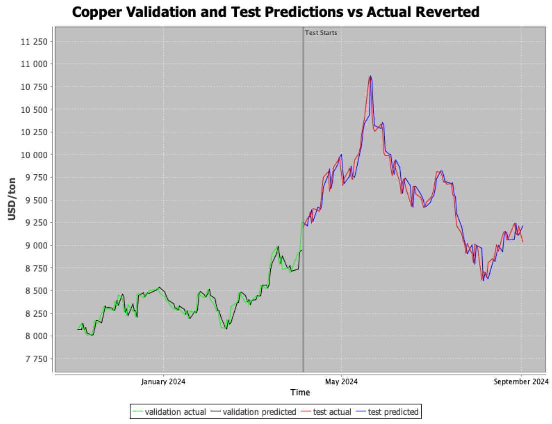


Figure 5.25: Copper Predictions Reverted

Table 5.14: Model Metrics Compared With Random Walk (RW) for Copper

Copper	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Percentage Change	MSE	**8.376e-01**	8.391e-01	1.608e+00	**1.585e+00**
	MAE	6.706e-01	**6.671e-01**	**9.738e-01**	9.800e-01
	R^2	**-0.020**	-0.022	-0.014	**-0.000**
	Directional Accuracy	0.476	**0.524**	**0.519**	0.452
Reverted	MSE	**6.003e+03**	6.018e+03	1.503e+04	**1.478e+04**
	MAE	5.643e+01	**5.615e+01**	**9.329e+01**	9.379e+01
	R^2	**0.886**	0.885	0.930	**0.931**

The comparison of the model and the RW baseline for copper reveals a mixed performance. In terms of MSE, the model demonstrates slightly better performance during validation, achieving a lower MSE compared to RW. However, in the test dataset, the RW exhibits superior performance with a lower MSE, suggesting it generalizes better for this metric.

For MAE, RW outperforms the model on the validation dataset, maintaining lower average errors. On the test dataset, the model achieves a marginally better MAE, highlighting its competitive performance in this regard.

Achieving superior MAE and inferior MSE indicates the model better handles the less common larger errors worse than the more common smaller errors.

The R^2 scores indicate a minimal difference between the two methods, with the model achieving slightly higher values during validation. However, RW outperforms the model on the test dataset, suggesting it explains variance slightly better in unseen data.

Overall there is not a clearly superior model of the two, a step in the right direction from the results seen for aluminium.

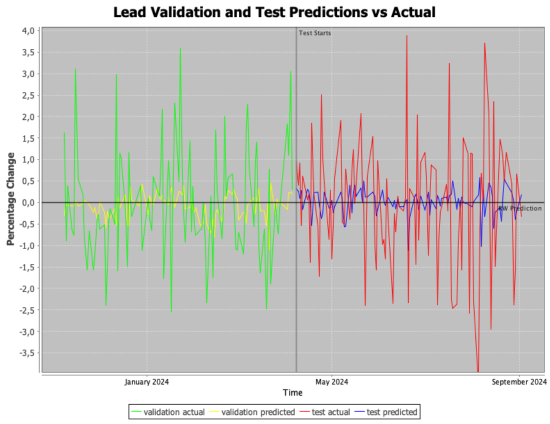


Figure 5.26: Lead Predictions Percentage Change

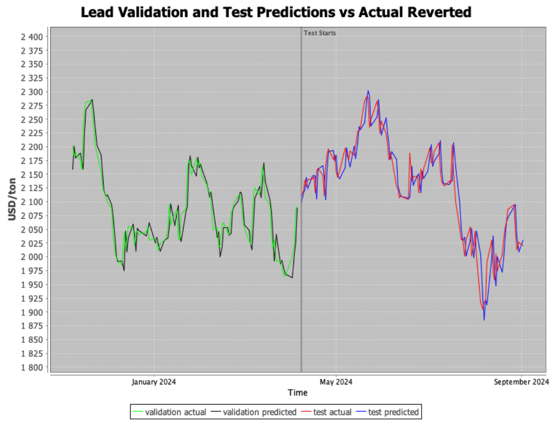


Figure 5.27: Lead Predictions Reverted

Table 5.15: Model Metrics Compared With Random Walk (RW) for Lead

Lead	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Percentage Change	MSE	1.609e+00	**1.578e+00**	2.215e+00	**2.070e+00**
	MAE	**9.845e-01**	9.859e-01	1.151e+00	**1.116e+00**
	R^2	-0.020	**-0.000**	-0.070	**-0.000**
	Directional Accuracy	**0.553**	0.456	0.485	**0.534**
Reverted	MSE	6.947e+02	**6.850e+02**	9.698e+02	**9.126e+02**
	MAE	**2.053e+01**	2.059e+01	2.428e+01	**2.357e+01**
	R^2	0.867	**0.869**	0.874	**0.881**

For Lead, the model underperforms in nearly every metric compared to the RW, with the exceptions of MAE and directional accuracy on the validation set. This suggests that the genetic algorithm may struggle to converge toward a solution akin to RW, which could serve as a robust baseline. The model's predictions appear to closely mirror the percentage changes, resulting in more pronounced spikes that suggest greater complexity. However, given that this complexity fails to translate into superior performance, as reflected in the metrics, it does not necessarily represent an advantage. These results underscore the need to refine the model's learning process to better balance complexity and predictive accuracy.

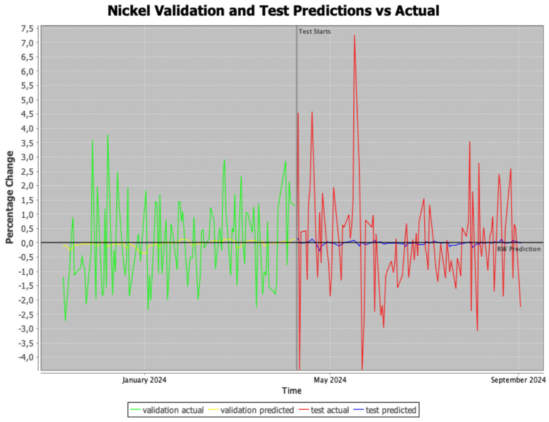


Figure 5.28: Nickel Predictions Percentage Change

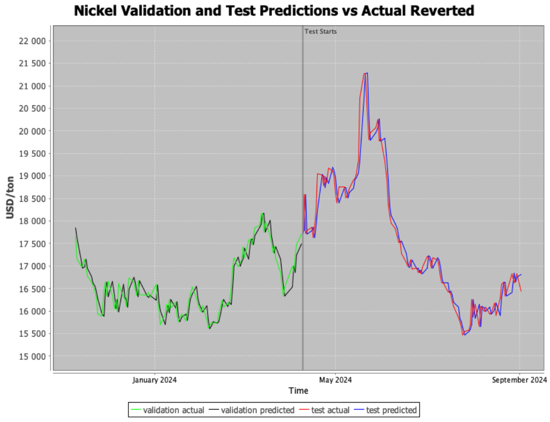


Figure 5.29: Nickel Predictions Reverted

Table 5.16: Model Metrics Compared With Random Walk (RW) for Nickel

Nickel	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Percentage Change	MSE	**2.053e+00**	2.066e+00	**2.965e+00**	2.991e+00
	MAE	**1.220e+00**	1.234e+00	**1.244e+00**	1.249e+00
	R^2	**0.007**	-0.000	**0.007**	-0.001
	Directional Accuracy	**0.592**	0.485	**0.569**	0.480
Nickel	MSE	**5.641e+04**	5.688e+04	**9.956e+04**	1.004e+05
	MAE	**2.025e+02**	2.049e+02	**2.222e+02**	2.231e+02
	R^2	**0.862**	0.861	0.950	0.950

The model for Nickel demonstrates superior performance compared to the RW across all metrics for both the validation and test datasets. Notably, the model's predictions exhibit a conservative nature, often remaining close to zero. This conservative approach, while reducing the magnitude of changes, still manages to achieve results that consistently outperform the RW baseline. This raises an intriguing question: why is the genetic algorithm employed in other models unable to converge on similarly effective solutions? This disparity suggests that the optimization process may be influenced by factors unique to the Nickel dataset, highlighting potential areas for further investigation to generalize the approach across other metals. Another possibility is that the GA isn't able to overfit the validation set in during the 60 generations and therefore settling on a more general solution.

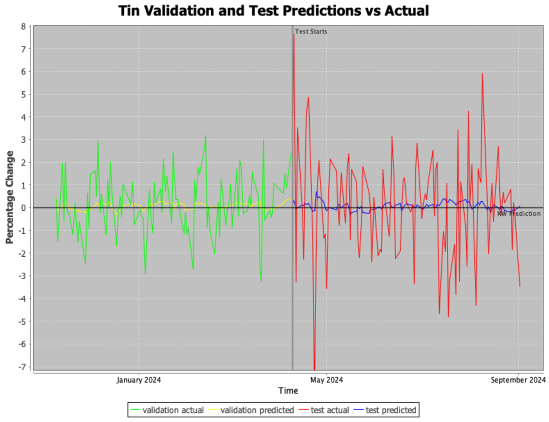


Figure 5.30: Tin Predictions Percentage Change

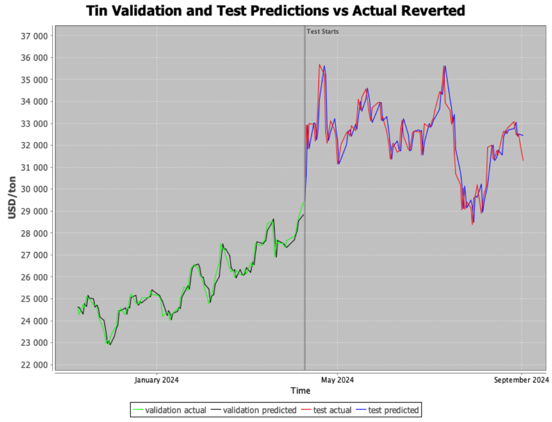


Figure 5.31: Tin Predictions Reverted

Table 5.17: Model Metrics Compared With Random Walk (RW) for Tin

Tin	**Metric**	**Model (Validation)**	**RW (Validation)**	**Model (Test)**	**RW (Test)**
Percentage Change	MSE	**1.619e+00**	1.788e+00	**5.527e+00**	5.582e+00
	MAE	**9.984e-01**	1.059e+00	1.816e+00	**1.810e+00**
	R^2	**0.077**	-0.019	**0.008**	-0.001
	Directional Accuracy	**0.680**	0.524	0.500	**0.569**
Reverted	MSE	**1.084e+05**	1.193e+05	**5.739e+05**	5.782e+05
	MAE	**2.566e+02**	2.721e+02	5.845e+02	**5.822e+02**
	R^2	**0.947**	0.942	**0.738**	0.736

For Tin, the model demonstrates competitive performance against the Random Walk (RW), though not as dominant as the Nickel model. On the test set, the model achieves better results in terms of MSE and comes close in MAE, suggesting that it handles larger, less frequent errors more effectively while underperforming on smaller, more common ones. The predictions are again notably conservative, often hovering around zero. This observation reinforces the need to investigate why the underperforming models fail to converge on similarly restrained solutions, which seem to be effective in outperforming the RW baseline.

It is also worth noting that achieving a directional accuracy above 50% is significant for the model, as it indicates possibly better-than-random performance in predicting the direction of changes. However, the RW directional accuracy metric serves primarily as a reference for the proportion of upward movements rather than as a direct prediction, since RW does not produce explicit directional forecasts.

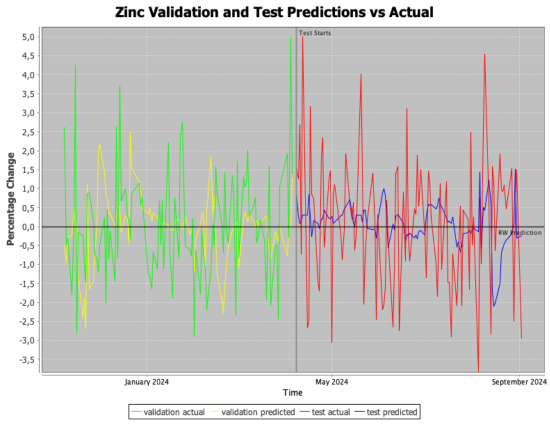


Figure 5.32: Zinc Predictions Percentage Change

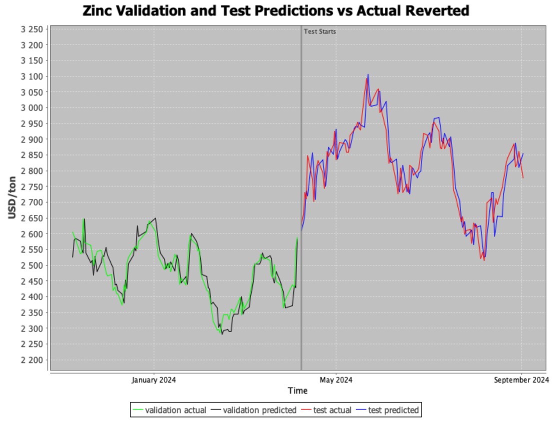


Figure 5.33: Zinc Predictions Reverted

Table 5.18: Model Metrics Compared With Random Walk (RW) for Zinc

Zinc	**Metric**	**Model (Validation)**	**RW (Validation)**	**Model (Test)**	**RW (Test)**
Percentage Change	MSE	2.667e+00	**2.044e+00**	3.433e+00	**2.999e+00**
	MAE	1.280e+00	**1.103e+00**	1.511e+00	**1.400e+00**
	R^2	-0.305	**-0.000**	-0.148	**-0.002**
	Directional Accuracy	**0.553**	0.456	0.485	**0.553**
Reverted	MSE	1.620e+03	**1.250e+03**	2.667e+03	**2.317e+03**
	MAE	3.158e+01	**2.730e+01**	4.226e+01	**3.910e+01**
	R^2	0.778	**0.828**	0.831	**0.853**

The Zinc model produces the most erratic predictions observed thus far, leading to underperformance against the Random Walk (RW) in almost every metric. This reinforces the pattern that overly complex predictions tend to degrade performance when compared to more restrained approaches. Interestingly, the arbitrary choice to record the RW's directional accuracy as all upward movements has notable implications: if the direction chosen was downwards, the model would underperform on the validation set while outperforming on the test set.

Despite this, the model achieves a directional accuracy above 50% on the validation set, suggesting a potential signal in the predictions.

To more easily summarize the results all respective metrics are gathered in the following tables:

Table 5.19: Model Comparison with Random Walk (RW) for Validation and Test Results (Percentage Changes)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**1.219e+00**	1.289e+00	2.534e+00	**1.925e+00**
	MAE	**8.303e-01**	8.742e-01	1.230e+00	**1.053e+00**
	R^2	**0.051**	-0.004	-0.316	**-0.000**
	Directional Accuracy	**0.660**	0.485	0.422	**0.510**
Copper	MSE	**8.376e-01**	8.391e-01	1.608e+00	**1.585e+00**
	MAE	6.706e-01	**6.671e-01**	**9.738e-01**	9.800e-01
	R^2	**-0.020**	-0.022	-0.014	**-0.000**
	Directional Accuracy	0.476	**0.524**	**0.519**	0.452
Lead	MSE	1.609e+00	**1.578e+00**	2.215e+00	**2.070e+00**
	MAE	**9.845e-01**	9.859e-01	1.151e+00	**1.116e+00**
	R^2	-0.020	**-0.000**	-0.070	**-0.000**
	Directional Accuracy	**0.553**	0.456	0.485	**0.534**
Nickel	MSE	**2.053e+00**	2.066e+00	**2.965e+00**	2.991e+00
	MAE	**1.220e+00**	1.234e+00	**1.244e+00**	1.249e+00
	R^2	**0.007**	-0.000	**0.007**	-0.001
	Directional Accuracy	**0.592**	0.485	**0.569**	0.480
Tin	MSE	**1.619e+00**	1.788e+00	**5.527e+00**	5.582e+00
	MAE	**9.984e-01**	1.059e+00	1.816e+00	**1.810e+00**
	R^2	**0.077**	-0.019	**0.008**	-0.001
	Directional Accuracy	**0.680**	0.524	0.500	**0.569**
Zinc	MSE	2.667e+00	**2.044e+00**	3.433e+00	**2.999e+00**
	MAE	1.280e+00	**1.103e+00**	1.511e+00	**1.400e+00**
	R^2	-0.305	**-0.000**	-0.148	**-0.002**
	Directional Accuracy	**0.553**	0.456	0.485	**0.553**

Table 5.20: Model Comparison with Random Walk (RW) for Validation and Test Results (Reverted)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**5.918e+02**	6.286e+02	1.525e+03	**1.137e+03**
	MAE	**1.827e+01**	1.926e+01	3.019e+01	**2.574e+01**
	R^2	**0.842**	0.832	0.909	**0.932**
Copper	MSE	**6.003e+03**	6.018e+03	1.503e+04	**1.478e+04**
	MAE	5.643e+01	**5.615e+01**	**9.329e+01**	9.379e+01
	R^2	**0.886**	0.885	0.930	**0.931**
Lead	MSE	6.947e+02	**6.850e+02**	9.698e+02	**9.126e+02**
	MAE	**2.053e+01**	2.059e+01	2.428e+01	**2.357e+01**
	R^2	0.867	**0.869**	0.874	**0.881**
Nickel	MSE	**5.641e+04**	5.688e+04	**9.956e+04**	1.004e+05
	MAE	**2.025e+02**	2.049e+02	**2.222e+02**	2.231e+02
	R^2	**0.862**	0.861	0.950	0.950
Tin	MSE	**1.084e+05**	1.193e+05	**5.739e+05**	5.782e+05
	MAE	**2.566e+02**	2.721e+02	5.845e+02	**5.822e+02**
	R^2	**0.947**	0.942	**0.738**	0.736
Zinc	MSE	1.620e+03	**1.250e+03**	2.667e+03	**2.317e+03**
	MAE	3.158e+01	**2.730e+01**	4.226e+01	**3.910e+01**
	R^2	0.778	**0.828**	0.831	**0.853**

For the validation set, the model demonstrates a clear ability to identify networks that outperform the RW in terms of both MSE and MAE. This indicates that the genetic algorithm is effectively optimizing for MSE, as expected given its design. The strong correlation between MSE and MAE reinforces the reliability of this optimization process.

However, the true test of model performance lies in its ability to generalize to unseen data in the test set. Here, the results diverge significantly. Only a subset of the models, particularly those with more conservative predictions such as for Copper, Lead, and Nickel, are competitive with the RW. Conversely, models that make more volatile predictions tend to underperform on the test set, which suggests overfitting or poor generalization.

This pattern highlights an important insight: while the genetic algorithm can effectively minimize error on the validation set, its capacity to ensure robustness on the test set is limited for more erratic models. This raises questions about how the algorithm and model architecture can be adjusted to favor conservative, generalizable predictions.

The R^2 metric is the only one that is really interesting to investigate for both the percentage changes and the reverted metrics. Where the directional accuracy is exactly the same, thus omitted, and the MSE and MAE will have different absolute values but the relationship to the random walk model will be the same. If the model has a superior MSE in percentage change this will still hold for the reverted values. In the case of R^2 the metric is calculated differently for the two categories of metrics. For the reverted metrics the benchmark is the average value of the test set and therefore the average price. Considering time series of this sort where the data is highly dependent, just using the average price during a period will not be a good prediction. This is reflected in the metrics where a random walk model explains more than 80% of the variance in every single case. Then when it comes to the R^2 values for the percentage changes, a value close to zero would indicate that the average change during the period is close to zero. Because the R^2 values for the random walk in percentage changes are all slightly negative that means there is a slight trend in the data for the period. Since the models for Tin and Nickel produce a higher than zero R^2 the models seem to be able to find a solution better than the average change during the period.

The R^2 metric provides unique insights depending on whether it is evaluated on percentage changes or reverted metrics. While MSE and MAE exhibit consistent relative performance patterns between the two representations, R^2 varies because of the differences in how the baseline variance is computed.

For reverted metrics, R^2 measures the proportion of variance explained relative to the average value of the test set (the mean price). This baseline is quite naive in the context of time-series data, where values are highly autocorrelated. Consequently, the random walk (RW) generally performs exceptionally well, with R^2 exceeding 80% in all cases. This reflects the RW's capacity to closely approximate the underlying trends of the reverted series.

Interestingly, the models also achieve high R^2 scores on reverted metrics, sometimes outperforming the RW, as seen with Tin and Nickel. This suggests that these models are able to identify patterns that exceed the predictive power of simple trend-following methods, even in this stringent comparison.

In percentage changes, R^2 shifts its baseline to the average change over the period. For data with minimal trend (where average changes are close to zero), R^2 values for the RW tend to zero. The negative R^2 values for the RW suggest that it underperforms this baseline, due to the presence of subtle trends in the data that the RW cannot capture.

Notably, the models for Tin and Nickel stand out with positive R^2 values on percentage changes. This indicates that these models are identifying systematic patterns in the data that slightly exceed the average change over the period, a promising sign of predictive capability.

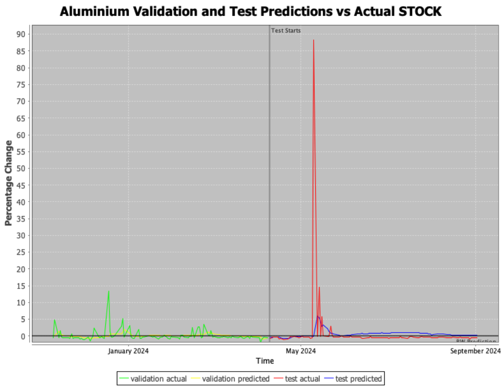
To assess the competency of the model in making directional predictions, one could argue that the model must outperform the baseline metric $\max(DA_{RW}, 1 - DA_{RW})$, where DA_{RW} refers to the directional accuracy of the RW. This implies that the model must consistently predict a direction that is better than simply choosing a random direction and maintaining it throughout the entire period.

This criterion holds for all cases except for Copper and Zinc on the validation set, where the models do not exceed the performance of the RW in directional accuracy. However, when we consider the test set, this condition is only satisfied by the Nickel model, which consistently outperforms the RW in directional accuracy.

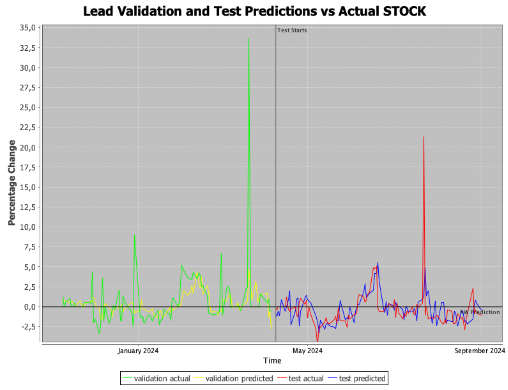
In conclusion, the standout performer is Nickel, followed by Tin in second place, and Copper in third. What these models have in common is their more conservative predictions. This trend is also reflected in the parameters chosen for the networks. Higher regularization, which reduces the complexity of the model, is a key characteristic of these top performers. Not surprisingly, the highest regularization coefficients are found in these three models.

A potential area for further investigation is understanding why the GA selects smaller regularization coefficients for some materials, such as Aluminium, where the coefficient is as low as $1.135e - 5$. One possible explanation lies in the fitness metric used by the GA. Since cash, three month, and stock data are all included in the MSE calculation, the algorithm is designed to balance the model's performance across these three components. This balance may be counterproductive when one type of data exhibits larger changes than the others, particularly since MSE assigns more weight to larger errors.

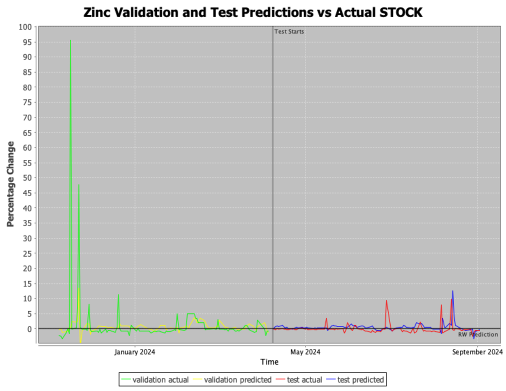
By inspecting the predictions made for the percentage changes in stock for the worst-performing models on the test set, we observe that the models fail to accurately capture significant shifts in the data, likely due to an imbalance in how the fitness function penalizes larger versus smaller errors across different data types. The plots of the stock predictions for the models follow:



(a) Aluminium



(b) Lead

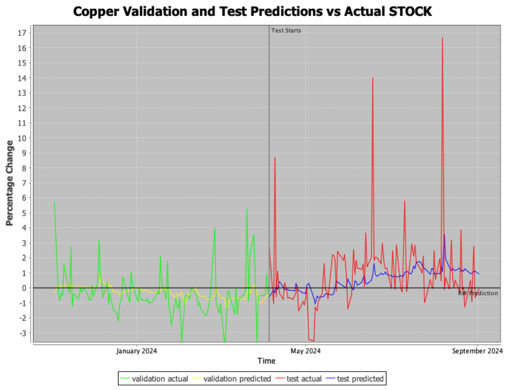


(c) Zinc

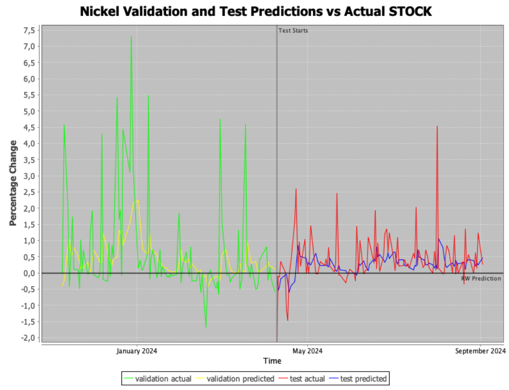
Figure 5.34: Validation and Test Predictions vs Actual Stock for Aluminium, Lead, and Zinc

Clearly, the few massive increases in stock prices have a significant impact on the model's performance. The solutions the GA identifies should heavily on capturing these dramatic spikes in the stock price, considering this is a good way to lower the MSE. This pursuit of capturing the extreme variations probably leads to more complex models, which in turn causes a trade-off in performance on the cash component of the model. The higher complexity required to model these large spikes in stock prices leads to overfitting for those outliers, which negatively affects the model's ability to generalize.

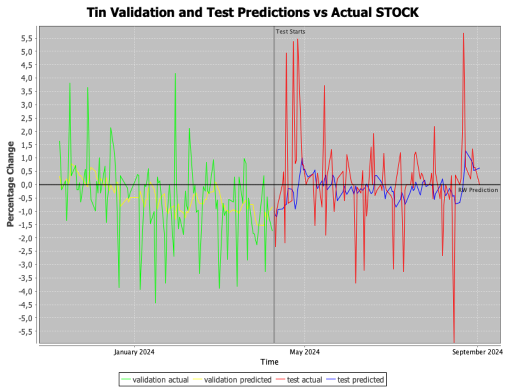
In contrast, when we inspect the predictions made by the more successful models, we observe the following:



(a) Copper



(b) Nickel



(c) Tin

Figure 5.35: Validation and Test Predictions vs Actual Stock for Copper, Nickel, and Tin

Note the difference in scale on the Y-axis. It is clear that the magnitude of the stock percentage changes present in the three more successful models is significantly smaller, especially on the validation set. Another notable observation is that the copper stock during the validation set, which is the set seen by the GA, does not contain any of the spikes seen in the subsequent test set.

A trivial solution to this issue could be to focus solely on the MSE for the cash dimension in the genetic algorithm. This approach is explored in the following section.

5.4.1 Results From Modified Fitness Function

The only modification made during this run of the GA is the fitness function. Previously the mean of the MSE of the three dimensions cash, three month and stock were used. Here the MSE for cash is what is used as a fitness function for the population in the GA. For conciseness most of the plots and tables for this run are included in Appendix A and present the metrics of the predictions and some plots of interest in this section. The metrics follow:

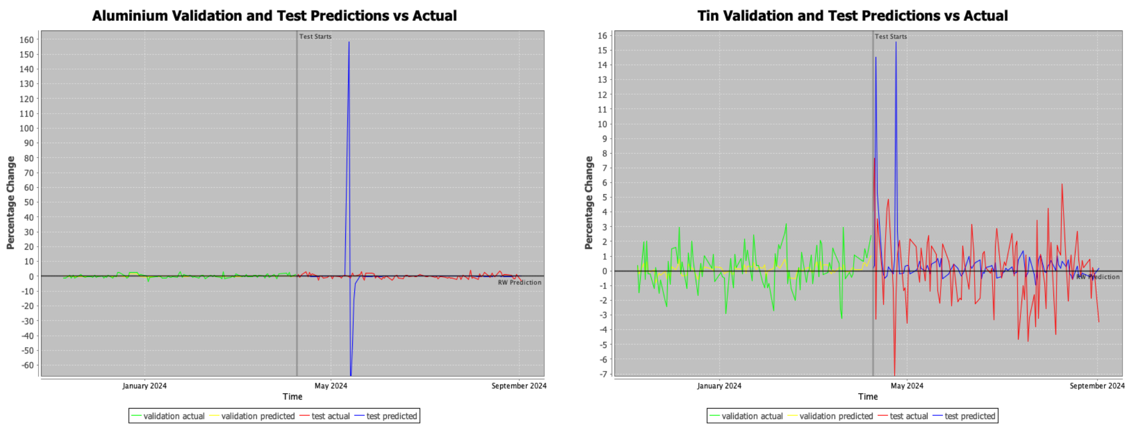
Table 5.21: Model Comparison with Random Walk (RW) for Validation and Test Results (Percentage Changes)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**1.126e+00**	1.289e+00	3.205e+02	**1.925e+00**
	MAE	**8.283e-01**	8.742e-01	3.943e+00	**1.053e+00**
	R^2	**0.123**	-0.004	-165.470	**-0.000**
	Directional Accuracy	**0.534**	0.485	0.412	**0.510**
Copper	MSE	**8.240e-01**	8.391e-01	**1.580e+00**	1.585e+00
	MAE	6.680e-01	**6.671e-01**	**9.787e-01**	9.800e-01
	R^2	**-0.004**	-0.022	**0.003**	-0.000
	Directional Accuracy	0.524	0.524	**0.529**	0.452
Lead	MSE	**1.423e+00**	1.578e+00	2.190e+00	**2.070e+00**
	MAE	**9.180e-01**	9.859e-01	1.158e+00	**1.116e+00**
	R^2	**0.098**	-0.000	-0.058	**-0.000**
	Directional Accuracy	**0.612**	0.456	0.447	**0.534**
Nickel	MSE	**1.939e+00**	2.066e+00	3.160e+00	**2.991e+00**
	MAE	**1.165e+00**	1.234e+00	1.259e+00	**1.249e+00**
	R^2	**0.061**	-0.000	-0.058	**-0.001**
	Directional Accuracy	**0.660**	0.485	**0.559**	0.480
Tin	MSE	**1.380e+00**	1.788e+00	1.170e+01	**5.582e+00**
	MAE	**9.278e-01**	1.059e+00	2.140e+00	**1.810e+00**
	R^2	**0.214**	-0.019	-1.099	**-0.001**
	Directional Accuracy	**0.689**	0.524	0.529	**0.569**
Zinc	MSE	**1.839e+00**	2.044e+00	3.262e+00	**2.999e+00**
	MAE	**1.011e+00**	1.103e+00	**1.387e+00**	1.400e+00
	R^2	**0.100**	-0.000	-0.090	**-0.002**
	Directional Accuracy	**0.592**	0.456	**0.621**	0.553

Table 5.22: Model Comparison with Random Walk (RW) for Validation and Test Results Reverted

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**5.485e+02**	6.286e+02	2.021e+05	**1.137e+03**
	MAE	**1.825e+01**	1.926e+01	9.824e+01	**2.574e+01**
	R^2	**0.854**	0.832	-11.026	**0.932**
Copper	MSE	**5.902e+03**	6.018e+03	**1.468e+04**	1.478e+04
	MAE	5.620e+01	**5.615e+01**	**9.357e+01**	9.379e+01
	R^2	**0.888**	0.885	**0.932**	0.931
Lead	MSE	**6.200e+02**	6.850e+02	9.617e+02	**9.126e+02**
	MAE	**1.920e+01**	2.059e+01	2.442e+01	**2.357e+01**
	R^2	**0.881**	0.869	0.875	**0.881**
Nickel	MSE	**5.296e+04**	5.688e+04	1.044e+05	**1.004e+05**
	MAE	**1.931e+02**	2.049e+02	2.251e+02	**2.231e+02**
	R^2	**0.871**	0.861	0.948	**0.950**
Tin	MSE	**9.271e+04**	1.193e+05	1.243e+06	**5.782e+05**
	MAE	**2.389e+02**	2.721e+02	6.922e+02	**5.822e+02**
	R^2	**0.955**	0.942	0.432	**0.736**
Zinc	MSE	**1.123e+03**	1.250e+03	2.549e+03	**2.317e+03**
	MAE	**2.499e+01**	2.730e+01	**3.885e+01**	3.910e+01
	R^2	**0.846**	0.828	0.839	**0.853**

The results pertaining to the validation set are positive. There is a clear improvement in the performance for all metals. The sole metric where the RW produces better results is the MAE for the copper model. However, when it comes to the test set the only metal that clearly beats the RW is copper. Yet again it is the model with the largest regularization coefficient. There seems to be an overfitting of the noise in the validation set for the other metals. There is also an issue of not being able to handle the large spikes in stock. Below are the predictions for aluminium and tin which produce massive errors corresponding to the spikes in stock. Note that the stock values still affect the states of the reservoir and thus if there are no such spikes during the training nor validation sets the parameters chosen by the GA will reflect that. Especially if they are not present in the validation set.



(a) Aluminium **(b)** Tin

Figure 5.36: Percentage Change Predictions of Aluminium and Tin

The large spikes in the case of aluminium stock and the many consecutive increases of tin stock seemingly create states in the reservoir that the models are unable to make sensible predictions from. So, two possible new approaches become apparent. First, the observation that a higher regularization seems to be key to successful predictions for unseen data. A solution is to pick a regularization coefficient separately from the rest of the parameters, optimizing for an unseen data set. The second option is to simply omit the stock data from the models. If the predictive power in the stock data is low and it disrupts the states of the reservoir, considering the percentage changes in some cases are of a much larger magnitude than for the other data types, then simply not using stock in the models is another solution.

5.4.2 Grid Search For Regularization

The modification to this run is to remove the regularization coefficient from the genome and instead optimize it using a grid search for each network. The grid search consists of simply testing different regularization coefficient on a logarithmic scale starting from 10^{-3} increasing by powers of 10 to 10^{3}. The ridge regression is conducted for the first 90% of the training set and then the MSE, for all three dimensions, on the remaining 10% of the training set determines which regularization is picked. A larger range of values should probably have been picked. Although the operation is quite cheap it still increases the duration of the GA substantially. Following are the result metrics from the run, the complete set of plots and tables can be found in Appendix B.

Table 5.23: Percentage Change Model Comparison with Random Walk (RW) for Validation and Test Results

Material	**Metric**	**Model (Validation)**	**RW (Validation)**	**Model (Test)**	**RW (Test)**
Aluminium	MSE	**1.283e+00**	1.289e+00	1.930e+00	**1.925e+00**
	MAE	**8.724e-01**	8.742e-01	1.053e+00	1.053e+00
	R^2	**0.001**	-0.004	-0.003	**-0.000**
	Directional Accuracy	**0.553**	0.485	0.510	0.510
Copper	MSE	**8.237e-01**	8.391e-01	1.610e+00	**1.585e+00**
	MAE	**6.655e-01**	6.671e-01	9.876e-01	**9.800e-01**
	R^2	**-0.004**	-0.022	-0.016	**-0.000**
	Directional Accuracy	**0.562**	0.524	**0.490**	0.452
Lead	MSE	**1.558e+00**	1.578e+00	**2.065e+00**	2.070e+00
	MAE	**9.757e-01**	9.859e-01	**1.112e+00**	1.116e+00
	R^2	**0.012**	-0.000	**0.002**	-0.000
	Directional Accuracy	**0.544**	0.456	**0.573**	0.534
Nickel	MSE	2.631e+00	**2.066e+00**	3.001e+00	**2.991e+00**
	MAE	1.256e+00	**1.234e+00**	**1.235e+00**	1.249e+00
	R^2	-0.274	**-0.000**	-0.005	**-0.001**
	Directional Accuracy	**0.641**	0.485	**0.578**	0.480
Tin	MSE	**1.660e+00**	1.788e+00	5.582e+00	5.582e+00
	MAE	**1.011e+00**	1.059e+00	1.823e+00	**1.810e+00**
	R^2	**0.054**	-0.019	-0.001	-0.001
	Directional Accuracy	**0.583**	0.524	0.510	**0.569**
Zinc	MSE	**1.957e+00**	2.044e+00	**2.995e+00**	2.999e+00
	MAE	**1.077e+00**	1.103e+00	**1.388e+00**	1.400e+00
	R^2	**0.042**	-0.000	**-0.001**	-0.002
	Directional Accuracy	**0.553**	0.456	**0.524**	0.553

Table 5.24: Reverted Values Model Comparison with Random Walk (RW) for Validation and Test Results

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**6.257e+02**	6.286e+02	1.141e+03	**1.137e+03**
	MAE	**1.922e+01**	1.926e+01	2.574e+01	2.574e+01
	R^2	**0.833**	0.832	0.932	0.932
Copper	MSE	**5.910e+03**	6.018e+03	1.509e+04	**1.478e+04**
	MAE	**5.603e+01**	5.615e+01	9.463e+01	**9.379e+01**
	R^2	**0.887**	0.885	0.930	**0.931**
Lead	MSE	**6.759e+02**	6.850e+02	**9.104e+02**	9.126e+02
	MAE	**2.037e+01**	2.059e+01	**2.350e+01**	2.357e+01
	R^2	**0.871**	0.869	**0.882**	0.881
Nickel	MSE	7.169e+04	**5.688e+04**	**9.961e+04**	1.004e+05
	MAE	2.080e+02	**2.049e+02**	**2.200e+02**	2.231e+02
	R^2	**0.825**	0.861	0.950	0.950
Tin	MSE	**1.115e+05**	1.193e+05	5.795e+05	**5.782e+05**
	MAE	**2.605e+02**	2.721e+02	5.867e+02	**5.822e+02**
	R^2	**0.946**	0.942	0.735	**0.736**
Zinc	MSE	**1.198e+03**	1.250e+03	2.322e+03	**2.317e+03**
	MAE	**2.665e+01**	2.730e+01	**3.879e+01**	3.910e+01
	R^2	**0.835**	0.828	0.853	0.853

Having previously established that the GA is capable of beating the RW on the validation set, emphasis is placed on the performance on the test set. Lead and zinc manage to outperform the RW in every metric and the model for aluminium produce the strongest results seen yet. This is a strong case for optimizing the regularization coefficient independently on unseen data. However, the models for copper, nickel and tin perform worse than previously seen. Copper and Nickel are the only models that do not use the highest regularization coefficient of 10^3. Instead, they settle on 10^0 for Copper and 10^{-2} for Nickel. This means that a grid search for a regularization coefficient is not the only remedy for performance on the test set as seen in the case of nickel.
With the results still not entirely satisfactory across all metals it is tempting to conclude that not a single model setup will be able to create superior results to the random walk on the test set, or simply that the market is efficient and that accurate next day predictions cannot be made using historical data. There is still one proposed option to go through namely simply removing the stock data from the predictions, as mentioned previously.

5.4.3 Removing the Stock Data

Having uncorrelated data in the model can be disruptive and lead to worse results. Besides dubious predictive power there is the concern that some stock values disrupt the reservoir states making predictions wildly incorrect. There is a case for using other normalization techniques such as log returns instead of percentage changes, however to keep the results comparable at least in the non-reverted sense, removing the stock data is a reasonable course of action. Following are the result metrics obtained and included in Appendix C are the rest of the related plots and tables.

Table 5.25: Model Comparison with Random Walk (RW) for Validation and Test Results (Percentage Changes)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**1.247e+00**	1.289e+00	1.943e+00	**1.925e+00**
	MAE	**8.613e-01**	8.742e-01	1.057e+00	**1.053e+00**
	R^2	**0.029**	-0.004	-0.009	**-0.000**
	Directional Accuracy	**0.602**	0.485	**0.529**	0.510
Copper	MSE	**8.146e-01**	8.391e-01	1.599e+00	**1.585e+00**
	MAE	**6.660e-01**	6.671e-01	9.817e-01	**9.800e-01**
	R^2	**0.008**	-0.022	-0.009	**-0.000**
	Directional Accuracy	**0.562**	0.524	**0.529**	0.452
Lead	MSE	**1.426e+00**	1.578e+00	2.141e+00	**2.070e+00**
	MAE	**9.361e-01**	9.859e-01	1.147e+00	**1.116e+00**
	R^2	**0.096**	-0.000	-0.035	**-0.000**
	Directional Accuracy	**0.641**	0.456	0.427	**0.534**
Nickel	MSE	**1.881e+00**	2.066e+00	3.072e+00	**2.991e+00**
	MAE	**1.158e+00**	1.234e+00	1.275e+00	**1.249e+00**
	R^2	**0.090**	-0.000	-0.028	**-0.001**
	Directional Accuracy	**0.631**	0.485	0.480	0.480
Tin	MSE	**1.315e+00**	1.788e+00	3.237e+01	**5.582e+00**
	MAE	**9.049e-01**	1.059e+00	2.969e+00	**1.810e+00**
	R^2	**0.250**	-0.019	-4.806	**-0.001**
	Directional Accuracy	**0.689**	0.524	0.451	**0.569**
Zinc	MSE	**1.970e+00**	2.044e+00	3.063e+00	**2.999e+00**
	MAE	**1.083e+00**	1.103e+00	1.400e+00	1.400e+00
	R^2	**0.036**	-0.000	-0.024	**-0.002**
	Directional Accuracy	**0.563**	0.456	0.505	**0.553**

Removing the stock produces the lowest MSE on the validation set for copper, nickel and tin. This suggests there might not be a lot of explanatory power in the stock data, at least for the three. However, the parameters found underperform on the test set across all metals. The models never outperform the random walk on unseen data, which is the case at least in some capacity for the other setups. While the issues pertaining to major spikes in stock are not present the issue of overfitting the validation set are still there. A mechanism for early stopping, or just fewer generations of the GA to avoid this should probably be considered in future.

5.5 Summary of Results

During the runs of the four setups the mean runtime for the GA for a single metal was 3 hours 32 minutes with a standard deviation of 1 hour 36 minutes. The shortest total run was about 10 hours and the longest close to 24 hours.

The best performing models, in terms of MSE_{cash} on the test set, produce the following results.

Table 5.26: Best Performing Models in Terms of Test MSE

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)	Setup
Aluminium	MSE	**1.283e+00**	1.289e+00	1.930e+00	**1.925e+00**	Grid Search
	MAE	**8.724e-01**	8.742e-01	1.053e+00	1.053e+00	
	R^2	**0.001**	-0.004	-0.003	**-0.000**	
	Directional Accuracy	**0.553**	0.485	0.510	0.510	
Copper	MSE	**8.240e-01**	8.391e-01	**1.580e+00**	1.585e+00	MSE_{cash}
	MAE	6.680e-01	**6.671e-01**	**9.787e-01**	9.800e-01	
	R^2	**-0.004**	-0.022	**0.003**	-0.000	
	Directional Accuracy	0.524	0.524	**0.529**	0.452	
Lead	MSE	**1.558e+00**	1.578e+00	**2.065e+00**	2.070e+00	Grid Search
	MAE	**9.757e-01**	9.859e-01	**1.112e+00**	1.116e+00	
	R^2	**0.012**	-0.000	**0.002**	-0.000	
	Directional Accuracy	**0.544**	0.456	**0.573**	0.534	
Nickel	MSE	**2.053e+00**	2.066e+00	**2.965e+00**	2.991e+00	MSE_{all}
	MAE	**1.220e+00**	1.234e+00	**1.244e+00**	1.249e+00	
	R^2	**0.007**	-0.000	**0.007**	-0.001	
	Directional Accuracy	**0.592**	0.485	**0.569**	0.480	
Tin	MSE	**1.619e+00**	1.788e+00	**5.527e+00**	5.582e+00	MSE_{all}
	MAE	**9.984e-01**	1.059e+00	1.816e+00	**1.810e+00**	
	R^2	**0.077**	-0.019	**0.008**	-0.001	
	Directional Accuracy	**0.680**	0.524	0.500	**0.569**	
Zinc	MSE	**1.957e+00**	2.044e+00	**2.995e+00**	2.999e+00	Grid Search
	MAE	**1.077e+00**	1.103e+00	**1.388e+00**	1.400e+00	
	R^2	**0.042**	-0.000	**-0.001**	-0.002	
	Directional Accuracy	**0.553**	0.456	0.524	**0.553**	

Seemingly there are models that can outperform the random walk on unseen data. The difference in performance is not large and requires different setups. The predictions the successful models make are conservative and similar to what a random walk produces. Thus it likely that the daily changes in cash for the metals are highly random and the signal from the historical data used is small. This is reflected in the negative or very small R^2 values suggesting the models possess little to no more explanatory power than the average change during the period of the test set. Although the average change during a period cannot be known beforehand a moving average in percentage changes could be investigated as a new benchmark.
All the models produce a directional accuracy of 50% or more on the test set. In the cases of nickel and lead the accuracy is larger than what an arbitrary up or down choice ($max(DA_{RW}, 1 - DA_{RW})$) could produce. This is possibly useful in practical application.
The only metal that is not better than a RW on the test set in any metric is aluminium. It equals the RW model in MAE and directional accuracy but produces slightly larger MSE and naturally R^2 too.
There is also a concern when the margins are small, and quite a few different tests have been conducted, that the superior performance of some of these models are because of chance. There is no guarantee that the model would outperform a random walk for a different test set. Discerning how much trust can be put into the model is a worthwhile extension of this research and a natural next step.

6 Discussion

This section discusses the results, insights and proposes a possible new research directions.

6.1 Effectiveness of the GA in Improving ESN Predictive Performance

For benchmark datasets such as Mackey-Glass and Lorenz, the GA significantly improves the predictive performance of ESNs on both training and validation sets. The performance consistently improves within 10 generations, avoiding stagnation during optimization.
For the metal futures datasets, the GA achieves marginal improvements in predictive performance over the initial random selection. Validation performance does not stagnate, although training performance occasionally reverts sometimes recovering towards the end of the optimization run. The underlying issue may not be with the GA but rather the inherent difficulties in predicting the metal futures data.

6.2 Sufficiency of the Genome

A well-defined genome or search space is crucial for effective optimization. For the Mackey-Glass and Lorenz datasets, a significant restriction on regularization values seems to limit performance gains. Allowing smaller regularization coefficients could further improve results by enabling the network to leverage online learning more effectively. Despite this the online learning seemingly makes up for this, reducing the error for the validation and test sets where it is deployed.
Another potential issue is the network topology. The scale-free topology, which requires fewer parameters than a uniformly random network, might make it easier for the GA to optimize. While scale-free networks are consistently preferred in this study, further investigation is necessary to confirm whether they are inherently superior. Naturally there are more options and different setups that can be included and configured that might improve performance.

6.3 Effectiveness of the Enhancements

The lack of stagnation in the benchmarks indicates that the GA avoids premature convergence and consistently finds good solutions. While this suggests that the enhancements are effective, a comparative study with alternative strategies would be necessary to draw definitive conclusions. Many other optimization strategies could be explored, but the current GA implementation performs adequately for the application.

6.4 Computational Costs and Time Requirements

A grid search over the genome used in this setup would be computationally infeasible. Training, validating, and evaluating a single optimized network for the metals takes approximately 5 seconds. With this setup, a grid search over just the input, output, and reservoir sizes with a step size of 10 would require evaluating 417,095 networks, which would take around 579 hours. Including additional parameters would make this approach entirely impractical.
In contrast, the longest GA run in this study took approximately 24 hours for all metals. While this duration is not ideal for experimenting with different setups, it is manageable. In practical applications, parameter optimization would not need to be performed frequently and could be significantly accelerated using more powerful hardware than the MacBook Air (2017) used in this study.

6.5 Robustness of GA-Optimized ESNs Across Datasets

For benchmark datasets, the results are consistent, with good predictive performance across different datasets. However, for the metal futures datasets, the models that perform best on the validation sets often fail to achieve the best performance on the test sets. This suggests potential overfitting of the GA to the validation data, which warrants further investigation.
The discrepancy between results for benchmark datasets and metal futures data highlights the importance of understanding the underlying dynamics of the data. For chaotic systems with discernible patterns, longer GA runs may improve performance. Conversely, for data with minimal signal, early stopping mechanisms could prevent overfitting and reduce computational costs.

6.6 Insights for Trading Strategies

The ESN models successfully optimized for metal futures tend to produce predictions close to zero, resembling a random walk. This suggests that daily price changes are highly random or only weakly dependent on previous changes in cash, three-month, or stock prices.
While the models achieve smaller errors and better directional accuracy than a random walk, these improvements vary across metals and setups. For example, rare and large daily movements are poorly predicted, and the built-in memory of the reservoir states often exacerbates prediction errors following such events.
To inform trading strategies, predictions should be paired with a notion of "normal market behavior" for reliability. During periods of exceptional volatility or erratic

behavior, the predictions are less reliable and should not be heavily relied upon. It is critical to recognize that these predictions reflect statistical patterns in past data rather than an intrinsic understanding of market dynamics.

6.7 Comparison with Random Walk Model

ESNs optimized using a GA can outperform random walk models for next-day metal futures predictions. For copper, lead, nickel, and zinc, the ESN models consistently achieve lower MSE and MAE on test sets. However, results for aluminium and tin are mixed. For aluminium, predictions match the random walk in MAE but are inferior in MSE. For tin, predictions are superior in MSE but inferior in MAE.
These results demonstrate the GA's capability to optimize ESNs for improved predictive performance over a random walk, but this depends on the specific setup and dataset. Expanding the search space, particularly for the regularization coefficient, could yield further improvements, albeit at the cost of increased run times. Including stock data introduces additional complexity but can improve robustness to large spikes in stock values.

6.8 Future Directions

The positive results suggest that reservoir computing warrants further exploration for predicting metal futures prices. Future studies could investigate:

- Incorporating additional and diverse data types.
- Analyzing robustness across varying market conditions.
- Extending prediction horizons to capture trends more effectively (e.g., weekly averages).
- Developing early stopping mechanisms to improve efficiency for datasets with minimal signal.
- Modifying parameter selection to be general for multiple metals.

While the current models show promise for informing hedging strategies, further analysis of their risk profiles is necessary. Marginal improvements in cost prediction can be significant for competitive industries like battery manufacturing, but speculative strategies must be coupled with robust safety standards. Overall, this study demonstrates the potential of ESNs and GAs for predictive applications but highlights the need for cautious and informed implementation in practice.

7

Conclusions

Effectiveness of the GA in Improving ESN's Predictive Performance: The Genetic Algorithm (GA) demonstrates clear effectiveness in improving the Echo State Network (ESN) performance for benchmarking functions. For metal futures, the improvements are consistent but minor, particularly on the validation set. Further adjustments to the methodology are recommended to mitigate overfitting to the validation set and enhance generalization.

Computational Costs and Time Requirements of Using GA to Optimize ESNs: Running the ESN-GA model requires a significant computational effort, with runtimes ranging between 12 and 24 hours. However, once the optimal parameters are identified, the ESN itself is highly efficient, completing initialization, warmup, training, and prediction in under five seconds. Notably, the parameters identified by the GA lack consistency across different metals, suggesting that a generalized parameter set would likely yield suboptimal results for diverse metals. While individual models for specific metals can capitalize on rapid training and prediction times, achieving a universally applicable model is unlikely.

Robustness of the GA-Optimized ESN Model Across Different Metals and Datasets: The current setup of the GA-optimized ESN model exhibits limited robustness. Performance on the validation set does not reliably translate to performance on the test set, indicating a need for refinement to ensure broader applicability and reliability across datasets.

Insights for Trading Strategies from ESN Model Predictions: Conservative predictions tend to yield better results, aligning closely with a random walk model. While some discernible patterns may exist, the overarching conclusion is that most price movements are inherently unpredictable. This underscores the importance of caution when leveraging these predictions for trading strategies. There might be a case for the model predicting directional movements well enough to be useful.

Comparative Performance of ESNs Optimized with GA Versus Random Walk Models: GA-optimized ESNs can outperform random walk models in predicting next-day metal futures prices on unseen data. However, the predictions should be approached with care, as they are often inaccurate and should not be solely relied upon for trading decisions.

Bibliography

[1] E. Alkafaween E. Abunawas A. Hammouri A. Hassanat, K. Almohammadi and V. B. S. Prasath. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(390), 2019.

[2] Apache Software Foundation. Apache commons math 3:3.6.1. `https://commons.apache.org/proper/commons-math/`, 2024. Accessed via Maven Central.

[3] J. Chayes B. Bollobás, C. Borgst and O. Riordan. Directed scale-free graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[4] A.-L. Barabasi. Network science. URL `http://networksciencebook.com`.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(15):509–512, 1999.

[6] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. URL `http://www.jstor.org/stable/1831029`.

[7] C. Blume and W. Jakob. Gleam - an evolutionary algorithm for planning and control based on evolution strategy. In *GECCO Late Breaking Papers*, 2002. URL `https://api.semanticscholar.org/CorpusID:18635630`.

[8] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

[9] R. L. Burden and J. D. Faires. *Numerical Analysis*, chapter 9.3. Cengage Learning, Boston, MA, 9th edition, 2010. ISBN 978-0-538-73351-9.

[10] A. Micheli C. Gallicchio and L. Pedrelli. Design of deep echo state networks. *Neural Networks*, 108:33–47, 2018.

[11] M. M. Vellasco C. H. Valencia and K. Figueiredo. Echo state networks: Novel reservoir selection and hyperparameter optimization model for time series forecasting. *Neurocomputing*, 545(126317), 2023.

[12] D. Cai B. Zhang S. Hong C. Sun, M. Song and H. Li. A systematic review of echo state networks from design to application. *IEEE Transactions on Artificial Intelligence*, 5(1):23–37, 2024. doi: 10.1109/TAI.2022.3225780.

[13] N. N. A. Sjarif C. Zhang and R. Ibrahim. Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022. *Advanced Review*, 2022. URL `mailto:zcheng582dx@gmail.com`. Edited by: Witold Pedrycz, Editor-in-Chief.

[14] M. Cococcioni. Mackey-glass time series generator. `https://www.mathworks.com/matlabcentral/fileexchange/24390-mackey-glass-time-series-generator`, 2024. URL `https://www.mathworks.com/matlabcentral/fileexchange/24390-mackey-glass-time-series-generator`. MATLAB Central File Exchange.

[15] PyRCN Contributors. *PyRCN: A Python Library for Reservoir Computing*, 2024. URL `https://pyrcn.readthedocs.io/en/latest/`. Accessed: 2024-12-10.

[16] ReservoirComputing.jl Contributors. *ReservoirComputing.jl: A Julia Library for Reservoir Computing*, 2024. URL `https://docs.sciml.ai/ReservoirComputing/stable/`. Accessed: 2024-12-10.

[17] D. Frisk and M. Gustaver. Chalmers university of technology: Master thesis template. `https://www.overleaf.com/latex/templates/chalmers-university-of-technology-master-thesis-template-2021/bvnmctrysxzj`, 2021. Updated 2020.

[18] D. Gilbert and contributors. Jfreechart 1.0.17. `https://sourceforge.net/projects/jfreechart/`, 2024. Accessed via Maven Central.

[19] M. Han D. Li and J. Wang. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):787–799, May 2012. doi: 10.1109/TNNLS.2012.2188414.

[20] London Metal Exchange. Trading, 2024. URL `https://www.lme.com/Trading`. Accessed: 15-Oct-2024.

[21] E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970. doi: 10.2307/2325486. URL `https://www.jstor.org/stable/2325486`. Papers and Proceedings of the Twenty-Eighth Annual Meeting of the American Finance Association, New York, N.Y. December, 28-30, 1969.

[22] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 5(1):3–14, 1994.

[23] Python Software Foundation. Python, 2020. Available at: `https://www.python.org`.

[24] The Apache Software Foundation. Maven, 2024. Available at: `https://maven.apache.org`.

[25] A. S. Fukunaga. Restart scheduling for genetic algorithms. In A. E. Eiben,

T. Bäck, M. Schoenauer, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature, Berlin*. Springer, Heidelberg, 1998.

[26] J. B. Héroux R. Nakane N. Kanazawa S. Takeda H. Numata D. Nakano G. Tanaka, T. Yamane and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[27] D. Popovici H. Jaeger, M. Lukosevicius and U. Siewert. Optimization and applications of echo state networks with leaky integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[28] A. Chaudhary H. M. Pandey and D. Mehrotrac. A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, 24:1047–1077, 2014.

[29] A. A. Ferreira J. C. de Albuquerque O. N. Neto M. M. da Silva Lira H. T. V. Gouveia, M. A. Souza and R. R. B. de Aquino. Application of augmented echo state networks and genetic algorithm to improve short-term wind speed forecasting. *Energies*, 16(6):2635, 2023. doi: 10.3390/en16062635. URL `https://doi.org/10.3390/en16062635`.

[30] C. P. Hallwood. On the efficiency of the london metal exchange: Copper prices. *Resources Policy*, 14(3):180–182, 1988. doi: 10.1016/0301-4207(88)90003-7. URL `https://doi.org/10.1016/0301-4207(88)90003-7`.

[31] J. C. Hull. *Options, Futures, and Other Derivatives*, chapter 3. Pearson, Boston, 2022. Maple Financial Group Professor of Derivatives and Risk Management, Joseph L. Rotman School of Management, University of Toronto.

[32] H. Jaeger I. B. Yildiz and S. J. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.

[33] O. Kosuchinas M. Rincon V. Sakenas I. Ilies, H. Jaeger and N. Vaskevicius. Stepping forward through echoes of the past: forecasting with echo state networks, 2007. URL `http://www.neural-forecasting-competition.com/downloads/NN3/methods/27-NN3_Herbert_Jaeger_report.pdf`.

[34] G. Ivo and S. Silva. Experiments on controlling overfitting in genetic programming. In *Local proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*, 2011.

[35] J. Hedley. jsoup 1.17.2. `https://jsoup.org/`, 2024. Accessed via Maven Central.

[36] C. J. Houghton J. McAllister, J. Wade and C. O'Donnell. Topological and simplicial features in reservoir computing networks. *UK Workshop on Computational Intelligence*, 2024.

[37] S. H. Kim J. Woo, H. Kim and K. Han. Echo state property upon noisy driving input. *Complexity*, Article ID 5593925, 2024.

[38] H. Jaeger. The "echo state" approach to analysing and training recurrent neural

networks. *GMD report*, 148 GMD — german national research institute for computer science, 2001.

[39] H. Jaeger. Erratum note for the techreport: the "echo state" approach to analysing and training recurrent neural networks. Technical report, 2010.

[40] H. Jaeger. Foreword. In *Reservoir Computing Theory: Physical Implementations, and Applications*, pages V–XI. Springer, 2021.

[41] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *SCIENCE*, 304:78–80, 2004.

[42] H. Jaeger and H. Haas. Supplementary material for: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. `https://www.science.org/doi/10.1126/science.1091277`, 2004. Accessed: 2023-10-28.

[43] JetBrains. *IntelliJ IDEA: The Java IDE for Professional Developers*, 2024. URL `https://www.jetbrains.com/idea/`. Version 2023.3.7 Ultimate Edition.

[44] JetBrains. Jetbrains ai assistant, 2024. URL `https://www.jetbrains.com`. AI-powered coding assistant, accessed 2024.

[45] J. Kubal K. Knehr and Ahmed S K. Estimated costs of ev batteries 2018-2023, 2023. URL `https://www.anl.gov/sites/www/files/2024-06/EV%20Costs%202023%20for%20GPRA%20reporting%20%282023-09-17%29.pdf`. Accessed: 2024-10-21.

[46] A. Lee. Chinese lithium producers embrace hedging to manage volatility. *Bloomberg*, sep 2024. Accessed on [insert date you accessed the article, if necessary].

[47] London Metal Exchange. Lme official opening stock (in tonnes), 2012. URL `https://web.archive.org/web/20120404214455/http://www.lme.com:80/nickel.asp#expand`. Accessed: 12-Nov-2024.

[48] London Metal Exchange. Fees, 2024. URL `https://www.lme.com/Trading/Access-the-market/Fees`. Accessed: 2024-12-19.

[49] London Metal Exchange. Historical data, 2024. URL `https://www.lme.com/Market-data/Accessing-market-data/Historical-data`. Accessed: 2024-12-19.

[50] London Metal Exchange. Find a member, 2024. URL `https://www.lme.com/Trading/Access-the-market/Find-a-member`. Accessed: YYYY-MM-DD.

[51] London Metal Exchange. Lme official price, 2024. URL `https://www.lme.com/Market-data/LME-reference-prices/LME-Official-Price`. Accessed: 2024-12-19.

[52] London Metal Exchange. London metal exchange rules and regulations: Position management where netting, aggregation or close out is applied to client-related cleared contracts, 2024. URL

https://www.lme.com/-/media/Files/Company/Market-regulation/Rulebook/Previous-releases/Rulebook-as-of-April-2024.pdf. Accessed: 2024-12-19.

[53] London Metal Exchange. 2023 volume report, 2024. URL https://www.lme.com/en/Market-data/Reports-and-data/Volumes/Annual-volumes. Accessed: 10-Oct-2024.

[54] London Metal Exchange. Warehouse network, 2024. URL https://www.lme.com/Sustainability-and-Physical-Markets/Warehousing/Warehouse-network. Accessed: 2024-12-19.

[55] London Metal Exchange. Risk management through hedging, 2024. URL https://www.lme.com. Provides an overview of how futures contracts are widely used to mitigate inventory risks for industrial metals.

[56] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[57] M. Lukoševičius. A practical guide to applying echo state networks. In G.B. Orr G. Montavon and K.R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 659–686. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-35289-8_36. URL https://doi.org/10.1007/978-3-642-35289-8_36.

[58] K. Kopczewska M. Buczyński, M. Chlebus and M. Zajenkowski. Financial time series models—comprehensive review of deep learning approaches and practical recommendations. In *Proceedings of the 9th International Conference on Time Series and Forecasting*, pages 12–14, Gran Canaria, Spain, July 2023. URL mailto:mp.buczynski2@uw.edu.pl. Presented at the 9th International Conference on Time Series and Forecasting.

[59] A. Sebald S. Stepney M. Dale, S. O'Keefe1 and M. A. Trefzer. Reservoir computing quality: connectivity and topology. *Natural Computing*, 2021. doi: https://doi.org/10.1007/s11047-020-09823-1.

[60] S. O'Keefe A. Sebald S. Stepney M. Dale, J. Dewhirst and M. A. Trefzer. The role of structure and complexity on reservoir computing quality. In *Unconventional Computation and Natural Computation*, pages 52–64. Springer Nature Switzerland, 2019. doi: https://doi.org/10.1007/978-3-030-19311-9_6.

[61] Y. Ikeda M. Inubushi, K. Yoshimura and Y. Nagasawa. On the characteristics and structures of dynamical systems suitable for reservoir computing. In Physical Implementation and Applications, editors, *Reservoir Computing: Theory*, pages 98–100. Springer, Nature, 2021.

[62] K. A. Asosega M. Omane-Adjepong and K. Osei-Assibey. Chaos in financial markets: Research insights, measures, and influences. *SSRN Electronic Journal*, April 2024. doi: 10.2139/ssrn.4804045. URL https://doi.org/10.2139/

ssrn.4804045. University of Energy and Natural Resources (UENR) and University of Johannesburg.

[63] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977. doi: 10.1126/science.267326.

[64] S. Malik and S. Wadhwa. Preventing premature convergence in genetic algorithm using dgca and elitist technique. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(6):410–418, 2014.

[65] B. B. Mandelbrot and R. L. Hudson. By the toss of a coin or the flight of an arrow?: Chance in finance. In *The (Mis)Behavior of Markets: A Fractal View of Risk, Ruin, and Reward*, chapter 2. Basic Books, New York, 1997.

[66] M. Mitchell. *An Introduction to Genetic Algorithms.* A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1999.

[67] K. P. Murphy. *Probabilistic Machine Learning.* An introduction, MIT Press, 2022.

[68] T. T. Dinh N. Trouvain, L. Pedrelli and X. Hinaut. Reservoirpy: an efficient and user-friendly library to design echo state networks. In *ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia*, 2020.

[69] NetworkX. Generators for some directed graphs, February 2024. URL https://github.com/networkx/networkx/blob/main/networkx/generators/directed.py.

[70] Northvolt. 2023 sustainability and annual report, 2024. URL https://northvolt.com/sustainability/report2023/. Accessed: 15-Oct-2024.

[71] Northvolt. Recycling lithium-ion batteries: What we recover, 2024. URL https://northvolt.com/recycling/. Accessed: 2024-12-19.

[72] S. Mannor O. Anava, E. Hazan and O. Shamir. Online learning for time series prediction. *arXiv preprint arXiv:1302.6927*, 2013. URL https://doi.org/10.48550/arXiv.1302.6927.

[73] M. U. Gudelek O. B. Sezer and A. M. Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *arXiv preprint arXiv:1911.13288*, November 2019. URL https://arxiv.org/abs/1911.13288. Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey.

[74] OpenAI. ChatGPT, 2024. URL https://openai.com/chatgpt. Large language model, accessed 2024.

[75] D. Ouerghi. Battery raw material price volatility spurs holistic hedging strategies, industry experts say. *Fastmarkets*, jul 2024. Accessed on [insert access date if necessary].

[76] P. Abeles and contributors. Efficient java matrix library (ejml) 0.43.1. `https://ejml.org/`, 2024. Accessed via Maven Central.

[77] S. Pothuganti. Review on over-fitting and under-fitting problems in machine learning and solutions. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 7(9):3692–3695, 2018.

[78] S. Conway, G. Smith, and contributors. Opencsv 5.9. `http://opencsv.sourceforge.net/`, 2024. Accessed via Maven Central.

[79] S. Scardapane and D. Wang. Randomness in neural networks: an overview. *WIREs Data Mining and Knowledge Discovery*, e1200:7, 2017.

[80] F. Sese. Personal communication with frederic sese, senior manager metals and raw materials at northvolt, March 2024. Online meeting.

[81] Inc. The MathWorks. Matlab. `https://www.mathworks.com`, 2024. Version 24.2.0.2754538.

[82] The Matplotlib development team. Lorenz attractor example. `https://matplotlib.org/stable/gallery/mplot3d/lorenz_attractor.html`, 2024. URL `https://matplotlib.org/stable/gallery/mplot3d/lorenz_attractor.html`. Matplotlib Example Gallery.

[83] F. Cauteruccio W. Chen, W. Hussain and X. Zhang. Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models. *Computer Modeling in Engineering & Sciences*, 2023. doi: 10.32604/cmes.2023.031388. URL `mailto:chenweisi@xmut.edu.cn;mailto:walayat.hussain@acu.edu.au`. Received: 09 June 2023; Accepted: 18 September 2023; Published: 30 December 2023.

[84] J. Xiong W.-J. Wang, Y. Tang and Y.-C. Zhang. Stock market index prediction based on reservoir computing models. *Expert Systems With Applications*, 178 (115022), 2021.

[85] M. Wahde. *Biologically Inspired Optimization Methods*. An Introduction, WIT Press, 2008.

[86] Westmetall. Market data. URL `https://www.westmetall.com/en/markdaten.php`. Accessed: 2024-12-18.

[87] Wikipedia contributors. Power iteration. `https://en.wikipedia.org/wiki/Power_iteration`, 2024. Accessed: 2025-01-24.

[88] F. Wyffels and B. Schrauwen. A comparative study of reservoir computing strategies for monthly time series prediction. *Neurocomputing*, 73(12):1958–1964, 2010. doi: 10.1016/j.neucom.2009.11.017. URL `https://doi.org/10.1016/j.neucom.2009.11.017`.

[89] Z. Yang X. Lin and Y. Song. Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 145:113110, 2020. doi: 10.1016/j.eswa.2019.113110. URL `https://www.sciencedirect.com/science/article/pii/S0957417419308410`.

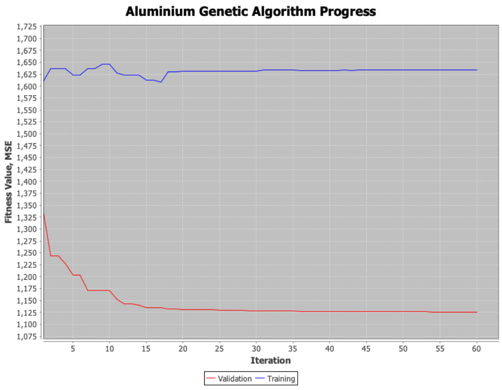
[90] Y. Wang Y. Zhang and G. Luo. A new optimization algorithm for non-stationary time series prediction based on recurrent neural networks. *Future Generation Computer Systems*, 102:738–745, 2020.

[91] A. G. Ţiţan. The efficient market hypothesis: Review of specialized literature and empirical research. *Procedia Economics and Finance*, 32:442–449, 2015. URL `https://doi.org/10.1016/S2212-5671(15)01416-1`.

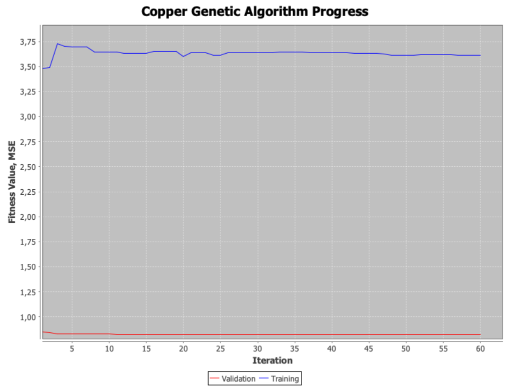
A

Appendix A

Modified Fitness Function GA run

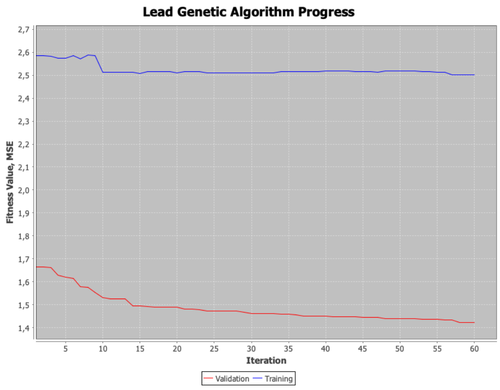


(a) Aluminium GA Progress

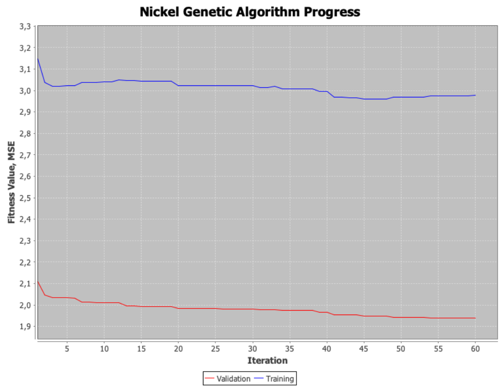


(b) Copper GA Progress

Figure A.1: Genetic Algorithm Progress for Aluminium and Copper.

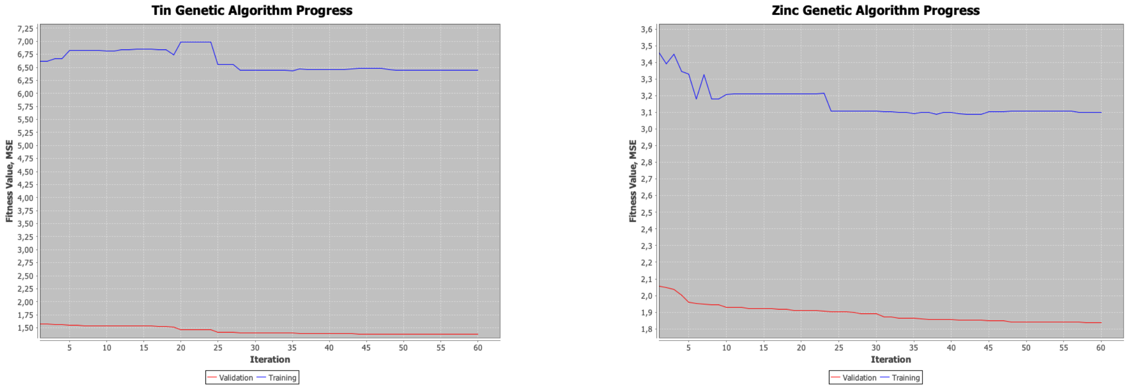


(a) Lead GA Progress



(b) Nickel GA Progress

Figure A.2: Genetic Algorithm Progress for Lead and Nickel.



(a) Tin GA Progress **(b)** Zinc GA Progress

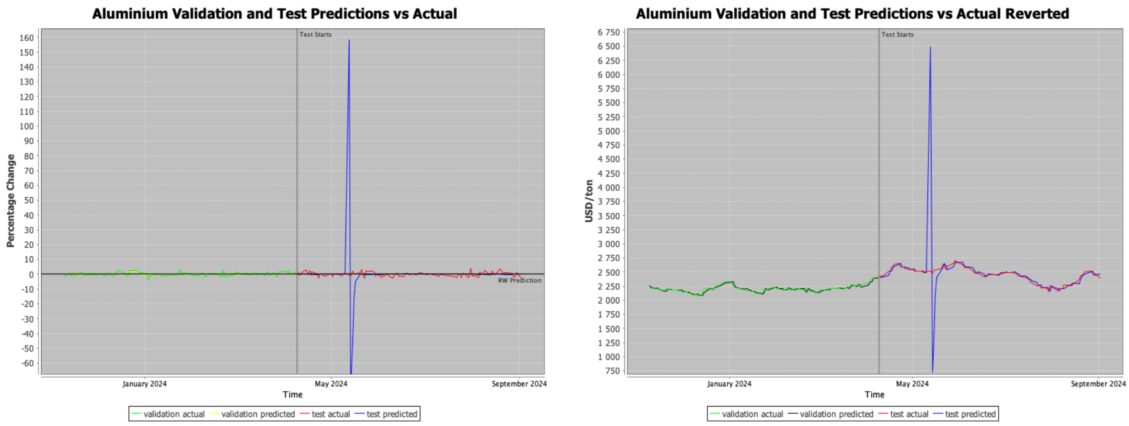
Figure A.3: Genetic Algorithm Progress for Tin and Zinc.

Parameters

Parameter	Aluminium	Copper	Lead	Nickel	Tin	Zinc
Topology	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free
Connectivity	0.131	0.128	0.114	0.071	0.074	0.143
Spectral Radius	0.807	0.894	0.592	0.574	0.654	0.976
Regularization	1.004E-5	8.872E-1	7.475E-5	1.090E-2	5.696E-4	8.002E-2
Learning Rate	1.511E-5	7.774E-5	2.691E-6	5.858E-6	8.501E-6	5.677E-7
Leakage Rate	0.772	0.260	0.731	0.048	0.743	0.526
Reservoir Size	703	529	718	806	1195	840
In Length	111	195	189	225	351	779
Out Length	111	195	189	225	351	347
In Bias Scale	99.259	87.634	22.662	0.729	15.583	91.264
In Factor Scale	[1.708, 2.158, 0.739]	[0.384, 1.067, 2.012]	[2.554, 1.070, 2.939]	[1.189, 1.504, 2.405]	[1.927, 0.479, 2.163]	[2.837, 1.912, 2.340]

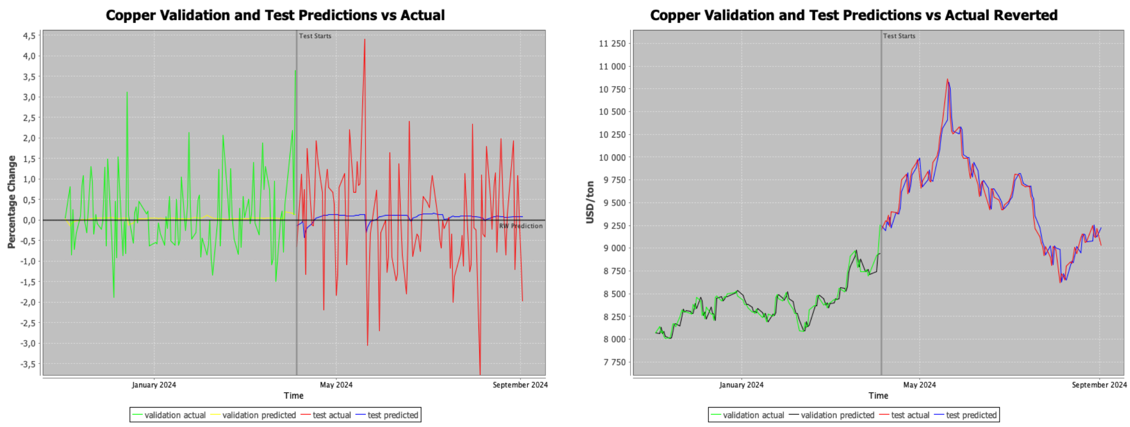
Table A.1: GA Optimized Parameters for the Metals.

Prediction Plots



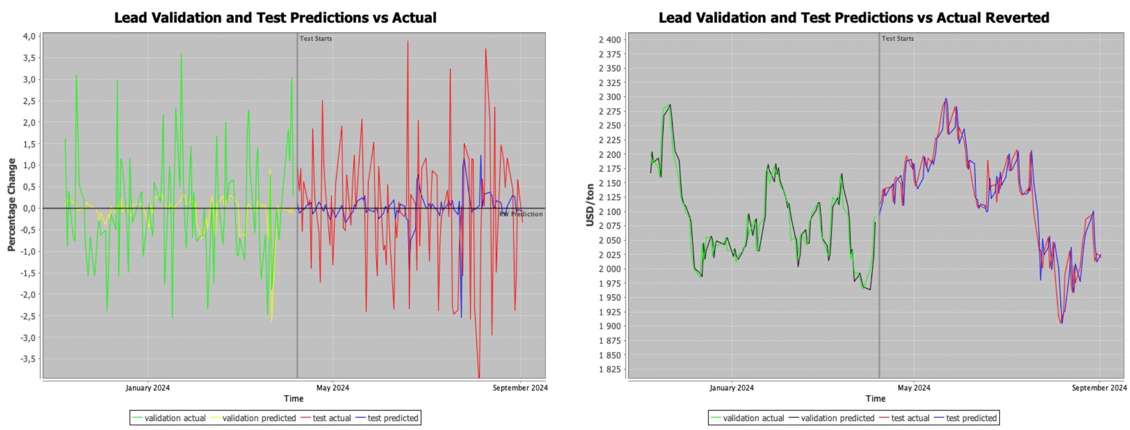
(a) Percentage Change **(b)** Reverted

Figure A.4: Comparison of Percentage Change and Reverted Predictions for Aluminium.



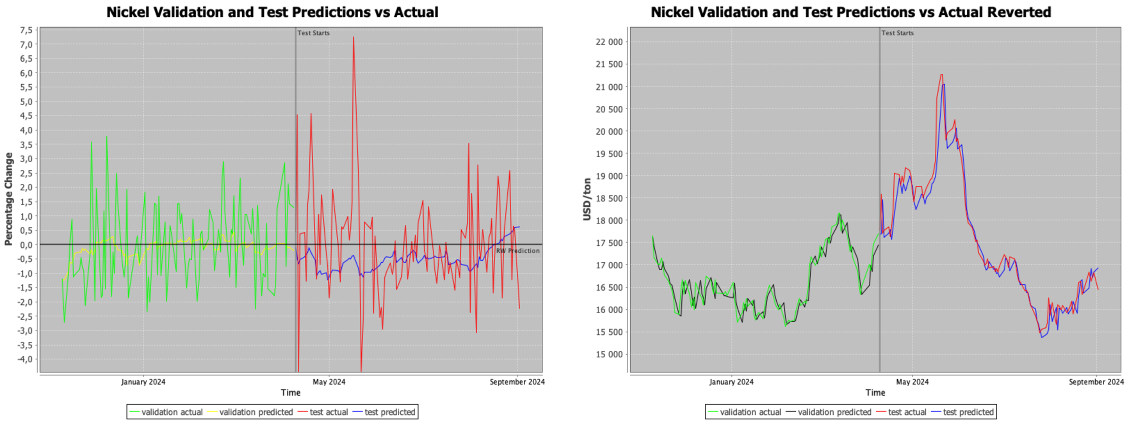
(a) Percentage Change **(b)** Reverted

Figure A.5: Comparison of Percentage Change and Reverted Predictions for Copper.



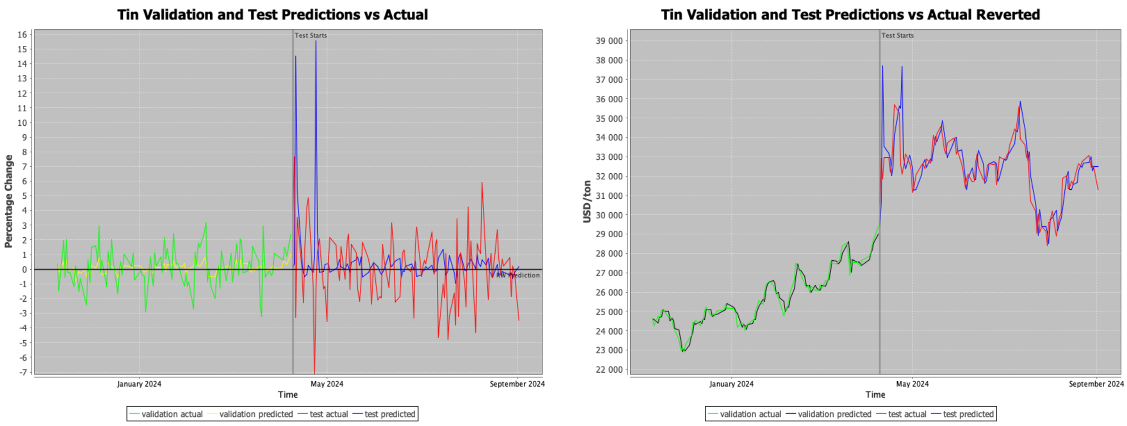
(a) Percentage Change **(b)** Reverted

Figure A.6: Comparison of Percentage Change and Reverted Predictions for Lead.



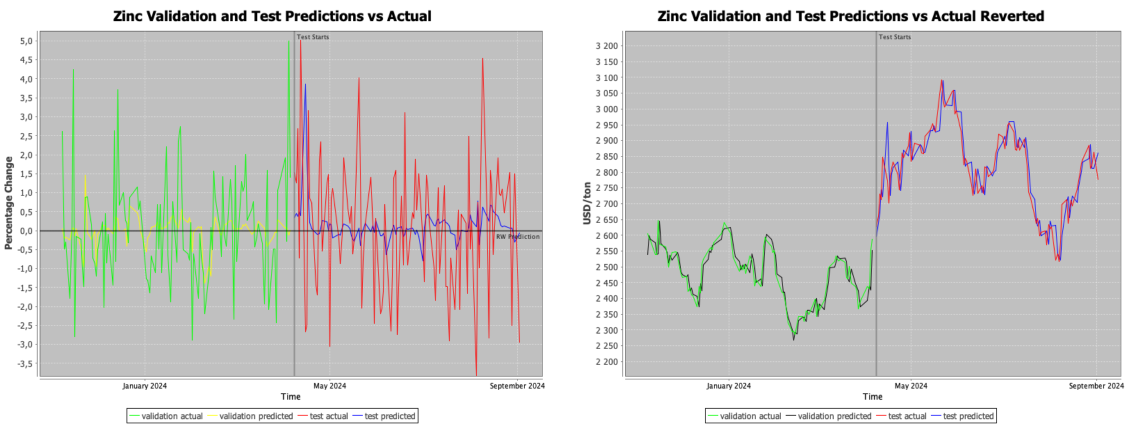
(a) Percentage Change **(b)** Reverted

Figure A.7: Comparison of Percentage Change and Reverted Predictions for Nickel.



(a) Percentage Change **(b)** Reverted

Figure A.8: Comparison of Percentage Change and Reverted Predictions for Tin.



(a) Percentage Change **(b)** Reverted

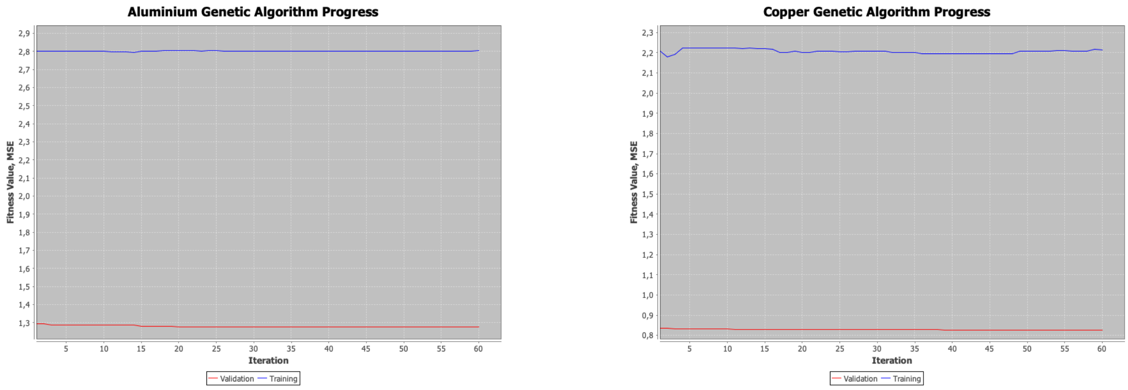
Figure A.9: Comparison of Percentage Change and Reverted Predictions for Zinc.

B

Appendix B

Grid Search Regularization GA Run:

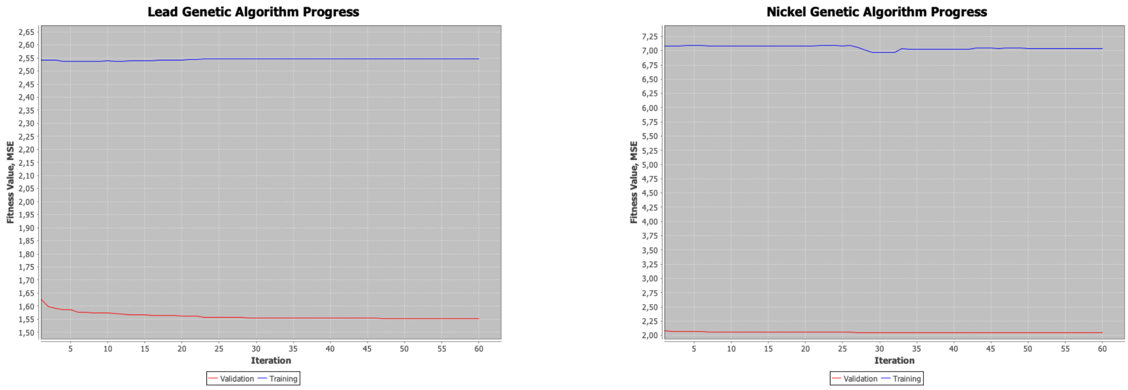
GA plots



(a) Aluminium GA Progress

(b) Copper GA Progress

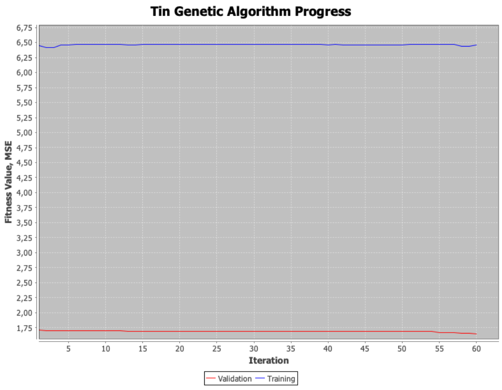
Figure B.1: Genetic Algorithm Progress for Aluminium and Copper.



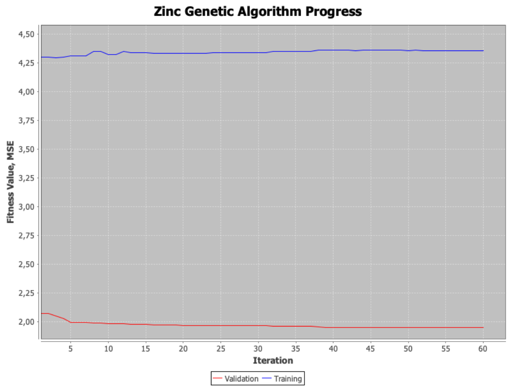
(a) Lead GA Progress

(b) Nickel GA Progress

Figure B.2: Genetic Algorithm Progress for Lead and Nickel.



(a) Tin GA Progress



(b) Zinc GA Progress

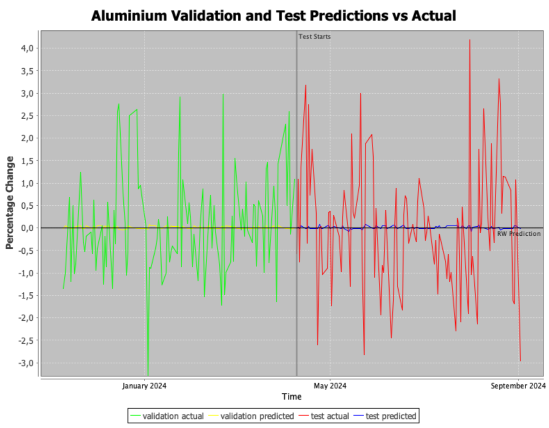
Figure B.3: Genetic Algorithm Progress for Tin and Zinc.

Parameters

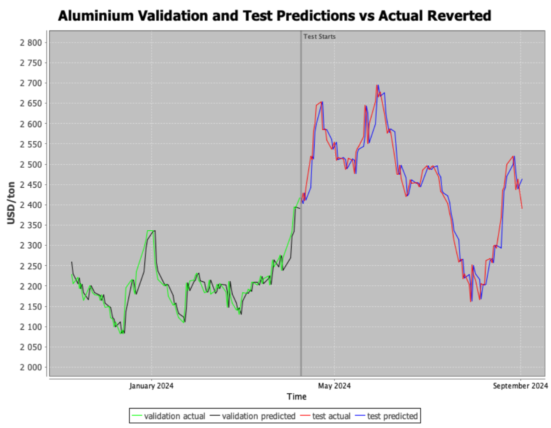
Parameter	Aluminium	Copper	Lead	Nickel	Tin	Zinc
Topology	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free
Connectivity	0.031	0.011	0.089	0.097	0.086	0.075
Spectral Radius	0.549	0.999	0.994	0.994	0.983	0.999
Regularization	1000.000	1.000	1000.000	0.010	1000.000	1000.000
Learning Rate	3.374E-6	5.750E-4	5.708E-7	1.075E-8	8.388E-7	1.238E-6
Leakage Rate	0.874	0.913	0.994	0.996	0.995	0.999
Reservoir Size	1138	847	638	1140	680	704
In Length	133	146	109	252	393	224
Out Length	133	83	66	252	287	224
In Bias Scale	51.940	30.355	20.750	99.828	4.062	2.723
In Factor Scale	[2.665, 0.890, 2.961]	[2.999, 0.606, 1.654]	[0.060, 2.874, 1.332]	[1.244, 2.991, 0.008]	[2.743, 0.960, 2.618]	[1.867, 0.952, 1.628]

Table B.1: GA Optimized Parameters for the Metals

Prediction Plots

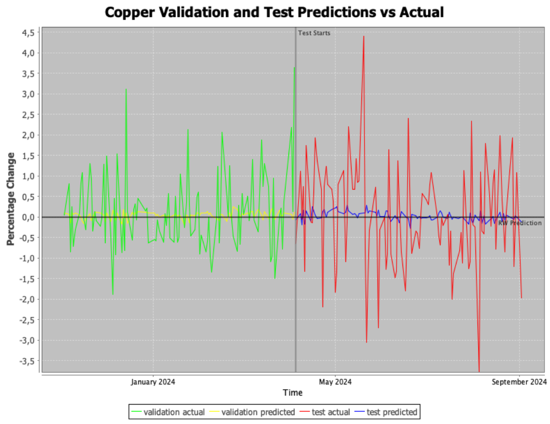


(a) Percentage Change

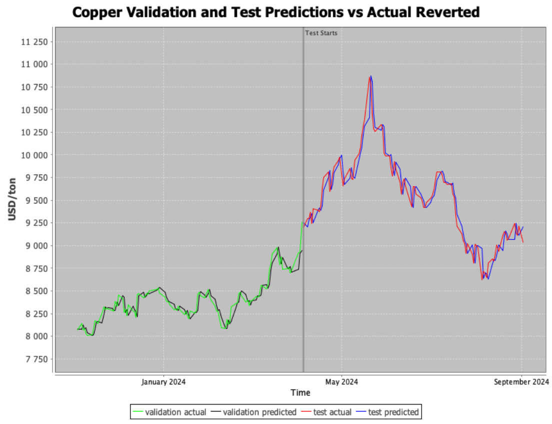


(b) Reverted

Figure B.4: Comparison of Percentage Change and Reverted Predictions for Aluminium.

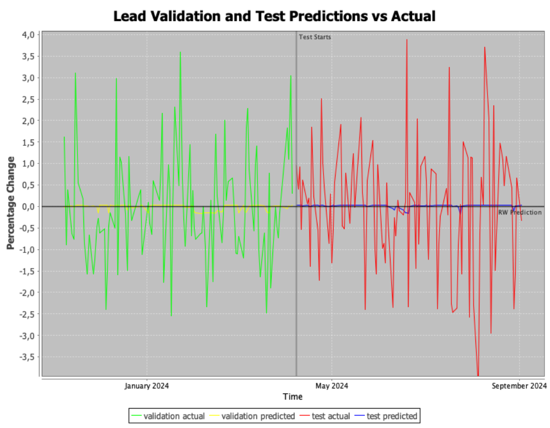


(a) Percentage Change

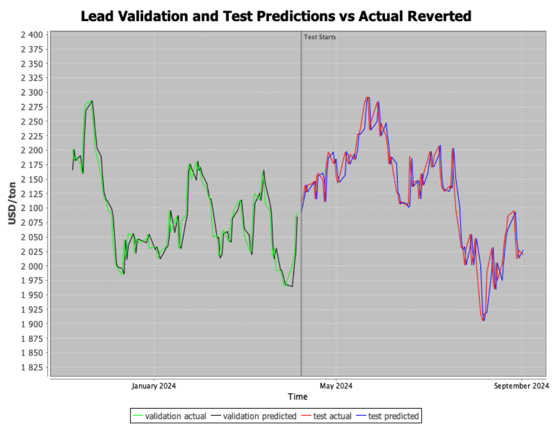


(b) Reverted

Figure B.5: Comparison of Percentage Change and Reverted Predictions for Copper.

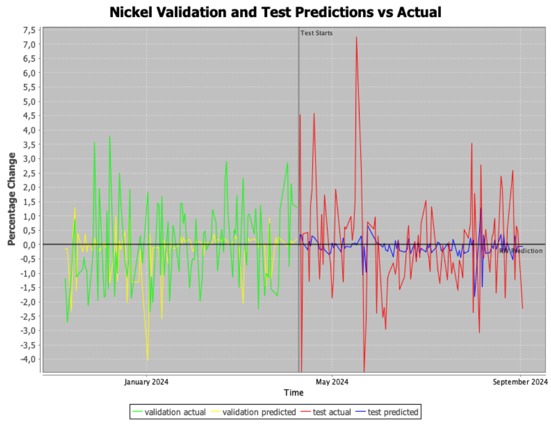


(a) Percentage Change

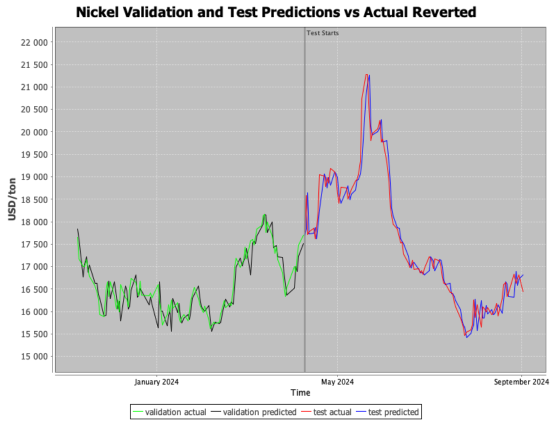


(b) Reverted

Figure B.6: Comparison of Percentage Change and Reverted Predictions for Lead.

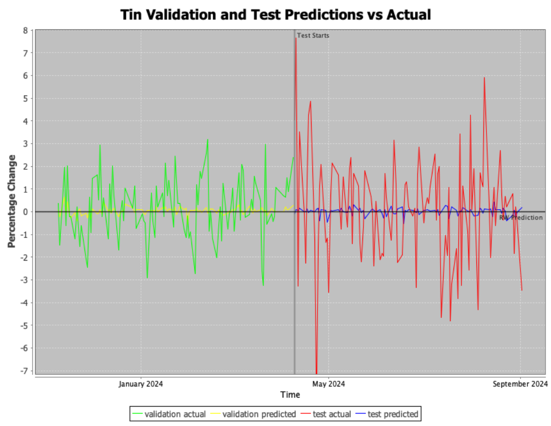


(a) Percentage Change

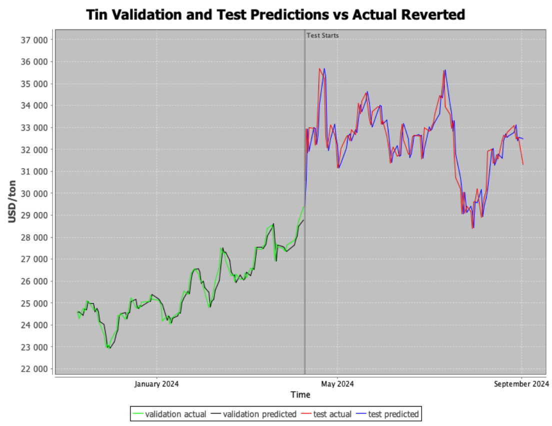


(b) Reverted

Figure B.7: Comparison of Percentage Change and Reverted Predictions for Nickel.

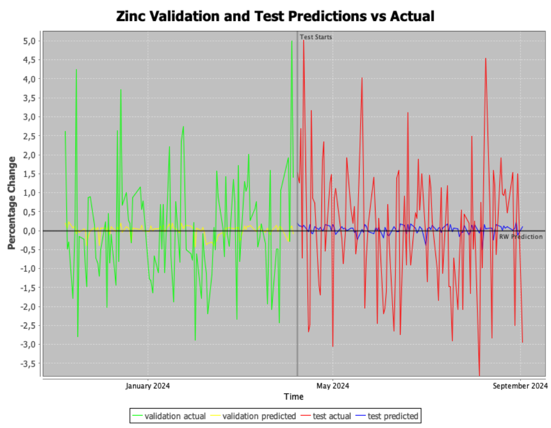


(a) Percentage Change

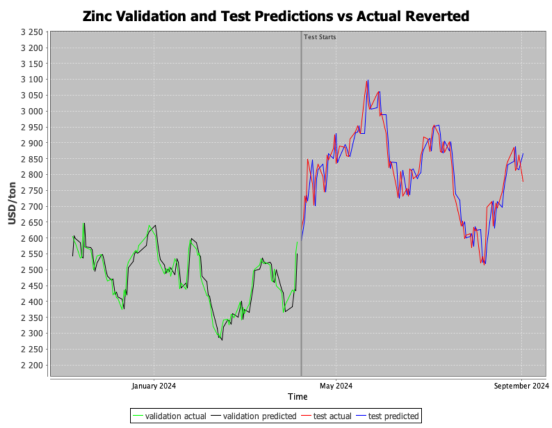


(b) Reverted

Figure B.8: Comparison of Percentage Change and Reverted Predictions for Tin.



(a) Percentage Change



(b) Reverted

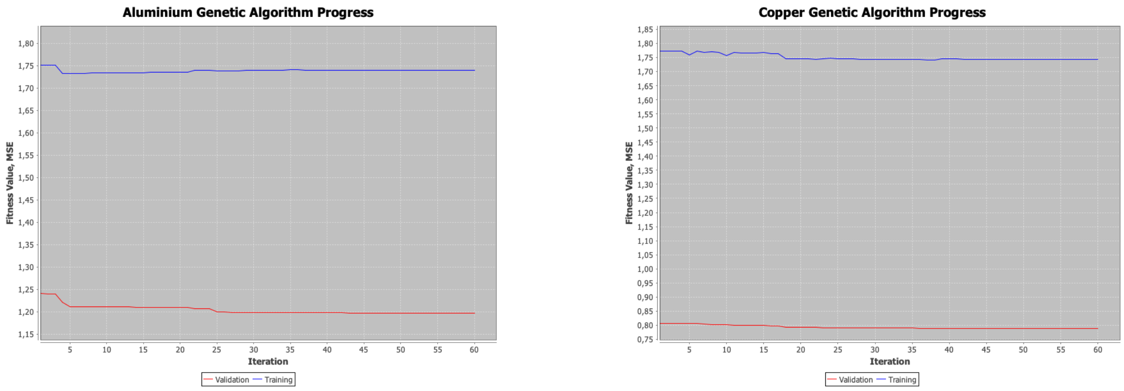
Figure B.9: Comparison of Percentage Change and Reverted Predictions for Zinc.

C

Appendix C

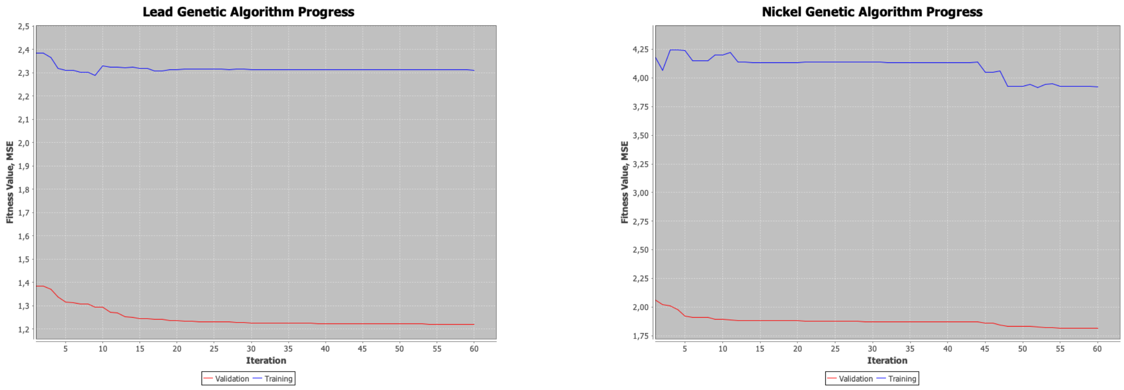
Removing Stock GA Run (MSE of cash and three month):

GA plots



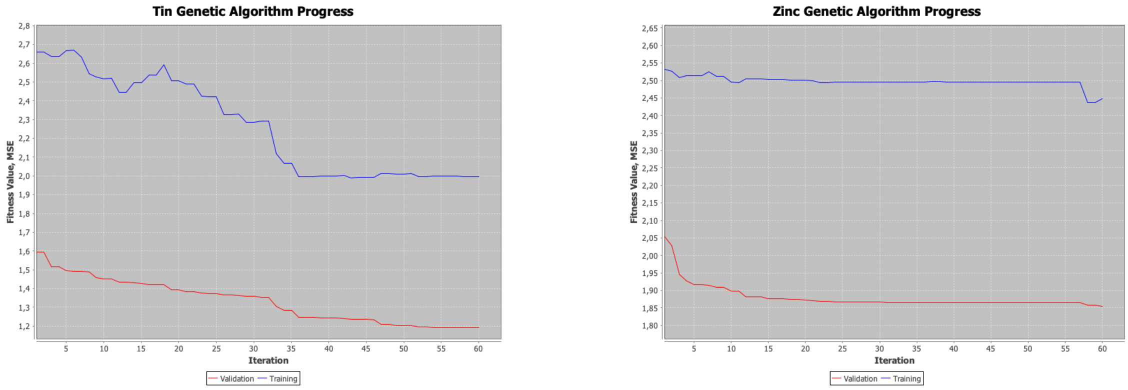
(a) Aluminium GA Progress **(b)** Copper GA Progress

Figure C.1: Genetic Algorithm Progress for Aluminium and Copper.



(a) Lead GA Progress **(b)** Nickel GA Progress

Figure C.2: Genetic Algorithm Progress for Lead and Nickel.



(a) Tin GA Progress **(b)** Zinc GA Progress

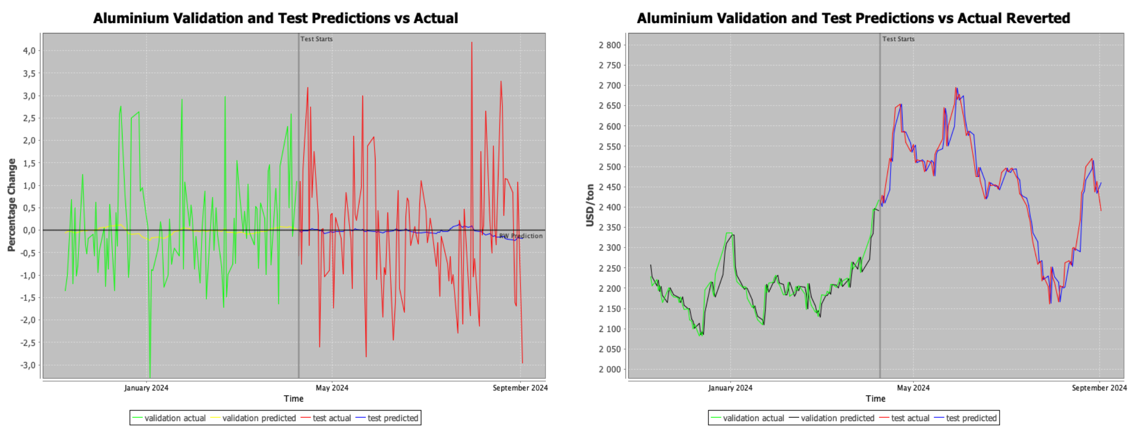
Figure C.3: Genetic Algorithm Progress for Tin and Zinc.

Parameter table

Parameter	Aluminium	Copper	Lead	Nickel	Tin	Zinc
Topology	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free	Scale Free
Connectivity	0.139	0.072	0.053	0.081	0.033	0.086
Spectral Radius	0.506	0.875	0.993	0.619	0.607	0.932
Regularization	5.065E-3	1.132E+1	6.408E-5	1.002E-5	1.094E-5	8.471E-4
Learning Rate	1.433E-6	1.558E-5	1.383E-5	6.908E-6	2.323E-5	5.270E-7
Leakage Rate	0.132	0.867	0.413	0.065	0.828	0.792
Reservoir Size	1161	1067	699	949	994	1181
In Length	135	1056	142	482	882	316
Out Length	135	505	110	464	751	316
In Bias Scale	10.010	11.096	8.129	0.096	20.670	71.583
In Factor Scale	[0.127, 2.839]	[2.519, 5.067E-4]	[2.917, 1.548]	[1.275, 0.104]	[2.872, 1.636]	[2.891, 2.798]

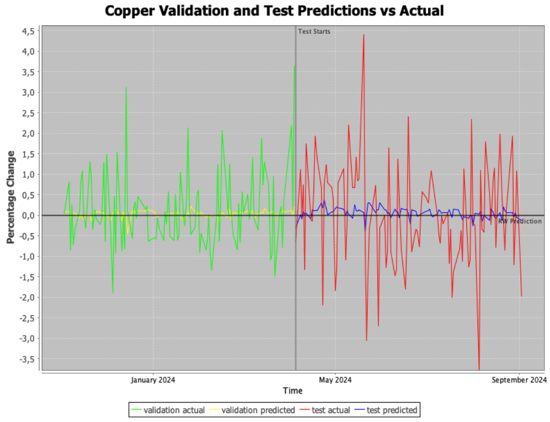
Table C.1: GA Optimized Parameters for the Metals (Regularization in Scientific Notation)

Prediction Plots

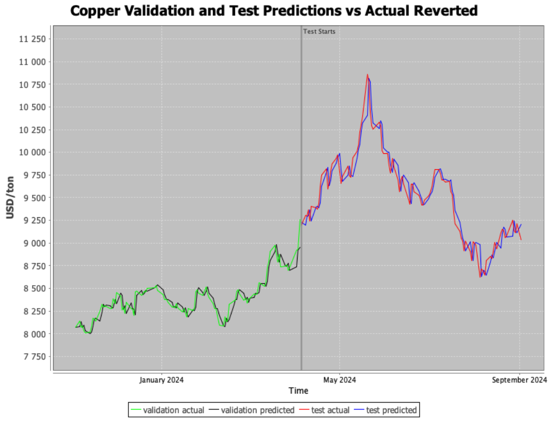


(a) Percentage Change **(b)** Reverted

Figure C.4: Comparison of Percentage Change and Reverted Predictions for Aluminium.

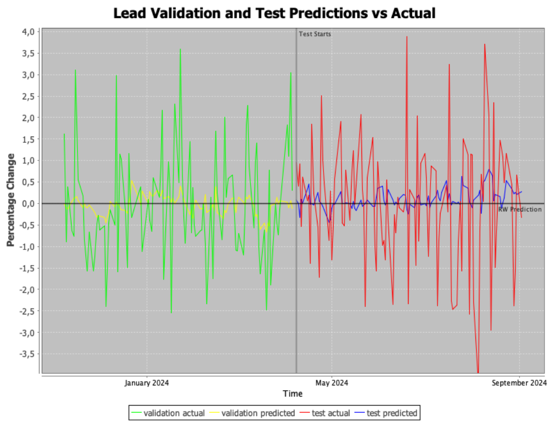


(a) Percentage Change

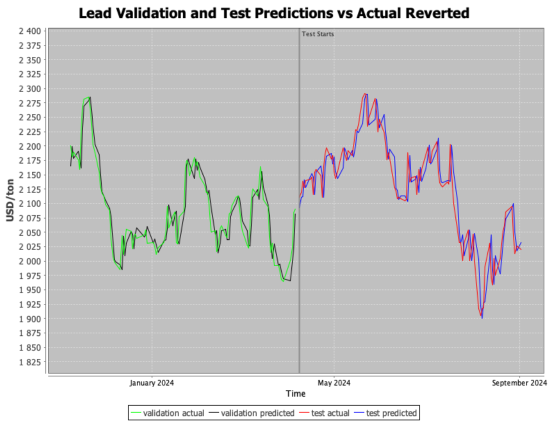


(b) Reverted

Figure C.5: Comparison of Percentage Change and Reverted Predictions for Copper.

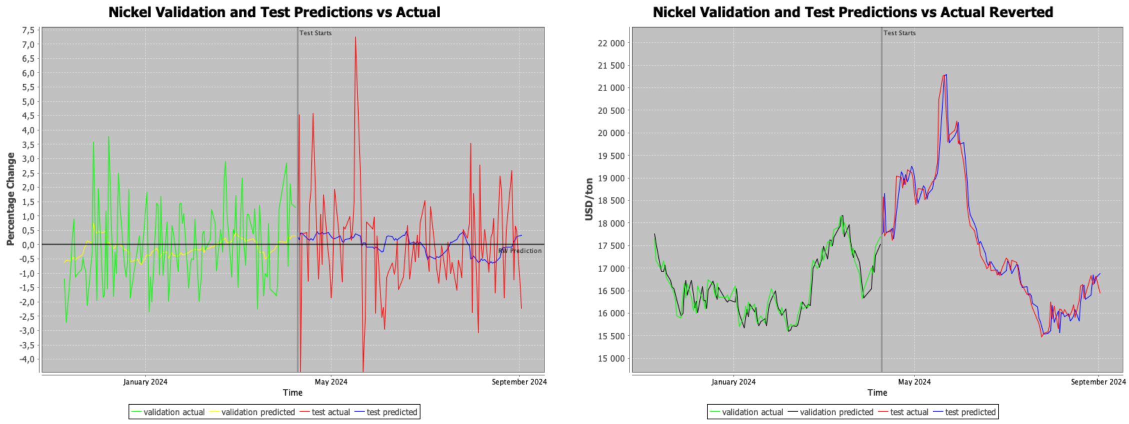


(a) Percentage Change



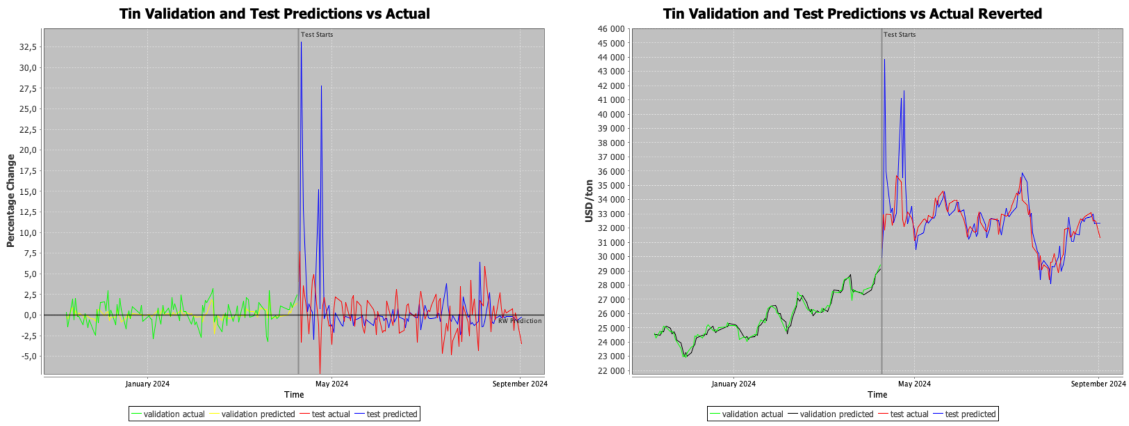
(b) Reverted

Figure C.6: Comparison of Percentage Change and Reverted Predictions for Lead.



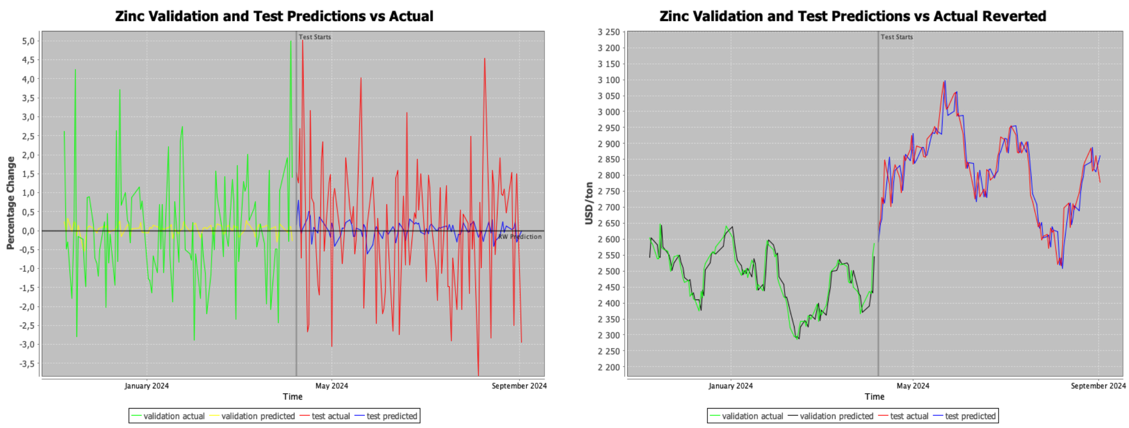
(a) Percentage Change **(b)** Reverted

Figure C.7: Comparison of Percentage Change and Reverted Predictions for Nickel.



(a) Percentage Change **(b)** Reverted

Figure C.8: Comparison of Percentage Change and Reverted Predictions for Tin.



(a) Percentage Change **(b)** Reverted

Figure C.9: Comparison of Percentage Change and Reverted Predictions for Zinc.

Metric Tables:

Table C.2: Model Comparison with Random Walk (RW) for Validation and Test Results (Percentage Changes)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**1.247e+00**	1.289e+00	1.943e+00	**1.925e+00**
	MAE	**8.613e-01**	8.742e-01	1.057e+00	**1.053e+00**
	R^2	**0.029**	-0.004	-0.009	**-0.000**
	Directional Accuracy	**0.602**	0.485	**0.529**	0.510
Copper	MSE	**8.146e-01**	8.391e-01	1.599e+00	**1.585e+00**
	MAE	**6.660e-01**	6.671e-01	9.817e-01	**9.800e-01**
	R^2	**0.008**	-0.022	-0.009	**-0.000**
	Directional Accuracy	**0.562**	0.524	**0.529**	0.452
Lead	MSE	**1.426e+00**	1.578e+00	2.141e+00	**2.070e+00**
	MAE	**9.361e-01**	9.859e-01	1.147e+00	**1.116e+00**
	R^2	**0.096**	-0.000	-0.035	**-0.000**
	Directional Accuracy	**0.641**	0.456	0.427	**0.534**
Nickel	MSE	**1.881e+00**	2.066e+00	3.072e+00	**2.991e+00**
	MAE	**1.158e+00**	1.234e+00	1.275e+00	**1.249e+00**
	R^2	**0.090**	-0.000	-0.028	**-0.001**
	Directional Accuracy	**0.631**	0.485	0.480	0.480
Tin	MSE	**1.315e+00**	1.788e+00	3.237e+01	**5.582e+00**
	MAE	**9.049e-01**	1.059e+00	2.969e+00	**1.810e+00**
	R^2	**0.250**	-0.019	-4.806	**-0.001**
	Directional Accuracy	**0.689**	0.524	0.451	**0.569**
Zinc	MSE	**1.970e+00**	2.044e+00	3.063e+00	**2.999e+00**
	MAE	**1.083e+00**	1.103e+00	1.400e+00	1.400e+00
	R^2	**0.036**	-0.000	-0.024	**-0.002**
	Directional Accuracy	**0.563**	0.456	0.505	**0.553**

Table C.3: Model Comparison with Random Walk (RW) for Validation and Test Results (Reverted)

Material	Metric	Model (Validation)	RW (Validation)	Model (Test)	RW (Test)
Aluminium	MSE	**6.075e+02**	6.286e+02	1.144e+03	**1.137e+03**
	MAE	**1.897e+01**	1.926e+01	2.581e+01	**2.574e+01**
	R^2	**0.838**	0.832	0.932	**0.932**
Copper	MSE	**5.832e+03**	6.018e+03	1.494e+04	**1.478e+04**
	MAE	**5.605e+01**	5.615e+01	9.398e+01	**9.379e+01**
	R^2	**0.889**	0.885	0.930	**0.931**
Lead	MSE	**6.177e+02**	6.850e+02	9.451e+02	**9.126e+02**
	MAE	**1.954e+01**	2.059e+01	2.423e+01	**2.357e+01**
	R^2	**0.882**	0.869	0.877	**0.881**
Nickel	MSE	**5.150e+04**	5.688e+04	1.018e+05	**1.004e+05**
	MAE	**1.920e+02**	2.049e+02	2.272e+02	**2.231e+02**
	R^2	**0.874**	0.861	0.949	**0.950**
Tin	MSE	**8.717e+04**	1.193e+05	3.504e+06	**5.782e+05**
	MAE	**2.320e+02**	2.721e+02	9.639e+02	**5.822e+02**
	R^2	**0.958**	0.942	-0.602	**0.736**
Zinc	MSE	**1.206e+03**	1.250e+03	2.376e+03	**2.317e+03**
	MAE	**2.678e+01**	2.730e+01	3.914e+01	**3.910e+01**
	R^2	**0.834**	0.828	0.850	**0.853**

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se