AN IMPROVED NADAM OPTIMIZER BY INCORPORATING GRADIENT CLIPPING FOR LSTM NETWORK

TU JUN

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2026

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

TU, JUN

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"AN IMPROVED NADAM OPTIMIZER BY INCORPORATING GRADIENT CLIPPING FOR LSTM NETWORK"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : **27 October 2025.**
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: ***27 October 2025.***

Pengerusi Viva *(Chairman for Viva)*	: **Prof. Dr. Huda Haji Ibrahim**	Tandatangan *(Signature)*	
Pemeriksa Luar *(External Examiner)*	: **Prof. Dr. Azlan Mohd Zain**	Tandatangan *(Signature)*	
Pemeriksa Dalam *(Internal Examiner)*	: **Assoc. Prof. Dr. Azizi Ab Aziz**	Tandatangan *(Signature)*	
Nama Penyelia I *(Name of Supervisor I)*	: **Prof. Dr. Azman Yasin**	Tandatangan *(Signature)*	
Nama Penyelia II *(Name of Supervisor II)*	: **Gs. Dr. Nur Suhaili Mansor**	Tandatangan *(Signature)*	

Tarikh:
(Date) ***27 October 2025***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUMCollege of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Kepekatan ammonia merupakan parameter penting dalam menilai kualiti air akuakultur, dan ramalan yang tepat dapat menyokong pembangunan akuakultur pintar. Rangkaian Long Short-Term Memory (LSTM) menunjukkan prestasi yang baik dalam ramalan siri masa, seperti paras ammonia, namun keberkesanannya bergantung kepada proses pengoptimuman yang sesuai. Nesterov-accelerated Adaptive Moment Estimation (Nadam) menggabungkan kecerunan dipercepat Nesterov dengan kadar pembelajaran adaptif bagi mencapai penumpuan yang pantas, namun kaedah ini berpotensi menghadapi isu ketidakstabilan seperti lonjakan kecerunan. Kajian ini memperkenalkan NadamClip, iaitu pengoptimum yang dipertingkatkan dengan dua penambahbaikan utama: (i) penambahan mekanisme pemangkasan kecerunan bagi mengurangkan lonjakan kecerunan dengan mengehadkan kesan kecerunan luar biasa semasa proses kemas kini pemberat; dan (ii) pelarasan manual ambang pemangkasan serta pengujian bagi mengenal pasti nilai ambang kecerunan yang optimum untuk meningkatkan kestabilan latihan dan ketepatan ramalan. Berbanding pelbagai algoritma penurunan kecerunan stokastik dan algoritma kecerunan gabungan, NadamClip menunjukkan kestabilan latihan yang lebih baik serta mengekalkan ketepatan ramalan yang kompetitif. Kaedah ini mengatasi prestasi banyak pengoptimum lain dengan nilai RMSE sebanyak 0.2644, MAE sebanyak 0.6595, dan R^2 sebanyak 0.9743, di samping menunjukkan kestabilan latihan yang sangat baik. Berbanding penambahbaikan pengoptimum sedia ada, NadamClip memperkenalkan integrasi berstruktur antara pengklipan kecerunan dan penganggaran momentum adaptif, sekaligus mengatasi pendekatan konvensional yang menjadikan pengklipan sekadar kawalan latihan luaran dan bukannya sebahagian intrinsik dalam algoritma. Sehubungan itu, NadamClip berpotensi menjadi penyelesaian yang boleh dipercayai dalam menstabilkan serta meningkatkan prestasi model pembelajaran mendalam dalam akuakultur dan pelbagai bidang lain.

Kata kunci: Penurunan kecerunan stokastik, Pengklipan kecerunan, Long Short-Term Memory (LSTM), Data siri masa, Ramalan persekitaran akuakultur.

Abstract

Ammonia concentration is crucial for evaluating aquaculture water quality, and accurate prediction can facilitate smart aquaculture. Long Short-Term Memory (LSTM) networks perform well in time series prediction, such as ammonia levels, but their effectiveness depends on proper optimization. Nesterov-accelerated Adaptive Moment Estimation (Nadam) combines Nesterov-accelerated gradients with adaptive learning rates to achieve rapid convergence, but it can encounter instability issues like gradient spikes. This study introduces NadamClip, an improved optimizer with two key enhancements: i) Adding a gradient trimming mechanism to reduce gradient spikes by limiting the impact of abnormal gradients during weight updates. ii) Manually adjusting the cropping threshold and testing to find the best gradient threshold to improve training stability and prediction accuracy. Compared with various stochastic and fused gradient descent algorithms, NadamClip demonstrates better training stability while maintaining competitive prediction accuracy. It outperforms many optimizers, achieving an RMSE of 0.2644, MAE of 0.6595, and R^2 of 0.9743, while demonstrating excellent training stability. Compared to existing optimizer improvements, NadamClip leads the way by integrating gradient clipping with adaptive momentum estimation, moving beyond the traditional approach, where clipping primarily served as an external training control rather than an intrinsic part of the algorithm. Therefore, NadamClip offers a reliable solution for stabilizing and improving deep learning models in aquaculture and other fields.

Keywords: Stochastic gradient descent, Gradient clipping, Long Short-term memory, Time series data, Aquaculture environmental prediction.

Acknowledgements

This thesis would not have been possible without the support and guidance of many individuals, to whom I extend my deepest gratitude.

First and foremost, I wish to express my sincere appreciation to my main supervisor, Prof. Dr. Azman Yasin, for his unwavering support, insightful guidance, and constant encouragement throughout the course of my doctoral studies. His expertise and mentorship have been invaluable, and I am truly grateful for the opportunity to learn under his supervision.

I am also deeply thankful to my co-supervisor, Dr. Nur Suhaili Mansor, for her thoughtful feedback, constructive suggestions, and kind support, particularly during the writing and revision phases of this thesis. Her contributions significantly enhanced the quality of my work.

I would like to acknowledge the support of my friends and fellow researchers, whose encouragement, collaboration, and shared experiences made this journey both enriching and memorable.

Above all, I am profoundly grateful to my family. To my parents, thank you for your unconditional love, understanding, and belief in me. To my beloved partner, Peng Yingming, your patience, support, and quiet strength have carried me through the most challenging times. This accomplishment would not have been possible without you.

Completing this thesis has been both a demanding and rewarding journey. I am sincerely thankful to all who have supported me along the way.

Table of Contents

List of Tables

List of Figures

CHAPTER ONE
INTRODUCTION

1.1 Background

Aquaculture is an economic endeavour that employs artificial methods to cultivate and rear diverse aquatic organisms in aquatic environments. This encompasses fish, shellfish, shrimp, algae, and similar organisms in marine, freshwater, or saltwater habitats.(Islam & Yasmin, 2017). Aquaculture is a sustainable agricultural practice that meets human demand for seafood. It also plays a significant role in global food production (Pradeepkiran, 2019). With population growth and rising demand for high-quality protein, aquaculture has emerged as a viable solution to the limitations of traditional fisheries in fulfilling this demand (Boyd et al., 2020). Aquaculture enhances the production of aquatic products in a regulated environment, ensuring product quality and safety, thereby offering a dependable food source for the global population (Freitas et al., 2020).

Ammonia is a hazardous compound in aquatic environments, primarily originating from the excretion of aquatic animals and from undigested dietary material (Datta S, 2012). Ammonia is present in water as two species, NH_3 and NH_4^+, with NH_3 prevailing at elevated pH (often above 7.5) and NH_4^+ predominating at lower pH (below 7.5) (Owaes et al. 2024). NH_3 is more toxic and significantly affects the respiratory and nervous systems, whereas NH_4^+ is less toxic (Sadiq, 2024). Ammonia is oxidised to nitrite (NO_2-) by ammonia-oxidising bacteria (e.g., Nitrosomonas), a process that requires adequate oxygen levels (Lehtovirta-Morley, 2018). Nitrite is extremely hazardous to aquatic organisms, particularly affecting their respiratory systems. Nitrite is subsequently transformed into nitrate (NO_3-) by nitrifying bacteria,

such as Nitrobacter. Nitrate is comparatively less harmful. However, elevated levels can induce eutrophication in aquatic systems, leading to algal overgrowth and disrupting oxygen levels and ecological equilibrium (Jensen, 2003).

The pH, temperature, and dissolved oxygen concentration of the aquatic environment significantly influence the transformation of nitrogen compounds (Kaur et al., 2024). Elevated pH enhances ammonia toxicity, while reduced pH stabilises ammonium ions. Ammonia volatilisation intensifies at higher temperatures, amplifying its toxicity. Sufficient dissolved oxygen enables the transformation of ammonia and nitrite, whereas inadequate oxygen results in their buildup, compromising water quality (Boyd, 2017). In aquaculture, effective water quality management through the regulation of pH, temperature, and dissolved oxygen facilitates nitrogen cycling, mitigates the buildup of detrimental nitrogen compounds, ensures stable water quality, and protects the health of aquatic species (Ip & Chew, 2010).

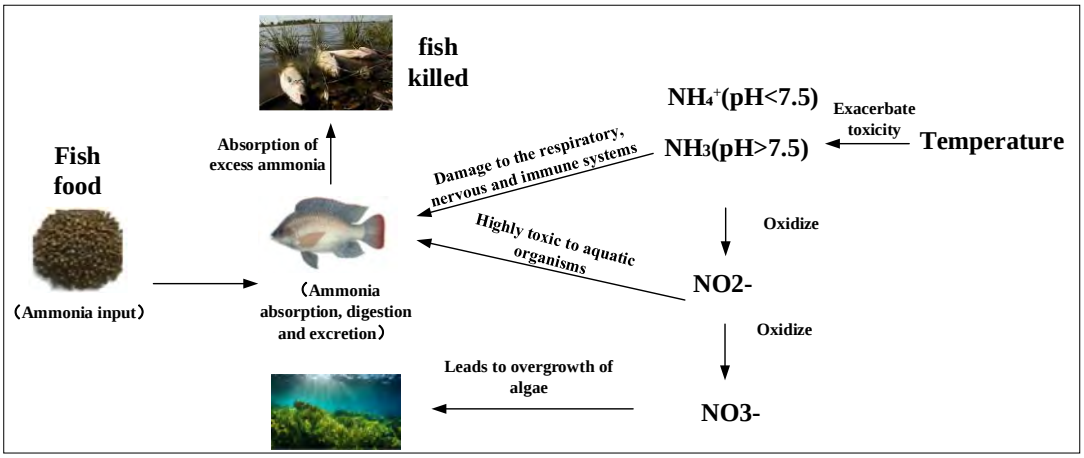

Figure 1.1. Flowchart of ammonia harms fish

Ammonia concentrations show a clear time dependence in aquaculture systems, driven mainly by the dynamics of biological excretion, environmental changes, and ammonia

transformation processes (Datta S, 2012; Lehtovirta-Morley, 2018; Kaur et al., 2024). Excretory activities of aquatic animals and undigested feeds periodically increase ammonia concentrations. At the same time, changes in water quality conditions (e.g., temperature, pH, and dissolved oxygen) can also affect ammonia volatilization and transformation rates. Therefore, due to the complexity of ammonia variations, advanced machine learning methods, especially long short-term memory (LSTM) networks, have been applied to ammonia prediction. LSTM models can effectively manage the serial characteristics of ammonia concentration data, improve prediction accuracy, and promote the sustainability of aquaculture practices (Wang et al., 2024).

Deep learning models, including LSTM models, typically rely on optimisation algorithms to adjust the model's weights and biases to minimise prediction error during training (Gambella et al., 2021). The Stochastic Gradient Descent (SGD) algorithm and its variants, such as Adam, Nadam, etc., are widely used optimisation methods that can effectively improve neural network performance by iteratively updating model parameters (Mehmood et al., 2023). In training LSTM and other neural networks, SGD helps the network gradually approach the optimal solution by continuously adjusting its parameters, thereby improving the accuracy and stability of the model's predictions of ammonia concentration. Therefore, the predictive performance of LSTM models heavily relies on the effectiveness of optimisation algorithms. Although stochastic gradient descent and its variants are commonly used for LSTM model training, they still face challenges, such as unstable training and insufficient accuracy in practical data prediction tasks.

To address these challenges, this study proposes improvements from an optimisation perspective, aiming to further enhance LSTM performance in ammonia concentration prediction. This advancement will more effectively support intelligent management and sustainable development in the aquaculture field.

1.2 Problem Statement

Nesterov-accelerated Adaptive Moment Estimation (Nadam) is an adaptive moment estimation algorithm, which is a variant of the stochastic gradient descent (SGD) algorithm that effectively reduces the training and validation losses and solves the limitations of the traditional SGD in terms of the adjustment of the learning rate, the gradient noise, the speed of convergence, and local optimal solutions (Dozat, 2016a). Nadam is often used in deep learning models, such as LSTM models, to improve training efficiency (Lozano Jimenez et al., 2019).

It has been shown that LSTM models have significant advantages for time-series data and tasks with long-term dependencies, and especially exhibit strong theoretical advantages in mitigating the gradient vanishing problem (Noh, 2021). However, due to the inherent architectural characteristics of its network structure, when training LSTM models with Nadam or other traditional optimisers (such as SGD, Adam, etc.), while convergence speed can be improved, it remains challenging to fully mitigate the occurrence of sharp gradient spikes and fluctuations in gradient magnitude, which may lead to unstable gradient dynamics (He et al., 2024). The impact of gradient instability, including both gradient vanishing and gradient spikes. As illustrated in Figure 1.2, gradient spikes during model training are evident in the blue curve, where the gradient norm increases abruptly at certain iterations.

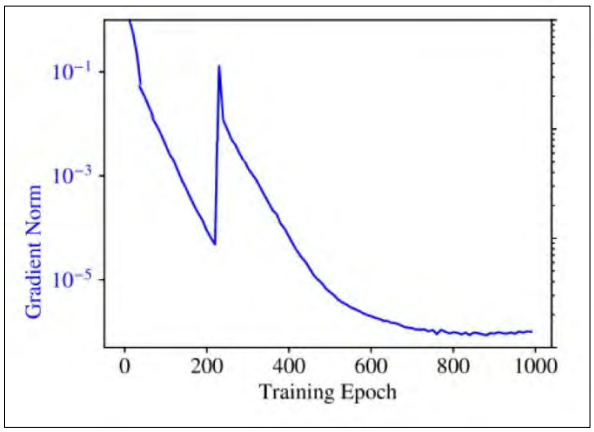

Figure 1.2. Gradient vanishing and explosion (Freire et al., 2022)

During the training of deep neural networks, it is common for the gradient norm to exhibit abrupt escalations, particularly in the early phases of optimisation. These transient high-magnitude gradient surges can induce disproportionately large parameter updates, thereby introducing local instability into the optimisation trajectory. While such gradient spikes may not immediately cause divergence, they constitute a form of gradient instability that disrupts the smooth progression of training and, if left unregulated, may elevate the risk of subsequent gradient spikes in more complex or sensitive training conditions. A direct consequence of such gradient behaviour is that the model may fail to gradually approach the optimal solution during training, instead oscillating around suboptimal regions or even diverging in extreme cases. Figure 1.3 illustrates the effect of large gradient variations on the optimisation process (Zhang et al., 2020), where the gradients deviate sharply from the desirable descent direction.

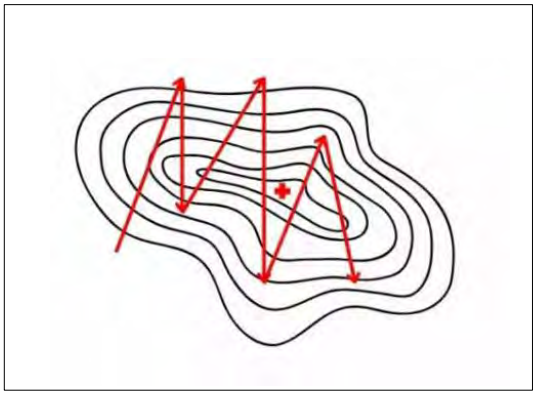

Figure 1.3. optimisation process of gradient spike impact parameters (Van Otten, 2023)

Gradient clipping is a solution to unstable gradient dynamics, constraining the magnitude of gradient updates to avoid excessively large gradient norms (Black, 2020). When the gradient norm exceeds a predetermined threshold, it is clipped to within that threshold. This helps the model to maintain training stability and efficient convergence by suppressing sharp gradient spikes and smoothing the gradient updates. As shown in Figure 1.4, when the gradient norm exceeds the red-dashed threshold, the value is clipped, effectively stabilising the training process and helping the model converge gradually.

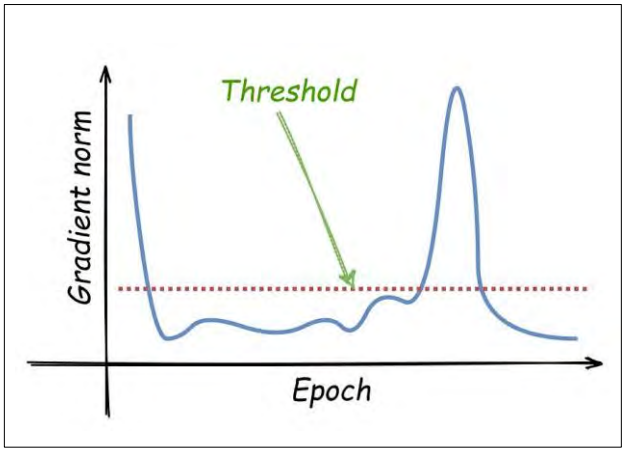

Figure 1.4. Gradient threshold clipping (Or, 2020)

By constraining the gradient norm within a predefined range through gradient clipping, the optimisation updates become smoother and more controlled. This allows the model

to adjust its parameters more steadily along the descent direction of the loss function and mitigates training instability caused by abrupt gradient surges and fluctuations in gradient magnitude. Such stabilisation of the optimisation process facilitates gradual convergence and enhances the overall effectiveness of model training. Figure 1.5 illustrates the optimisation process after applying gradient clipping.

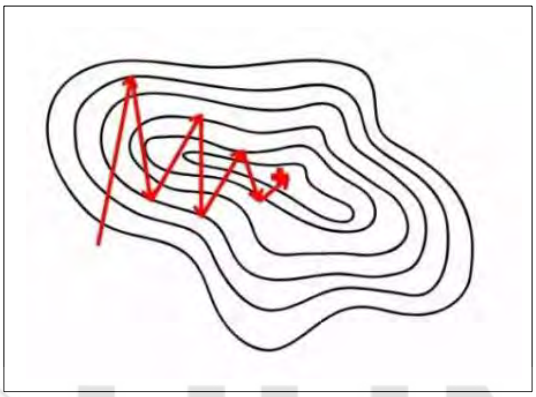

Figure 1.5. optimisation process of gradient clipping impact parameters (Van Otten, 2023)

To optimise model training and address the gradient spike problem, previous studies have introduced gradient clipping techniques into stochastic gradient descent algorithms. For example, the algorithms DPSGD and SGD with U-Clip (Abadi et al., 2016; Elesedy & Hutter, 2023) introduced the gradient clipping technique into SGD. Introducing the gradient clipping technique into Adam's CGAdam (Sun, Yu et al., 2024). ClipRAdaGradD that introduced gradient clipping to AdaGrad (Chezhegov et al., 2024), and AdamW_AGC combined gradient clipping with AdamW (Sun, Cui, et al., 2024).

However, while the aforementioned algorithms address the gradient spike issue by introducing gradient clipping into stochastic gradient descent, their ability to escape local optima on complex loss surfaces is limited by the lack of Nesterov momentum

or excessive reliance on historical gradient information. Consequently, in fields such as aquaculture, industrial production, and finance, where data volumes surge and task complexity increases, these methods still face challenges of insufficient predictive accuracy or unstable model training in practical applications, resulting in suboptimal model performance in real-world deployment.

The combination of Nesterov accelerated momentum and adaptive learning rates in the Nadam optimiser allows each gradient update to rely not only on the current gradient but also to anticipate future gradient directions, thereby improving the model's predictive accuracy. However, to date, no studies have effectively integrated gradient clipping into the Nadam algorithm to enhance predictive performance. At the same time, suppress sharp gradient spikes, smooth gradient variations, and reduce the potential risk of gradient spikes. This study proposes an improved optimisation algorithm, NadamClip, that incorporates gradient clipping directly into Nadam's internal mechanism, unlike conventional approaches that treat gradient clipping as an external training control. By incorporating clipping into the optimiser's internal dynamics, NadamClip offers a principled approach to stabilising gradient updates, refining the theoretical structure of Nadam, and extending the foundational design space of adaptive momentum optimisers. Furthermore, through extensive experimentation, the optimal gradient clipping threshold is identified, enabling the algorithm to balance improving prediction accuracy with maintaining training stability.

1.3 Research Questions

The questions to be answered by this research are as follows:

1. How can Nadam be modified to integrate gradient clipping internally to improve convergence stability and mitigate gradient spikes?

2. How to find the appropriate range for gradient clipping into NadamClip?

3. How can NadamClip be evaluated in terms of its theoretical stability, convergence properties, and generalizability across architectures and data distributions?

1.4 Research Objectives

1. To develop a NadamClip algorithm by embedding gradient clipping within Nadam and to establish its theoretical convergence and gradient stability properties.

2. To determine the optimal clipping range, different gradient clipping thresholds were experimentally explored and adjusted to improve the prediction performance.

3. To evaluate NadamClip through theoretical validation and empirical comparison, examining its practical convergence behaviour, gradient stability performance, and generalizability across different models and datasets.

1.5 Scope of Research

This study investigates NadamClip as a general optimisation framework that integrates gradient clipping into the Nadam update rule, aiming to improve optimisation stability and convergence across a broad range of neural architectures and learning tasks. Beyond its empirical application to ammonia concentration forecasting, the research extends to a theoretical examination of NadamClip's convergence behaviour, gradient stability, and gradient norm boundedness. The scope further includes analysing its

robustness and generalizability across different model structures, data distributions, and task types to ensure that the algorithm provides a foundational contribution rather than an application-specific enhancement. Through both analytical and empirical evaluations, this study establishes suitable gradient clipping ranges, validates NadamClip across diverse optimisation conditions, and demonstrates its potential as a theoretically grounded, widely applicable optimisation method. I will delineate the specific research techniques depicted in Figure 1.6:

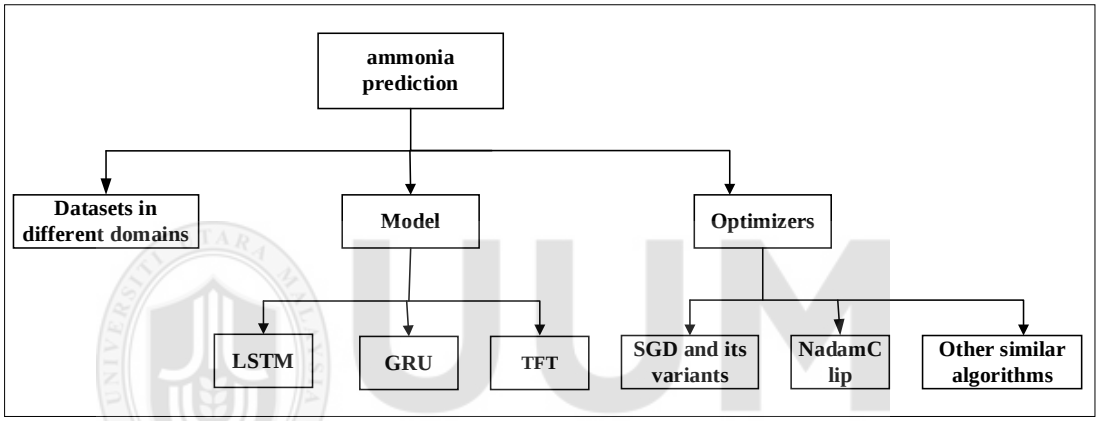

Figure 1.6. Flowchart of the scope of research

1.6 Significance of Study

1.6.1 Theoretical Significance

The NadamClip algorithm proposed in this study combines the Nadam optimisation algorithm and gradient clipping, innovatively addressing the gradient spike problem in deep learning model training and improving training stability and prediction accuracy. The Nadam algorithm itself has strong adaptive learning, which can effectively adjust the learning rate and optimise the model training speed, but is still insufficient in the face of the gradient spike phenomenon. However, it is still insufficient in the face of a gradient spike. This study provides a new way of thinking

by introducing the gradient clipping technique, which further improves the performance of the Nadam algorithm. This innovation not only offers a new direction for research on optimisation algorithms but also provides a more effective training strategy for deep learning models when tackling complex tasks.

In addition, the introduction of gradient clipping technology enables NadamClip to avoid the slow convergence and local optimal solution problems associated with traditional methods when addressing gradient spikes. This not only enriches the theoretical system of gradient optimisation techniques but also provides a new theoretical framework for academics to improve the convergence and accuracy of algorithms by combining adaptive learning rate and gradient clipping techniques, especially in high-dimensional, complex loss-surface problems.

1.6.2 Practical Significance

In practical applications, especially in aquaculture, industrial production, finance, and other fields, when faced with large data volumes, dynamic data changes, and high-dimensional features, model training is often plagued by problems such as gradient spike, slow convergence, and over-reliance on historical gradient information. These problems seriously affect the performance of deep learning models in practical applications, especially in real-time prediction, dynamic adjustment, and other demanding tasks, often leading to unsatisfactory prediction results. The proposed NadamClip algorithm avoids the gradient spike problem in deep learning training and completes training more successfully, improving the model's prediction accuracy.

In aquaculture, accurate prediction of ammonia concentration changes is crucial for water quality management. Through the NadamClip optimisation algorithm, the deep learning model can more accurately predict changes in ammonia concentration, helping farmers adjust the aquaculture environment in time to avoid the harm of high ammonia levels to aquatic organisms. This not only improves farming efficiency and reduces resource waste but also promotes the sustainable development of the aquaculture industry. By improving the stability and accuracy of model training, the NadamClip algorithm helps solve complex problems in practice, promotes the widespread use of deep learning models across industry applications, and provides efficient decision support for related fields.

In conclusion, the NadamClip algorithm not only bridges the gap between optimisation algorithms and gradient clipping in theory but also provides an effective solution in practice. This has significant application value for improving the performance of deep learning models, especially in high-complexity, high-dimensional tasks.

1.7 Theoretical Organization

This study revolves around NadamClip, a new optimisation algorithm that combines Nadam and gradient clipping technology with a clear and hierarchical structure, which is organised as follows:

i. Chapter One Introduction

This chapter briefly introduces the study's background, problem statement, goals and significance, and scope and purpose. By providing an overall overview of the research, it provides a theoretical foundation and direction for the subsequent chapters.

ii. Chapter Two: Literature Review

This chapter provides an in-depth review of the existing literature relevant to this study. It starts with an introduction to the effects of ammonia in aquaculture and to advances in traditional and modern prediction methods. Then, the advantages and limitations of various optimisation algorithms, especially gradient-based methods such as Nadam, are discussed, with a focus on the application of gradient clipping techniques in optimisation. By comprehensively analysing the methods and results of existing research, this study provides a theoretical basis.

iii. Chapter Three Methodology

This chapter details the overall methodological design of the study, focusing on the construction and optimisation process of the NadamClip algorithm. First, high-quality data were provided for the experiment through preprocessing, cleaning, and normalisation. Next, the design of the NadamClip algorithm is proposed, and the gradient clipping threshold range is experimentally explored to optimise the model's training stability and prediction accuracy.

iv. Chapter Four: Enhanced Nesterov Accelerated Adaptive Moment Estimation Algorithm

This chapter introduces the basic principles of the Nadam optimisation algorithm and provides an in-depth discussion of the gradient clipping technique. The process of combining Nadam with gradient clipping is described in detail, and the NadamClip algorithm is proposed, with its working principle and steps clarified. The

implementation of the NadamClip optimisation algorithm and its operation mechanism are clearly demonstrated through pseudo-code and an algorithmic flow diagram.

v. Chapter 5 Experiments and Discussion

This chapter first describes the preparation for the experiment, including the setup of the experimental environment and data processing. Then, the experimental results of the NadamClip algorithm are highlighted and compared with those of existing optimisation algorithms (e.g., SGD, Adam). Through comparative analysis, the performance of NadamClip is evaluated in terms of training stability, prediction accuracy, and computational overhead. Meanwhile, this chapter also explores the applications of NadamClip in other fields, further validating its broad practicality and innovation.

vi. Chapter 6 Conclusion and Future Work

This chapter summarises the study's contributions, both theoretical and practical. The limitations in the study are analysed, and future research directions are proposed, aiming to provide valuable references and insights for the subsequent development of optimisation algorithms and the application of deep learning models.

Through the detailed discussions in the above chapters, the paper systematically demonstrates the innovativeness and application value of the NadamClip optimisation algorithm. It provides new ideas for improving the training efficiency and prediction accuracy of deep learning models.

1.8 Summary of the Chapter

This chapter examines the potential benefits of combining the Nadam algorithm with gradient clipping to train LSTM models for predicting ammonia levels in aquaculture. The study's issues and problems were presented, followed by an explanation of the problem statement, research questions, research objectives, and the study's significance. It seeks to enhance this combination for greater precision, evaluate its efficacy relative to alternative methods, and offer pragmatic recommendations for its application in real-world aquaculture contexts.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

This chapter provides a literature review of the study. As outlined in the preceding chapter, forecasts were generated employing a range of methodologies, encompassing statistical, chemical, and machine learning approaches. This section examines the impact of ammonia in aquaculture and reviews prior studies employing diverse methodologies, including machine learning techniques. A summary of LSTM, optimisation methods, and gradient clipping techniques is provided. The advantages and disadvantages of the Nadam optimisation technique, as well as existing gaps in the field, are emphasised.

2.2 Effects of Ammonia

2.2.1 Ammonia in Aquaculture

Ammonia nitrogen is highly toxic to aquatic organisms, with safe long-term exposure concentrations being below 0.2 mg/L for total ammonia nitrogen (TAN) and below 0.02 mg/L for inorganic ammonia (NH3-N). The acute toxicity thresholds are 1.5-3.3 mg/L NH3-N and 2.9-5.0 mg/L TAN, respectively. Toxicity is markedly increased, particularly at elevated pH levels and high temperatures, necessitating intensified monitoring and regulation (Levit, 2010; Zhang et al., 2023; Zhou & Boyd, 2015).

Moreover, elevated concentrations of ammonia/ammonium enhance the primary production of plankton communities, leading to eutrophication of water bodies and accelerated algal proliferation, which can adversely affect the entire aquatic ecosystem

(Bhateria & Jain, 2016). The concentration of dissolved organic matter will increase following the reduction in the algal population, promoting microbial respiration and decreasing the dissolved oxygen levels. Ammonia toxicity is particularly detrimental to marine fish and invertebrates in anoxic environments (Li et al., 2020).

Furthermore, ammonia and its salts are toxic to humans and crustaceans, with aquatic organisms being particularly vulnerable (Karri et al., 2018). Yang et al. (2017) posited that ammonia can lead to the extinction of entire populations of fish and other invertebrate larvae during their developmental stages, thereby threatening numerous vital ecosystems. John et al. (2020) asserted that the ecology, life cycle, and production of ammonia nitrogen are all significant. Therefore, to ensure the sustainable utilisation of water resources and safeguard aquatic organisms from the acute and chronic impacts of ammonia, it is essential to detect, sense, and monitor ammonia nitrogen concentrations in lakes, rivers, seawater, and other water bodies.

The expanding global population, projected to reach 8 billion by 2022, has driven increased food demand, thereby accelerating the production of aquaculture species (Chen et al., 2019). Mustafa et al. (2021) assert that aquaculture significantly supports the sustainable supply of aquatic animal demand. Ahmed et al. (2020) posited that aquaculture is vital for supplying a burgeoning human population with organic food rich in protein and minerals. In the last four decades, aquaculture has expanded significantly, becoming a crucial source of valuable animal protein to satisfy global demand. According to Datta S (2012), ammonia is a natural byproduct of fish metabolism and the decomposition of organic matter in aquaculture systems.

Excessive ammonia concentrations can adversely affect aquatic ecosystems by inducing water pollution, eutrophication, and fish mortality. Examining ammonia concentrations helps mitigate these adverse environmental effects (Nagaraju, Chaudhary, et al., 2023). Elevated ammonia levels in aquaculture systems can induce stress in fish, resulting in reduced growth rates, heightened vulnerability to disease, and increased mortality (Li et al., 2022). Likewise, Syanya et al. (2023) discovered that elevated ammonia concentrations in aquaculture systems can lead to the accumulation of ammonia residues in fish tissues, potentially posing health risks to humans upon consumption.

Consequently, the study by Li et al. (2023) indicated that the long-term viability of aquaculture relies on the management and mitigation of its detrimental environmental impacts, particularly ammonia pollution. Furthermore, Ahmad et al. (2022) emphasise the need to investigate ammonia levels to safeguard the health and welfare of farmed fish. Thus, analysing and forecasting ammonia concentrations in aquaculture is essential for fostering industry expansion, minimising environmental impacts, safeguarding fish health and welfare, optimising resource use, and meeting consumer demand for sustainable, safe seafood.

2.2.2 Ammonia Data Set Characteristics

Ammonia in water constitutes time-series data due to its concentration fluctuations and inherent temporal dependence. Wang et al. (2023) conducted a comprehensive study of ammonia nitrogen (NH3-N) in river water quality using Pearson correlation coefficient analysis and time-series trend analysis. The analysis revealed that ammonia nitrogen concentration exhibited notable temporal trends: daily variations indicated a

peak in the early morning (4:00-8:00) and a decline in the afternoon (16:00-20:00); monthly variations demonstrated elevated levels in the middle of the month and reduced levels at the month's end; and seasonal variations showed the highest concentrations in summer and the lowest in spring, influenced by environmental factors such as temperature, precipitation, and microbial metabolism. Yu et al. (2021) examined the characteristics of ammonia nitrogen (NH3-N) as time-series data in aquaculture and employed the empirical mode decomposition (EMD) method to analyse its temporal dynamics. The outcomes of the time series trend analysis indicated that ammonia nitrogen concentration was influenced by various environmental factors, including water temperature, dissolved oxygen, and rainfall, and exhibited nonlinearity and nonstationarity. Li (2023) examined the characteristics of NH3-N as time series data in the surface water domain using an enhanced deep learning hybrid model, Bias Correction-Maximal Overlap. Discrete Wavelet Transform - Data Assimilation - Long Short-Term Memory (BC-MODWT-DA-LSTM). The time-series trend analysis indicates that the fluctuations in ammonia nitrogen concentration exhibit multiscale periodicity and nonlinearity, and are significantly influenced by dissolved oxygen, wind speed, temperature, and other factors. The study indicated that the predictive accuracy of ammonia nitrogen concentration at the peak of high pollution was notably high, with the model's Nash-Sutcliffe Efficiency (NSE) exceeding 0.990 over a five-day period, demonstrating exceptional early warning capability and stability and offering significant support for water quality monitoring and pollution management.

Kang et al. (2021) examined the characteristics of ammonia nitrogen concentration as time-series data in wastewater treatment using Grey Correlation Analysis (GRA) and

a Time Convolution Network (TCN). The data encompass fundamental water quality parameters, including NH3-N, pH, dissolved oxygen (DO), and chemical oxygen demand (COD), as well as meteorological parameters such as air temperature (T), wind direction and speed (WDSP), precipitation (PRCP), and sea level pressure (SLP). The variables Dissolved Oxygen (DO), Total Phosphorus (TP), and Oxidation-Reduction Potential (ORP) in wastewater showed strong correlations with ammonia nitrogen concentration as determined by the GRA method, and the processed multivariate time series data were modelled using TCN. The time-series trend analysis indicated that the ammonia nitrogen concentration exhibited significant nonlinearity, pronounced temporal variability, lag effects, and was affected by numerous intricate factors.

Wu et al. (2023) examined the dynamic characteristics of ammonia nitrogen as time-series data in aquaculture, employing LSTM in conjunction with game theory (SHAP values) to predict the impact of low ammonia nitrogen concentrations on fish mortality, specifically in largemouth bass. The findings from the time-series trend analysis indicated that ammonia nitrogen concentration and fish mortality exhibited significant nonlinear and time-accumulative traits, influenced by dynamic fluctuations in environmental factors, including water temperature, dissolved oxygen, and pH.

Wang et al. (2024) developed a multi-model framework combining the Expectation Maximisation (EM) algorithm and Fuzzy C-Means (FCM) clustering to analyse and predict fluctuations in ammonia nitrogen concentrations in aquaculture water for Stichopus japonicus. The study identified significant nonlinear relationships between ammonia nitrogen levels and key water quality parameters, including temperature, pH, salinity, and nitrite concentration. By leveraging a multi-model fusion approach,

prediction accuracy was notably enhanced, enabling improved representation of seasonal variability in ammonia nitrogen. The proposed method was validated with empirical data from an aquaculture facility in Dalian, confirming its reliability and practical applicability. Table 2.1 summarizes these articles.

Table 2.1

Ammonia data set characteristics

Author	**Analysis Method**	**Findings**	**Dataset**
X. Wang et al., (2023)	Pearson correlation coefficient	Ammonia nitrogen concentration showed a significant time-series trend	9 water quality parameters
H. Yu et al., (2021)	Empirical model decomposition method	Ammonia nitrogen is characterised by nonlinearity and nonstationarity	Water quality data were collected from a total of 750 samples
Y. Li & Li, (2023)	BC-MODWT-DA-LSTM	The variation of ammonia nitrogen concentration is characterised by multi-scale periodicity.	Water quality monitoring data and meteorological data
Kang et al., (2021)	GRA and TCN	Ammonia nitrogen concentration is significantly nonlinear, strongly time-varying and lagging.	NH_3-N、pH、DO、COD、T、WDSP、PRCP and SLP
Wu et al., (2023)	LSTM and SHAP	Ammonia nitrogen concentration and fish mortality showed significant nonlinear and time-cumulative characteristics.	Water quality data from 100 ponds

Table 2.1 continued

Author	Analysis Method	Findings	Dataset
Wang et al. (2024)	EM and FCM	Identified strong nonlinear relationships between ammonia nitrogen concentration and water quality parameters	Temperature, pH, salinity, nitrite, Marine Biological Seed Technology

In summary, ammonia exhibits substantial time-dependent and nonlinear characteristics as time series data in river water quality, aquaculture, surface water, and wastewater treatment. Its concentration variations are influenced by numerous environmental factors, including temperature, dissolved oxygen, and precipitation, resulting in a cyclical, multi-scale, and dynamic pattern of change.

2.3 Ammonia Prediction Methods

Predicting ammonia concentrations in aquaculture requires integrating monitoring, data analysis, and modelling. Previous research has suggested various methodologies for ammonia prediction in aquaculture, including chemical analysis (Li et al., 2020), statistical models (Nagaraju et al., 2023), and, more recently, machine learning (Suarin et al., 2022; Yu et al., 2021a), among others.

Scientific evidence indicates that forecasting ammonia levels in aquaculture is a complex endeavour owing to the myriad variables and factors that can affect ammonia dynamics within aquaculture systems (Levit, 2010). The selection of a method often depends on the specific aquaculture system, its objectives, and the resources available. Furthermore, current research in this domain may help develop more precise and

resilient predictive models to regulate ammonia concentrations in aquaculture (Gladju et al., 2022).

2.3.1 Traditional Methods

Conventional ammonia prediction techniques typically employ chemical analysis methods to directly assess ammonia concentration. The chemical analysis methods provide exceptional accuracy and precision, making them suitable for research and regulatory compliance (Ivleva, 2021). Nevertheless, they generally necessitate intermittent sampling (Bazar, 2021). The predictive methodology for chemical analysis is a multi-phase process designed to obtain valuable information via sample collection, preparation, chemical analysis, and data documentation and evaluation. Figure 2.1 illustrates the specific process.

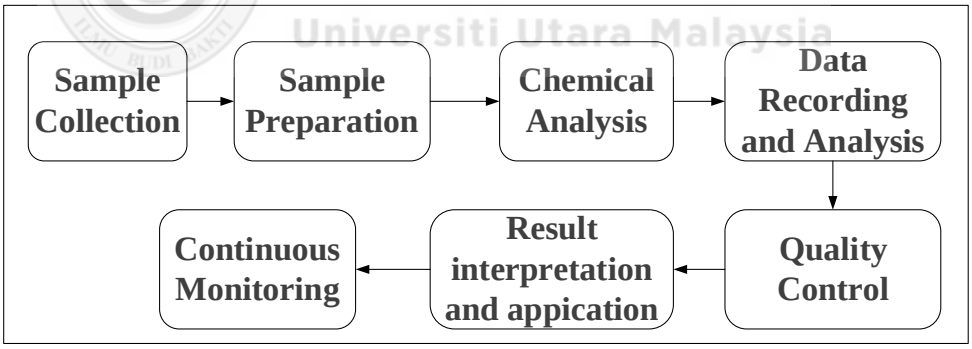

Figure 2.1. Flowchart of chemical analysis (Renu Nayar, 2020)

Shefat et al. (2021) focus on the physicochemical properties of water from the Pasur River Estuarine (PRE) in relation to the mangrove ecosystem of the Sundarbans during the dry season. Temperature (20.7–33.7°C), pH (7.1–7.9), salinity (8.5–16.2 PSU), and dissolved oxygen (5.9–8.4 mg/L) were quantified at fifteen locations. Phosphate showed significant variability, whereas ammonium-nitrogen was the predominant

component associated with mangrove sediment leaching. Correlations indicated associations among temperature, salinity, and dissolved oxygen. In the dry season, the PRE was identified as a nitrogen-limited estuarine system. Alum et al. (2023) assessed the physicochemical properties, metal concentrations, microbial load, and antibiotic resistance of three drinking water samples. resources in Amaozara Ozizza, Nigeria. Although most parameters met standards, deviations in COD, turbidity, PO43-, and NH4 indicated the need for water treatment. Detected pathogens indicate faecal contamination and pose health hazards to the community. The isolates exhibited diverse antibiotic sensitivities, underscoring the significance of water safety protocols. Könemann et al. (2018) delineate efficient screening techniques, standardise monitoring practices, and assist in Water Framework Directive evaluations. Water samples from throughout Europe were analysed, demonstrating similar 17β-estradiol levels, although discrepancies were observed depending on the substance's potency. Significantly, derived equivalents exhibit strong correlation with Liquid Chromatography-Tandem Mass Spectrometry (LC-MS/MS) analyses, underscoring the importance of effect-based screening in environmental monitoring. This aligns with the EU 2015/495 decision, which includes steroidal estrogens in the Water Framework Directive watch list, necessitating rigorous chemical monitoring. Zhou et al. (2024) developed an electrochemical method using a nickel–copper carbonate hydroxide-modified carbon cloth electrode for the rapid detection of ammonia nitrogen in water. The electrode was prepared via a straightforward one-step hydrothermal process and used the i–t (current–time) technique for highly sensitive analysis. Experimental results revealed a remarkable linear response from 1 to 400 μM, with sensitivities of 3.9 and 3.13 μA μM^{-1} cm^{-2}, and a detection limit as low as 0.62 μM, demonstrating excellent analytical performance. Additionally, the sensor exhibited

strong anti-interference against common anions and cations, and offered low cost, simple operation, and suitability for rapid monitoring of drinking and environmental water quality. Table 2.2 summarises the analytical methods, summaries, sampling techniques, and datasets from the aforementioned papers.

Table 2.2

Chemical analysis

Author	Analysis Method	Findings	Sample Method	Dataset
Shefat et al. (2021)	Cadmium Reduction Method	Correlations among temperature, salinity, and total dissolved solids. Ammonium-nitrogen dominated, attributed to tidal wash-out, while phosphate varied.	In-situ measurement	pH, temperature, salinity
Alum et al., (2023)	APHA method	The majority of the parameters remained within the allowed range set by the predetermined standards. Some parameters, such as COD, turbidity, and PO4, deviated from set standards; NH4 can be improved by adequate water treatment.	On-site measurement using standard methods	pH, temperature
Könemann et al., (2018)	Effect-based methods for wastewater and surface water monitoring.	Chemical analytical methods were limited in their ability to quantify E2 and EE2 at the required concentrations. Effect-based methods aid estrogenic compound screening, complementing routine monitoring. Effect-based methods are valuable; chemical analysis confirmation needed.	Chemical analytical measurement	estrogens level

Table 2.2 continued

Author	Analysis Method	Findings	Sample Method	Dataset
Zhou et al. (2024)	Ni–Cu carbonate hydroxide -modified carbon cloth electrode	Developed a highly sensitive and rapid electrochemical sensor for ammonia nitrogen detection in water; showed excellent linearity (1–400 μM), sensitivities of 3.9 and 3.13 μA μM^{-1} cm^{-2}, and a low detection limit of 0.62 μM; exhibited strong anti-interference and cost-effectiveness	Laboratory prepared ammonia nitrogen standard solutions	Water samples

The investigations on PRE water offer significant physicochemical insights but lack a thorough pollution assessment and a long-term perspective. Concentrating on a singular season restricts comprehension. The research by Alum et al. (2023) on Nigerian drinking water is vital for public health; nonetheless, it is deficient in specifics regarding detection methodologies and requires an expanded geographic and temporal scope. Könemann et al. (2018) advocate screening methods for European water, but their work lacks clarity regarding methodological details and potential sample selection biases. Chemical analysis, in conjunction with historical data and predictive modelling, is beneficial in regulating water quality in aquaculture, especially for quantifying ammonia concentrations. Although chemical analysis is precise and reliable, it has downsides, including labour and time intensity, as it requires sample collection, preservation, and laboratory analysis. This presents difficulties with regular oversight. Inconsistencies in sample collection methods may lead to inaccuracies, and the potential for tampering raises apprehensions. To address these challenges, research indicates the use of statistical models for continuous forecasts derived from real-time data from water quality sensors, facilitating prompt responses

to fluctuating conditions. The ensuing literature review examines various statistical models.

2.3.2 Predictive modelling methods

Predicting nonlinear time series data is a challenging analytical domain that entails modelling the nonlinear dependencies and dynamics inherent in time series data. In this domain, there is no definitive "gold standard" model, yet certain strategies are prevalent owing to their adaptability and efficacy. Numerous models serve as benchmarks for forecasting nonlinear time series, including statistical models, support vector machines (SVMs), random forests (RFs), artificial neural networks (ANNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). Linear models, including ARIMA models and their derivatives, can accommodate linear relationships and non-seasonal fluctuations in time series data. While they predominantly address linear relationships, they can also forecast certain fundamental nonlinear patterns via suitable modifications (Kontopoulou et al., 2023). Support Vector Machines (SVMs) use kernel methods to map input data into a high-dimensional space to identify the optimal decision boundary within that space. This allows SVMs to handle nonlinear and high-dimensional data effectively (Ghosh et al., 2019). Random forests enhance predictive accuracy and stability by integrating numerous decision trees. They exhibit reduced sensitivity to feature selection and are adept at managing intricate nonlinear interactions (Xuan et al., 2021). Artificial Neural Networks (ANNs) can learn and represent intricate nonlinear relationships due to their multi-layer architecture, rendering them especially adept at handling many forms of nonlinear data. While CNNs are primarily used for image processing, they adeptly capture local interdependencies and temporal dynamics in time series data, particularly

in multivariate time series prediction problems (Wang et al., 2019). Recurrent Neural Networks (RNNs) are particularly adept at handling sequential data by forecasting future values from prior information; Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures are optimal choices for processing long sequences (Zeebaree et al., 2021).

2.3.2.1 Statistical Models

Conventional time series models for forecasting ammonia concentrations in aquaculture systems include mathematical and statistical methodologies to analyse historical data and derive forecasts based on variable interrelationships (Liu et al., 2023). As illustrated in Figure 2.2, the statistical model prediction methodology is an extensive process that generates precise predictions through data collection and processing, model selection, model training and validation, model optimisation and deployment, and ongoing enhancement. The process commences with the acquisition of pertinent data and encompasses gathering information from various sources.

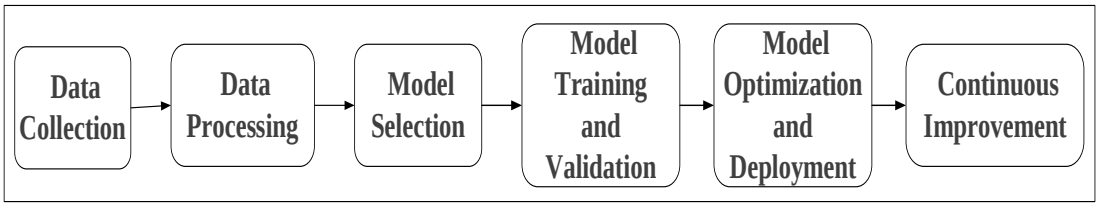

Figure 2.2. Flowchart of statistical model (Silva et al., 2021)

Statistical models, such as the Autoregressive Moving Average (ARMA) and the Autoregressive Integrated Moving Average (ARIMA), are extensively employed for time series analysis. Sansa et al. (2020) forecast winter solar radiation using an ARMA model. The simulation results indicate that the ARMA model accurately predicts solar

radiation performance. Wang et al. (2018) use an ARMA model to forecast short-term wind power using historical data. By analysing prediction errors using measures such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), the ARMA model demonstrates its efficacy in forecasting wind power. In reaction to environmental issues, there has been an increased global focus on reducing carbon emissions and promoting renewable energy, especially wind power. Alabdulrazzaq et al. (2021) evaluate the efficacy of an ARIMA model in forecasting COVID-19 cases over a prolonged duration in Kuwait. Researchers refined the parameters and identified a best-fit model, with a strong correlation (Pearson's coefficient of 0.996) between predicted and actual data. Notwithstanding the pandemic's fluctuating characteristics, the ARIMA model demonstrated precision and adequacy.

Cui et al. (2021) evaluate the feasibility of using CO2 for geothermal energy production in exhausted high-temperature gas sources from both technical and financial perspectives. The research investigates the interplay of gas composition, temperature, and pressure during enhanced gas recovery and geothermal exploitation by analysing a depleted gas reservoir at 120°C. The results demonstrate that maintaining a 10 MWth heat extraction rate for 30 years is feasible. Thermodynamic cycle analyses indicate a power output of 132.7 kW for an organic Rankine cycle system, with a cost of $0.45 per kW · h. In contrast, using CO2 directly for turbine power increases output to 718.5 kW and reduces cost to $0.10 per kW·h.

Kharchenko (2021) evaluated the efficacy of linear techniques, including Principal Component Analysis (PCA), against nonlinear downscaling methods (e.g., t-SNE, UMAP) utilising single-cell RNA sequencing (scRNA-seq) data. The study revealed

that PCA inadequately represents intricate expression manifolds, highlighting the limitations of linear methods such as PCA and conventional linear regression in capturing nonlinear relationships in high-dimensional data. Song et al. (2017) employed an Autoregressive (AR) model for deterioration modelling and Remaining Useful Life (RUL) prediction of lithium-ion batteries. It was observed that the AR model, being a linear regression method, is inadequate for the non-linear deterioration process of lithium-ion batteries, leading to underfitting in long-term predictions. Siddique and Mahalder (2025) used ARIMA and ARIMAX models in their PLOS One study to analyse the effects of climatic and water-quality factors on tilapia growth and ammonia nitrogen levels. The results showed that ammonia nitrogen exhibited clear seasonal patterns, mainly influenced by water temperature and solar radiation, demonstrating that traditional statistical models can effectively predict changes in ammonia nitrogen and support water quality management. Table 2.3 summarises the methods, findings, measurements and datasets of the above articles.

Table 2.3

Statistical analysis

Author	Method	Finding	Measurement	Dataset
Sansa et al. (2020)	Complete Orthogonal Decomposition (COD) Learning	EELM excels and outperforms ELM and SVR in river forecasting (2009–2014). Reliability persists.	MSE =0.2931 MAE=0.2581 RMSE=0.5414	Rainfall, water level, and physical parameters

Table 2.3 continued

Author	Method	Finding	Measurement	Dataset
Wang et al. (2018)	Ultra-short-term predictions method	A diagnostic system effectively aids in predicting heart disease risk factors.	MAE= 9.14% RMSE = 30.24% MAPE= 9.86%	Time series of wind power historical data
Alabdulrazzaq et al. (2021)	Statical method	ARIMA to forecast long-term COVID-19 cases in Kuwait	MAPE= 3.92	COVID-19 sites of the WHO and the Kuwait government's Feb to May, 2020
Cui et al. (2021)	Geothermal exploitation	The accuracy of the ARMA model in the prediction of solar radiation in winter	ES4 exhibits varying porosity and Permeability, with Average values ranging from 10.6% to 13.6% and 3.0e10 mD, respectively.	Reservoir pressure, Temperature and MPa
Kharchenko (2021)	PCA, UMAP	Linear methods such as PCA and traditional linear regression fail to capture non-linear relationships in high-dimensional data.	ROC-AUC values close to 0.9	Single-cell RNA sequencing

Table 2.3 continued

Author	Method	Finding	Measurement	Dataset
Song et al., (2017)	AR, ND-AR, IND-AR	The degradation process of lithium-ion batteries is non-linear, which leads to linear models exhibiting underfitting in long-term predictions.	The RUL error of the IND-AR model decreases gradually, from 10.7 per cent at the 40 per cent starting point to 2.1 per cent at the 80 per cent starting point.	CALCE battery dataset and satellite battery dataset
Siddique & Mahalder (2025)	ARIMA and ARIMAX time series models	Ammonia nitrogen showed seasonal variation and was mainly affected by water temperature and sunlight; ARIMA models accurately predicted its changes and fish growth.	R^2 = 0.93 and low prediction errors (<5%)	Three-year continuous monitoring data (2021–2024) from tilapia broodfish ponds in Bangladesh

In conclusion, conventional statistical models like linear regression are appropriate for linear data but struggle to encapsulate intricate nonlinear interactions. This constrains their efficacy in contexts such as forecasting ammonia concentrations in aquaculture. Machine sensor technology has emerged as a viable alternative, garnering significant academic attention.

2.3.2.2 Traditional Machine Learning Models

Recently, the utilisation of machine learning techniques in aquaculture has garnered considerable interest among researchers (Suarin et al., 2022). Predictive modelling in this domain can enhance operational efficiency, optimise resource allocation, and increase agricultural productivity. Widely utilised machine learning models in predictive tasks within aquaculture include SVM and Random Forests RF.

i. Support Vector Machine

SVM has emerged as a prominent area of research within the machine-learning community and is widely utilised across multiple domains, including prediction and pattern recognition (Sheykhmousa et al., 2020). The SVM modelling procedure consists of several essential stages. First, data preparation ensures that the input dataset is clean and analytically suitable. Next, feature scaling is applied to normalise variable ranges and stabilise the optimisation process. The appropriate SVM type and kernel function are then selected according to the characteristics of the learning task. Following this, parameter tuning is performed to optimise key hyperparameters and improve model generalisation. Finally, the configured settings are used to train the model, resulting in an effective decision function for the target problem. As shown in Figure 2.3.

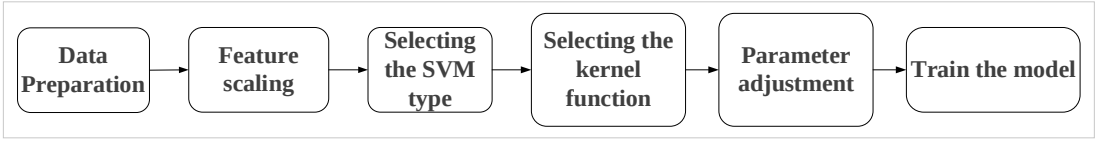

Figure 2.3. Flowchart of SVM model (S. Agarwal, 2020)

The training of a Support Vector Machine (SVM) involves iteratively adjusting the model's weight vector to maximise the separation margin between classes. The following pseudocode outlines a simplified version of the SVM training procedure, illustrating the iterative update process of the weight parameters. As shown in Table 2.4.

Table 2.4

SVM pseudocode (Harimoorthy & Thangavelu, 2021)

Algorithm 2.1: Pseudo code for the SVM algorithm

Input:

D = [X, Y]; X (array of input with m features), Y (array of labels)

SVM_task_type ∈ {"classification", "regression"}

kernel_type ∈ {"linear", "rbf", "polynomial"}

param_C

number_of_runs

Y = array(C)

Output: system_performance

split X into X_train, X_test; split Y into Y_train, Y_test

initialize Scaler

Fit Scaler on X_train

X_train_scaled = transform X_train using Scaler

X_test_scaled = transform X_test using Scaler

function train_svm(X_train_scaled, Y_train, kernel_type, param_C, number_of_runs)

initialize:

learning_rate = Math.random();

w = random_initial_vector()

λ = 1/param_C

for run in 1 to number_of_runs

for i in X_train_scaled

Table 2.4 continued

Algorithm 2.1: Pseudo code for the SVM algorithm

```
        kernel_score = kernel(kernel_type, X_train_scaled[i], X_train_scaled) · w
          if (Y_train[i] * kernel_score) < 1 then
          update: w = w + learning_rate * (Y_train[i] * kernel(kernel_type,
X_train_scaled[i], X_train_scaled) - λ * w)
        else
          update: w = w - learning_rate * λ * w
        end if
      end
    end
    return w
end
trained_w = train_svm(X_train_scaled, Y_train, kernel_type, param_C,
number_of_runs)
Y_pred = kernel(kernel_type, X_test_scaled, X_train_scaled) · trained_w
if SVM_task_type == "classification"
    system_performance = accuracy(Y_test, Y_pred)
else
    system_performance = mean_squared_error(Y_test, Y_pred)
end if
```

Leong et al. (2021) employed Support Vector Machine (SVM) and Least Squares Support Vector Machine (LS-SVM) models to directly estimate the Water Quality Index (WQI) from physical data, circumventing the sub-index calculation. The SVM model with a polynomial kernel function achieved the highest performance, with an R^2 of 0.8796. Minimising the number of extraneous input variables improved model accuracy and reduced the computational complexity of the LS-SVM model. enhance precision and diminish computational complexity. The LS-SVM exhibited superior accuracy compared to the SVM (R^2 = 0.9227). Çakir et al. (2023) employed Support

Vector Machine (SVM), Naive Bayes (NB), Random Forests (RF), and K-Nearest Neighbours (KNN) algorithms to identify water characteristics associated with bacterial infections. SVM and RFerns demonstrated 100% accuracy, surpassing KNN and NB, which achieved 91.3% accuracy. The efficacy of RFerns, comparable to that of SVM, is a significant discovery. Li et al. (2022) employed Support Vector Machine (SVM), Backpropagation Neural Network (BPNN), Radial Basis Function Neural Network (RBFNN), and Least Squares Support Vector Machine (LS-SVM) to forecast dissolved oxygen (DO), pH, ammonia nitrogen (NH3-N), nitrate nitrogen (NO3-N), and nitrite nitrogen (NO2-N). The SVM demonstrated superior accuracy, achieving 99% correlation across the majority of parameters, surpassing other models. A Support Vector Machine (SVM) is recommended for forecasting water quality in industrial aquaculture systems that utilise groundwater. (2017) employed Artificial Neural Networks (ANN) and Support Vector Machines (SVM) to forecast temporal fluctuations in the Freshwater-Saltwater Interface Level (FSL) at a groundwater observatory on Jeju Island, South Korea. The models' efficacy was evaluated utilising RMSE, Mean Absolute Relative Error (MARE), and Correlation Coefficient (CORR). The SVM model attained an average RMSE of 0.040, MARE of 6.865%, and CORR of 0.949 for the direct prediction of the upper FSL. Although SVM exhibits superior accuracy and stability, it has limitations, such as computational complexity and difficulty in selecting appropriate parameters. Enayatollahi (2024) applied a Support Vector Machine (SVM) model to predict ammonia nitrogen (NH_3-N) in petroleum wastewater using 26 weekly samples. The model, built with a Gaussian kernel, achieved high accuracy, showing that SVM is an effective tool for reliable ammonia nitrogen prediction in wastewater treatment. Table 2.5 delineates the research domains

of the aforementioned studies, the models employed, the data utilised, and the evaluations conducted.

Table 2.5

Support vector machine

Author	Method	Finding	Measurement	Dataset
Leong et al. (2021)	SVM, LS-SVM	Multinomial kernel function works best for SVM models, and reducing irrelevant input variables improves model accuracy and reduces computational complexity	LS-SVM: R^2: 0.9227 SVM: R^2: 0.8796	Ministry of Environment, Malaysia, 780 samples from 48 monitoring stations in the Perak River basin from 2009-2014
Çakir et al. (2023)	RFerns, NB, SVM, KNN	SVM and RFerns classifiers produced accurate results (100% for both) during the testing phase, while KNN and NB classifiers achieved lower accuracy (91.3% accuracy for both). SVM and RFerns performed better than KNN and NB in both the training and testing phases.	Accuracy (100% for SVM and RFerns, 91.3% for KNN and NB)	3 different farms in two-month periods for 1 year
Li et al. (2022)	SVM, BPNN, RBFNN, LS-SVM	SVM has the highest accuracy	SVM 的 R^2= 0.99	Includes water quality parameters such as DO, pH, NH_3-N, NO_3-N and NO_2-N

Table 2.5 continued

Author	Method	Finding	Measurement	Dataset
Yoon et al.,(2017)	ANN, SVM	The SVM-based time series model for FSL prediction is more accurate and stable than the ANN at the study site. The T-R-G-F type ANN model was best for upper FSL prediction, and the T-R-G-F type SVM model was best for lower FSL prediction.	Average RMSE of 0.040, MARE of 6.865%, and CORR of 0.949	Time series data from a groundwater observatory on Jeju Island, South Korea
Enayatollahi et al. (2025)	SVM	The SVM model accurately predicted ammonia nitrogen (NH_3-N) concentration in petroleum wastewater	R^2 = 0.91 and RMSE = 0.27	26 weekly samples of petroleum wastewater collected from an industrial treatment facility

In conclusion, the aforementioned research has substantially advanced the prediction of aquaculture water quality using the SVM approach, empirically validating the model's effectiveness in forecasting optimal water quality. Support Vector Machines (SVM) are adept at handling linear and mildly nonlinear datasets. Still, for extremely nonlinear or complicated datasets, more advanced kernel functions or other models, such as neural networks, may be necessary to enhance performance. Multiple factors may account for these findings. Initially, SVMs are appropriate for smaller datasets with limited characteristics. Secondly, Support Vector Machines are more

interpretable and less computationally demanding than Artificial Neural Networks. Nonetheless, a significant limitation of SVMs is that they are computationally demanding and resource-intensive when handling large datasets, such as time-series data. The efficacy of SVM models can vary markedly with the chosen parameters, necessitating meticulous, and perhaps arduous, optimisation (Namdeo & Singh, 2021). Consequently, the subsequent part examines the literature on Random Forests as an alternative machine learning methodology.

ii. Random Forest

The Random Forest (RF) methodology, introduced by Breiman (2001), is a machine learning algorithm that consists of several decision trees, integrating Bootstrap Aggregating (Breiman, 2020) and Random subspace techniques. The training process of a Random Forest model involves several key stages that collectively enhance model diversity and predictive robustness. As illustrated in the following workflow, the procedure begins with bootstrap sampling, followed by randomised tree construction and controlled stopping criteria, ultimately yielding a robust ensemble model. This pipeline provides a structured overview of how Random Forests are generated and subsequently used for inference, as shown in Figure 2.4.

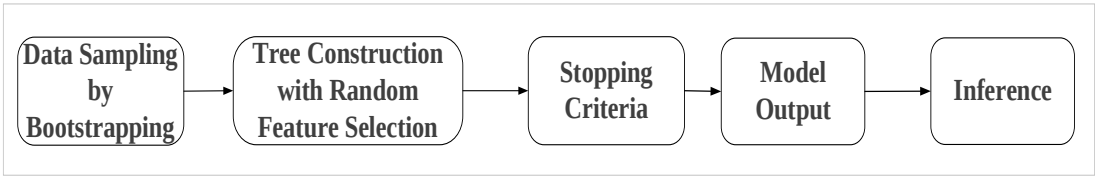

Figure 2.4. Flowchart of random forest model (Rachmawati et al., 2023)

Table 2.6 provides pseudocode for generating a classifier forest and recursively constructing each decision tree.

Table 2.6

Random forest pseudocode (H. Guo et al., 2019)

Algorithm 2.2: Pseudo code for the random forest algorithm

```
Input:  D = [X, Y] , c , x%
Output:  final_prediction
forest = []
for i from 1 to c
    D_i = Randomly sample D with replacement
    tree_i = BuildTree(D_i, x%)
    Add tree_i to forest
end for

function BuildTree(N, x%)
    if N contains instances of only one class
        return Leaf Node with that class
    end if
    F = Randomly select x% of the possible splitting features
    best_split = Find the optimal split from features F
    left_child, right_child = Split N using best_split
    left_subtree = BuildTree(left_child, x%)
    right_subtree = BuildTree(right_child, x%)
    return Internal Node with best_split, left_subtree, right_subtree
end function

trained_forest = forest
function Predict(trained_forest, X_new)
    predictions = []
    for each tree in trained_forest
```

Table 2.6 continued

Algorithm 2.2: Pseudo code for the random forest algorithm

```
        tree_pred = Traverse tree to predict label for X_new
        Add tree_pred to predictions
    end for

    if Task is Classification
        final_pred = Majority vote from predictions
    else if Task is Regression
        final_pred = Mean value of predictions
    end if

    return final_pred
end function
final_prediction = Predict(trained_forest, X_new)
```

Previous research on RF has shown its effectiveness in numerous predictive tasks. Boateng et al. (2020) surveyed 68 research publications and textbooks published between 2000 and 2017, examining KNN, SVM, RF, and Neural Networks (NN). The comparison of these algorithms focused on robustness, sensitivity, data fitting, stability, data management, noise sensitivity, parameter adjustment, and accuracy. Results demonstrate that RF exhibits overfitting and instability, KNN is inefficient with large datasets, and NN requires extensive tuning, whereas SVM and RF are favoured for their simplicity and precision. Ilkanat (2020) assessed the predictive capacity of the RF algorithm for COVID-19 case numbers across 190 countries, utilising data from January 23, 2020, to June 17, 2020. The RF model achieved high accuracy, with R^2 values ranging from 0.843 to 0.995 for testing and from 0.690 to 0.968 for estimating,

after partitioning the data into training, testing, and estimation subsets. The findings illustrate the efficacy of RF in predicting imminent cases during swift epidemics.

In aquaculture applications, RF has been used to predict fish growth, assess water quality parameters, and optimise feeding strategies. For instance, Rahman et al. (2021) formulate forecasts based on Machine Learning (ML) utilising meteorological data. The RF model is trained on two decades of data. An ensemble method attains R^2 values of 0.75, 0.81, and 0.55 for marine, freshwater, and brackish water, respectively, surpassing the performance of individual models. Palaiokostas (2021) assesses the efficacy of machine learning models for predicting disease resistance using both simulated and real datasets. Models such as random forests, adaptive boosting, and extreme gradient boosting (XGB) are compared with genomic best linear unbiased prediction. XGB surpasses RF while necessitating fewer computational resources, indicating the significance of machine learning in aquaculture breeding for disease resistance. Deng et al., (2025) used a Random Forest regression model to predict ammonia nitrogen (NH_3-N) levels in the Yangtze River Basin. The model achieved good accuracy and revealed clear spatial patterns of NH_3-N influenced mainly by human and hydrological factors, showing Random Forest's effectiveness for large-scale water quality prediction. Table 2.7 delineates the research domains of the aforementioned studies, the models employed, the data utilised, and the evaluations conducted.

Table 2.7

Random forest

Author	Method	Finding	Measurement	Dataset
Ilkanat (2020)	Random Forest	Using the Random Forest algorithm, the study forecasts near-term COVID-19 cases for 190 countries and verifies the model's good predictive performance.	Average R^2 = 0.959, Mean RMSE = 259.38	Number of confirmed cases between 23/01/2020 - 17/06/2020
Boateng et al. (2020)	KNN, SVM, RF, NN	Indicate RF overfits and is unstable, KNN is slow with large data, NN requires complex tuning, while SVM and RF are preferred for their ease and accuracy	ANN accuracy 77%, SVM with 68%, RF with 62%	68 research articles and textbooks from 2000 to 2017
Palaiokostas (2021)	DT, SVM, RF, XGB	The study benchmarks various ML models against GBLUP for predicting disease resistance in aquaculture species. XGB had the highest prediction efficiency, slightly outperforming GBLUP-MCMC.	XGB: ~10 minutes RF: ~3 minutes SVM: ~30 minutes	1,255 carp juveniles genotyped for 15,615 SNPs
Deng et al. (2025)	RF	The model accurately predicted ammonia nitrogen (NH_3-N) concentration in the Yangtze River Basin. NH_3-N showed clear spatial clustering, mainly influenced by human and hydrological factors	R^2 = 0.61, RMSE = 0.13	Yangtze River Basin Water quality monitoring data

In summary, the aforementioned works have significantly contributed to the traditional machine learning literature by empirically validating the efficacy of the random forest model for addressing various tasks in aquaculture production (classification and regression) with limited datasets. The results of these investigations demonstrate that Random Forests are generally constructed for static datasets in which the sequence of data points is irrelevant (Jackson & Adam, 2021; Michael, 2016; Sheykhmousa et al., 2020). RF presumes that the data are independently and identically distributed (Huan et al., 2021). Random forests do not inherently account for the temporal ordering of data points (Huang et al., 2020). They analyse each data point in isolation, perhaps neglecting significant temporal correlations. Consequently, when managing time series data, various concerns and obstacles may arise, including insufficient temporal awareness and non-stationarity.

2.3.2.3 Artificial Neural Networks

Artificial Neural Networks are employed in artificial intelligence and machine learning. These models are designed to autonomously learn and extract intricate patterns and representations from data via a hierarchical arrangement of numerous layers, allowing them to identify complicated and abstract relationships within the input data (Najafabadi et al., 2015). Deep learning models are a category of artificial neural network learning methods based on the principles of the biological neural brain. Chen et al. (2020) asserted that deep learning models are a fundamental component of machine learning. Classic instances of artificial neural networks (ANNs) employed in conventional machine learning include simple feedforward neural networks, such as the single-layer perceptron, which consist solely of input and output layers. A study by Pham et al. (2021) found that Basic artificial neural networks have been used for

pattern recognition and linear regression problems. Numerous studies have established annotation networks to elucidate complex links within data. This procedure ensures that the models are both theoretically valid and reliable, as well as useful in practical applications, by integrating multiple processes. Figure 2.5 depicts the ANN training procedure.

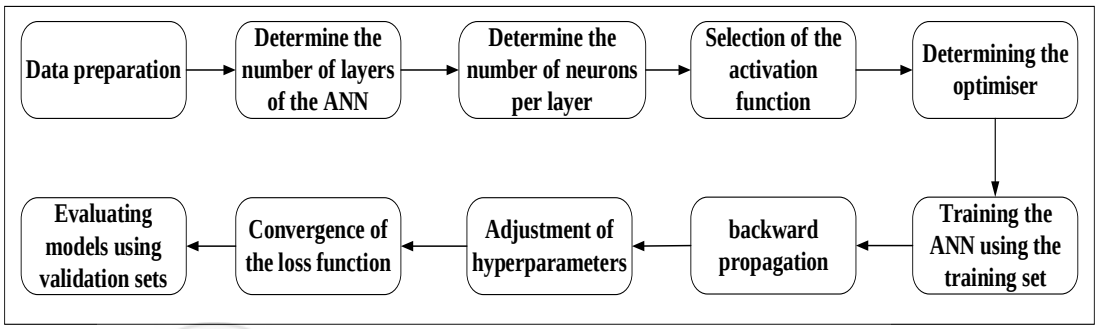

Figure 2.5. Flowchart of ANN model(H. Yu et al., 2021)

Kadam et al. (2019) use ANN and utilise Multiple Linear Regression (MLR) techniques in the Western Ghats, specifically the Shivganga River Basin in India, to assess the suitability of groundwater quality for drinking purposes. The results confirm that performance remains consistently acceptable across both seasons and that the ANN model's forecasts are satisfactory. The proposed ANN model may be beneficial for related studies aimed at forecasting groundwater quality for human consumption. Noori et al. (2020) developed a hybrid model by integrating the proposed process-based watershed model with an artificial neural network (ANN). The two models collaborate to optimise calibration and validation, and the hybrid model demonstrates strong predictive performance for nitrate, ammonium, and phosphate. It surpasses the SWAT model calibrated at each location. Jagannath et al. (2018) propose a practical and effective Automatic Modulation Classification (AMC) system based on Artificial

Neural Networks (ANNs). The system's primary advantages include robust performance across various SNRs, minimal computational complexity for straightforward implementation, and adaptability for future expansion of the modulation format dictionary, demonstrating its appropriateness for real-time commercial applications. Ibrahim et al. (2023) examined potential pollution sources and significant parameters contributing to geofigure variability and used PCA and ANN to identify optimal inputs for water quality modelling. Table 2.8 summarises the research domains of the aforementioned studies, the models employed, the data cycles, and the evaluations.

Table 2.8

Artificial neural network

Author	Research Area	Method	Data cycle	Evaluation metric
Kadam et al., (2019)	Predicting water quality	ANN, MLR	Years	MAE, and RMSE
Noori et al. (2020)	Predicting water quality	Process-based models, ANN	Monthly	Coefficient of determination (R^2)
Jagannath et al. (2018)	Automatic Modulation Classification (AMC) system	Nadam, ANN	Years	Accuracy
Aminu Ibrahim et al. (2023)	Predicting water quality	PCA, ANN	Years	Root Mean Square Error (RMSE)

The experiments highlighted in the paragraph underscore the substantial contributions of artificial neural networks to machine learning literature, demonstrating their versatility and efficacy. Nonetheless, it is observed that artificial neural networks are

not intrinsically equipped to manage temporal dependencies and variable-length sequences in time series data, such as ammonia nitrogen concentrations in aquaculture. Artificial Neural Networks (ANNs) are limited by their inability to accurately capture temporal dependencies and patterns in sequential data (X. Zhou et al., 2024). The text indicates that artificial neural networks may not be optimally suited for time-series forecasting across multiple domains.

i. Convolutional Neural Network

CNN is a deep learning architecture primarily utilised for analysing structured grid data, including photos and movies (Bhatt et al., 2021). Conclusive evidence demonstrates that CNNs transform multiple domains, especially in computer vision, by autonomously learning and extracting hierarchical features. These networks include layers such as convolutional layers that perform convolution with learnable filters to detect features such as edges and textures (Bhatt et al., 2021). Pooling layers, using procedures such as max pooling, downsample feature maps, thereby efficiently reducing spatial dimensions (Gholamalinezhad & Khosravi, 2020).

Figure 2.6 illustrates the methodologies for CNN model prediction, encompassing data collection and preprocessing, CNN model design and training, model evaluation and validation, real-time model deployment, and ongoing model optimisation. This procedure integrates data processing methods with sophisticated machine learning algorithms to ensure efficient, precise forecasting.

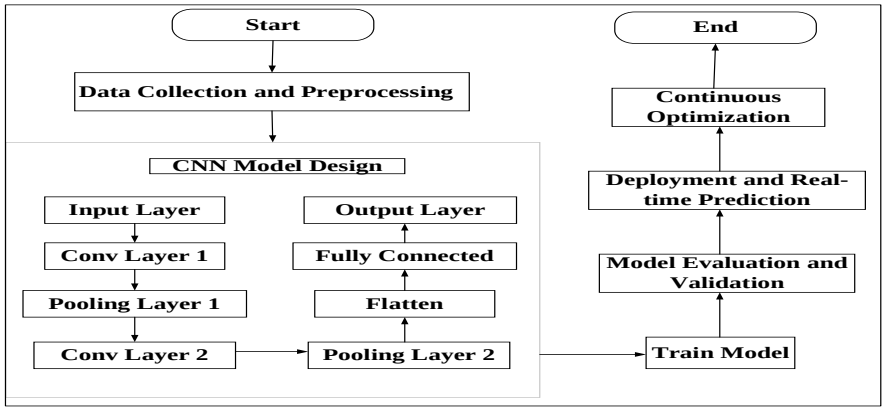

Figure 2.6. Flowchart of CNN model (M. Sun et al., 2020)

The following pseudocode outlines the training pipeline of the Parallel-CNN model, which transforms each sample into a sequence of word embeddings before constructing the corresponding feature matrices. This procedure summarises the full preprocessing and model training workflow used to evaluate the CNN on a given dataset. As shown in Table 2.9.

Table 2.9

CNN pseudocode (Vizel et al., 2015)

Algorithm 2.3: Pseudocode for the CNN model

Input: d: dataset, l: true labels, W: word embedding matrix
Output: final_result
collect and preprocess dataset d
for each sample i in d do
  v_i = embed(i, W)
  append v_i to feature matrix f
end for
split f into f_train, f_val, f_test
split l into l_train, l_val, l_test
build CNN model with convolution, pooling, and fully connected layers
train CNN model on (f_train, l_train)

Table 2.9 continued

Algorithm 2.3: Pseudocode for the CNN model
val_perf = evaluate(model, f_val, l_val)
test_perf = evaluate(model, f_test, l_test)
deep_features = extract features from the trained model
result_compare = compare features and predictions with baselines
final_result = {test_perf, result_compare}
return final_result

Numerous studies have employed CNN for predictive tasks (Ghorbanzadeh et al., 2022; Xie et al., 2022; Zhang et al., 2020). Hassan et al. (2021) employ deep CNN models for effective plant disease identification, addressing issues of high parameter counts and computational expense. Substituting ordinary convolution with depth-separable convolution results in exceptional accuracy rates: InceptionV3 (98.42%), InceptionResNetV2 (99.11%), MobileNetV2 (97.02%), and EfficientNetB0 (99.56%). In contrast to conventional approaches, these models exhibit higher accuracy and shorter training time, with MobileNetV2 designed for mobile devices. The findings indicate potential uses in real-time agricultural systems, specifically in enhancing disease detection in crops, hence improving agricultural production. Phan et al. (2019) introduced a framework that concurrently classifies a singular input epoch and forecasts labels for adjacent epochs. This approach, unlike conventional categorisation schemes, leverages the interdependence of successive sleep epochs to address associated challenges. The approach uses a singular model to generate many judgments, providing benefits akin to ensemble-of-models techniques with negligible computational burden. Probabilistic aggregation methods are employed to leverage numerous decisions. Investigations on Sleep-EDF The Expanded and Montreal Archive of Sleep Studies datasets illustrate the framework's efficacy, attaining

classification accuracies of 82.3% and 83.6%, respectively. The suggested framework surpasses baseline methods and current deep-learning techniques, demonstrating its superiority and leading the investigation of multitask neural networks for autonomous sleep staging. This research establishes a basis for subsequent investigations into various neural network topologies within the domain.

Jing et al. (2022) proposed the CNN-RWP model, which stands for CNN-based regional wave prediction. A CNN in an hourglass shape generates the mapping relationship between wind and wave data. The wind datasets from the European Centre for Medium-Range Weather Forecasts and the validated wave datasets simulated by SWAN were used for model training, optimisation, verification, and generalisation analysis. The Convolutional Neural Network with Random Weight Projection (CNN-RWP) and Structured What-If Analysis (SWAN) models are evaluated on a dataset from the Gulf of Mexico. The average absolute inaccuracy of the SWAN output and the CNN-RWP model prediction is below 10%, while computational efficiency has improved by around 1000 times. The proposed CNN-RWP model provides an effective approach for achieving highly accurate wave predictions. Tamut et al. (2025) applied a CNN to detect fish diseases in freshwater aquaculture automatically. Using 2,444 images across seven disease categories, the CNN achieved 99.71% accuracy and a test loss of 0.0119, demonstrating high efficiency and robustness. The study shows that CNNs can effectively classify bacterial, fungal, and parasitic infections, providing a rapid, reliable tool for disease monitoring and management in aquaculture. Table 2.10 below encapsulates the training methodologies, results, metrics, and datasets of the aforementioned paper's extensive CNN model.

Table 2.10

Convolutional neural network

Author	Method	Finding	Measurement	Dataset
Hassan et al. (2021)	Implemented models	CNNs promise efficient crop disease identification, which is vital for real-time agriculture.	Accuracy rates of 98.42%, 99.11%, 97.02%, 99.56%	38 plant diseases, 14 plant species, and healthy plant leaves
Phan et al. (2019)	CNN	Framework surpasses baselines, deep learning.	Accuracies of 82.3% and 83.6%	Archive of Sleep Studies datasets
Jing et al. (2022)	CNN-RWP model	Offers an effective and efficient solution for achieving high-precision regional wave prediction	MAE of less than 10%	Wind datasets from the European
Tamut et al. (2025)	CNN	CNN accurately detected seven freshwater fish diseases, demonstrating high reliability in automated diagnosis of aquaculture diseases.	99.71% accuracy and a test loss of 0.0119	2,444 labelled fish images across seven classes collected from freshwater aquaculture environments

In certain cases, employing a CNN for time series forecasting may yield a complex model that is difficult to train and deploy effectively. Less complex models, such as autoregressive models or conventional statistical techniques, may be more appropriate for some time series applications. CNNs are specifically engineered for structured grid data, such as photographs, and they demonstrate superior performance in computer vision applications. Although CNNs can be used for time series data, they face several hurdles in time series prediction, notably the need for a fixed input size. This poses

difficulties when addressing time-series data of varying lengths, such as ammonia nitrogen levels. Altering the dimensions of the data may result in information loss or heightened processing complexity.

ii. A Recurrent Neural Network

RNN is a type of artificial neural network used in machine learning for processing sequential input (Wang et al., 2021). It is extensively utilised across several fields, including natural language processing (Xie et al., 2022), speech recognition (Cui et al., 2018), and time series analysis (Li et al., 2019). One feature of RNNs is their capacity to analyse sequential data of varying lengths and to capture temporal dependencies. RNNs, unlike traditional feedforward neural networks, possess feedback connections that enable them to retain information from previous inputs in their hidden states (Sherstinsky, 2020). Numerous RNN architectures and variants have been developed to address various challenges and improve the efficacy of conventional RNNs. Nevertheless, it suffers from gradient vanishing and long-term dependencies (Noh, 2021). LSTM and GRU are proposed as enhanced variants of RNN to address these issues.

a) Gated Recurrent Unit

The GRU, developed by Cho et al. (2014), is a sophisticated variant of an RNN that addresses the vanishing gradient problem by employing gating mechanisms similar to those in LSTM networks. Gated Recurrent Units (GRUs) manage information flow through two gates: the update gate and the reset gate, governed by sigmoid functions and the tanh activation function to maintain hidden state values within the range of -1 to 1 (Althobaiti et al., 2021). The GRU integrates the previous state with the new

candidate state to encapsulate dependencies across many time scales (Lee et al., 2022). Developing a GRU model entails data acquisition, preprocessing, cleansing, normalisation, network architecture design with suitable layers and activation functions, and training that includes optimiser selection (e.g., Adam or SGD) and loss function specification (e.g., mean squared error), followed by validation and testing for accuracy and generalizability before implementing real-time monitoring and prediction. The procedures are illustrated in Figure 2.7.

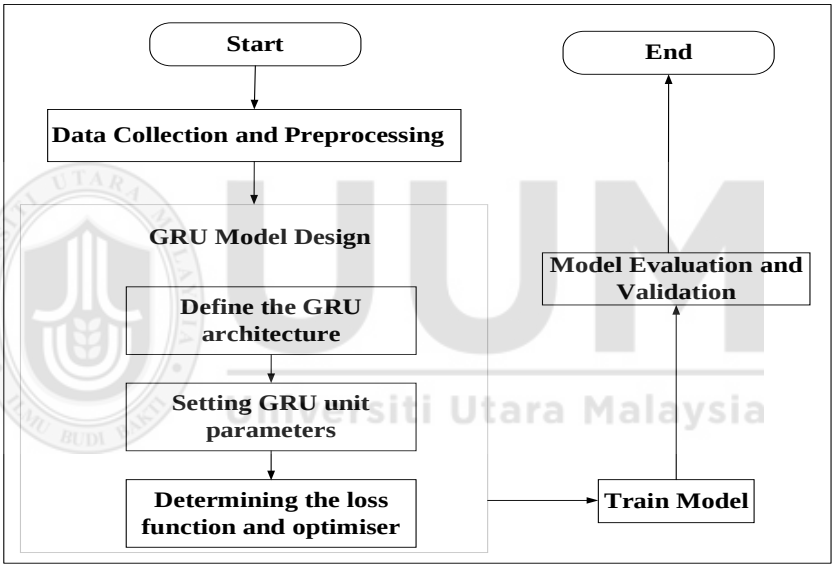

Figure 2.7. Flowchart of GRU model (Li et al., 2021)

To illustrate the GRU's computation process, we provide pseudocode outlining its forward propagation procedure. This pseudocode clearly describes how the update gate, reset gate, and candidate hidden state are computed at each time step, as shown in Table 2.11.

Table 2.11

GRU pseudocode (Ni et al., 2024)

Algorithm 2.4: Pseudocode for the GRU model

```
Input: d: dataset, l: true labels, W: word embedding matrix
Output: model_perf
collect d
preprocess d
f = sequence matrix
for i in d
   append vectorize(i, W) to f
end for
split f into f_train, f_val, f_test; split l into l_train, l_val, l_test

function GRU_Step(x_t, h_prev)
   r_t = sigmoid(weight_r*[h_prev,x_t]+bias_r)
   z_t = sigmoid(weight_z*[h_prev,x_t]+bias_z)
h_tilde = tanh(weight*[r_t*h_prev,x_t]+bias)
h_t = (1-z_t)*h_prev + z_t*h_tilde
   return h_t
end function

model = Sequential()
model.add(GRU(), Dense())

compile model
train model on f_train, l_train
trained_model = model
val_pred = predict trained_model on f_val
test_pred = predict trained_model on f_test
model_perf = {evaluate(val_pred, l_val), evaluate(test_pred, l_test)}
return model_perf
```

Arepalli et al. (2022) examine the shortcomings of current methods that concentrate on one or two water quality criteria. The research used a GRU algorithm to forecast water-quality conditions affecting salmon, accounting for various factors. The findings demonstrate that the proposed GRU algorithm surpasses current ANN and LSTM algorithms, yielding more precise predictions for efficient aquaculture. Arepalli and Naik (2023) emphasise the essential role of water quality monitoring in aquaculture to protect the health and production of aquatic species. The work presents a smart monitoring system that employs IoT devices and provides a Water Toxic Index (WTI) to classify water quality as either polluted or non-contaminated. The authors introduce an improved lightweight Multi-Headed Gated Recurrent Unit (MHGRU) model to address the constraints of early deep learning models. The results demonstrate that the proposed model exceeds the performance of current models, with an exceptional 99.7% accuracy on real-time data and 99.12% on a public dataset, outstripping the highest-performing existing ANN models.

Li et al. (2021) employed deep learning models often utilised for time series forecasting, including GRU, LSTM, and RNN. Dissolved oxygen, an essential water quality parameter in aquaculture, significantly influences aquatic production yield and the survival rate of marine species. The objective of forecasting dissolved oxygen levels in fisheries ponds is to facilitate the implementation of cost-effective and energy-efficient artificial aeration. To choose the most effective model for fisheries ponds, generate forecasts. The research develops three DO prediction models utilising RNN, LSTM, and GRU methodologies. A performance comparison is executed utilising four assessment metrics: Mean Absolute Error (MAE), Mean Square Error (MSE), Mean Absolute Percentage Error (MAPE), and the Coefficient of

Determination (R^2). GRU exhibits 0.450 mg/L (MAE), 0.411 (MSE), 0.054 (MAPE), and 0.994 (R^2). The LSTM indicates are 0.407 mg/L, 0.294, 0.059, and 0.970, respectively, demonstrating comparable performance; however, the GRU requires less time and uses fewer parameters, rendering it more efficient.

Lee et al. (2022) primarily focus on predicting short-term fluctuations in dissolved oxygen levels near the confluence of the Han River in Anyangcheon, an area prone to recurrent water-quality issues. Two separate models were developed: a classification model using decision trees, random forests, and Extreme Gradient Boosting (XGBoost), and a real-time prediction model employing Artificial Neural Networks (ANN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) algorithms. The categorisation model identified key parameters affecting dissolved oxygen variation, including electrical conductivity, cumulative precipitation, total nitrogen, and water temperature. Random forest and XGBoost exhibited exceptional performance, attaining sensitivities of 0.9962/1.0000 and accuracies of 0.9981/0.9822, respectively. In real-time short-term DO prediction, the LSTM and GRU models greatly surpassed the ANN. LSTM produced an R^2 between 0.93 and 0.97 in the first half and between 0.95 and 0.96 in the second half, whereas GRU demonstrated an R^2 from 0.94 to 0.98 in the first half and from 0.96 to 0.98 in the second half. These models, utilizing real-time automatic measurement data in urban rivers, have shown efficacy in swiftly detecting and modelling short-term, real-time fluctuations in water quality, exceeding the performance of current static models.

Airlangga et al., (2025) applied a GRU network to predict key water quality parameters, including ammonia nitrogen, in aquaponics systems. The GRU model achieved high

predictive accuracy with a mean absolute error (MAE) of 0.0478 and effectively captured temporal dependencies in noisy time-series data. The study confirmed the suitability of GRU for real-time ammonia nitrogen forecasting and water quality management in aquaponics. Table 2.12 encapsulates the methodologies, results, quantified metrics, and datasets of the aforementioned articles.

Table 2.12

Gated recurrent unit

Author	Method	Finding	Measurement	Dataset
Arepalli et al. (2022)	GRU, ANN, LSTM	Indicate that the proposed GRU algorithm outperforms existing artificial neural network (ANN)	Root Mean Squared Error (RMSE)	Nitrate, pH, BOD, DO, and temperature
Arepalli & Naik (2023)	Using the cutting-edge MHGRU model and Internet of Things devices	Identify crucial features for classifying water toxicity, enhancing water quality management.	99.52% and 98.71% accuracy	pH, DO, temperature, ammonia & nitrate analyzed
Li et al. (2021)	The two most popular linear modelling techniques were multiple linear regression (MLR) and ARMA.	Multiple linear regression (MLR) and ARMA were the most widely used linear modelling techniques.	LSTM MAE=0.407, MSE=0.294, MAPE= 0.059, $R^2 = 0.970$	Three fish ponds in Guntur, India: 74,759 records

Table 2.12 continued

Author	Method	Finding	Measurement	Dataset
Airlangga Nugroho et al. (2025)	GRU	The GRU model accurately predicted water quality parameters, including ammonia, with strong temporal learning ability, demonstrating its suitability for real-time aquaponic monitoring.	MAE = 0.0478	Time-series water quality data from aquaponics systems
Lee et al. (2022)	Classification model using decision tree, random forest, and XGBoost. ANN, LSTM, and GRU algorithms	RF and XGBoost showed excellent classification compared to current static models, it swiftly represents and replicates transient, real-time variations in water quality	Random forest Performance sensitivity of 0.9962/1.0000 XGBoost Accuracy of 0.9981/0.9822 LSTM achieved an R^2 ⁼0.93–0.97 in the first half and 0.95–0.96 in the second half, while GRU R^2=0.94–0.98 0.96–0.98	Data from the Han River and urban rivers

In conclusion, GRU is widely used for analysing time series data. In situations with small datasets or limited processing resources, GRUs exhibit greater stability due to their simpler architecture and reduced susceptibility to overfitting. In contrast to LSTM,

GRU has a simpler architecture and does not include independent memory units (Shiri et al., 2023). Nonetheless, in the context of large datasets or complex tasks, LSTM may be more appropriate due to its ability to capture long-term relationships in the data with greater precision.

b) Long-Short-Term Memory

LSTMs have a more intricate recursive connection architecture that effectively captures long-term temporal dependencies. This makes them suitable for tasks such as ammonia time-series prediction, offering a potential remedy for the difficulties conventional artificial neural networks encounter in handling temporal data (Sherstinsky, 2020). The LSTM is a recurrent neural network architecture specifically engineered to mitigate the vanishing gradient problem inherent in conventional RNNs (Mienye et al., 2024). It is furnished with a series of gates, comprising forget gates, input gates, and output gates, which employ sigmoid and tanh activation functions to manage the flow of information (Staudemeyer & Morris, 2019). The LSTM unit comprises two states: the cell state, which stores long-term memory, and the hidden state, which stores short-term memory. The cell state is revised using a combination of the prior cell state, the preceding hidden state, and the current input, whereas the hidden state is derived from the updated cell state and the output gate (Mienye et al., 2024). This architecture enables the LSTM to judiciously determine which information to retain, alter, or generate at each stage of the sequence, making it highly effective for tasks involving sequential data (Sahin & Kozat, 2019). LSTM has achieved widespread adoption owing to its efficacy and versatility in handling sequential data, including time series analysis, speech recognition, and natural language processing (Oruh et al., 2022). Its adaptability enables customisation via architectural

modifications and precise hyperparameter adjustments to accommodate diverse tasks (Greff et al., 2017). LSTMs have emerged as a crucial instrument in deep learning, demonstrating efficacy in sequence modelling and predictive tasks such as ammonia level predictions (Ergen & Kozat, 2018). This method is especially effective for analysing and forecasting intricate data with temporal dependencies. The procedure is illustrated in Figure 2.8.

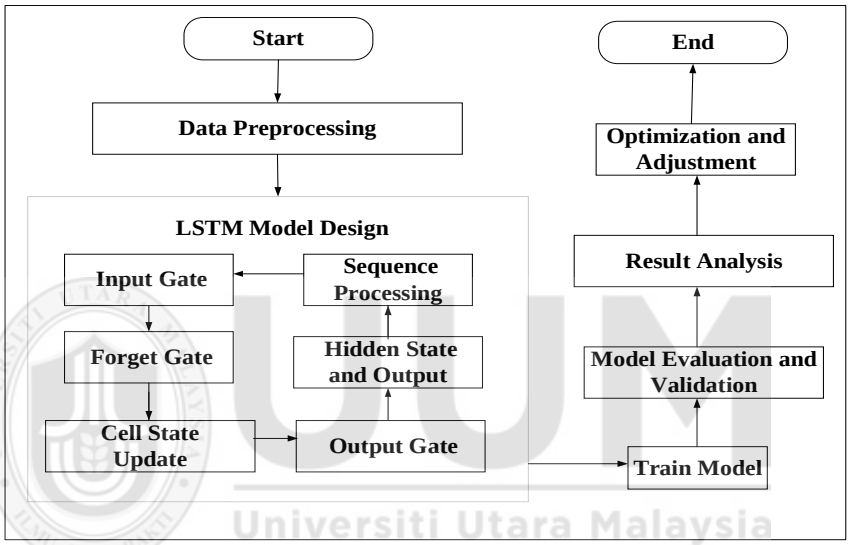

Figure 2.8. Flowchart of LSTM model

LSTM networks are designed to capture long-range dependencies in sequential data through gated mechanisms that regulate information flow across time steps. The following pseudocode outlines the core computational steps of a standard LSTM cell, including its input processing, hidden-state updates, and parameterised gating operations, as shown in Table 2.13.

Table 2.13

LSTM pseudocode (Nakisa et al., 2018)

Algorithm 2.5: Pseudocode for the LSTM model

```
Input: d: dataset, l: labels
Output: final_result
// Data Preprocessing
X = process d to sequence
split X into X_train, X_val; split l into l_train, l_val
// LSTM Model Design
function LSTM_Step(x_t, h_prev, c_prev)
    f = sigmoid(Wf*x_t + Uf*h_prev + bf)
    i = sigmoid(Wi*x_t + Ui*h_prev + bi)
    c_tilde = tanh(Wc*x_t + Uc*h_prev + bc)
    c = f*c_prev + i*c_tilde
    o = sigmoid(Wo*x_t + Uo*h_prev + bo)
    h = o*tanh(c)
    return h, c
end function
// Train Model
model = Sequential()
model.add(LSTM())
compile model
train model on X_train, l_train
// Model Evaluation and Validation
val_perf = evaluate(model, X_val, l_val)
// Result Analysis & optimisation and Adjustment
analyze val_perf
adjust model params

final_result = val_perf
    return final_result
```

Han et al. (2019) concentrate on forecasting bus passenger flow for public transportation management and the development of smart cities. It presents an optimised LSTM with the Nadam and SGD algorithms. The suggested hybrid LSTM model markedly surpasses non-hybrid LSTM models and traditional approaches, with performance enhancements of 4% to 20%. The study examines parameter sensitivity and illustrates the model's robustness and efficacy, providing significant insights for dynamic bus scheduling and regional coordination scheduling. Baek et al. (2020) devised an innovative approach for forecasting water levels and quality by integrating CNN and LSTM deep learning models. The methodology was implemented in the Nakdong River Basin in South Korea, concentrating on modelling water levels alongside total nitrogen, total phosphorus, and total organic carbon concentrations. The study demonstrated that both the CNN (water level) and LSTM (water quality) models exhibited commendable performance, indicating high accuracy and reliability in their predictions. The research is crucial for efficient water management and sustainable development plans.

Javed et al. (2022) examine the challenges of Short-Term Load Forecasting (STLF), emphasising the overfitting problems associated with complex machine learning and deep learning models. It introduces an innovative two-stage Encoder-Decoder (ED) architecture that integrates Short Receptive Field-based Dilated Causal Convolutional (SRDCC) networks with Bi-directional Long Short-Term Memory (BiLSTM) networks. Assessment against current models reveals a 35% enhancement in precision. The suggested architecture effectively captures localised load patterns while maintaining reasonable computational complexity. Awoke et al. (2021) utilise deep learning prediction models, specifically LSTM and GRU, to predict Bitcoin's price

volatility with high accuracy. Their LSTM-GRU methodology achieved notable accuracy compared to prior methods.

Sharma et al. (2022) present an innovative long-term solar photovoltaic (PV) power forecasting approach that employs an LSTM model with the Nadam optimiser. Their method exhibits notable superiority, demonstrating a 30.56% enhancement compared to the autoregressive integrated moving average, a 47.48% improvement over the seasonal autoregressive combined moving average, and considerable advances over various neural network models. This technique significantly improves the precision of solar PV power forecasting, offering valuable benefits for system planning and management.

In a related study, Zhu et al. (2023) focus on improving the accuracy of SF6 gas pressure prediction in complex power grid settings. The authors advocate a neural network model based on APSO-LSTM and conduct simulation experiments using three months of recorded SF6 gas pressure data for training and testing. The outcomes reveal that this model effectively diminishes the prediction error of SF6 gas pressure, providing valuable insights to inform strategies for reducing emissions in the power industry. Chen et al. (2022) aim to improve the prediction of cardiovascular disease. It suggests a cardiovascular disease prediction model based on R-Lookahead-LSTM, presents three new feature vectors for disease prediction and selects features using the random forest algorithm. To encourage model convergence, this model replaces the Tanh activation function with the soft sign function and optimises the stochastic gradient descent algorithm used by the Lookahead algorithm with the Rectified Adam algorithm. The R-Lookahead algorithm further optimises the long-term memory

network model, improving stability and potentially enabling its use in predicting cardiovascular disease.

Hossain et al. (2020) describe the development of an Intrusion Detection System (IDS) to secure in-car Controller Area Network (CAN) bus communications using LSTM neural networks. This IDS aims to identify and prevent potential cyberattacks on the CAN bus, an essential part of modern cars for internal communication. The study most likely assesses how well an LSTM performs at detecting unusual patterns indicative of security breaches, thereby enhancing automotive systems' cybersecurity. Fang et al. (2025) used an LSTM model to predict water quality indicators, including ammonia, across multiple river basins. The model achieved R^2 values between 0.59 and 0.85, showing strong predictive performance and stability across varying hydrological conditions. The study confirmed LSTM's reliability and high confidence in modelling temporal variations in ammonia nitrogen for water quality management. Table 2.14 summarises the key elements of the papers above.

Table 2.14

Some reviewed literature on long-short term memory

Author	Area of Application	Optimiser used	Measurement
Han et al. (2019)	Public transport system and the smart city 5 station	Hybrid Nadam-SGD-LSTM	MAE=24.628,18.133, 14.913,14.978, 15.309 MAPE =26.751, 32.508,32.190,41.586 RMSE= 20.482

Table 2.14 continued

Author	Area of Application	Optimiser used	Measurement
Baek et al. (2020)	Water quality	SGD-LSTM	Nash–Sutcliffe Efficiency (NSE) is above 0.75
Javed et al. (2022)	Short-term load forecasting	Adam- SRDCC - BiLSTM	Accuracy improved by approximately 35%
Awoke et al. (2021)	Bitcoin's price volatility	SGD-LSTM	RMSE= 0.045
Sharma et al. (2022)	Photovoltaic (PV) power forecasting	Nadam-LSTM	RMSE= 30.56% MSE= 47.48%
Zhu et al. (2023)	Gas Pressure Time Series Data	APSO-LSTM	MBE = 3.9270 and MSE = 3.80
Chen et al. (2022)	Disease Prediction	R-Lookahead-LSTM	Accuracy = 89.61%, Recall = 85.72%, F1-score = 87.16%
Hossain et al. (2020)	In-Vehicle CAN Bus Communications	LSTM-Based Intrusion	AUC-ROC
Fang & Wang (2025)	Water quality prediction in river basins	LSTM	R^2 (0.59–0.85)

These studies highlight the benefits of LSTM models in time series forecasting while also revealing notable deficiencies. This research overlooked the issue of gradient fluctuations during training. The Nadam optimiser is used alongside LSTM, yet its specific role in alleviating the gradient spike problem remains insufficiently examined. These studies overlook the use of gradient clipping techniques in the optimiser, which

are essential for stability and improved predictive performance. Consequently, the use of stochastic gradient descent algorithms alongside gradient clipping warrants further discussion.

c) Temporal Fusion Transformer

The Temporal Fusion Transformer (TFT) is a sophisticated deep learning framework tailored for multi-horizon time series forecasting, combining the advantages of transformer-based attention mechanisms with recurrent neural networks to capture both short-term and long-term temporal dependencies adeptly (Joseph et al., 2024). TFT is designed to manage complex, multidimensional datasets by using attention layers to selectively focus on relevant time steps and attributes, thereby improving its capacity to simulate intricate patterns and interactions within the data (Ayhan et al., 2024). TFT integrates mechanisms for processing both static covariates and time-varying covariates, enabling the seamless incorporation of diverse data sources prevalent in aquaculture and water quality monitoring, including environmental sensors and operational parameters (Ho & Hung, 2024). A salient characteristic of TFT is its focus on interpretability; the model elucidates which variables and temporal intervals exert the most influence on predictions via its attention weights and variable selection networks. This transparency is especially beneficial in fields such as aquaculture and water quality, where comprehending the factors influencing predictions can enhance management and decision-making (Ayhan et al., 2024). TFT's effective handling of missing data and its ability to accommodate varied input durations render it exceptionally adaptive to real-world situations where data quality and availability may vary (Zhu et al., 2023).

The TFT integrates multiple specialised components to capture both short-term temporal dynamics and long-range dependencies while maintaining strong interpretability. The following diagram illustrates the overall architecture of TFT, highlighting the sequential flow from input processing and variable selection to temporal encoding, attention-based fusion, and final prediction generation. As show in figure 2.9.

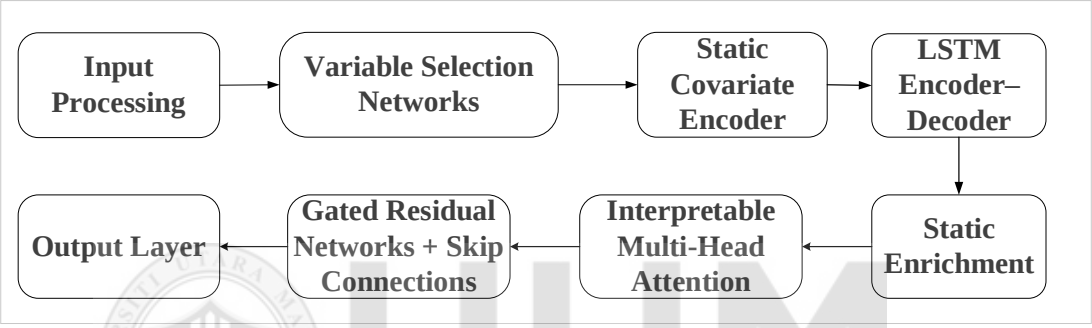

Figure 2.9. Flowchart of TFT model (Shen et al., 2025)

The TFT integrates recurrent layers, attention mechanisms, and gating components to capture both short-term temporal patterns and long-range dependencies in multivariate time series. The following pseudocode summarises the core computational steps of TFT, including variable selection, sequence encoding, temporal attention, and final prediction generation, as show in table 2.15.

Table 2.15

TFT pseudocode (B. Lim et al., 2021)

Algorithm 2.6: Pseudocode for the TFT model
Input: raw_data
Output: $\hat{Y}$_future
// Input Processing

Table 2.15 continued

Algorithm 2.6: Pseudocode for the TFT model

```
X_static, X_past, X_future = process(raw_data)

// Variable Selection Networks
x_t = PastVariableSelection(X_past)
z_t = FutureVariableSelection(X_future)

// Static Covariate Encoder
s = StaticVariableSelection(X_static)
c_s = StaticContext(s)

// LSTM Encoder-Decoder
h_past = EncoderLSTM(x_t)
h_future = DecoderLSTM(z_t, init=h_past)

// Static Enrichment
h = StaticEnrichment(h_past, h_future, c_s)

// Interpretable Multi-Head Attention
H = MultiHeadAttention(h)

// Gated Residual Networks + Skip Connections
H_res = GatedResidual(H, h)

// Output Layer
Ŷ_future = OutputLayer(H_res)

return Ŷ_future
```

TFT has evolved into a robust architecture for multi-horizon and multivariate time-series forecasting, exhibiting exceptional performance and interpretability across

diverse domains. Laborda et al. (2023) employed TFT for multi-country, multi-horizon GDP forecasting, demonstrating that TFTs outperform conventional regression models, especially during economic disruptions such as the COVID-19 shock. Their research emphasised TFT's capacity to yield interpretable outcomes and to proficiently illustrate temporal patterns, thereby augmenting the comprehension of economic dynamics. In the agricultural domain, (2023) used TFT to predict wheat yields, achieving greater accuracy than traditional machine learning and deep learning models. The interpretability provided by TFT was essential for agricultural decision-making, enabling stakeholders to understand the elements affecting crop yields.

Lim (2019) introduced TFT for multi-horizon time-series forecasting, highlighting its ability to capture temporal correlations across multiple scales. Their study indicated substantial performance improvements and provided clear insights into temporal dynamics, establishing TFT as an essential tool for complex forecasting. Ayhan et al. (2024) employed TFT for anomaly identification in multivariate aviation time-series data within the domain of aviation safety analytics. The model successfully detected deviations from standard behaviour, thus improving the dependability of aviation safety monitoring systems. Rasiya Koya and Roy (2024) utilised TFT for streamflow prediction, integrating attention mechanisms with recurrent architectures to surpass LSTM networks and conventional models. This integration improved accuracy and expanded explainability for hydrological forecasts, facilitating superior water resource management.

Temperature forecasting has also benefited from TFT's sophisticated capabilities. Jdi & Falih (2024) employed TFT to forecast temperature trends, attaining greater

accuracy than ARIMA and GRU models. This development is especially beneficial for irrigation and climate-related applications, where accurate temperature predictions are essential. Qiu et al. (2025)applied a Temporal Fusion Transformer (TFT) model to predict ammonia nitrogen (NH_3-N) concentrations in a sequencing batch reactor system for wastewater treatment. The model achieved high predictive accuracy ($R^2 > 0.95$) and significantly reduced process variability and energy consumption. The study demonstrated TFT's strong ability to capture long-term temporal dependencies in water quality forecasting and to optimise treatment efficiency. Table 2.16 encapsulates the principal components of the aforementioned papers.

Table 2.16

Some reviewed literature on temporal fusion transformer

Author & Date	Area of Application	Optimiser Used	Measurement
Laborda, Ruano, and Zamanillo (2023)	Multi-country and multi-horizon GDP forecasting	Not explicitly mentioned	TFT outperforms ARIMA in terms of MAE and RMSE; ARIMA RMSE is 188.27% higher than TFT for annual forecasts.
Junankar, Sondhi, and Nair (2023)	Wheat yield prediction for precision farming	Not specified	R^2 of TFT = 0.9705; compared to SVM (Linear: 0.3698, RBF: 0.6897), XGBoost (0.3246), and Random Forest (0.9337).
Lim et al. (2019)	Multi-horizon time series forecasting for diverse datasets	Adam optimiser	TFT achieved 7% lower P50 loss and 9% lower P90 loss than the next-best model across datasets.

Table 2.16 continued

Author & Date	Area of Application	Optimiser Used	Measurement
Ayhan, Vargo, and Tang (2024)	Anomaly detection in aviation time-series data	Not explicitly mentioned	TFT-2 (single output): Mean RMSE = 3.06; TFT-1 (multi-output): Mean RMSE = 3.18; TFT-2 (multi-output): Mean RMSE = 3.13; TFT-3 (multi-output): Mean RMSE = 3.27.
Koya and Roy (2023)	Streamflow prediction in hydrology	Not specified	Median Kling-Gupta Efficiency (KGE) Scores: TFT = 0.705; LSTM = 0.647; Transformer = 0.563.
Jdi and Falih (2024)	Temperature forecasting	Not explicitly mentioned	MAE: 1.5143; RMSE: 1.9527; MAPE: 11.6156%; R^2: 0.9359.

TFT offers numerous benefits, making it an effective tool for time-series forecasting, especially in aquaculture and water-quality assessment. Its principal strength lies in its ability to capture both short- and long-term temporal dependencies by integrating recurrent neural network layers with transformer-based attention mechanisms (Abadi & Brunnermeier, 2018). This enables the model to manage intricate, multivariate datasets, incorporating both static and dynamic factors to improve forecasting precision (Ho & Hung, 2024). The model's inherent interpretability features, such as attention weights and variable selection networks, offer significant insights into the primary determinants of predictions, making it especially beneficial in fields where understanding the influencing factors is essential (Behrens et al., 2022). TFT effectively handles missing data and accommodates varying input lengths, demonstrating adaptability to real-world situations often characterised by incomplete or inconsistent data (Giacomazzi et al., 2023). Nonetheless, TFT also has specific

obstacles. The computational complexity and resource-intensive training process may hinder adoption in resource-constrained situations. The model's dependence on hyperparameter optimisation for optimal performance may necessitate considerable skill and experimentation. Moreover, although its interpretability attributes are advantageous, they can occasionally hinder the model's application due to the necessity for meticulous examination of attention weights and feature importance scores.

2.3.3 Comparison of Predictive Models

To compare statistical models such as ARIMA, SVM, Random Forest, ANN, CNN, GRU, and LSTM, it is essential to have a thorough understanding of their fundamentals, strengths, and weaknesses in handling time-series data. This is particularly crucial when addressing large, nonlinear datasets, such as those used for ammonia forecasting in aquaculture. The ARIMA model depends on linear correlations in time series data via autoregression and moving averages. It is appropriate for short-term forecasting and for smooth data (Grzegorz Dudek, 2016). Nevertheless, it encounters difficulties with nonlinear modes and necessitates comprehensive data preparation to guarantee stationarity (Bento et al., 2021).

Furthermore, owing to its linear nature, ARIMA's efficacy may diminish when applied to large datasets, rendering it less suitable for complex non-linear time series (Mandrikova et al., 2021). SVMs can determine the optimal hyperplane that separates data into several categories, making them advantageous for both classification and regression problems (Cervantes et al., 2020). SVMs are effective for linear and mildly nonlinear datasets. Still, for extremely nonlinear or intricate datasets, more

sophisticated kernel functions or other models, such as neural networks, may be necessary to enhance performance (Leong et al., 2021). The complexity and memory requirements of SVMs make them challenging to use for large-scale nonlinear time series data, such as ammonia concentrations in aquaculture. Random forests are ensemble learning techniques that use multiple decision trees to improve predictive accuracy. It demonstrates proficiency in addressing nonlinear issues while resisting overfitting across diverse datasets (Parmar et al., 2019). Nonetheless, random forest models may require substantial computational resources and do not inherently account for temporal correlations in time series data (Nagendra & Singh, 2023). This constraint diminishes their forecasting efficacy for time-dependent variables.

ANNs possess the potential to capture nonlinearity and adapt to various data types (Kurani et al., 2023). Nonetheless, it is crucial to acknowledge that, although ANNs are adaptable, they require substantial training data and are unable to capture temporal relationships independently (Mundt et al., 2021). This constraint renders them less appropriate for sequential data unless integrated with a circular architecture. CNNs employ convolutional layers to capture spatial hierarchies in data, primarily for image processing. They excel at feature extraction from geospatial data but are ineffective for time series data until substantially altered (Lopez Pinaya et al., 2019). GRU and LSTM are both variants of RNNs specifically engineered for the processing of sequential input. Grus streamlines LSTM architecture by using fewer gates, reducing computational costs, though it may exhibit diminished expressiveness for long-term dependencies (Cahuantzi et al., 2023). TFT employs attention mechanisms to effectively capture both short- and long-term relationships in multivariate time series

data, achieving excellent accuracy in forecasting tasks. However, it often requires substantial computational resources and intricate tuning (Lim et al., 2021).

The architecture of LSTM, featuring forget, input, and output gates, enables it to more effectively simulate complex, nonlinear connections in data than classic linear models such as ARIMA (Siami-Namini et al., 2019). LSTM is explicitly engineered to capture long-term dependencies in sequential data, which are essential for time series forecasting, as historical values affect future values over extended periods (Lindemann et al., 2021). LSTMs can handle large datasets owing to their ability to process sequences of variable length and their resilience to the vanishing gradient problem (Shewalkar et al., 2019). LSTMs are exceptionally versatile for various forms of time series data, including those with irregular intervals and missing values, which are frequently encountered in practical applications such as ammonia forecasting in aquaculture (M. Sun et al., 2020). Empirical research and practical applications demonstrate that LSTMs outperform alternative models for time series forecasting, particularly on complex, nonlinear datasets. In ammonia prediction in aquaculture, where data are influenced by numerous nonlinear parameters (such as temperature and pH) over extended periods, LSTMs offer a strong foundation for identifying complex patterns and correlations, making them particularly appropriate for this application.

2.4 Hyperparameter Setting Method

Hyperparameter tuning is crucial for improving model performance in machine learning and deep learning, focusing on parameters such as learning rate, step size, and network depth (Liao et al., 2022). Contemporary hyperparameter optimisation techniques mostly include grid search, stochastic search, and Bayesian optimisation.

These approaches effectively reduce human interaction and offer numerous possibilities for hyperparameter optimisation across various circumstances (Bischl et al., 2023).

2.4.1 Grid Search

Grid search is a methodical hyperparameter optimisation technique that systematically explores all potential parameter configurations by combining the available values of each hyperparameter in a grid, training and assessing the model sequentially to identify the optimal parameter combination (Shekar & Dagnew, 2019).

Zararsız et al. (2017) examined the Generalised Additive Model for Generating Height and Weight Percentile Curves in Children and Adolescents (GAMLSS), employing a grid search to optimise the distribution parameters (median, skewness, coefficient of variation, and kurtosis) within the GAMLSS framework. The Lambda-Mu-Sigma (LMS) approach was found to be more appropriate for modelling boys' height and weight, whereas the Lambda-Mu-Sigma-Box-Cox Power (LMSP) method was more effective for girls' weight. Sipper (2022) proposed a grid search-based hyperparameter optimisation strategy to enhance random forest models and identify microarray cancer data. The research employed 5-fold cross-validation alongside grid search to identify the ideal hyperparameters. Experimental results indicate that the proposed method achieves 100% classification accuracy on the 2-class Leukaemia, Ovarian, and Small Round Blue Cell Tumour (SRBCT) datasets and 97% on the 3-class Leukaemia and MLL datasets. Belete and Huchaiah (2022) examined the optimisation of hyperparameters by a grid search to enhance the efficacy of machine learning models in predicting HIV/AIDS test outcomes in Ethiopia. Grid search was used to optimise

the hyperparameters of eight models (e.g., RF, SVM, and GB) using 10-fold cross-validation. The findings indicated that the optimised models showed substantial improvements in precision, recall, and F1 scores, achieving a maximum precision of 87.7%.

Budiman (2019) optimised the SVM-Radial Basis Function Kernel (SVM-RBF) model for multi-category image classification, focusing on optimising the hyperparameters C and γ via grid search and cross-validation. Experiments were performed on the Indonesian traditional batik image dataset, achieving optimal classification accuracy of 86.4% with parameters C=27 and γ=2^-15, effectively mitigating overfitting and underfitting. El-Shahat et al. (2024) examined the utilisation of deep learning and machine learning models for short-term solar irradiation forecasting and optimised hyperparameters via grid search cross-validation. The comparison of multiple models, including CNN-LSTM, Random Forest, and Gradient Boosting, demonstrated that CNN-LSTM outperformed the other deep learning models, achieving an adjusted R^2 value of 0.984. (2024) optimised the hyperparameters of various machine learning models for predicting the Water Quality Index (WQI) and Water Quality Classification (WQC) using a grid search. Gradient boosting (GB), random forest (RF), XGBoost, and AdaBoost were employed as classification models, with GB achieving the highest accuracy of 99.5%. Additionally, K-nearest neighbours (KNN), decision tree (DT), support vector regression (SVR), and multilayer perceptron (MLP) were used as regression models; the MLP demonstrated superior performance, achieving an R^2 of 99.8%. Table 2.17 encapsulates the research domains of the aforementioned studies, the models employed, and the evaluations conducted.

Table 2.17

Grid search

Author	Research Area	Model	Evaluation metric
Zararsız et al.,(2017)	Growth and developmental assessment	GAMLSS	Percentile curve
Sipper (2022)	Cancer Classification	Random forest	Accuracy, Precision, Recall, F1-Score, MAE, MSE
Belete & Huchaiah (2022)	Disease Prediction	8 Machine Learning Models	Precision, Recall, F1 Score
Budiman (2019)	Image classification	SVM-RBF	AUC= 0.998, F1 Score= 0.9968
El-Shahat et al. (2024)	Short-term solar irradiation projections	CNN-LSTM, Random Forest and Gradient Boosting	Neural networks or LightGBM perform optimally on AUROC (0.87-0.90+)
Shams et al., (2024)	Predicting water quality indices and water quality classifications	GB, RF, XGBoost, AdaBoost, KNN, DT, SVR, MLP	R^2 99.8%, MAE 0.003, MSE 2.8×10^{-5}, MedAE 0.0009

The research presented in the article underscores the significant role of web searches in the hyperparameter optimisation of machine learning models, illustrating their versatility and efficacy. Nonetheless, grid search incurs significant computing costs,

particularly when the parameter space is extensive or when individual model training sessions are protracted, resulting in an exponential increase in computational complexity (Yu & Zhu, 2020). Moreover, optimal parameter combinations at non-grid points may be overlooked due to the discrete nature of the search processes. In contrast to other optimisation algorithms, such as stochastic search or Bayesian optimisation, grid search is less efficient in high-dimensional parameter spaces and may lead to superfluous calculations.

2.4.2 Random Search

Random search is a hyperparameter optimisation technique that produces a finite set of parameter combinations for training and evaluation by randomly sampling the hyperparameter value space (Zabinsky, 2009). Stochastic search is more efficient than grid search in large search spaces and can identify near-optimal parameter combinations with fewer processing resources (Buslim et al., 2021).

Consequently, the approach is widely used for optimising model hyperparameters. Rom et al. (2024) suggested a multi-objective hyperparameter optimisation strategy utilising random search to enhance the performance of deep learning models. They optimised essential hyperparameters (e.g., learning rate, batch size, and activation function) for MLP, CNN, and AlexNet models and performed tests using the MNIST and CIFAR-10 datasets. The CNN model achieves the best accuracy and F1 score of 99.23% via random search on the MNIST dataset, and it markedly surpasses manual and grid search on the CIFAR-10 dataset. Nugroho & Suhartanto (2020) used random search to optimise the learning rate and batch size of the DenseNet model, thereby enhancing model performance on the CIFAR-10 and CIFAR-100 datasets. Their

research determined that the ideal batch size is 64, the learning rate is 0.1-0.3, and the DenseNet model attains an average accuracy of 95% with this configuration.

Villalobos-Arias et al. (2020) examined the efficacy of a stochastic search-optimised support vector regression model for estimating software development workloads. Experiments demonstrate that the stochastic search-optimised SVR model achieves accuracy and stability comparable to those of grid search while incurring significantly lower computational expense, yielding a standardised accuracy improvement of 0.227. Tarek et al. (2023) introduced the RS-RF model for forecasting soil erosion status by integrating random search and random forests. The research tunes the hyperparameters of random forests (e.g., tree count and splitting criterion) via random search and assesses the model's generalisation performance using cross-validation. Experimental results indicate that the RS-RF model achieves 97.4% accuracy on the test set, surpassing previous machine-learning models. Table 2.18 delineates the research domains of the aforementioned studies, the models employed, and the evaluations conducted.

Table 2.18

Random search

Author	Research Area	Model	Evaluation metric
Rom et al., (2024)	Image classification	MLP, CNN, AlexNet	Random Forest, in Accuracy=85.5%, F1-score=85.6% and AUC=0.90
Nugroho & Suhartanto (2020)	Image classification	DenseNet	Accuracy ≈ 88.5%

Table 2.18 continued

Author	Research Area	Model	Evaluation metric
Villalobos-Arias et al., (2020)	Workload estimates	SVR	Standardised increase 0.129 - 0.227
Tarek et al., (2023)	Predicting soil erosion status	Random forest	Accuracy=97.4%

In conclusion, random search is especially effective for high-dimensional and extensive parameter searches. Given constrained computational resources, it typically identifies near-optimal parameter combinations more quickly than grid searches by avoiding unnecessary exploration of insignificant parameters.

2.4.3 Bayesian Optimisation

Bayesian optimisation is an effective global optimisation approach that aims to identify the optimal solution of a black-box function with a limited number of evaluations.

It is frequently employed for hyperparameter optimisation of machine learning models. Di et al. (2022) introduced a BO-LSTM model that integrates Bayesian optimisation (BO) with LSTM networks to forecast county-level winter wheat yield. Bayesian optimisation was employed to refine the hyperparameters of LSTM (such as the number of hidden units, learning rate, and batch size), utilising multi-source remote sensing and meteorological data as input variables, and employing R^2, RMSE, and MAPE as evaluation metrics. The findings indicated that the BO-LSTM model outperformed all other predictive models, achieving an R^2 of 0.82 and an RMSE of 177.84 kg/ha, surpassing both SVM and Lasso regression. Abbasimehr & Paki, (2021)

proposed a predictive methodology utilising deep learning (multi-head attention model, LSTM, and CNN) in conjunction with Bayesian optimisation for forecasting COVID-19 verified cases in time series. The findings indicated that the LSTM-BO model excelled at long-term forecasting, achieving an SMAPE of 2.59, which markedly surpassed the benchmark model, making it an effective tool for predicting outbreak trends and formulating policies.

Salama et al. (2019) examined a technique that integrates convolutional neural networks with Bayesian optimisation for sheep identification. The trials used an augmented image dataset comprising 52 sheep and 52,000 images, achieving an accuracy of 98%, surpassing the conventional method's 96%. This research offers an effective, automated method for animal identification, particularly well-suited for broad applications in practical settings. Riboni et al. (2022) optimisation of hyperparameters of Spatiotemporal Long Short-Term Memory (ST-LSTM) networks using Bayesian optimisation for the prediction of steering wheel angle in autonomous driving systems. Experiments demonstrate that the optimised BO_ST-LSTM model surpasses traditional end-to-end driving models in prediction accuracy on a public dataset, offering an effective method to enhance the precision and safety of autonomous driving. Table 2.19 delineates the research domains of the aforementioned studies, the models employed, and the evaluations conducted.

Table 2.19

Bayesian optimisation

Author	Research Area	Model	Evaluation metric
Di et al., (2022)	Forecasted wheat production	BO-LSTM, SVM, Lasso	BO-LSTM MAPE= 6.18%
Abbasimehr & Paki, (2021)	COVID-19 case forecast	Multiple attention model, LSTM, CNN	SMAPE =0.25
Salama et al., (2019)	Image recognition	CNN	Accuracy=98%
Riboni et al., (2022)	Steering wheel angle prediction	ST-LSTM	BO-ST-LSTM has the lowest MSE (0.5019), MAE (0.4042) and St. AE (0.5820) on the validation set

Consequently, Bayesian optimisation is especially appropriate for computationally intensive, non-convex, or multimodal problems. It may effectively estimate globally optimal solutions in intricate parameter spaces, offering greater intelligence and resource efficiency compared to grid and random searches. Nonetheless, the computational complexity of Bayesian optimisation is considerable, particularly when dealing with extensive or high-dimensional data, resulting in a significant increase in the costs of training and updating the agent model (Frazier, 2018).

In conclusion, grid search is appropriate for low-dimensional scenarios requiring thorough coverage of the parameter space; stochastic search is ideal for high-

dimensional, extensive parameter exploration, particularly under resource constraints where performance is enhanced; Bayesian optimisation is advantageous for costly evaluations of the objective function, especially when the problem's complexity and the accuracy requirements of the task are elevated. The optimal strategy can be chosen based on the specific characteristics of the problem and the available resources.

2.4.4 Hyperparameter Search Range

Effective hyperparameter optimisation hinges on judiciously defining a search range for hyperparameters, a critical determinant of both optimisation efficiency and model performance (Liao et al., 2022). The hyperparameter range delineates the parameter space accessible to the optimisation method and directly establishes the model's maximum potential performance (Diaz et al., 2017). A substantial body of work has extensively examined the hyperparameter optimisation of LSTMs and their associated deep learning models.

Wang et al. (2022) introduced a hyperparameter optimisation algorithm based on a multilayer perceptron (MLP) for the hyperparameters of the LSTM model, including activation function, dropout, number of hidden units, batch size, and optimiser, with the following search ranges: dropout (0 to 0.5), number of hidden units (20 to 200), and batch size (32 to 512). The trials utilise operational data from servers and air conditioners in a data centre to assess the algorithm's performance and efficiency. Buslim et al. (2021) performed a Bitcoin price prediction analysis using GRU, RNN, and LSTM models, employing Grid Search and Random Search to optimise hyperparameters. The optimisation range spans the number of hidden-layer neurons (16, 32, 64, 128). The findings indicate that GRU exhibits optimal performance under

Grid Search optimisation, achieving a test set MAE of 0.0594. Opoku et al. (2024) examined hyperparameter optimisation for forecasting flow from karst springs utilising LSTM networks in conjunction with Bayesian optimisation. Hyperparameter searches encompassed the number of LSTM cells (32-512, increasing by 32 at each step), the Dropout ratio (0-0.9, incremented by 0.1 at each step), and the learning rate (10^-4 to 10^-2, on a logarithmic scale). The optimised hyperparameter combination substantially improved the model's prediction accuracy, achieving an R^2 of 98.01%.

Reimers and Gurevych (2017) examined the effects of hyperparameter optimisation for BiLSTM networks in natural language processing tasks, including named entity recognition and event detection. Hyperparameter searches encompassed pre-trained word vectors, dropout rates (0.05-0.5), the number of LSTM layers (1-3), and mini-batch sizes (1-64). Research indicates that selecting suitable pre-trained word vectors (e.g., Komninos and Manandhar Embeddings) and optimisers (e.g., Nadam) markedly enhances model efficacy. Hosseini et al. (2024) examined the hyperparameter optimisation of a stochastic search method within a regional hydrological LSTM model, exploring hyperparameter ranges that included the number of hidden units (16-256), batch size (32-256), output dropout rate (0-0.4), learning rate (1×10^{-1} to 1×10^{-2}), target noise standard deviation (0-0.1), daily and hourly sequence lengths (146-1095 days and 168-8760 hours, respectively), and regularization techniques. The experimental findings indicate that the optimised network attains highly precise predictions across 40 catchments. Sarjerao and Sudhagar (2024) suggested an LSTM-based model for an intelligent irrigation system aimed at optimising agricultural water resources. The LSTM hyperparameters were optimised using a stochastic search over learning rate (10^-5 to $10^{\wedge-1}$), hidden layer size (16 to 128), batch size (8 to 64),

dropout rate (0.1 to 0.5), and number of iterations (50 to 500). The experimental findings indicate that the optimisation model markedly surpasses the traditional method for water resource prediction and use efficiency.

Luaran and Alfred (2022) examined hyperparameter configurations for optimising LSTM networks for predicting marine tidal time series. The hyperparameter search encompasses both static and dynamic parameters: mini-batch size (10-200), number of iterations (10-400), and dropout ratio (0.1-0.9); dynamic parameters include the number of LSTM layers (1-4), number of hidden units (1-200), and initial learning rate (0.0001-1). Experiments demonstrate that the optimised hyperparameter configurations markedly enhance the model's predictive accuracy, achieving a minimum RMSE of 0.047 and an R^2 of 0.996. Nejedly et al. (2019) suggested a genetic algorithm-optimised LSTM model for predicting sepsis in ICU patients. The hyperparameter search encompasses learning rates (10^{-4} to 10^{-5}), iteration counts (5-50), and probability thresholds, among others, while maintaining a constant Dropout rate of 0.5. Experiments confirmed the optimised model's efficacy in improving prediction accuracy. Deva Hema and Ashok Kumar (2021) examined the prediction of aggressive driving behaviour in drivers by optimising LSTM hyperparameters. The hyperparameters were explored within the following ranges: time window size (1-10 seconds), learning rate (0.0001-0.01), number of hidden layers (1-3), and number of hidden units per layer (10-100). The experimental findings indicate that hyperparameter configurations obtained via Bayesian optimisation markedly enhance the model's accuracy in identifying driving behaviour, culminating in a final accuracy of 97.02%.

Kervanci and Akay (2023) examined the use of LSTM to optimise hyperparameters for Bitcoin price prediction, including Learning Rate (0.001-0.2), Dropout Rate (0.1-0.2), Batch Size (32-128), Number of LSTM Layers (1-4), and Backtracking Parameters (True, False). The findings indicate that the model trained on the hourly dataset outperforms that trained on the daily dataset, and the optimised parameter configuration markedly enhances prediction accuracy. El Ghazi and Aknin (2023) examined improvements in sensor-based human activity recognition (HAR) performance through the optimisation of LSTM model hyperparameters. Hyperparameters were explored within the following ranges: number of LSTM cells (64-256, increasing by 32), dropout ratio (0.1-0.5, increasing by 0.1), learning rate (1e-3, 1e-4, 1e-5), batch size (32, 64, 128), and optimiser (Adam, RMSprop, SGD). The experimental findings indicate that the Bayesian-optimised LSTM model achieves an average accuracy of 97.71% in 10-fold cross-validation, with an F1 score of 96.66%, thereby far surpassing current methodologies. Mao et al. (2022) examined a PSO-based LSTM hyperparameter optimisation technique to enhance the precision of PM2.5 concentration forecasting. The hyperparameters optimised encompassed batch size (64, 128), neuron count (1-100), and iteration count (1-100). The findings indicated that the optimised model markedly diminished the prediction error (RMSE). Several prior investigations have been undertaken to predict water quality. (2020) examined hyperparameter configurations to enhance the prediction of water quality parameters (dissolved oxygen, DO, and chlorophyll a, Chl-a) using a hybrid CNN-LSTM deep learning model. The hyperparameter search ranges encompass the number of filters in CNN convolutional layers (32-128), filter size (1-2), number of neurons in LSTM layers (32-64), Dropout ratio (0.001), and learning rate (0.001-0.01). The experimental findings indicate that the CNN-LSTM hybrid model outperforms both

the individual model and conventional machine learning models in accurately capturing fluctuations in DO and Chl-a concentrations. Baek et al. (2020) examined the hyperparameter optimisation of an LSTM-based deep learning model for predicting water quality in an inland aquatic setting. The hyperparameter search ranges consist of the number of hidden units (10-100, increasing by 10 at each step), the learning rate (0.001-0.01, increasing by 0.001 at each step), and the Dropout ratio (0.2-0.5, increasing by 0.1 at each step). The experimental findings indicate that the optimised hyperparameter combinations markedly enhance the model's accuracy for water quality prediction, achieving a maximum R^2 of 0.94 and a lower RMSE of 0.12. Zhang et al. (2018) examined the forecasting of water quality variations in Xili Reservoir, Shenzhen, by optimising the hyperparameters of an LSTM model. The hyperparameters were explored within the following ranges: training rounds (epochs: 10-200), batch size (8-128), and hidden layer neuron count (50-250). The integrated model of CEEMDAN data decomposition and LSTM (CEEMDAN-LSTM) markedly enhanced the predictive accuracy for total nitrogen, ammonia nitrogen, total phosphorus, and pH, achieving a prediction R^2 of 0.959 for total nitrogen and a reduction in prediction error (RMSE) to 0.0811. Table 2.20 encapsulates the aforementioned studies.

Table 2.20

Hyperparameter search range

Author	LSTM Layer	Learning Rate	Dropout	Batch Size	Neurons	Epoch
Gorgolis et al., (2019)	2-4	-	-	10-50	200-1200	1-100

Table 2.20 continued

Author	LSTM Layer	Learning Rate	Dropout	Batch Size	Neurons	Epoch
Wang et al., (2022)	-	-	0-0.5	32-512	20-200	-
Buslim et al., (2021)	-	-	-	-	16, 32, 64, 128	-
Opoku et al., (2024)	-	10^{-4}- 10^{-2}	0-0.9	-	32-512	-
Reimers & Gurevych, (2017)	1-3	-	0.05-0.5	1-64	-	-
Hosseini et al., (2024)	-	0.01 -0.1	0-0.4	32-256	16-256	-
Sarjerao & Sudhagar, (2024)	-	10^{-5}- 10^{-1}	0.1 - 0.5	8 -64	16- 128	-
Luaran & Alfred, (2022)	1-4	0.0001-1	0.1-0.9	10-200	1-200	10-400
Barzegar et al., (2020)	-	0.001-0.01	0.001	-	CNN:32-128 LSTM: 32-64	-
Baek et al., (2020)	2	0.001-0.01	0.2-0.5	-	10-100	-
Nejedly et al., (2019)		0.001-0.01	0.5	-		5-50
Deva Hema & Ashok Kumar, (2021)	1-3	0.0001-0.01	-	-	1-100	-
Zhang et al., (2018)	-	-	-	8-128	50-250	10-200
KERVANCI & AKAY (2023)	1-4	0.001-0.2	0.1-0.2	32-128	-	-
El Ghazi & Aknin, (2023)	-	1e-3, 1e-4, 1e-5	0.1-0.5	32, 64, 128	64-256	-
Mao et al., (2022)	-		-	64, 128	1-100	1-100

In conclusion, it is advisable to empirically determine the parameter range to optimise the LSTM model's hyperparameters. The number of hidden units is specifically set to 1-256, accommodating tasks of varying complexity while maintaining a balance between model capacity and computational performance. The recommended range for the number of LSTM layers is 1-4. Fewer layers are appropriate for uncomplicated issues, whereas additional layers might enhance the model's expressive capacity, albeit at increased risk of overfitting. The learning rate is a crucial hyperparameter that influences the model's convergence speed and stability, typically ranging from 0.001 to 0.01. This range encompasses the operational period of prevalent optimisers, effectively balancing stability and training velocity. The advised Dropout Ratio range is 0.1 to 0.5, with increments of 0.1, which effectively mitigates overfitting while preserving the model's learning capacity. The batch size is centrally configured between 8 and 256. Reduced batch sizes are appropriate for memory-limited systems, but larger batches yield more consistent gradient updates and accelerate training. The recommended range for training epochs is typically 1-200, with the option to adjust based on data size and model complexity. Reduced training iterations are appropriate for situations with limited data and straightforward model convergence, whereas increased training iterations are advantageous for extensive datasets and intricate tasks. The aforementioned range of settings is derived from the experience of multiple successful examples, capable of adapting to diverse work conditions while ensuring the efficiency and efficacy of hyperparameter optimisation. These recommendations enhance model performance and offer resilient solutions that address many limitations, particularly in complex timing jobs and extensive data analysis. Table 2.21 summarises the parametric design.

Table 2.21

Parametric design summary

Hyperparameter	Recommended Search Range
Dropout Ratio	0 - 0.9
Number of Hidden Units	1 - 256
Batch Size	8 - 256
Learning Rate	0.001-0.1
Number of Epochs	1 - 200
Number of LSTM Layers	1 - 4

2.5 Optimisation Methods for Models

Optimisation methods are procedures and approaches used to identify the optimal solution from a range of potential solutions, typically in mathematical or computational contexts. In machine learning and deep learning, optimisation techniques are essential for model training and determining optimal parameter values. Previous research identified and examined several prevalent optimisation techniques utilised across diverse domains, including machine learning.

2.5.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a fundamental optimisation approach extensively employed for training machine learning models, especially neural networks (Jentzen et al., 2021). Stochastic Gradient Descent (SGD) is an iterative technique that updates model parameters by computing the gradient of the loss function with respect to the parameters and moving in the direction that reduces the loss (Jentzen et al., 2021). In contrast to conventional gradient descent, which computes the gradient using the entire dataset at each iteration, stochastic gradient

descent (SGD) approximates the gradient by sampling a subset of the data, either a single data point or a mini-batch (Keuper & Pfreundt, 2015). This stochastic method diminishes computational complexity, enhancing scalability for extensive datasets and high-dimensional parameter spaces (Cutkosky & Busa-Fekete, 2018). The SGD optimiser's specific formula is shown in Formula 2.1.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla L(\theta_t; x_i, y_i) \qquad (2.1)$$

Diab (2019) proposed an enhancement to the performance of the SGD method in text categorisation. The work employs a Grid-Search methodology for hyperparameter optimisation, examining diverse configurations for feature representation, transformation, and weighting in summaries of terrorist attack incidents. Evaluating SGD on SVM, Logistic Regression, and Perceptron classifiers using 10-K-fold cross-validation indicates that grid-search optimisation improves pre-classification parameters and classifier performance in terms of accuracy and execution time. Liu et al. (2021) discuss obstacles in medical image processing using deep learning, emphasising the time-intensive nature of model training due to the large number of parameters. This work presents a new technique, Particle Swarm Optimisation-Stochastic Gradient Descent with Momentum (PSO-SGD), which integrates the benefits of stochastic gradient descent and particle swarm optimisation. The proposed technique aims to improve classification accuracy and optimise network parameters. Experimental validation across the COVID-19 Radiography and Blood Cell Image datasets demonstrates that the method improves efficiency without compromising accuracy. Zou & Sugihara (2020) present a technique for establishing a human-skeleton marker model for precise motion analysis using data from an optical

motion capture system. The dual-phase nonlinear least-squares error minimisation addresses the chicken-and-egg problem by concurrently identifying the model and estimating the entire body posture. Stochastic Gradient Descent significantly reduces computational costs while maintaining accuracy. Table 2.22 summarises the models, findings, and datasets used in the aforementioned articles that employed the SGD technique.

Table 2.22

Stochastic gradient descent review

Author	Tasks	Findings	Dataset
Diab (2019)	Terrorist attacks incidents prediction using SVM	Grid-search optimizes SGD classification hyperparameters, impact pre-classification and classifier performance in accuracy	Global Terrorism Database
Liu et al. (2021)	Medical image classification	PSO-SGD is optimal solution more rapidly and with improved classification accuracy solution efficiency	Two data sets: COVID-19 Radiography Data Set (COVID-19) and Blood Cell Images Data Set (BCIDS).
Zou & Sugihara (2020)	recognise a human model as a rigid chain of motion.	SGD saves a great deal of computation expense without compromising accuracy	Data from the optical motion capture system

According to the findings above, SGD is used in neural networks to improve model training performance. Although SGD has made significant contributions to time series

prediction, its limitation of using a uniform learning rate for all parameters can lead to slow convergence or divergence (Ziyin et al., 2021). To address these issues, adaptive optimisers such as Adam have emerged with promising results and have dominated several research areas. Alternative methods, such as Adam, can alleviate convergence challenges by individually adjusting the learning rate based on historical gradient behaviour, thereby overcoming the shortcomings associated with SGD.

2.5.2 Adaptive Gradient

Adaptive Gradient (AdaGrad) is an early and influential algorithm within the family of adaptive learning rate optimisers. Proposed by Duchi et al. (2011), AdaGrad's key insight was to adjust the effective learning rate dynamically for each parameter based on the historical accumulation of its gradients. In simpler terms, parameters that have experienced larger updates over time receive smaller subsequent learning rates, whereas parameters that have received fewer updates benefit from a relatively larger learning rate (Ward et al., 2019). This element-wise adaptation enables AdaGrad to handle scenarios in which different features or parameters exhibit varying degrees of importance and frequency during training (Hadgu et al., 2015). The AdaGrad optimiser's specific formula is shown in Formulas 2.2 and 2.3.

1. Accumulated Squared Gradients:

$$G_t = G_{t-1} + g_t^2 \qquad (2.2)$$

Here, G_t is the sum of the squares of the gradients up to time step t.

2. Parameter Update:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t \qquad (2.3)$$

$\sqrt{G_t}$ computes the element-wise square root of the accumulated squared gradients.

The learning rate η is scaled by $\frac{1}{\sqrt{G_t + \epsilon'}}$, effectively reducing the learning rate for parameters associated with frequently occurring features.

AdaGrad has been effectively utilised across a diverse range of applications, demonstrating its versatility and robustness in optimising machine learning models. In the field of image classification, Luo et al. (2019) integrated AdaGrad into deep CNNs to classify images within extensive datasets. This integration led to improved early-stage convergence and higher accuracy in time-series forecasting, particularly across datasets with varying levels of image sparsity. Yang & Liu (2021) applied AdaGrad to LSTM networks to predict stock prices, weather conditions, and energy consumption trends. AdaGrad's ability to handle non-stationary patterns and sparse features in time-series data was instrumental in enhancing the predictive performance of these models. In the domain of water quality and aquaculture data prediction, AdaGrad has been effectively employed to enhance the accuracy and robustness of predictive models through various innovative approaches. Liao & Zhao (2016) incorporated AdaGrad into a Principal Component Analysis-Fuzzy Neural Network-Deep Elastic Back-Propagation (PCA-FNN-DEBP) model to predict dissolved oxygen levels in shrimp culture ponds. This approach significantly enhanced the model's robustness and accuracy, yielding substantial improvements in prediction error metrics compared to traditional methods. Additionally, Yang et al. (2017) applied AdaGrad to deep neural networks designed to monitor and predict critical parameters, such as dissolved oxygen

and pH, in fish ponds. The optimised models achieved higher prediction precision, thereby supporting more informed decision-making for maintaining optimal aquaculture environments. Collectively, these studies underscore AdaGrad's pivotal role in advancing predictive modelling for water quality and aquaculture applications, demonstrating its capability to handle complex, nonlinear datasets and improve the reliability of critical environmental forecasts. Table 2.23 summarises the models, results, and datasets utilised in the papers above.

Table 2.23

Adaptive gradient review

Author	Tasks	Findings	Dataset
Liangchen Luo et al. (2019)	AdaGrad in deep CNNs for image classification	Improved early-stage convergence and higher accuracy, particularly for datasets with varying levels of image sparsity.	Extensive image classification datasets
Huanhai Yang et al. (2021)	AdaGrad in LSTM networks for time-series forecasting	Enhanced predictive performance for stock prices, weather conditions, and energy consumption.	Non-stationary time-series datasets
Fan Liao (2016)	AdaGrad in PCA-FNN-DEBP model for shrimp culture ponds	Enhanced robustness and accuracy in predicting dissolved oxygen, with significant improvements in error metrics.	Shrimp culture pond datasets
Po-Yuan Yang et al. (2017)	AdaGrad in deep neural networks for fish pond monitoring	Higher prediction precision for dissolved oxygen and pH, supporting better aquaculture decision-making.	Fish pond water quality datasets

One of the foremost advantages of AdaGrad is its ability to adapt learning rates per parameter, which is particularly useful when dealing with sparse or skewed data

distributions. By accumulating squared gradients, parameters that rarely receive updates can benefit from a relatively larger step size, thereby potentially improving convergence in underrepresented dimensions (Tian et al., 2023). This feature makes AdaGrad especially appealing for natural language processing tasks and certain sensor-based time-series prediction tasks (Hadgu et al., 2015).

However, the algorithm also presents notable challenges. Chief among them is the continual accumulation of squared gradients over time, which causes the effective learning rate to shrink excessively (Zou et al., 2019). As training progresses, this can result in sluggish learning or eventual stagnation, in which updates become too small to support further progress (Chaudhury & Yamasaki, 2021). Consequently, while AdaGrad can excel in some scenarios, its inherent limitations often prompt researchers to explore modifications and alternative optimisers such as RMSprop and Adam to maintain higher learning rates throughout longer training sessions (Guo et al., 2020).

2.5.1 Root Mean Square Propagation

The Root Mean Square Propagation (RMSprop) algorithm, proposed by Hinton (2012) for neural networks, uses an adaptive learning rate gradient descent optimisation method. Its core idea is to dynamically adjust the learning rate using an exponential-weighted moving average of past gradient squares. This avoids instability or slow convergence during training (Ida et al., 2017). The algorithm performs better than traditional stochastic gradient descent when training deep learning models, especially on small-batch data. Its ability to achieve fast convergence while maintaining robustness has led to widespread adoption in applications such as image processing,

speech recognition, and time series prediction (Chaudhury & Yamasaki, 2021). The AdaDelta update rules are as formula 2.8-2.11:

1. Accumulated Decaying Average of Squared Gradients:

$$E[g^2]_t = \rho \cdot E[g^2]_{t-1} + (1-\rho) \cdot g_t^2 \quad (2.8)$$

$E[g^2]_t$ represents the decaying average of past squared gradients up to time step t.

2. Update Parameters:

$$\theta_{t+1} = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \quad (2.9)$$

In the broader field of machine learning, RMSprop has been used for tasks such as image classification, speech recognition, and time series forecasting. For instance, in a comparative study by Ida et al. (2017), various neural network architectures (including both feedforward and convolutional networks) were trained on standard benchmark datasets to demonstrate how RMSprop approximates Hessian-based preconditioning using first-order gradients, resulting in more efficient training and faster convergence relative to alternative optimisers. Likewise, Chaudhury and Chaudhury & Yamasaki (2021) applied RMSprop to convolutional neural networks (CNNs) trained on noisy datasets such as CIFAR-10 and MNIST, noting marked improvements in stability and test error reduction compared to vanilla SGD. Meanwhile, (2020) integrated RMSprop with dynamic constraints in both RNNs and CNNs and observed superior performance, particularly in terms of generalisation and convergence speed, across standard image and text classification benchmarks such as

CIFAR-10. Finally, Zhong et al. (2020) demonstrated RMSprop's advantages in natural language processing tasks, with experiments on benchmark NLP datasets showing faster convergence and better overall training performance than SGD. Taken together, these studies underscore RMSprop's versatility and efficiency across multiple deep learning architectures and data domains.

In the context of water quality data prediction and aquaculture data prediction, Li et al. (2022) integrated RMSprop into a Support Vector Machine (SVM) framework to estimate parameters such as dissolved oxygen, pH, ammonium nitrogen, nitrate nitrogen, and nitrite nitrogen, thereby achieving a notable 99% prediction accuracy using both published and measured data from industrial systems. Meanwhile, Rasheed Abdul Haq & Harigovindan (2022) employed RMSprop in hybrid CNN-LSTM and CNN-GRU models to forecast similar parameters, finding that the CNN-LSTM model with RMSprop achieved superior prediction accuracy and computational efficiency across two distinct water quality datasets. In another approach, Eze et al. (2021) leveraged RMSprop within an Ensemble Empirical Mode Decomposition (EEMD) and LSTM hybrid model, demonstrating faster convergence and improved prediction precision for dissolved oxygen and pH data from South African aquaculture farms. More recently, Hu et al. (2024) adopted a Simple Recurrent Unit (SRU) model optimised with RMSprop, achieving robust predictions and enhanced decision support for aquaculture operations using real-time water-quality data.

Despite these successes, RMSprop also presents several advantages and challenges. One clear advantage lies in its resilience to hyperparameter tuning. Because RMSprop adapts the learning rate for each weight, it typically requires fewer manual adjustments

once a reasonable initial setting is found (Zaheer et al., 2018). This property can be particularly beneficial when modelling complex time series data, as is often the case in water quality and aquaculture applications (Huang et al., 2019). Moreover, RMSprop is computationally efficient and simple to implement, making it attractive for real-world deployment where processing speed and ease of integration are crucial (Zhang et al., 2024). Using the RMSprop algorithm, Table 2.24 summarises the models, results, and datasets utilised in the papers above.

Table 2.24

Root mean square propagation review

Author	Tasks	Findings	Dataset
Ida et al. (2016)	Comparative study on various neural network architectures (e.g., feedforward and convolutional networks)	RMSprop approximates Hessian-based preconditioning using first-order gradients, resulting in more efficient training and faster convergence compared to alternative optimisers.	Standard benchmark datasets
Chaudhury and Yamasaki (2021)	RMSprop applied to CNNs trained on noisy datasets	Improved stability and reduced test error compared to vanilla SGD.	CIFAR-10, MNIST
Dong Liang et al. (2020)	Integration of RMSprop with dynamic constraints in RNNs and CNNs	Superior performance in generalisation and convergence speed.	CIFAR-10, standard text classification benchmarks

Table 2.24 continued

Author	Tasks	Findings	Dataset
Hui Zhong et al. (2020)	RMSprop in NLP tasks	Faster convergence and better overall training performance compared to SGD.	Benchmark NLP datasets
Tingting Li et al. (2022)	RMSprop integrated into SVM for water quality prediction	Achieved 99% prediction accuracy for dissolved oxygen, pH, ammonium nitrogen, nitrate nitrogen, and nitrite nitrogen.	Published and measured data from industrial systems
Rasheed Abdul Haq and Harigovindan (2022)	RMSprop in hybrid CNN-LSTM and CNN-GRU models for water quality forecasting	CNN-LSTM with RMSprop offered superior prediction accuracy and computational efficiency.	Two distinct water quality datasets
Eze et al. (2021)	RMSprop in EEMD and LSTM hybrid model	Faster convergence speeds and improved prediction precision for dissolved oxygen and pH data.	Data from South African aquaculture farms
Wu-Chih H et al. (2024)	SRU model optimised with RMSprop for aquaculture predictions	Robust predictions and enhanced decision support for real-time water quality data.	Real-time water quality data

However, some challenges persist: RMSprop can still require careful tuning of its decay rate hyperparameter, and in some cases it may converge to sharp local minima if not combined with other regularisation or scheduling techniques (Luo et al., 2019).

The performance benefits of RMSprop may be less pronounced for highly sparse data or for tasks that require large batch sizes (Chaudhury & Yamasaki, 2021). Additionally, the performance benefits of RMSprop may be less pronounced for highly sparse data or tasks that require large batch sizes, prompting researchers to investigate variants and alternative optimisers that address these specific limitations.

2.5.2 Adaptive Delta

Adaptive Delta (AdaDelta) is an adaptive learning rate optimisation algorithm introduced by Zeiler (2012) to address the limitations of its predecessor, AdaGrad. Unlike AdaGrad, which accumulates the sum of squared gradients and can cause the learning rate to diminish excessively over time, AdaDelta restricts the window of accumulated past gradients by using an exponentially decaying average of squared gradients (Zeiler, 2012). This modification allows AdaDelta to adaptively adjust the learning rate based on recent gradient information, ensuring more consistent and stable convergence during training (Ding et al., 2019). Additionally, AdaDelta eliminates the need for a manually set global learning rate by incorporating a parameter-free mechanism to determine update magnitudes, thereby simplifying the optimisation process (Wanjau et al., 2021). The algorithm introduces a decay rate parameter, rho (ρ), which controls the influence of recent versus older gradient information, balancing adaptability and stability (Zhang et al., 2019). By maintaining this balance, AdaDelta effectively prevents the learning rate from shrinking too much, making it particularly suitable for training deep neural networks and handling complex, non-stationary data patterns (Mahsa & Lee, 2018). The AdaDelta optimiser's specific formula is shown in Formula 2.4-2.7.

1. Accumulated Decaying Average of Squared Gradients:

$$E[g^2]_t = \rho \cdot E[g^2]_{t-1} + (1-\rho) \cdot g_t^2 \qquad (2.4)$$

$E[g^2]_t$ represents the decaying average of past squared gradients up to time step t.

2. Accumulated Decaying Average of Squared Parameter Updates:

$$E[\Delta\theta^2]_t = \rho \cdot E[\Delta\theta^2]_{t-1} + (1-\rho) \cdot \Delta\theta_{t-1}^2 \qquad (2.5)$$

$E[\Delta\theta^2]_t$ represents the decaying average of past squared parameter updates up to time step t.

3. Compute Parameter Update:

$$\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]_{t-1}+\epsilon}}{\sqrt{E[g^2]_t+\epsilon}} \cdot g_t \qquad (2.6)$$

This scales the gradient g_t by the ratio of the root of the accumulated parameter updates to the root of the accumulated gradients, ensuring adaptive step sizes.

4. Update Parameters:

$$\theta_{t+1} = \theta_t + \Delta\theta_t \qquad (2.7)$$

The parameters are updated by adding the computed $\Delta\theta_t$.

AdaDelta has been effectively applied across a wide range of machine learning prediction tasks, demonstrating its versatility and robustness in optimising diverse

models. In the domain of crop yield prediction, (2019) employed AdaDelta to train deep CNNs with six convolutional layers, using NDVI and RGB data from UAVs. This approach resulted in significant accuracy improvements, achieving an MAE of 484.3 kg/ha and a MAPE of 8.8% during the early growth stages of crops. In the field of neurological prediction, Wendt et al. (2022) optimised regression models using AdaDelta to predict walking distance during the 2-minute walk test in patients with neurological impairments. Their models achieved high prediction accuracy with a relative error of just 6.1%, demonstrating robustness across diverse patient demographics and mobility aids.

AdaDelta has also been instrumental in predicting team performance in software engineering projects. Nevavuori et al. (2019) utilised DNNs with two hidden layers, optimised using AdaDelta, to predict teams' academic performance. The model achieved a prediction accuracy of 76.73% and provided explainable outputs through Shapley Additive exPlanations (SHAP), leveraging a substantial dataset of over 30,000 entries grouped into 74 teams. In real-time water quality monitoring, Kumar et al. (2024) implemented AdaDelta in hybrid models that integrate real-time sensor data with predictive algorithms, optimising aquaculture water quality management. This integration enabled proactive management and enhanced adaptability to diverse aquaculture environments, achieving higher accuracy in predicting dissolved oxygen and pH levels using data from aquaculture farms.

Furthermore, AdaDelta has been applied to deep learning for time-series data. RahulGandhi et al. (2023) enhanced LSTM and GRU models with AdaDelta, resulting in improved prediction accuracy and computational efficiency for nonlinear and

dynamic water-quality data derived from historical aquaculture datasets. In the context of hybrid aquaculture prediction models, Liu et al. (2013) combined genetic algorithms with SVMs and integrated AdaDelta, achieving enhanced prediction accuracy and reduced root mean square errors for critical water quality parameters such as ammonia and nitrites using data from aquaculture production systems. Additionally, Jongjaraunsuk et al. (2024) applied AdaDelta in CNN-LSTM hybrid models to predict water quality in outdoor recirculating aquaculture systems, specifically focusing on red tilapia production. Their models successfully predicted dissolved oxygen, ammonia, and other key parameters with high R^2 values, thereby optimising environmental management practices in aquaculture settings. Table 2.25 summarises the models, results, and datasets utilised in the papers above.

Table 2.25

Adaptive delta review

Author & Date	Tasks/Model Used	Findings	Dataset
Nevavuori et al. (2019)	AdaDelta in CNNs for crop yield prediction	Achieved significant accuracy improvements with MAE of 484.3 kg/ha and MAPE of 8.8% during early growth stages.	NDVI and RGB data from UAVs
Wendt et al. (2022)	AdaDelta in regression models for predicting walking distances	Achieved high prediction accuracy with a relative error of 6.1%, robust across diverse patient demographics.	Neurological impairment patient data
Giannakas et al. (2021)	AdaDelta in DNNs for team performance prediction in software engineering projects	Achieved prediction accuracy of 76.73% and explainable outputs using SHAP.	Dataset of over 30,000 entries grouped into 74 teams

Table 2.25 continued

Author & Date	Tasks	Findings	Dataset
Kumar et al. (2024)	AdaDelta in hybrid models for real-time water quality monitoring	Enhanced adaptability and proactive management of aquaculture water quality.	Real-time sensor data from aquaculture farms
Gandh et al. (2023)	AdaDelta-enhanced LSTM and GRU models for time-series prediction	Improved prediction accuracy and computational efficiency for non-linear and dynamic water quality data.	Historical aquaculture datasets
Liu et al. (2013)	AdaDelta in genetic algorithms and SVM hybrid models for aquaculture prediction	Achieved enhanced prediction accuracy and reduced RMSE for critical water quality parameters.	Data from aquaculture production systems
Jongjaraunsuk et al. (2024)	AdaDelta in CNN-LSTM hybrid models for outdoor aquaculture systems	Successfully predicted dissolved oxygen, ammonia, and other key parameters with high R^2 values.	Data from outdoor recirculating aquaculture systems (e.g., red tilapia production)

AdaDelta efficiently handles sparse gradients, making it well-suited for processing large-scale sensor data common in aquaculture systems (Khan & Byun, 2023). However, AdaDelta also presents challenges, such as the introduction of additional hyperparameters, such as the decay rate (rho), which require careful tuning to balance the influence of recent and older gradient information. Improper settings can lead to suboptimal performance or instability (Lu & Ma, 2020). Furthermore, while AdaDelta improves upon AdaGrad, it may not always outperform newer adaptive optimisers like Adam, which incorporate mechanisms for bias correction and momentum (Dogo et al.,

2018). The computational overhead of maintaining multiple moving averages can also be a concern for very large models or extensive datasets (Nitya Nand Jha, 2024). Additionally, AdaDelta's performance can be sensitive to data characteristics and neural network architecture, necessitating thorough experimentation and validation (Malek et al., 2014).

2.5.3 Adaptive Moment Estimation

The Adaptive Moment Estimation (Adam) optimisation algorithm is a widely used method in machine learning and deep learning for training models. Introduced by Kingma & Ba (2014), Adam combines the benefits of two popular optimisers, AdaGrad and RMSprop, by integrating momentum and adaptive learning rates. Adam maintains and updates moving averages of both the gradient (first moment) and the squared gradient (second moment), using these estimates to adapt the learning rate for each parameter dynamically (Dereich & Jentzen, 2024). This dual moment estimation allows Adam to converge efficiently by smoothing noisy updates and avoiding overly aggressive parameter adjustments. Additionally, Adam includes bias-correction mechanisms to ensure accurate estimates of these moments, especially during the early stages of training (Guo et al., 2021). The algorithm's ability to handle sparse gradients and non-stationary objectives makes it highly effective for a broad range of applications, from natural language processing and computer vision to time series forecasting (Modoranu et al., 2024). Its default hyperparameters (learning rate, beta1, and beta2) often work well across tasks, reducing the need for extensive tuning (Malviya et al., 2023). This versatility and computational efficiency have made Adam a cornerstone in modern machine learning, particularly for optimising deep neural

networks. The idea of combining RMSprop and momentum yields the formulas 2.12-2.17.

1. Compute the gradient:

$$g_t = \nabla_\theta L(\theta_t) \tag{2.12}$$

2. Update Biased First and Second Moment Estimates:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.13}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.14}$$

m_t captures the exponentially decaying average of past gradients (similar to momentum).v_t captures the exponentially decaying average of past squared gradients.

3. Compute Bias-Corrected Moment Estimates: Since m_t and v_t are initialized at zero, they are biased towards zero, especially during the initial time steps. To counteract this bias, compute bias-corrected estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.15}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.16}$$

4.Update Parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t \tag{2.17}$$

The learning rate η is adapted for each parameter based on the estimates of $\hat{m}_t$ and $\hat{v}_t$.

Prior studies have investigated Adam in different application areas (Jackson & Adam, 2021; Liu et al., 2021; Sharma et al., 2022; Yi et al., 2019). For instance, Reddi et al. (2018) identify the issue as the exponential moving average and present improved variants of the Adam algorithm with long-term memory, addressing convergence problems and often yielding better performance. Bock et al. (2018) scrutinise the widely used Adam optimiser, identify errors in its convergence proof, and propose an enhanced proof to address these issues. It emphasises a case in which Adam fails to converge and critiques prior analyses of the algorithm. The study suggests resolving convergence problems by integrating the 'long-term memory' of past gradients into the algorithm, introducing new Adam variants. These variants not only rectify convergence issues but also enhance practical performance, offering a more comprehensive understanding of Adam's behaviour in neural network training.

Prabowo et al. (2019) introduce a Deep Belief Network (DBN) model to improve exchange rate forecasting compared to traditional ANNs. It employs the Adam optimisation method for faster learning and achieves better results, outperforming the DBN-SGD model in convergence and performance metrics on USD/IDR exchange rate data. Attrapadung et al. (2022) address the need for privacy-preserving machine learning (PPML) systems that securely analyse sensitive data. Unlike previous approaches that required modifying machine learning algorithms, the authors propose a framework that enables efficient and secure evaluation of standard ML algorithms via secure multi-party computation (MPC). Their protocols handle challenging computations crucial for deep neural networks (DNNs), providing both high accuracy

and efficiency. The study employs Adam optimisation, and results show that their framework outperforms existing three-party systems for secure DNN training, with significant speed improvements on benchmark networks like AlexNet and VGG16. Table 2.26 shows prior studies on Adam optimiser.

Table 2.26

Adaptive moment estimation review

Author	Tasks	Comments	Dataset
Bock et al. (2018)	Identify errors in the convergence proof of the optimiser, rendering it inaccurate.	Addressed the identified mistakes of ADAM-Optimiser's convergence properties	Big data
Reddi et al. (2018)	Demonstrate how the algorithms' use of an exponential moving average contributes to these failures.	ADAM describes the specific issues and does not converge to the best solution.	MNIST dataset
Prabowo et al. (2019)	Predict USD / IDR daily exchange rate	The Adam algorithm improves the Deep Belief Network during the fine-tuning phase.	Daily exchange rate information for USD/IDR
Attrapadung et al. (2022)	A framework for MPC-unfriendly computations that makes it possible to evaluate machine learning algorithms securely and effectively using third-party protocols.	Efficient division protocol for accurate fixed-point arithmetic in ML algorithms.	2 datasets MNIST 28 x 28 pixel and images of CIFAR- 10

Despite Adam being a widely used optimiser for neural network training, it has drawbacks, such as slow convergence or oscillations during optimisation due to inflexible gradient update directions, especially in complex optimisation spaces. This

led to the development of Nadam, which integrates Nesterov momentum with Adam to enhance convergence. The introduction of Nadam addresses the limitations of Adam-based techniques, making it a promising option for optimising neural networks and other machine learning models. Ongoing efforts to explore alternative techniques or improve existing methods underscore the commitment to overcoming the challenges associated with hyperparameter tuning and optimising the training process for artificial neural networks.

2.5.4 Adaptive Moment Estimation with Weight Decay

Adaptive Moment Estimation with Weight Decay (AdamW) is an optimisation approach that incorporates a Weight Decay strategy derived from the Adam algorithm to enhance the model's generalisation performance (Loshchilov & Hutter, 2019). The conventional Adam approach enhances convergence by computing first- and second-order momentum estimates of the gradient to dynamically adjust the learning rate (Zhou & Li, 2022). However, the L2 regularisation component is applied passively to the gradient, potentially leading to imprecise weight updates (Zeke Xie, Issei Sato, 2021). AdamW implements a more precise regularisation approach by decoupling weight decay from gradient computation, successfully alleviating overfitting (Jia et al., 2024). The precise implementation is illustrated in Equation 2.18-2.22.

1. Gradient Calculation and First- and Second Moment Estimates

Let g_t represent the gradient of the loss function with respect to the weights θ_t, and m_t and v_t denote the first and second-moment estimates, respectively. These are updated as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.18)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.19)$$

where β_1 and β_2 are exponential decay rates for the moments.

2. Bias-Correction for the Moments

To address bias introduced during initialisation, bias-corrected moment estimates are computed as:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (2.20)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (2.21)$$

3. Weight Update with Decoupled Weight Decay

The key update step in AdamW includes the explicit weight decay term, given by:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda \theta_t \right) \quad (2.22)$$

where:

- η is the learning rate,
- ϵ is a small constant to prevent division by zero,
- λ is the weight decay coefficient.

In the domain of machine learning prediction, AdamW is widely used for its strong convergence performance and superior generalisation capability. Remya and Sasikala (2020) investigated the development of a neuro-fuzzy forecasting model utilising AdamW optimisation in the domain of agricultural forecasting. The authors employed

a system of back-propagation neural networks to attain effective parameter tuning via the AdamW optimisation method. The model demonstrates superior performance on agricultural datasets, encompassing yield and environmental factors, achieving a prediction accuracy of 96.78%, markedly surpassing conventional optimisation techniques. This work confirms the efficacy and resilience of AdamW in modelling intricate nonlinear data and showcases its optimisation potential, particularly in the essential task of predicting agricultural yields.

Agarwal et al. (2024) further substantiated AdamW's efficacy in number recognition. The team employed AdamW to train a Sequential Convolutional Neural Network (Sequential CNN), integrating data augmentation and regularisation methods for a numeral identification challenge. Experimental findings indicate that AdamW markedly improves the efficiency and accuracy of model training, yielding exceptional recognition performance on standard datasets such as MNIST. This indicates that AdamW is highly applicable to computer vision tasks, particularly those that require high precision.

In the domain of water quality prediction, Guan (2023) introduced an enhanced AdamW algorithm that integrates a weight-prediction mechanism to accelerate neural network convergence. While the primary experiments in this paper were on picture classification and language modelling tasks, the findings offer novel optimisation strategies for water quality prediction. The weight prediction mechanism significantly improves the convergence rate and model accuracy of AdamW. This method is especially appropriate for situations that require processing high-dimensional data and intricate features, such as real-time monitoring and forecasting of water quality metrics.

While the study does not directly focus on water quality, its methodology offers valuable insights for predictive modelling in associated domains. Table 2.27 summarises the AdamW optimiser.

Table 2.27

Adaptive moment estimation with weight decay review

Author & date	Tasks	Comments	Dataset
Remya & Sasikala (2020)	Predictive modelling in agriculture	Developed a neuro-fuzzy prediction model using a backpropagation neural network optimised with AdamW, achieving a prediction accuracy of 96.78%	Agricultural datasets containing yield and environmental metrics
Agarwal et al. (2024)	Numeral recognition in computer vision tasks	Applied AdamW to train a Sequential CNN with data augmentation and regularisation, achieving precise numeral recognition	Digit datasets such as MNIST or similar numerical recognition benchmarks
Lei Guan (2023)	Enhancing convergence in deep neural networks training	Proposed a weight prediction mechanism for AdamW, improving convergence and accuracy; findings have potential applications in predictive modelling (e.g., water quality)	Image classification and language modelling datasets (specific datasets not disclosed)

AdamW enhances model generalisation by implementing an independent weight decay process during gradient updates, thereby eliminating the interference associated with typical regularisation methods. Moreover, AdamW demonstrates robust stability and convergence when handling high-dimensional, sparse, or non-smooth data, making it suitable for a wide range of deep learning problems. Nonetheless, its

drawback is that in certain very intricate problems, AdamW may become trapped in local optima, hindering comprehensive exploration of the global solution space.

2.5.5 Nesterov-accelerated Adaptive Moment Estimation

Nesterov-accelerated Adaptive Moment Estimation (Nadam), a variant of Adam, was developed by Dozat (2016). The uniqueness of Nadam lies in its integration of Adam and Nesterov momentum, enabling the optimiser to not only rely on the exponentially weighted average of past gradients during parameter updates, but also to implicitly estimate the future gradient direction, thereby adaptively adjusting the current update direction and improving both convergence speed and stability (Dozat, 2016). Nadam as shown in formula 2.23-2.28:

1. Update gradient

$$g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1}) \tag{2.23}$$

2. Update Biased First and Second Moment Estimates:

$$m_t \leftarrow \mu_t m_{t-1} + (1-\mu_t) g_t \tag{2.24}$$

$$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2 \tag{2.25}$$

3. Compute Bias-Corrected Moment Estimates:

$$\hat{m} \leftarrow \left(\mu_{t+1} m_t / (1 - \textstyle\prod_{i=1}^{t+1} \mu_i)\right) + \left((1-\mu_t) g_t / (1 - \textstyle\prod_{i=1}^{t} \mu_i)\right) \tag{2.26}$$

$$\hat{n} \leftarrow \nu n_t / (1-\nu^t) \tag{2.27}$$

4. Update Parameters:

$$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{n}_t + \varepsilon}} \times \hat{m}_t \tag{2.28}$$

Dozat (2016) finds that Nadam and Adam achieve the best results even without adjusting the learning rate, which is often unavoidable, even though they have the most hyperparameters. Importantly, Nadam significantly outperforms other algorithms, including its parent Adam, in reducing training and validation losses. As shown in Figure 2.10:

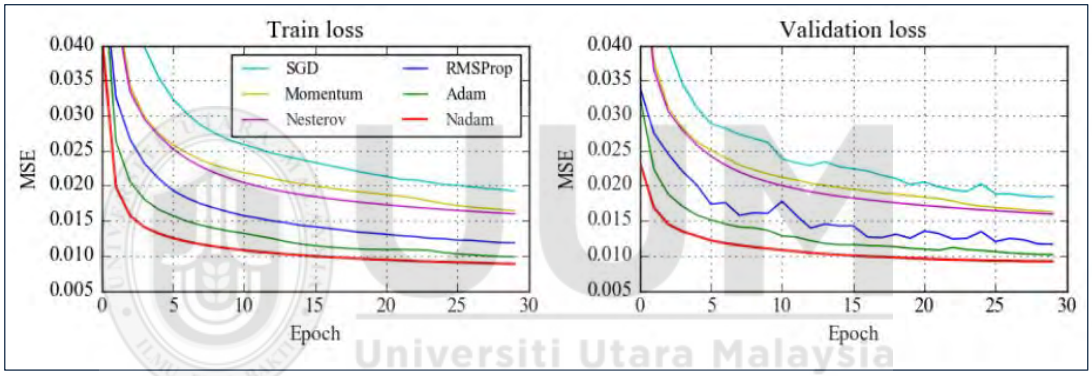

Figure 2.10. Training and validation loss of different optimisers on the MNIST dataset 5 (Dozat, 2016)

Similarly, the Nadam optimiser has demonstrated superior performance across various machine learning and deep learning tasks. Some of the areas where Nadam has demonstrated its effectiveness include training deep neural networks, including CNNs and RNNs (recurrent neural networks). In computer vision tasks like image classification (Khan et al., 2021), sentiment analysis (Luo et al., 2021), machine translation, and text generation (Bago et al., 2022), it has demonstrated superior convergence speed and performance compared to other optimisers.

(2022) investigates the use of LSTM neural networks as a novel method for long-term solar photovoltaic (PV) power generation prediction. The Nadam optimiser, a variation of the Adam optimiser that includes the Nesterov momentum and enables finer network weight tuning, is exclusive to this study. In particular, this method seeks to improve the accuracy and efficiency of solar power generation forecasting.

Zhang et al. (2023) introduced a novel technique to strengthen adversarial attacks in deep neural networks: the Nadam Iterative Fast Gradient Method (NAI-FGM). The Nadam optimiser, renowned for its quick convergence and practical gradient computation, is integrated into the adversarial attack technique via NAI-FGM. This integration enhances the efficacy and transferability of adversarial attacks, particularly in black-box settings. The paper makes a substantial contribution to adversarial machine learning by proving the superiority of NAI-FGM over conventional attack techniques. A text classification model that combines the CNN and Transformer architectures using the Nadam Optimiser was presented by Xie et al. (2022). The model's Nadam Optimiser is a crucial component. An improved version of the Adam optimiser is provided by incorporating Nesterov momentum. This enhancement helps better control the learning rate and directly affects the gradient update, thereby improving the efficiency and effectiveness of the text classification model.

Kuppusamy et al. (2023) presented a custom convolutional neural network (CNN) for emotion recognition, a present-based model for detecting emotions from human facial expressions. The study compares the performance of two optimisation algorithms, Adam and Nadam, on the FER-2013 dataset, and the experimental results show that the custom CNN model with the Nadam optimiser performs the best in training and

testing, achieving a classification accuracy of 84.1%. The study demonstrates the importance of optimiser selection on the performance of emotion recognition models for applications such as smart healthcare, robotics, and surveillance. Ezgi & Onan (2023) investigate methods for automated assessment incorporating deep neural networks for grading the severity of knee osteoarthritis (OA). By comparing the performance of five network architectures, VGG-16, VGG-19, ResNet-101, EfficientNet-B6, and EfficientNet-B7, as well as five optimisation algorithms, SGD, Adam, Nadam, AdamW, and AdaDelta, the experiments found that EfficientNet-B7 combined with the Nadam optimiser achieved the highest grading accuracy (70.1%). The results show that deep learning techniques have great potential for automated grading of knee OA, providing a reliable and efficient tool for clinicians and researchers alike. Table 2.28 summarises the Nadam optimiser review.

Table 2.28

Nadam as an pptimiser review

Author	Tasks	Comments	Dataset
Sharma et al. (2022)	PV power prediction approach	Nadam outperforms RMSprop, Adam, Adamax, and SGD, respectively.	Energy consumption data
Zhang et al. (2023)	Adversarial attacks	Nadam increases the success rate by 8% to 11% compared to the other attack techniques.	CIFAR-10 and CIFAR-100
Xie et al. (2022)	Mobile internet & information transmission bridge in life	Nadam + CNN + Transformer model. Compared to the baseline model, the model improves the text and classification evaluation index.	Detecting fraudulent credit card transactions

Table 2.28 continued

Author	Tasks	Comments	Dataset
(Ezgi & Onan, 2023)	automatic knee OA severity-grading system	EfficientNet-B7, combined with Nadam optimiser, achieved the highest grading accuracy (70.1%)	Knee osteoarthritis disease curated dataset

The Nadam is a popular optimisation algorithm used for training neural networks. While it can be effective across various machine learning tasks and has demonstrated its excellence as an optimisation technique for time series predictions with non-linear data (Wang et al., 2022). This excellence may be attributed to its effectiveness in training deep neural networks with uncomplicated implementation. It was established that Nadam is a good default choice for many tasks and applications in different domains. However, despite its many strengths, Nadam as an optimiser has some weaknesses of its own, as follows:

One significant drawback of Nadam is that it can be challenging to handle gradient explosion in some situations, especially when working with deep neural networks and time-series prediction tasks. Nadam may struggle to stabilise gradients when backpropagation causes gradients to grow extremely large. Time series data, especially ammonia nitrogen levels in aquaculture, are crucial for monitoring parameters and managing water quality, which are crucial for the health and well-being of aquatic organisms such as fish. Along with ammonia nitrogen levels, several other parameters are commonly measured to assess environmental conditions in aquaculture systems. The specific parameters of interest may vary depending on the type of aquaculture. The species being cultured, temperature, pH level, and oxygen might contain extreme values that can lead to large gradients during training. Nadam's gradient mechanisms

may not always be robust enough to handle such extreme values effectively (Lewkowycz et al., 2020; Ceesay et al., 2023; Nie et al., 2019). Therefore, to eliminate the limitations of expert experience, save time spent on parameter modification, and improve the training stability and prediction accuracy of models (e.g., LSTM), this study proposes combining gradient clipping with the Nadam optimiser as an enhancement strategy to mitigate the gradient spike problem, and names it NadamClip.

2.6 Gradient Clipping Stochastic Gradient Descent Algorithm

In time-series prediction tasks, gradient explosion is a critical challenge (van Dijk et al., 2022) that needs to be addressed. This is because gradient explosion may lead to training instability and a fall into a local optimum (Shi et al., 2013). For that matter, during training neural networks such as LSTMs, gradients may become too large, causing numerical instability and making it challenging for optimisation algorithms such as Nadam to avoid local minima. Mitigating the exploding gradient problem is necessary to improve generalisation performance.

Gradient clipping is a technique used in neural network training, particularly useful for addressing the issue of exploding gradients (Zhang et al., 2020). Exploding gradients occur when the gradients during training become too large, leading to unstable network behaviour (Kanai et al., 2017). Gradient clipping involves setting a threshold, and if any gradient exceeds it, it is scaled down to keep its magnitude within a defined range (Kanai et al., 2017). This process ensures that gradients do not become excessively large, thereby stabilising the training process. It is a simple yet effective way to prevent destabilisation of the training process in deep learning models (Zhang et al., 2020).

In their seminal study, Abadi et al. (2016) first proposed the Differentially Private Stochastic Gradient Descent (DPSGD) algorithm. This algorithm is based on SGD and protects data privacy by introducing gradient clipping and noise addition. It also ensures that the training process satisfies the (ε, δ)-differential privacy requirement through the addition of Gaussian noise and the clipping of the gradient parameter. DPSGD is applied to the image classification task and is especially validated on the MNIST and CIFAR-10 datasets. The experimental results show that DPSGD achieves 97% of the tested accuracy on the MNIST dataset (privacy budget $\varepsilon = 1$), while on the CIFAR-10 dataset, the test accuracy is about 70-75%. Although DPSGD provides privacy protection, gradient clipping introduces bias, affecting model performance and convergence speed.

(2024) introduced the Clipped Reweighted AdaGrad with Delay (Clip-RAdaGradD) method, which integrates gradient clipping into AdaGrad to address heavy-tailed noise in stochastic gradients. The approach was employed in optimisation trials for NLP tasks, namely, fine-tuning the ALBERT Base v2 model on the CoLA and RTE datasets. In the study, gradient clipping was applied using predetermined values, with some optimised by experimentation; for instance, on the CoLA dataset, the optimal threshold for layer clipping was 1, while the optimal threshold for coordinate clipping was 0.02. The experimental results indicate that the method demonstrates superior stability and a greater likelihood of convergence in noisy conditions compared to the uncropped version; for instance, on the CoLA dataset, the error range is considerably reduced with outstanding performance.

Sun, Yu et al. (2024) introduced an enhanced Adam algorithm, termed Control Restart Strategy Gradient Norm Joint Clipping Adam (CG-Adam), that integrates gradient-parameter joint clipping with a regulated restart strategy to address optimisation challenges in medical image classification. The research employs a manually established clipping threshold for gradient clipping, constraining the L2 norm of the gradient to a designated range, thereby preventing gradient spike or vanishing while simultaneously expediting the convergence of the optimisation process. The algorithm was tested on the MNIST, CIFAR-10, and stomach medical picture datasets. The experimental findings indicate that CG-Adam attains 98.59% accuracy on the MNIST dataset (compared to Adam's 98.52%), 70.7% accuracy on the CIFAR10 dataset (a 1.15% improvement over Adam), and 73.2% accuracy on the stomach dataset (a 3.4% improvement over Adam). These results indicate that CG-Adam enhances the model's generalisation while also markedly accelerating convergence.

Sun, Cui, et al. (2024) introduced an enhanced Adam optimisation approach, termed AdamW with Adaptive Gradient Clipping (AdamW_AGC), which integrates Weight Decay with Adaptive Gradient Clipping (AGC) methodologies for medical picture classification, particularly in intricate gastrointestinal image classification. Gradient clipping uses a dynamic adaptive method that adjusts the clipping threshold based on the ratio of the gradient to the parameter, addressing issues of exploding and vanishing gradients while enhancing model training stability. The tests use the HyperKvasir dataset (medical images of the gastrointestinal tract) alongside the standard MNIST and CIFAR-10 datasets. The experimental results indicate that on the HyperKvasir dataset, AdamW_AGC attains a classification accuracy of 75.8%, surpassing the conventional Adam by 1.6%. On the MNIST dataset, the accuracy is 98.69%,

surpassing other optimisation techniques; on the CIFAR10 dataset, AdamW_AGC achieves 71.7%, further substantiating its outstanding performance on difficult, high-dimensional data.

Elesedy & Hutter (2023) proposed On-Average Unbiased Stochastic Gradient Clipping (U-Clip). U-Clip is a further improvement on standard gradient clipping that solves the bias problem introduced by gradient clipping by introducing a carry buffer mechanism. The method accumulates cropped gradient parts into the next update, effectively reducing the clipping bias and ensuring an unbiased gradient update. U-Clip's efforts primarily focused on image categorisation, using the CIFAR-10 and ImageNet datasets. In the CIFAR10 dataset, U-Clip paired with SGD demonstrates superior optimisation stability and efficiency compared to the baseline technique, particularly at small batch sizes ($\leq$ 100). On the ImageNet dataset, U-Clip demonstrates performance comparable to the base optimiser, such as SGD, with processing power.

Comparisons with sub-gradient-based clipping, autoclip, clipped-SSTM, Batch Clipping, adaptive layer-wise clipping, and unitary clip, following Table 2.29.

Table 2.29

Clipping techniques

Author	**Method**	**Dataset**	**Measurement**	**Gradient threshold setting**
Abadi et al., (2016)	DPSGD	MNIST and CIFAR-10 dataset	MNIST dataset Accuracy = 97 % CIFAR-10 accuracy=70-75%	manually fixed (3 or 10)

Table 2.29 continued

Author	Method	Dataset	Measurement	Gradient threshold setting
Chezhegov et al. (2024)	Clip-RAdaGradD	CoLA, RTE	For CoLA: Achieved stable validation loss with error bounds between the 5th and 95th percentiles	Manually fixed values (e.g., 1 for layer-wise clipping, 0.02 for coordinate-wise clipping)
Sun et al. (2024)	CG-Adam	MNIST, CIFAR10, Stomach	MNIST: 98.59% accuracy; CIFAR10: 70.7% accuracy; Stomach: 73.2% accuracy; faster convergence	Manually fixed (e.g., clip_value = 1 for gradient norm)
Sun et al. (2024)	AdamW_AGC	HyperKvasir, MNIST, CIFAR10	HyperKvasir: 75.8% accuracy; MNIST: 98.69% accuracy; CIFAR10: 71.7% accuracy; improved stability and generalisation	Adaptive Gradient Clipping
Elesedy, Bryn et al. (2023)	unitary clip	CIFAR10 and ImageNet	With SGD, U-Clip reaches 99% training accuracy.	Clipping region set as a constant value, adaptive based on gradient mean and variance, or proportional settings

In summary, while methods such as DPSGD, Clip-RAdaGradD, CG-Adam, unitary clip, and AdamW_AGC have been proposed to enhance training stability by incorporating gradient clipping mechanisms. In recent years, the combination of gradient clipping and momentum methods has also demonstrated significant effectiveness in addressing training instability, with corresponding theoretical understanding gradually maturing. Li et al., (2023) provide a high-probability analysis of clipped SGD, momentum, and adaptive methods under non-convex settings, showing that clipping gradients before entering momentum ensures stability even when only finite α-moment assumptions hold. Their results demonstrate that clipping significantly improves robustness and generalisation in heavy-tailed noise regimes. B. Zhang et al. (2020a) develop a unified framework that clips the combined update direction of momentum-based methods, achieving convergence rates comparable to those of standard first-order algorithms under (L_0, L_1)-smoothness. Their experiments show that mixed clipping strategies outperform both unclipped SGD and conventional gradient-only clipping. Pethick et al. (2025) extend clipping–momentum techniques to non-Euclidean geometries by generalising the (L_0, L_1)-smoothness condition and introducing a clipped momentum estimator. They establish an order-optimal $O(n^{-1/4})$ convergence rate, highlighting the method's stability and applicability under generalized norms and heavy-tailed noise. But a key limitation remains: none of these algorithms incorporates Nesterov momentum into their update rules. Nesterov momentum is known for its look-ahead property, which allows the optimiser to anticipate the future position of parameters based on current momentum. This predictive capability leads to more informed and adaptive gradient updates, thereby improving convergence behavior and often resulting in better generalisation performance. The absence of this mechanism in the above optimisers represents a

significant gap, particularly for complex models such as LSTMs, where precise and stable gradient updates are critical.

2.7 Gap Analysis

The use of deep learning has significantly expanded, particularly in model optimisation using algorithms such as SGD, Adam, and Nadam. Despite these advances, challenges such as gradient spikes remain prevalent, especially in recurrent neural networks (RNNs), where gradients can become excessively large during training and lead to numerical instability (Zhang et al., 2020).

To mitigate this, gradient clipping is a common strategy that constrains the magnitude of gradients during optimisation, thereby improving training stability. Several recent approaches, such as DPSGD, Clip-RAdaGradD, CG-Adam, Unitary Clip, and AdamW_AGC, have extended gradient clipping techniques to optimisers including SGD, AdaGrad, Adam, and AdamW. However, a common limitation of these methods is the absence of Nesterov momentum, which plays a crucial role in optimisation by enabling the algorithm to anticipate future gradient directions. The lack of this forward-looking mechanism can limit prediction accuracy.

In contrast, Nadam combines the benefits of Adam with Nesterov acceleration, resulting in more accurate and stable updates. However, to date, no research has integrated gradient clipping directly into Nadam's internal mechanism, which creates an important gap, particularly in the training of deep models such as LSTMs, where both precision and gradient stability are critical.

To address this, the present study proposes a novel algorithm, NadamClip, that embeds gradient clipping within the Nadam optimiser and determines an optimal clipping range. The objective is to enhance model performance by simultaneously improving prediction accuracy and suppressing gradient spikes. The proposed method is empirically evaluated on an LSTM-based ammonia prediction task in aquaculture and compared against both conventional optimisers (e.g., Adam, SGD, RMSProp) and gradient-clipping-enhanced optimisers (e.g., SGD with clipping).

2.8 Summary of the Chapter

This chapter examines studies on the resolution of prediction problems using various methodologies, including artificial intelligence and statistical techniques. Owing to their nonlinear long-term series structure, LSTM models may be more effective for predictive modelling. The training procedure for LSTM requires optimisation using various stochastic gradient descent methods, including SGD, Adam, and Nadam. Nonetheless, stochastic gradient descent optimisation techniques possess certain limitations. They may encounter the problem of local optima and gradient spikes while seeking optimal parameters for RNN models such as LSTMs, leading to unstable model training and diminished prediction accuracy. Consequently, an effective optimisation strategy is essential to enhance LSTM's predictive capability in these tasks. This study reviews extensive research on shearing strategies for optimisation algorithms to address gradient spikes during neural network training, highlighting their merits and limitations and identifying existing gaps.

CHAPTER THREE
METHODOLOGY

3.1 Introduction

This chapter delineates the methods to be utilised in the planned research. The research will follow a systematic methodology comprising four phases, as depicted in Figure 3.1. The stages comprise four essential tasks: data collection and preparation, initialising NadamClip-LSTM, implementation, and evaluation. The streamlined version of the suggested methodology is depicted as a process in Figure 3.1 to elucidate the procedures for enhancing the Nadam optimiser.

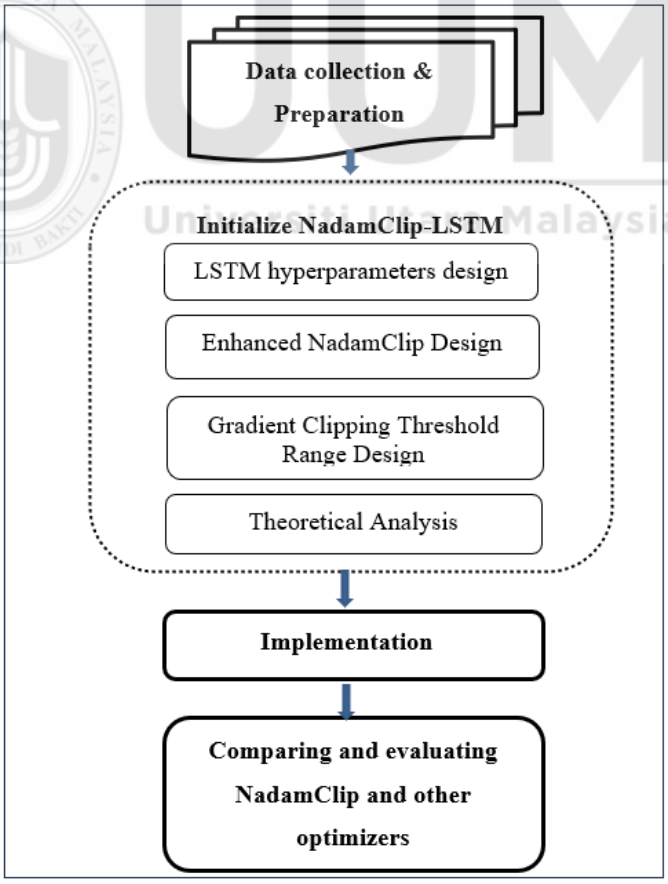

Figure 3.1. Research process

The research is mainly divided into three stages:

In the initial phase, the target research areas are delineated, and the research gaps are identified. This phase encompasses identifying the research significance and issues, and analysing and synthesising pertinent methodologies. A thorough literature review is conducted to analyse existing research findings and technologies to identify unresolved issues and opportunities for improvement.

The second phase involves data collection, commencing with the acquisition of current secondary aquaculture data from open data platforms such as Kaggle and the collection of real-time data from sensor-based aquaculture systems, including temperature, Dissolved Oxygen (DO), ammonia, and pH levels. The data undergo pre-processing, including denoising, correlation testing, and normalisation, to ensure quality and consistency.

The third phase is the model design phase. During the model design phase, the work is systematically divided into three components tailored to distinct research objectives. The following sections introduce these three components in detail.

Firstly, to address potential stability issues of the traditional Nadam optimiser under extreme gradient fluctuations, this paper designs a novel optimisation algorithm, NadamClip, which integrates a gradient clipping mechanism. Building upon the Nadam optimiser, NadamClip incorporates internal gradient clipping to mitigate gradient spikes, thereby enhancing model training stability and predictive accuracy.

Subsequently, to achieve efficient gradient clipping, this phase involves the systematic design of the clipping range selection. Multiple candidate gradient clipping ranges are configured as initial settings and sequentially applied during model training. The model's MSE and RMSE performance is recorded for each experimental group. Through comparative analysis of performance metrics across different clipping ranges, the optimal clipping range is ultimately identified. This provides a parameter basis for the NadamClip algorithm and enhances its practical utility.

Finally, to comprehensively evaluate the practical efficacy of NadamClip, this paper applies it to train and validate an LSTM network. This phase constructs multiple comparative experimental groups. Evaluation is conducted through regression metrics, loss behaviour, architecture comparisons, and generalizability testing, resulting in a comprehensive performance assessment report. This provides empirical evidence demonstrating NadamClip's optimisation capabilities and generalisation properties.

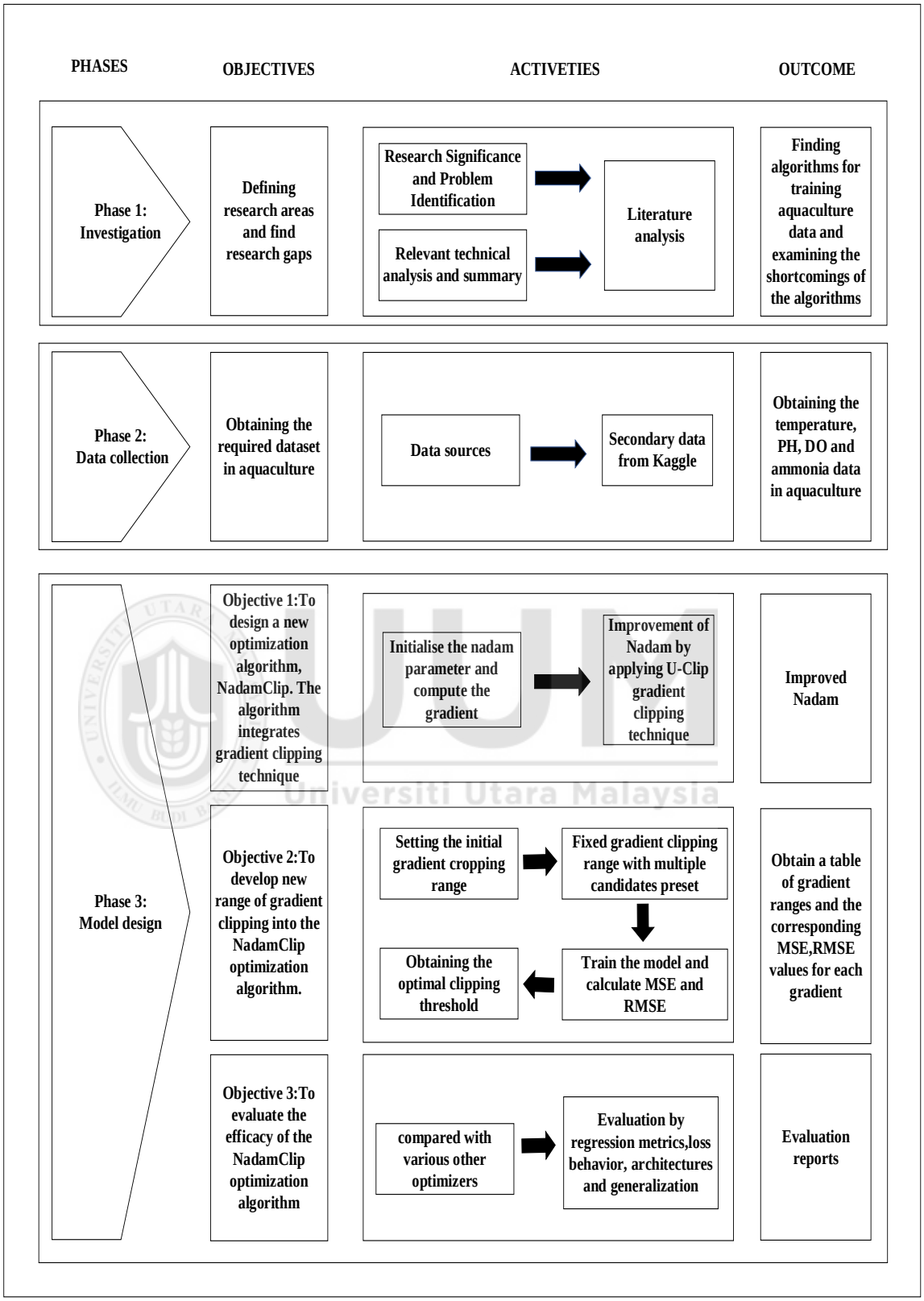

Figure 3.2. Research design flowchart

3.2 Methodological design

The proposed NadamClip-LSTM algorithm comprises three primary phases: the Data Pre-processing Phase, the initialisation of the NadamClip-LSTM Phase, and the Evaluation Phase.

1. Data preprocessing phase:

Before commencing model training, systematic preprocessing is performed on the raw data. This process includes dataset importation, cleaning, outlier or noise removal, analysis of variable correlations, and normalisation. Data cleansing and standardisation effectively enhance the quality of input data, provide a more robust numerical range and feature set for subsequent model training, thereby improving training efficiency and predictive performance.

2. Initialise NadamClip-LSTM:

Following data preprocessing, the model training process consists of several key steps. First, the structure and hyperparameters of the NadamClip-LSTM model are initialised, including the learning rate, batch size, number of iterations, and the gradient clipping threshold range. Subsequently, model parameters are initialised, and the training phase begins.

During each iteration (epoch), a forward pass is first executed to compute the prediction and loss for the current batch. Then, gradients are computed during backpropagation, and a gradient clipping strategy is applied to mitigate gradient spikes. The clipped gradients are then used to update the momentum estimates (such as the

first and second moments) within the optimiser, thereby completing the model parameter update. This process continues iteratively until the predefined number of training epochs is reached. Upon completion of training, evaluation metrics (e.g., RMSE, MAE, R^2) and training/validation loss curves are recorded, providing the basis for assessing model performance.

3. Evaluation phase:

Upon completion of model training, its performance is evaluated from multiple perspectives. The evaluation includes comparative analyses at several levels: Firstly, NadamClip is compared against other mainstream optimisers (e.g., Adam, Nadam, SGD) and similar optimisers (e.g., DPSGD, Clip-RAdaGradD) using RMSE, MAE, and R^2 metrics to examine its prediction accuracy. Secondly, the descent performance of each optimiser on the loss function is further compared. Additionally, a horizontal comparison is conducted between the LSTM model and other neural network architectures (e.g., ARIMA, GRU and TFT) regarding their predictive capabilities on the same task. Finally, the generalizability and potential application value of NadamClip in other tasks are investigated by training the model on datasets from other domains. Illustrate the proposal model in accordance with the aforementioned procedures, as depicted in Table 3.3.

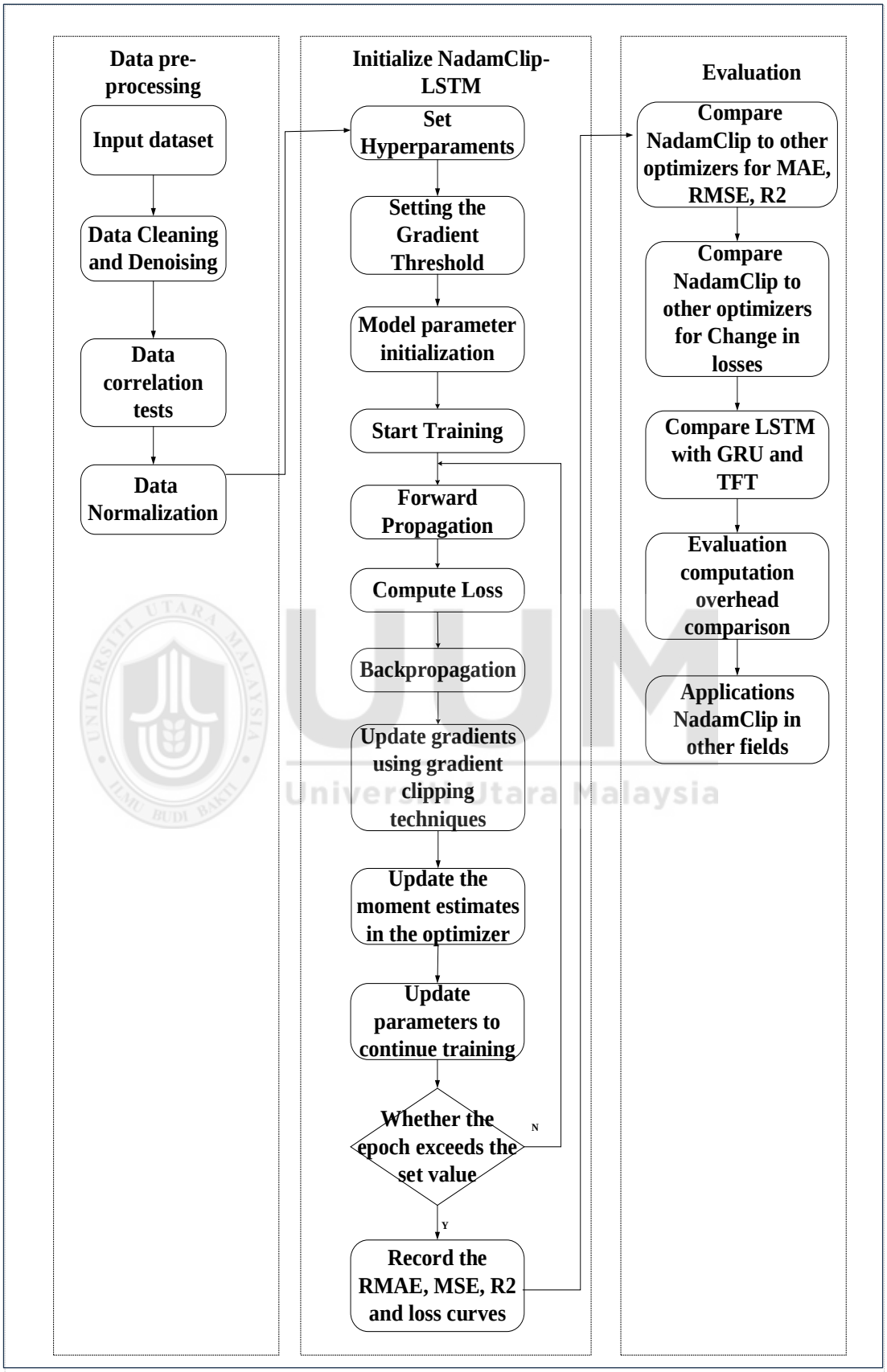

Figure 3.3. Training flowchart

3.3 Data Pre-processing

This proposal will use one set of time-series data, referred to as dataset P1. The primary time-series dataset for this study is a single-pond dataset, specifically P1, sourced from Kaggle. This dataset is an IoT-tagged compilation for monitoring water quality in aquaculture fish ponds, collected by the HiPIC Research Group at the Department of Computer Science, Nsuka University, Nigeria. The datasets were collected using the following water-quality sensors controlled by an ESP32 microcontroller: Dallas Instruments Temperature Sensor (DS18B20), DF Robot Turbidity Sensor, DF Robot Dissolved Oxygen Sensor, DF Robot pH Sensor V2.2, MQ-137 Ammonia Sensor, and MQ-135 Nitrate Sensor. The initiative is financed by the 2020 Lakuna Prize for Agriculture in sub-Saharan Africa and administered by the Meridian Institute in Colorado, USA. The dataset and results comprise sensor measurements from June to mid-October 2021. The specifications and attributes of the pond datasets equipped with sensors are presented in Table 3.1.

Table 3.1

Datasets template

Datasets	Historical	Input	Symbol
P1	19 June to 31 October 2021	Temperature	TP
		Dissolved Oxygen	DO
		potential of hydrogen	PH
		Ammonia	Ammonia

This dataset offers a robust basis for examining the prediction of ammonia levels in aquaculture. The dataset encompasses other variables, not solely ammonia content. This multivariate data structure facilitates advanced statistical analyses and machine

learning modelling to investigate the correlation between ammonia content and variables such as temperature, dissolved oxygen, and pH. This enables scientists and researchers to develop multidimensional predictive models that enhance the precision of ammonia concentration forecasts.

Furthermore, a significant characteristic of the dataset is its ability to sample at high frequency. The timestamps indicate that the data were taken at minute or even second intervals, and this high-resolution data can precisely monitor trends in ammonia concentrations. This time-series data is ideal for modelling as it reflects short-term dynamic correlations between water quality measures and ammonia concentrations. The data's origin from actual fish pond testing settings enhances its practical relevance. This data accurately reflects actual water quality characteristics and provides significant insights for effective aquaculture management.

In summary, this dataset, featuring comprehensive records of water quality parameters, authentic experimental context, and high-frequency sampling, offers an extensive database for predicting ammonia concentration and serves as a significant resource for aquaculture scientific research and practical application.

Dataset P1 has 83,127 entries and is accessible in the Sensor-Based Aquaponics Fish Pond Dataset. Optimal conditions for aquaculture include temperatures between 20 and 30°C, dissolved oxygen levels of 5-8 mg/L, and pH values between 6.5 and 8.5. Safe limits for prolonged exposure are below 0.2 mg/L for total ammonia and between 2.9 and 5.0 mg/L for acute toxicity. A sample of the dataset is presented in Table 3.2.

Table 3.2

Sample of data set P1

Date	TP	DO	pH	Ammonia
2021-06-19 00:00:05 CET	22.94504	4.219706	12.75389	0.45842
2021-06-19 00:01:02 CET	22.99544	6.313918	12.79817	0.44741
2021-06-19 00:01:22 CET	22.8894	15.50348	14.79148	0.40778

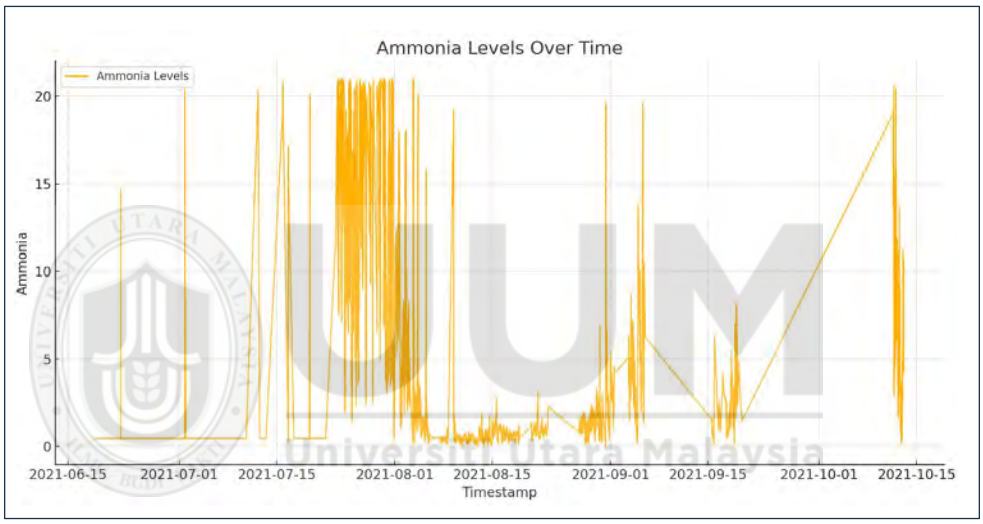

Figure 3.4. P1 data set ammonia visualisation

Figure 3.4 illustrates the continuous recording of ammonia concentrations over time, with time on the horizontal (X-axis) axis and ammonia concentration on the vertical (Y-axis) axis. A fundamental characteristic of time-series data is the chronological ordering of observations, in which each data point in this figure is distinctly associated with a timestamp, illustrating the trend in ammonia concentration over various temporal intervals. Furthermore, the data illustrated in the image reveal distinct volatility and periodicity, with ammonia concentrations exhibiting significant fluctuations and stabilising at different temporal phases, a phenomenon commonly

observed in time-series data. Consequently, owing to the robust correlation between ammonia concentration and time, along with the data's temporal and dynamic properties, it can be classified as time series data.

For time series model selection, when the forecasting task emphasises short-term fluctuations (e.g., concentrations over minutes or hours), the GRU is a more advantageous option due to its lower parameter count and effective handling of short-term dependencies. This study must model the potential impacts of other periodic factors (e.g., temperature or pH value) on ammonia concentrations; hence, LSTM is more suitable for complicated long-term dependency tasks in which temporally distant observations exert non-linear and multi-factor influences on subsequent states, requiring the preservation and integration of information across extended temporal horizons, as in this investigation.

3.2.1 Data Cleaning and Denoising

This dataset pertains to aquaculture environmental monitoring and includes critical metrics such as temperature, dissolved oxygen, pH, and ammonia nitrogen content. Data cleansing and denoising are vital in aquaculture, as precise and reliable data are necessary to ensure the aquatic environment is conducive to aquaculture (Hu et al., 2020). Data cleaning primarily involves correcting or eliminating erroneous records, including outliers caused by sensor malfunctions or reading inaccuracies, as well as addressing missing data due to transient sensor failures or communication issues (Karkouch et al., 2016). Denoising mitigates random variations induced by environmental changes or sensor sensitivity, thereby refining the data better to reflect the actual condition of the aquatic environment (Campisi-Pinto et al., 2012).

Table 3.3 outlines the key steps of this process in pseudocode, including: removing redundant attributes, identifying and eliminating missing or duplicate entries, performing index normalisation, and validating the dataset's consistency. These operations collectively produce a clean, reliable dataset for subsequent analysis.

Table 3.3

Data cleaning pseudocode

Algorithm 3.1: Pseudocode for the Data Cleaning

Input: Raw dataset D

Output: Cleaned dataset D_clean

```
    D ← drop_columns(D, ["entry_id"])
    missing_count ← count_missing(D)
    duplicate_count ← count_duplicates(D)
    D ← drop_missing_rows(D)
    D ← drop_duplicate_rows(D)
    D ← reset_index(D, drop=True)
    missing_count_final ← count_missing(D)
    duplicate_count_final ← count_duplicates(D)
return D as D_clean
```

To enhance the reliability of the time-series data and reduce the influence of short-term fluctuations, a denoising procedure was applied before conducting further analysis. This process involved sorting the dataset chronologically, generating a working copy for smoothing operations, and applying a moving-average filter to key target variables to reduce noise while preserving underlying trends. Table 3.4's pseudocode outlines the data denoising workflow, summarising the core steps for generating clean, smoothed time-series datasets ready for analysis.

Table 3.4

Data cleaning pseudocode (Ni et al., 2024)

Algorithm 3.2: Pseudocode for the Data Cleaning

Input: Raw time-series dataset D

Output: Smoothed and cleaned dataset D_smooth

D ← sort_values(D, column="created_at")

D_smooth ← copy(D)

for each target variable v in {Temperature, Dissolved_Oxygen, pH, Ammonia} do

D_smooth[v] ← moving_average(D_smooth[v], window=3)

end for

D_smooth ← drop_missing_rows(D_smooth)

return D_smooth

3.2.2 Test for Correlation

Correlation analysis is crucial for predictive modelling, as it elucidates the interrelationships among variables and provides a scientific foundation for model development and feature selection (Gogtay & Thatte, 2017). It helps identify the principal features that are strongly associated with the target variables and enhances the model's predictive capacity (Obilor & Amadi, 2020). During the correlation testing phase, we employed the Spearman Correlation Coefficient method to examine the relationship between the variables. This approach relies on variable ranks and therefore does not depend on assumptions of linear relationships or normality between variables. The correlation matrix was computed to assess the degree of correlation among the numerical variables (e.g., Temperature, Dissolved Oxygen, pH, and Ammonia) in the time-series data. Thereafter, the correlation matrix was depicted as a heat map, where the colour gradients represent the magnitude and orientation of the correlation between

variables (positive or negative) (Schloss et al., 2019). This method enables the rapid identification of potential correlations between variables, serving as a foundation for later modelling and feature selection. The step is shown in Figure 3.5.

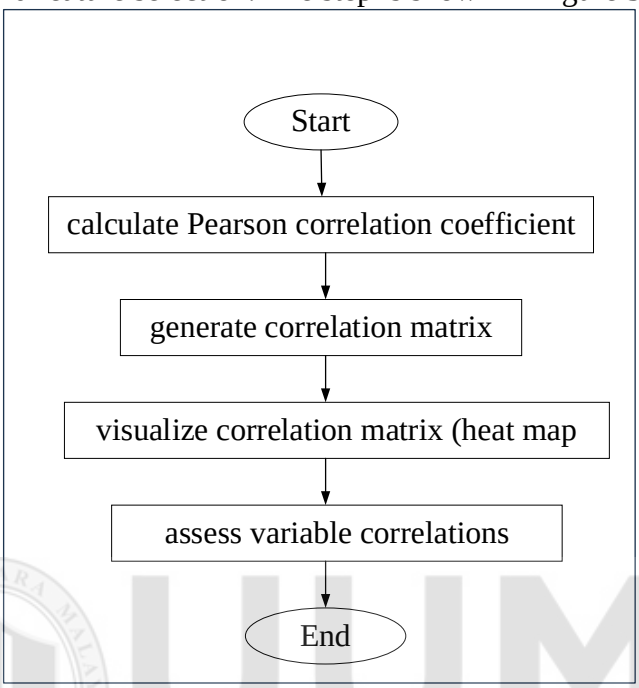

Figure 3.5. Test for correlation steps

3.2.3 Data Normalisation

The quality of the data utilised by machine learning algorithms dictates their efficacy (Masthurah et al., 2021). Consequently, data preparation techniques, such as data normalisation, are essential for standardising the values of input and output variables within a defined range. This normalisation ensures that larger input values do not disproportionately affect smaller ones, which could otherwise impede the training of predictive models. Consequently, it improves the predictive accuracy and efficiency of recurrent neural networks (Attrapadung et al., 2022). This inquiry will utilise Min Max Normalisation, as outlined by Yaqub et al. (2020), with its definition presented in formula 3.1. Min–Max normalisation is used because the LSTM model is sensitive to input scale and performs better when features are mapped to a fixed [0,1] range.

Unlike Z-score standardisation, which mitigates the influence of extreme values, it implicitly assumes a near-Gaussian distribution, which the present dataset does not exhibit (Moeller, 2025). Min–Max also preserves the temporal structure of the data, making it a more suitable choice for this time-series task.

$$\text{Normalized value} = \frac{\text{Value} - \text{Min}}{\text{Max} - \text{Min}} \tag{3.1}$$

This formula normalises the range of characteristics to [0, 1] or [−1, 1]. It is especially advantageous when standard deviations are minimal or when maintaining zero entries in sparse data is crucial.

3.4 Initialize NadamClip-LSTM

This phase delineates the procedure for configuring the model's hyperparameters. The NadamClip algorithm is subsequently refined, the training process of the LSTM model utilising this algorithm is delineated, and ultimately, the optimum gradient range is introduced.

3.4.1 Hyperparameters Design

This study conducted hyperparameter searches for the number of layers, number of neurons, learning rate, and batch size, the primary factors influencing model performance. This also diminishes the search dimensions, optimisation duration, and resource utilisation. For the hyperparameters requiring optimisation in this work (e.g., number of layers, number of neurons, learning rate, and batch size), which are high-dimensional, $4^4 = 256$ model training is necessary. Consequently, the dataset of this study has a high parameter dimensionality and a moderate cost of model training. In

this study, random search was selected for hyperparameter tuning because it is more efficient than grid search and requires fewer computational resources than Bayesian optimisation. Given the small hyperparameter space and the focus on evaluating the optimiser rather than extensive tuning, random search offers a practical and effective solution that still produces strong model performance. This work used stochastic search to optimise model hyperparameters, with the search range delineated in Table 3.5.

Table 3.5

Hyperparameter search scope

Parameter	**Set range**
layer	(1,4)
units	(10,201)
Learning rate	[0.001, 0.01,0.1]
Batch size	[8,16, 32,64]

The hyperparameter range for this dataset is selected to align comprehensively with the data properties and the LSTM model's requirements. The number of layers varies from 1 to 4, facilitating a balance between model depth and the risk of overfitting, with one to two layers appropriate for simple time series patterns, whereas three to four layers are capable of capturing more intricate temporal dependencies; the quantity of hidden units spans from 10 to 200, offering a spectrum from lightweight to moderately complex models to effectively capture feature relationships in the data while considering computational efficiency; the learning rate ranges from The learning rate varies from 0.001 to 0.1, encompassing the effective range of prevalent optimisers, thereby ensuring model convergence stability and accommodating diverse task

complexities. The batch sizes are established at 8, 16, 32, and 64, which consider the adaptability of smaller batches in memory-constrained systems while leveraging the benefits of larger batches for smooth gradient updates and enhanced computational efficiency, thus adequately addressing the dataset size and training requirements (J et al., 2021; Masters & Luschi, 2017). This can completely fulfil the dataset's size requirements and the demands of model training.

3.4.2 Enhanced NadamClip Design

The NadamClip algorithm refines the original Nadam optimiser by integrating the U-Clip gradient clipping mechanism directly into Nadam's internal update rule. Unlike approaches in which gradient clipping is applied as an external post-processing step, NadamClip constrains the gradient before it contributes to the computation of the first- and second-moment estimates or the Nesterov-accelerated update. By incorporating clipping at this stage, the adjusted gradient consistently participates in the adaptive learning rate and momentum calculations, helping suppress gradient spikes while preserving the directional integrity of the update.

Although Nadam benefits from adaptive learning rates and Nesterov momentum, it may still exhibit instability when training deep LSTM networks, particularly when gradients become excessively large or sudden spikes occur. Applying U-Clip before momentum and adaptive term updates reduces the impact of abnormal gradients on the optimisation dynamics, leading to smoother, more controlled parameter updates while retaining the convergence advantages of Nadam.

When applied to nonlinear time-series tasks, this integration of gradient clipping into Nadam's optimisation framework yields more stable convergence. By effectively regulating the gradient magnitude and mitigating early-stage gradient spikes, NadamClip demonstrates greater robustness and accuracy in training LSTM models. This makes it well-suited for complex sequence prediction tasks, such as forecasting ammonia nitrogen concentrations in aquaculture. The overall procedure is illustrated in Figure 3.6.

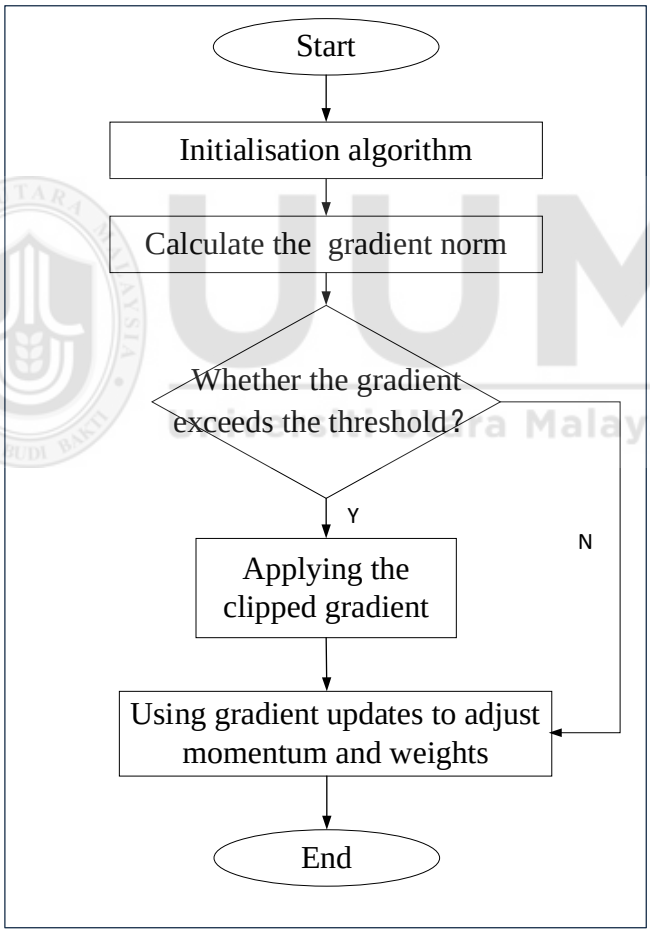

Figure 3.6. Enhanced NadamClip step

3.4.3 Gradient Clipping Threshold Range Design

To ensure the deep learning model reliably and effectively uses gradient information for parameter updates during training, it is essential to conduct an optimal gradient clipping experiment. Mitigating gradient spikes by establishing a suitable gradient clipping threshold enhances model training stability and optimises the use of gradient information, thereby elevating model performance and stability (Kanai et al., 2017).

Gradient clipping is implemented by setting a threshold that defines the maximum permissible gradient range. When the computed gradients exceed this threshold, they are normalised to that range, ensuring a steady training process. The gradient clipping threshold is a variable parameter that can be calibrated based on the particular job and the model's performance. Selecting the appropriate gradient clipping threshold may necessitate testing and adjustment (Mai & Johansson, 2021).

The optimal gradient clipping experiment table systematically documents and analyses the impact of varying gradient clipping thresholds on the model's training performance, thereby enhancing the model's stability and efficacy. The determined threshold range shall subsequently be deployed as a configurable parameter in the NadamClip optimiser to regulate gradient magnitude variations during training. This experimental procedure seeks to identify an appropriate range of gradient clipping thresholds to successfully mitigate gradient spikes and disappearances, thereby enhancing the efficiency and efficacy of model training.

The experimental table meticulously documents the training and validation outcomes across various gradient clipping thresholds, including training loss and validation loss, among others. By comparing these results, the researchers were able to intuitively

assess the impact of different settings on model performance and identify an optimal range for gradient clipping thresholds. This technique not only reduces trial-and-error time and enhances training efficiency but also improves the model's stability throughout training, mitigating potential issues such as gradient spikes and gradient vanishing.

Consequently, creating the gradient clipping optimal range test table is a crucial measure to enhance model training and improve model stability and performance. Through rigorous documentation and analysis of various gradient clipping thresholds, researchers identified the optimal range, enabling the training of more efficient and stable models. Consequently, a table of gradient clipping thresholds has been established to identify the best gradient range.

To establish an initial reference value, the gradient threshold should be set to a higher value. In this context, it can be presumed that gradient clipping is not executed during model training. Consequently, during this preliminary phase, all gradients in the training process remain unrestricted, allowing for the acquisition of the original gradient norm and the associated model performance metrics. At this juncture, the model's performance indicators, including RMSE, MAE, and R^2, must be documented as a reference for future comparisons. These metrics will guide subsequent modifications to the gradient thresholds.

The gradient thresholds are progressively diminished, and the model's performance at each threshold is documented. When modifying the gradient threshold, a suitable decrement step size should be chosen, often reduced in absolute terms, such as 0.05 or 0.1 incrementally. This guarantees that the gradient thresholds adjust precisely and

consistently throughout the experiment, hence facilitating a more precise observation of the impact of varying thresholds on model training efficacy. Following each modification of the gradient threshold, the model must be retrained, and the relevant performance measures, including RMSE, MAE, and R^2, must be documented for future examination.

After multiple experimental iterations, a gradient threshold interval that demonstrates substantial performance improvement can be identified from the trajectory of performance metrics. Gradient thresholds within these ranges generally enhance model performance considerably, so the emphasis will be on fine-tuning within that threshold band. The adjustment step size may be decreased to a more precise level, such as 0.01 or 0.005, to investigate the ideal threshold within this further range.

Ultimately, based on the experimental findings, an optimal gradient threshold is determined. The ideal gradient threshold should yield optimal model performance characterised by minimal RMSE, minimal MAE, and maximal R^2. In certain cases, multiple gradient thresholds may yield comparable performance, allowing selection of an appropriate value based on the stability and convergence rate of the training process. To ensure the chosen gradient threshold is effective, numerous training-validation cycles must be conducted to verify that it consistently improves model performance across multiple training cycles and effectively mitigates overfitting and gradient spikes.

In conclusion, by systematically decreasing the gradient threshold and meticulously documenting the model's performance, one can manually determine the optimal gradient clipping value. This strategy enhances both the model's training stability and

the efficiency of gradient updates throughout training, thereby improving the model's generalisation capability.

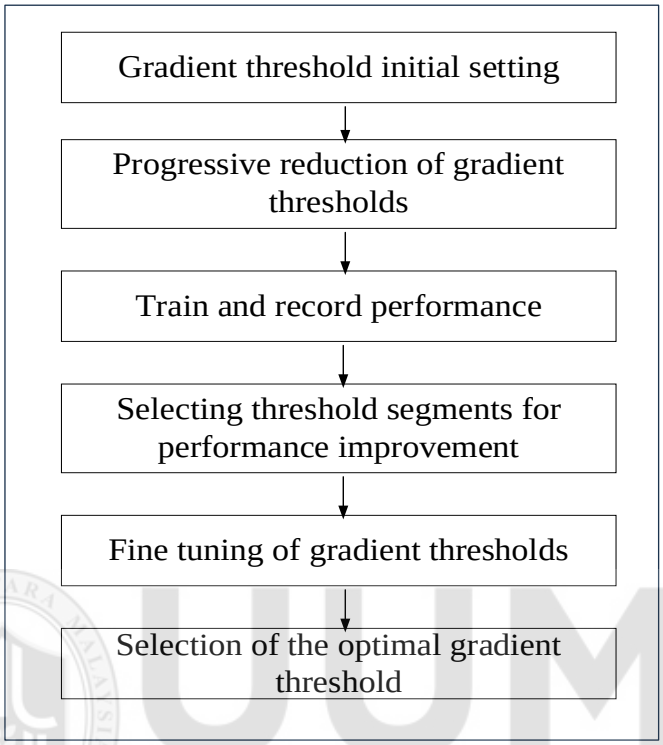

Figure 3.7. Optimal gradient clipping experiment steps

3.4.4 Theoretical Analysis Methodology

The theoretical analysis provides a structured mathematical analysis framework for understanding the NadamClip optimisation method from the perspectives of operator theory and constrained optimisation. Since NadamClip integrates gradient clipping, error compensation, and momentum-based adaptive updates, its update rule can be viewed as a composite mapping composed of several operators acting on the gradient space. Gradient clipping is formulated as a norm-constrained minimisation problem, allowing the clipped gradient to be interpreted as a projection or rescaled version of the original gradient. This projection operator acts within a bounded set, ensuring that update directions remain controlled and forming the foundation for analysing stability.

Additionally, the error compensation mechanism recursively accumulates the truncated gradient components, preserving long-term consistency of the update direction and mitigating the information loss typically associated with simple clipping.

Within this operator-based interpretation, the momentum update and adaptive moment estimation in NadamClip can be viewed as a combination of smoothing and scaling operators. Because clipping occurs before momentum accumulation, the momentum system receives gradients whose magnitudes have already been regulated, effectively suppressing the amplification of unstable signals. Meanwhile, the compensatory term gradually reintroduces clipped components, ensuring that the optimisation trajectory evolves coherently even under constrained perturbations. The continuity, non-expansiveness, and iterative behaviour of this composite operator provide the conceptual basis for analysing the algorithm's stability.

To accommodate the temporal characteristics of LSTM architectures, the analysis also considers how clipping interacts with recurrent gradient flow. In long-sequence training, clipping restricts the rapid growth of gradients along the temporal dimension, while momentum smoothing and compensation influence how information propagates across time steps. As a result, parameter updates remain within bounded regions throughout training, preventing instability induced by long-term dependencies. These structural properties provide insight into NadamClip’s behaviour in practical training scenarios and establish a theoretical foundation for discussing stability regions, threshold effects, and optimisation dynamics. The process is illustrated in Figure 3.8.

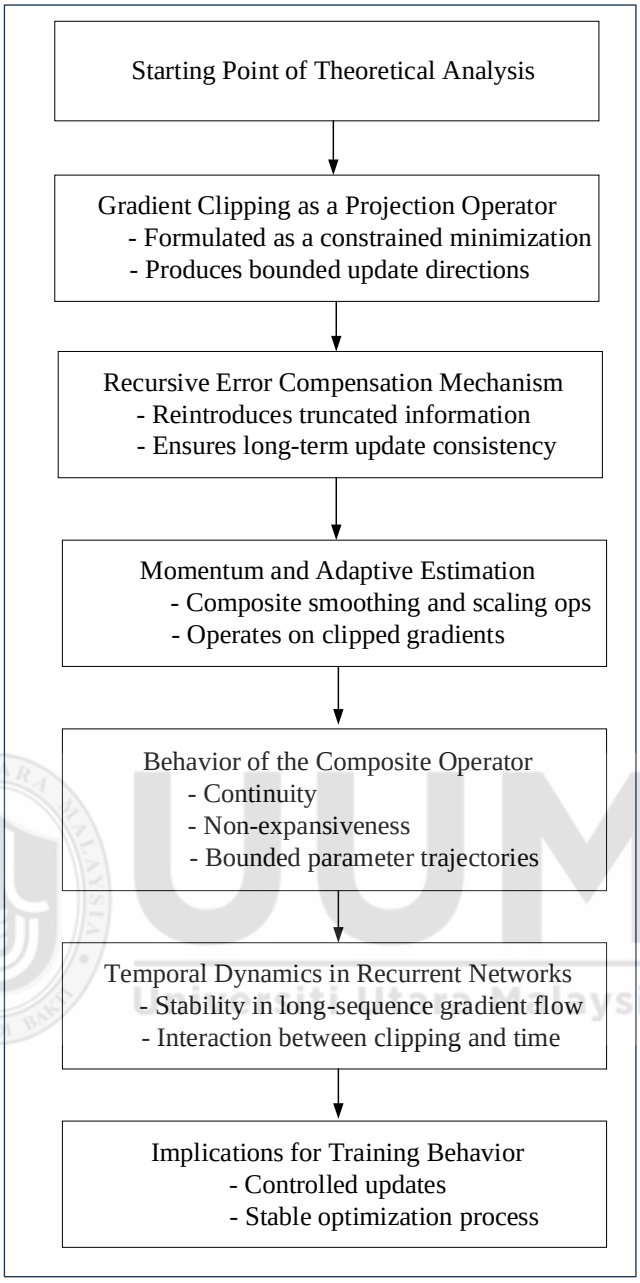

Figure 3.8. Flow of the theoretical analysis

3.5 Evaluation

The NadamClip algorithm will be evaluated using the Model Performance Evaluation Metrics, and Nadam-LSTM will be compared with various other optimisers (e.g., Adam, SGD with momentum, RMSprop) and those that integrate gradient clipping (e.g., SGD with gradient clipping).

3.5.1 Performance Metrics

This proposed study will employ three performance evaluation measures to assess the efficacy of the upgraded NadamClip algorithms across various datasets, as outlined in Chapter 3. This proposed study relies on time-series analysis, making it essential to emphasise the importance of selecting an appropriate assessment metric. The choice of a suitable evaluation metric is of utmost importance as it justifies the anticipated outcomes. This study used MAE, RMSE, and R^2 as performance metrics. MAE quantifies the average absolute magnitude of the error between expected and actual values, hence illustrating prediction accuracy. RMSE quantifies the root-mean-square of prediction errors, highlighting the influence of larger errors on model performance. R^2 is utilised to evaluate the degree to which the model accounts for the variability in the dependent variable, reflecting the model's goodness of fit (Alexei Botchkarev, 2022). The performance evaluation index formulas for MAE, RMSE, and R^2 are shown in Formulas 3.2, 3.3, and 3.4, respectively (Jongjaraunsuk et al., 2024).

$$\boldsymbol{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.2}$$

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.3}$$

$$\boldsymbol{R^2} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{3.4}$$

- y_i : actual value
- $\hat{y}_i$: model prediction
- $\bar{y}$: mean of actual values
- n : number of samples

3.5.2 Benchmarks

Utilise a baseline optimisation approach to train the LSTM and establish baseline performance. Assess training and validation performance, encompassing loss and pertinent data. Figure 3.9 illustrates the benchmark design steps.

Table 3.6

Benchmark design steps

Phase	Problam	Method	Output	Metric
Phase 1	Identification of the optimisation Challenge	Review existing algorithms for gradient clipping	Understanding of challenges in optimisation	
Phase 2	Algorithm Enhancement	Modify Nesterov Accelerated Adaptive Moment Estimation with clipping	Enhanced optimisation NadamClip algorithm and determined the range of gradient	
Phase 3	Time Series Prediction Framework Integration	Apply the enhanced algorithm to the LSTM model	Improved prediction accuracy	MAE, RMSE, R^2
Phase 4	Performance Evaluation	Compare optimizers (NadamClip & clipped variants) and models (ARIMA, GRU, TFT, LSTM) across datasets for generalization.	Performance report	MAE, RMSE, R^2

3.6 Summary

This chapter outlines the approach used to achieve the study's specified objectives. The dataset is initially delineated. The application used to train the NadamClip improvement algorithm for LSTM is delineated. The integration of the clipping approach with the Nadam optimiser to create NadamClip-LSTM will be outlined next. The metrics and benchmarking techniques employed to assess the enhancement phases of the NadamClip algorithm will be discussed.

CHAPTER FOUR
ENHANCED NESTEROV-ACCELERATED ADAPTIVE MOMENT ESTIMATION ALGORITHM

4.1 Introduction

In deep learning, particularly for intricate sequence data such as natural language processing (NLP) and temporal analysis tasks, RNNs and their variants, such as LSTMs, demonstrate significant efficacy (Muawanah et al., 2023). Nevertheless, these networks frequently encounter gradient explosion during training, particularly when processing long sequential inputs. Gradient spikes not only induce numerical instability but also significantly impair the model's training efficacy and convergence rate (Philipp et al., 2017). To effectively address this issue, gradient clipping is commonly employed in optimisation algorithms to constrain the gradient magnitude and improve training stability.

The U-Clip algorithm, which reduces cumulative variation across various gradient clipping methods, has attracted significant attention. By using the carrying vector Δt, the U-Clip method effectively retains and utilises the information that is eliminated during each gradient clipping. U-Clip incorporates the existing gradient data at each iteration, reintegrating the previously clipped segment into the current gradient. This method can reduce cumulative bias and maintain, on average, an unbiased update (Elesedy & Hutter, 2023).

Nonetheless, there is a lack of research on the implementation of the U-Clip method in sophisticated optimisers such as Nadam. Nadam is an optimisation algorithm that combines the adaptive learning rate of Adam with the Nesterov accelerated gradient

method. It exhibits superior performance and achieves convergence more rapidly than Adam (Dozat, 2016b). Therefore, it is essential to integrate the U-Clip algorithm into the Nadam optimiser and explore its potential for gradient clipping. This study's goal function is directed by RMSE and MAE. Lower RMSE and MAE values indicate superior accuracy in time-series prediction (see Chapter 3, section 3.5.1). Figure 4.1 illustrates the training process of the novel algorithm NadamClip, which integrates U-Clip into the Nadam optimiser.

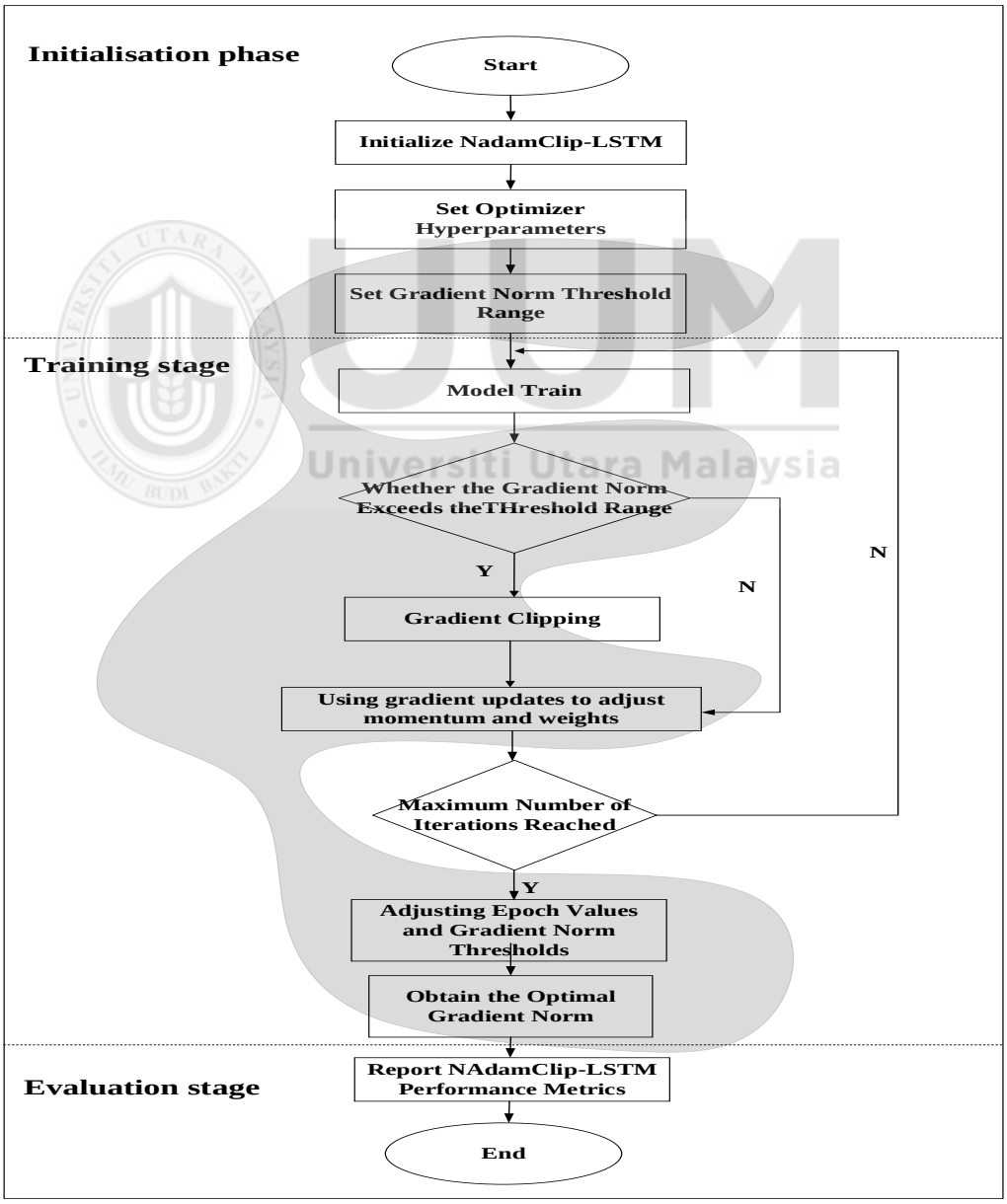

Figure 4.1. Flow of NadamClip-LSTM

Here is NadamClip-LSTM's detailed explanation of the process:

1. initialisation phase:

The NAdam Clip-LSTM model is initially instantiated. This method entails delineating the model architecture, specifying the initial parameter values, and configuring the hyperparameters required by the optimiser, including the learning rate and momentum. A gradient threshold range is established to regulate gradient clipping in subsequent training sessions.

2. Training stage:

(1) Model training: In each training iteration, data is fed into the model for forward propagation, and the difference between the model's anticipated output and the actual label is calculated. Backpropagation is executed using the errors derived by forward propagation, and the gradients of the parameters in each model layer are computed. These gradients indicate the extent to which the model parameters influence the error and are essential for adjusting them.

(2) Gradient clipping: Once the gradient is computed, it is transformed into the gradient norm to ensure that only the magnitude of the gradient is altered, while the direction remains unchanged during the clipping process. Subsequently, verify if the gradient norm surpasses the established threshold range. An excessively big gradient may render model training unstable or potentially divergent. Consequently, when the gradient exceeds the threshold, it is clipped to keep the gradient within the threshold

range, thereby ensuring training stability. If the gradient value falls inside the threshold range, the model's training proceeds.

(3) Parameter modification: Adjust the weight matrix and bias vector of the input gate, forget gate, output gate, and cell state in the LSTM model according to the Nadam specification.

(4) Iterative training: Continuously execute the aforementioned forward propagation, reverse propagation, gradient clipping, and parameter updating procedures until the maximum iteration limit is attained.

3. Evaluation stage

The performance of the NadamClip-LSTM model was assessed on an independent test set after training. The model's performance is assessed by computing metrics such as MAE and RMSE on the test set and comparing them with those of other algorithms.

4.2 New optimisation Algorithm

The Nadam algorithm, an advanced optimisation technique, integrates the benefits of Adam and Nesterov momentum methods, exhibiting rapid convergence and strong robustness. Nevertheless, when the model faces intricate optimisation challenges, such as gradient spikes or vanishing gradients, the efficacy of the Nadam approach may be compromised. This study proposes the NadamClip method to enhance the performance of the Nadam algorithm by integrating the U-Clip gradient clipping approach, which addresses gradient spikes during training.

4.2.1 Introduction of Nadam Formula

The Nadam algorithm (Dozat, 2016b), a modification of Adam and Nesterov momentum, is renowned for its adaptable learning rate and momentum adjustment. Nevertheless, it occasionally has instability during training. The integration of Nesterov momentum facilitates adaptable learning rates and momentum rejuvenation. Nadam preserves Adam's initialisation and momentum update while improving deviation correction and parameter updates via a more efficient Nesterov momentum update. This algorithm is illustrated in Table 4.1.

Table 4.1

Nesterov-accelerated adaptive moment estimation

Algorithm 4.1 Nadam

Require: Hyperparameters: $\alpha_0, \ldots, \alpha_T; \mu_0, \ldots, \mu_T; \nu; \epsilon$.

$m_0; n_0 \leftarrow 0$(first/second moment vectors)

While θ_t not converged do

$g_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$

$m_t \leftarrow \mu_t m_{t-1} + (1-\mu_t) g_t$

$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2$

$\hat{m} \leftarrow \mu \cdot m_t/(1-\mu^t) + (1-\mu) \cdot g_t/(1-\mu^t)$

$\hat{n} \leftarrow n_t/(1-\nu^t)$

$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{n}_t+\varepsilon}} \times \hat{m}_t$

end while

Return θ_t

The flow of the Nadam algorithm is shown in Figure 4.2.

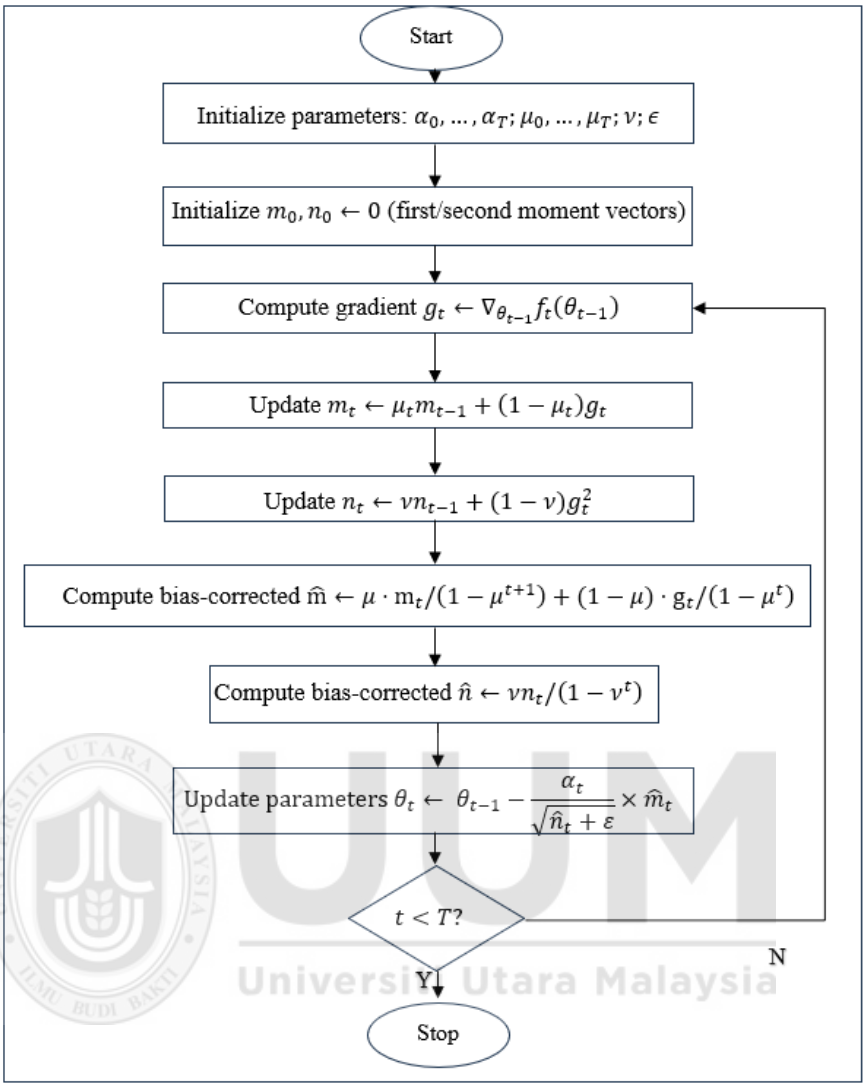

Figure 4.2. Nadam algorithm flowchart

Utilise the aforementioned flowchart to elucidate each step. Commence by initialising the settings. Subsequently, compute the gradient of the objective function with respect to the parameters. Revise the estimates for first- and second-order momentum, then compute the deviation-adjusted first- and second-order momentum estimates, and revise the parameters based on these adjusted estimates. At each iteration, the method updates the moving averages of the gradient (momentum) and the squared gradient, while mitigating estimation bias in the initial phase via bias correction. The direction and amplitude of parameter updates are influenced by the current gradient, prior

gradients, and anticipated changes in Nesterov momentum to facilitate faster convergence and more effective optimisation.

- α_t: The learning rate controls the step size of each parameter update. A common value is 0.001.
- μ_t: The initial momentum parameter is set to a value close to 1, such as 0.9.
- ν: Exponential decay rate that controls second-order momentum estimation. The common value is 0.99.
- ε: Constant to prevent division by zero errors. The value is generally set to 1e-7.
- g_t : Gradient of the objective function at time step t.
- m_t: First-moment estimate (moving average of gradients), initialise to 0.
- n_t: Second-moment estimate (moving-average of squared gradients), initialised to 0.
- $\hat{m}_t$: Bias-corrected first moment estimate.
- $\hat{n}_t$: Bias-corrected second moment estimate.
- θ_t : Parameters being optimised.
- t: Current iteration times.
- T: Epoch value.

Notwithstanding these benefits, Nadam may exhibit instability during training due to exploding gradients, unsuitable learning rates, and momentum fluctuations. To mitigate these issues, approaches such as gradient clipping, which constrains the gradient magnitude, and adaptive learning rate scheduling, which dynamically adjusts the learning rate, are typically employed to ensure steady and effective training, particularly in deep neural networks.

4.2.2 Introduction to Gradient Clipping Formula

Gradient clipping is a method used to mitigate the problem of exploding gradients by limiting gradient magnitudes during backpropagation. These mitigation measures may potentially resolve the problem of ballooning gradients.

Standard gradient clipping restricts the magnitude of the gradient to ensure training stability, but it also introduces bias. U-Clip enhances this process by retaining a buffer to hold the clipped segment of the gradient and subsequently reapplying those values to the new gradient in the following iteration (Elesedy & Hutter, 2023). U-Clip guarantees that the cumulative update deviation is constrained, ensuring that the clipped updates are unbiased on average and enhancing the stability and convergence of the optimisation method. The U-Clip gradient clipping technique mitigates bias in random gradient clipping, as demonstrated in equations 4.1-4.3:

For $x \in \mathbb{R}$, coordinate clipping to scale $\gamma > 0$ is given by

Introduction of U-Clip:

1. Gradient clipping:

$$u_t = min\left(1, \frac{\gamma}{\|g_t + \Delta_t\|}\right)(g_t + \Delta_t) \quad (4.1)$$

2. Update buffer:4.1

$$\Delta_{t+1} = g_t + \Delta_t - u_t \quad (4.2)$$

Where Δ_{t+1} is the updated buffer.

3. Parameter update:

$$\theta_{t+1} = \theta_t - \eta u_t \quad (4.3)$$

- g_t: Gradient

- γ: Gradient clipping threshold, limiting the maximum value of the gradient. You can choose a fixed value, such as 0.5
- Δ_t: A buffer that holds the gradient that was clipped in the previous steps.
- u_t: The gradient after the clip function processing.
- η: Learning rate, control parameter update step size. Usually, the learning rate is 0.01 or 0.001.
- θ: Weights and biases need to be trained.

The U-Clip algorithm comprises these formulae, which diminish update deviation and enhance the stability and convergence of the optimisation algorithm by preserving and employing the clipped gradient component throughout the gradient clipping process. Flowchart 4.3 illustrates the fundamental steps of the U-Clip algorithm: the algorithm commences with the initialisation of parameters, including initial values, the learning rate, the clipping region, the total iterations, and cumulative variables. The procedure then proceeds through iterations, in which each iteration randomly selects a batch from the dataset, computes the current gradient, updates the clipped gradient and cumulative variables, and subsequently adjusts the parameters using the learning rate and the clipped gradient. Subsequently, ascertain if the present iteration number is inferior to the total number of iterations. If affirmative, proceed with the iteration; if otherwise, terminate the procedure. This procedure is repeated until a specified number of iterations is reached, thereby guaranteeing incremental parameter optimisation.

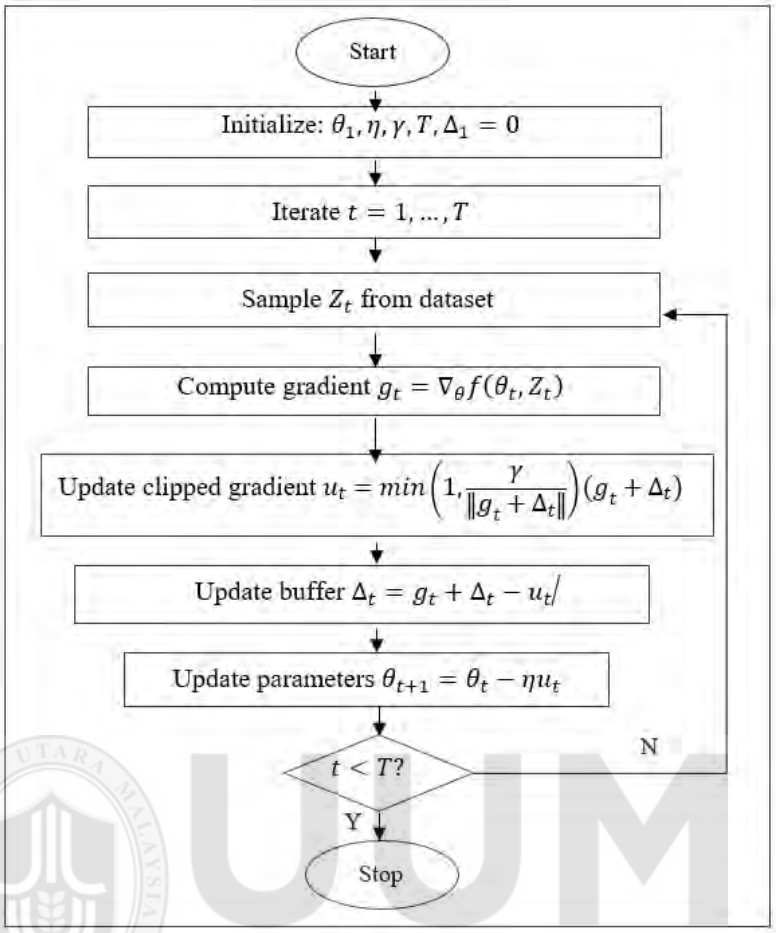

Figure 4.3. U-Clip algorithm flowchart

Figure 4.3 illustrates the flowchart of the U-Clip algorithm, where T represents the epoch value and t denotes the number of iterations completed to date. Additional parameters are elucidated in Formulas 4.1-4.3.

The following pseudocode summarises the core mathematical steps of U-Clip: clipped-gradient computation, buffer updating, and parameter refinement. As show in table 4.2.

Table 4.2

U-Clip pseudocode (Harimoorthy & Thangavelu, 2021)

Algorithm 4.2: Pseudo code for the U-Clip algorithm

Input:

θ_0 # initial parameters

η # learning rate

γ # clipping scale (> 0)

T # total training steps

Initialize:

$\Delta_t \leftarrow 0$ # buffer for gradient correction

for t = 0 … T − 1 do

1. Compute gradient at current parameters

$g_t \leftarrow \nabla_{\theta_t} L\theta_t$

2. Gradient clipping (U-Clip)

combined_grad $\leftarrow g_t + \Delta_t$

norm ← ||combined_grad||

scale ← γ / norm

scale ← min(1, scale)

$u_t \leftarrow$ scale * combined_grad # clipped gradient u_t

3. Update buffer Δ_{t+1}

$\Delta_{t+1} \leftarrow g_t + \Delta_t - u_t$

4. Parameter update

$\theta_{t+1} = \theta_t - \eta u_t$

end for

return θ_T

4.2.3 Integration of Gradient Clipping into Nadam

4.2.3.1 Proposed Algorithm

To enhance the Nadam optimiser, the gradient clipping technique is integrated into the Nadam algorithm. Gradient clipping is a technique used to prevent gradient spikes, a common problem in deep learning models, especially LSTM networks. Gradient

clipping helps stabilise the training process by capping gradients during backpropagation. The U-Clip algorithm is used to enhance the Nadam algorithm, and the gradient range of the NadamClip training LSTM model is established. The following describes how NadamClip works.

Figure 4.4 shows the workflow of the NadamClip algorithm. The process is divided into several functional modules, each of which corresponds to the figures specifically mentioned earlier, namely Figure 4.2 and Figure 4.3. The training begins with the initialisation of key parameters, including the model weights θ, learning rate η, gradient clipping threshold γ, total number of iterations T, and the buffer term Δ. This sets the foundation for subsequent optimisation steps. Next, the model computes the gradient g_t at the current iteration. The procedure then proceeds to the gradient clipping stage, as detailed in Figure 4.4. Here, the current gradient is combined with the accumulated buffer to determine if clipping is necessary. If so, the gradient is clipped accordingly, and the buffer Δ_{t+1} is updated to preserve the truncated components. Subsequently, the optimiser updates the first- and second-order moment estimates m_t and n_t, following the rules of the Nadam algorithm. This part of the process corresponds to the operations illustrated in Figure 4.3. Bias correction is then applied to enhance the reliability of the moment estimates. Finally, the model parameters θ_t are updated using the corrected estimates. The entire procedure iterates until the predefined maximum number of epochs is reached. By seamlessly integrating gradient clipping into the Nadam optimisation framework, the NadamClip method achieves a favourable trade-off between stability and convergence efficiency.

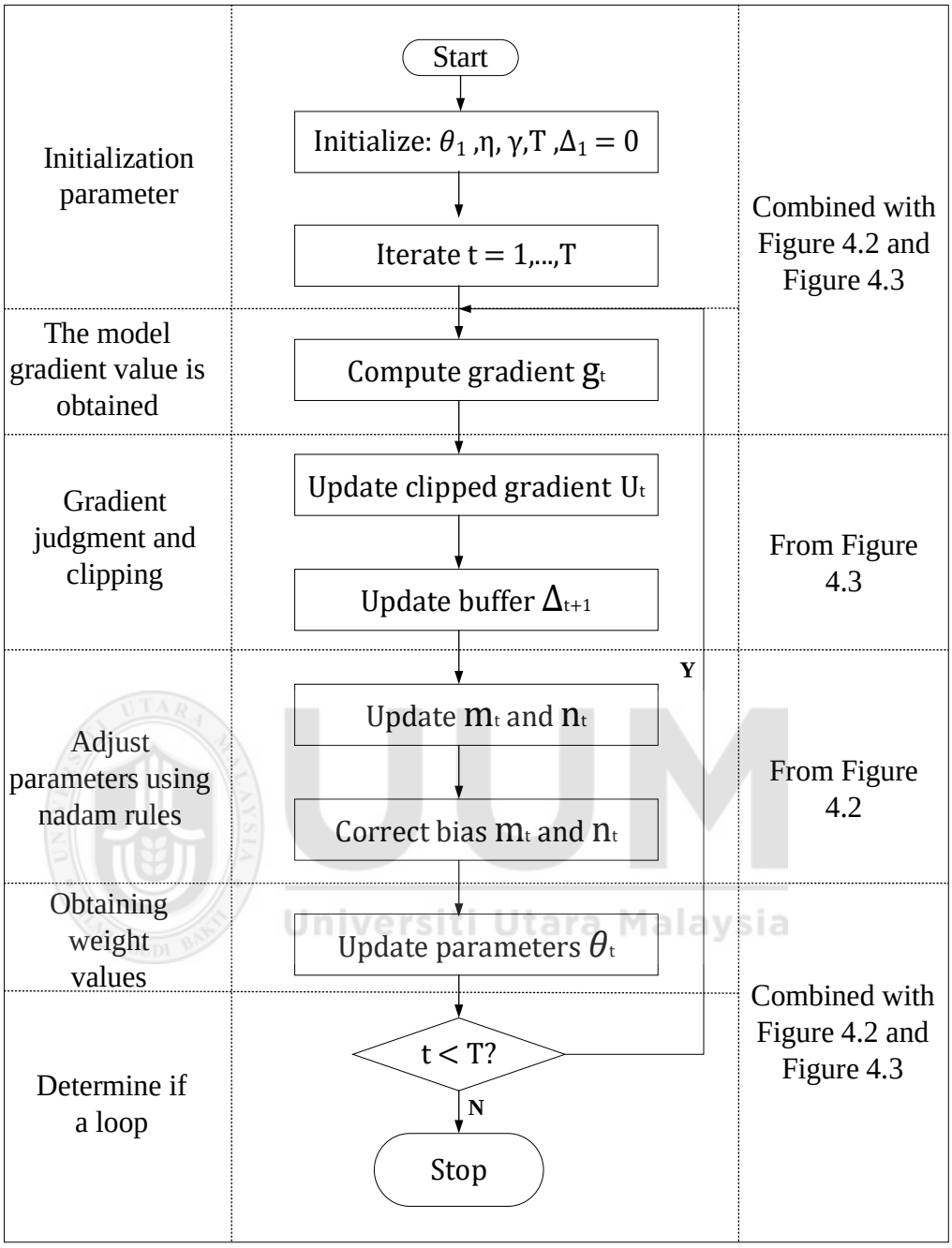

Figure 4.4. Proposed NadamClip model

The NadamClip algorithm formula as show in table 4.3:

Table 4.3

Proposed NadamClip optimisation

Algorithm 4.3 Nesterov-accelerated Adaptive Moment Estimation Clipping (NadamClip)

Require: Hyperparameters: $\alpha_0, \dots, \alpha_T; \mu_0, \dots, \mu_T; \nu; \epsilon$

$\Delta_1 = 0, \gamma \in \mathbb{R}_{>0}$ (clipping region)

$\mathrm{m}_0; \mathrm{n}_0 \leftarrow 0$(first/second moment vectors)

While θ_t not converged do

$\mathrm{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$

$U_t = min\left(1, \frac{\gamma}{\|g_t + \Delta_t\|}\right)(g_t + \Delta_t)$

$\Delta_{t+1} = g_t + \Delta_t - U_t$

$\mathrm{m}_t \leftarrow \mu_t \mathrm{m}_{t-1} + (1 - \mu_t) U_t$

$\mathrm{n}_t \leftarrow \nu \mathrm{n}_{t-1} + (1 - \nu) U_t^2$

$\widehat{\mathrm{m}} \leftarrow \mu \cdot \mathrm{m}_t / (1 - \mu^t) + (1 - \mu) \cdot U_t / (1 - \mu^t)$

$\hat{n} \leftarrow \nu \mathrm{n}_t / (1 - \nu^t)$

$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{n}_t + \varepsilon}} \times \widehat{m}_t$

end while

Return θ_t

Where:

α is the learning rate.

- β_1 and β_2 are the exponential decay rates for the moment estimates.
- m_t and v_t are the first- and second-moment estimates.
- ϵ is a small constant to prevent division by zero.
- U_t *is the clipped gradient.*
- γ applies gradient clipping with a specified threshold (clip norm).

Table 4.2 shows the mathematical formulation of the algorithm. The process begins with the initialisation of the first and second moment estimates, m_t and n_t, along with a residual buffer Δ_t required for gradient clipping. At each iteration, the gradient g_t of the objective function with respect to the current parameters is first computed through forward propagation. Next, the gradient is updated using the error-compensated gradient clipping formula. Specifically, the current gradient g_t is added to the buffer Δ_t, and the resulting vector is rescaled based on a predefined threshold γ, producing the clipped gradient u_t. This mechanism avoids discarding the clipped portion entirely, thereby alleviating the estimation bias introduced by standard hard clipping. The buffer Δ_t is then updated to retain the residual information excluded during clipping. Subsequently, the clipped gradient u_t is used to update the first moment m_t and the second moment n_t, followed by bias correction to obtain $\hat{m}_t$ and $\hat{n}_t$. Finally, the model parameters are updated using the corrected moment estimates. This process is repeated iteratively until the convergence criteria are met. By incorporating error-compensated gradient clipping, NadamClip preserves the adaptive learning capability of the original Nadam algorithm while improving numerical stability and generalisation in deep neural network training.

4.2.3.2 Theoretical Foundations and Convergence Properties of NadamClip

To ensure that NadamClip is not only empirically effective but also conceptually well motivated, the algorithm is examined through theoretical analysis aimed at providing insight into its underlying design principles. The following analysis presents a formal treatment of the gradient projection operator, momentum dynamics, second-moment stability, bias-correction behaviour, and contractive properties of the update rule. These components collectively support the validity and convergence behaviour of

NadamClip within both convex and non-convex optimisation settings. As shown in formula 4.4-4.30.

(1) Variational Interpretation of Gradient Clipping

The gradient clipping operation solves a constrained optimisation problem:

$$U_t = \arg\min_{v \in \mathbb{R}^d} \|v - g_t\|^2 \text{ s.t. } \|v\| \leq \gamma \qquad (4.4)$$

The Karush-Kuhn-Tucker (KKT) conditions yield the analytical solution:

$$U_t = \begin{cases} g_t & \|g_t\| \leq \gamma \\ \gamma \cdot g_t / \|g_t\| & \|g_t\| > \gamma \end{cases} \qquad (4.5)$$

This is equivalent to

$$U_t = \text{proj}_{\mathcal{B}(0,\gamma)}(g_t) \qquad (4.6)$$

where $\mathcal{B}(0,\gamma)$ is the L2-ball of radius γ. This projection operator satisfies the following:

- Non-expansiveness: $\|U_t - U_s\| \leq \|g_t - g_s\|$.
- The sub-gradient relation: $U_t \in \partial \tilde{f}_t(\theta_{t-1})$.

Here, $\tilde{f}_t(\theta) = f_t(\theta) + \delta_{\mathcal{B}(0,\gamma)}(\nabla f_t)$ is a regularised form of the original function.

(2) Lyapunov Analysis of the Momentum System

Define the momentum error vector:

$$e_t = m_t - \mathbb{E}[U_t] \qquad (4.7)$$

Its dynamics follow:

$$e_t = \mu_t e_{t-1} + (1-\mu_t)(U_t - \mathbb{E}[U_t]) + \mu_t(\mathbb{E}[U_{t-1}] - \mathbb{E}[U_t]) \quad (4.8)$$

Construct the Lyapunov function:

$$V_t = \|e_t\|^2 + \lambda\|\mathbb{E}[U_t] - \nabla f(\theta^*)\|^2 \quad (4.9)$$

where θ^* is the optimal solution. Its difference satisfies

$$\Delta V_t \leq -c_1\|e_t\|^2 + c_2 v^t + c_3\|\nabla f(\theta_{t-1}) - \nabla f(\theta^*)\|^2 \quad (4.10)$$

Assuming κ-strong convexity of ∇f,

$$\Delta V_t \leq -\kappa V_t + \mathcal{O}(v^t) \quad (4.11)$$

Proving exponential stability of the momentum system.

(3) Stochastic Stability of the Second Moment

Consider the stochastic difference equation for the second moment:

$$n_t - \mathbb{E}[U_t^2] = v(n_{t-1} - \mathbb{E}[U_{t-1}^2]) + \zeta_t \quad (4.12)$$

Where $\zeta_t = (1-v)(U_t^2 - \mathbb{E}[U_t^2]) + v(\mathbb{E}[U_{t-1}^2] - \mathbb{E}[U_t^2])$.

The covariance matrix satisfies:

$$\mathrm{Var}(n_t) \leq v^2\mathrm{Var}(n_{t-1}) + (1-v)^2\sigma_U^2 I \quad (4.13)$$

Where $\sigma_U^2 = \sup_t \mathrm{Var}(\|U_t\|^2) \leq 4\gamma^4$. The solution to this recurrence inequality is

$$\|\mathrm{Var}(n_t)\|_{\mathrm{op}} \leq \frac{(1-v)^2\sigma_U^2}{1-v^2} + \mathcal{O}(v^t) \quad (4.14)$$

Proving boundedness in probability for the second moment estimate.

(4) Spectral Analysis of Bias Correction

The bias correction term can be rewritten as

$$\hat{m}_t = \underbrace{\left(\frac{\mu_{t+1}}{1-\prod_{i=1}^{t+1}\mu_i}\sum_{k=0}^{t-1}\omega_k U_k\right)}_{\text{historical term}} + \underbrace{\left(\frac{1-\mu_t}{1-\prod_{i=1}^{t}\mu_i} + \frac{\mu_{t+1}(1-\mu_t)}{1-\prod_{i=1}^{t+1}\mu_i}\right)U_t}_{\text{current term}} \quad (4.15)$$

Its coefficient matrix $A_t = \text{diag}\left(a_t^{(1)}, \ldots, a_t^{(d)}\right)$ satisfies

$$\min_i a_t^{(i)} \geq \frac{1-\mu_{\max}}{1-\mu_{\min}^t} \quad (4.16)$$

$$\|A_t\|_{\text{op}} \leq \frac{2}{1-\mu_{\max}} \quad (4.17)$$

$$\text{cond}(A_t) \leq \frac{2(1-\mu_{\min}^t)}{(1-\mu_{\max})(1-\mu_{\min})} \quad (4.18)$$

The bounded condition number ensures numerical stability.

(5) Contractility of Update Operator

The parameter update can be expressed as

$$\theta_t = \theta_{t-1} - \alpha_t \Phi_t(\hat{m}_t, \hat{n}_t) \quad (4.19)$$

where $\Phi_t(x, y) = x/(\sqrt{y} + \epsilon)$ is a nonlinear operator. In the domain $\mathcal{D} = \{(x, y): \|x\| \leq \gamma, y \geq \delta\}\left(\delta = \min_i \mathbb{E}\left[U_{t,i}^2\right] > 0\right)$,

$$\|\Phi_t(x, y) - \Phi_t(x', y')\| \leq L_\Phi(\|x - x'\| + |y - y'|) \quad (4.20)$$

The Lipschitz constant is $L_\Phi = \max\left(\frac{1}{\sqrt{\delta}}, \frac{\gamma}{2\delta^{3/2}}\right)$. Combined with stability from Steps 2-4, the update operator is quasi-contractive.

(6) Convergence Analysis

Under standard non-convex assumptions:

M-smoothness:

$$\|\nabla f(x) - \nabla f(y)\| \leq M\|x - y\| \quad (4.21)$$

Bounded gradient variance:

$$\mathbb{E}[\|g_t - \nabla f(\theta_{t-1})\|^2] \leq \sigma^2 \quad (4.22)$$

Define the Lyapunov function:

$$L_t = f(\theta_t) + \frac{\alpha_t}{\lambda} V_t + \beta\|\mathbf{n}_t - \mathbb{E}[U_t^2]\|^2 \quad (4.23)$$

Define gradient mapping:

$$g_t = \frac{1}{\alpha_t}\|\theta_t - \theta_{t-1} + \alpha_t \nabla f(\theta_{t-1})\| \quad (4.24)$$

Its expected difference satisfies

$$\mathbb{E}[L_t - L_{t-1}] \leq -K\alpha_t \mathbb{E}[\|\nabla f(\theta_{t-1})\|^2] + C_1\alpha_t^2 + C_2 v_t + C_3\gamma^{-1}\sigma^2 \quad (4.25)$$

Selecting $\alpha_t = \mathcal{O}(1/t), \gamma = \mathcal{O}(\sigma)$ yields

$$\liminf_{t\to\infty} \mathbb{E}[\|G_t\|^2] \leq \mathcal{O}(1) \quad (4.26)$$

Analytical insight: M-smoothness gives $f(\theta_t) \leq f(\theta_{t-1}) - \alpha_t\langle \nabla f(\theta_{t-1}), \Phi_t\rangle + \frac{M\alpha_t^2}{2}\|\Phi_t\|^2$.

The boundedness of $\|\Phi_t\|$ (from clipping and $y \geq \delta$) and the stability of V_t, n_t complete the bound. Summing over t yields the stationary point guarantee.

(7) Error Propagation vs Original Nadam

Let θ_t^{Nadam} be the original Nadam iteration and θ_t^{Clip} the NadamClip iteration. Define the error:

$$\varepsilon_t = \|\theta_t^{\text{Clip}} - \theta_t \quad (4.27)$$

The error dynamics satisfy:

$$\varepsilon_t \leq \left(1 + \alpha_t L_f\right)\varepsilon_{t-1} + \alpha_t \left\|\Phi_t\left(\hat{m}_t^{\text{Clip}}, \hat{n}_t^{\text{Clip}}\right) - \Phi_t\left(\hat{m}_t^{\text{Nadam}}, \hat{n}_t^{\text{Nadam}}\right)\right\| \quad (4.28)$$

Lipschitz continuity gives:

$$\varepsilon_t \leq \prod_{k=1}^{t}\left(1 + \alpha_k L_f\right)\varepsilon_0 + L_\Phi \sum_{k=1}^{t} \alpha_k \prod_{j=k+1}^{t}\left(1 + \alpha_j L_f\right)\|U_k - g_k\| \quad (4.29)$$

When $\|g_t\| \leq \gamma_1 \|U_t - g_t\| = 0$; otherwise $\|U_t - g_t\| \leq \|g_t\|$. By the law of large numbers:

$$\mathbb{E}[\varepsilon_t] \leq \mathcal{O}\left(e^{-\lambda t} + \gamma \Pr(\|g_t\| > \gamma)\right) \quad (4.30)$$

The mathematical analysis provides theoretical insight into the validity of the NadamClip formulation by examining the behaviour of its core components. The gradient projection operation is shown to preserve directional information while enforcing Lipschitz continuity, with the momentum system demonstrating exponential stability under Lyapunov criteria. Second-moment estimates maintain bounded variance through stochastic difference equations, while bias correction terms exhibit spectral properties ensuring numerical robustness. Crucially, the composite update operator satisfies con-tractive properties that guarantee iterative stability, with global convergence established under standard convexity and smoothness assumptions.

4.2.3.3 Computational Time and Space Complexity

Let p denote the total number of trainable parameters in the neural network. The proposed NadamClip optimiser extends the conventional Nadam algorithm by incorporating an internal gradient clipping mechanism, while preserving the original first- and second-order moment estimation framework and the Nesterov-accelerated update structure.

For each training iteration, the standard Nadam optimiser performs a constant number of arithmetic operations per parameter, including the updates of the first-order moment m_t, the second-order moment v_t, bias correction, and the Nesterov-style parameter update. Consequently, the per-iteration time complexity of Nadam is linear in the parameter size and can be expressed by formula 4.31.

$$T_{Nadam} = O(p) \qquad (4.31)$$

where p denotes the total number of model parameters.

In NadamClip, an additional global ℓ_2-norm gradient clipping operation is introduced. Specifically, this process requires computing the gradient norm $\| g \|$, which incurs a computational cost of $O(p)$, followed by formula 4.32 of the clipping coefficient.

$$\alpha = \min\left(1, \frac{c}{\|g\|}\right) \qquad (4.32)$$

which is a constant-time operation $O(1)$. Subsequently, all gradient components are rescaled according to formula 4.33.

$$g \leftarrow \alpha g \qquad (4.33)$$

which again requires $O(p)$operations. Since all additional procedures introduced by the gradient clipping mechanism scale linearly with the number of parameters, the overall per-iteration time complexity of NadamClip remains as in Formula 4.34.

$$T_{NadamClip} = O(p) + O(p) = O(p) \qquad (4.34)$$

indicating that NadamClip preserves the same asymptotic time complexity as Adam and Nadam, differing only by a small constant factor introduced by the clipping operation.

When applied to sequence models such as LSTM networks with a mini-batch size $b$and sequence length T, the dominant computational cost is governed by the forward and backward propagation processes, which scale as $O(bTp)$.

In this context, the additional $O(p)$ computational overhead induced by gradient clipping becomes asymptotically negligible relative to the total training cost, thereby exerting minimal impact on the overall training efficiency.

With respect to space complexity, the standard Nadam optimiser maintains, for each parameter, the parameter value θ, the instantaneous gradient g, and two auxiliary moment variables m and and v, leading to a memory consumption determined by Formula 4.35.

$$S_{Nadam} = O(4p) \qquad (4.35)$$

The NadamClip algorithm reuses the original gradient tensor during the clipping process. It does not introduce any additional persistent per-parameter storage, aside

from temporary scalar variables such as the global gradient norm and the clipping coefficient. Consequently, its memory complexity remains as in Formula 4.36.

$$S_{NadamClip} = O(4p) = O(p) \quad (4.36)$$

In summary, the proposed NadamClip optimiser preserves the linear time and space complexity characteristics of conventional adaptive optimisation algorithms, as in formula 4.37.

$$T_{NadamClip} = O(p), S_{NadamClip} = O(p) \quad (4.37)$$

thereby demonstrating that the introduction of gradient clipping enhances training stability by effectively suppressing gradient spikes, without incurring any additional asymptotic computational or memory overhead. This confirms that NadamClip achieves improved numerical robustness while maintaining computational scalability suitable for large-scale deep learning models.

4.2.3.4 NadamClip Algorithm Working Step

The shaded portion of Figure 4.1 shows the application of the new optimisation algorithm to the model, and the shadow part is extracted separately in Figure 4.5.

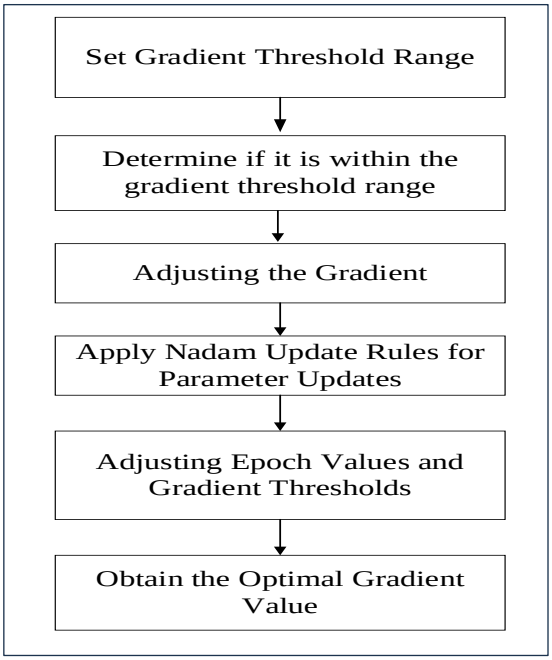

Figure 4.5. Brief training step of the new optimisation algorithm

During LSTM model training, it is essential to regulate the scale and range of gradients to maintain stability and efficiency. Figure 4.3 illustrates the operational idea of the proposed method. A gradient range (i.e., gradient threshold) is set to regulate gradient updates during LSTM model training. This threshold is typically established through experience or experimentation to avert the occurrence of a gradient spike, in which the gradient norm increases significantly during backpropagation, leading to excessive updates to the model parameters and thereby compromising the stability and convergence of the model.

The gradient criteria are consistently evaluated at each training step. When the gradient norm exceeds the established threshold, the gradient clipping method is applied to reduce it. Gradient clipping is a prevalent method that constrains the gradient to an acceptable range by capping its maximum absolute value. This action mitigates abrupt fluctuations in model parameters during updates, hence enhancing model stability.

When a gradient exceeds a specified threshold, the gradient clipping procedure either scales or truncates it according to a predetermined rule. This typically entails constraining the gradient norm to a threshold to maintain its magnitude within tolerable limits. This efficiently regulates the gradient magnitude, preventing it from being excessively large or small, ensuring that the model parameters are updated in a judicious and effective manner. If the gradient criterion remains below the threshold, no alterations are implemented, and the existing gradient criterion is utilised directly.

The maximum gradient norm, defined as the largest absolute value of the gradient, was documented at the conclusion of each training session. The maximum gradient norm provides direct insight into the magnitude of the gradient and helps understand its variations throughout the training process. Modify the gradient threshold according to the maximum gradient criterion for each training session. Ultimately, by systematically evaluating the model's performance across different truncation thresholds and identifying the range that yields optimal performance and stability, the appropriate gradient threshold is determined. By analysing MAE, RMSE, R^2, and other training dynamics across these settings, a threshold range can be selected that both suppresses gradient spikes and preserves sufficient gradient information for effective parameter updates, thereby ensuring stable training and improved predictive performance.

This method guarantees that the LSTM model's gradient norms consistently remain within a suitable range during training, thereby enhancing the model's training efficiency.

4.2.3.5 Pseudocode of NadamClip

This section presents the Pseudocode of NadamClip, which illustrates where a clipping technique appears in a program. In this study, the NadamClip optimisation process, m and v, are initialised as zero vectors matching the parameter shapes, with an initialised flag ensuring they are set only once. The first-moment vector m represents the mean of gradients, while the second-moment vector v captures the uncentered variance of gradients. To correct bias, m and v are computed. Nesterov acceleration is applied to obtain m_hat, thereby enhancing optimisation. Gradient clipping prevents exploding gradients by scaling gradients that exceed a specified gamma. Finally, parameters are updated using the Nesterov-accelerated gradient m_hat and the bias-corrected second moment estimate v_hat. This process iterates over batches of data, incrementing the time step t with each batch to maintain bias correction. As show in table 4.4.

Table 4.4

NadamClip pseudocode

Algorithm 4.4: Pseudocode for the NadamClip algorithm

Input:

Model parameters θ

Hyperparameters (γ, α, β_1, β_2, ε)

Gradients g_t for each training step

Initialize:

For each parameter θ_i:

$m_t \leftarrow 0$ # First-order moment

$n_t \leftarrow 0$ # Second-order moment

$\Delta_t \leftarrow 0$ # Gradient correction term

$t \leftarrow 0$ # Time step

Table 4.4 continued

Algorithm 4.4: Pseudocode for the NadamClip algorithm

Procedure Apply Gradients(g_t):

$t \leftarrow t + 1$

For each parameter θ_i with gradient g_i:

$\tilde{g}_i \leftarrow g_i + \Delta_t$ # Combined gradient

$norm \leftarrow \| \tilde{g}_i \|$

$scale \leftarrow \gamma / (norm + 1e{-}6)$

$scale \leftarrow \min(1, scale)$

$U_i \leftarrow \tilde{g}_i * scale$ # Clipped gradient

$\Delta_i \leftarrow \tilde{g}_i - U^i$

$m_i \leftarrow \beta_1 m_i + (1 - \beta_1) U_i$ # First moment update

$m\_nest\_i \leftarrow \beta_1 m_i + (1 - \beta_1) U_i$ # Nesterov prediction

$v_i \leftarrow \beta_2 v_i + (1 - \beta_2) U_i^2$

$m\_hat \leftarrow m\_nest\_i / (1 - \beta_1^t)$

$v\_hat \leftarrow v\_i / (1 - \beta_2^t)$

$update \leftarrow \alpha * m\_hat / (sqrt(v\_hat) + \varepsilon)$

$\theta i \leftarrow \theta i - update$

Return updated parameters θ

The U-clip technique is executed by establishing a threshold that delineates the maximum permissible range of gradients. When the computed gradients exceed this threshold, they are normalised to that range, ensuring a steady training process. The gradient clipping threshold is a modifiable parameter that can be adjusted based on integration specifics and range selection.

U-Clip involves adjusting the gradients before updating the model parameters. If a gradient exceeds a specified threshold, it is reduced to fall within the specified range. This mitigates overly large gradients that may induce unstable updates and obstruct

the training process. The Nadam optimiser, recognised for integrating the advantages of Nesterov momentum and Adam's adaptive learning rates, can be enhanced by the application of gradient clipping. This paper introduces NadamClip, a variant that incorporates gradient clipping, preserving Nadam's adaptive learning rate while addressing gradient spiking problems.

4.5 Summary

This chapter systematically presents a novel optimisation algorithm - the NadamClip algorithm. This chapter first examines the Nadam and U-Clip algorithms. The notion of gradient clipping and its implementation are elucidated in detail, accompanied by pseudo-code examples to enhance the reader's comprehension of this approach.

CHAPTER FIVE
EXPERIMENT AND DISCUSSION

5.1 Introduction

This chapter presents a series of experiments to examine the efficacy of the NadamClip optimisation technique in LSTM models and assess its suitability for ammonia concentration prediction tasks. This chapter meticulously documents the configuration of each optimisation algorithm and the procedural steps of the experimental process.

The forecasting of ammonia concentration in aquaculture served as a test case for modelling and analysing time-series data. The experiment evaluates the efficacy of the NadamClip optimisation algorithm based on the LSTM model. It contrasts it with several other optimisers, including Adam, SGD with momentum, RMSprop, and SGD with gradient clipping.

The experiment evaluates the gradient clipping function of the NadamClip algorithm by setting various clipping thresholds and analysing their impact on the training process. This experiment collects data on the clipping threshold to ascertain the relevant gradient clipping range.

This chapter conducts several comparative experiments to thoroughly assess the training efficiency and performance of NadamClip alongside other optimisation algorithms, using standardised experimental protocols and metrics. All algorithms utilise identical initial parameter configurations and data splitting methods to guarantee result comparability. The experimental method delineates essential

processes, including data preprocessing, hyperparameter configuration, training, and performance documentation.

This chapter thoroughly evaluates the performance of the NadamClip optimisation algorithm within the LSTM model through experimental design and implementation.

5.2 Experimental Preparation

The optimisation and validation of the NadamClip technique and LSTM model for predicting ammonia concentration in aquaculture were conducted in two phases: data preprocessing and hyperparameter setting. The raw ammonia concentration data are first converted into a standardised format suitable for the LSTM model through data cleaning, denoising, and normalisation. At the same time, correlations among variables are analysed to reveal pertinent causal and statistical links. Thereafter, hyperparameters are defined, including the number of network layers, the number of neurons, the learning rate, and the size of the LSTM training batch, as seen in Figure 5.1.

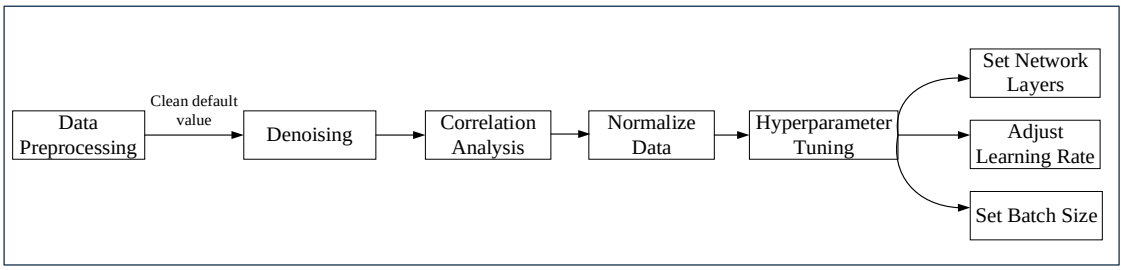

Figure 5.1. Experimental preparation procedure

5.2.1 Experimental Environment

The experimental assessments in this work were conducted painstakingly in a controlled, standardised computing environment to ensure the repeatability and consistency of the outcomes. The principal hardware employed for these tests was

Google Collaboratory (Colab), a cloud-based interactive platform with substantial computational resources. The studies utilised the v2-8 Tensor Processing Unit (TPU), comprising eight TPU cores. The TPU v2-8 architecture is distinguished by its exceptional machine learning acceleration capabilities, making it particularly well-suited for training and deploying large-scale deep learning models (Y. E. Wang et al., 2019). Every TPU core has 8 GB of memory, and the system benefits from high-speed interconnect bandwidth that enables effective communication among the cores. The unique ASICs (Application-Specific Integrated Circuits) in the TPU are optimised for TensorFlow operations, hence improving computational performance.

The experimental configuration was implemented in Python, chosen for its comprehensive support in machine learning and data science, as well as its seamless integration with TensorFlow. The tests utilised TensorFlow version 2.15.0, which provides a comprehensive framework for constructing and deploying machine learning models, improved usability, and interaction with Keras for expedited model construction and experimentation (Pang et al., 2020). The Python environment included several crucial libraries installed with pip, including NumPy for numerical computations and array manipulation (Grout et al., 2019), Pandas for data management and preprocessing, Matplotlib and Seaborn for data visualisation, and Scikit-learn for additional machine learning tasks and evaluation metrics (AKGÜN, 2021). The operating system used by Google Colab is Linux-based, providing a robust, scalable environment for computing.

The dynamic allocation of computational resources in Google Colab was utilised to ensure a stable experimental configuration. The TPU duration was deliberately chosen

to fully utilise the TPU v2-8's acceleration capabilities (Kimm et al., 2021). Experiments were structured to comply with Colab's session constraints, with regular checkpoints and data backups to mitigate potential disruptions. Integrating with Google Drive met persistent storage requirements, facilitating effortless access to datasets and the preservation of model checkpoints. The summary of the experimental environment is shown in Table 5.1.

Table 5.1

Experimental environment summary

Concept	Code / Implementation
Computing Platform	Google Colaboratory (Colab), a cloud-based environment
Hardware	TPU v2-8 (8 TPU cores) (Wang et al., 2019)
TPU Core Specs	8 GB memory per core, high-bandwidth interconnect
TPU Architecture	ASICs optimised for TensorFlow operations.
Programming Language	Python
ML Framework	TensorFlow 2.15.0 + Keras API (Pang et al., 2020)
Key Python Libraries	NumPy (numerical computing), Pandas (data handling), Matplotlib/Seaborn (visualisation), Scikit-learn (evaluation)
Operating System	Linux-based (Colab default)

5.2.2 Dataset Pre-processing

The objective of the data analysis phase of this project was to preprocess the aquaculture dataset through a series of meticulously developed stages to establish a robust data foundation for future predictive modelling and decision assistance. The

subsequent subsections pertain to the data analysis portion (the dataset sources are elaborated in 3.3 Data Pre-processing).

5.2.2.1 Data Cleaning and Denoising

Data cleaning and denoising are essential tasks in data analysis and critical steps to ensure data quality. Raw data frequently originates from various sources and may include missing values, outliers, duplicate records, and erroneous information that contradicts business logic; this "noise" can significantly compromise the quality and reliability of future analyses (Brown & Kros, 2003). Consequently, the experiment will minimise missing values and duplicate data, reduce data noise, preserve authentic and pertinent information, and establish a robust data foundation for subsequent correlation analysis and model development.

Table 5.2 illustrates that the dataset has five essential information elements. These records encompass creation time, temperature, C-value, dissolved oxygen, pH, and ammonia nitrogen levels. The dataset originally contained 83,074 items and underwent a series of procedures to enhance data integrity and reliability.

Table 5.2

Dataset information sheet

Serial number	Variant	Symbolic	Abbreviation
1	Creation time	created_at	None
2	Temperature	Temperature (°C)	TP
3	Dissolved Oxygen	Dissolved Oxygen(g/ml)	DO
4	chemistry	pH	pH
5	Ammonia	Ammonia(g/ml)	None

Using the functions provided by the Pandas library to perform basic data cleaning operations, 18,837 rows containing Not a Number (NaN) of missing data rows were checked, as shown in Table 5.3.

Table 5.3

Data rows with missing values

No.	Created_at	Temperature (C)	Dissolved Oxygen(g/ml)	PH	Ammonia(g/ml)
64237	NaN	NaN	NaN	NaN	NaN
64238	NaN	NaN	NaN	NaN	NaN
64239	NaN	NaN	NaN	NaN	NaN
…	…	…	…	…	…
83072	NaN	NaN	NaN	NaN	NaN
83073	NaN	NaN	NaN	NaN	NaN

The Dropna() method provided by Pandas was used to remove the line containing NaN, resulting in a dataset containing 64237.

This dataset is then denoised. Set the window size in the moving average method parameter to 3. The window size was set to 3 because it smooths noise without distorting the original signal. A larger window would oversmooth the data, weakening short-term fluctuations that are important for ammonia prediction. A smaller window would not reduce noise effectively. So, 3 provides a good balance between noise reduction and keeping useful temporal patterns. The data set obtained after de-noising is shown in Table 5.4.

Table 5.4

Example of dataset after denoising

Created_at	Temperature (°C)	Dissolved Oxygen(g/ml)	PH	Ammonia(g/ml)
2021-06-19 00:00:05	NaN	NaN	NaN	NaN
2021-06-19 00:01:02	NaN	NaN	NaN	NaN
2021-06-19 00:01:22	22.94329	8.679036	8.403272	0.43787
2021-06-19 00:02:07	22.96351	19.46128	8.392673	0.437307

Then, NaNs are removed from the data set, and the resulting data set is shown in Table 5.5.

Table 5.5

Example of dataset after removing duplicates

Created_at	Temperature (C)	Dissolved Oxygen(g/ml)	PH	Ammonia(g/ml)
2021-06-19 00:01:22	22.94329	8.679036	8.403272	0.43787
2021-06-19 00:01:44	22.95948	8.858681	8.403203	0.433637
2021-06-19 00:02:07	22.96351	19.46128	8.392673	0.437307
2021-06-19 00:02:27	23.00289	15.48569	8.391434	0.454187
2021-06-19 00:02:47	22.98671	14.74804	8.393012	0.45842

To sum up, the total amount of data is 83,074; 18,837 NAN values are cleaned; 2 data sets are denoised and deleted; and the final number of data sets is 64,235, as shown in Table 5.6.

Table 5.6

Data set cleaning and denoising results

Operation	Quantities
Initial number of data items	83,074
After processing missing values	18,837
De-duplication of data	0

Table 5.6 comtinued

Operation	Quantities
Removal of invalid time data	0
Delete noise data	2
Final number of remaining data entries	64,235

5.2.3 Data Correlation Analysis

The corr () method of Pandas was used to calculate the specific correlation coefficients between variables. Table 5.7 presents the results of a Spearman's rank correlation analysis between three variables and ammonia concentration, including the correlation coefficient (ρ), significance level (p-value), and interpretation.

Table 5.7

Variable correlation values

Variable Name	Spearman ρ	*P*-value
TP	0.6713	0.000
DO	0.4095	0.000
PH	0.5284	0.000

In this study, Spearman's rank correlation analysis revealed statistical relationships among temperature, dissolved oxygen, pH, and ammonia concentration. The results demonstrated statistically significant positive correlations ($p < 0.001$) between all three parameters and ammonia concentration. The Spearman correlation coefficient for temperature and ammonia reached 0.671, indicating a moderate-to-strong monotonic positive relationship. This finding may be attributed to enhanced ammonia volatilisation or intensified microbial activity under elevated water temperatures, thereby promoting ammonia accumulation.

Furthermore, pH showed a correlation coefficient of 0.528 with ammonia, indicating a moderately strong positive correlation. This phenomenon aligns with the theoretical principle that ammonia tends to exist in its free form in alkaline environments, where both its toxicity and concentration increase substantially at higher pH levels. Although comparatively weaker ($\rho = 0.409$), the correlation between dissolved oxygen and ammonia remained statistically significant. This may reflect constraints imposed by inherent aquatic characteristics or the influence of shared factors (e.g., eutrophication or aquatic organism metabolism) on dissolved oxygen fluctuations, indirectly affecting ammonia concentrations in the study area.

In conclusion, temperature, pH, and dissolved oxygen all demonstrated significant associations with aqueous ammonia concentrations to varying degrees. Therefore, these three parameters can be considered key predictive factors for modelling and forecasting ammonia concentrations, thereby providing a scientific basis and technical support for water quality assessment and management.

The heatmap() method of the Seaborn library generates a heatmap that illustrates the correlation between variables, effectively visualising the strength of these correlations, as depicted in Figure 5.2.

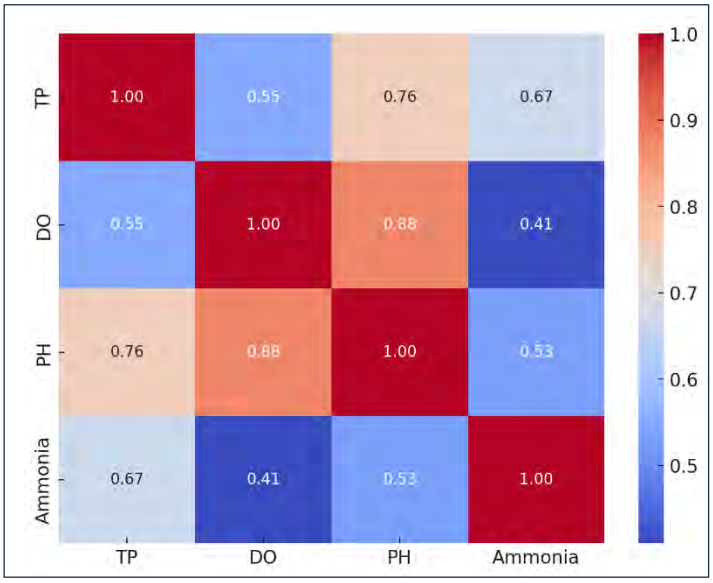

Figure 5.2. Heat map of variable correlations

The correlation heatmap above illustrates the strength and direction of correlations among variables using colour and numerical values from -1 to 1, where -1 indicates a perfect negative correlation, 1 a perfect positive correlation, and 0 no correlation. The colour gradient typically represents the magnitude of the correlation coefficient, with positive correlations shown in red, negative in blue, and no correlation in grey. This figure facilitates the rapid identification of pairs of substantially correlated variables and offers a clear framework for data analysis.

The strong association suggests that these factors better elucidate fluctuations in ammonia concentration. Consequently, temperature, dissolved oxygen, and pH values in the aquaculture environment can serve as fundamental factors for predicting ammonia concentration in the development of predictive models.

5.2.3.1 Data Normalization

The dataset is normalised using Min-Max Normalisation, which standardises variables with disparate units and value ranges to a uniform scale (e.g., [0,1] or [-1,1]), thereby mitigating the influence of dimensional discrepancies on model training and analytical outcomes. The results of normalisation are presented in Table 5.8.

Table 5.8

Sample table for data normalisation process

Created_at	Temperature (°C)	Dissolved Oxygen(g/ml)	PH	Ammonia(g/ml)
2021/6/19 0:00	0.152859	0.105493	0.203842	0.021545
2021/6/19 0:01	0.156401	0.157848	0.206831	0.02102
2021/6/19 0:01	0.156271	0.118966	0.235081	0.020939
2021/6/19 0:02	0.157251	0.953043	0.641651	0.021545

5.2.4 Hyperparameter Settings

Before hyperparameter optimisation, the data is partitioned into a training and validation set and a test set in an 80% to 20% ratio, with the test set comprising 20% of the overall dataset. The training and validation set was subsequently split into training and validation subsets at an 80%:20% ratio. Ultimately, the training set comprised 64% of the total data (10,278 instances), the validation set comprised 16% (12,847 instances), and the test set comprised 20%, thereby guaranteeing distinct subsets of data for training, validation, and testing the model. The studies employed a combination of stochastic search and cross-validation optimisation for hyperparameter tuning, utilising three cross-validations, 100 distinct configurations for each potential parameter combination, and a cumulative total of 300 fits. The studies revealed varying

training times and performance across several parameter combinations, ultimately identifying the best hyperparameter combination, as presented in Table 5.9.

Table 5.9

Results of hyperparameter optimisation

Parameter	Value
layer	2
Unit1	144
Unit2	124
Learning rate	0.001
Batch size	32

The optimal model comprises two LSTM layers: the first with 144 neurons and the second with 124 neurons. Experiments indicate that a learning rate of 0.001 effectively mitigates overfitting during training and improves the model's generalisation. Choosing a batch size of 32 during training also enhances model performance.

5.3 NadamClip Algorithm Training Model

The NadamClip method is incorporated into the LSTM model's training during the initial phase of the experiment to assess its efficacy in handling complex time-series data. NadamClip integrates the adaptive learning rate of the Nadam optimisation algorithm with U-Clip gradient clipping to mitigate gradient spikes during training. To thoroughly examine the impact of gradient clipping, the clipping threshold was set significantly high (e.g., 100) in the initial tests to replicate a training process with minimal effective gradient clipping.

In this setting, the LSTM model is essentially unconstrained by gradient clipping during training, and gradient spikes may arise during backpropagation, characterised by sudden increases in gradient magnitude at certain iterations. If such gradient spikes are not properly controlled, they may introduce instability into the training process, thereby impairing the model's ability to approach an optimal solution along a stable optimisation trajectory steadily. Moreover, in scenarios involving more complex data structures or longer input sequences, the risk of severe gradient amplification further increases. Under such conditions, gradient clipping serves as a preventive strategy that enhances the model's robustness and generalisation.

If the gradient clipping threshold is established at 100, the gradient remains untrimmed. Upon completion of 100 training iterations, the model's performance is assessed, yielding the findings presented in Table 5.10.

Table 5.10

Training model evaluation results with gradient threshold of 100

Evaluation methodology	**Value**
MAE	0.2772
RMSE	0.6752
R^2	0.9731
Gradient norm max	0.2887

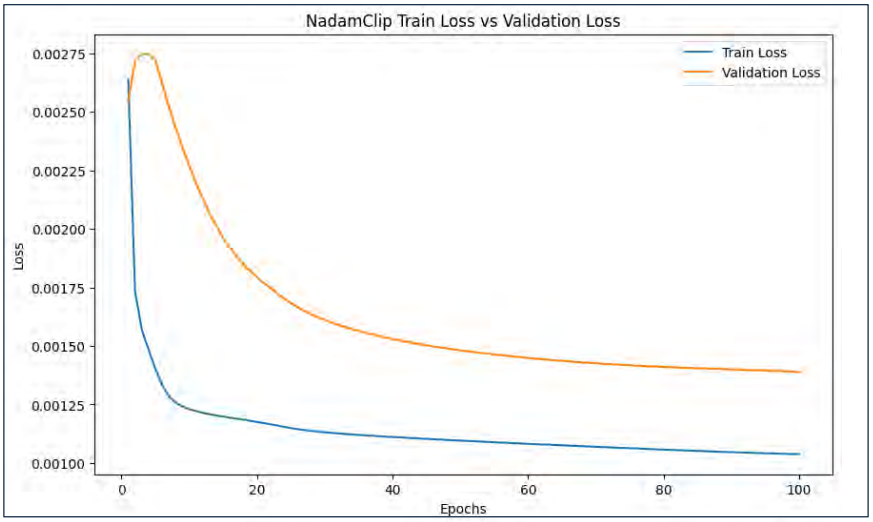

Figure 5.3. NadamClip of loss changes when clipping threshold is 100

Figure 5.3 illustrates the progression of training and validation losses over the course of the training phase. The figure shows that the training loss decreases rapidly in the initial phase and then stabilises as training progresses, indicating continuous improvement in the model's fit to the training set. Nonetheless, the validation loss exhibited notable fluctuations, particularly during the latter phases of training. The limited fluctuation range of validation loss suggests that the model does not exhibit significant overfitting and retains strong generalisation capability. The overall trend of the training and validation loss curves indicates that the model maintains a stable optimal state throughout the learning process, thereby preserving its generalisation capability.

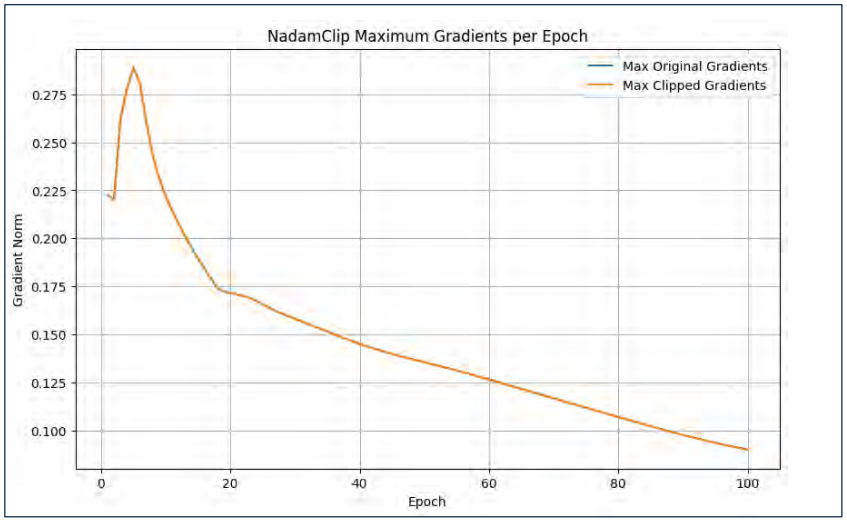

Figure 5.4. Gradient norm change when clipping threshold is 100

Figure 5.4 illustrates a gradient variation plot comparing the original maximum gradient with the clipped maximum gradient at each epoch, assessing the efficacy of gradient clipping in constraining gradient magnitudes and mitigating gradient spikes, thereby ensuring training stability. This comparison allows for the assessment of gradient clipping's impact on the optimisation process and its efficacy in enhancing outcomes. The figure shows that the maximum original gradient and the maximum clipped gradient coincide, indicating that the gradient is not genuinely clipped. Concurrently, the maximum gradient value consistently increases or remains stable during training, indicating that the model's optimisation progressively approaches the ideal solution as training progresses. This trend suggests that, despite the rise in training losses, the model's optimisation remains effective, with stable gradient updates. The stability of the maximum gradient indicates that the model progressively converges during training, demonstrating a successful optimisation process that subsequently improves model performance.

The original gradients exhibited a noticeable upward spike. This sudden increase may delay parameter updates, affecting the stability of training and the accuracy of predictions. To enhance the model's training stability and prediction accuracy, the subsequent experiment will investigate the optimal gradient clipping range by varying the gradient clipping threshold. This process will help identify the optimal gradient clipping threshold, ensuring training stability while maximising the model's predictive performance. By preventing excessive gradient fluctuations, it enhances the model's overall capability.

5.4 Gradient Clipping Range Experiment

This experiment aims to identify a gradient clipping range that optimally balances mitigating gradient spikes and preserving the model's learning capacity. Initially, the gradient threshold was set at 0.25, a value selected based on the observed distribution of raw gradient magnitudes, in which the maximum gradients typically appeared around 0.28, 0.28, 0.27, and similar values, indicating that these represent near-extreme cases. Setting the threshold at 0.25, therefore, avoids these extreme gradient values while still encompassing the majority of normal gradient behaviour. Furthermore, preliminary validations demonstrated that thresholds such as 0.26 and 0.27 do not yield optimal performance, thereby supporting 0.25 as an effective choice for mitigating gradient spikes without significantly diminishing the usable gradient magnitude. The associated model performance metrics, including RMSE, MAE, and R^2, were subsequently documented to evaluate this threshold's effectiveness. Employing absolute value reduction to modify the gradient threshold facilitates precise, consistent adjustments throughout the experiment. The application of absolute-value reduction ensures a uniform step size for each modification, facilitating a more

seamless examination of the effects of threshold alterations on model performance. This facilitates a more precise examination of model performance trends as the threshold varies, hence preventing unstable or erroneous outcomes caused by excessive threshold fluctuations. Particularly as the gradient threshold approaches the optimal range, small absolute variations are beneficial for identifying the ideal gradient clipping value and mitigating model performance fluctuations resulting from excessive modifications.

i. The gradient threshold is 0.25

Set the gradient threshold to 0.25 as a large clipping value, meaning that during training, the gradient will only be clipped if its absolute value exceeds 0.25. Otherwise, the gradient remains unchanged. After training iterations with epoch=100, the model's performance is evaluated, and the results are shown in Table 5.11.

Table 5.11

Training model evaluation results with gradient threshold of 0.25

Evaluation methodology	**Value**
MAE	0.2723
RMSE	0.6707
R^2	0.9735
Gradient norm max	0.2626

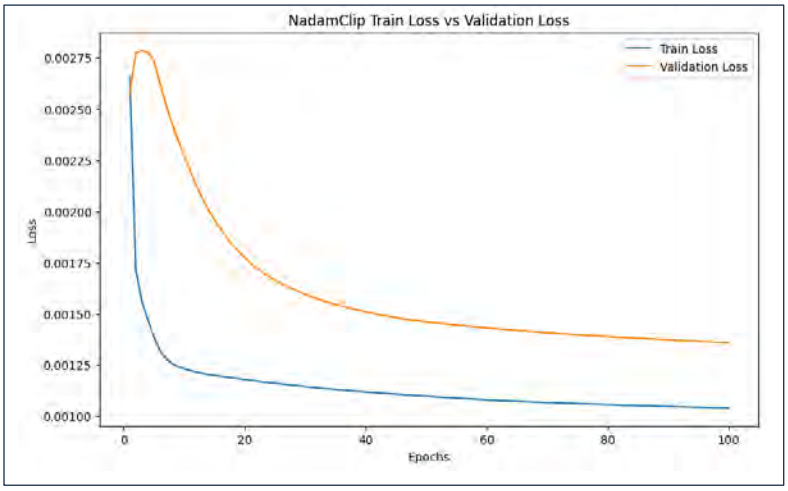

Figure 5.5. NadamClip of loss changes when the gradient threshold is 0.25

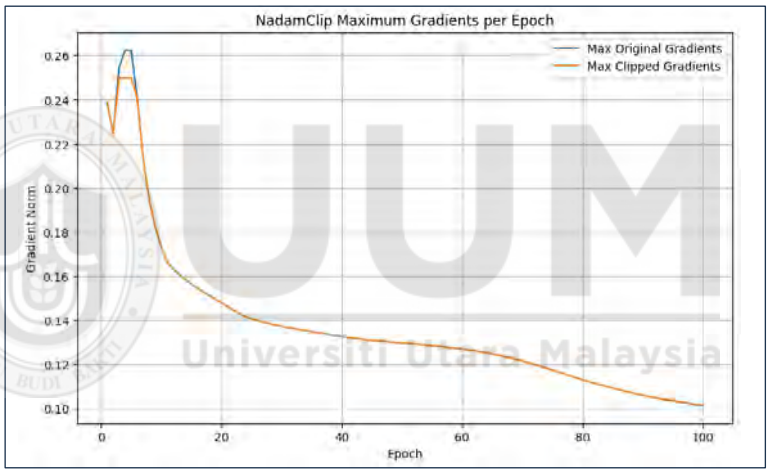

Figure 5.6. Gradient norm changes when the gradient threshold is 0.25

The loss changes plots for the models with gradient thresholds of 100 and 0.25, as illustrated in Figures 5.5 and 5.6, are largely consistent, indicating no major variation in the models' training processes under these two gradient threshold settings. Nevertheless, in evaluating performance measures, a gradient threshold of 0.25 marginally outperforms 100, particularly for MAE, RMSE, and R^2.

When the gradient threshold is set at 100, the model exhibits a Mean Absolute Error (MAE) of 0.2772, a Root Mean Square Error (RMSE) of 0.6752, an R^2 value of 0.9731,

and a maximum gradient norm of 0.2887. Conversely, with a gradient threshold of 0.25, the MAE is marginally reduced (0.2723), the RMSE is also marginally reduced (0.6708), R^2 is marginally increased (0.9735), and the highest gradient norm is 0.2626, which is below the threshold of 0.25. This indicates that, while the loss curves at both settings appear comparable, the model with a gradient threshold of 0.25 exhibits marginally superior fitting accuracy and training stability across measures.

Despite the threshold of 100 not inducing notable training variations, there was virtually no constraint on gradient updating, with the maximum gradient attaining 0.2887. This could lead to unstable training due to excessively large gradients. With a gradient threshold of 0.25, the maximum gradient norm is minimal (0.2626), indicating that the model's gradient updates are effectively constrained by clipping, thereby mitigating the detrimental effects of excessive gradient updates on training and enhancing model stability.

Consequently, although the loss graphs of the two models are similar, the performance index indicates that the model with a gradient threshold of 0.25 has superior training efficacy. This threshold optimally balances gradient clipping and model training accuracy, suggesting that 0.25 is the superior option.

ii. The gradient threshold is 0.20

To enhance model performance, the gradient threshold was set to 0.2 to investigate the impact of reduced gradient shear on model training. Modifying the threshold from 0.25 to 0.2 can further constrict the extent of the gradient update. After 100 training epochs, the model's performance is assessed, yielding results shown in Table 5.12.

Table 5.12

Training model evaluation results with gradient threshold of 0.20

Evaluation methodology	Value
MAE	0.2631
RMSE	0.6622
R^2	0.9741
Gradient norm max	0.2619

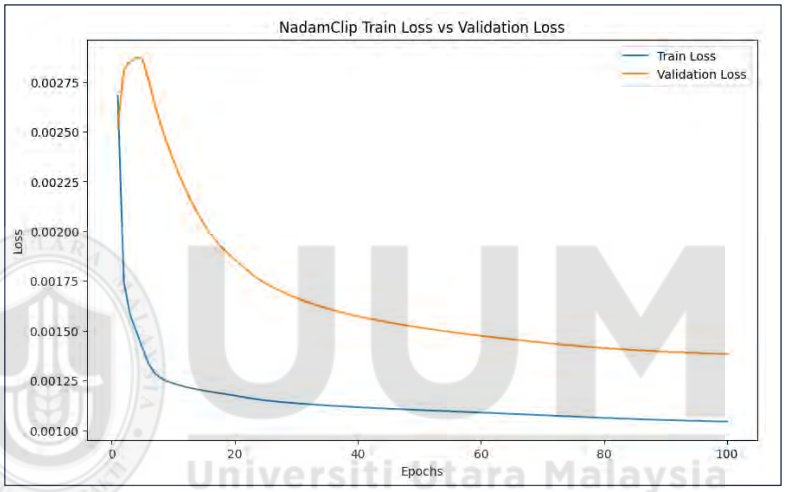

Figure 5.7. NadamClip of loss changes when the gradient threshold is 0.20

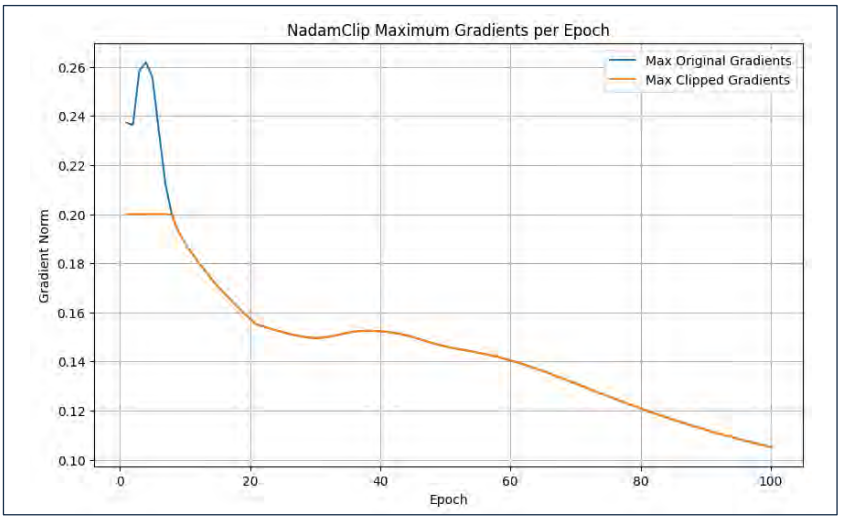

Figure 5.8. Gradient norm changes when the gradient threshold is 0.20

The results from Figures 5.7 and 5.8 indicate that when the gradient threshold is set to 0.2, the model's performance improves relative to 0.25. When the gradient threshold is set at 0.2, the model exhibits a Mean Absolute Error (MAE) of 0.2631, a Root Mean Square Error (RMSE) of 0.6622, an R^2 value of 0.9741, and a maximum gradient norm of 0.2619. This demonstrates enhanced performance relative to MAE (0.2723), RMSE (0.6708), and R^2 (0.9735) at a threshold of 0.25, along with a greater maximum gradient norm of 0.2626. Specifically, for RMSE and R^2, models with a gradient threshold of 0.2 exhibit lower errors and a superior fit, while the maximum gradient norm approaches the established threshold, indicating more efficient gradient management within a narrower range. This modification indicates that reducing the gradient threshold further constrains the magnitude of gradient updates, facilitates more stable training, and enhances both accuracy and fit. Consequently, a gradient threshold of 0.2 performed better than 0.25 in this experiment.

iii. The gradient threshold is 0.15

The gradient threshold was set to 0.15 to improve model training stability. The model's performance was assessed after 100 epochs of training, and the results are presented in Table 5.13.

Table 5.13

Training model evaluation results with gradient threshold of 0.15

Evaluation methodology	Value
MAE	0.2679
RMSE	0.6687
R^2	0.9736
Gradient norm max	0.3125

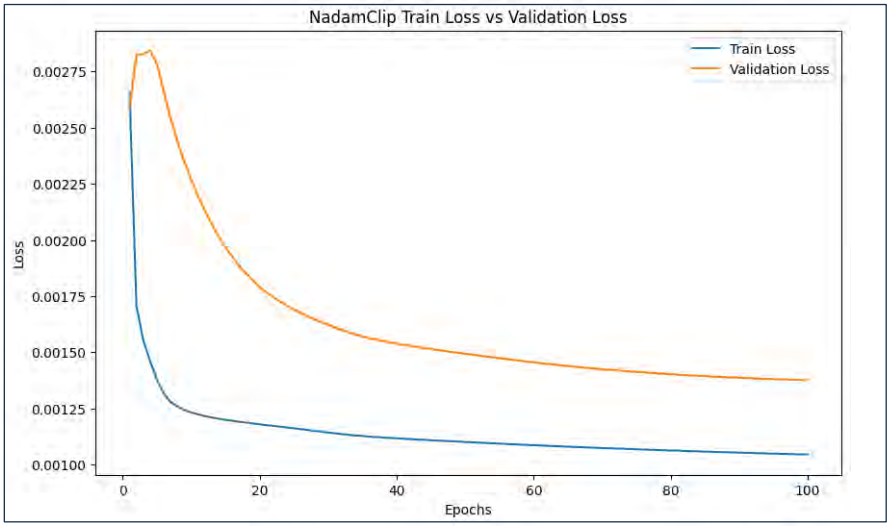

Figure 5.9. NadamClip of loss changes when the gradient threshold is 0.15

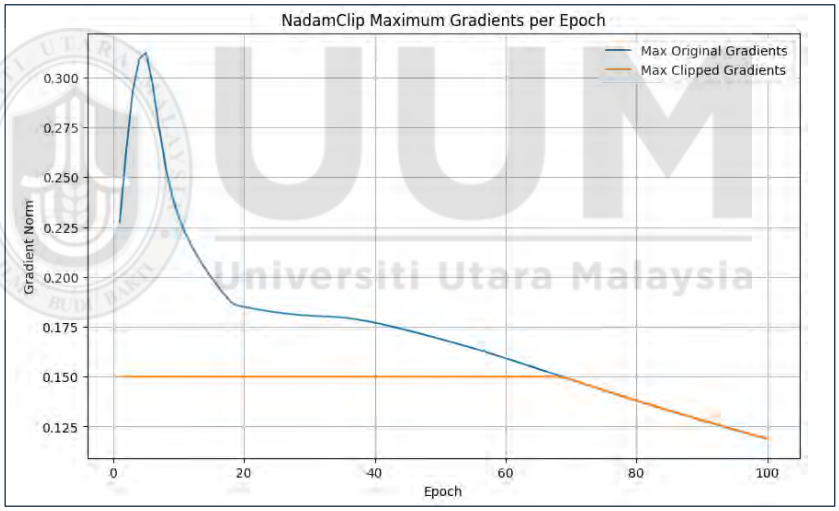

Figure 5.10. Gradient norm changes when the gradient threshold is 0.15

The results from Figures 5.9 and 5.10 indicate that when the gradient threshold is set at 0.15, the model exhibits distinct behaviour compared to when it is set at 0.2 or 0.25. When the gradient threshold is set at 0.15, the model exhibits a Mean Absolute Error (MAE) of 0.2679, a Root Mean Square Error (RMSE) of 0.6687, an R^2 value of 0.9736, and a maximum gradient norm of 0.3125. The results indicate a marginal decline in

performance relative to the configuration utilising a gradient threshold of 0.2 (MAE = 0.2631, RMSE = 0.6622, R^2 = 0.9741).

The rise in the constant mode of the maximum gradient (from 0.2619 to 0.3125) indicates that reduced gradient thresholds, although constraining the overall gradient magnitude, could heighten instability during training due to the cumulative impact of clipping and optimiser interactions, potentially leading to a higher maximum gradient. Consequently, a persistent reduction in the gradient threshold impairs the model's learning capacity and performance, even when gradients are clipped more frequently.

In conclusion, an excessively small gradient threshold may unduly restrict model training and impede the model's ability to learn from the input. Consequently, further reductions in the gradient threshold are suboptimal, particularly with respect to overall performance.

The optimal performance, as indicated by the testing results, is attained with a gradient threshold of 0.2. Reducing the threshold to 0.15 increases MAE and RMSE and reduces R^2, suggesting that severe gradient clipping hampers model training and diminishes performance. Consequently, a more precise experimental calibration will be conducted between 0.2 and 0.25 to evaluate the equivalents of 0.21, 0.22, and 0.23. The decision to perform additional trials within the range of 0.2-0.25, rather than 0.15-0.2, is primarily founded on the subsequent observations and experimental findings:

The gradient threshold adjustment between 0.2 and 0.25 demonstrates optimal balance and stability. Reducing the gradient threshold from 0.25 to 0.2 markedly enhances the model's performance. The MAE and RMSE at 0.2 are reduced, the R^2 is elevated, and

the maximum gradient norm approaches 0.2, signifying that the gradient clipping is modest. This helps the model maintain good training stability while preventing excessive clipping. Subsequent experimental findings indicate that 0.2 serves as an optimal equilibrium point, successfully constraining excessive gradient fluctuations while preserving enough learning capacity.

Nevertheless, when the gradient threshold is reduced to 0.15, the MAE and RMSE increase while R^2 decreases, indicating that an excessively low gradient threshold impairs model performance.

Conversely, increasing the gradient threshold from 0.25 to 0.2 resulted in a notable improvement in performance, whereas decreasing it from 0.2 to 0.15 led to a deterioration in performance. Consequently, fine-tuning the gradient threshold between 0.2 and 0.25, using 0.2 as the optimal initial reference, enhances the pursuit of the ideal value. This interval can enhance the model's fitting capability while maintaining stability. Subsequently, the gradient threshold within the range of 0.2-0.25 is subjected to further testing.

iv. The gradient threshold is 0.21

The gradient threshold is set to 0.21 to improve model training stability. After 100 epochs of iterative training, the model's performance is assessed, and the results are presented in Table 5.14.

Table 5.14

Training model evaluation results with gradient threshold of 0.21

Evaluation methodology	Value
MAE	0.2679
RMSE	0.6687
R^2	0.9736
Gradient norm max	0.3125

Loss changes (Figure 5.11) and gradient changes (Figure 5.12) during training are as follows:

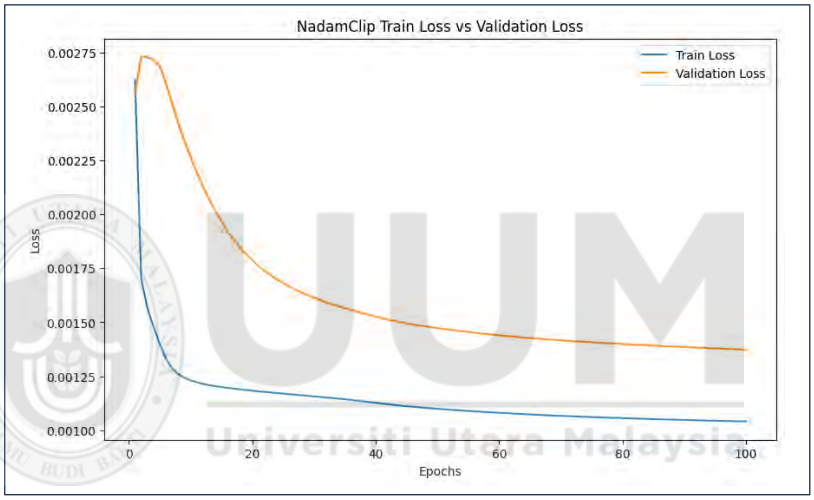

Figure 5.11. NadamClip of loss changes when the gradient threshold is 0.21

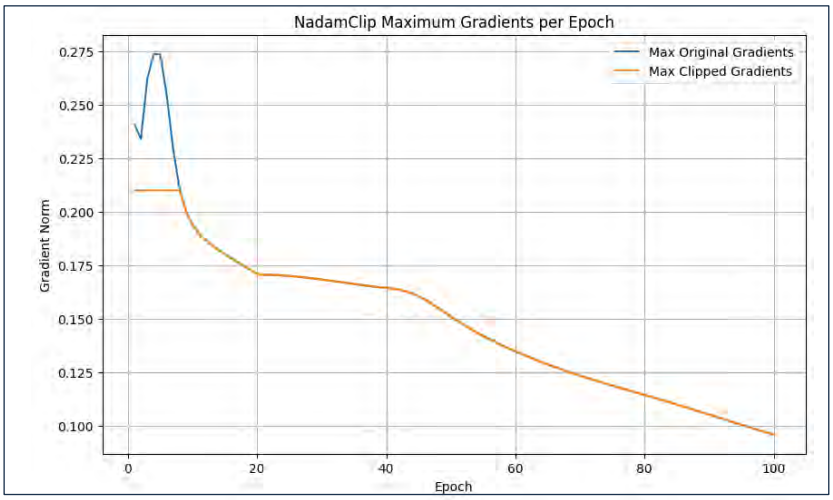

Figure 5.12. Gradient norm changes when the gradient threshold is 0.21

v. The gradient threshold is 0.22

Set the gradient threshold to 0.22. After training iterations with epoch=100, the performance of the model is evaluated, and the results are obtained as in Table 5.15:

Table 5.15

Training model evaluation results with gradient threshold of 0.22

Evaluation methodology	**Value**
MAE	0.2675
RMSE	0.6717
R^2	0.9734
Gradient norm max	0.2548

Loss changes (Figure 5.13) and gradient changes (Figure 5.14) during training are as follows:

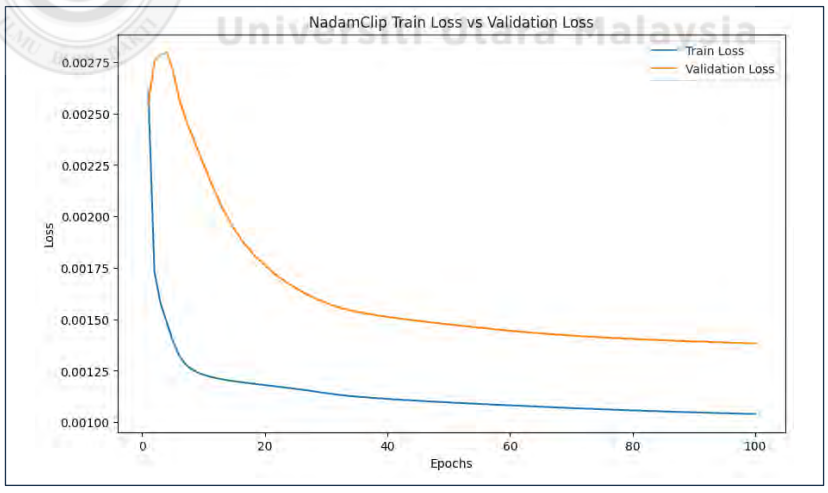

Figure 5.13. NadamClip of loss changes when the gradient threshold is 0.22

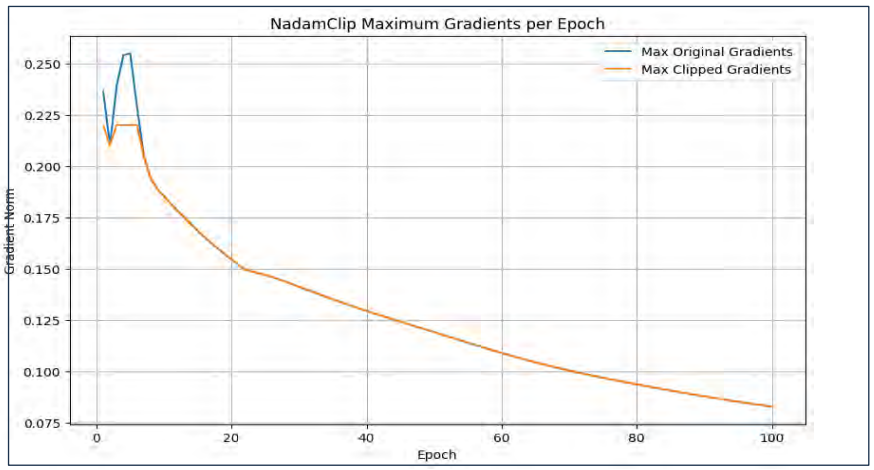

Figure 5.14. Gradient max changes when the gradient threshold is 0.22

vi. The gradient threshold is 0.23

Set the gradient threshold to 0.23. After training iterations with epoch=100, the performance of the model is evaluated, and the results are obtained as in Table 5.16:

Table 5.16

Training model evaluation results with gradient threshold of 0.23

Evaluation methodology	**Value**
MAE	0.2649
RMSE	0.6645
R^2	0.9740
Gradient norm max	0.2582

Loss changes (Figure 5.15) and gradient changes (Figure 5.16) during training are as follows:

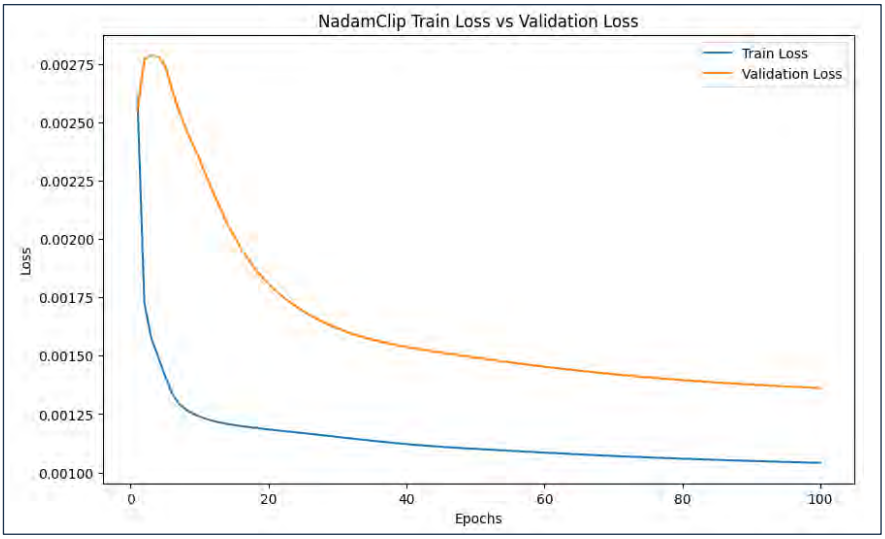

Figure 5.15. NadamClip of loss changes when the gradient threshold is 0.23

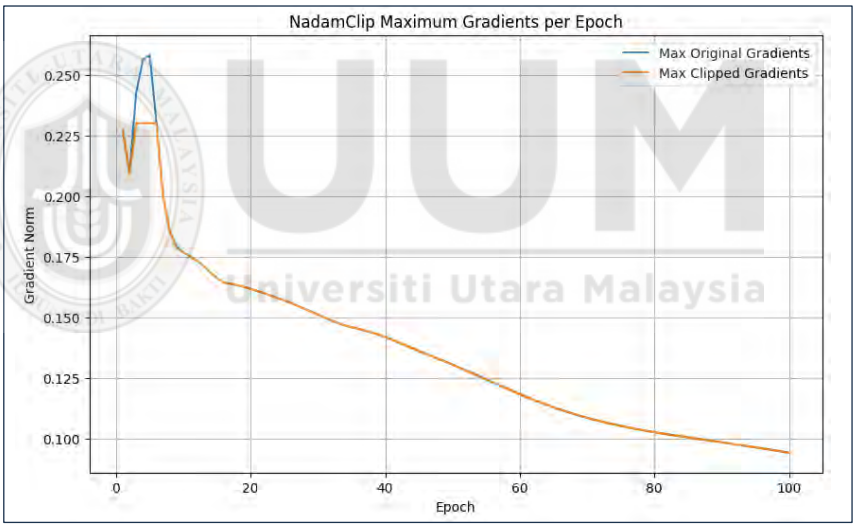

Figure 5.16. Gradient norm changes when the gradient threshold is 0.23

vii. The gradient threshold is 0.24

Set the gradient threshold to 0.24. After training iterations with epoch=100, the performance of the model is evaluated, and the results are obtained as in Table 5.17.

Table 5.17

Training model evaluation results with gradient threshold of 0.24

Evaluation methodology	Value
MAE	0.2644
RMSE	0.6595
R^2	0.9744
Gradient norm max	0.2895

Loss changes (Figure 5.17) and gradient changes (Figure 5.18) during training are as follows:

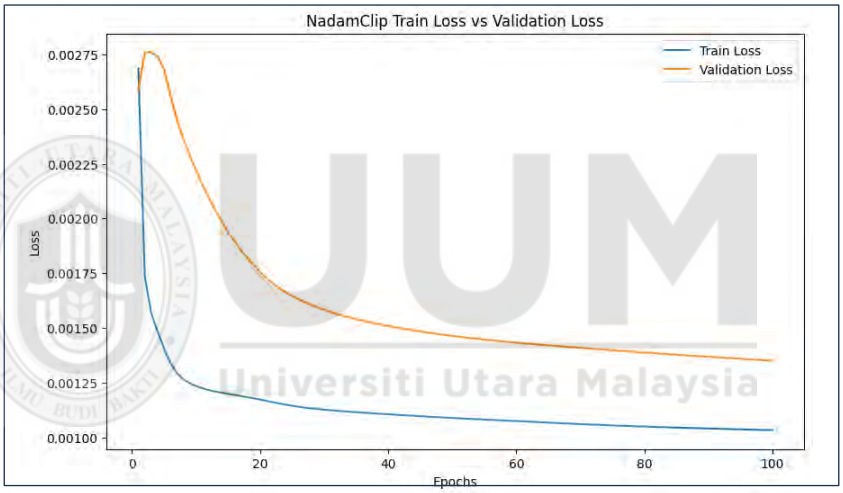

Figure 5.17. NadamClip of loss changes when the gradient threshold is 0.24

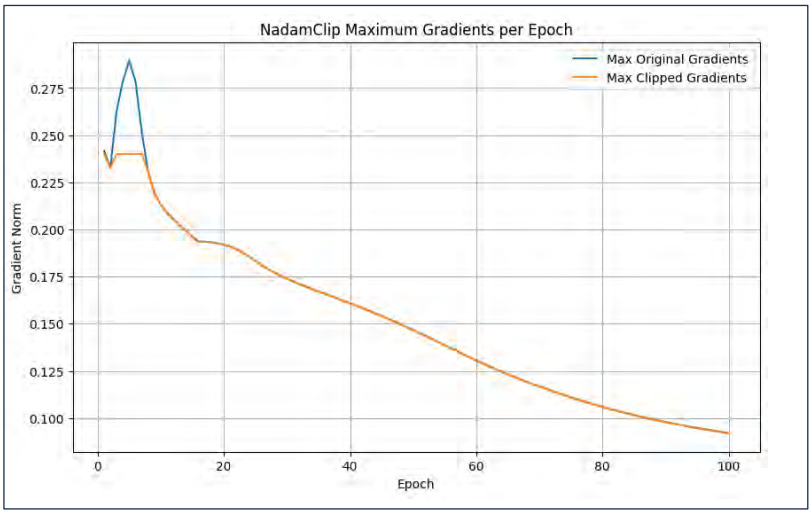

Figure 5.18. Gradient norm changes when the gradient threshold is 0.24

All the above training results are summarised in Table 5.18.

Table 5.18

Summarized results

Experiment No.	Epoch	Clipping Threshold	RMSE	MAE	R^2
1	100	100	0.6752	0.2772	0.9731
2	100	0.25	0.6708	0.2723	0.9735
3	100	0.24	0.6595	0.2644	0.9743
4	100	0.23	0.6645	0.2649	0.9740
5	100	0.22	0.6717	0.2675	0.9734
6	100	0.21	0.6688	0.2711	0.9736
7	100	0.2	0.6622	0.2631	0.9741
8	100	0.15	0.6687	0.2679	0.9736

The experimental results indicate that the choice of the gradient clipping threshold yields a slight improvement in model performance. As shown in Table 5.17, the model achieved optimal training performance in this study when the gradient clipping threshold was set to 0.24, resulting in the lowest fitting error while effectively modelling the data and maintaining a high R^2 value. The highest gradient norm remains within a tolerable range, and there are no gradient spikes. Upon further reducing the gradient threshold to 0.23, the alterations in MAE and RMSE remain negligible; nevertheless, the R^2 value experiences a minor decline to 0.9740, and the maximum gradient norm diminishes to 0.2582, signifying a further inhibition of gradient updating. Despite the severe nature of gradient clipping, the model's fitting capacity is somewhat diminished, likely due to the excessive restriction on gradient updates, which constrains learning and results in a small reduction in training accuracy.

As the gradient threshold decreases to 0.22, the model's MAE escalates to 0.2675, the RMSE increases to 0.6717, and the R^2 diminishes to 0.9734. Currently, the gradient norm has decreased to 0.2548, indicating that gradient clipping has become more stringent. This leads to the model's inadequate use of gradient information during training, reduced fitting efficacy, and increased training error. Ultimately, when the gradient threshold is set to 0.21, both MAE and RMSE increase significantly, reaching 0.2711 and 0.6688, respectively, while R^2 declines to 0.9736, somewhat lower than the value at 0.24. This outcome indicates that further reducing the gradient threshold renders the training process excessively conservative, thereby constraining the model's capacity to discern complex relationships in the data properly.

In summary, a dataset-specific gradient clipping threshold of 0.24, empirically optimal for this study rather than a universal value, yields the best overall performance and the most effective training outcomes. As the threshold decreases, particularly below 0.22, the model's performance deteriorates, indicating that excessive gradient clipping impairs the model's learning capacity. The ideal gradient threshold should range from 0.2 to 0.25, with 0.24 optimal in this experiment, as it enhances the model's fitting and generalisation capabilities while ensuring training stability. The performance variations of NadamClip under different gradient clipping thresholds arise from the intrinsic trade-off between optimisation stability and effective learning capacity. Excessively large thresholds fail to adequately suppress extreme gradient updates, leading to unstable convergence. In contrast, overly small thresholds overly restrict parameter updates and hinder the model’s ability to learn complex patterns. Consequently, only an intermediate threshold achieves an optimal balance between stability and learning efficiency, resulting in the best predictive performance.

Figure 5.19 compares the actual and forecast values when the gradient threshold is set to 0.24.

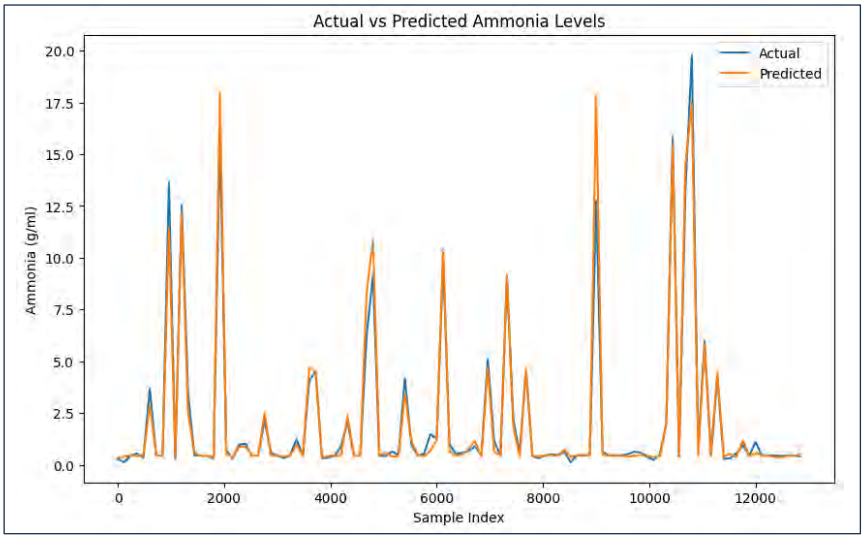

Figure 5.19. NadamClip training model real value and predicted value comparison

The anticipated value of the NadamClip optimiser closely aligns with the actual value, and the curve's trajectory nearly coincides, indicating that the NadamClip optimiser effectively fits the data during training, resulting in more precise predictions. Throughout the entire data range, NadamClip's predicted values exhibit reduced fluctuations and align consistently with the true values, demonstrating its superior performance in model fitting and predictive accuracy.

5.5 Analysis of Experimental Results

A thorough performance comparison of NadamClip with other prevalent optimisation methods must be conducted under identical settings to illustrate its advantages and disadvantages. A series of comparative tests is devised to evaluate NadamClip against several other optimisers (e.g., Adam, SGD with momentum, RMSprop) and gradient-clipping optimisers (e.g., SGD with gradient clipping).

The primary aim of this experiment is to assess whether NadamClip demonstrates superior performance in handling intricate time-series data, particularly within LSTM models, through a multi-dimensional comparison. The experiment will be assessed and contrasted from multiple critical viewpoints, including stability and predictive accuracy. The experiments will meticulously control the training conditions of the algorithms (e.g., identical datasets, network architectures, initial conditions) to ensure the comparability of the experimental outcomes. The loss variation curves for each method are drawn, and essential performance metrics, including MAE and RMSE, are documented for the training, validation, and test sets. The strengths and drawbacks of each algorithm are evident in this data, particularly in their performance on difficult time-series tasks.

5.5.1 Analysis of Algorithmic Results

5.5.1.1 Comparison with the Stochastic Gradient Descent Algorithm

This segment of the experiment examines the efficacy of the NadamClip method compared with conventional gradient descent algorithms. The prevalent algorithms comprise SGD, AdaGrad, AdaDelta, RMSProp, Adam, Nadam and AdamW. The aforementioned techniques have broad applicability and research significance in deep learning optimisation tasks. By comparing these algorithms, we can thoroughly examine the convergence characteristics, stability, and application of NadamClip across various training conditions, thereby gaining a deeper understanding of its attributes and potential.

i. SGD Algorithm

SGD is essential in algorithms because it randomly selects a sample or a small batch of samples at each iteration to compute the gradient, significantly reducing computational overhead and enabling model training on large-scale datasets. Simultaneously, SGD can avoid local gradient optima by incorporating noise, thereby facilitating convergence to a global or near-optimal solution.
To evaluate the merits and drawbacks of the SGD and NadamClip optimisation techniques, this experiment was conducted over 100 epochs, with all other parameters aligned with the hyperparameters of the NadamClip training model. SGD was used for training, and differences in stability and ultimate performance between the two were observed. Following 100 training epochs, the model's performance is assessed, yielding the findings presented in Table 5.19.

Table 5.19

SGD training model performance results

Evaluation methodology	Value
MAE	0.7505734623731953
RMSE	1.1711739274031217
R^2	0.9190955541603905

A comprehensive comparison of the experimental outcomes of the two optimisation algorithms, SGD and NadamClip, demonstrates that NadamClip offers significant benefits across several respects. The MAE of the comparable SGD was 0.7506. In contrast, the MAE of NadamClip decreased markedly to 0.2644, demonstrating that NadamClip achieved lower absolute errors between predicted and actual values, thereby greatly enhancing prediction accuracy. Correspondingly, the RMSE of SGD

was 1.1712, whereas the RMSE of NadamClip decreased to 0.6595, thereby substantiating the significant impact of NadamClip on enhancing prediction accuracy. NadamClip's R^2 Score of 0.9743 markedly surpasses SGD's score of 0.9191, signifying superior model fit and explanatory power for NadamClip. In conclusion, relative to SGD, NadamClip demonstrates considerable improvements in predictive accuracy, convergence rate, and model stability, thereby affirming its efficacy as an optimisation algorithm and offering a more efficient optimisation strategy for model training.

The loss changes (Figure 5.20) in the training process are shown as follows:

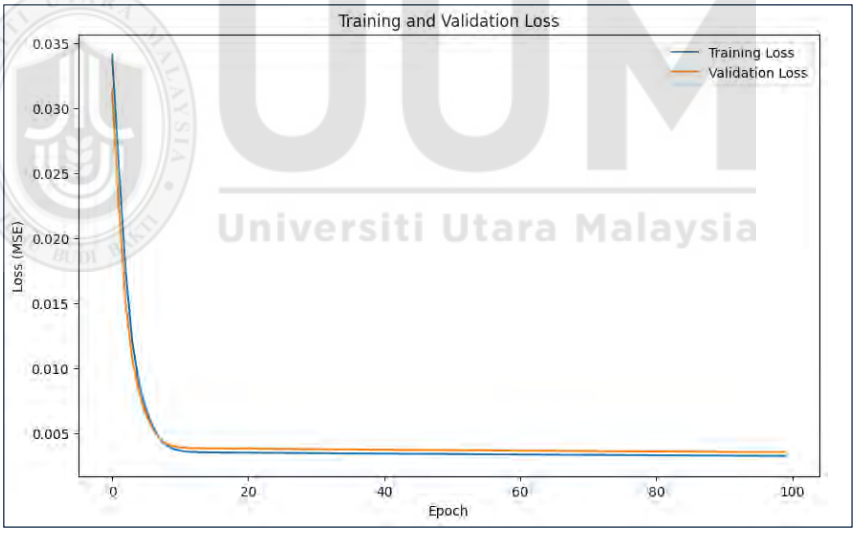

Figure 5.20. SGD of loss changes

Figure 5.20 illustrates that the convergence rate of SGD is inferior, and the model's fitting on the training set is slower. Despite a gradual reduction in losses and a smoother transition, SGD exhibits sluggish optimisation. Despite the absence of large fluctuations or increases in SGD losses, the overall drop was minimal, suggesting a marginally diminished capacity for generalisation. Consequently, NadamClip

typically attains superior model performance more rapidly under the same training settings.

A comparative analysis of the experimental outcomes of SGD and NadamClip reveals that NadamClip offers significant improvements in prediction accuracy, convergence speed, and model stability. NadamClip converges more rapidly, offering superior fitting capabilities and data interpretation performance. Despite the stability of the SGD optimisation process, its convergence rate is sluggish, and its generalisation capacity is comparatively limited, resulting in somewhat inadequate forecast accuracy. NadamClip is a superior and more dependable optimisation technique.

ii. AdaGrad Algorithm

AdaGrad, an early adaptive learning rate optimisation approach, has garnered significant attention in the study of optimisation algorithms due to its superior performance in sparse data contexts. The technique adeptly addresses inconsistent step sizes in parameter updates by aggregating historical gradient sums of squares and dynamically adjusting the learning rate for each parameter. This study compares AdaGrad with NadamClip to assess performance.

To compare the advantages and disadvantages of AdaGrad and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. AdaGrad was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, and the results are shown in Table 5.20.

Table 5.20

AdaGrad training model performance results

Evaluation methodology	Value
MAE	0.7821207436452577
RMSE	1.2046766764728003
R^2	0.9144006236449291

The experimental results indicate that NadamClip outperforms AdaGrad across all metrics. Specifically, NadamClip achieves an MAE of 0.2643, significantly lower than AdaGrad's 0.7821. In terms of RMSE, NadamClip records 0.6595, which is markedly lower than AdaGrad's 1.2047. Furthermore, NadamClip attains a R^2 score of 0.9743, clearly superior to AdaGrad's 0.9144. NadamClip has an R^2 score of 0.9743, markedly surpassing AdaGrad's score of 0.9144. This indicates that NadamClip delivers superior performance in model fitting accuracy and error management, effectively capturing data properties and achieving enhanced predictive power.

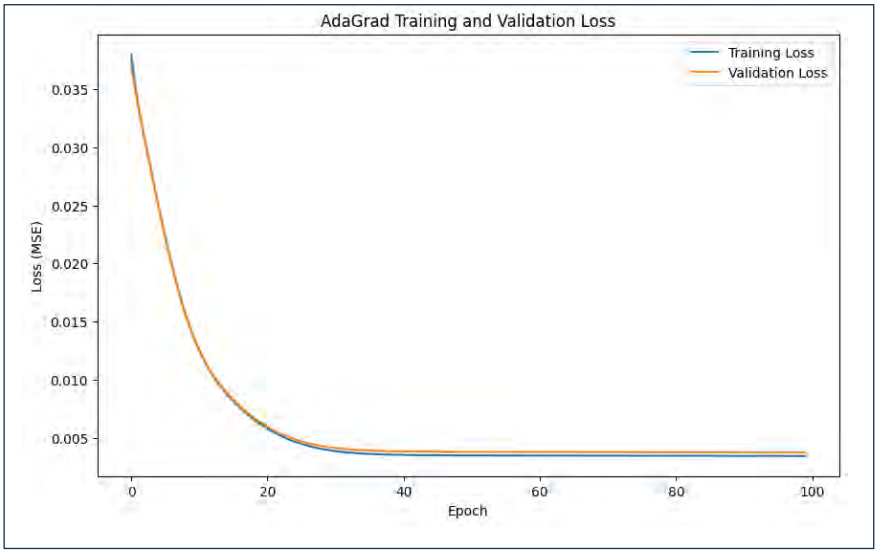

Figure 5.21. AdaGrad of loss changes

Figure 5.21 illustrates the progression of the AdaGrad algorithm's training and validation losses over epochs. The training and validation losses decline swiftly in the initial phase, indicating that the model effectively captures the data properties early on. After approximately 20 iterations, the loss values stabilise, indicating the model is nearing convergence. The training and validation loss curves are closely aligned, suggesting the model did not experience substantial overfitting or underfitting during training.

The loss curve graphs and evaluation data indicate that NadamClip exhibits greater optimisation performance than AdaGrad. In the AdaGrad loss curve, while the training and validation losses stabilise, they decrease at a slower rate, resulting in a final loss value that is considerably greater than that of NadamClip. NadamClip outperforms AdaGrad across assessment metrics, particularly MAE, RMSE, and R^2, with a notable advantage in error control (MAE and RMSE). This indicates that NadamClip can optimise model parameters more effectively, resulting in reduced error and improved model fitting accuracy while maintaining training speed. Consequently, it can be inferred that NadamClip surpasses AdaGrad in this experimental work.

iii. AdaDelta Algorithm

AdaDelta, an enhanced version of AdaGrad, mitigates the rapid deterioration in learning rate characteristic of AdaGrad by using sliding averaging, thereby augmenting the algorithm's robustness during prolonged training. The fundamental concept is to restrict the gradient's update step size, resulting in a more gradual and stable parameter adjustment, particularly advantageous for managing high-

dimensional sparse data. This experiment compares NadamClip with AdaDelta to evaluate its performance across various workloads and datasets.

To compare the advantages and disadvantages of AdaDelta and NadamClip optimisation algorithms, 100 epochs were used in this experiment, and other parameters were set to the hyperparameters of the NadamClip training model. AdaDetla was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, and the results are shown in Table 5.21.

Table 5.21

AdaDetla training model performance results

Evaluation methodology	Value
MAE	0.9210511548045627
RMSE	1.4274229448826565
R^2	0.8798192407754606

The experimental findings indicate that NadamClip much surpasses AdaDelta across multiple metrics. The MAE of NadamClip is 0.2643, significantly lower than AdaDelta's 0.9211, highlighting its superior prediction accuracy. In terms of RMSE, NadamClip registers 0.6595, whereas AdaDelta registers 1.4274, underscoring NadamClip's efficacy in error management. Furthermore, the R^2 score for NadamClip is 0.9743, markedly higher than AdaDelta's 0.8798, signifying its enhanced model fitting capability. This suggests that NadamClip surpasses AdaDelta in both optimisation efficiency and generalisation performance, thereby more effectively fulfilling the requirements for precise prediction and stable optimisation.

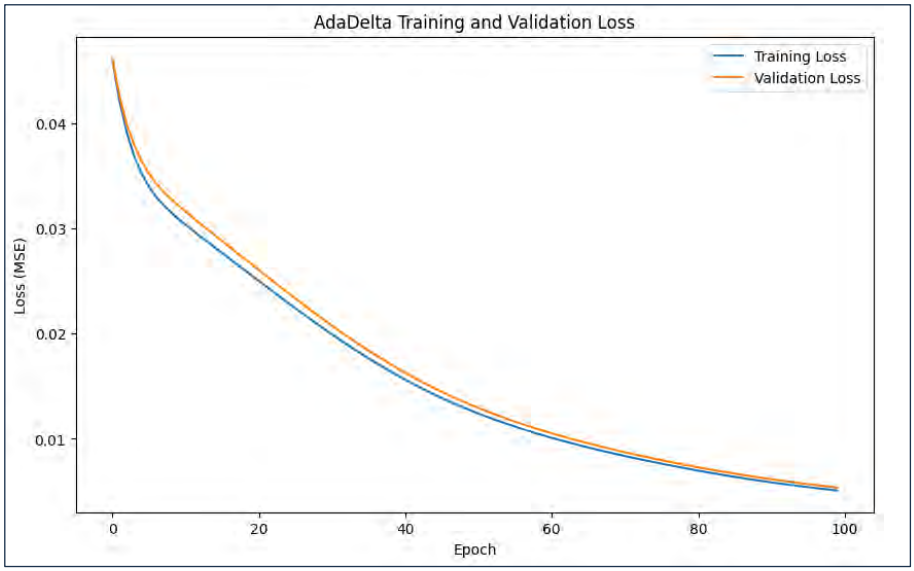

Figure 5.22. AdaDelta of loss changes

Figure 5.22 illustrates the trend of training loss and validation loss over the epochs of the AdaDelta algorithm during the training process. The figure shows a consistent decline in both training and validation losses, indicating that the model is persistently learning and optimising. Compared with other optimisation techniques, AdaDelta exhibits a more gradual decline in loss, particularly during the later stages of iteration, where the decrease in loss values becomes significantly constrained. This indicates that AdaDelta's update strategy is more cautious, leading to slower convergence. Furthermore, the training and validation loss curves remain consistently close, indicating that the model shows no evident overfitting or underfitting during training and demonstrating commendable stability.

The assessment data and loss curves indicate that NadamClip substantially surpasses AdaDelta regarding optimisation performance and convergence efficiency. NadamClip exhibits superior performance in evaluation metrics, demonstrating enhanced model fitting accuracy and improved error control; concurrently, its loss

curves decrease more rapidly and converge more consistently, with validation loss increasingly aligning with training loss, indicating improved generalisation. The validation loss converges with the training loss, indicating improved generalisation. Conversely, AdaDelta's loss curve declines at a slower rate, and the evaluation metrics indicate that its model fitting efficacy and error optimisation capacity are comparatively inferior, with its convergence performance exhibiting a more conservative nature. NadamClip demonstrates exceptional optimisation performance in this assignment.

iv. RMSProp Algorithm

In the examination of optimisation algorithms, RMSProp, a traditional adaptive learning rate algorithm, effectively addresses the issue of excessive learning rate decay in the AdaGrad algorithm by incorporating an exponentially weighted moving average to smooth the gradient squares. RMSProp is widely used in numerous deep learning applications due to its exceptional performance in handling non-smooth objectives and its accelerated convergence rate. This experiment compares NadamClip with RMSProp to thoroughly assess its performance.

To compare the advantages and disadvantages of RMSProp and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. RMSProp was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, and the results are shown in Table 5.22.

Table 5.22

RMSProp training model performance results

Evaluation methodology	Value
MAE	0.3220789522245996
RMSE	0.6784517512974149
R^2	0.9728501439025696

NadamClip and RMSProp have strong optimisation capabilities in the assessment criteria; nevertheless, NadamClip has superior overall performance. The RMSE of NadamClip is 0.6595, marginally superior to RMSProp's 0.6785, indicating a reduced disparity in error control between the two. NadamClip's R^2 score of 0.9743 surpasses RMSProp's 0.9729, demonstrating that NadamClip retains superior model-fitting accuracy. These findings indicate that whereas RMSProp approaches NadamClip in certain respects, the latter exhibits superior overall optimisation capability and generalisation performance.

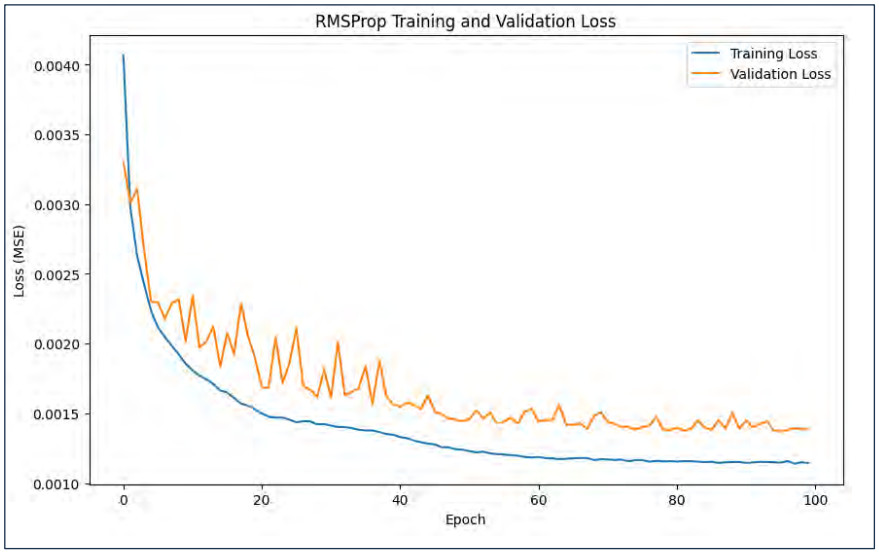

Figure 5.23. RMSProp of loss changes

Figure 5.23 illustrates the pattern of training loss and validation loss over epochs throughout the training process of the RMSProp algorithm. The figure shows that both training and validation losses decline rapidly during the early phase, indicating the model's effective acquisition of data properties at this time. The validation loss curve exhibits significant fluctuations, particularly during the mid and late stages of training, suggesting that the model's performance on the validation data is unstable, potentially influenced by noise or by constraints on the model's generalisation capacity. Nevertheless, the training loss curve decreases smoothly, demonstrating the robust convergence of RMSProp. Moreover, the minimal disparity between the training and validation losses suggests that RMSProp effectively reconciles training and generalisation performance to a certain degree.

The evaluation data and loss curves indicate that NadamClip surpasses RMSProp in optimisation performance and stability. Its evaluation metrics demonstrate superior model fitting accuracy and enhanced error management, with its loss curves converging rapidly and smoothly. Conversely, whereas RMSProp shows robust early convergence, the validation loss curve exhibits greater fluctuations and instability, thereby limiting its efficacy in complex applications. NadamClip exhibits superior optimisation efficiency and generalisation capability.

v. Adam Algorithm

The significance of Adam in the stochastic gradient descent algorithm is most evident in its integration of the momentum technique and adaptive learning rate, hence enhancing the efficiency and stability of the optimisation process. Adam dynamically modifies the learning rate for each parameter by computing the first (mean) and second

(variance) moments of the gradient, thus eliminating the need for manual learning rate adjustments. The momentum mechanism in Adam helps the model escape local optima and accelerates convergence.

This experiment was conducted over 100 epochs to assess the advantages and disadvantages of the Adam and NadamClip optimisation techniques—other parameters were set to match the hyperparameters of the NadamClip training model. Adam was utilised for training, and the disparities in stability and ultimate performance between the two were noted. After 100 training epochs, the model's performance is assessed, yielding the findings presented in Table 5.23.

Table 5.23

Adam training model performance results

Evaluation methodology	Value
MAE	0.26517207121185304
RMSE	0.6520466855995359
R^2	0.974922341745961

Upon comparison of the aforementioned data with the experimental outcomes of NadamClip, the Mean Absolute Error (MAE) for Adam was 0.2652, the Root Mean Square Error (RMSE) was 0.6520, and the R^2 Score was 0.9749. Compared with NadamClip's data (MAE = 0.2644, RMSE = 0.6595, R^2 = 0.9743), Adam exhibits marginally superior RMSE performance. Furthermore, the R^2 Score is somewhat elevated at 0.9749, indicating a superior match relative to NadamClip's 0.9743. In summary, while the two perform closely, Adam shows a slight advantage in RMSE

and R^2; the difference is minimal, and both demonstrate good accuracy and robust fitting.

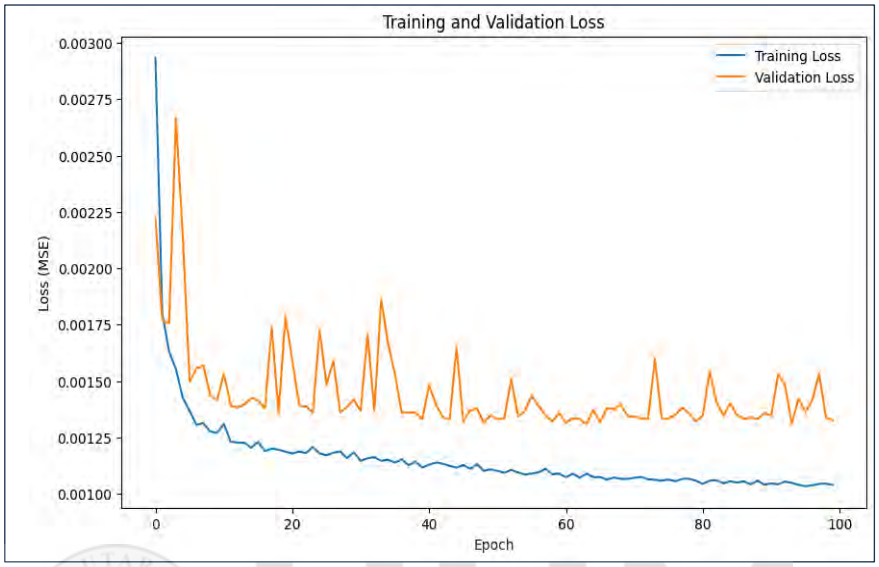

Figure 5.24. Adam of loss changes

As illustrated in figure 5.24, the Adam training loss declined rapidly during the initial phase, with a pronounced reduction; however, in the subsequent phase, it stabilised, and the rate of decline diminished progressively. The validity loss diminishes swiftly during the initial period but continues to fluctuate. When comparing Adam with NadamClip, it is seen that the NadamClip optimiser exhibits greater consistency in reducing training and validation loss, with less fluctuation in validation loss, suggesting superior generalisation capability and stability.

The performance of the Adam and NadamClip optimisers can be extensively examined by examining variations in training and validation losses, alongside comparisons of model evaluation metrics (MAE, RMSE, and R^2). The Adam optimiser exhibits marginal superiority in RMSE and R^2 metrics, while NadamClip demonstrates a

modest advantage in the MAE statistic. Nonetheless, the disparity between the two is minimal, and both exhibit high accuracy and robust fitting. The NadamClip optimiser's stationarity and robust generalisation capacity may make it more appropriate for tasks that require greater stability and lower volatility. Although Adam rapidly optimises models in the short term, it inadequately assesses loss volatility, which may limit its use in certain complex tasks. The NadamClip optimiser may excel at long-term stability and generalisation, particularly in situations that require avoiding overfitting and maintaining high stability.

vi. AdamW Algorithm

In the realm of optimisation algorithms, AdamW, an enhanced variant of Adam, is better suited to the regularisation requirements of contemporary deep learning problems by modifying how weight decay is handled. In contrast to conventional L2 regularisation, AdamW decouples weight decay from gradient updates, thereby mitigating the impact of the learning rate on regularisation strength and enhancing the model's generalisation performance. This experiment compares NadamClip with the AdamW algorithm to evaluate their performance in terms of model stability and error management.

To compare the advantages and disadvantages of AdamW and NadamClip optimisation algorithms, 100 epochs were used in this experiment, and other parameters were set to the hyperparameters of the NadamClip training model. AdamW was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, yielding results as shown in Table 5.24.

Table 5.24

AdamW training model performance results

Evaluation methodology	Value
MAE	0.28577964900318875
RMSE	0.6770786502305564
R^2	0.9729599284872327

NadamClip and AdamW exhibit strong performance across all metrics, with NadamClip demonstrating a marginal advantage. The MAE of NadamClip is 0.2643, marginally lower than AdamW's 0.2858, suggesting superior error management; regarding RMSE, NadamClip's 0.6595 outperforms AdamW's 0.6771; and in terms of the R^2 score, NadamClip's 0.9743 exceeds AdamW's 0.9730, indicating enhanced model performance. AdamW has an R^2 of 0.6771, while NadamClip achieves 0.9743, marginally surpassing AdamW's 0.9730, indicating superior model fitting. The results indicate that while AdamW is comparable to NadamClip on assessment metrics, the latter performs better in minimising errors and achieving higher fitting accuracy.

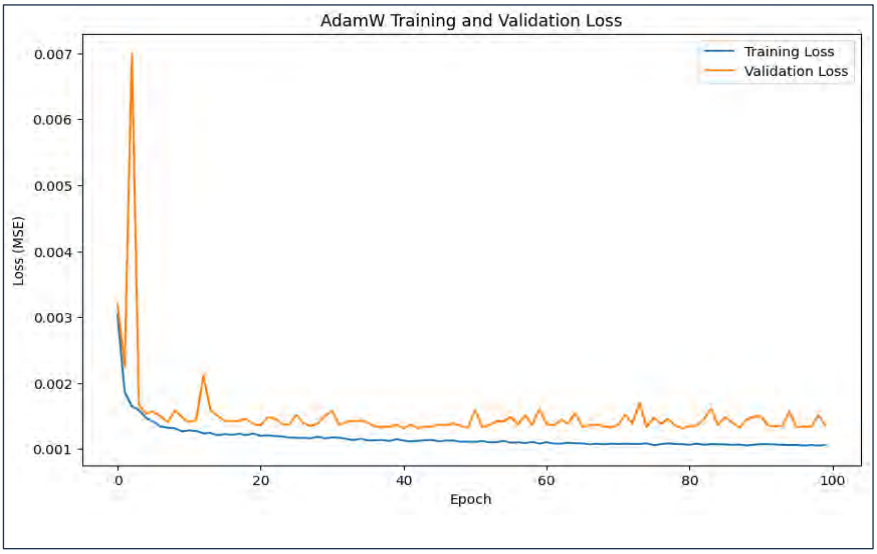

Figure 5.25. AdamW of loss changes

Figure 5.25 illustrates the trends in training and validation loss for the AdamW algorithm throughout training. The training loss exhibits a rapid decline and smooth convergence, indicating that AdamW effectively optimises the model parameters. Nonetheless, the validation loss exhibits considerable variability throughout both the initial and final phases of training.

According to the evaluation data and loss curves, NadamClip marginally surpasses AdamW in optimisation performance and convergence stability. The assessment metrics for NadamClip indicate superior model fit and reduced error, with loss curves converging rapidly and smoothly. The validation loss aligns with the training loss, implying enhanced generalisation performance. Conversely, the validation loss of AdamW exhibits fluctuations throughout both the initial and final phases of training, and its convergence is marginally less stable than that of NadamClip, albeit with a lower total loss. This indicates that although AdamW performs well, NadamClip is superior in terms of stability and predictive accuracy.

vii. Nadam Algorithm

The significance of Nadam in the stochastic gradient descent algorithm is evident in its integration of the benefits of Nesterov accelerated gradient and Adam optimisation algorithms. Nadam incorporates Nesterov's acceleration concept into gradient updates, enhancing momentum, accelerating convergence, and effectively mitigating the problem of local optima. Nadam incorporates the adaptive learning rate mechanism of the Adam algorithm, enabling automatic adjustment of each parameter's learning rate based on its gradient history, hence enhancing convergence speed and training stability.

To evaluate the merits and demerits of the two optimisation methods, Nadam and NadamClip, this experiment was conducted across 100 epochs. Other settings aligned with the hyperparameters of the NadamClip training model, and Nadam was employed for training to evaluate disparities in stability and final performance between the two algorithms. After 100 training epochs, the model's performance was assessed, with the results presented in Table 5.25.

Table 5.25

Nadam training model performance results

Evaluation methodology	Value
MAE	0.2684995375430839
RMSE	0.6678549834047971
R^2	0.9736916301768074

Comparing the aforementioned data with the trial outcomes of NadamClip reveals distinct performance discrepancies. The current model has a Mean Absolute Error (MAE) of 0.2685, a Root Mean Square Error (RMSE) of 0.6679, and an R^2 value of 0.9737. Compared with NadamClip (MAE = 0.2644, RMSE = 0.6595, R^2 = 0.9743), the present model shows somewhat lower performance, particularly in RMSE (0.6679). NadamClip equals 0.6595. Nevertheless, the R^2 values of the two models are comparable, with the current model achieving 0.9737, slightly lower than NadamClip's 0.9743.

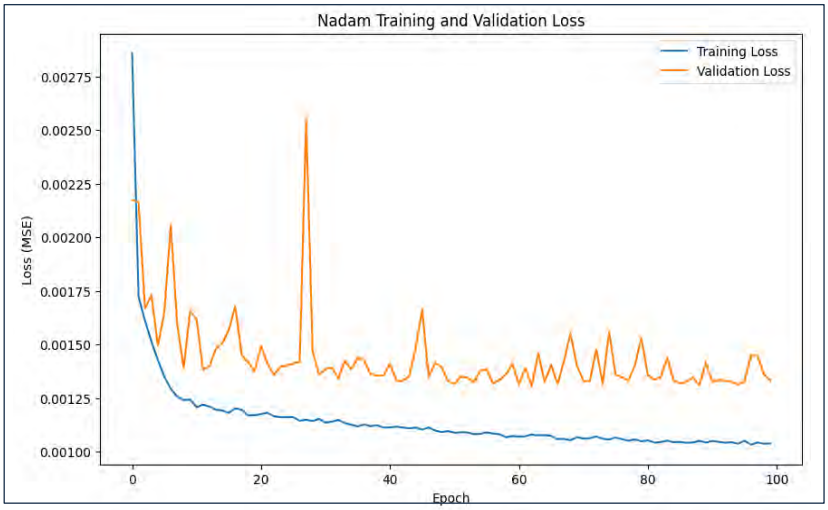

Figure 5.26. Nadam of loss changes

As shown in Figure 5.26, the Nadam training loss decreases faster in the initial phase, but the loss curve fluctuates significantly as training progresses. In addition, the validation loss showed large fluctuations during training, especially in the later stages; it tended to increase, further indicating the possibility of overfitting and the model's weak generalisation ability.

In summary, although Nadam has certain advantages in the initial training process, especially in the case of rapid loss decline, the later training fluctuations and the upward trend of validation loss reveal its deficiency in generalisation ability. NadamClip, on the other hand, showed a more stable optimisation process in the training process, avoided the overfitting phenomenon, and maintained a better generalisation ability. In terms of performance indicators, NadamClip consistently outperforms Nadam, demonstrating superior stability and generalisation, particularly in practical time-series forecasting tasks such as ammonia concentration prediction in aquaculture systems.

5.5.1.2 Comparison with Gradient Clipping Stochastic Gradient Descent

This section of the study presents a comprehensive performance comparison of the NadamClip algorithm, using various representative gradient-clipping stochastic gradient descent algorithms as benchmarks. The comparison algorithms include SGD with U-Clip, which integrates gradient clipping techniques into SGD (Elesedy & Hutter, 2023); DPSGD, which incorporates SGD (M. Abadi et al., 2016); CGAdam, which Utilises Adam (H. Sun, Yu et al., 2024); ClipRAdaGradD, which employs AdaGrad (Chezhegov et al., 2024); and AdamW_AGC, which incorporates AdamW (H. Sun, Cui, et al., 2024). These algorithms include many foundational optimisation techniques, such as gradient clipping, to provide enhanced versions for diverse situations. Comparing NadamClip with these algorithms enables us to assess its performance regarding convergence, stability, and applicability from several viewpoints, hence further validating its potential and relevance in deep learning.

i. SGD with U-Clip Algorithm

SGD with U-Clip is an improved version of the classical SGD algorithm that combines gradient clipping to suppress gradient spikes that may occur during training.

To compare the advantages and disadvantages of SGD with U-clip and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. SGD with U-clip was used for training, and differences in stability and final performance were observed. After training iterations with epoch=100, the performance of the model is evaluated, and the results are obtained as in Table 5.26:

Table 5.26

SGD with U-Clip training model performance results

Evaluation methodology	Value
MAE	0.8108290857223689
RMSE	1.2470619331350854
R^2	0.9082712149511512

When comparing the performance of SGD with the U-Clip and NadamClip optimisation algorithms, significant differences emerge across various indicators. First, the NadamClip optimisation algorithm performed better across all key metrics. Specifically, the absolute MAE of NadamClip is 0.2644, which is significantly lower than that of SGD with U-Clip (0.8108), indicating that NadamClip has greatly improved prediction accuracy. The RMSE was also superior to the NadamClip, with the NadamClip at 0.6595 versus the SGD with U-Clip at 1.247, indicating a smaller margin of error for the NadamClip. In terms of R^2, NadamClip reached 0.9743, which is much higher than SGD with U-Clip's 0.9083, indicating that NadamClip provides a better fit and better captures the data trend.

In general, the NadamClip optimisation algorithm performs better than SGD with U-Clip in terms of training performance and model stability, especially in improving prediction accuracy and fitting ability. NadamClip is better suited to tasks that require high precision and strong generalisation.

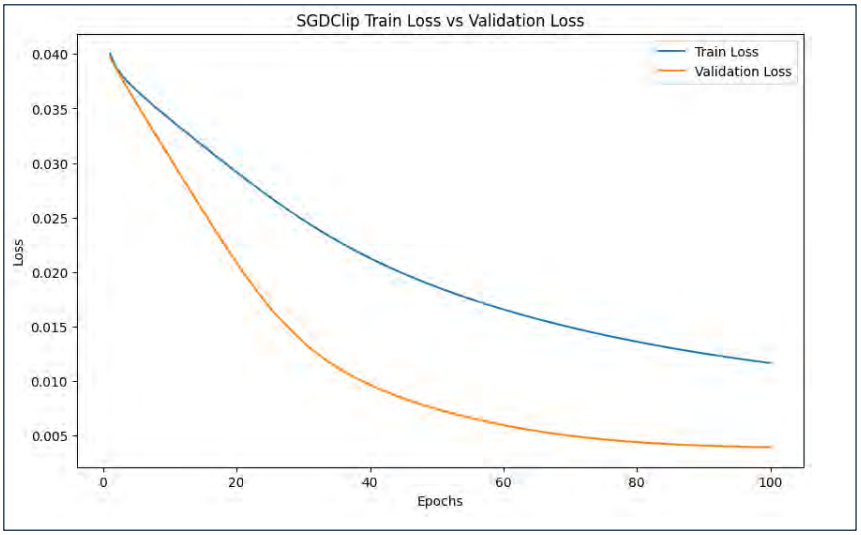

Figure 5.27. SGD with U-Clip of loss Changes

From Figure 5.27, the SGD with U-Clip loss curve shows a steady decline, indicating that the model is relatively stable during training and that both the training and validation losses are gradually decreasing. This shows that the model fits well to both the training and validation data, and that the training process is smooth. However, as training deepened, the model's performance did not continue to improve significantly, and the verification loss remained at a relatively stable low level, indicating that the training process was close to convergence but could not be further optimised.

Based on the above analysis, the loss curve of SGD with U-Clip decreases steadily, and the training process is relatively stable. However, at the end of the training, its final loss value is high, and it cannot be further optimised, resulting in weak generalisation. Despite better initial performance, with lower training and validation losses, SGD with U-Clip failed to achieve higher accuracy or better fitting capabilities. In contrast, the training process of NadamClip fluctuated, but as training deepened, the loss value gradually converged, and the final training and validation losses were lower than those of SGD with U-Clip, indicating that NadamClip can effectively optimise

and improve the model's performance in the later stages of training. Especially in key indicators such as MAE, RMSE, and R^2, NadamClip has shown clear advantages, demonstrating better prediction accuracy, fitting ability, and data trend capture.

Therefore, considering the final model performance, NadamClip is significantly better than SGD with U-Clip, especially in optimisation ability and late-stage model accuracy. NadamClip offers stronger advantages and is better suited to tasks requiring high accuracy and generalisation.

ii. DPSGD

DPSGD is an optimisation algorithm that integrates privacy preservation with gradient-based optimisation, ensuring data confidentiality through the incorporation of noise and gradient clipping during gradient updates. This experiment compares the NadamClip algorithm with the DPSGD algorithm.

To compare the advantages and disadvantages of the DPSGD and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. DPSGD was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, yielding results as shown in Table 5.27.

Table 5.27

DPSGD training model performance results

Evaluation methodology	Value
MAE	1.2594584818344572
RMSE	0.8108990966161156
R^2	0.9064384716813607

NadamClip demonstrably surpasses DPSGD in all criteria. Its MAE is considerably lower than that of DPSGD, suggesting it is superior in error management. Its RMSE value is considerably lower than that of DPSGD, signifying superior optimisation of total error. The R^2 ratings reveal that NadamClip's model fitting capability much surpasses that of DPSGD. Despite DPSGD's distinctive advantage in privacy protection via gradient clipping, there is a considerable disparity between its optimisation performance and model correctness compared to NadamClip.

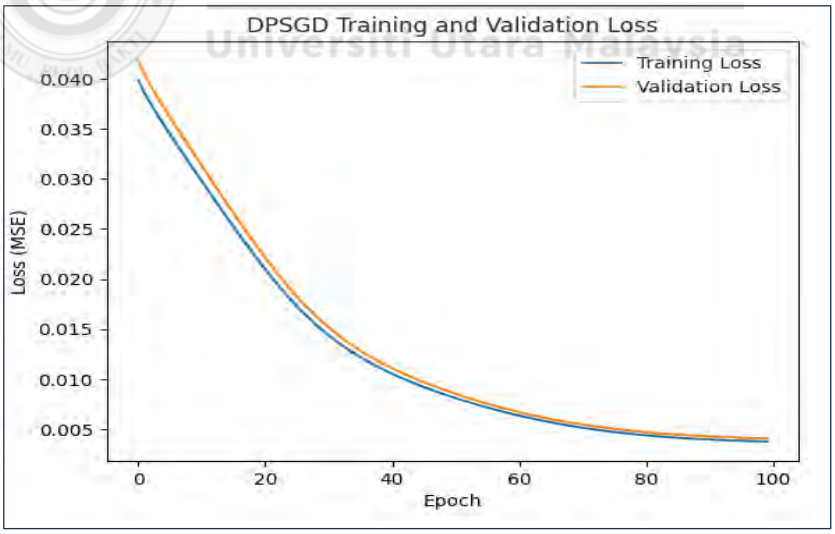

Figure 5.28. DPSGD of loss changes

Figure 5.28 illustrates a more gradual decline in the loss. This may be attributable to the introduction of gradient clipping and noise in the gradient update by DPSGD,

which may constrain the model's capacity for rapid optimisation while safeguarding privacy.

Evaluation data and loss curves indicate that NadamClip far surpasses DPSGD in optimisation performance and model fitting capability. The assessment metrics of NadamClip indicate reduced error and enhanced fitting accuracy, with its loss curves decreasing more rapidly and converging more smoothly, demonstrating superior optimisation efficiency and generalisation capability. The convergence of DPSGD is comparatively sluggish. This suggests that while DPSGD possesses distinct benefits in privacy protection, its optimisation capacity and error management are inferior to those of NadamClip.

iii. Clip-RAdaGradD

Clip-RAdaGradD is an optimisation technique that uses gradient clipping to enhance the robustness and convergence of AdaGrad in the presence of gradient noise. This experiment compares Clip-RAdaGradD with NadamClip to evaluate the effect of Clip-RAdaGradD on model performance and prediction accuracy.

To compare the advantages and disadvantages of Clip-RAdaGradD and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. Clip-RAdaGradD was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, yielding results as shown in Table 5.28.

Table 5.28

Clip-RAdaGradD training model performance results

Evaluation methodology	Value
MAE	0.6614861471415543
RMSE	1.07210966116476
R^2	0.9322033842309119

NadamClip exhibits superior optimisation performance compared to Clip-RAdaGradD, as evidenced by its numerical metrics: the Mean Absolute Error (MAE) for NadamClip is 0.2644, markedly lower than Clip-RAdaGradD's 0.6615, signifying enhanced error control; the Root Mean Square Error (RMSE) for NadamClip is 0.6595, substantially better than Clip-RAdaGradD's 1.0721, indicating superior overall error optimisation; furthermore, NadamClip's R^2 score of 0.9743 surpasses Clip-RAdaGradD's 0.9322, reflecting its enhanced model fitting capability. The results demonstrate that while Clip-RAdaGradD improves its resilience to gradient noise by gradient clipping, it remains less effective than NadamClip in terms of accuracy and optimisation efficiency.

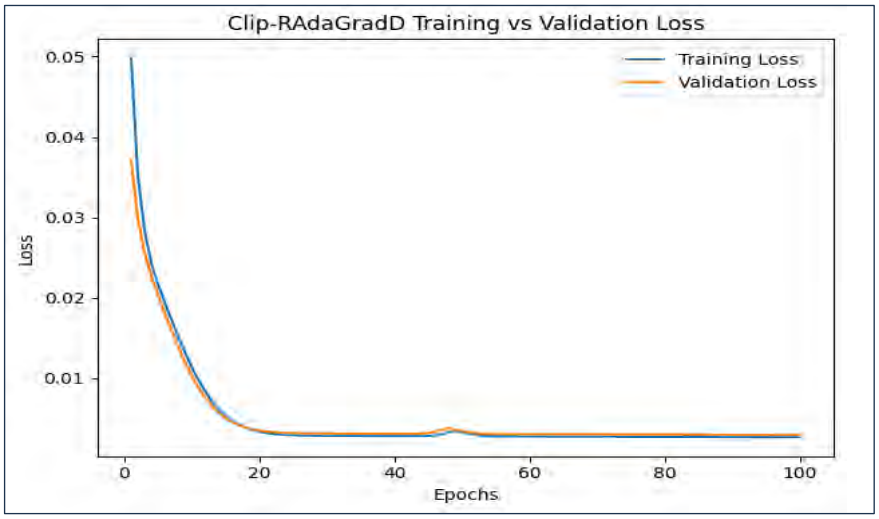

Figure 5.29. Clip-RAdaGradD of loss Changes

Figure 5.29 shows a rapid decline in both training and validation losses during the initial phase, indicating the model's efficient learning of data features. After approximately 20 iterations, the loss values stabilise, indicating that the model has effectively converged. Moreover, the training and validation loss curves nearly coincide, indicating that the model exhibits commendable generalisation performance throughout the training phase, with no evident overfitting or underfitting.

According to the evaluation data and loss curves, NadamClip clearly outperforms Clip-RAdaGradD across all metrics. The evaluation data indicate that NadamClip exhibits superior error control and model fitting, whereas Clip-RAdaGradD exhibits higher error rates and poorer fitting. The loss curves indicate that the training and validation losses of Clip-RAdaGradD decline swiftly and ultimately stabilise, with the two curves closely aligned, demonstrating superior convergence and generalisation capacity. Nonetheless, compared to NadamClip, its ultimate loss is higher, and the reduction in validation loss is less significant. This indicates that whereas Clip-RAdaGradD excels in stability and consistency, NadamClip demonstrates a more pronounced advantage in optimisation efficacy and model performance.

iv. CGAdam

CGAdam is an enhanced optimisation algorithm derived from the Adam algorithm. It is designed to address issues of local optima, overfitting successfully, and sluggish convergence by incorporating a control-restart approach and a joint shear method for the gradient method. The restart method enables the algorithm to escape local optima. At the same time, the joint shear technique mitigates the impact of aberrant gradients on the stability and efficiency of model optimisation by imposing appropriate

constraints on the gradients. This experiment compares NadamClip with CGAdam to evaluate the optimisation method's impact on model performance and correctness.

To compare the advantages and disadvantages of the CGAdam and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. CGAdam was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, yielding results as shown in Table 5.29.

Table 5.29

CGAdam training model performance results

Evaluation methodology	Value
MAE	0.26186467218314385
RMSE	0.6581578685817608
R^2	0.9744500677986624

The numerical performance of CGAdam is comparable to that of NadamClip, exhibiting a marginal superiority; specifically, the Mean Absolute Error (MAE) for CGAdam is 0.2619, whereas NadamClip's is 0.2644, indicating a slight advantage in error management. In terms of Root Mean Square Error (RMSE), CGAdam registers at 0.6582, slightly lower than NadamClip's 0.6595, further demonstrating enhanced overall error optimisation. Additionally, the R^2 score for CGAdam is 0.9745, marginally exceeding NadamClip's 0.9743, suggesting a slight improvement in model fit. These results demonstrate that CGAdam achieves more sophisticated optimisation through enhanced technique.

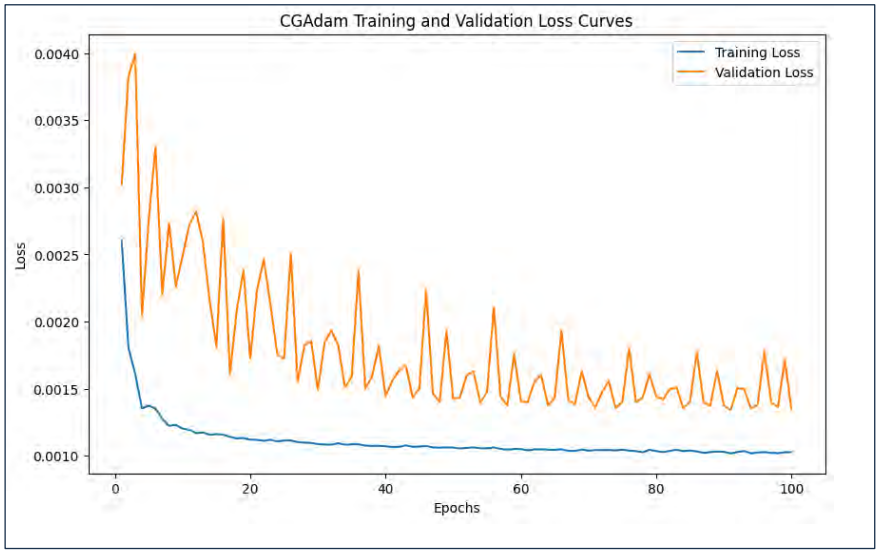

Figure 5.30. CGAdam of loss changes

As shown in figure 5.30, the training loss decreases rapidly and then flattens in the early stage, indicating that CGAdam can optimise the model quickly and converge stably. However, the validation loss curve shows significant fluctuations, especially in the early and middle phases, suggesting instability despite the overall trend of a slight decrease.

The evaluation data and loss curves indicate that both CGAdam and NadamClip have superior optimisation skills, albeit with distinct emphases. The evaluation data indicate that CGAdam marginally surpasses NadamClip in error management and model fitting, exhibiting superior accuracy. NadamClip exhibits a more gradual reduction in the training and validation loss curves, along with a distinct trend in validation losses, indicating enhanced stability and superior generalisation capability. The validation loss curve of CGAdam exhibits significant variations; although there is a general downward tendency, this instability may compromise the consistency of its optimisation on the validation set. NadamClip excels in stability, although CGAdam possesses a marginal edge in predictive accuracy.

v. AdamW_AGC

AdamW_AGC is an enhanced optimisation technique that integrates AdamW with adaptive gradient clipping. The algorithm improves the model's regularisation by incorporating weight decay and simultaneously integrating the AGC strategy to adaptively apply gradient clipping based on the gradient-to-parameter ratio, thereby effectively mitigating the adverse effects of gradient outliers on model training. This experiment compares NadamClip with AdamW_AGC to evaluate its impact on model performance.

To compare the advantages and disadvantages of AdamW_AGC and NadamClip optimisation algorithms, this experiment used 100 epochs, with other parameters set to the hyperparameters of the NadamClip training model. AdamW_AGC was used for training, and the differences in stability and final performance between the two were observed. After training iterations with epoch=100, the model's performance is evaluated, and the results are presented in Table 5.30.

Table 5.30

AdamW_AGC training model performance results

Evaluation methodology	Value
MAE	0.27305611051047424
RMSE	0.6829421034527225
R^2	0.9724895704203821

NadamClip and AdamW_AGC exhibit comparable optimisation performance across several metrics, with NadamClip demonstrating marginal superiority. NadamClip's MAE is 0.2644, marginally lower than AdamW_AGC's 0.2731, indicating superior

error control; its RMSE is 0.6595, also better than AdamW_AGC's 0.6829. Furthermore, NadamClip's R^2 score of 0.9743 surpasses AdamW_AGC's 0.9725, suggesting a slight advantage in model fitting. These findings indicate that while AdamW_AGC enhances performance through the integration of weight decay and adaptive gradient clipping, it remains somewhat less effective than NadamClip for error optimisation and model fitting precision.

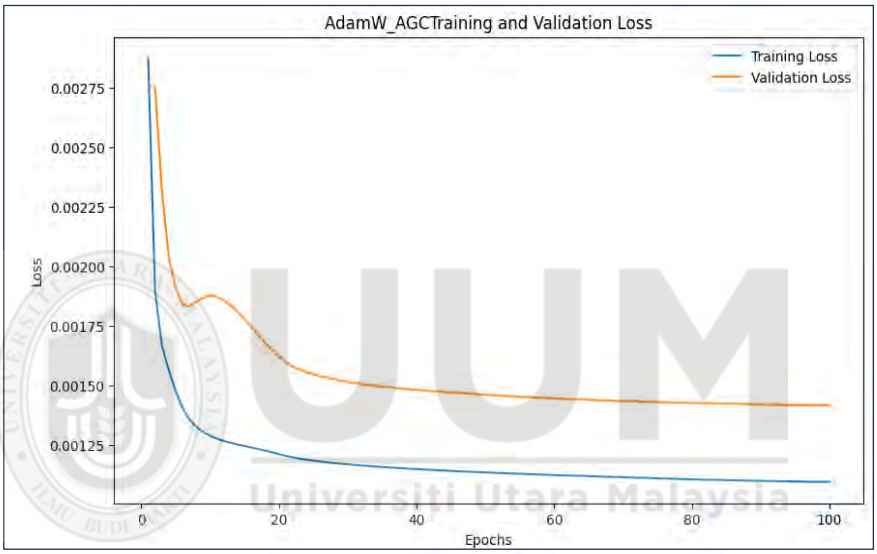

Figure 5.31. AdamW_AGC of loss changes

Figure 5.31 illustrates the trends in training and validation loss for the AdamW_AGC algorithm over the course of training. The training loss exhibits a rapid decline in the initial phase and stabilises in the later phase, indicating the swift convergence capability of AdamW_AGC.

The assessment data and loss curves indicate that NadamClip and AdamW_AGC exhibit strong yet distinct optimisation performance. The evaluation data indicate that NadamClip marginally surpasses AdamW_AGC in error management and model-fitting precision, resulting in lower error and improved fitting. The loss curves indicate

that both the training and validation losses exhibit a rapid decline, with the validation loss continuing to drop in the latter phase of training, suggesting greater optimisation potential. NadamClip exhibits superior optimisation efficacy and generalisation capacity, whereas AdamW_AGC excels at consistently reducing validation loss.

5.5.2 Analysis of Training Model Results

This segment of the experiment will involve a comparative analysis of the training performance of the NadamClip method across various models, including the LSTM, GRU, and TFT models. These three models exemplify the prevailing classical and avant-garde techniques in time series modelling and sequence learning, respectively: LSTM and GRU are prevalent versions of recurrent neural networks, offering substantial benefits in addressing long- and short-term dependency issues; TFT is a contemporary deep learning model that integrates the attention mechanism to manage intricate time series data and external inputs effectively. Through this research, we aim to thoroughly assess the applicability and properties of NadamClip across various model architectures by comparing its convergence time, training stability, and final performance, thereby providing foundational data to support further generalisation.

5.5.2.1 GRU

In our tests, to further validate the applicability of the NadamClip technique across various recurrent neural network models, we compare the training outcomes of the LSTM and GRU models. GRU, a streamlined version of LSTM, has fewer parameters and lower computational complexity while still effectively capturing analogous long- and short-term dependencies in time series data. Comparing the training performance

of NadamClip on LSTM and GRU allows for a more thorough analysis of its impact across different network architectures.

This experiment compared the application of NadamClip to LSTM and GRU over 100 epochs, with all other parameters set to the LSTM model's hyperparameters. After 100 epochs of iterative training, the model's performance is assessed, and the results are presented in Table 5.31.

Table 5.31

NadamClip training GRU model performance results

Evaluation methodology	**Value**
MAE	0.2754651315408184
RMSE	0.6737194091774394
R^2	0.9732275747561396

Numerically, employing NadamClip for LSTM training yields marginally better performance than GRU. Regarding error control, the MAE of LSTM is 0.2644, lower than GRU's 0.2755, signifying superior predictive accuracy; for RMSE, LSTM's value is 0.6595, also lower than GRU's 0.6737, demonstrating enhanced overall error management; in terms of model fitting capability, LSTM's R^2 is 0.9743, marginally exceeding GRU's 0.9732, indicating LSTM's superior capacity to capture and fit data characteristics.

In summary, whereas GRU has a streamlined structure and computational efficiency, our experiment reveals that LSTM exhibits superior error management and model fitting capabilities through optimised NadamClip, making it more suitable for tasks requiring higher precision.

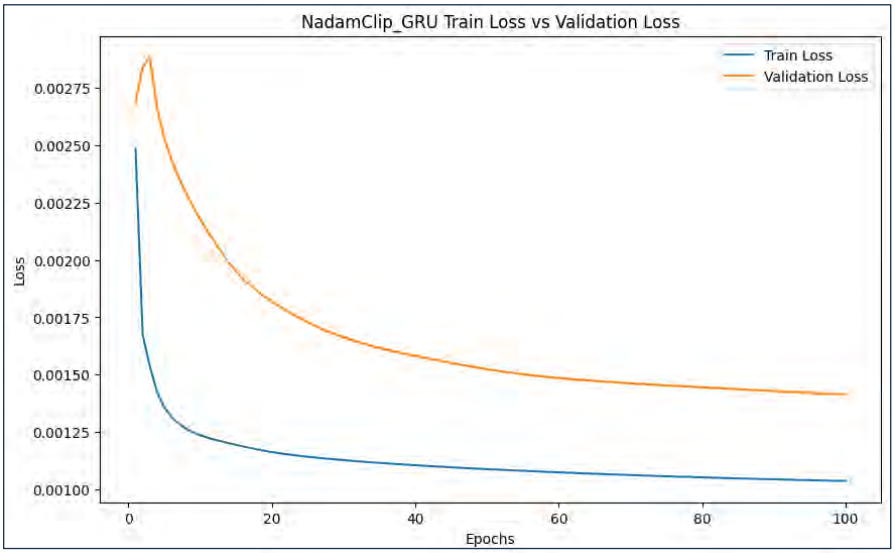

Figure 5.32. GRU of loss changes

Figure 5.32 shows that the training loss declines rapidly in the initial phase and then stabilises, indicating that the model parameters converge rapidly throughout the training process. The validation loss curve exhibits a general decline, though at a slightly slower rate than the training loss, ultimately stabilising in the later stages.

The evaluation data and loss curves indicate that the LSTM and GRU models optimised with NadamClip exhibit distinct overall performance characteristics. The evaluation data indicate that LSTM outperforms GRU somewhat in error management and model fitting, resulting in lower error and greater fitting precision. The loss curve for LSTM exhibits a more stable decline in validation loss, with a significant disparity between validation and training loss, while maintaining a consistent downward trajectory, indicating its superior capacity to optimise validation data. Conversely, the validation loss of the GRU, although slower to decline, exhibits a trajectory that aligns more closely with the training loss in the latter stages, indicating superior

generalisation capability. LSTM is preferable for applications that require greater precision, but GRU excels in model simplicity and generalisation performance.

5.5.2.2 TFT

In our tests, we assess the efficacy of the NadamClip algorithm on intricate models by applying it to the TFT model and juxtaposing it with the LSTM model. TFT is an advanced deep learning model tailored for time series prediction, including an attention mechanism and various feature fusion algorithms to capture global temporal dependencies and local feature interactions effectively. Comparing the training performance of NadamClip on LSTM and TFT enables a thorough analysis of its applicability and potential within conventional recurrent neural networks alongside contemporary sequence modelling techniques.

A basic TFT model architecture T is structured with two primary concealed layers, a multi-head attention mechanism, and an output layer. The initial hidden layer contains 144 neurons, while the subsequent hidden layer comprises 124 neurons. The multi-attention technique utilises four attention heads to augment the sequence representation by dynamically allocating weights across time steps. NadamClip optimiser is employed alongside gradient clipping, using the empirically determined optimal threshold of 0.24 for this study, to maintain stable gradient updates. A learning rate of 0.001 and momentum parameters β_1=0.9 and β_2=0.999 are used for weight adjustment.

Table 5.32

NadamClip training TFT model performance results

Evaluation methodology	Value
MAE	0.31395
RMSE	0.7623525577474964
R^2	0.9657199600182456

As shown in 5.32, the performance metrics indicate that LSTM outperforms TFT. Primarily due to the complex structure and large number of parameters in TFT, it is more prone to generalisation instability and performance degradation under current conditions of limited sample size and feature complexity. Regarding error control, the MAE of LSTM is 0.2644, lower than TFT's 0.3139, demonstrating LSTM's superior prediction accuracy; regarding overall error optimisation, the RMSE of LSTM is 0.6595, significantly lower than TFT's 0.7624, highlighting its robust performance. Furthermore, regarding model fitting, the R^2 of LSTM is 0.9743, surpassing TFT's 0.9657, indicating that LSTM captures data properties more effectively and achieves more precise fitting. LSTM surpasses TFT across all key performance metrics, demonstrating superior optimisation and predictive accuracy.

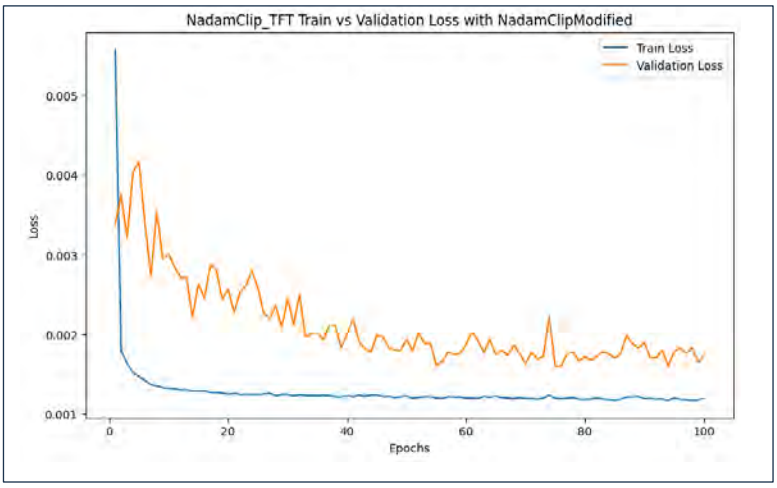

Figure 5.33. TFT of loss changes

Figure 5.33 shows that the training loss declines rapidly in the initial phases and then stabilises, indicating the model's capacity for rapid convergence on the training set. The validation loss shows significant swings, particularly during the first and intermediate phases, with pronounced variations. Despite an overall downward trend, it fails to achieve complete stability in the later stages. This behaviour may suggest that the TFT model's performance on the validation data is unstable, potentially indicating underfitting or excessive regularisation.

According to the evaluation data and loss curves, LSTM outperforms TFT across both overall performance and stability. The evaluation data indicate that LSTM exhibits superior error management and model fitting capabilities, resulting in enhanced prediction accuracy. The loss curve indicates that LSTM exhibits a more consistent decline in validation loss, demonstrating superior overall convergence, notwithstanding the disparity between validation and training loss. Conversely, the validation loss curve of TFT exhibits greater fluctuations and does not attain a stable state, despite an overall declining trend, signifying inferior generalisation capability and consistency. In summary, LSTM demonstrates greater consistency and efficiency in the present studies, but TFT may require more intricate tuning to realise its full potential.

5.5.2.3 ARIMA

To further compare model performance, a traditional statistical time-series forecasting method, namely the ARIMA model, is introduced as a benchmark. ARIMA is one of the most classical linear time series models and has been extensively applied in

economic forecasting, environmental monitoring, and signal processing due to its solid theoretical foundation and clear interpretability.

In this experiment, ARIMA is employed as a forecasting model, using only the historical ammonia concentration data for modelling and prediction. Before model training, the ammonia concentration time series is examined for stationarity through differencing. A first-order difference (d = 1) is adopted to eliminate the non-stationary trend in the original series. After multiple experimental trials, the ARIMA (2,1,2) configuration is selected for performance evaluation. The dataset is divided into a training set and a testing set using an 80%–20% split, based solely on chronological order, to ensure the temporal integrity of the forecasting process.

After training the ARIMA model using the training set, multi-step forecasting is conducted on the test set. The ARIMA model's predictive performance is quantitatively evaluated using three widely adopted regression metrics: MAE, RMSE, and R^2. The results are shown in Table 5.3.

Table 5.33

ARIMA model performance results

Evaluation methodology	**Value**
MAE	0.59777
RMSE	1.55544
R^2	-0.13455

The experimental results indicate that the ARIMA model exhibits relatively weak predictive performance on the ammonia concentration dataset. Although the MAE remains moderate, the RMSE is comparatively large, reflecting substantial deviations

between predicted and actual values during certain time intervals. More importantly, the negative R^2 value indicates that the ARIMA model performs worse than a simple baseline that uses the test set's mean for prediction. This suggests that the linear modelling capacity of ARIMA is insufficient to capture the complex temporal dynamics and nonlinear fluctuation characteristics inherent in ammonia concentration data.

The inferior performance of the ARIMA model can be primarily attributed to two fundamental limitations. First, ARIMA assumes a linear dependency structure and can only model linear temporal relationships. In contrast, ammonia concentration is strongly influenced by nonlinear interactions among environmental factors such as temperature, dissolved oxygen, and pH. Second, as a univariate model, ARIMA does not utilise any auxiliary environmental variables and therefore fails to describe the multivariate coupling mechanism governing ammonia variation in aquaculture systems.

In summary, although ARIMA serves as an important and classical time-series forecasting baseline, its predictive accuracy and fitting capability are significantly inferior to those of the proposed NadamClip-optimised LSTM model on the ammonia prediction task. This experimental comparison further validates the necessity and superiority of deep learning-based models for complex environmental time-series forecasting.

5.5.3 Comparison of Experimental Results

This section provides a detailed comparative analysis of the optimisation algorithms used in the experiment, including SGD, AdaGrad, AdaDelta, RMSProp, Adam, AdamW, Nadam, SGD with U-Clip, DPSGD, Clip-RAdaGrd, CGAdam, AdamW_AGC and the proposed NadamClip, to comprehensively evaluate the effectiveness of the proposed method. The goal is to determine which algorithm achieves the best model performance on this complex task.

Alongside the model performance metrics, assess Train Time, Current Memory Usage, and Peak Memory Usage to evaluate the computational overhead of the optimisation technique. These metrics not only indicate the algorithm's computational efficiency during training but also assess its resource utilisation. The duration of training directly indicates the algorithm's operational efficiency, whereas memory consumption reflects the model's complexity and computational demands. Comparing these computational overhead metrics enables a more intuitive assessment of the feasibility of various algorithms for practical applications and the optimisation of resource consumption, particularly in scenarios with constrained resources or stringent real-time demands.

Through these analyses, this section not only provides a theoretical basis for selecting optimisation algorithms but also helps further optimise the model's training process and improve its performance in complex tasks.

5.5.3.1 Comparison of Loss Trends

Comparing the loss trajectories of various optimisation techniques and models during training is essential for thoroughly evaluating their efficacy. Analysing the curves of

each method for training and validation loss across iterations allows visualisation of convergence speed, final convergence state, and the volatility and stability of validation loss.

i. Comparison of optimisation Algorithms

This section compares the loss trends of the NadamClip, SGD, AdaGrad, AdaDelta, RMSProp, Adam, Nadam, AdamW, DPSGD, SGD with U-Clip, Clip-RAdaGradD, CGAdam, and AdamW_AGC algorithms during training to assess their performance under varying training conditions. Comparing the training loss and validation loss of these six algorithms provides insights into the efficacy of each approach. The study will specifically examine the asymptotic variations and stability of the loss values.

First, the training loss of each algorithm is visualised with the change of time or number of iterations, as shown in the figure 5.30 below:

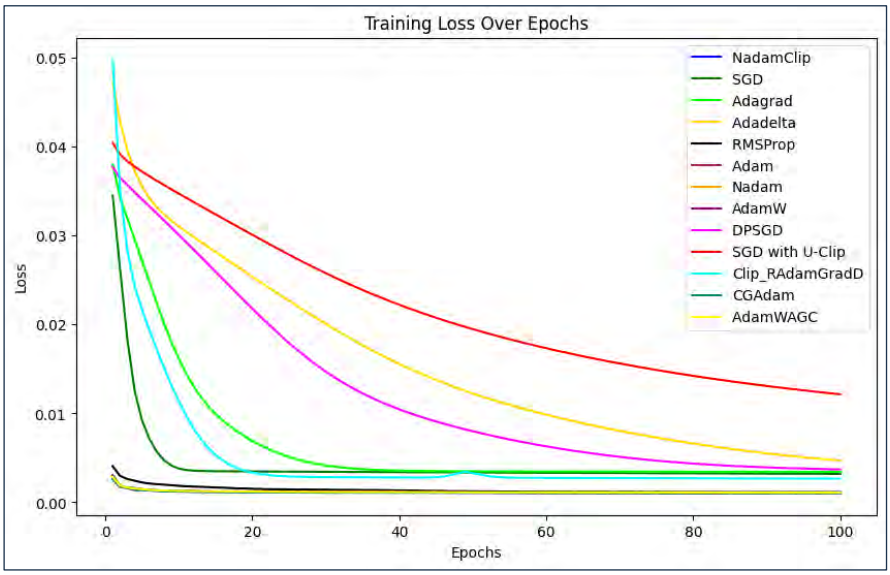

Figure 5.34. Training loss comparison

Figure 5.34 shows that all algorithms converge sufficiently within 100 epochs. The loss rates of SGD with U-Clip (red), AdaDelta (yellow), and DPSGD (pink) decrease more gradually during the initial training phase, particularly for SGD with U-Clip, which exhibits a flatter loss curve with a smaller decline. AdaGrad (lime), Clip_RAdamGradD (cyan), and SGD (green) exhibit a rapid decrease in loss during the initial phase of training; nevertheless, their final losses are still larger than that of NadamClip (blue), RMSProp (black), Adam (brown), Nadam (orange), AdamW (purple), CGAdam (teal), and AdamWAG (yellow).

Subsequently, upon comparing Nadam, RMSProp, Adam, NadamW, AdamW, CGAdam, and AdamWAG, it is observed that the loss curves of these algorithms largely converge in the latter phases of training, exhibiting similar patterns in training loss with minimal discrepancies. The overlap of the loss curves for these algorithms makes it difficult to distinguish them in the graphs, preventing analysis of the small differences between them.

To facilitate a clearer analysis of the nuanced distinctions among these algorithms, SGD with U-Clip, AdaDelta, DPSGD, AdaGrad, Clip_RAdamGradD, and SGD have been excluded. This can more clearly illustrate the variations in the loss curves of these algorithms, enabling more effective comparisons of their convergence rates and the nuanced differences in their ultimate loss values, thus aiding a deeper comprehension of the merits of these optimisation techniques.

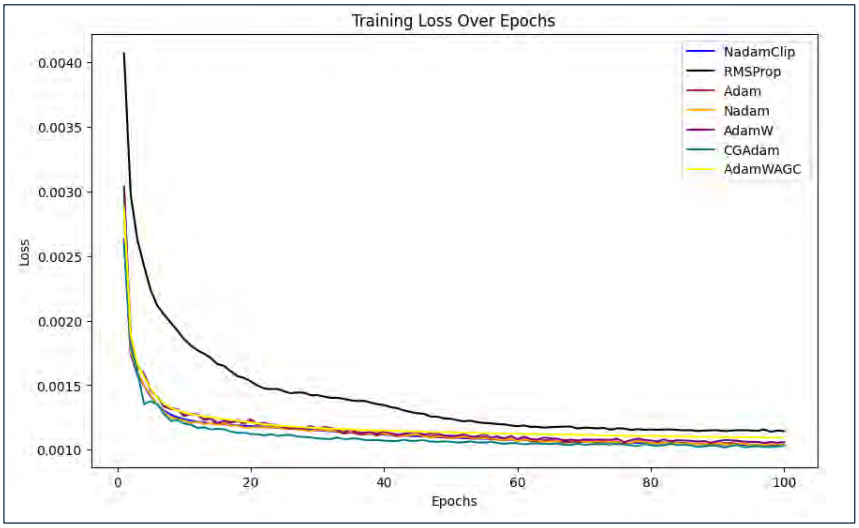

Figure 5.35. Comparison of training losses for NadamClip, RMSProp, Adam, Nadam, AdamW, CGAdam, AdamW_AGC

As shown in Figure 5.35, upon closer examination and comparison, it was observed that the RMSProp loss diminishes at a markedly slower rate than that of the other algorithms, resulting in a final loss that exceeds those of the alternatives; hence, RMSProp was excluded from further comparisons.

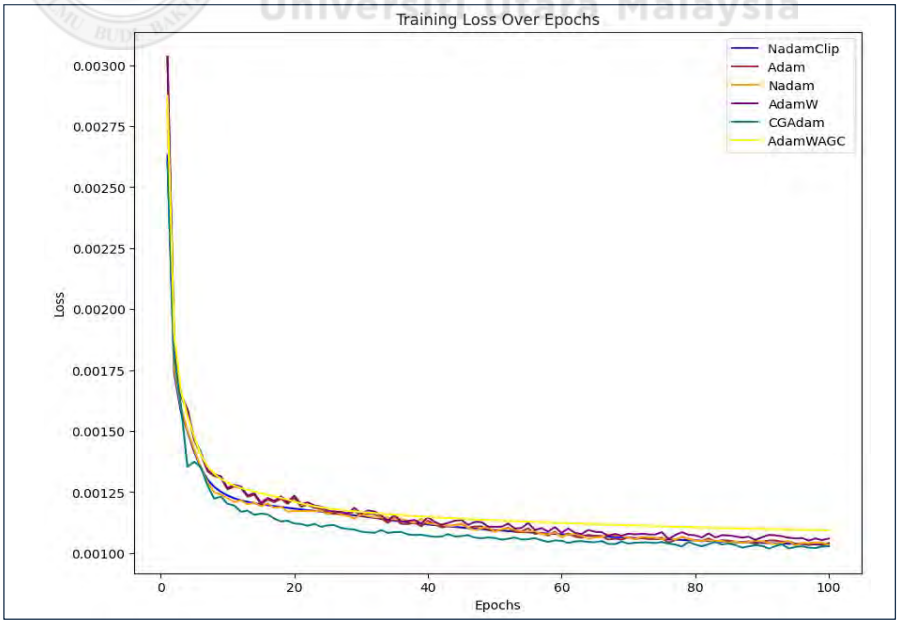

Figure 5.36. Comparison of training losses for NadamClip, Adam, Nadam, AdamW, CGAdam, AdamW_AGC

Figure 5.36 illustrates that the loss curves of Nadam, Adam, CGAdam, and AdamW exhibit a certain degree of oscillation throughout the training phase. The oscillation phenomenon typically indicates the model's sensitivity to gradient variations during optimisation, perhaps resulting in recurrent adjustments in specific areas and an inability to converge stably to the global optimal solution. NadamClip and AdamW_AGC exhibit relative stability, while Nadam and Adam have been excluded to facilitate a clearer comparison of the performance between NadamClip, AdamW_AGC, CGAdam, and AdamW, as illustrated in Figure 5.

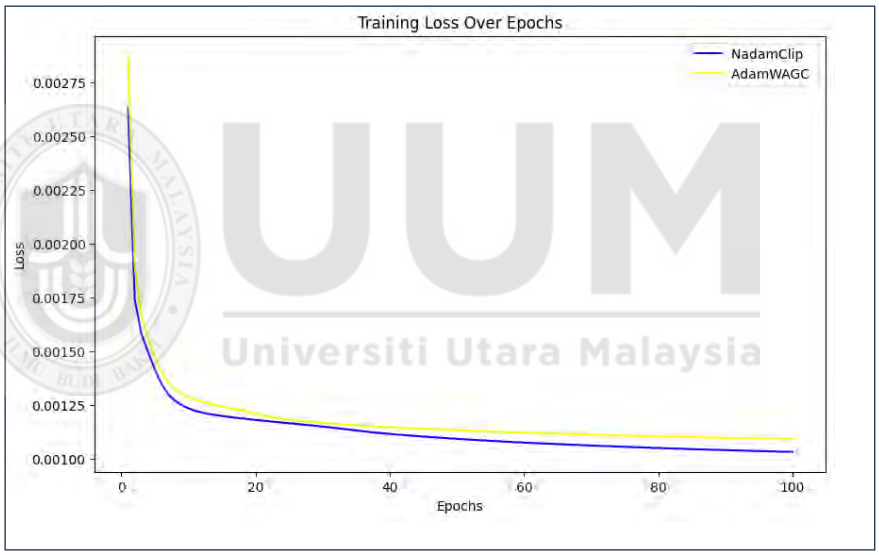

Figure 5.37. Comparison of training losses for NadamClip, AdamW_AGC

Figure 5.37 shows that the loss curves of both optimisation techniques decline swiftly in the initial phase, indicating a faster convergence rate. As training progresses, the curves of both models stabilise, indicating improved convergence in the later stages.

The NadamClip training loss is marginally lower than that of AdamW_AGC throughout training, particularly in the later stages, where the disparity becomes more pronounced. This indicates that NadamClip has superior optimisation efficiency and

final-loss management, demonstrating an enhanced capacity to adapt to the training data and identify optimal solutions more effectively. In conclusion, while both exhibit commendable optimisation performance, NadamClip demonstrates less loss in the latter stages of training and marginally superior convergence compared to AdamW_AGC. Therefore, in this comparison, NadamClip outperforms and enhances the model's stability during training. The subsequent comparison confirms the alteration in the loss curve, as illustrated in Figure 5.38.

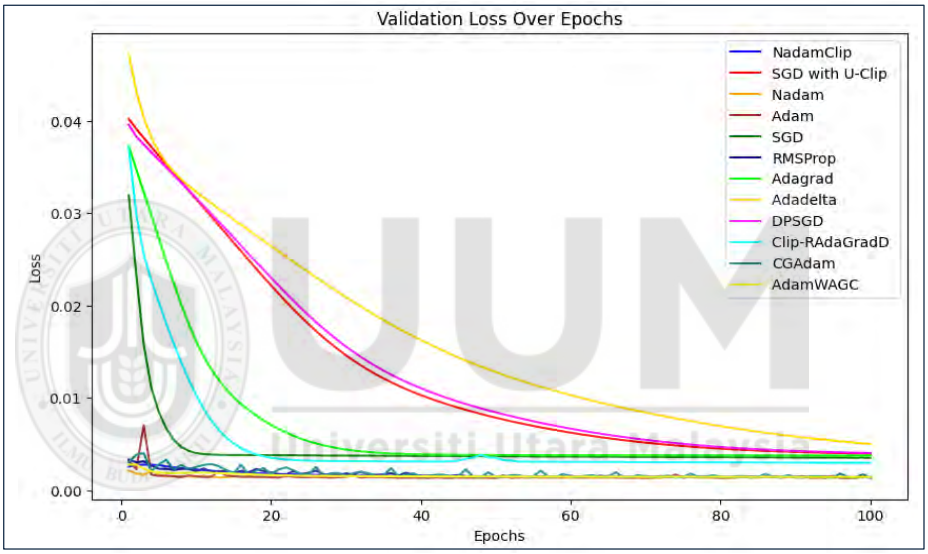

Figure 5.38. Comparison of validation losses

NadamClip, RMSProp, Adam, Nadam, AdamW, CGAdam, and AdamWAG achieve better results than SGD with U-Clip, AdaDelta, DPSGD, AdaGrad, Clip_RAdamGradD, and SGD, indicating ongoing overlap in their efficacy.

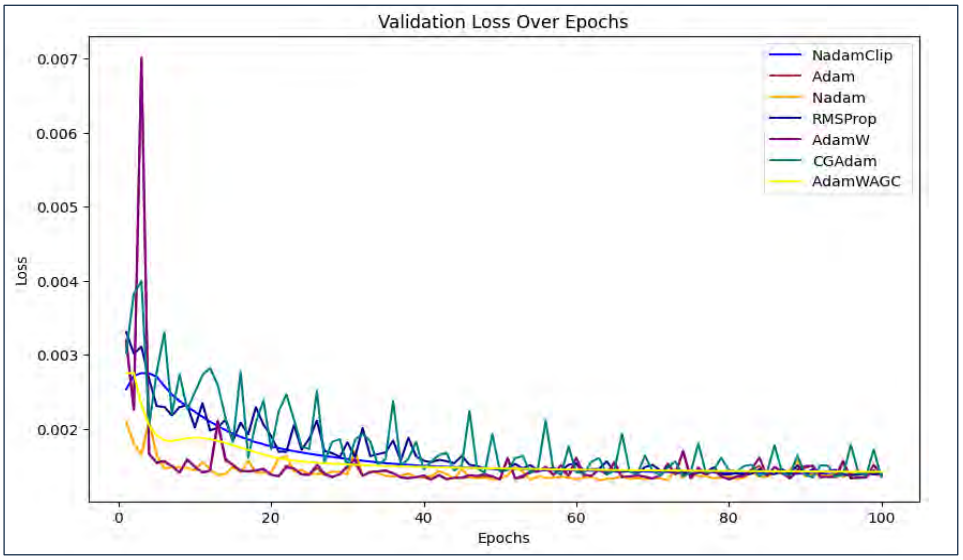

Figure 5.39. Comparison of validation losses for NadamClip, RMSProp, Adam, Nadam, AdamW, CGAdam, AdamW_AGC

Figure 5.39 shows that the validation loss parallels the training loss, and the loss curves for Nadam, Adam, CGAdam, and AdamW exhibit some oscillation during training. Eliminating them enables a clearer comparison between NadamClip and AdamW_AGC.

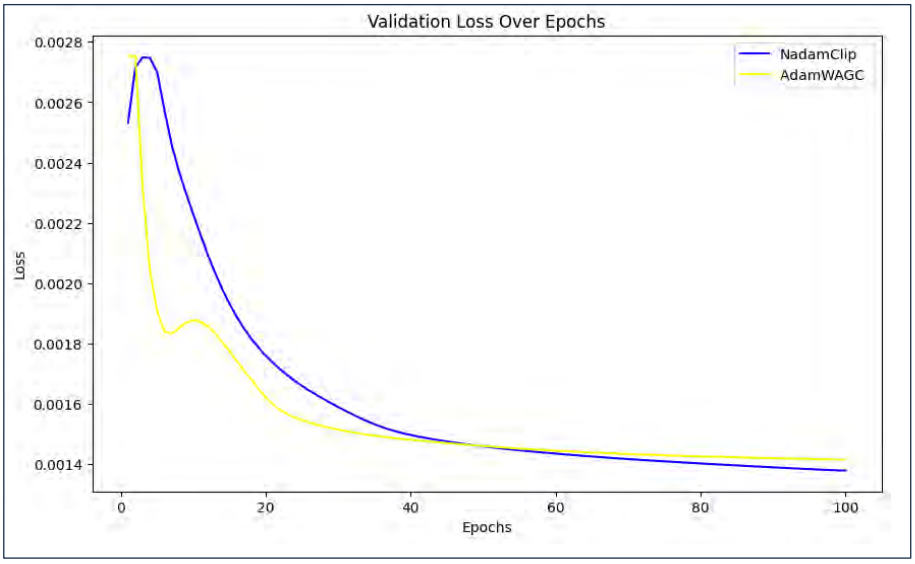

Figure 5.40. Comparison of validation losses for NadamClip, AdamW_AGC

Figure 5.40 compares the final validation loss between the two algorithms. It shows that the final validation loss of AdamW_AGC and NadamClip at the end of training is nearly identical, with NadamClip marginally lower.

The data indicates that AdamW_AGC exhibits rapid optimisation in the initial phases, whereas NadamClip demonstrates a more consistent reduction in validation loss during the subsequent stages. This may suggest that NadamClip possesses a marginal advantage in long-term generalisation during training, while AdamW_AGC is more effective at attaining lower loss in a shorter timeframe.

Both algorithms exhibit excellent performance, evidenced by minimal final validation losses. AdamW_AGC is preferable for rapid optimisation, whereas NadamClip is advantageous for achieving stable, long-term performance.

A thorough comparison of the training loss and validation loss across all algorithms reveals that NadamClip and AdamW_AGC demonstrate superior optimisation efficiency and generalisation capability. They exhibit rapid convergence rates, minimal final loss values, and stable validation curves, indicating exceptional optimisation performance and generalisation ability. Conversely, conventional algorithms like SGD and AdaGrad, while exhibiting steady performance, are inefficient optimisers and suitable for straightforward problems; whereas techniques such as DPSGD and Clip_RAdaGradD exhibit greater variability during training, indicating instability on intricate tasks. NadamClip and AdamW_AGC exhibit substantial benefits on intricate tasks, achieving rapid convergence and exceptional final performance, making them the optimal choices for overall efficacy.

ii. Comparison of Model

To comprehensively assess the optimisation algorithm's efficacy across various model architectures, the training and validation performance of the NadamClip optimisation method is compared among three models: LSTM, GRU, and TFT, as illustrated in Fig. 5.

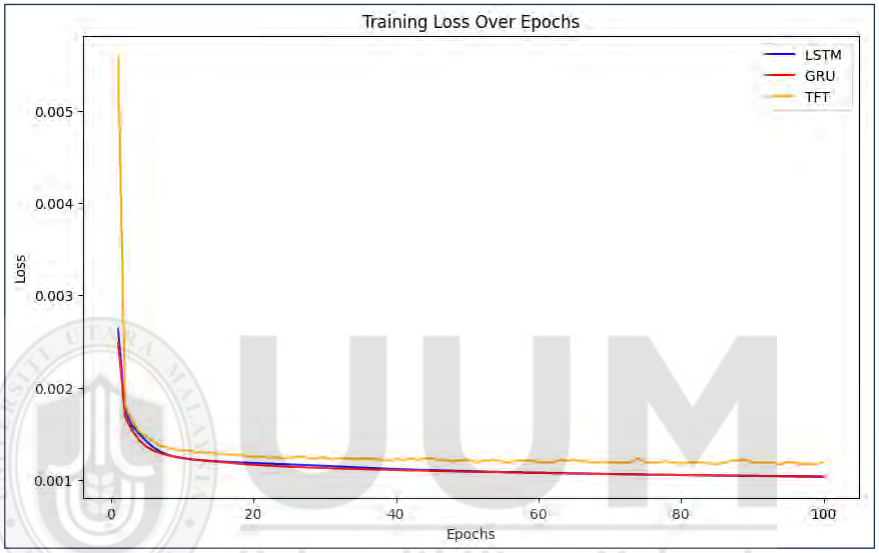

Figure 5.41. Comparison of training loss for LSTM, GRU, TFT

Figure 5.41 shows that the training loss curves for LSTM (blue) and GRU (red) exhibit a similar trend throughout training: a rapid decline in the initial phase, followed by gradual stabilisation, ultimately resulting in lower loss values. This indicates that both models converge more swiftly and effectively during training. The training loss curves of LSTM and GRU nearly coincide, attributable to their distinct gating mechanisms being closely aligned, particularly in numerous real-world applications where both GRU and LSTM demonstrate comparable learning and optimisation capabilities. Conversely, the loss of TFT (orange) is markedly greater than that of LSTM and GRU in the early phase, but it progressively diminishes and stabilises as training advances.

Nonetheless, the ultimate training loss of TFT is somewhat more than that of LSTM and GRU, suggesting a somewhat less efficient optimisation of the training dataset. This may pertain to the need for models to be built with intricate architectures and to their sensitivity to hyperparameters. LSTM and GRU demonstrate superior performance in terms of training loss, making them optimal choices for training efficiency and effectiveness.

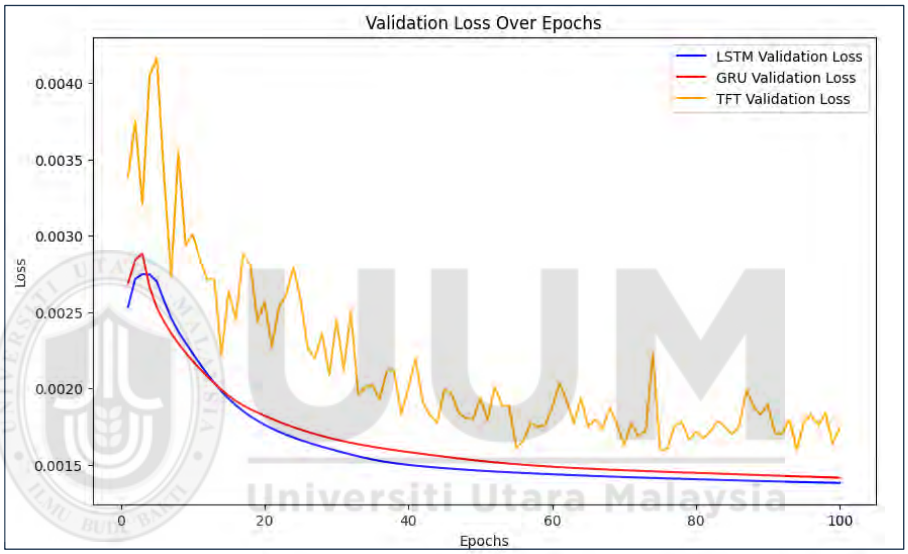

Figure 5.42. Comparison of validation loss for LSTM, GRU, TFT

As shown in Figure 5.42, in terms of the overall performance of validation loss, LSTM and GRU perform more stably, and LSTM has a lower final validation loss, which indicates that they are able to better avoid overfitting during the training process and achieve better generalisation performance on the validation set. On the other hand, TFT not only fluctuates more on the validation loss but also has a higher final validation loss value, which may require further optimisation of the hyperparameters and model structure to improve its validation set performance.

A thorough comparison of the training and validation losses reveals that LSTM exhibits superior overall performance, particularly because its final validation loss is markedly lower than those of GRU and TFT. Additionally, the smoothness of the curves indicates their enhanced generalisation capability and stability. Although GRU has a training loss comparable to LSTM's, its validation loss is somewhat larger, indicating a slight deficiency in generalisation capability relative to LSTM. Yet, it demonstrates superior efficiency and faster convergence. Conversely, the training loss of TFT diminishes more rapidly; nevertheless, the validation loss exhibits greater fluctuations and culminates at a higher final value, suggesting that its performance on the validation set lacks stability and its generalisation capacity is comparatively inadequate. LSTM is preferable for scenarios that require enhanced model stability and generalisation performance; GRU is ideal for tasks that demand greater efficiency and computational resources; while TFT is better suited for intricate temporal tasks, albeit requiring optimisation with more complex hyper-parameters and model tuning to improve validation performance.

5.5.3.2 Comparison of Prediction Accuracy

In model evaluation, prediction accuracy is one of the key metrics for measuring model performance. Common evaluation criteria include RMSE, MAE, and R^2. These indices reflect the model's prediction error and fitting effect from different angles. RMSE and MAE measure the difference between the predicted and true values, with RMSE emphasising large errors and MAE giving equal weight to all errors. R^2 reflects the model's ability to explain the variation in the data, and the closer its value is to 1, the better the model fits.

i. Comparison of optimisation Algorithms

To comprehensively assess the prediction accuracy of different optimisation algorithms, 13 optimisation algorithms were used, as shown in Table 5.34. Comparing the performance of these algorithms in terms of RMSE, MAE, and R^2 can provide a deeper understanding of their adaptability and performance differences across tasks and a more scientific basis for model selection.

Table 5.34

Summary of 13 Algorithmic measurements

Algorithm	**RMSE**	**MAE**	**R^2**
NadamClip	0.2644	0.6595	0.9743
SGD	0.7506	1.1712	0.9191
AdaGrad	1.2047	0.7821	0.9144
AdaDetla	0.9211	1.4274	0.8798
RMSProp	0.3221	0.6785	0.9729
Adam	0.2652	0.6520	0.9749
Nadam	0.2685	0.6679	0.9737
AdamW	0.2858	0.6771	0.9730
SGD with U-Clip	0.8108	1.2470	0.9083
DPSGD	1.2594	0.8108	0.9064
Clip-RAdaGradD	0.6615	1.0721	0.9322
CGAdam	0.2619	0.6582	0.9745
AdamW_AGC	0.2731	0.6829	0.9725

Among the evaluated algorithms, CGAdam, NadamClip, and Adam demonstrate superior performance. Within this top tier, CGAdam achieves the lowest RMSE (0.2619), marginally outperforming both NadamClip (0.2644) and Adam (0.2652), establishing it as the overall leader in this metric. Its MAE (0.6582) also ranks competitively, second only to Adam's optimal value (0.6520), while its R^2 (0.9745)

indicates an effective balance between precision and model fit. Adam distinguishes itself with the best MAE (0.6520) and the highest R^2 (0.9749), reflecting exceptional stability. Although NadamClip slightly trails CGAdam and Adam across metrics, it remains highly robust with near-optimal performance in all three criteria.

In contrast, traditional or privacy-enhanced optimisers, including SGD, AdaDelta, SGD with U-Clip, and DPSGD, exhibit markedly inferior results. DPSGD yields the highest RMSE (1.2594), a substantially elevated MAE (0.8108), and the lowest R^2 (0.9064), demonstrating that its privacy guarantees come at the cost of substantial model accuracy. AdaDelta performs particularly poorly in MAE (1.4274) and RMSE (0.9211), indicating fundamentally inadequate fitting capability for this task.

Mid-tier performers comprise RMSProp, Nadam, AdamW, and AdamW_AGC. RMSProp achieves an RMSE of 0.3221, slightly higher than Adam-type optimisers but still surpassing traditional SGD, while its R^2 (0.9729) confirms strong fitting performance. AdamW_AGC delivers balanced, though unexceptional, metrics across all criteria, outperforming algorithms like AdaGrad and Clip-RAdaGradD in stability, though it falls short of Adam and CGAdam.

CGAdam, NadamClip, and Adam emerge as the most effective optimisers for this regression task, making them ideal for modelling scenarios that strictly require high precision. Conversely, traditional and privacy-preserving algorithms incur significant accuracy trade-offs.

ii. Comparison of Model

Table 5.35

Summary of model measurements results

Model	RMSE	MAE	R^2
LSTM	0.2644	0.6595	0.9743
GRU	0.2755	0.6737	0.9732
TFT	0.3139	0.7623	0.9657
ARIMA	1.5554	0.5978	−0.1346

From table 5.35, it can be seen that the overall performance of LSTM is better than that of GRU and TFT across all metrics. LSTM has the lowest RMSE and MAE, 0.2644 and 0.6595, respectively, and the highest R^2 value, 0.9743, indicating that LSTM dominates in both error control and model fitting. GRU is slightly inferior to LSTM. However, its performance is still close, with RMSE of 0.2755, MAE of 0.6737, and R^2 of 0.9732, indicating better optimisation and lower error, making it suitable for tasks with limited resources or high requirements for model efficiency. In contrast, the RMSE and MAE of TFT are significantly higher than those of LSTM and GRU, and the R^2 is the lowest at 0.9657, indicating a certain disadvantage in error control and fitting ability, which may be related to its complex structure and sensitivity to hyperparameters. Compared with the above deep learning models, the ARIMA model shows significantly inferior predictive performance. Its RMSE and MAE are much higher, reaching 1.5554 and 0.5978, respectively, and its R^2 value is −0.1346, which is far lower than those of LSTM, GRU, and TFT. The negative R^2 further indicates that the ARIMA model fails to capture the variation pattern of ammonia concentration effectively and even performs worse than the mean-based baseline model. This highlights the clear limitations of traditional linear statistical models when dealing

with complex nonlinear time series data. Overall, LSTM is optimal in terms of performance and is suitable for tasks requiring high accuracy and stability, while GRU follows closely and is more suitable for scenarios in which a balance between efficiency and performance is sought; TFT is suitable for dealing with complex timing tasks, but further optimisation is needed to improve performance.

5.5.4 Analysis of Consolidated Results

In the experimental comparisons in this study, there are differences in the performance of different optimisation algorithms in the time-series prediction task. Overall, CGAdam, NadamClip, and Adam demonstrate strong performance across several evaluation metrics, achieving high regression accuracy. CGAdam shows a slight advantage in RMSE, while Adam performs slightly better on MAE and R^2. While NadamClip slightly underperforms CGAdam and Adam on individual metrics, it performs very close to both methods on all three core evaluation criteria.

In terms of the dynamic performance during the training process, the loss curves of optimisers such as Nadam, Adam, CGAdam, and AdamW showed a certain degree of oscillation during the training process, indicating that they are relatively sensitive to gradient changes and have problems such as frequent local adjustments and difficulty in stable convergence. Relatively speaking, NadamClip and AdamW_AGC show a more stable trend of loss variation. Especially in the later stage of training, the training curves of both are more stable, indicating stronger convergence stability.

Further comparison and verification of the loss trend showed that AdamW_AGC decreased rapidly in the early stage of training, demonstrating strong initial

optimisation ability. Still, the decline was gentler in the middle and later stages. However, NadamClip shows a more sustained and stable downward trend. Especially in the later stages of training, its validation loss decreases further and eventually becomes slightly lower than AdamW_AGC. This performance indicates that NadamClip has a stronger optimisation ability and better generalisation performance during long-term training.

In summary, NadamClip shows a better performance on the error assessment metrics, and is only slightly below CGAdam and Adam on some metrics. However, the overall trend of training and validation losses shows more stable, persistent optimisation performance. Therefore, in this comparative experiment, NadamClip has the best comprehensive performance and is the most stable and efficient optimisation algorithm, suitable for high-quality training tasks of complex models.

Collectively, CGAdam is the preferred optimiser for intricate, high-precision jobs, as it achieves superior efficiency and optimal predictive performance in such endeavours. NadamClip is better suited to job circumstances that require a high level of consistency and reliability, given its steady verification loss profile. Both algorithms are suitable options for contemporary optimisation projects, and the selection should depend on the job's needs to balance accuracy and stability.

In the studies, LSTM exhibits efficient performance, achieving rapid convergence in training and validation loss, yielding low final loss values, superior prediction accuracy (RMSE and MAE), and excellent stability, albeit at a marginally increased computational cost. GRU exhibits marginally lower loss convergence speed and accuracy compared to LSTM. However, it is more efficient, with the shortest training

duration and lowest memory usage, making it appropriate for resource-limited environments. Conversely, the training and validation loss curves of TFT exhibit greater fluctuations, particularly during the intermediate and later phases. However, the ultimate loss is lower, resulting in performance marginally inferior to that of LSTM and GRU. TFT is better suited to intricate operations that demand stringent timing relationships due to its increased computational overhead.

LSTM is often preferred for high-precision applications; GRU offers a compromise between performance and efficiency; and TFT is appropriate for particularly difficult situations. Model selection must include stability, precision, and efficiency based on task needs.

5.6 Applications in other fields

To demonstrate the widespread use of the NadamClip algorithm, it was trained using another set of Google stock prediction datasets from Kaggle. The dataset has 14 columns and 1257 rows, with the adjusted high price, adjusted low price, adjusted open price, and adjusted trading volume designated as input features, while the adjusted close price is the target variable for prediction. The NadamClip optimisation approach is employed to train the LSTM model while maintaining consistent hyperparameter settings with the ammonia prediction trials, thereby further substantiating the usefulness and resilience of NadamClip for complex time-series tasks across several domains.

Table 5.36

Stock Predict performance results

Evaluation methodology	Value
MAE	24.85343
RMSE	35.72394
R^2	0.99032

This dataset contains a target variable with a range of 668.2 to 2521.6 and a mean value of 1216.32. Table 5.36 above indicates that the model exhibits a Mean Absolute Error (MAE) of 24.85 and a Root Mean Square Error (RMSE) of 35.72, corresponding to 2.04% and 2.94% of the mean and 1.34% and 1.93% of the range of the target variable values, respectively. The error proportions are minimal, suggesting that the model's prediction error is nearly insignificant compared to the scale of the target variable. Furthermore, the model has an R^2 value of 0.9903, indicating that it accounts for approximately 99% of the variation in the target variable. Collectively, these measurements indicate that the model's predictive accuracy is exceptionally high and effectively captures stock price trends, providing reliable support for financial time-series forecasting.

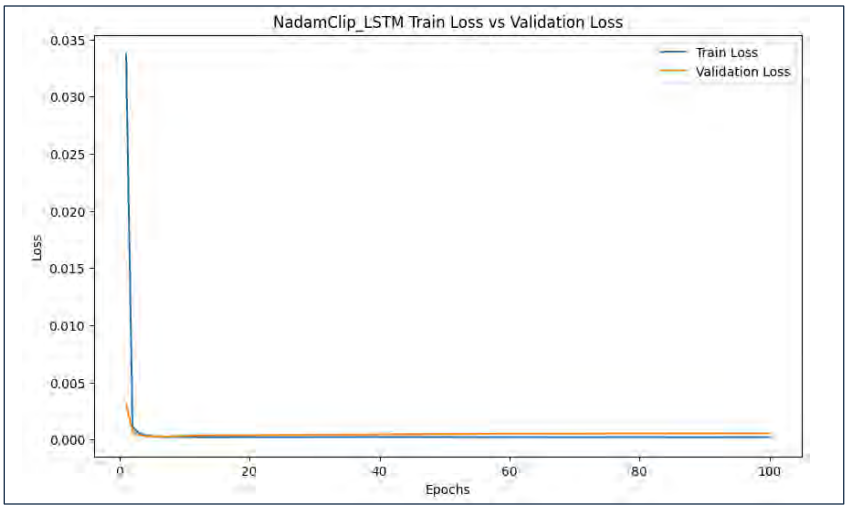

Figure 5.43. Stock predict of loss changes

As shown in Figure 5.43, the training loss and validation loss decline swiftly during the training process, achieving a stable condition within the initial epochs, ultimately plateauing and maintaining low levels. Simultaneously, the training loss and validation loss curves nearly coincide, suggesting that the model's training process has strong generalisation capabilities, with no evident signs of overfitting or underfitting.

The smoothness of the curves indicates that NadamClip's optimisation trajectory in this task is highly stable, confirming its efficacy in improving the model's stability and convergence efficiency for complex time-series prediction. Ultimately, the convergence of minimal-loss values indicates that the model effectively captures the relationship between input features and target variables, providing a basis for high-precision financial data forecasting.

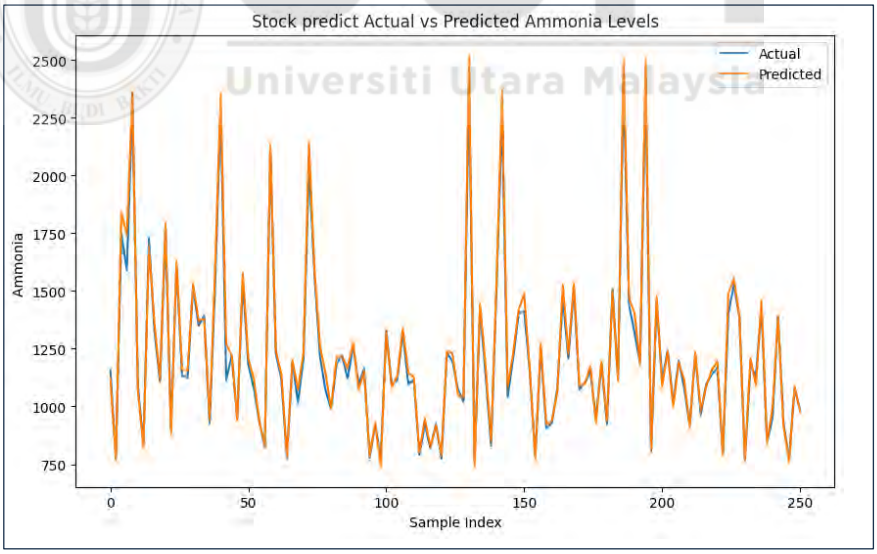

Figure 5.44. Stock forecasts true value vs predicted value

Figure 5.44 shows a strong correlation between the anticipated (orange) and actual (blue) values, with a general trend. This suggests that the LSTM model optimised by the NadamClip algorithm effectively captures the evolving patterns of the time series

of adjusted closing prices. The model effectively tracks the peaks and troughs of the actual data, particularly in volatile areas, with predicted values closely aligning with the true values, demonstrating its efficacy in predicting complex financial data.

Experimental findings indicate that NadamClip excels in both environmental data (e.g., ammonia prediction) and financial forecasting (e.g., Google stock price prediction). NadamClip demonstrates high accuracy, stability, and robust generalisation across training and validation losses and prediction results. The extensive applicability demonstrates the versatility of the NadamClip optimisation method in managing intricate time-series tasks, adapting to data characteristics across many domains.

5.7 Summary

In this section, an LSTM model was first trained using the NadamClip optimisation algorithm to predict ammonia values in aquaculture successfully, and the optimal gradient threshold was experimentally optimised to improve model performance. Subsequently, a comprehensive comparative analysis of 13 optimisation algorithms and 3 different models (LSTM, GRU, TFT) was conducted to evaluate their strengths, weaknesses and applicability scenarios in terms of training loss, validation loss, model performance and computational overhead, combined with the specific numerical values and curve trends, which revealed the performance of the algorithms in terms of improving the stability of the model and the prediction accuracy. Meanwhile, the suitability of different models in complex tasks is analysed, providing a detailed guide for optimisation algorithms and model selection. Finally, by applying NadamClip to stock price prediction in the financial domain, its wide applicability across diverse tasks is further verified.

CHAPTER SIX
CONCLUSION AND FUTURE WORK

NadamClip is an improved variant of the Nadam algorithm that aims to accelerate convergence and improve prediction accuracy during deep learning model training by introducing gradient clipping. Its main purpose is to address gradient spike and instability issues in deep learning training by combining gradient clipping with the traditional Nadam optimiser, thereby improving the model's stability and final performance. From the perspective of optimisation dynamics, NadamClip constrains the magnitude of stochastic gradients within a bounded range, which effectively reshapes the loss landscape traversal by preventing excessively large update steps that could cause oscillation or divergence. To verify the effectiveness of NadamClip, the research focuses on the following three core problems: First, how to improve the Nadam algorithm to improve the convergence speed and stability of the model; Secondly, how to find the gradient clipping threshold range suitable for NadamClip algorithm to optimise the gradient update in the training process? Finally, how to evaluate the performance of NadamClip optimisation algorithm in practical tasks, especially in LSTM model training performance.

The first goal of the research is to enhance the NadamClip algorithm by combining the Nadam optimiser with gradient clipping to improve the model's training efficiency and accuracy. This improvement is not only reflected in empirical performance gains but also in enhanced optimisation stability under non-stationary, highly non-convex loss surfaces typical of recurrent neural networks. The second goal is to explore different gradient clipping thresholds through experiments, determine the optimal clipping

range, and further optimise the algorithm's training performance. The third objective is to verify its practical advantages by comparing it with other optimisation algorithms.

6.1 Contributes

This research has made several important contributions to deep learning optimisation, especially in addressing the gradient spike problem during LSTM training. With the widespread application of deep learning to complex tasks such as time series prediction, improving the stability of training and prediction accuracy has become an urgent challenge. In this study, an innovative optimisation algorithm, NadamClip, is proposed that combines the Nadam optimiser with U-Clip gradient clipping to address gradient spikes and significantly improve model performance effectively. The next section details the main theoretical and practical contributions of this research. It illustrates the breakthroughs made by the NadamClip algorithm in improving the training efficiency, stability and accuracy of LSTM models.

6.1.1 Theoretical Contribution

The main theoretical contribution of this study is the proposal of a new optimisation algorithm, NadamClip, to address the gradient spike problem during deep learning model training, especially in the context of LSTM applications. By combining the Nadam optimiser and U-Clip gradient clipping technology, the algorithm provides a new solution that significantly improves the stability and prediction accuracy during training. More importantly, from a theoretical optimisation standpoint, NadamClip modifies the effective update rule of Nadam by introducing an implicit upper bound on gradient-induced parameter displacement, thus improving the robustness of the

optimisation trajectory in regions where the loss surface exhibits sharp curvature. Specific contributions include the following aspects:

1. Proposal and innovation of NadamClip algorithm:

In this study, the NadamClip algorithm is proposed for the first time and combined with the Nadam optimiser and U-Clip gradient shear technique to optimise the gradient spike problem in LSTM models. Compared with existing optimiser enhancements, NadamClip pioneers the structural integration of gradient clipping with adaptive momentum estimation, overcoming the conventional paradigm in which clipping primarily served as external training control rather than an intrinsic algorithmic component. Mechanistically, this integration allows the adaptive first- and second-order moment estimates of Nadam to operate under a controlled gradient regime, thereby reducing the amplification of heavy-tailed gradient noise and improving convergence behaviour in deep recurrent structures. This innovation introduces a new approach to deep learning, especially suitable for high-dimensional, complex time-series prediction tasks, such as ammonia concentration prediction in aquaculture.

2. optimisation of gradient clipping range

To further improve the model's performance, this study explored different gradient clipping thresholds through experiments and analysed the effects of different clipping ranges on training. By precisely adjusting the gradient clipping threshold, it is found that a suitable clipping range can effectively improve the LSTM model's prediction accuracy, especially for water quality data with high fluctuations. From the perspective of loss landscape dynamics, an excessively large threshold fails to suppress gradient

explosion, while an overly small threshold may lead to gradient vanishing or slow convergence. The experimental results reveal a clear trade-off between stability and learning capacity, suggesting an optimal clipping interval that balances exploration and stability in parameter space. This not only optimises the effect of gradient clipping but also further demonstrates its adaptability and importance in deep learning optimisation.

3. Experimental analysis and performance comparison:

In this study, the effectiveness of NadamClip was verified through a large number of experiments, and it was compared in detail with SGD, Adam, and Nadam, as well as their variants combined with gradient clipping, such as AdamClip and SGD with U-Clip. The experimental results show that NadamClip is superior to other optimisation algorithms in terms of accuracy and stability. Especially when dealing with long-term and high-dimensional data, NadamClip shows significant advantages, with lower training error and higher prediction accuracy. These performance gains can be attributed to the synergistic interaction of bounded gradient updates, which reduces oscillatory behaviour near sharp minima and improves generalisation by avoiding unstable parameter jumps.

4. Long-term impact on the field/discipline and application prospects:

The innovative algorithm in this study not only solves the gradient spike problem but also offers new ideas for deep learning, particularly in time series prediction and complex data modelling. The design and application of the NadamClip algorithm are expected to advance research on deep learning optimisers, especially for tasks that

require efficient, stable training, thereby significantly improving model performance and reliability. In the future, NadamClip can be widely used in environmental monitoring, financial forecasting, intelligent manufacturing, climate forecasting, and other fields to provide more accurate and stable model support for intelligent decision-making.

6.1.2 Practical Contribution

By limiting the gradient norm, NadamClip prevents gradient spikes and model divergence, optimises training, and adapts to the complexity and volatility of water quality data. The practical contributions of this study are described in detail below, illustrating the potential and impact of the NadamClip optimisation algorithm in aquaculture.

1. Practical application of the algorithm:

The NadamClip optimisation algorithm proposed in this study has broad application potential, especially in aquaculture. Ammonia-level management in aquaculture systems is very important, and this study effectively improves the prediction accuracy and training stability of ammonia concentration by combining an LSTM model with the NadamClip optimisation algorithm. This method can not only improve the accuracy of ammonia concentration prediction but also yield reliable predictions in dynamic water quality environments, thereby providing effective water quality management support for the aquaculture industry. By introducing gradient clipping, the NadamClip optimisation algorithm avoids gradient spikes during training, helps ensure the stability and efficiency of the LSTM model in long-term time series data

processing, and provides a more efficient decision-support tool for the aquaculture industry.

2. Practical impact in specific areas:

The method of this study has significant practical implications in the field of aquaculture. By improving the accuracy of ammonia concentration predictions, the NadamClip algorithm helps aquaculture better manage water quality and avoid the negative effects of ammonia fluctuations on the health and growth of aquatic organisms. This can not only improve the stability of the farming environment but also improve the productivity and quality of aquatic products, ensuring the sustainability and profitability of the farming process. In addition, the time series prediction method combined with LSTM in this study can provide similar predictive support for other water quality parameters (such as dissolved oxygen, temperature, pH, etc.), further enhancing the level of intelligent management of aquaculture.

3. Future application prospects:

The NadamClip optimisation algorithm proposed in this study has broad application prospects, especially in agricultural production, environmental monitoring, food safety and other fields. In the future, with further optimisation and improvement of the algorithm, NadamClip can be widely used in more fields involving dynamic data prediction, such as agricultural irrigation management, intelligent greenhouse control, and environmental pollution monitoring. The advantage of this method is that it can process complex time-series data, provide more accurate and efficient predictions, and help decision-makers extract valuable information from large amounts of

environmental data to support sustainable development in agriculture and the environment.

This study not only proposed the innovative NadamClip optimisation algorithm, which demonstrated strong performance in deep learning, especially for predicting time series data, but also demonstrated its wide application prospects in aquaculture. In the future, as the method is applied across different fields, NadamClip is expected to become an important tool to improve prediction accuracy, stabilise the training process, optimise decision-making, and promote the intelligent management and sustainable development of aquaculture and related industries.

Overall, this research's contribution is of great significance for both theory and practice. In theory, the NadamClip algorithm provides a new optimisation approach and promotes the development of deep learning optimisation techniques. In practice, the algorithm provides a more accurate and stable ammonia concentration prediction tool for aquaculture, enhancing water quality management and promoting the sustainable development of the aquaculture industry. Therefore, this study not only enriches the theoretical research of optimisation algorithms but also provides feasible technical solutions for practical problems in related industries.

6.2 Limitation

Although the NadamClip optimisation algorithm proposed in this study has achieved significant performance improvements in LSTM model training, it still has some limitations. First, the selection of the gradient clipping threshold is based on manual testing results. Although different threshold ranges are explored through experiments

and the clipping effect is optimised, this method still has certain limitations. Specifically, under extreme non-stationary data distributions or abrupt regime shifts, a fixed clipping threshold may either fail to suppress sudden gradient spikes or overly restrict necessary large updates, leading to suboptimal convergence. In practical applications, different data sets and tasks may require different gradient clipping thresholds, and manually selecting thresholds cannot be applied to all cases. Therefore, adaptive gradient clipping will be an important research direction in the future. By dynamically adjusting the clipping threshold, the algorithm can automatically adapt to gradient changes during training, thereby further improving the stability of training and enhancing the universality and automation of the algorithm.

6.3 Future Work

Although this study has achieved some success in optimising the Nadam optimiser and proposing the NadamClip algorithm, there are still many directions for further study and improvement. Future research can be expanded and deepened from the following aspects.

First of all, selecting a gradient clipping threshold is a key step in this study, but the threshold is set through manual experiments and lacks self-adaptation. In the future, the method of adaptive gradient clipping can be explored, and the dynamic changes in the gradient during training can be used to automatically adjust the clipping threshold, thereby achieving a more efficient training process. This can not only improve the algorithm's stability but also increase the model's universality across tasks and datasets, making gradient clipping technology more flexible and automated.

Second, although the NadamClip algorithm solves the gradient spike problem during training by introducing gradient clipping, excessive clipping can sometimes limit the model's learning of complex patterns. Therefore, a hybrid clipping strategy can be explored in the future, combining different clipping methods to balance the effects of gradient clipping on training stability and model learning ability. This hybrid strategy may include layer-wise clipping, norm-based adaptive clipping, or trust-region-inspired update constraints.

Third, while this study focuses on the LSTM model, the potential of the NadamClip algorithm does not stop there. Future research could explore applying this algorithm to other neural network models, such as CNNs and transformers, especially for complex time-series prediction tasks. Cross-domain applications may reveal the performance and advantages of the optimisation algorithm in different tasks. This will help verify whether the stabilising mechanism of NadamClip generalises across different loss geometries and optimisation regimes. In summary, future research can explore the automation of the gradient clipping strategy and the algorithm's cross-domain application to better meet the challenges in practical applications and promote further development of deep learning optimisation algorithms.

REFERENCES

Abadi, J., & Brunnermeier, M. (2018). Blockchain Economics. In *NBER Working Paper*. http: //doi.org/10.3386/w25407

Abadi, M., McMahan, H. B., Chu, A., Mironov, I., Zhang, L., Goodfellow, I., & Talwar, K. (2016). Deep learning with differential privacy. *Proceedings of the ACM Conference on Computer and Communications Security*, *24-28-Octo*, 308–318. https://doi.org/10.1145/2976749.2978318

Abbasimehr, H., & Paki, R. (2021). Prediction of COVID-19 confirmed cases combining deep learning methods and Bayesian optimisation. *Chaos, Solitons and Fractals*, *142*, 110511. https://doi.org/10.1016/j.chaos.2020.110511

Agarwal, M., Gill, K. S., Malhotra, S., & Devliyali, S. (2024). Exploring Numerical Analysis with Sequential Convolutional Neural Networks Leveraging Adam optimisation Technique. *2024 International Conference on Innovations and Challenges in Emerging Technologies, ICICET 2024*, 1–5. https://doi.org/10.1109/ICICET59348.2024.10616317

Agarwal, S., Dandge, S. S., & Shankar Chakraborty, P. (2020). A support vector machine-based prediction model for electrochemical machining process. *Karbala International Journal of Modern Science*, *6*(2), 8. https://doi.org/10.33640/2405-609X.1508

Ahmad, A. L., Chin, J. Y., Mohd Harun, M. H. Z., & Low, S. C. (2022). Environmental impacts and imperative technologies towards sustainable treatment of aquaculture wastewater: A review. *Journal of Water Process Engineering*, *46*(December 2021), 102553. https://doi.org/10.1016/j.jwpe.2021.102553

Ahmed, N., Thompson, S., & Turchini, G. M. (2020). Organic aquaculture productivity, environmental sustainability, and food security: insights from organic agriculture. *Food Security*, *12*(6), 1253–1267. https://doi.org/10.1007/s12571-020-01090-3

Airlangga, G., Nugroho, O. I. A., & Sugianto, L. F. (2025). Deep Learning Approaches for Water Quality Prediction in Aquaponics Systems: A Comparative Study of Recurrent and Feedforward Architectures. *Buletin Ilmiah Sarjana Teknik Elektro*, *7*(1), 1-8. https://doi.org/10.12928/biste.v7i1.12411

AKGÜN, D. (2021). a Tensorflow Based Method for Local Derivative Pattern. *Mugla Journal of Science and Technology*, *7*(1), 59–64. https://doi.org/10.22531/muglajsci.830691

Alabdulrazzaq, H., Alenezi, M. N., Rawajfih, Y., Alghannam, B. A., Al-Hassan, A. A., & Al-Anzi, F. S. (2021). On the accuracy of ARIMA based prediction of COVID-19 spread. *Results in Physics*, *27*, 104509. https://doi.org/10.1016/j.rinp.2021.104509

Alexei Botchkarev. (2022). Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. *ArXiv Preprint ArXiv*, *16*(1), 1–23. https://doi.org/https://doi.org/10.28945/4184

Althobaiti, M. M., Pradeep Mohan Kumar, K., Gupta, D., Kumar, S., & Mansour, R. F. (2021). An intelligent cognitive computing based intrusion detection for industrial cyber-physical systems. *Measurement: Journal of the International Measurement Confederation*, *186*(September), 110145. https://doi.org/10.1016/j.measurement.2021.110145

Alum, E. U., Uti, D. E., Agah, V. M., Orji, O. U., Ezeani, N. N., Ugwu, O. P., Bawa, I., Omang, W. A., & Itodo, M. O. (2023). Physico-chemical and Bacteriological Analysis of Water used for Drinking and other Domestic www.jchr.org Purposes in Amaozara Ozizza , Afikpo. *Nigerian Journal of Biochemistry and Molecular Biology*, *38*(1), 1–8.

Arepalli, P. G., Akula, M., Kalli, R. S., Kolli, A., Popuri, V. P., & Chalichama, S. (2022). Water Quality Prediction for Salmon Fish Using Gated Recurrent Unit (GRU) Model. *2022 2nd International Conference on Computer Science, Engineering and Applications, ICCSEA 2022*, 1–5. https://doi.org/10.1109/ICCSEA54677.2022.9936539

Arepalli, P. G., & Naik, K. J. (2023). An IoT-based water contamination analysis for aquaculture using lightweight multi-headed GRU model. *Environmental Monitoring and Assessment*, *195*(12), 1–21. https://doi.org/10.1007/s10661-023-12126-4

Attrapadung, N., Hamada, K., Ikarashi, D., Kikuchi, R., Matsuda, T., Mishina, I., Morita, H., & Schuldt, J. C. N. (2022). Adam in Private: Secure and Fast Training of Deep Neural Networks with Adaptive Moment Estimation. *Proceedings on Privacy Enhancing Technologies*, *2022*(4), 746–767. https://doi.org/10.56553/popets-2022-0131

Awoke, T., Rout, M., Mohanty, L., & Satapathy, S. C. (2021). Bitcoin price prediction and analysis using deep learning models. In *Communication software and networks* (Vol. 134, Issue October, pp. 631–640). Springer Singapore. https://doi.org/10.1007/978-981-15-5397-4_63

Ayhan, B., Vargo, E. P., & Tang, H. (2024). On the Exploration of Temporal Fusion Transformers for Anomaly Detection with Multivariate Aviation Time-Series Data. *Aerospace*, *11*(8). https://doi.org/10.3390/aerospace11080646

Baek, S. S., Pyo, J., & Chun, J. A. (2020). Prediction of water level and water quality using a cnn-lstm combined deep learning approach. *Water (Switzerland)*, *12*(12). https://doi.org/10.3390/w12123399

Bago, P., Castilho, S., Celeste, E., Dunne, J., Gaspari, F., Gislason, N. R., Kasen, A., Klubička, F., Kristmannsson, G., McHugh, H., Moran, R., Loinsigh, O. N., Olsen, J. A., Escartin, C. P., Ramesh, A., Resende, N., Sheridan, P., & Way, A. (2022). Sharing High-Quality Language Resources in the Legal Domain To Develop Neural Machine Translation for Under-Resourced European Languages. *Revista de Llengua i Dret*, *78*, 9–34. https://doi.org/10.2436/rld.i78.2022.3741

Barzegar, R., Aalami, M. T., & Adamowski, J. (2020). Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model. *Stochastic Environmental Research and Risk Assessment*, *34*(2), 415–433. https://doi.org/10.1007/s00477-020-01776-2

Bazar, C. (2021). Physico-chemical analysis of ground water quality from. *Journal of Chemical and Pharmaceutical Research*, *7817*(March), 94–98.

BBEIMAN, L. (2020). Bagging predictors. *Risks*, *8*(3), 1–26. https://doi.org/10.3390/risks8030083

Behrens, F., Leiprecht, S., Brantl, J., & Finkenrath, M. (2022). Temporal Fusion Transformer for thermal load prediction in district heating and cooling networks. *Proceedings of the 63rd International Conference of Scandinavian Simulation Society, SIMS 2022, Trondheim, Norway, September 20-21, 2022*, *192*, 333–338. https://doi.org/10.3384/ecp192047

Belete, D. M., & Huchaiah, M. D. (2022). Grid search in hyperparameter optimisation of machine learning models for prediction of HIV/AIDS test results. *International Journal of Computers and Applications*, *44*(9), 875–886. https://doi.org/10.1080/1206212X.2021.1974663

Bento, P. M. R., Pombo, J. A. N., Calado, M. R. A., & Mariano, S. J. P. S. (2021). Stacking ensemble methodology using deep learning and ARIMA models for short-term load forecasting. *Energies*, *14*(21), 1–21. https://doi.org/10.3390/en14217378

Bhateria, R., & Jain, D. (2016). Water quality assessment of lake water: a review. *Sustainable Water Resources Management*, *2*(2), 161–173. https://doi.org/10.1007/s40899-015-0014-7

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., & Ghayvat, H. (2021). Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics (Switzerland)*, *10*(20), 1–28. https://doi.org/10.3390/electronics10202470

Bischl, B., Richter, J., Becker, M., Binder, M., & Pielok, T. (2023). *Hyperparameter optimisation : Foundations , algorithms , best practices , and open challenges*. *November 2022*, 1–43. https://doi.org/10.1002/widm.1484

Boateng, E. Y., Otoo, J., & Abaye, D. A. (2020). Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review. *Journal of Data Analysis and Information Processing*, *08*(04), 341–357. https://doi.org/10.4236/jdaip.2020.84020

Bock, S., Goppold, J., & Weiß, M. (2018). An improvement of the convergence proof of the ADAM-Optimiser. *ArXiv Preprint ArXiv:1804.10587*, 1–5. http://arxiv.org/abs/1804.10587

Boyd, C. E. (2017). General relationship between water quality and aquaculture performance in ponds. In *Fish Diseases: Prevention and Control Strategies*. Elsevier. https://doi.org/10.1016/B978-0-12-804564-0.00006-5

Boyd, C. E., D'Abramo, L. R., Glencross, B. D., Huyben, D. C., Juarez, L. M., Lockwood, G. S., McNevin, A. A., Tacon, A. G. J., Teletchea, F., Tomasso, J. R., Tucker, C. S., & Valenti, W. C. (2020). Achieving sustainable aquaculture: Historical and current perspectives and future needs and challenges. *Journal of the World Aquaculture Society*, *51*(3), 578–633. https://doi.org/10.1111/jwas.12714

Breiman, L. (2001). Random Forests. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12343 LNCS*, 503–515. https://doi.org/10.1007/978-3-030-62008-0_35

Brown, M. L., & Kros, J. F. (2003). Data mining and the impact of missing data. *Industrial Management and Data Systems*, *103*(8–9), 611–621. https://doi.org/10.1108/02635570310497657

Budiman, F. (2019). SVM-RBF parameters testing optimisation using cross validation and grid search to improve multiclass classification. *Scientific Visualization*, *11*(1), 80–90. https://doi.org/10.26583/sv.11.1.07

Buslim, N., Rahmatullah, I. L., Setyawan, B. A., & Alamsyah, A. (2021). Comparing Bitcoin's Prediction Model Using GRU, RNN, and LSTM by Hyperparameter optimisation Grid Search and Random Search. *2021 9th International Conference on*

Cyber and IT Service Management, CITSM 2021, 1–6. https://doi.org/10.1109/CITSM52892.2021.9588947

Cahuantzi, R., Chen, X., & Güttel, S. (2023). A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences. *Lecture Notes in Networks and Systems*, *739 LNNS*, 771–785. https://doi.org/10.1007/978-3-031-37963-5_53

Çakir, M., Yilmaz, M., Oral, M. A., Kazanci, H. Ö., & Oral, O. (2023). Accuracy assessment of RFerns, NB, SVM, and kNN machine learning classifiers in aquaculture. *Journal of King Saud University - Science*, *35*(6), 102754. https://doi.org/10.1016/j.jksus.2023.102754

Campisi-Pinto, S., Adamowski, J., & Oron, G. (2012). Forecasting Urban Water Demand Via Wavelet-Denoising and Neural Network Models. Case Study: City of Syracuse, Italy. *Water Resources Management*, *26*(12), 3539–3558. https://doi.org/10.1007/s11269-012-0089-y

Ceesay, R., Boonchoo, T., & Rattanatamrong, P. (2023). Machine Learning Approaches for Quality Control in Pulp Packaging Manufacturers. In *Proceedings of JCSSE 2023 - 20th International Joint Conference on Computer Science and Software Engineering*. https://doi.org/10.1109/JCSSE58229.2023.10202113

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*, 189–215. https://doi.org/10.1016/j.neucom.2019.10.118

Chaudhury, S., & Yamasaki, T. (2021). Robustness of Adaptive Neural Network optimisation under Training Noise. *IEEE Access*, *9*, 37039–37053. https://doi.org/10.1109/ACCESS.2021.3062990

Chen, H., Du, M., Zhang, Y., & Yang, C. (2022). Research on Disease Prediction Method Based on R-Lookahead-LSTM. *Computational Intelligence and Neuroscience*, *2022*(2), 1–13. https://doi.org/10.1155/2022/8431912

Chen, Y., Fang, X., Yang, L., Liu, Y., Gong, C., & Di, Y. (2019). Artificial Neural Networks in the Prediction and Assessment for Water Quality: A Review. *Journal of Physics: Conference Series*, *1237*(4), 042051. https://doi.org/10.1088/1742-6596/1237/4/042051

Chen, Y., Song, L., Liu, Y., Yang, L., & Li, D. (2020). A Review of the Artificial Neural Network Models for Water Quality Prediction. *Applied Sciences*, *10*(17), 5776. https://doi.org/10.3390/app10175776

Chezhegov, S., Klyukin, Y., Semenov, A., Beznosikov, A., Gasnikov, A., Horváth, S., Takáč, M., & Gorbunov, E. (2024). *Gradient Clipping Improves AdaGrad when the Noise Is Heavy-Tailed*. https://doi.org/10.48550/arXiv.2406.04443

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1724–1734. https://doi.org/10.3115/v1/d14-1179

Cui, G., Pei, S., Rui, Z., Dou, B., Ning, F., & Wang, J. (2021). Whole process analysis of geothermal exploitation and power generation from a depleted high-temperature gas reservoir by recycling CO2. *Energy*, *217*(November), 119340. https://doi.org/10.1016/j.energy.2020.119340

Cui, X., Zhang, W., Tüske, Z., & Picheny, M. (2018). Evolutionary stochastic gradient descent for optimisation of deep neural networks. In *Advances in Neural Information Processing Systems* (Vols. 2018-Decem, pp. 6048–6058).

Cutkosky, A., & Busa-Fekete, R. (2018). Distributed stochastic optimisation via adaptive SGD. *Advances in Neural Information Processing Systems*, *2018-Decem*(NeurIPS), 1910–1919.

Datta S. (2012). Management of Water Quality in Intensive Aquaculture. *Respiration* , *6*(602), 1–18.

Deng, F., Liu, W., Sun, M., Xu, Y., Wang, B., Liu, W., Yuan, Y., & Cui, L. (2025). *Fine Estimation of Water Quality in the Yangtze River Basin Based on a Geographically Weighted Random Forest Regression Model*. 1–20.

Dereich, S., & Jentzen, A. (2024). *Convergence rates for the Adam optimiser*. http://arxiv.org/abs/2407.21078

Deva Hema, D., & Ashok Kumar, K. (2021). Hyperparameter optimisation of LSTM based Driver's Aggressive Behavior Prediction Model. *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, 752–757. https://doi.org/10.1109/ICAIS50930.2021.9396047

Di, Y., Gao, M., Feng, F., Li, Q., & Zhang, H. (2022). A New Framework for Winter Wheat Yield Prediction Integrating Deep Learning and Bayesian Optimisation. *Agronomy*, *12*(12), 1–15. https://doi.org/10.3390/agronomy12123194

Diab, S. (2019). Optimising Stochastic Gradient Descent in Text Classification Based on Fine-Tuning Hyper-Parameters Approach. *ArXiv Preprint ArXiv:1902.06542*, *16*(12), 1–6.

Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., & Samulowitz, H. (2017). An effective algorithm for hyperparameter optimisation of neural networks. *IBM Journal of Research and Development*, *61*(4), 1–11. https://doi.org/10.1147/JRD.2017.2709578

Ding, J., Ren, X., Luo, R., & Sun, X. (2019). *An Adaptive and Momental Bound Method for Stochastic Learning*. http://arxiv.org/abs/1910.12249

Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala, B., & Aigbavboa, C. O. (2018). A Comparative Analysis of Gradient Descent-Based optimisation Algorithms on Convolutional Neural Networks. *Proceedings of the International Conference on Computational Techniques, Electronics and Mechanical Systems, CTEMS 2018*, *December*, 92–99. https://doi.org/10.1109/CTEMS.2018.8769211

Dozat, T. (2016a). Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, *1*, 2013–2016.

Dozat, T. (2016b). Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, *1*, 2013–2016.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimisation. *Journal of Machine Learning Research*, *12*(7).

El-Shahat, D., Tolba, A., Abouhawwash, M., & Abdel-Basset, M. (2024). Machine learning and deep learning models based grid search cross validation for short-term solar irradiance forecasting. *Journal of Big Data*, *11*(1), 134. https://doi.org/10.1186/s40537-024-00991-w

El Ghazi, M., & Aknin, N. (2023). Optimising Deep LSTM Model through Hyperparameter Tuning for Sensor-Based Human Activity Recognition in Smart Home. *Informatica (Slovenia)*, *47*(10), 109–122. https://doi.org/10.31449/inf.v47i10.5268

Elesedy, B., & Hutter, M. (2023). U-Clip: On-Average Unbiased Stochastic Gradient Clipping. *ArXiv Preprint ArXiv*, 1–24. https://doi.org/10.48550/arXiv.2302.02971

Enayatollahi, A. (2024). *Modeling and Predicting Petroleum Wastewater Treatment Outcomes with Neural Networks*. *Iconip*, 1–3.

Ergen, T., & Kozat, S. S. (2018). Online Training of LSTM Networks in Distributed Systems for Variable Length Data Sequences. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(10), 5159–5165. https://doi.org/10.1109/TNNLS.2017.2770179

Eze, E., Halse, S., & Ajmal, T. (2021). Developing a novel water quality prediction model for a South African aquaculture farm. *Water (Switzerland)*, *13*(13), 1–19. https://doi.org/10.3390/w13131782

Ezgi, A., & Onan, A. (2023). 5 th International Conference on Applied Engineering and Natural Sciences Automatic Knee Osteoarthritis Severity Grading using Deep Neural Networks : Comparative Analysis of Network Architectures and optimisation Functions. *All Sciences Proceeding*, 197–203.

Fang, P., Wang, Y., Zhao, Y., & Kang, J. (2025). *Analysis of Prediction Confidence in Water Quality Forecasting Employing LSTM*.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimisation. *ArXiv Preprint ArXiv:1807.02811*, *Section 5*, 1–22. http://arxiv.org/abs/1807.02811

Freire, P. J., Napoli, A., Spinnler, B., Costa, N., Turitsyn, S. K., & Prilepsky, J. E. (2022). *Neural Networks-based Equalizers for Coherent Optical Transmission : Caveats and Pitfalls*. *28*, 1–24.

Freitas, J., Vaz-Pires, P., & Câmara, J. S. (2020). From aquaculture production to consumption: Freshness, safety, traceability and authentication, the four pillars of quality. *Aquaculture*, *518*(December 2019), 734857. https://doi.org/10.1016/j.aquaculture.2019.734857

Gambella, C., Ghaddar, B., & Naoum-Sawaya, J. (2021). optimisation problems for machine learning: A survey. *European Journal of Operational Research*, *290*(3), 807–828. https://doi.org/10.1016/j.ejor.2020.08.045

Gholamalinezhad, H., & Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. *ArXiv Preprint ArXiv:2009.07485*. http://arxiv.org/abs/2009.07485

Ghorbanzadeh, O., Shahabi, H., & Crivellari, A. (2022). Landslide detection using deep learning and object - based image analysis. *Landslides*, *2*(December 2021), 929–939. https://doi.org/10.1007/s10346-021-01843-x

Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A study on support vector machine based linear and non-linear pattern classification. *Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2019*, *Iciss*, 24–28. https://doi.org/10.1109/ISS1.2019.8908018

Giacomazzi, E., Haag, F., & Hopf, K. (2023). Short-Term Electricity Load Forecasting Using the Temporal Fusion Transformer: Effect of Grid Hierarchies and Data Sources. In *e-Energy 2023 - Proceedings of the 2023 14th ACM International Conference on Future Energy Systems* (Vol. 1, Issue 1). Association for Computing Machinery. https://doi.org/10.1145/3575813.3597345

Gladju, J., Kamalam, B. S., & Kanagaraj, A. (2022). Applications of data mining and machine learning framework in aquaculture and fisheries: A review. *Smart Agricultural Technology*, *2*(November 2021), 100061. https://doi.org/10.1016/j.atech.2022.100061

Gogtay, N. J., & Thatte, U. M. (2017). Principles of correlation analysis. *Journal of Association of Physicians of India*, *65*(MARCH), 78–81.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Grout, I., De Ferreira, W. A. P., & Silva, A. C. R. Da. (2019). On-Line Electrical Supply Generation Fuel Mix Data Analysis using Python and TensorFlow. *Proceedings of the 2019 International Conference on Power, Energy and Innovations, ICPEI 2019*, *2019*, 20–23. https://doi.org/10.1109/ICPEI47862.2019.8944972

Grzegorz Dudek. (2016). Pattern-based local linear regression models for short-term load forecasting. *Electric Power Systems Research*, *4*(June), 2016.

Guan, L. (2023). Weight Prediction Boosts the Convergence of AdamW. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13935 LNCS*, 329–340. https://doi.org/10.1007/978-3-031-33374-3_26

Guo, H., Nguyen, H., Vu, D., & Bui, X. (2019). Forecasting mining capital cost for open-pit mining projects based on arti fi cial neural network approach. *Resources Policy*, *December 2018*, 101474. https://doi.org/10.1016/j.resourpol.2019.101474

Guo, L., Li, M., Xu, S., & Yang, F. (2020). Application of {Stochastic} {Gradient} {Descent} {Technique} for {Method} of {Moments}. *2020 {IEEE} {International} {Conference} on {Computational} {Electromagnetics} ({ICCEM})*, 97–98. https://doi.org/10.1109/ICCEM47450.2020.9219400

Guo, Z., Xu, Y., Yin, W., Jin, R., & Yang, T. (2021). *A Novel Convergence Analysis for Algorithms of the Adam Family*. 1–13. http://arxiv.org/abs/2112.03459

Hadgu, A. T., Nigam, A., & Diaz-Aviles, E. (2015). Large-scale learning with AdaGrad on Spark. *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, *2*, 2828–2830. https://doi.org/10.1109/BigData.2015.7364091

Han, Y., Wang, C., Ren, Y., Wang, S., Zheng, H., & Chen, G. (2019). Short-term prediction of bus passenger flow based on a hybrid optimised LSTM network. *ISPRS International Journal of Geo-Information*, *8*(9), 366. https://doi.org/10.3390/ijgi8090366

Harimoorthy, K., & Thangavelu, M. (2021). Multi-disease prediction model using improved SVM-radial bias technique in healthcare monitoring system. *Journal of Ambient Intelligence and Humanized Computing*, *12*(3), 3715–3723. https://doi.org/10.1007/s12652-019-01652-0

Hassan, S. M., Maji, A. K., Jasiński, M., Leonowicz, Z., & Jasińska, E. (2021). Identification of plant-leaf diseases using cnn and transfer-learning approach. *Electronics (Switzerland)*, *10*(12), 1388. https://doi.org/10.3390/electronics10121388

He, Y., Huang, P., Hong, W., Luo, Q., Li, L., & Tsui, K.-L. (2024). In-Depth Insights into the Application of Recurrent Neural Networks (RNNs) in Traffic Prediction: A Comprehensive Review. *Algorithms*, *17*(9), 398. https://doi.org/10.3390/a17090398

Hinton, G. (2014). *Dropout : A Simple Way to Prevent Neural Networks from Overfitting*. *15*, 1929–1958.

Ho, R., & Hung, K. (2024). CEEMD-based Multivariate Financial Time Series Forecasting using a Temporal Fusion Transformer. *14th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2024*, 209–215. https://doi.org/10.1109/ISCAIE61308.2024.10576340

Hossain, M. D., Inoue, H., Ochiai, H., Fall, D., & Kadobayashi, Y. (2020). LSTM-based intrusion detection system for in-vehicle can bus communications. *IEEE Access*, *8*, 185489–185502. https://doi.org/10.1109/ACCESS.2020.3029307

Hosseini, F., Prieto, C., & Alvarez, C. (2024). Hyperparameter optimisation of regional hydrological LSTMs by random search: A case study from Basque Country, Spain. *Journal of Hydrology*, *643*(September), 132003. https://doi.org/10.1016/j.jhydrol.2024.132003

Hu, W. C., Chen, L. B., Wang, B. H., Li, G. W., & Huang, X. R. (2024). Design and Implementation of a Full-Time Artificial Intelligence of Things-Based Water Quality Inspection and Prediction System for Intelligent Aquaculture. *IEEE Sensors Journal*, *24*(3), 3811–3821. https://doi.org/10.1109/JSEN.2023.3340295

Hu, Z., Li, R., Xia, X., Yu, C., Fan, X., & Zhao, Y. (2020). A method overview in smart aquaculture. *Environmental Monitoring and Assessment*, *192*(8). https://doi.org/10.1007/s10661-020-08409-9

Huan, J., Chen, B., Xu, X. G., Li, H., Li, M. B., & Zhang, H. (2021). River dissolved oxygen prediction based on random forest and LSTM. *Applied Engineering in Agriculture*, *37*(5), 901–910. https://doi.org/10.13031/aea.14496

Huang, F., Cao, Z., Guo, J., Jiang, S. H., Li, S., & Guo, Z. (2020). Comparisons of heuristic, general statistical and machine learning models for landslide susceptibility prediction and mapping. *Catena*, *191*(April). https://doi.org/10.1016/j.catena.2020.104580

Huang, H., Wang, C., & Dong, B. (2019). Nostalgic ADAM: Weighting more of the past gradients when designing the adaptive learning rate. *IJCAI International Joint Conference on Artificial Intelligence*, *2019-Augus*, 2556–2562. https://doi.org/10.24963/ijcai.2019/355

Ibrahim, A., Ismail, A., Juahir, H., Iliyasu, A. B., Wailare, B. T., Mukhtar, M., & Aminu, H. (2023). Water quality modelling using principal component analysis and artificial neural network. *Marine Pollution Bulletin*, *187*(September 2022), 114493. https://doi.org/10.1016/j.marpolbul.2022.114493

Ida, Y., Fujiwara, Y., & Iwamura, S. (2017). Adaptive learning rate via covariance matrix based preconditioning for deep neural networks. *IJCAI International Joint Conference on Artificial Intelligence*, *0*, 1923–1929. https://doi.org/10.24963/ijcai.2017/267

Ilkanat, C. M. Y. ¸s. (2020). Spatio-temporal estimation of the daily cases of COVID-19 in worldwide using random forest machine learning algorithm. *Chaos, Solitons & Fractals*, *January*.

Ip, Y. K., & Chew, S. F. (2010). Ammonia production, excretion, toxicity, and defense in fish: A review. *Frontiers in Physiology*, *1 OCT*(October), 1–20. https://doi.org/10.3389/fphys.2010.00134

Islam, M., & Yasmin, R. (2017). International Journal of Fisheries and Aquatic Studies 2017; 5(4): 100-107 Impact of Aquaculture and Contemporary environmental issues in Bangladesh. *Ijfas*, *5*(4), 100–107. www.fisheriesjournal.com

Ivleva, N. P. (2021). Chemical Analysis of Microplastics and Nanoplastics: Challenges, Advanced Methods, and Perspectives. *Chemical Reviews*, *121*(19), 11886–11936. https://doi.org/10.1021/acs.chemrev.1c00178

J, S. D., Y, P. X., & Y, P. J. (2021). Micro Batch Streaming: Allowing the Training of DNN models Using a large batch size on Small Memory Systems. *ArXiv E-Prints*.

Jackson, C. M., & Adam, E. (2021). Machine learning classification of endangered tree species in a tropical submontane forest using worldview-2 multispectral satellite imagery and imbalanced dataset. *Remote Sensing*, *13*(24), 1–10. https://doi.org/10.3390/rs13244970

Jagannath, J., Polosky, N., O'Connor, D., Theagarajan, L. N., Sheaffer, B., Foulke, S., & Varshney, P. K. (2018). Artificial neural network based automatic modulation classification over a software defined radio testbed. *IEEE International Conference on Communications*, *2018-May*, 1–6. https://doi.org/10.1109/ICC.2018.8422346

Javed, U., Ijaz, K., Jawad, M., Khosa, I., Ahmad Ansari, E., Shabih Zaidi, K., Nadeem Rafiq, M., & Shabbir, N. (2022). A novel short receptive field based dilated causal

convolutional network integrated with Bidirectional LSTM for short-term load forecasting. *Expert Systems with Applications*, *205*(February), 117689. https://doi.org/10.1016/j.eswa.2022.117689

Jdi, H., & Falih, N. (2024). Leveraging transformer models for enhanced temperature forecasting: a comparative analysis in the Beni Mellal region. *Indonesian Journal of Electrical Engineering and Computer Science*, *36*(3), 1694–1700. https://doi.org/10.11591/ijeecs.v36.i3.pp1694-1700

Jensen, F. B. (2003). *Nitrite disrupts multiple physiological functions in aquatic animals*. *135*, 9–24.

Jentzen, A., Kuckuck, B., Neufeld, A., & Von Wurstemberger, P. (2021). Strong error analysis for stochastic gradient descent optimisation algorithms. *IMA Journal of Numerical Analysis*, *41*(1), 455–492. https://doi.org/10.1093/imanum/drz055

Jia, X., Feng, X., Yong, H., & Meng, D. (2024). Weight Decay With Tailored Adam on Scale-Invariant Weights for Better Generalisation. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(5), 6936–6947. https://doi.org/10.1109/TNNLS.2022.3213536

Jing, Y., Zhang, L., Hao, W., & Huang, L. (2022). Numerical study of a CNN-based model for regional wave prediction. *Ocean Engineering*, *255*(May), 111400. https://doi.org/10.1016/j.oceaneng.2022.111400

John, E. M., Krishnapriya, K., & Sankar, T. V. (2020). Treatment of ammonia and nitrite in aquaculture wastewater by an assembled bacterial consortium. *Aquaculture*, *526*(October 2019), 735390. https://doi.org/10.1016/j.aquaculture.2020.735390

Jongjaraunsuk, R., Taparhudee, W., & Suwannasing, P. (2024). Comparison of Water Quality Prediction for Red Tilapia Aquaculture in an Outdoor Recirculation System Using Deep Learning and a Hybrid Model. *Water (Switzerland)*, *16*(6), 907. https://doi.org/10.3390/w16060907

Joseph, S., Jo, A. A., & Raj, E. D. (2024). Improving Time Series Forecasting Accuracy with Transformers: A Comprehensive Analysis with Explainability. *2024 3rd International Conference on Electrical, Electronics, Information and Communication Technologies, ICEEICT 2024*, 1–7. https://doi.org/10.1109/ICEEICT61591.2024.10718609

Junankar, T., Sondhi, J. K., & Nair, A. M. (2023). Wheat Yield Prediction using Temporal Fusion Transformers. *2023 2nd International Conference for Innovation in Technology, INOCON 2023*, 1–6. https://doi.org/10.1109/INOCON57975.2023.10101144

Kadam, A. K., Wagh, V. M., Muley, A. A., Umrikar, B. N., & Sankhua, R. N. (2019). Prediction of water quality index using artificial neural network and multiple linear regression modelling approach in Shivganga River basin, India. *Modeling Earth Systems and Environment*, *5*(3), 951–962. https://doi.org/10.1007/s40808-019-00581-3

Kanai, S., Fujiwara, Y., & Iwamura, S. (2017). Preventing gradient explosions in gated recurrent units. *Advances in Neural Information Processing Systems, Nips*, 436–445.

Kang, L., Cui-Li, Y., & Jun-Fei, Q. (2021). Research on Forecasting Method for Effluent Ammonia Nitrogen Concentration Based on GRA-TCN. *3rd International Conference on Industrial Artificial Intelligence, IAI 2021*, 1–6. https://doi.org/10.1109/IAI53119.2021.9619251

Karkouch, A., Mousannif, H., Al Moatassime, H., & Noel, T. (2016). Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications*, *73*, 57–81. https://doi.org/10.1016/j.jnca.2016.08.002

Karri, R. R., Sahu, J. N., & Chimmiri, V. (2018). Critical review of abatement of ammonia from wastewater. *Journal of Molecular Liquids*, *261*, 21–31. https://doi.org/10.1016/j.molliq.2018.03.120

Kaur, G., Basak, N., & Kumar, S. (2024). State-of-the-art techniques to enhance biomethane/biogas production in thermophilic anaerobic digestion. *Process Safety and*

Environmental Protection, *186*(April), 104–117. https://doi.org/10.1016/j.psep.2024.03.123

KERVANCI, I. sibe., & AKAY, F. (2023). LSTM Hyperparameters optimisation with Hparam parameters for Bitcoin Price Prediction. *Sakarya University Journal of Computer and Information Sciences*, *6*(1), 1–9. https://doi.org/10.35377/saucis...1172027

Keuper, J., & Pfreundt, F. J. (2015). Asynchronous parallel stochastic gradient descent: A numeric core for scalable distributed machine learning algorithms. *Proceedings of MLHPC 2015: Machine Learning in High-Performance Computing Environments - Held in Conjunction with SC 2015: The International Conference for High Performance Computing, Networking, Storage and Analysis*, *Ml*. https://doi.org/10.1145/2834892.2834893

Khan, M. U. S., Jawad, M., & Khan, S. U. (2021). Adadb: Adaptive diff-batch optimisation technique for gradient descent. *IEEE Access*, *9*, 99581–99588. https://doi.org/10.1109/ACCESS.2021.3096976

Khan, P. W., & Byun, Y. C. (2023). Optimised Dissolved Oxygen Prediction Using Genetic Algorithm and Bagging Ensemble Learning for Smart Fish Farm. *IEEE Sensors Journal*, *23*(13), 15153–15164. https://doi.org/10.1109/JSEN.2023.3278719

Kharchenko, P. V. (2021). The triumphs and limitations of computational methods for scRNA-seq. *Nature Methods*, *18*(7), 723–732. https://doi.org/10.1038/s41592-021-01171-x

Kimm, H., Paik, I., & Kimm, H. (2021). Performance Comparision of TPU, GPU, CPU on Google Colaboratory over Distributed Deep Learning. *Proceedings - 2021 IEEE 14th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip, MCSoC 2021*, 312–319. https://doi.org/10.1109/MCSoC51149.2021.00053

Kingma, D. P., & Ba, J. (2014). Adam: {A} method for stochastic optimisation. *ArXiv Preprint ArXiv:1412.6980*.

Könemann, S., Kase, R., Simon, E., Swart, K., Buchinger, S., Schlüsener, M., Hollert, H., Escher, B. I., Werner, I., Aït-Aïssa, S., Vermeirssen, E., Dulio, V., Valsecchi, S., Polesello, S., Behnisch, P., Javurkova, B., Perceval, O., Di Paolo, C., Olbrich, D., … Carere, M. (2018). Effect-based and chemical analytical methods to monitor estrogens under the European Water Framework Directive. *TrAC - Trends in Analytical Chemistry*, *102*, 225–235. https://doi.org/10.1016/j.trac.2018.02.008

Kontopoulou, V. I., Panagopoulos, A. D., Kakkos, I., & Matsopoulos, G. K. (2023). A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks. *Future Internet*, *15*(8), 1–31. https://doi.org/10.3390/fi15080255

Kumar, R. P., Prabhu, T., Sowmya, J., & Kumar, A. N. (2024). Water Quality Prediction for Smart Aquaculture. *5th International Conference on Electronics and Sustainable Communication Systems, ICESC 2024 - Proceedings, Icesc*, 1376–1382. https://doi.org/10.1109/ICESC60852.2024.10690049

Kumar, S., Chandra Dutta, S., Goswami, K., & Mandal, P. (2021). Vulnerability assessment of building structures due to underground blasts using ANN and non-linear dynamic analysis. *Journal of Building Engineering*, *44*(May), 102674. https://doi.org/10.1016/j.jobe.2021.102674

Kuppusamy, P., Raga Siri, P., Harshitha, P., Dhanyasri, M., & Iwendi, C. (2023). Customized CNN with Adam and Nadam Optimisers for Emotion Recognition using Facial Expressions. *WiSPNET 2023 - International Conference on Wireless Communications, Signal Processing and Networking, June*, 1–5. https://doi.org/10.1109/WiSPNET57748.2023.10134002

Kurani, A., Doshi, P., Vakharia, A., & Shah, M. (2023). A Comprehensive Comparative Study of Artificial Neural Network (ANN) and Support Vector Machines (SVM) on Stock Forecasting. *Annals of Data Science*, *10*(1), 183–208. https://doi.org/10.1007/s40745-021-00344-x

Laborda, J., Ruano, S., & Zamanillo, I. (2023). Multi-Country and Multi-Horizon GDP Forecasting Using Temporal Fusion Transformers †. *Mathematics*, *11*(12), 1–26. https://doi.org/10.3390/math11122625

Lee, J. H., Lee, J. Y., Lee, M. H., Lee, M. Y., Kim, Y. W., Hyung, J. S., Kim, K. B., Cha, Y. K., & Koo, J. Y. (2022). Development of a short-term water quality prediction model for urban rivers using real-time water quality data. *Water Supply*, *22*(4), 4082–4097. https://doi.org/10.2166/ws.2022.038

Lehtovirta-Morley, L. E. (2018). Ammonia oxidation: Ecology, physiology, biochemistry and why they must all come together. *FEMS Microbiology Letters*, *365*(9), 1–9. https://doi.org/10.1093/femsle/fny058

Leong, W. C., Bahadori, A., Zhang, J., & Ahmad, Z. (2021). Prediction of water quality index (WQI) using support vector machine (SVM) and least square-support vector machine (LS-SVM). *International Journal of River Basin Management*, *19*(2), 149–156.

Levit, S. M. (2010). A Literature Review of Effects of Ammonia on Fish. *The Nature Conservancy. Protecting Nature. Preserving Life. Nature.Org*, *November*, 1–11. https://www.conservationgateway.org/ConservationByGeography/NorthAmerica/UnitedStates/alaska/sw/cpa/Documents/L2010ALR122010.pdf

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., & Gur-Ari, G. (2020). The large learning rate phase of deep learning: the catapult mechanism. *ArXiv Preprint ArXiv*. https://doi.org/https://doi.org/10.48550/arXiv.2003.02218

Li, D., Xu, X., Li, Z., Wang, T., & Wang, C. (2020). Detection methods of ammonia nitrogen in water: A review. *TrAC - Trends in Analytical Chemistry*, *127*, 115890. https://doi.org/10.1016/j.trac.2020.115890

Li, H., Cui, Z., Cui, H., Bai, Y., Yin, Z., & Qu, K. (2023). Hazardous substances and their removal in recirculating aquaculture systems: A review. *Aquaculture*, *569*(106), 739399. https://doi.org/10.1016/j.aquaculture.2023.739399

Li, L., Jiang, P., Xu, H., Lin, G., Guo, D., & Wu, H. (2019). Water quality prediction based on recurrent neural network and improved evidence theory: a case study of Qiantang River, China. *Environmental Science and Pollution Research*, *26*(19), 19879–19896. https://doi.org/10.1007/s11356-019-05116-y

Li, S., Liu, Y., & Methods, A. (2023). High Probability Analysis for Non-Convex Stochastic optimisation with Clipping. *ArXiv Preprint*. https://doi.org/10.48550/arXiv.2307.13680

Li, T., Lu, J., Wu, J., Zhang, Z., & Chen, L. (2022). Predicting Aquaculture Water Quality Using Machine Learning Approaches. *Water (Switzerland)*, *14*(18), 1–15. https://doi.org/10.3390/w14182836

Li, W., Wu, H., Zhu, N., Jiang, Y., Tan, J., & Guo, Y. (2021). Prediction of dissolved oxygen in a fishery pond based on gated recurrent unit (GRU). *Information Processing in Agriculture*, *8*(1), 185–193. https://doi.org/10.1016/j.inpa.2020.02.002

Li, Y., & Li, R. (2023). Predicting ammonia nitrogen in surface water by a new attention-based deep learning hybrid model. *Environmental Research*, *216*(P3), 114723. https://doi.org/10.1016/j.envres.2022.114723

Liang, D., Ma, F., & Li, W. (2020). New Gradient-Weighted Adaptive Gradient Methods with Dynamic Constraints. *IEEE Access*, *8*, 110929–110942. https://doi.org/10.1109/ACCESS.2020.3002590

Liao, F., & Zhao, C. (2016). *Water quality Prediction Model Based on fuzzy neural network*. *Mmebc*, 592–595. https://doi.org/10.2991/mmebc-16.2016.127

Liao, L., Li, H., Shang, W., & Ma, L. (2022). An Empirical Study of the Impact of Hyperparameter Tuning and Model optimisation on the Performance Properties of Deep Neural Networks. *ACM Transactions on Software Engineering and Methodology*, *31*(3), 1–40. https://doi.org/10.1145/3506695

Lim, B., Arık, S., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Lim, H. Il. (2019). A linear regression approach to modeling software characteristics for classifying similar software. *Proceedings - International Computer Software and Applications Conference*, *1*(2), 942–943. https://doi.org/10.1109/COMPSAC.2019.00152

Lindemann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, *99*(July 2020), 650–655. https://doi.org/10.1016/j.procir.2021.03.088

Liu, G., Jiang, Y., Zhong, K., Yang, Y., & Wang, Y. (2023). A time series model adapted to multiple environments for recirculating aquaculture systems. *Aquaculture*, *567*(January), 739284. https://doi.org/10.1016/j.aquaculture.2023.739284

Liu, S., Tai, H., Ding, Q., Li, D., Xu, L., & Wei, Y. (2013). A hybrid approach of support vector regression with genetic algorithm optimisation for aquaculture water quality prediction. *Mathematical and Computer Modelling*, *58*(3–4), 458–465. https://doi.org/10.1016/j.mcm.2011.11.021

Liu, X., Feng, R., Zhou, S., & Yang, Y. (2021). A Novel PSO-SGD with Momentum Algorithm for Medical Image Classification. *Proceedings - 2021 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2021*, 3408–3413. https://doi.org/10.1109/BIBM52615.2021.9669876

Liu, Z., Shen, Z., Li, S., Helwegen, K., Huang, D., & Cheng, K. T. (2021). How Do Adam and Training Strategies Help BNNs Optimisation? *Proceedings of Machine Learning Research*, *139*, 6936–6946.

Lopez Pinaya, W. H., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2019). Convolutional neural networks. In *Machine Learning: Methods and Applications to Brain Disorders*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-815739-8.00010-9

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*, *100*.

Lozano Jimenez, D. A., Kotteda, V. M. K., Kumar, V., & Gudimetla, V. S. R. (2019). Implementing Artificial Intelligence in Predicting Metrics for Characterizing Laser Propagation in Atmospheric Turbulence. *Journal of Fluids Engineering*, *141*(12). https://doi.org/10.1115/1.4043706

Lu, H., & Ma, X. (2020). Hybrid decision tree-based machine learning models for short-term water quality prediction. *Chemosphere*, *249*, 126169. https://doi.org/10.1016/j.chemosphere.2020.126169

Luaran, N., & Alfred, R. (2022). Assessment of the optimisation of Hyperparameters in Deep LSTM for Time Series Sea Water Tidal Shift. *Research Square*, 1–16. https://doi.org/10.21203/rs.3.rs-1669035/v1

Luo, J., Huang, S., & Wang, R. (2021). A fine-grained sentiment analysis of online guest reviews of economy hotels in China. *Journal of Hospitality Marketing and Management*, *30*(1), 71–95. https://doi.org/10.1080/19368623.2020.1772163

Luo, L., Xiong, Y., Liu, Y., & Sun, X. (2019). Adaptive gradient methods with dynamic bound of learning rate. *7th International Conference on Learning Representations, ICLR 2019*, *2018*, 1–19.

Mahsa, M., & Lee, T. (2018). Comparison of optimisation Algorithms in Deep Learning-Based Neural Networks for Hydrological Forecasting: Case Study of Nam River Daily Runoff. *Journal of the Korean Society of Hazard Mitigation*, *18*(6), 377–384. https://doi.org/10.9798/kosham.2018.18.6.377

Mai, V. V., & Johansson, M. (2021). Stability and Convergence of Stochastic Gradient Clipping: Beyond Lipschitz Continuity and Smoothness. *Proceedings of Machine Learning Research*, *139*, 7325–7335.

Malek, S., Mosleh, M., & Syed, S. M. (2014). Dissolved Oxygen Prediction Using Support Vector Machine. *International Journal of Computer, Information, Systems and Control Engineering*, *8*(1), 46–50.

Malviya, P., Mordido, G., Baratin, A., Harikandeh, R. B., Huang, J., Lacoste-Julien, S., Pascanu, R., & Chandar, S. (2023). *Promoting Exploration in Memory-Augmented Adam using Critical Momenta*. 1–26. http://arxiv.org/abs/2307.09638

Mandrikova, O., Fetisova, N., & Polozov, Y. (2021). Hybrid model for time series of complex structure with arima components. *Mathematics*, *9*(10), 1122. https://doi.org/10.3390/math9101122

Mao, Y., Pranolo, A., Wibawa, A. P., Putra Utama, A. B., Dwiyanto, F. A., & Saifullah, S. (2022). Selection of Precise Long Short Term Memory (LSTM) Hyperparameters based on Particle Swarm Optimisation. *Proceedings - International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2022*, *Icaaic*, 1114–1121. https://doi.org/10.1109/ICAAIC53929.2022.9792708

Masters, D., & Luschi, C. (2017). *Revisiting Small Batch Training for Deep Neural Networks*. 1–18.

Masthurah, A., Juahir, H., & Mohd Zanuri, N. B. (2021). Case study malaysia: Spatial water quality assessment of juru, kuantan and johor river basins using environmetric techniques. *Journal of Survey in Fisheries Sciences*, *7*(2), 19–40. https://doi.org/10.18331/SFS2021.7.2.2

Mehmood, F., Ahmad, S., & Whangbo, T. K. (2023). An Efficient optimisation Technique for Training Deep Neural Networks. *Mathematics*, *11*(6). https://doi.org/10.3390/math11061360

Michael, K. (2016). RFID/NFC implants for bitcoin transactions. *IEEE Consumer Electronics Magazine*, *5*(3), 103–106. https://doi.org/10.1109/MCE.2016.2556900

Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, *15*(9), 517. https://doi.org/10.3390/info15090517

Modoranu, I.-V., Safaryan, M., Malinovsky, G., Kurtic, E., Robert, T., Richtarik, P., & Alistarh, D. (2024). *MicroAdam: Accurate Adaptive optimisation with Low Space Overhead and Provable Convergence*. *NeurIPS*. http://arxiv.org/abs/2405.15593

Moeller, J. (2025). *Why and When You Should Avoid Using z-scores in Graphs Displaying Profile or Group Differences* . *11*(1), 58–78.

Muawanah, S., Muzayanah, U., Pandin, M. G. R., Alam, M. D. S., & Trisnaningtyas, J. P. N. (2023). file:///C:/Users/Tu Jun/Desktop/Updates following the defence of the proposal/literature/The exploding gradient problem demystified - definition,.pdf. *Qubahan Academic Journal*, *3*(4), 206–218. https://doi.org/10.48161/Issn.2709-8206

Mundt, M., Johnson, W. R., Potthast, W., Markert, B., Mian, A., & Alderson, J. (2021). A comparison of three neural network approaches for measurement units. *Sensors*, *21*(4535), 1–14.

Mustafa, S., Estim, A., Shapawi, R., Shalehand, M. J., & Sidik, S. R. M. (2021). Technological applications and adaptations in aquaculture for progress towards sustainable development and seafood security. *IOP Conference Series: Earth and Environmental Science*, *718*(1). https://doi.org/10.1088/1755-1315/718/1/012041

Nagaraju, T. V., Sunil, B. M., Chaudhary, B., Prasad, C. D., & R, G. (2023). Prediction of ammonia contaminants in the aquaculture ponds using soft computing coupled with wavelet analysis. *Environmental Pollution*, *331*(P2), 121924. https://doi.org/10.1016/j.envpol.2023.121924

Nagendra, B., & Singh, G. (2023). Comparing ARIMA, Linear Regression, Random Forest, and LSTM for Time Series Forecasting: A Study on Item Stock Predictions. *2023 4th IEEE Global Conference for Advancement in Technology, GCAT 2023*, 1–8. https://doi.org/10.1109/GCAT59970.2023.10353372

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1–21. https://doi.org/10.1186/s40537-014-0007-7

Nakisa, B., Rastgoo, M. N., Rakotonirainy, A., Maire, F., & Chandran, V. (2018). Long short term memory hyperparameter optimisation for a neural network based emotion recognition framework. *IEEE Access*, *6*, 49325–49338. https://doi.org/10.1109/ACCESS.2018.2868361

Namdeo, A., & Singh, D. (2021). WITHDRAWN: Challenges in evolutionary algorithm to find optimal parameters of SVM: A review. *Materials Today: Proceedings*, *16*(4), 2076. https://doi.org/10.1016/j.matpr.2021.03.288

Nejedly, P., Plesinger, F., Viscor, I., Halamek, J., & Jurak, P. (2019). Prediction of Sepsis Using LSTM with Hyperparameter optimisation with a Genetic Algorithm. *2019 Computing in Cardiology Conference (CinC)*, *45*, 1–4. https://doi.org/10.22489/cinc.2019.022

Nevavuori, P., Narra, N., & Lipping, T. (2019). Crop yield prediction with deep convolutional neural networks. *Computers and Electronics in Agriculture*, *163*(July), 104859. https://doi.org/10.1016/j.compag.2019.104859

Ni, J., Wang, Y., Tang, G., Cao, W., & Yang, S. X. (2024). *A lightweight GRU-based gesture recognition model for skeleton dynamic graphs*. 70545–70570. https://doi.org/10.1007/s11042-024-18313-w

Nie, Z., Zhang, X., Zhao, X., Xu, Y., Shi, D., Duan, J., & Wang, Z. (2019). Adaptive Online Learning with Momentum for Contingency-based Voltage Stability Assessment. *IEEE Power and Energy Society General Meeting*, *2019-Augus*, 1–5. https://doi.org/10.1109/PESGM40551.2019.8973825

Nitya Nand Jha. (2024). Computational Machine Learning Analytics for Prediction of Water Quality. *Communications on Applied Nonlinear Analysis*, *31*(4s), 448–465. https://doi.org/10.52783/cana.v31.942

Noh, S. H. (2021). Analysis of gradient vanishing of RNNs and performance comparison. *Information (Switzerland)*, *12*(11), 4422. https://doi.org/10.3390/info12110442

Noori, N., Kalin, L., & Isik, S. (2020). Water quality prediction using SWAT-ANN coupled approach. *Journal of Hydrology*, *590*. https://doi.org/10.1016/j.jhydrol.2020.125220

Nugroho, A., & Suhartanto, H. (2020). Hyper-Parameter Tuning based on Random Search for DenseNet Optimisation. *7th International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2020 - Proceedings*, 96–99. https://doi.org/10.1109/ICITACEE50144.2020.9239164

Obilor, E. I., & Amadi, E. C. (2020). Test for Significance of Pearson's Correlation Coefficien. *International Journal of Innovative Mathematics, Statistics & Energy Policies*, *6(1)*(August), 11–23.

Opoku, P. A., Shu, L., Ansah-Narh, T., Banahene, P., Yao, K. B. M. O., Kwaw, A. K., & Niu, S. (2024). Prediction of karst spring discharge using LSTM with Bayesian optimisation hyperparameter tuning: a laboratory physical model approach. *Modeling Earth Systems and Environment*, *10*(1), 1457–1482. https://doi.org/10.1007/s40808-023-01828-w

Or, B. (2020). *The Exploding and Vanishing Gradients Problem in Time Series*. Medium. https://medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22/

Oruh, J., Viriri, S., & Adegun, A. (2022). Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition. *IEEE Access*, *10*, 30069–30079. https://doi.org/10.1109/ACCESS.2022.3159339

Owaes, M., Gani, K. M., Kumari, S., Seyam, M., & Bux, F. (2024). Implementation of partial nitrification in wastewater treatment systems by modifications in operational strategies– a review. *Environmental Technology Reviews*, *13*(1), 379–397. https://doi.org/10.1080/21622515.2024.2354518

Palaiokostas, C. (2021). Predicting for disease resistance in aquaculture species using machine learning models. *Aquaculture Reports*, *20*(March), 100660. https://doi.org/10.1016/j.aqrep.2021.100660

Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics*, *45*(2), 227–248. https://doi.org/10.3102/1076998619872761

Parmar, A., Katariya, R., & Patel, V. (2019). A Review on Random Forest: An Ensemble Classifier. *Lecture Notes on Data Engineering and Communications Technologies*, *26*, 758–763. https://doi.org/10.1007/978-3-030-03146-6_86

Pethick, T., Antonakopoulos, K., & Silveti-falls, A. (2025). *Generalized Gradient Norm Clipping*. *NeurIPS*.

Pham, Q. B., Mohammadpour, R., Linh, N. T. T., Mohajane, M., Pourjasem, A., Sammen, S. S., Anh, D. T., & Nam, V. T. (2021). Application of soft computing to predict water quality in wetland. *Environmental Science and Pollution Research*, *28*(1), 185–200. https://doi.org/10.1007/s11356-020-10344-8

Phan, H., Andreotti, F., Cooray, N., Chen, O. Y., & De Vos, M. (2019). Joint Classification and Prediction CNN Framework for Automatic Sleep Stage Classification. *IEEE Transactions on Biomedical Engineering*, *66*(5), 1285–1296. https://doi.org/10.1109/TBME.2018.2872652

Philipp, G., Song, D., & Carbonell, J. G. (2017). The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions. *ArXiv Preprint ArXiv:1712*, *2014*. http://arxiv.org/abs/1712.05577

Prabowo, A. S., Sihabuddin, A., & SN, A. (2019). Adaptive Moment Estimation On Deep Belief Network For Rupiah Currency Forecasting. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *13*(1), 31. https://doi.org/10.22146/ijccs.39071

Pradeepkiran, J. A. (2019). Aquaculture role in global food security with nutritional value: A review. *Translational Animal Science*, *3*(2), 903–910. https://doi.org/10.1093/tas/txz012

Qiu, C., Li, Q., Jing, J., Tan, N., Wu, J., Wang, M., & Li, Q. (2025). *Transforming Prediction into Decision : Leveraging Transformer-Long Short-Term Memory*

Networks and Automatic Control for Enhanced Water Treatment Efficiency and Sustainability.

Rachmawati, D. A., Ibadurrahman, N. A., Zeniarja, J., & Hendriyanto, N. (2023). *Implementation of The Random Forest Algorithm in Classifying The Accuracy of Graduation Time for Computer Engineering Students at Dian Nuswantoro University*. *4*(3), 565–572. https://doi.org/10.52436/1.jutif.2023.4.3.920

Rahman, L. F., Marufuzzaman, M., Alam, L., Bari, M. A., Sumaila, U. R., & Sidek, L. M. (2021). Developing an ensembled machine learning prediction model for marine fish and aquaculture production. *Sustainability (Switzerland)*, *13*(16), 1–14. https://doi.org/10.3390/su13169124

Rahul Gandh, D., Rasheed Abdul Haq, K. P., Harigovindan, V. P., & Bhide, A. (2023). LSTM and GRU based Accurate Water Quality Prediction for Smart Aquaculture. *Journal of Physics: Conference Series*, *2466*(1). https://doi.org/10.1088/1742-6596/2466/1/012027

Rasheed Abdul Haq, K. P., & Harigovindan, V. P. (2022). Water Quality Prediction for Smart Aquaculture Using Hybrid Deep Learning Models. *IEEE Access*, *10*, 60078–60098. https://doi.org/10.1109/ACCESS.2022.3180482

Rasiya Koya, S., & Roy, T. (2024). Temporal Fusion Transformers for streamflow Prediction: Value of combining attention with recurrence. *Journal of Hydrology*, *637*(2018). https://doi.org/10.1016/j.jhydrol.2024.131301

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 1–23.

Reimers, N., & Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *ArXiv Preprint ArXiv:1707*, 1–6. http://arxiv.org/abs/1707.06799

Remya, S., & Sasikala, R. (2020). Performance evaluation of optimised and adaptive neuro fuzzy inference system for predictive modeling in agriculture. *Computers and Electrical Engineering*, *86*, 106718. https://doi.org/10.1016/j.compeleceng.2020.106718

Renu Nayar. (2020). Assessment of Water Quality Index and Monitoring of Pollutants by Physico-Chemical Analysis in Water Bodies: A Review. *International Journal of Engineering Research And*, *V9*(01), 178–185. https://doi.org/10.17577/ijertv9is010046

Riboni, A., Ghioldi, N., Candelieri, A., & Borrotti, M. (2022). Bayesian optimisation and deep learning for steering wheel angle prediction. *Scientific Reports*, *12*(1), 1–12. https://doi.org/10.1038/s41598-022-12509-6

Rom, A. R. M., Jamil, N., & Ibrahim, S. (2024). Multi objective hyperparameter tuning via random search on deep learning models. *Telkomnika (Telecommunication Computing Electronics and Control)*, *22*(4), 956–968. https://doi.org/10.12928/TELKOMNIKA.v22i4.25847

Sadiq, S. (2024). WATER QUALITY MANAGEMENT IN MARICULTURE. *ICAR-Central Marine Fisheries Research Institute*, 96–107.

Sahin, S. O., & Kozat, S. S. (2019). Nonuniformly Sampled Data Processing Using LSTM Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(5), 1452–1461. https://doi.org/10.1109/TNNLS.2018.2869822

Salama, A., Hassanien, A. E., & Fahmy, A. (2019). Sheep Identification Using a Hybrid Deep Learning and Bayesian optimisation Approach. *IEEE Access*, *7*, 31681–31687. https://doi.org/10.1109/ACCESS.2019.2902724

Sansa, I., Boussaada, Z., Mazigh, M., & Bellaaj, N. M. (2020). Solar radiation prediction for a winter day using ARMA model. *6th IEEE International Energy Conference, ENERGYCon 2020*, *2*(1), 326–330. https://doi.org/10.1109/ENERGYCon48941.2020.9236541

Sarjerao, J. S., & Sudhagar, G. (2024). Water Resource optimisation by Using a Hyper Parameter Tuned LSTM of a Smart Agriculture. *2024 International Conference on Emerging Smart Computing and Informatics, ESCI 2024*, 1–11. https://doi.org/10.1109/ESCI59607.2024.10497403

Schloss, K. B., Gramazio, C. C., Silverman, A. T., Parker, M. L., & Wang, A. S. (2019). Mapping Color to Meaning in Colormap Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, *25*(1), 810–819. https://doi.org/10.1109/TVCG.2018.2865147

Shams, M. Y., Elshewey, A. M., El-kenawy, E. S. M., Ibrahim, A., Talaat, F. M., & Tarek, Z. (2024). Water quality prediction using machine learning models based on grid search method. *Multimedia Tools and Applications*, *83*(12), 35307–35334. https://doi.org/10.1007/s11042-023-16737-4

Sharma, J., Soni, S., Paliwal, P., Saboor, S., Chaurasiya, P. K., Sharifpur, M., Khalilpoor, N., & Afzal, A. (2022). A novel long term solar photovoltaic power forecasting approach using LSTM with Nadam optimiser: A case study of India. *Energy Science and Engineering*, *10*(8), 2909–2929. https://doi.org/10.1002/ese3.1178

Shefat, S., Chowdhury, M., Haque, F., Hasan, J., Salam, M., & Shaha, D. (2021). Assessment of Physico-Chemical Properties of the Pasur River Estuarine Water. *Annals of Bangladesh Agriculture*, *24*(1), 1–16. https://doi.org/10.3329/aba.v24i1.51932

Shekar, B. H., & Dagnew, G. (2019). Grid search-based hyperparameter tuning and classification of microarray cancer data. *2019 2nd International Conference on Advanced Computational and Communication Paradigms, ICACCP 2019*, 1–8. https://doi.org/10.1109/ICACCP.2019.8882943

Shen, J., Qin, P., & Zhong, R. (2025). optimisation *of Temporal Feature Attribution and Sequential Dependency Modeling for High-Precision Multi-Step Resource Forecasting : A Methodological Framework and Empirical Evaluation*. 1–19.

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*(March), 1–43. https://doi.org/10.1016/j.physd.2019.132306

Shewalkar, A., nyavanandi, D., & Ludwig, S. A. (2019). Performance Evaluation of Deep neural networks Applied to Speech Recognition: Rnn, LSTM and GRU. *Journal of Artificial Intelligence and Soft Computing Research*, *9*(4), 235–245. https://doi.org/10.2478/jaiscr-2019-0006

Sheykhmousa, M., Mahdianpari, M., Ghanbari, H., Mohammadimanesh, F., Ghamisi, P., & Homayouni, S. (2020). Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 6308–6325. https://doi.org/10.1109/JSTARS.2020.3026724

Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. *ArXiv Preprint ArXiv:2305, Ml*, 1–61. http://arxiv.org/abs/2305.17473

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM. *ArXiv Preprint ArXiv:1911*. https://doi.org/https://doi.org/10.48550/arXiv.1911.09512

Silva, L. C. B., Lopes, B., Pontes, M. J., Blanquet, I., Segatto, M. E. V., & Marques, C. (2021). Fast decision-making tool for monitoring recirculation aquaculture systems based on a multivariate statistical analysis. *Aquaculture*, *530*(September 2020), 735931. https://doi.org/10.1016/j.aquaculture.2020.735931

Sipper, M. (2022). High Per Parameter: A Large-Scale Study of Hyperparameter Tuning for Machine Learning Algorithms. *Algorithms*, *15*(9), 1–13. https://doi.org/10.3390/a15090315

Song, Y., Liu, D., Yang, C., & Peng, Y. (2017). Data-driven hybrid remaining useful life estimation approach for spacecraft lithium-ion battery. *Microelectronics Reliability*, *75*, 142–153. https://doi.org/10.1016/j.microrel.2017.06.045

Staudemeyer, R. C., & Morris, E. R. (2019). *Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 1–42. http://arxiv.org/abs/1909.09586

Suarin, N. A. S., Lee, J. S., Chia, K. S., Fuzi, S. F. Z. M., & Al-Kaf, H. A. G. (2022). Artificial Neural Network and Near Infrared Light in Water pH and Total Ammonia Nitrogen Prediction. *International Journal of Integrated Engineering*, *14*(4), 228–238. https://doi.org/10.30880/ijie.2022.14.04.017

Sun, H., Cui, J., Shao, Y., Yang, J., Xing, L., Zhao, Q., & Zhang, L. (2024). A Gastrointestinal Image Classification Method Based on Improved Adam Algorithm. *Mathematics*, *12*(16). https://doi.org/10.3390/math12162452

Sun, H., Yu, H., Shao, Y., Wang, J., Xing, L., Zhang, L., & Zhao, Q. (2024). An Improved Adam's Algorithm for Stomach Image Classification. *Algorithms*, *17*(7), 1–13. https://doi.org/10.3390/a17070272

Sun, M., Yang, X., & Xie, Y. (2020). Deep Learning in Aquaculture: A Review. *Journal of Computers*, *31*(1), 294–319. https://doi.org/10.3966/199115992020023101028

Syanya, F. J., Litabas, J. A., Mathia, W. M., & Ntakirutimana, R. (2023). Nutritional Fish Diseases in Aquaculture: A Human Health Hazard or Mythical Theory: An Overview. *European Journal of Nutrition & Food Safety*, *15*(8), 41–58. https://doi.org/10.9734/ejnfs/2023/v15i81326

Tamut, H., Ghosh, R., & Gosh, K. (2025). *Enhancing Disease Detection in the Aquaculture Sector Using Convolutional Neural Networks Analysis*. *March 2024*, 1–19.

Tarek, Z., Elshewey, A. M., Shohieb, S. M., Elhady, A. M., El-Attar, N. E., Elseuofi, S., & Shams, M. Y. (2023). Soil Erosion Status Prediction Using a Novel Random Forest Model Optimised by Random Search Method. *Sustainability (Switzerland)*, *15*(9), 7114. https://doi.org/10.3390/su15097114

Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent Advances in Stochastic Gradient Descent in Deep Learning. *Mathematics*, *11*(3), 1–23. https://doi.org/10.3390/math11030682

van Dijk, M., Nguyen, P. H., Nguyen, T. N., & Nguyen, L. M. (2022). Generalizing DP-SGD with Shuffling and Batching Clipping. *ArXiv Privacy Policy*, *2*(3). https://doi.org/https://doi.org/10.48550/arXiv.2212.05796

Van Otten, N. (2023). *Gradient Clipping Explained & Practical How To Guide In Python. Spot Intelligence*. Spot Intelligence. https://spotintelligence.com/2023/12/01/gradient-clipping-explained-practical-how-to-guide-in-python/%0A

Villalobos-Arias, L., Quesada-López, C., Guevara-Coto, J., Martínez, A., & Jenkins, M. (2020). Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation. *PROMISE 2020 - Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering, Co-Located with ESEC/FSE 2020*, 31–40. https://doi.org/10.1145/3416508.3417121

Vizel, Y., Weissenbacher, G., & Malik, S. (2015). Boolean Satisfiability Solvers and Their Applications in Model Checking. *Proceedings of the IEEE*, *103*(11), 2021–2035. https://doi.org/10.1109/JPROC.2015.2455034

Wang, J., Li, J., Wang, X., Wang, J., & Huang, M. (2021). Air quality prediction using CT-LSTM. *Neural Computing and Applications*, *33*(10), 4779–4792. https://doi.org/10.1007/s00521-020-05535-w

Wang, J., Zhou, Q., & Zhang, X. (2018). Wind power forecasting based on time series ARMA model. *IOP Conference Series: Earth and Environmental Science*, *199*(2). https://doi.org/10.1088/1755-1315/199/2/022015

Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., & Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*, *360*, 107–119. https://doi.org/10.1016/j.neucom.2019.05.023

Wang, M., Yang, Y., Dai, E., & Rao, W. F. (2022). Fast optimize arm wearable piezoelectric energy harvesters via artificial neural network. *Materials Letters*, *326*(August), 132944. https://doi.org/10.1016/j.matlet.2022.132944

Wang, S., Ma, C., Xu, Y., Wang, J., & Wu, W. (2022). A Hyperparameter optimisation Algorithm for the LSTM Temperature Prediction Model in Data Center. *Scientific Programming*, *2022*. https://doi.org/10.1155/2022/6519909

Wang W, D, X., J, L., J, R., D, H., & S., L. (2024). Multi-model of ammonia nitrogen in aquaculture water based on EM algorithm. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, *48*(3), 233–244. https://doi.org/10.3233/JIFS-239032

Wang, X., Qiao, M., Li, Y., Tavares, A., Qiao, Q., & Liang, Y. (2023). Deep-Learning-Based Water Quality Monitoring and Early Warning Methods: A Case Study of Ammonia Nitrogen Prediction in Rivers. *Electronics (Switzerland)*, *12*(22), 4645. https://doi.org/10.3390/electronics12224645

Wang, Y. E., Wei, G.-Y., & Brooks, D. (2019). *Benchmarking TPU, GPU, and CPU Platforms for Deep Learning*. http://arxiv.org/abs/1907.10701

Wanjau, S., Wambugu, G., & Oirere, A. (2021). *Empirical Evaluation of Adaptive optimisation on the Generalisation Performance of Convolutional Neural Networks*. *October*.

Ward, R., Wu, X., & Bottou, L. (2019). Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 11574–11583.

Wendt, K., Trentzsch, K., Haase, R., Weidemann, M. L., Weidemann, R., Aßmann, U., & Ziemssen, T. (2022). Transparent Quality optimisation for Machine Learning-Based Regression in Neurology. *Journal of Personalized Medicine*, *12*(6). https://doi.org/10.3390/jpm12060908

Wu, Y., Wang, X., Wang, L., Zhang, X., Shi, Y., & Jiang, Y. (2023). Dynamic and explainable fish mortality prediction under low-concentration ammonia nitrogen stress. *Biosystems Engineering*, *228*, 178–192. https://doi.org/10.1016/j.biosystemseng.2023.03.003

Xie, Z., Wang, Y., & Sun, S. (2022). TCNNn Text Classification Model based on nAdam Optimiser. *ACM International Conference Proceeding Series*, 590–597. https://doi.org/10.1145/3558819.3565149

Xuan, Y., Si, W., Zhu, J., Sun, Z., Zhao, J., Xu, M., & Xu, S. (2021). Multi-Model Fusion Short-Term Load Forecasting Based on Random Forest Feature Selection and Hybrid Neural Network. *IEEE Access*, *9*, 69002–69009. https://doi.org/10.1109/ACCESS.2021.3051337

Yang, H., & Liu, S. (2021). A prediction model of aquaculture water quality based on multiscale decomposition. *Mathematical Biosciences and Engineering*, *18*(6), 7561–7579. https://doi.org/10.3934/mbe.2021374

Yang, J., Zhang, X., Xie, Y., Song, C., Sun, J., Zhang, Y., Giesy, J. P., & Yu, H. (2017). Ecogenomics of Zooplankton Community Reveals Ecological Threshold of Ammonia Nitrogen. *Environmental Science and Technology*, *51*(5), 3057–3064. https://doi.org/10.1021/acs.est.6b05606

Yang, P. Y., Tsai, J. T., Chou, J. H., Ho, W. H., & Lai, Y. Y. (2017). Prediction of water quality evaluation for fish ponds of aquaculture. *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2017*, *2017-Novem*, 545–546. https://doi.org/10.23919/SICE.2017.8105455

Yaqub, M., Asif, H., Kim, S., & Lee, W. (2020). Modeling of a full-scale sewage treatment plant to predict the nutrient removal efficiency using a long short-term memory (LSTM) neural network. *Journal of Water Process Engineering*, *37*(May), 101388. https://doi.org/10.1016/j.jwpe.2020.101388

Yi, D., Ji, S., & Bu, S. (2019). An Enhanced optimisation Scheme Based on Gradient Descent Methods for Machine Learning. *Symmetry*, *11*(7), 942. https://doi.org/10.3390/sym11070942

Yoon, H., Kim, Y., Ha, K., Lee, S. H., & Kim, G. P. (2017). Comparative evaluation of ANN-and SVM-time series models for predicting freshwater-saltwater interface fluctuations. *Water (Switzerland)*, *9*(5), 323. https://doi.org/10.3390/w9050323

Yu, H., Yang, L., Li, D., & Chen, Y. (2021). A hybrid intelligent soft computing method for ammonia nitrogen prediction in aquaculture. *Information Processing in Agriculture*, *8*(1), 64–74. https://doi.org/10.1016/j.inpa.2020.04.002

Yu, T., & Zhu, H. (2020). Hyper-Parameter Optimisation: A Review of Algorithms and Applications. *ArXiv Preprint ArXiv*, 1–56. http://arxiv.org/abs/2003.05689

Zabinsky, Z. B. (2009). Random Search Algorithms. *Wiley Encyclopedia of Operations Research and Management Science*. https://doi.org/10.1002/9780470400531.eorms0704

Zaheer, M., Reddi, S. J., Sachan, D., Kale, S., & Kumar, S. (2018). Adaptive methods for nonconvex optimisation. *Advances in Neural Information Processing Systems*, *2018-Decem*(NeurIPS), 9793–9803.

Zararsız, G., Çiçek, B., Kondolot, M., Mazıcıoğlu, M. M., Öztürk, A., & Kurtoğlu, S. (2017). Comparison of updated weight and height percentiles with previous references in 6-17-year-old children in Kayseri, Turkey. *JCRPE Journal of Clinical Research in Pediatric Endocrinology*, *9*(1), 39–47. https://doi.org/10.4274/jcrpe.3482

Zeebaree, D. Q., Abdulazeez, A. M., Abdullrhman, L. M., Hasan, D. A., & Kareem, O. S. (2021). The Prediction Process Based on Deep Recurrent Neural Networks: A Review. *Asian Journal of Research in Computer Science*, *August*, 29–45. https://doi.org/10.9734/ajrcos/2021/v11i230259

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *ArXiv*, *arXiv:1212*. https://doi.org/https://doi.org/10.48550/arXiv.1212.5701

Zeke Xie, Issei Sato, M. S. (2021). STABLE WEIGHT DECAY REGULARIZATION. *ICLR 2021*, 1–18.

Zhang, B., Jin, J., Fang, C., & Wang, L. (2020a). Improved analysis of clipping algorithms for non-convex optimisation. *Advances in Neural Information Processing Systems*, *2020-Decem*(NeurIPS).

Zhang, B., Jin, J., Fang, C., & Wang, L. (2020b). Improved analysis of clipping algorithms for non-convex optimisation. *Advances in Neural Information Processing Systems*, *2020-Decem*(NeurIPS), 15511–15521.

Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2020). Why Gradient Clipping Accelerates Training: a Theoretical Justification for Adaptivity. *8th International Conference on Learning Representations, ICLR 2020*, *2016*, 1–21.

Zhang, J., Hu, F., Li, L., Xu, X., Yang, Z., & Chen, Y. (2019). An adaptive mechanism to achieve learning rate dynamically. *Neural Computing and Applications*, *31*(10), 6685–6698. https://doi.org/10.1007/s00521-018-3495-0

Zhang, N., Lei, D., & Zhao, J. F. (2018). An Improved Adagrad Gradient Descent optimisation Algorithm. *2018 Chinese Automation Congress CAC*, 2359–2362. https://doi.org/10.1109/CAC.2018.8623271

Zhang, Q., Zhang, Y., Shao, Y., Liu, M., Li, J., Yuan, J., & Wang, R. (2023). Boosting Adversarial Attacks with Nadam Optimiser. *Electronics (Switzerland)*, *12*(6), 1–16. https://doi.org/10.3390/electronics12061464

Zhang, Q., Zhou, Y., & Zou, S. (2024). *Convergence Guarantees for RMSProp and Adam in Generalized-smooth Non-convex optimisation with Affine Noise Variance*. http://arxiv.org/abs/2404.01436

Zhang, S., Yang, X., Wang, Y., Zhao, Z., Liu, J., Liu, Y., Sun, C., & Zhou, C. (2020). Automatic Fish Population Counting by Machine Vision and a Hybrid Deep Neural Network Model. *Animals*, *10*(2), 364. https://doi.org/10.3390/ani10020364

Zhang, T. X., Li, M. R., Liu, C., Wang, S. P., & Yan, Z. G. (2023). A review of the toxic effects of ammonia on invertebrates in aquatic environments. *Environmental Pollution*, *336*(July), 122374. https://doi.org/10.1016/j.envpol.2023.122374

Zhong, H., Chen, Z., Qin, C., Huang, Z., Zheng, V. W., Xu, T., & Chen, E. (2020). Adam revisited: a weighted past gradients perspective. *Frontiers of Computer Science*, *14*(5), 1–17. https://doi.org/10.1007/s11704-019-8457-x

Zhou, G., Wang, G., Zhao, X., He, D., Zhao, C., & Suo, H. (2024). *Electrochemical Detection of Ammonia in Water Using NiCu Carbonate Hydroxide-Modified Carbon Cloth Electrodes: A Simple Sensing Method*. *24*(15), 4824. https://doi.org/10.3390/s24154824

Zhou, L., & Boyd, C. E. (2015). An assessment of total ammonia nitrogen concentration in Alabama (USA) ictalurid catfish ponds and the possible risk of ammonia toxicity. *Aquaculture*, *437*, 263–269. https://doi.org/10.1016/j.aquaculture.2014.12.001

Zhou, P., & Li, H. (2022). *Adan : Adaptive Nesterov Momentum Algorithm for Faster Optimising Deep Models*.

Zhou, X., Meng, X., & Li, Z. (2024). ANN-LSTM-A Water Consumption Prediction Based on Attention Mechanism Enhancement. *Energies*, *17*(5), 1102. https://doi.org/10.3390/en17051102

Zhu, B., Zhang, Y., Yin, J., Shi, J., & Li, L. (2023). Prediction of Sulfur Hexafluoride (SF6) Gas Pressure Time Series Data Based on Adaptive Particle Swarm Optimisation- Long Short Term Memory (APSO-LSTM). *ACM International Conference Proceeding Series*, 372–375. https://doi.org/10.1145/3592686.3592753

Zhu, T., Chen, T., Kuangt, L., Zeng, J., Li, K., & Georgiou, P. (2023). Edge-Based Temporal Fusion Transformer for Multi-Horizon Blood Glucose Prediction. *Proceedings - IEEE International Symposium on Circuits and Systems*, *2023-May*. https://doi.org/10.1109/ISCAS46773.2023.10181448

Ziyin, L., Li, B., Simon, J. B., & Ueda, M. (2021). Sgd with a constant large learning rate can converge to local maxima. *Published as a Conference Paper at ICLR*, 1–25. http://arxiv.org/abs/2107.11774

Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*(1), 11119–11127. https://doi.org/10.1109/CVPR.2019.01138

Zou, T., & Sugihara, T. (2020). Fast identification of a human skeleton-marker model for motion capture system using stochastic gradient descent method. *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, *2020-Novem*, 181–186. https://doi.org/10.1109/BioRob49111.2020.9224442