Optimizing sluggish state-based neural networks for effective time-series processing

Oluwatamilore Oluwatoyin Orojo

School of Science and Technology

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of

Doctor of Philosophy

June, 2021

"My thesis is dedicated to my parents, I am immensely grateful for your sacrifice, putting the boys and I first, and investing so much in us. Thank you so much, I love you both so dearly."
Your baby girl, Oluwatamilore

Acknowledgements

I am so grateful to God for making this a reality, for putting the right people in my life to support me through this journey and for providing all the opportunities I needed.

My deepest gratitude to Dr. Jonathan Tepper, without whom this would not have been a success. Jon, thank you for unwavering support and belief in me, for inspiring, encouraging and motivating me. You are an exceptional supervisor and mentor. I would also like to thank the Prof. T.M McGinnity, thank you for your wealth of knowledge and expertise you offered to me on this journey, specifically for your invaluable inputs, corrections and contributions, I am grateful. Many thanks to Dr. Mufti Mahmud for all your contribution and support, for encouraging and guiding me with publications and developing as an academic. I am indebted to all three of my supervisors, without them this would not be a reality. Thank you all for prioritising my work.

I am grateful to my parents, who provided the support for me to commence and complete this journey. To my dad, thank you for offering to read my work, pray for, and to encourage and support me. I love you daddy. Thank you mama for being my mum and a friend, and for praying for me, love you loads.

I am grateful for my best friends, Keisha Coggins, who consistently checked up on my progress, encouraged me, and so much more, and Adejoke Osuntogun, who motivated and encouraged me throughout the course of my journey. I am thankful for amazing friends I have been blessed with, To Christelle Ngakoumou, you have been an emotional rock for me through the ups and downs and craziness, I am

indebted. To Ketty Ngankam Mambou, who I met half-way through my journey, who prayed for and with me consistently and encouraged me constantly. To Clement Eke, you are an exceptional friend, you have been by my side since day one and I am so thankful. I am grateful for wise counsel and support from many like Roger Frimpong who constantly called me and prayed for me, thank you for being a friend and father figure to me and Marilyn Dean, what a privilege to have you in my life, thank you for the time you devoted to support me with my thesis I am indeed thankful. Merci beaucoup Ferdinand Akouete-kuevey for being one of my biggest supporters and being there for me.

Finally, I am thankful for the friends and colleagues I have made along the way during my journey at NTU. Thank you to everyone who in one way or another contributed to the success of my research.

This work was supported by Nottingham Trent University's Vice Chancellors' Research Scheme award, for the duration 2017 - 2020.

Oluwatamilore Oluwatoyin Orojo
June 2021

Abstract

The devastating personal and economic upheaval caused by the financial crises in 2007/2008 and more recently, the spread of Covid-19 from Feb 2020 till date (June 2021) strongly highlighted the need for effective time-series processing models that can provide useful insights and accurate forecasts in a timely manner to inform critical decision-making that affects lives and livelihoods. Thus, this research is focused on identifying an effective and suitable time-series approach that harnesses advantages of the current state-of-the-art forecasting models whilst mitigating their challenges. A critical review of existing state-of-the-art methods revealed the following two key attributes are required for effective time-series processing: *a robust yet flexible memory mechanism* and *minimal computational complexity for modelling complex dynamic time-series*. The Multi-recurrent Neural Network (MRN) was identified as the preferred model and subject to critical examination and enhancement due to its unique and powerful sluggish state-based memory mechanism that has largely gone unnoticed since its first introduction by Claudia Ulbricht from the University of Austria in 1994.

This thesis subsequently makes the following four meaningful contributions to the research field: a) the MRN was applied to different real-world temporal problems (where it had not previously been applied) *(of varying complexity)*. It was then compared to current state-of-the-art forecasting methods, where it demonstrated superior performance. It was critically assessed to identify limitations and points of extension; b) the MRN's hidden layer was endowed with periodically attentive units to tackle two well-known issues affecting artificial neural

networks; *vanishing gradient problem* and *catastrophic interference*. This innovation applied to the hidden layer encouraged the network to organise features according to different units of time. Therefore, reducing the information processing load placed on individual hidden units. Thus, alleviating the issue of catastrophic interference. In addition, the network was able to hold information for longer periods of time, as the unit partitions only responded at specific time intervals. This provided a means to mitigating the vanishing gradient problem, which in most instances led to better performance; c) the MRN was endowed with an innovative self-learning mechanism, to reduce user input and identify architectural hyper-parameters. This extension enabled the MRN to inform and enhance its internal memory composition *(and thus quality)* through incorporating Ratio Control Units to learn the layer-link ratios. This technique provided a new outlook on algorithm development, in particular pointing to the abilities of recurrent neural networks, and in particular, the MRN, to innately learn the importance of historical context rather than relying on hyper-parameters manually set by the user and d) a framework incorporating one of the proposed MRN innovations together with a one-shot pruning algorithm (based on the learnt ratio similarity) was proposed. The framework specifically provided a means of obtaining 'good' models by eliminating the need to train numerous models to exhaustively explore the search space for the optimum memory bank configuration. The new innovation simply requires one large over-parameterised MRN to be trained. More specifically, the pruning algorithm will automatically identify the optimum memory bank configuration in a robust manner which minimises the coupling of memory banks whilst maximising, or at least retaining, strong generalisation ability.

Finally, a critical discussion of the innovations proposed is given together with a number of insights and suggestions for further work. The key areas for improvements are i) employing different activation functions for the ratio learning, ii) extending the pruning algorithm to

not only prune based on ratio similarity but on memory bank importance, iii) learning the self-link ratios rather than just the layer-link ratios iv) developing deep MRNs for more complex temporal problems and v) applying knowledge extraction techniques to understand the quality of the underlying state-based representations formed. The work of this thesis has been published in IEEE Symposium Series on Computational Intelligence, International Joint Conference on Neural Networks and the Applied Intelligence and Informatics Conference.

Contents

Nomenclature

Acronyms

ANN Artificial Neural Network

AR Auto-regressive

ARMA Auto-Regressive Moving Average

BP Back-Propagation

BPTT Back-Propagation Through Time

COD Change of Direction

ERG Embedded Reber Grammar

ESN Echo State Network

EWMA Exponential Weighted Moving Average

EWS Early Warning System

GESN Growing Echo State Network

H-TAU Holistic Time Attentive Units

IoT Internet of Things

IOVMRN Improvement over Vanilla MRN

LFP Local Field Potential

LLR Layer-link ratios

LSTM Long-Short Term Memory

LVQ Learning Vector Quantisation

MA Moving Average

MAPE Mean Absolute Percentage Error

MB Memory Bank

MCC Matthew Correlation Coefficient

ML Machine Learning

MLFN Multilayered Feedforward Neural Network

MLP Multi-layer Perceptron

MRN Multi-recurrent Neural Network

NARX Nonlinear Autoregressive models with eXogenous input Neural Network

NBER National Bureau of Economic Research

NeST Neural Network Synthesis Tool

OBD Optimal Brain Damage

OBS Optimal Brain Surgeon

PA Periodically Attentive

PA-MRN Periodically Attentive Multi-recurrent Neural Network

RCU Ratio Control Units

RMSE Root-Mean Squared Error

RNN Recurrent Neural Network

RW Random Walk

SGD Stochastic Gradient Descent

SI Similarity Index

SL-MRN Self Learning Multi-recurrent Neural Network

SLR Self-link ratio

SRN Simple Recurrent Network

SVM Support Vector Machine

SWaP Size, Weight and Power

TAU Time Attentive Units

TDNN Time-delay Neural Network

TS Traditional Statistical

VAR Vector AutoRegressive

VARMA Vector AutoRegressive Moving Average

VMA Vector Moving Average

WMA Weighted Moving Average

Subscripts

t time step

Greek Symbols

β Auto-regressive model parameters

ϵ White noise

ϕ Moving average model parameters

σ^2 variance

Other Symbols

$\mathbf{X}$	input vector
A	actual output
b	bias
D	delta
d	memory order
E	error
F	momentum
f	function
f^{-1}	inverse function
h	hidden outputs
h	hidden units
i	input gate
L	Learning rate
l	forget gate
M	memory
N	neuron set
n	total number
O	calculated output
o	output gate
o	output units
p	pattern
S_t	state variable

U weight matrix

V connections between neurons

W weights

x data input

Roman Symbols

$\mathbb{R}$ real numbers

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the introduction to this thesis. Section 1.1 provides the background and motivation for the research and Section 1.2 presents a research overview of the chosen topic and describes the proposed work to be undertaken. The research questions are outlined in Section 1.3 and Section 1.3.1 outlines the aim and objectives of this research. Finally, the thesis structure and a brief chapter summary is given in Section 1.4.

1.1 Background and Motivation

Time-series data *(also known as temporal data)* is a collection of chronologically ordered observations [55]. Such data typically have embedded signal information through both spatial and temporal dimensions. For example, daily, monthly or quarterly observations of financial and business data series can be analysed to map underlying trends within the data, which aids the prediction of associated target variables such as changes in Gross Domestic Product (GDP) for predicting business cycle turning points or changes in daily or weekly Covid-19 cases to forecast future cases, informing decisions by policymakers to mitigate its spread. Employing time-series data rather than static data particularly for such essential tasks is key as more meaningful insights can be gleaned from the dynamically evolving nature of the important factors (for example, exponential growth in Covid-19 cases or consecutive downward movements in GDP). However, despite

their advantages, time-series data poses challenges not typically seen with static data, such as increased complexity due to structural breaks and non-stationarity, and the introduction of spatio-temporal dependencies.

The rapid spread and upheaval of Covid-19 over the last year lucidly demonstrated the need for sophisticated modelling techniques, to better capture the underlying trends and patterns within the series, providing more accurate forecasts. Current state-of-the-art time-series forecasting methods are mainly centred on Artificial Neural Networks (ANNs) [71] and in particular, Recurrent Neural Networks (RNNs), that embed historical information and have been shown to inform the learning and identification of spatio-temporal dependencies. RNN models have varying complexities from simple memory-based networks with little integration of past and recent information *(such as; Simple Recurrent Network)* to more complex gated-based memory networks *(such as; Long-Short Term Memory (LSTM))* that utilise sophisticated mechanisms to learn to latch and integrate long- and short-term information over time. LSTM [84] variants of the RNN class have typically been the preferred model, even though they generally require many adjustable parameters, and have ad hoc complex memory mechanisms. In addition, the training process is lengthy and cumbersome with the requirement of user intervention for hyper-parameter selection, to the point where their use could become intractable.

This thesis therefore focuses on assessing and extending the relatively unknown yet powerful Multi-recurrent Neural Network (MRN) [208]. MRNs were first proposed in 1994 by Claudia Ulbricht at The University of Austria and are a variant from the simple recurrent class of RNNs (Elman [52] and Jordan networks [96]), however, they are endowed with a more sophisticated memory mechanism than the standard SRN. The MRN allows recent information fed back from the input, hidden and output layers to be integrated with historical information from these layers stored in the context layer (without any complex gating mechanisms as found in the LSTM). This approach to integrating historic and current information allows the formation of a sluggish-state based memory that reduces 'forgetting' (a key issue with most RNNs from the simple recurrent network class). The potential of the MRN has been demonstrated by [20,149,150,183,200,208] in terms of predictive accuracy and reduced network complexity, however, despite

this, the MRN has largely gone unnoticed. The thesis will provide a critical review of the two main model categories exploited for time-series modelling and forecasting: *Traditional Statistical (TS) models* and *Machine Learning (ML)*, particularly focusing on the shift towards ML given the limitations of TS. This thesis will address the following question: ***"Is the MRN a suitable alternative class of RNNs, that mitigates the limitations of current techniques and is efficacious with respect to both generalisation performance and model complexity?"***.

1.2 Overview of the Research

This research will investigate the suitability of MRNs in line with the literature and with respect to the current state-of-the-art models for time-series forecasting. The literature review motivates and informs a critique and subsequent evaluations of the MRN against current state-of-the-art methods using datasets from real-world problem domains *(such as; financial and medical)*. In this research, experiments will be conducted with these datasets across various forecast horizons *(for example, predicting monthly oil price for '$t+1$' or '$t+12$', that is 1 or 12 month(s) ahead)*. This is particularly useful as the model's credibility and versatility can be thoroughly assessed. In general, for '$t + n$' as n gets bigger, more errors are accumulated, due to inherent unpredictability for complex events further in time, which can obscure the underlying signal. Thus, applying the models for different forecast horizons provides a means to assess their abilities to reveal the underlying signal in different temporal contexts *(particularly in complex time-varying domains)* and as such assessing their robustness for modelling and forecasting. The MRN will be critically assessed to identify inherent limitations, highlighting points of extension. Model extensions will be proposed and assessed to identify whether they mitigate the limitations identified, encourage better learning and enhance performance. Should these optimisations of the MRN be successful, this would provide researchers and decision-makers with an effective prediction tool for (non-linear and volatile) time-series.

1.3 Research Challenges and Questions

Following the overview of the research, the following research questions are investigated in this thesis:

1. In light of the growing availability of time-series data and as such the need for effective time-series models, the first research question is:

   Does the MRN offer a robust alternative to current state-of-the-art models, such that it more effectively models and forecasts time-series data across a range of different problem domains and consistently provides superior performance?

2. ANNs are notoriously known to suffer from inherent learning limitations such as the vanishing gradient problem and catastrophic inference. In addition, RNNs have specific architectural limitations that inhibit their performance. For example, the use of simple feedback in Elman Networks *(also known as Simple Recurrent Networks)* utilises only the most recent hidden state information, this leads to a lack of explicitness of historical information, thus, limiting its ability to preserve important historic information as more inputs are processed. These limitations significantly impede performance and thus the second research question is:

   Can the MRN class of models be extended and endowed to mitigate such inherent learning & architectural limitations to encourage more robust learning of both recent and historical information such that performance is enhanced?

3. A major disadvantage with RNNs is identifying suitable (general and architectural specific) hyper-parameters in the search space. Whilst the MRN appears to have an effective flexible memory mechanism, the model size can drastically increase as memory banks are added or units increase. This motivates the following third research question:

   How can the MRN be adapted to reduce model complexity, particularly, reducing the search space whilst minimising the impact on generalisation ability?

1.3.1 Research Aims and Objectives

The aim and objectives of this research to address the research questions presented are outlined in this section. The first aim of this research is to thoroughly investigate the MRN's suitability for time-series forecasting, critically assessing it, to identify its limitations. The second aim is to explore and extend the MRN to mitigate the limitations identified and develop a suitable paradigm for time-series modelling. To achieve the two aims, the following research objectives will be met:

Objective 1.1 Critically review the current research literature for existing time-series forecasting *(particularly TS & ANNs)* models and assess the MRN in light of the review to identify its suitability and whether it possess key attributes for effective time-series forecasting.

Objective 1.2 Identify suitable time-series datasets and appropriate data pre-processing and transformation techniques for a number of real-world problem domains.

Objective 1.3 Define a set of appropriate model fitting, selection and evaluation techniques, and metrics to evaluate performance.

Objective 1.4 Critically assess the MRN to identify the inherent learning and architectural limitations.

Objective 1.5 Identify appropriate solutions to extend the MRN which mitigate the inherent learning and architectural limitations (such as catastrophic interference, vanishing gradients and model complexity) and enhance performance.

Objective 1.6 Evaluate the efficacy of the paradigm developed and its potential for time-series modelling & forecasting using the selected datasets representing real-world problems.

1.4 Thesis structure

This thesis comprises the following eight chapters:

Chapter 2 presents a comprehensive literature review of time-series modelling and forecasting using traditional statistical methods and Artificial Neural Network models. The literature review critically assesses and identifies the key attributes for time-series processing and forecasting. The MRN is identified as a suitable modelling paradigm and assessed in light of the key attributes; *appropriate memory mechanism* and *suitable computational complexity*, required for time-series modelling, for application, exploration and extension.

In Chapter 3, the proposed methodology for time-series forecasting is presented. The chapter presents an in-depth explanation of the chosen paradigm, its structure and architectural design, the forecasting methodologies and the data utilised. The Multi-recurrent Neural Network coupled with the sliding window technique along with a forecast horizon and ensemble approach is presented. A summary of the current research to date with the chosen paradigm along with its suitability for time-series applications is discussed.

In Chapter 4, the MRN is first applied across domains where it has not been previously applied. The MRN is evaluated on real-world problems from the time-series domain *(of varying complexity)*, to ascertain its suitability for time-series processing. The performance of the MRN is also critically compared to the current state-of-the-art forecasting models to determine its comparative performance. Finally, the limitations of the MRN are presented.

Chapter 5 presents an extension employing periodic attentiveness, to mitigate two key learning limitations, namely, *vanishing gradient problem* and *catastrophic interference*. This extension is compared to the standard MRN *(as presented in Chapter 3)*, to assess its comparative performance across four different problem domains introduced in Chapter 3.

Chapter 6 introduces and extends the MRN with self-learning attributes. The extension is proposed to alleviate an inherent architectural limitation of the MRN, namely, the requirement for designer input (particularly for the memory ratios). The model extension is applied to four different problem domains introduced in Chapter 3 and compared to the standard MRN *(as presented in Chapter 3)*, to

understand whether they provide an improvement in performance.

Chapter 7 introduces a novel framework to obtain good models while pruning the memory where possible. The framework incorporates the model extension presented in Chapter 6 and is applied to four different problem domains introduced in Chapter 3. This framework is conclusively assessed and discussed for efficacy and performance.

Chapter 8 presents a summary of the research findings. In addition, the contributions of this thesis relative to the aims and objectives are discussed and reviewed. The contributions of this thesis lie in context of assessing the MRN for time-series modelling, extending it to mitigate current limitations and to identify a suitable time-series modelling framework. Finally, the limitations are discussed and suggested future work and recommendations presented.

Chapter 2

Literature Review

In this chapter, a summarised critical review of the main computational approaches to time-series processing is presented. The chapter begins with a brief introduction to time-series data and its characteristics, which is pivotal to understanding the main tenets of each method. The review highlights notable techniques in the literature including traditional state-of-the-art statistical approaches *(for example, the Box-Jenkins and Regime switching approaches)* and machine learning approaches such as feed-forward and recurrent neural networks. The methodological, predictive strengths and limitations of each approach are presented. The chapter concludes with an argument that further investigation of the Multi-Recurrent Neural Networks class of artificial neural networks for advanced time-series processing is warranted. A case is made for this particular paradigm, due to their ability to capture the computational power of existing statistical and neural network approaches whilst retaining computational simplicity and elegance not observed in other current state-of-the-art neural networks, such as Long short-term memory models.

2.1 Time-series data

Time-series data is formally defined as a sequence of chronologically ordered observations indexed in time, t [73]. Each observation within a sequence is typically

dependent indirectly or directly on the preceding observation.

$$\vec{x} = (x_1, x_2, ..., x_n) \tag{2.1}$$

Time-series data is usually observed a finite number of times. Specifically, a time-series with a single variable is referred to as a *uni-variate series* whilst one with multiple variables is referred to as a *multivariate series* [2, 46, 73, 187].

Time-series data can be observed in various forms (discrete, continuous, binary) and different sectors (for example, finance: stock market prices for a given quarter; medicine: patient vital signs during a hospital stay; education: yearly student intake for a county). This contrasts with static data, such as stock records stored in a transactional database for a retail company or snapshot of images captured by a camera for a property security system.

2.1.1 Components of time-series

Time-series data are generally affected by four main components, which can then be separated from the actual observations [2]. These components are *Trend*, *Cyclical*, *Seasonal* and *Random* [2].

Trend: this is described as the long-term direction of the data, which could be either upward, downward or constant, in either linear or non-linear form [61]. For example, the spread and death rate of Covid-19 in 2020 followed an upward trend.

Cyclical: these are potentially irregular swings indicating an upward or downward movement of a given trend (its duration is dependent on the sector) [61]. For example, most financial and economic data demonstrate cyclical variations, such as business cycle turning points. Another example, retail sales in winter, there is a peak for heavy woollen coats, hats and mitts, and in summer, for t-shirts and shorts.

Seasonal: these are regular, repetitive and relatively short upward or downward fluctuations [61]. This component is usually measured in quarters (Winter, Spring, Summer, and Autumn) [61]. For example, the sales of boots in winter,

or the number of holiday bookings in summer.

Random: these are random upward or downward variations over a specific time period [61]. This is the unpredictable aspect of any given time-series. For example, pandemic, war, drought or political upheaval.

Understanding and identifying the various components of any given time-series dataset is essential in order to properly model the underlying signal rather than the components of the data [46].

For the purpose of this research, with a focus on time-series modelling and forecasting, a brief overview of time-series data is provided. The remainder of this chapter presents methods that attempt to learn statistics and patterns from various time-series datasets, with a view to making meaningful predictions for the future values of these time-series, thereby modelling the underlying data generation. For a more detailed review, the reader is referred to Adhikari and Agrawal [2] and Dorffner [46].

2.2 Time-series analysis and forecasting

In the literature, time-series analysis can be understood as "*the systematic approach of developing mathematical models to describe, understand and model time-series data*" [186]. Identifying and fitting an appropriate model is vital as reliable analysis i) enables the transformation of raw data to insights, which informs decisions and propels change where required, and ii) provides forecasts from the identified inherent patterns and insights, which can be projected to inform the future [61].

2.2.1 Traditional statistical models for time-series processing

Statistical models have been popularly used for time-series analysis, particularly due to their flexibility and ability to model stationary processes. This section describes the notable and well-known statistical models employed for time-series

analysis. In general, these models estimate parameters[1] to make predictions.

2.2.1.1 Moving Average (MA)

A *moving average (MA) model* is a stationary process where previous forecast errors are used to make forecast for univariate time-series [236].

A moving average model of order[2] q (MA(q)) is given as:

$$y_t = \epsilon_t + \phi_1\epsilon_{t-1} + ... + \phi_q\epsilon_{t-q} = \epsilon_t + \sum_{j=1}^{q} \phi_j\epsilon_{t-j} \tag{2.2}$$

where ϵ_{t-1}, .. , ϵ_{t-q} are the white noise error terms, ϵ_t ~ $\epsilon_n(0, \sigma_\epsilon^2)$ and ϕ_1, .. , ϕ_q ($\phi_q \neq 0$) are the model parameters [236].

The moving average also known as 'running mean' or 'rolling average' is a simple technique that combines the mean of a fixed number of sequential output values from the model [87]. The average 'moves' through time as the newest observation is included and the latest disregarded. The assumption that the current value at any given time, t is a weighted sum of previous white noise underpins the MA process, and as such it is always a stationary process[3] [236].

Different types of moving average can be computed, for example, Simple Moving Average (as described above), Weighted Moving Average (WMA), where weightings are given to each data point [156], Exponential Weighted Moving Average (EWMA) derives the mean estimates by giving recent values more exponential weighting than distant values [37] and finally Cumulative Moving Average, where all the available data up to the current value is used to calculate the average.

Pilcher *et al.* [160] used a risk-adjusted EWMA (RAEWMA) chart to monitor outcomes of ICU patients. They showed that the RAEWMA chart had the ability to track fluctuations, particularly in contrast to the standard mortality ratio. They were confident that the RAEWMA was capable to be used as a routine mon-

[1] A parameter is considered to be a numerical or other measurable factor of a model.

[2] The order, q, denotes the number of past errors considered up to and including time t (the present time).

[3] A stationary process is a stochastic process whose joint probability distribution remains constant. For example, a white noise with mean, 0 and variance σ^2.

itoring tool in ICUs. In addition, Coulson *et al.* [39] highlighted a key advantage of their approach using MA over other control charts, namely interpretability.

Despite the observed improvement by Pilcher *et al.* [160] employing the MA, a simple technique which enables easy computation, such simplicity hinders proper handling of complex and evolving time-series *(reflecting real-world dynamics)*. Specifically, Pilcher *et al.* [160] proposed the use of the EMWA for ICUs with a few admissions each year, but given the enormous growth in ICU admissions such techniques pose little to no value. In particular, MA models have short-term memory and are limited when modelling and analysing time-series data, where the historical information provides considerable insights. In Wang and Zhang's [215] analysis, the adaptive MA model employed is very sensitive to small changes in the data, which could be attributed to the short-term nature of the memory mechanism. In addition, parameter estimation of MA models is known to be difficult because; the lagged error terms are not observable [236]. Finally, there are a number of problems associated with the moving average window; the two main drawbacks are i) introduction of temporal autocorrelation[1] which can amplify noise, obscuring learning, and ii) the historical information is quite 'rigid' as the window size is predetermined but also averaged and as a result, co-dependencies in time are not accounted for which is vital for identifying patterns and modelling the series.

2.2.1.2 Auto-regressive (AR)

An *auto-regressive (AR) model* is a process where variable forecasts are calculated as a linear combination of previous input values (that is, the given variable is regressed unto itself) with a random error [236].

An auto-regressive model of order[2] p (AR(p)) is given as:

$$y_t = \beta_1 y_{t-1} + ... + \beta_p y_{t-p} + \epsilon_t = \sum_{i=1}^{p} \beta_i y_{t-i} + \epsilon_t \tag{2.3}$$

where β_1, .. ,β_p are the parameters of the model and ϵ_t ~ $\epsilon_n(0, \sigma_\epsilon^2)$, is white noise

[1] Autocorrelation is the similarity between observations as a function of the time lag.
[2] The order, p, denotes the number of past outputs considered.

uncorrelated with y_s for each $s < t$ [236].

The model order (that is, the number of past observations) needs to be experimentally determined and the values of the coefficients in Equation 2.3 can be estimated with methods such as Maximum Likelihood Estimation and Ordinary Least Squares [215, 236].

AR processes possess a relatively "long" memory, where the model linearly combines decreasing coefficients with the previous values, to calculate the current value [4]. This characteristic is essential as historical information is significantly important for analysis, modelling and forecasting. In addition, these processes are based on previous observed values unlike MA models, which reduces the level of ambiguity for modelling and predicting.

Chaves and Pascual [27] presented seasonal AR to model and predict Cutaneous Leishmaniasis disease. Chaves and Pascual [28] later compared a number of models for early warning of neglected tropical diseases. In particular, a seasonal AR model is used as an Early Warning System (EWS) for prediction, where it outperformed the other models for the four forecast horizons (1, 3, 6 and 12 month(s)) evaluated. This superiority can be attributed to the "autoregressive treatment of seasonality, providing an adequate approximation to the asymptotic cyclical structure of a time-series" [28].

However, given that the independent variables are time-lagged values of the dependent variable, the assumption of uncorrelated error is easily violated. This behaviour may violate one of the key features of an EWS for diseases, as correlation between errors may hinder the ability to model covariates robustly [28]. An effective model should have the ability to adapt as data is presented, however, an AR model does not possess this ability as it has a relatively "long" memory; thus it is limited for modelling and prediction of short-term tasks [4]. The limitation with the MA and AR warrant the need for models that better capture the short-term and long-term temporal dependencies to more accurately model time-series data.

2.2.1.3 Auto-Regressive Moving Average (ARMA)

An *Auto-Regressive Moving Average (ARMA) model* combines the properties of an AR and MA model, enabling the ARMA to benefit from key advantages of the two processes. An ARMA model of order p and q (ARMA(p, q)) is given as

$$y_t = \epsilon_t + \sum_{i=1}^{p} \beta_i y_{t-i} + \sum_{j=1}^{q} \phi_j \epsilon_{t-j} \tag{2.4}$$

where $\beta_p \neq 0$, $\phi_q \neq 0$, β_i and ϕ_j are the parameters of the model, $\sigma_\epsilon^2 > 0$ and ϵ_t: $\epsilon_t \sim \epsilon_n(0, \sigma_\epsilon^2)$ [236].

The ARMA model possesses the stationarity conditions of an AR(p) model and invertibility[1] conditions of an MA(q) model. In particular, combining both AR and MA means the model subsequently requires fewer parameters to represent the data [155].

The *stationarity* condition is one of the bases of ARMA models [132], however, most time-series are composed of a non-stationary[2] trend series and a zero-mean stationary series [132]. Another condition that underpins ARMA models is *ergodicity*; which is where the autocovariance[3] decays rapidly to zero as the lag value increases, indicating that more accurate estimates can be obtained as the sample size increases [132]. However, this is not necessarily always true, especially for very complex series with structural breaks (for example, oil prices). These two main conditions do not replicate real-world dynamics and as a result ARMA models have limited modelling and predictive abilities.

Introducing a finite number of differencing (or other transformations) to remove non-stationarity is referred to as the so called Integrated ARMA, popularly known as the Auto-Regressive Integrated[4] Moving Average (ARIMA) [2,201,207]. ARIMA models are well suited to short-term prediction tasks, as they give more prominence to the recent past rather than the distant past [176]. The ARIMA's

[1] Invertibility condition is fulfilled when the summation of the model's parameters is less than infinity ∞.

[2] A non-stationary process is characterized by a continuous change of statistical properties over time, for example, random walks.

[3] Autocovariance estimates the covariance between a variable at a given point in time with itself at a lagged point in time.

[4] Integrated represents the differencing of observations to ensure stationarity.

ability to distinguish the signal from the noise enables the identification of a tangible correlation, thus improving prediction accuracy [189]. Researchers have employed ARIMA models for time-series analysis such as EWS to detect anomalies [227], to make early warning analysis by forecasting the process level network traffic [211], to forecast future mortality ratios [176] and an integration of ARIMA models into an online early warning system of power systems security and stability [100].

However, this method also has a number of limitations, Meyler *et al.* [102] presented evidence for the limited capability of ARIMA models to deal with volatility. They found that the ARIMA had difficulty modelling the Harmonised Index of Consumer Prices, particularly due to the ARIMA's poor handling of noisy data. In addition, they pointed out the model's 'backward nature' and poor ability to predict turning points, which further impedes its modelling and predictive abilities [102]. Stevenson [194] stated that the ARIMA's backward looking nature and price reverberations limit its ability to predict changes in the market direction following an extended cycle. More specifically, turning points[1] are where accurate forecasts are most desired and the ARIMA's limited ability with turning points renders it less effective as a modelling tool. In addition, stationarity, an essential process in ARIMA, might not be achieved as shown by Williamson and Hudson [223], due to large deviations between observations. William and Hudson [223] pointed out another limitation with ARIMAs, i.e. their inability to establish patterns and handle series with random or non-seasonal occurrences.

2.2.1.4 Box-Jenkins methodology

George Box and Gwilym Jenkins developed the *Box-Jenkins method* in 1970, to identify the best fit for the ARIMA/ARMA class of models. They proposed a three-stage iterative process to estimate the ARIMA [87, 236]. The three stages are *Identification*, *Estimation*, and *Diagnostic checking* [236].

- **Identification**: selecting the model specification, ARIMA (p, d, p), at this

[1] Turning points are defined by Chin and Miller [32] "as occurring when a swing in one direction ends and a swing in the other direction begins".

stage stationarity is ensured and differencing performed if necessary [163, 236].

- **Estimation**: inferring and estimating the model parameters, the aim at this stage is to minimise the error term [236].
- **Diagnostic checking**: evaluating the final model using available data, identifying areas of improvement and incorporating these improvements (*Note: if it failed, go back to Identification and repeat* [236].

Researchers such as Hikichi *et al.* [82] presented the Box-Jenkins methodology for certification forecasting and obtained satisfactory results when compared to logistic regression models, and provided new insights on the predicted certifications diffusion. Petrevska [157] applied the methodology for short-run predictions for tourism demand. In particular, they proposed the ARIMA(1,1,1) model for reliable forecasts of international tourism. Williamson and Weatherby Hudson [223] employed the Box-Jenkins method along with several statistical process control charts, and it proved to be an effective early warning tool to identify large unusual increases (or decreases) in disease reports.

Additionally, Commandeur and Koopman [36] stated this approach is intrinsically problematic as it does not replicate real series which are largely non-stationary[1], despite the inclusion of differencing. Despite Hikichi *et al.*'s [82] success with the Box-Jenkins method, they alluded to the method's lack of precision for long-term forecasting and implementation difficulty with short data series. Similarly, Petrevska [157] stated "*the model is not highly accurate, probably due to the presence of several structural breaks (that is, a sudden change in the series, proper capturing of which is essential for modelling) during the sample period. This means that the suggested model has some difficulties in producing forecasts with minimal errors, implying that different models could be tested.*". Finally, the Box-Jenkins methodology is based on the ARIMA class of models and as a result it is susceptible to the limitations presented in Section 2.2.1.3.

[1]Non-stationarity describes a process with a variable mean and variance.

2.2.1.5 Vector Models

The methods discussed above are univariate and thus an extension of these methods has been developed for *multivariate (that is, when there are two or more independent variables)* time-series processing. For a multivariate framework, the extension of ARMA models give Vector AutoRegressive (VAR) models, Vector Moving Average (VMA) models and Vector AutoRegressive Moving Average (VARMA) models.

The most common of these, the VAR model, is given as:

$$\mathbf{y_t} = \beta_\mathbf{1} y_{t-1} + ... + \beta_\mathbf{p} y_{t-p} + \epsilon_\mathbf{t}) \tag{2.5}$$

for $\mathbf{y_t} = (y_{1t}, .., y_{Kt})^T$ a K variate random process, where β_i are fixed (K $\times$ K) coefficients matrices and ϵ_t is the K-dimensional white noise with a non-singular covariance matrix $\sum u$ [75].

A VMA model is given as:

$$\mathbf{y_t} = \phi_\mathbf{1} y_{t-1} + ... + \phi_\mathbf{p} y_{t-p} + \epsilon_\mathbf{t} \tag{2.6}$$

for $y_t = (y_{1t}, ..., y_{Kt})^T$ a K variate random process, where ϕ_i are fixed (K $\times$ K) coefficients matrices and ϵ_t is the K-dimensional white noise with a non-singular covariance matrix $\sum u$ [57].

Authors such as [8, 11, 167] demonstrated the VAR model's efficacy and usefulness as an EWS. In particular, Lui *et al.* [125] showed that the VAR models with exogenous variables outperformed an ANN model. Mostafa [17] presented VARs and showed they provided more accurate economic forecasts than structural macroeconomic models.

Conversely, Aydin and Cavdar [8] used the VAR model to predict financial distress or stock market crashes before the 2007 financial crises and concluded that the VAR model underperformed compared to the alternative neural network model[1] used. Mostafa [17] admitted that despite the improvement in predictions obtained with VAR models, they are unable to fully describe the functioning mechanisms of the economy. In particular, for two of the variables, the insight

[1] Multilayered Feedforward Neural Network (MLFN) architecture

provided by the VAR model was the same as that provided from a random walk[1] model. These results, particularly those obtained by [17] are indicative of the unsuitability of VAR models for time-series modelling.

These models are based on MA/AR techniques, and thus they are prone to the limitations of MA, AR and ARIMA models presented in Sections 2.2.1.1, Section 2.2.1.2 and Section 2.2.1.3. As a result, they do not possess the qualities required to properly model and predict time-series data. In addition, Stock and Watson [195] pointed out further limitations such as inaccurate modelling of non-linearities, conditional heteroskedasticity[2] and parameter drifts. These limitations point to the need for more dynamic state based models that can model different aspects of the data, to properly handle volatility, non-linearities and identify underlying patterns.

2.2.1.6 Markov Switching models

Stationary models as seen above are good for modelling when the parameters *(mean and variance)* of the model remain constant. However, this is not usually the case for real-world data and Markov switching models enable the relaxation of this restriction. Markov switching models, also known as *Regime switching models*, are models "based on a mixture of parametric distributions whose probabilities depend on unobserved state variable(s)" [151]. Modelling data using different regimes enables better capturing of the underlying signal, as data properties, such as the mean and standard deviation, may vary temporally.

[1] The Random Walk is linked to the *Efficient Market Hypothesis*, which states "that financial markets are efficient, that prices already reflect all known information concerning a stock or other security, and that prices rapidly adjust to any new information" [202].

[2] Heteroskedasticity occurs when standard deviations of a predicted variable over time are non-constant.

The Markov switching model is given by

$$\begin{aligned} &f(y_t|\phi_1, \mathbf{F_{t-1}}) \text{ if } S_t = 1 \\ &f(y_t|\phi_2, \mathbf{F_{t-1}}) \text{ if } S_t = 2 \\ &. \\ &. \\ &. \\ &f(y_t|\phi_r, \mathbf{F_{t-1}}) \text{ if } S_t = r \end{aligned} \tag{2.7}$$

where ϕ_i are parameters of the model i and $\phi_i \neq \phi_j$ if $i \neq j$, S_t is an unobserved discrete state variable that determines the conditional distribution of Y_t and $F_{t-1} = f(X_t, X_{t-1}, .., X_{t-p}, Y_{t-1}, .., Y_{t-p})$ that is, the information known up to time, $t-1$ [151].

Goldfeld and Quandt [69] and Hamilton [77] notably introduced these models to the research community. The Markov switching model is well-noted for nonlinear time-series modelling and has the ability to distinguish time-series behaviour into different regimes and such behaviour enables the model to capture and represent complex dynamic patterns [111].

"A novel feature of the Markov switching model is a switching mechanism controlled by an unobservable state variable that follows a first-order Markov chain" [111]. In these models, S_t evolves over time as a discrete time, discrete space Markov process[1] and the transition probability of the state variable, S_t are dependent on the previous value of the state variable [151].

Lee and Chen [116] showed that Markov switching models are suitable for predicting exchange rates. Lu *et al.* [123] applied Markov switching models for a bioterrorism event detection task, which obtained a faster detection speed and higher detection sensitivity compared to the Serfling and ARMA models. They stated that the Markov switching models mitigated computational issues associated with temporal detection techniques. They later compared the three models used for predicting bioterrorism to disease outbreak detection where an extension

[1] A Markov process can be defined as a random process where predictions can be determined solely by the last previous state (that is, the last previous state has all the information required for prediction).

of the Markov model (with jumps and a filtering component) performed best with 23% - 328% higher detection sensitivity [124]. Song *et al.* [193] similarly applied the Markov model for wind forecasting and showed the worst performance of the Markov switching model was better than the other models *(AR model, Neural Network, and Bayesian structural break model)*. In particular, the different regimes of the Markov model enabled the forecasting of not only wind speed points but also wind speed intervals, which they stated is beneficial for operational planning.

Despite the success obtained with Markov switching models, Briggs and Sculpher [1] pointed out their inherent limitations; i) restrictive assumptions and ii) lack of memory. These limitations indicate that Markov models are particularly limited for more complex and dynamic series, as restrictive parameterisation limits its modelling abilities as the data 'evolves' and they do not have the necessary historical information to map and model the signal. In addition, Song *et al.* [193] stated large datasets may be required to identify optimal regimes, thus requiring increased computational complexity. As a result, state based models with appropriate memory mechanism (internally or externally) that can adequately and properly represent time-series and its features should be explored for reliable insights and predictions.

2.2.1.7 Review of Statistical Approaches

Statistical methods have been widely applied across several domains for time-series application (as presented in Section 2.2.1.1 - Section 2.2.1.6). They offer key advantages such as simplicity, computational efficacy and interpretability for time-series modelling and forecasting. However, despite the benefits offered by these models, they suffer from significant limitations such as difficulty processing complexities (present in real-world data) and inability to properly handle numerical complications (that arise with latent state variables[1]) [49].

In particular, the ARIMA class of models are unable to handle and process

[1]These are internal system states unobservable from the environment by the user and thus their behavioural characteristics must be informed from observational variables (inputs from and outputs to the environment), for example: hidden units in a neural network.

asymmetries[1], irregularities[2], and volatility[3] present in the data, due to their rigid linearity assumptions. Petrica *et al.* [158] showed these models are limited in mimicking intrinsic properties of time-series and as such limiting their predictive abilities. Finally, a number of these traditional methods are linear and consequently, they are unable to properly predict the structural shifts. *Note: to thoroughly evaluate the robustness and suitability of the models employed in this thesis, real-world data with the characteristics discussed in this section will be employed.*

Given the limitations of traditional statistical models, advances have been made to apply Machine Learning (ML) techniques for time-series analysis. Makridakis *et al.* [127] carried out experiments with the well-known M3 dataset and interestingly the results obtained by the author indicated that ML methods were not superior to statistical models. Makridakas *et al.* [127] thoroughly discussed what they deemed inadequacies of ML methods and concluded that the shift to ML methods had not been sufficiently justified (in comparison to statistical models).

Cerqueira *et al.* [26] addressed Makridakis et al's [127] criticisms of machine learning by highlighting a key bias: *sample size*. Although Makridakis et al [127] drew their findings from 1045 monthly observations to properly assess the models, the length of the series were 'extremely short' *(with an average, minimum and maximum of 118, 66, and 144, respectively)* [26]. They hypothesized and tested the claim that ML models generalize poorly on *'small datasets'*. Their results demonstrated that statistical methods performed better on small dataset, however, as the sample size grew, ML models performed better. They stated that because "ML models assume a functional form such that they are more flexible than statistical models, they are more prone to overfit." They explained that the poorer performance with ML models for a smaller dataset was due to the

[1]Asymmetric data has variable values at irregular or haphazard intervals, for example: medical datasets with controls where the number of healthy subjects significantly outnumbers the number of subjects with disease.

[2]There are irregularities present in a dataset due to deviations causing bias or skew, for example: spike increase in oil prices or drop in interest rates due to financial crisis.

[3]A dataset is referred to as volatile when there is a high rate of change in the variable values over a period leading to increased uncertainty of the future trend of the time-series, for example: stock prices, inflation rate, oil prices.

lack of sufficient representation in the data to enable appropriate mapping of the underlying signal. Although, Makridakis *et al.* [127] raised concerns about the superiority of ML models over statistical models, the work in [26] highlights that ML are suitable and should be explored (particularly with the vast wealth of data available today).

2.2.2 Neural Networks for time-series processing

Machine Learning is a branch of artificial intelligence and according to Shalev-Shwartz and Ben-David [180] is defined as "the automated detection of meaningful patterns in data". More specifically, machines are given data in order to facilitate 'learning' and perform given tasks. Learning can be supervised, unsupervised, semi-supervised or reinforced. Some example of these learning types are:

1. *Supervised*: observations: images of vehicles, and their labels: name of the vehicle (bus, car).
2. *Unsupervised*: using the characteristics of the input data to group and classify the data.
3. *Semi-supervised*: using a subset of labelled data to aid the identification of hate speech.
4. *Reinforced*: training using a reward and punishment mechanism.

ML is commonly viewed as an intersection between computer science and statistics. Statistically the aim is to identify "how meaningful inferences can be obtained from data?" while computationally the aim is to understand "how machines can be used for problem-solving?" [133]. The overarching goal is to develop "a computer program that learns from experience E with respect to some class of tasks T and performance measure P and its performance P at tasks T improves with experience E" [133].

Similar to statistical methods, ML methods aim to minimize a cost function in order to fit the data, where cost refers to some system error measure [127]. ML includes, but is not limited to, Artificial Neural Networks (ANNs), Case Based

Reasoning, Classification and Regression Trees, Rule Induction, Support Vector Machines, Gaussian Processes, Genetic Algorithms and Genetic Programming [190].

Specifically, Artificial Neural Networks (ANNs) have been shown to be an effective learning paradigm which provide state-of-the-art performance in a variety of tasks such as image recognition, financial forecasting, natural language processing and strategic game playing. These networks are based on and modelled (that is, they receive, process and output information) after the mechanisms of the brain *(a complex and powerful interconnected mechanism)*, using simplified numerical computations [180].

Kriesel [109] formally defines ANNs as "a sorted triple (N, V, w) with two sets N, V and a function w, where N is the set of neurons and V is a set $\{(i, j)|i, j \in N\}$ whose elements are called connections between neuron i and neuron j". "The function $w : V \rightarrow \mathbb{R}$ defines the weights, where $w((i, j))$, the weight of the connection between neuron i and neuron j, $w_{i,j}$" [109]. Information processing occurs as the network receives inputs which are multiplied by the appropriate weights to give the activation of the next layer and this is repeated for the succeeding layers (that is, the network forward propagates information) [62].

Particularly, for supervised learning with back-propagation, the network is initialised with random weights (and biases) which are used for the desired calculations. The network starts to learn once the error (the difference between the actual label and the network's prediction) along with the error surface gradient are propagated back (fed into the network) such that the weights (and biases) are adjusted in order to obtain network predictions closer to the actual label (thus producing a lower error) [9]. ANNs are powerful tools with the ability to i) approximate complex non-linear mappings and deal with 'noisy' signals, ii) predict without a priori distribution assumptions (which can obscure learning) and iii) adaptively incorporate new data [9]. In particular, these are properties which state-based memory mechanisms such as ARIMA and Regime Switching models do not possess.

These advantages are key for time-series processing, particularly for complex series. Having an adaptable model such as ANNs, that does not form prior assumptions, aids the input-output mapping and can separate the noise from the

signal to provide useful insights and inferences for informed decisions. ANN models can be broadly categorized into feed-forward neural networks and Recurrent Neural Networks (RNNs). The distinction is made based on their network topology i.e. how nodes within the network are interconnected and thus the direction in which information is allowed to flow [178].

2.2.3 Feed-forward Neural Networks (FFNNs)

Feed-forward Neural Networks are ANNs that utilise only forward connections (i.e. all information is received in a particular order, no feedback or context from layers within the network is provided) for numerical computation [178]. A number of FNN network variants have been developed and applied for time-series tasks, for example, Radial Basis Function (RBF) Networks, Perceptrons, Time-delay Neural Networks, Learning Vector Quantization and Probabilistic Neural Networks.

2.2.3.1 Multi-layer Perceptrons (MLPs)

Perceptrons, initially developed by Rosenblatt [164] in 1958 with a basic learning method, were the first type of neural networks built for binary classification. A later adaptation was with the Least Mean Square algorithm by Widrow and Hoff [221] in 1960. Werbos in 1974 developed a back-propagation algorithm for training multi-layer networks in his thesis and this was popularised by Rumelhart, Hinton, and Williams [166] in 1986. Back-propagation enabled the generation of meaningful internal representations to solve both linear and non-linear problems. More specifically, a perceptron has an input and output layer. Then MLPs which consist of an input layer, one or more hidden layer(s) and an output layer were later developed. A diagram of the simplest MLP is shown in Figure 2.1.

The equations for an MLP are given as follows:

$$\begin{aligned} S_{(j+1)t} &= f(W_j S_{jt}) \text{ for } j = (1, .., n) \\ y_t &= f(W_o S_{(j+1)t}) \end{aligned} \tag{2.8}$$

where n is the total number of hidden layers, W_j is the weight between the j^{th}

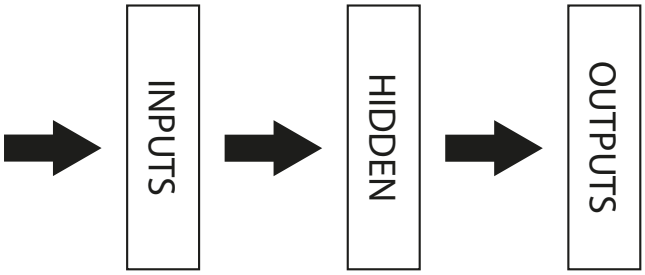


Figure 2.1: Multi-layer Perceptron

layer and the succeeding layer, S_{jt} is the output of the j^{th} layer at time t *(Note:* S_{1t} *is the input layer)* and f is the activation function [70].

MLPs provide a simple framework for time-series analysis with temporal information presented in a static manner. A number of researchers have applied MLPs for various time-series tasks. For example, to predict ICU outcome [50,63,86,184] and length of ICU stay [51,206]. Others such as Salmon *et al.* [174] explored MLPs with different processing techniques to detect land cover change; Dudek [48] proposed an MLP for forecasting probabilistic electricity price; and Pano-Azucena *et al.* [152] showed that MLPs outperformed Least-squares Support Vector Machines and Adaptive Neuro-Fuzzy Inference System when predicting chaotic time-series. Zhou *et al.* [239] proposed a hybrid which uses an ARIMA; a statistical approach and an ANN. Authors such as Frank [54] *et al.* and Murphy and Dieterich [136] have incorporated a sliding window with MLPs to better account for the time-series component within the data. Kohzadi *et al.* [106] particularly found that MLPs with a sliding window performed better and was able to capture a significant number of turning points in (non-linear and chaotic) datasets.

MLPs have shown potential for time-series modelling, for example as demonstrated in [127], however, they do not have 'memory' mechanisms and can not account for temporal features thus hindering effective learning and meaningful insights. Fawaz *et al.* [90] stated that MLPs are limited for time-series as they do not exhibit any spatial invariance and temporal information is lost. Similarly, Koskela *et al.* [108] pointed to their static nature and particularly for multi-step prediction MLPs could not perform as well as some other models *(Elman and Finite Impulse Response neural network)*. They attributed the better performance of the two networks to their temporal extension, which MLPs lack. Therefore,

identifying and exploring feed-forward approaches that employ memory is key for better representation of time-series and accurate forecasting.

2.2.3.2 Time-delay Neural Networks (TDNNs)

Time-delay Neural Networks (TDNNs) are networks that employ time delays to treat temporal and time-invariant signals. These time delays enable the input neurons to store 'history', as they have access to inputs at $T+1$ points, that is, the current value, x_t along with T delays $(x_{t-T}, x_{t-(T+1)}, ..)$ [98].

The equations for the TDNN is as follows:

$$\begin{aligned} h_t &= f_h(\mathbf{X}W_h) \\ y_t &= f_o(h_t W_o) \end{aligned} \tag{2.9}$$

where $\mathbf{X} = (x_{t-T}, .., x_t)$, W_h is the hidden weight matrix, W_o is the output weight matrix, h_t, y_t are the hidden and output vectors at time t [98] and f_h and f_o are the hidden and output activation functions.

Shao and Lin [181] used a TDNN to identify variables that cause out-of-control signals for a Multivariate Normal Process with variance shifts. They found that the TDNN outperformed and provided satisfactory results compared to other techniques employed *(ANN, Support Vector Machine and Multivariate Adaptive Regression Splines)*. TDNNs have particularly been widely applied for speech recognition tasks, for example by Peddinti *et al.* [154] and Sawai [177]. Zhang *et al.* [233] used a TDNN which improved pattern recognition ability of an electronic nose in discriminating four different spices. The TDNN outperformed the Discriminant Function Analysis and MLP. They attributed the superiority of the TDNN to its ability to capture the difference in time related patterns among sensors' responses.

However, Marques *et al.* [129] showed that the TDNN underperformed for response prediction and required more hidden neurons when compared to an RNN variant. This is possibly due to the lack of adequate temporal representation of the series, which is essential to obtain meaningful predictions for time-series. In particular, time-series data can be quite complex, requiring many feedback loops within the model architecture for effective processing to conserve temporal

dynamics, however, TDNNs do not possess this feature.

2.2.3.3 Limitations of Feed-Forward Neural Networks

Shabri and Samsudin [179] pointed out that ANNs suffer from local minima, parameter selection sensitivity and over-fitting [115, 217, 226, 231]. From Section 2.2.3.1 and Section 2.2.3.2, it is evident that feedforward networks do not possess the necessary feedback loops to appropriately map time-series data to capture the underlying signal. Natarajan and Ashok [139] further highlighted the need for advanced models such as recurrent neural networks for time-series processing as MLPs are not dynamic and as a result are inadequate to properly identify interactions between variables across time. Thus, they believe that neural networks with recurrent features possess the ability to properly utilise past observations and are more suitable methods for time-series processing.

2.2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are ANNs with recurrent connections to one or more layers within the network enabling the capturing and modelling of sequential data [173]. The hidden states in an RNN store information from previous states, thus creating a 'memory' for the network [173]. This 'memory' mechanism enables RNNs to properly handle time-series data [22, 24, 47, 81, 93].

2.2.4.1 Nonlinear Autoregressive models with eXogenous input Neural Network (NARX)

Nonlinear Autoregressive models with eXogenous input Neural Network (NARX) is a simple RNN variant employing lagged variable values to enable the modelling of non-stationary and non-linearity in time-series data [43]. Although, NARX networks do not possess state units, they utilise feedback of historical inputs and outputs in a shift-register manner such that time is mapped onto space.

The equation for the NARX is as follows:

$$y_{t+1} = f(y_t, ..., y_{t-d_y+1}, u_t, u_{t-1}, ..., u_{t-d_u+1}) \tag{2.10}$$

where y_t and u_t are the input and output vectors at time t and d_y and d_u are the output and input memory orders *(the amount of memory embedded)* [130].

NARX models have been employed by a number of authors, such as Maria and Barreto [130], specifically for predicting and early warning indication. Xie *et al.* [225] applied NARX for long-term prediction of a univariate time-series which constantly outperformed the TDNN. They attributed this to the feedback loops in the NARX which enabled it to handle complex time-series data. Asgari *et al.* [6] found that despite the simplicity of NARX models, they had the ability to model and predict 'unstable' behaviour within the data.

Nevertheless, DiPietro *et al.* [45] showed that the NARX model alongside other RNN variants contains the shortest paths of length such that historical information is lost rapidly and given the NARX's architecture with non-contiguous delays the paths are further shortened. This implies that the NARX network does not possess sufficient ability for long-term dependency problems and as a result other RNN variants should be explored.

2.2.4.2 Jordan Network

ARMA models as seen in Section 2.2.1.3 are a combination of both MA and AR techniques such that both the errors and outputs are used for modelling. *Jordan networks* are neural network extensions of the ARMA ideology forming a non-linear function. In particular, a Jordan network with a time window approximates a non-linear ARIMA model of past sequence elements and error estimates [46]. The Jordan network has feedback connections from the output layer to the input layer *(see Figure 2.2)* [46].

The equations for a Jordan network are:

$$\begin{aligned} S_t &= f_h(W_h I_t + U_h y_{t-1}) \\ y_t &= f_o(W_o S_t) \end{aligned} \tag{2.11}$$

where I_t is the input vector at time t, S_t is the state vector at time t, W and U are weight matrices, y_t is the output at time t and f represents the chosen activation functions [95].

The output feedback in the Jordan network enables the exploitation of past

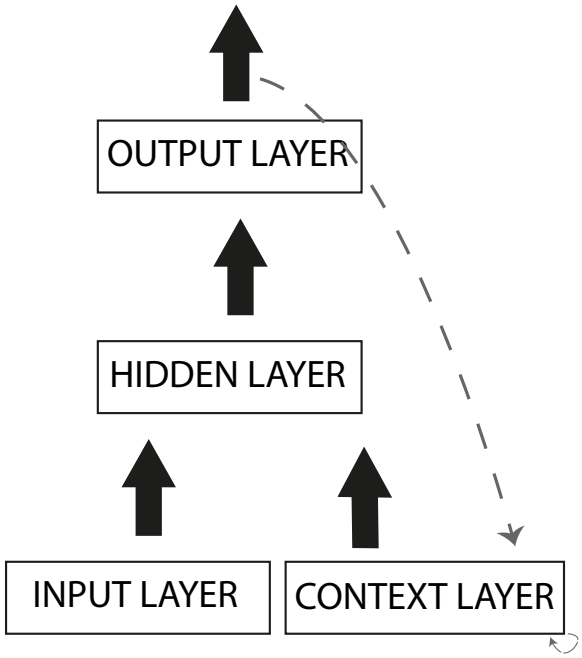


Figure 2.2: Jordan Network

information, particularly as the activations within the network are a function of all past estimates. Furthermore, the Jordan network was extended to incorporate self-recurrent loops for the units in the context layer *(also known as memory)* such that the units are connected to themselves through a weight that is less than 1 [46].

Authors such as Kasiran *et al.* [101], Wysocki and Ławryńczuk [224], Yasdi [229] and Song [192] have employed the Jordan network. Wysocki and Ławryńczuk [224] demonstrated that despite the simple architecture of the Jordan network, it possesses predictive abilities for model predictive control. Song [192] used the Jordan network along with recurrent constrained learning for non-linear time-series prediction, which achieved guaranteed weight convergence and network stability. The Jordan network boasts two key features; simplicity and non-linearity mapping, which are essential for volatile time-series and encourage computational efficacy. In addition, the Jordan network contains 'unlimited' historical information *(as the activations are a function of all past estimates)* [46]. Thus, enabling the network to exploit information beyond a limited time period (which most techniques presented thus far do not possess), enhancing its performance [46].

However, although this simplicity with access to 'unlimited' historical information is useful for the network, Dorffner [46] pointed out that the structure of the Jordan network leads to a loss of explicitness of information (as all past estimates are aggregated into one activation function). Wysocki and Ławryńczuk, [224] despite their success with the Jordan network, pointed out that 'unavoidable' inaccuracies are encountered and did not state whether these inaccuracies are peculiar to the Jordan network. However, Dorffner's [46] comments on the oversimplified history of the network explains why these inaccuracies arise. In addition, when sigmoid functions are used, the units saturate quickly and additional inputs have little effect [46]. This implies that the Jordan network has a theoretically rich memory mechanism but in practice, it is not dynamic and saturation can occur quickly with little to no further benefits realised [46]. As a result, Jordan networks are limited when modelling real-word dynamics and prediction of ordered sequence of actions (for example, robotic arm movements).

2.2.4.3 Simple Recurrent Network (SRN)

Elman [52] introduced an RNN that utilises its previous internal state as an additional input. This model is commonly known as either the *Simple Recurrent Network (SRN)* or *Elman Network*. State space models were presented earlier in Section 2.2.1.6 and this concept is extended to neural networks such as the SRN, as hidden states are fed back (or output states for Jordan networks) at the next time step. Each new observation is presented within the context of a compressed representation of previous input observations within the sequence [46]. More specifically, SRNs are computationally equivalent to a one-step Markov process as predictions can be made by the network's state *(regardless of how the state was reached)* such that all historical information required for predicting can be expressed by one state vector [46].

The SRN therefore extends MLP networks by incorporating these feedback loops from the hidden layer (generated in response to the current input stimuli) to the input layer (for use as inputs to the model at the next time step, along with any new input stimuli) as shown in Figure 2.3. The units that hold these historical hidden unit activations (from the hidden layer) on the input layer are

popularly known as state units and represent the memory mechanism in the SRN.

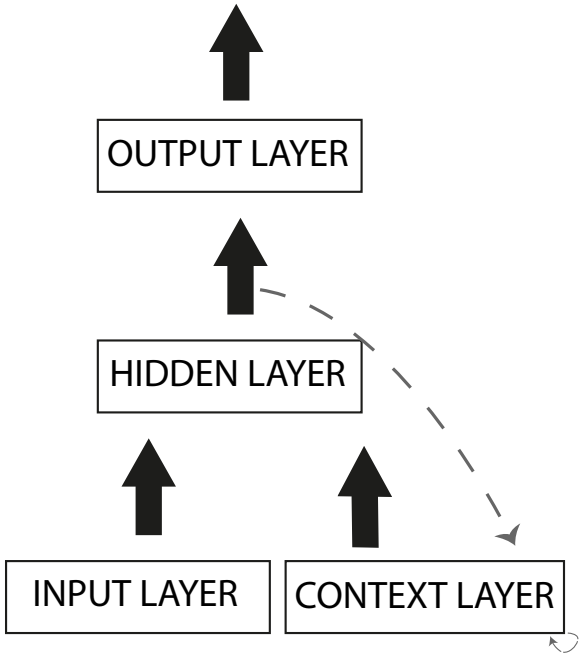


Figure 2.3: Simple Recurrent Network

The equations for an SRN are:

$$\begin{aligned} S_t &= f_h(W_h I_t + U_h S_{t-1}) \\ y_t &= f_o(W_o S_t) \end{aligned} \tag{2.12}$$

where I_t is the input vector at time t, S_t is the state vector at time t, W and U are weight matrices, y_t is the output at time t and f represents the chosen activation function [52].

The SRN has been used for time-series forecasting and early warning indication as shown by Sugiartawan and Hartati [196] and Wang *et al.* [213]. Sugiartawan and Hartati [196] demonstrated that the SRN had the ability to identify the predictive patterns of tourist visits. They also demonstrated that the SRN had a faster training time than the Jordan network, an RNN and a feedforward network [196]. Wang *et al.* [213] showed that the proposed model, an SRN with a Stochastic Time Effective Function, had the ability to model the financial market and predict stock market indices. Others have successfully implemented

hybrids incorporating the SRN for enhanced time-series analysis, for example, SRN–NARX neural networks for chaotic time-series modelling [5] and a hybrid RBF-Elman neural network for anomaly detection [205].

The SRN uses simple feedback loops to inform predictions, which admittedly simplifies implementation and requires less computational resource. However, although different from the Jordan network in terms of the type of feedback employed, the SRN suffers from the same limitation with simple feedback loops, loss of information explicitness, albeit to a lower degree [46]. Thus, the SRN is unable to fully represent historical information [46]. In addition, Dorffner [46] stated that SRNs can not deal with arbitrarily long history. The SRN's feedback loops favour the most recent state-based response over older historical responses and as a result the inherent simplicity causes the historical knowledge (past inputs and states) to decay rapidly [208]. Consequently, gradients concerning historical, yet important, input observations begin to vanish quickly (i.e. the vanishing gradient problem), thus limiting their prediction capability. Tepper *et al.* [200] stated this decay occurs as early as 5 to 10 discrete time steps. The SRNs, like the Jordan network, is a simple RNN and has been widely applied for time-series modelling, they are however, limited due to their lack of appropriate memory mechanism. Thus, a more sophisticated, yet simple paradigm is required.

2.2.4.4 Echo State Network (ESN)

A common issue with SRNs such as Elman and Jordan networks, is that they are trained with gradient descent algorithms. It has been proven that these algorithms suffer from the vanishing gradient descent problem, where the error vanishes as it gets propagated back in time [83]. In addition, the over-simplification of memory in these networks point to the need for other dynamic mechanisms. One such mechanism is the *Echo State Network*, developed by Jaeger [92] in 2001. These networks are similar to other RNNs in that they have an input, hidden and output layer. Specifically, the hidden layer within this network is a recurrent dynamic reservoir, the input sequence information is retained as activations rebound (that is, inputs to the reservoir are projected to a high dimensional feature space that allows the application of simpler training techniques) around the

recurrent units [94, 200].

The equation for an ESN are given as follows:

$$\begin{aligned} x_{t+1} &= f(W^{in}u_{t+1} + Wx_t + W^{back}y_n) \\ y_{t+1} &= f^{out}(W^{out}(u_{t+1}, x_{t+1}, y_t)) \end{aligned} \tag{2.13}$$

where f^{in} are the functions for the internal units, f^{out} are the functions for the output units and u, x, y are the input, internal and output activation vectors [92].

The reservoir is the core of the ESN and the connectivity in the reservoir is sparse and random, which encourages the development of individual dynamics [92]. More specifically, the reservoir must have the *Echo State Property (ESP)*. The ESP asserts that "the reservoir state should asymptotically depend only on the driving input signal *(the state is an echo of the input)*, while the influence of initial conditions should progressively vanish with time" [58]. ESNs are notably known to alleviate the training difficulties encountered in conventional RNNs [14, 165].

Tong *et al.* [204] used an ESN for a grammar task, they demonstrated that ESNs possess the ability to learn grammatical structure, particularly without the need of specialized learned representations akin to SRNs. They further stated that such behaviour is compensated by the ESN's reservoir that enables the network to capture some statistical regularities of the inputs [204]. This finding is particularly interesting as these specialized representations have been elevated in the literature as essential for learning, but ESNs appear to refute this notion [204]. Ruffing and Venayagamoorthy [165] found that ESNs were suitable models for Solar Irradiance prediction. Li *et al.* [117] used an ESN in a Bayesian framework to predict chaotic time-series. Others applied ESN variants such as Double-Reservoir ESN which adopts two reservoirs to handle multi-regime time-series [238] and Support Vector ESN where the "kernel trick" is replaced with a "reservoir trick" *(that is, linear support vector regression is performed in the high-dimension "reservoir" state space)* [185]. Basterrech [14] sought to boost the ESN's performance by 'weakly' initialising an ESN and incorporating an L2-Boost technique and found this added no computational effort and obtained improved performance for 'weakly' initialised ESNs.

ESNs appear to alleviate the problem associated with over-simplification of historical information prevalent with the SRN and Jordan network. However, as seen from Bianchi *et al.*'s [19] experiments despite obtaining the best performance, ESNs are of greater computational complexity. Tong *et al.* [204] quantified this increased complexity, stating for an SRN with 70 hidden units, an ESN required 300 hidden units (4 times as many parameters). For more complex series, using an ESN which requires over 4x as many trainable parameters is computationally intensive, thus mitigating the acclaimed benefits of the ESN. This computational complexity reduces the suitability of ESNs for effective time-series modelling. In addition, Ruffing and Venayagamoorthy [165] pointed out that ESNs had a noisier memory, leading to more errors. Although, the ESN's dynamics have shown some success with time-series prediction, the increased number of parameters and noisy memory lead to increased complexity and inaccuracies. Tepper *et al.* [200] also showed that the ESN was unable to fully learn a complex context free grammar induction task. The network showed preference for early commitment to one of the paths through the grammar rather than being able to maintain robust representations of both paths.

2.2.4.5 Long-Short Term Memory (LSTM)

LSTMs were introduced to tackle the vanishing gradient problem associated with the back-propagation learning process by incorporating a gating mechanism that uses feedback connections and controls feedback weights enabling gradients to flow unchanged [198]. Their structure additionally provides a means to tackle the issues that arise from over-simplification associated with SRN and Jordan network. LSTMs incorporate a memory cell, multiplicative input and output gate and forget gate to protect the memory contents (such that information can be removed and added to the cell state) [103].

The gating equations for LSTMs are shown below:

$$\begin{aligned} i_t &= f_{in}(W_i x_t + U_i h_{t-1} + b_i) \\ l_t &= f_{in}(W_l x_t + U_l h_{t-1} + b_l) \\ o_t &= f_{in}(W_o x_t + U_o h_{t-1} + b_o) \end{aligned} \tag{2.14}$$

and the cell-memory/input block equations are:

$$\begin{aligned} \hat{c}_t &= f(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= l_t \odot c_{t-1} + i_t \odot \hat{c}_t \\ h_t &= o_t \odot f(c_t) \end{aligned} \tag{2.15}$$

where x_t is the input vector (sequence), c_t is the memory "state", i_t, l_t and o_t are the three gate signal vectors, h_t is the hidden activation, f represents the chosen activation function, b represents the biases and W and U are the model weights [103].

Authors such as Pham *et al.* [159], Gopalswam *et al.* [72], Sak *et al.* [171], Kim and Kang [104] and Ge *et al.* [59] have successfully implemented the LSTM for time-series analysis in a number of domains, and it is believed to dynamically model signals and mitigate the vanishing gradient problem. Sak *et al.* [171] applied LSTMs for a large vocabulary speech recognition task, in particular, then endowed the LSTM with thousands of context dependent states to encourage more flexibility. This extension outperformed Deep Neural Networks, and they showed that the LSTM quickly converged, obtaining an outstanding performance. Gopalswamy *et al.* [72] demonstrated that LSTMs are effective mechanisms for extracting patterns from patient data and modelling long-term dependencies in the data. In particular, the LSTM performed better than Support Vector Machine and k-Nearest Neighbours. Their results are indicative that an increased level of model complexity aids proper representation. LSTM variations have also been applied for time-series processing such as, bidirectional LSTM [241] and attention based LSTM [99]. The LSTM and its variants are currently recognised as the state-of-the-art RNN, however, they are not without problems.

Chen *et al.* [30] argued that LSTMs are not adaptable to new changes, as seen when they applied it for oil prices prediction. In addition, Le *et al.* [113] pointed out that although LSTMs provide accurate forecasts at specific points, they should be combined with other models to obtain improved performance for long-term prediction. Danihelka *et al.* [42] stated that the LSTM's limited ability with representation makes it a poor candidate for learning due to its lack of memory mechanism. Additionally, Yu *et al.* [232] presented a systematic re-

view on different LSTM cells and pointed to the need for 'external' memory to strengthen their memory capacity. This indicates that although an increased level of complexity can aid adequate representation, another important component for effective time-series processing and modelling is an appropriate memory mechanism. Sainath *et al.* [170] pointed out the modelling flaws with the LSTM such that the 'true' underlying signal is not fully unveiled and proposed higher-level modelling. Finally, LSTMs are ad-hoc techniques with high dimensionality and researchers such as Tepper *et al.* [200] and Jozefowicz *et al.* [97] debate whether there are more optimal architectures for prediction tasks. LSTMs can be built as deep networks such that they consist of multiple hidden layers. However, these models are overly complex which i) overfit on the data and ii) have intractable training times (without state-of-the art GPUs).

2.2.4.6 Limitations of current recurrent neural networks for time-series processing

Section 2.2.4.1 - Section 2.2.4.5 presented the most notable RNNs discussed in the literature for time-series processing. These models have enabled better capturing of time-invariant features to process, model and predict time-series compared to traditional statistical methods and feed-forward ANNs. However, it is clear that these methods are not ideal for time-series processing due to their identified limitations. The review of these methodologies indicate the need for two key attributes: i) mechanisms that incorporate memory and ii) the 'right' level of complexity, in particular, current methods lack the right balance between these two key attributes.

2.2.4.7 Long-term dependencies

The vanishing gradient problem has serious implications when using RNNs for learning long-term dependencies. Bengio *et al.* [15] described the simplest form of long-term dependencies in an RNN as "storing information about the initial input values for an arbitrary duration". This is essential when the output at any given time step t is dependent on the inputs from the distant past. Consider subject-auxiliary verb agreement problems in Natural Language Processing, e.g. *The dog*

that chased the cat who was chasing the mouse that had eaten the cheese is hungry. The network must remember the correct subject over a given number of time-steps to correctly predict the appropriate auxiliary verb. More specifically, Bengio *et al.* [15] demonstrated that the gradient descent learning has difficulty learning long-term dependencies [16]. This difficulty is largely attributed to vanishing or exploding[1] gradients which impede learning [173].

Bengio *et al.* [15] demonstrated using a simple recurrent network candidate solution the inefficiencies and issues encountered when learning a long-term dependency task [15]. In particular, their experiments showed that the network candidate could not robustly latch onto information about the input and long-term input/output dependencies, and mapping could not be achieved with gradient descent [15]. In order to explain the difficulty with using gradient descent for long-term dependencies, they used a system with additive inputs to explain the issues encountered, namely sensitivity to noise or vanishing gradients [15]. They stated that there is a trade-off between efficient learning with gradient descent and information latching for extended periods [15]. Thus, they proposed other systems and optimization methods (such as, Multi-Grid Random Search, Time-Weighted Pseudo-Newton Optimization) which provided better learning and had long plateaus.

Bengio *et al.* [15] conclusively encouraged the implementation and use of other techniques, giving the difficulties faced with gradient descent for long-term dependency. Interestingly, the network candidate solution they presented to demonstrate and prove this, utilises a similar structure to the SRN and Jordan network. A review of these methods in Section 2.2.4.2 and Section 2.2.4.3 supports the claims presented by [15], particularly as historical information is aggregated, and information explicitness lost, thus preventing appropriate information latching. However, they do not address whether this issue persists for RNNs which utilise gradient descent but employ mechanisms to tackle the vanishing gradient problem or mechanisms that encourage better information latching.

[1] Exploding gradients occur when large error gradients accumulate leading to large parameter updates during training.

2.3 Multi-recurrent Neural Network (MRN)

Ulbricht [208] explored this notion that complexity and memory are key attributes for effective time-series processing and introduced the **Multi-recurrent Networks**. The MRN integrates features of both the SRN and Jordan network, along with input layer memory and variable layer-link and self-link recurrency. These properties enhanced the model's complexity sufficiently to encourage better processing without the added over-complexity observed in LSTMs or Gated Recurrent Units and as such positioning the MRN as a worthy competitor to be further explored for time-series modelling. In addition, the MRN allows for different and flexible representation of past information, unlike the Jordan, SRN, ESN and LSTM networks which have a 'rigid' representation. More recently, Tepper *et al.* [200] showed that this more sophisticated class of SRNs, is better able to capture the latent signal in time-series data and found that the MRN dynamics improved learning and achieved better accuracy (compared to the SRN, NARX and ESN networks). Other researchers such as [20, 149, 150, 183] have investigated the potential of the MRN for time-series modelling, they applied and assessed the MRN and found that it offered comparative predictive performance to current state-of-the-art models. The MRN specifically appears to employ an extensive historical context within a simple recurrent framework, which as identified from the limitations of current methods, is essential for time-series modelling. A detail description of this method is provided in Chapter 3.

Giles *et al.* [66] highlighted the impact of embedded memory within RNNs for long-term dependency tasks using three types of RNNs *(Globally connected RNNs*[1]*, Locally recurrent networks*[2] *and NARX networks)* with varying parameters and memory orders. Interestingly, the MRN mimics the embedding of memory, particularly that implemented for the globally connected RNNs. Through the use of 'rigid' memory banks, where past information is retained for longer, the MRN repeatedly encodes historical information that changes very slowly over time and informs parameter updates during learning. For long-term dependency tasks, Bengio *et al.* [15] presented three basic requirements for any parametric

[1] where feedback connections are obtained from the state vector to the hidden layer

[2] where feedback connections are only allowed from neurons to themselves and nodes are connected in a feed forward manner

dynamical system, namely information latching, noise resistance and trainable parameters. In particular, Bengio *et al.* [15] stated systems must possess the ability to robustly latch onto information, such that information stored internally about the initial values of the input cannot be deleted easily (by events that are unrelated to the classification criterion (and assumably for a regression task)). More specifically, the combination of memory banks in the MRN enables the network to preserve information from the distant past, particularly as the rigid memory is loosely sensitive to new information, thus encouraging the network to latch onto information for longer. Secondly, the different representations of the information provided by the MRN's memory banks enable the network to effectively filter the noise. Finally, Bengio *et al.* [15] presented two components for the third requirement, input processing and state variables latching. The memory mechanism of the MRN not only allows information latching but also state latching, as the memory contain different and flexible representation of both past inputs and previous states. The MRN provides a suitable structure for long-term dependency tasks as it meets these requirements defined by [15] and in addition demonstrates the importance of appropriate memory mechanism for complex time-series tasks.

2.4 Summary of current state-of-the-art

In summary, current techniques have been thoroughly assessed across various domains for time-series modelling, however, they are bound by key inherent limitations. More specifically, traditional statistical models are unable to properly handle structural breaks and shifts in the signal. The ARIMA/ARMA class of models, although they are simple techniques with easy calculations and interpretation, they are unable to account for temporal correlations and their memory is limited. In addition, Markov switching models possess strong abilities to deal with non-linearities and distinguish the behaviour of the series into different regimes, which enables the capturing of dynamic patterns. However, these models do not possess memory and have restrictive assumptions.

Furthermore, feedforward networks are simple paradigms, however, they do not possess adequate memory to map temporal dependencies. Therefore, these

models are not dynamic or robust to map time-series data. Simple recurrent networks such as SRNs and Jordan networks incorporate memory, however, these networks aggregate history into one activation function. This leads to a loss of information explicitness, thus they have limited abilities to reflect real-world dynamics. More complex models such as ESNs and LSTMs alleviate the over-simplification of historical information in the SRN and Jordan network, however, the noisy memory of ESNs and inadequate memory in LSTMs, limit their potentials for time-series modelling.

More specifically, the MRN has been shown to outperform other models within its class and worthy of further exploration due to its simpler yet powerful memory mechanism. In particular, it alleviates the complexity issues encountered with models such as the ESN and the LSTM requiring significantly fewer number of parameters. It also employs a dynamic memory mechanism comprised of various memory bank types forming sluggish state spaces such that explicitness of historical information is not lost as in the SRN and Jordan network.

2.5 Conclusion

This chapter has presented a critical review on the current notable modelling techniques employed for time-series forecasting. Based on this review, it is concluded that current state-of-the-art statistical and ANNs have shown tremendous success for numerous tasks. However, despite this success, they are limited for time-series processing. In particular, the limitations of statistical methods were presented in Section 2.2.1.7, for feed-forward ANNs in Section 2.2.3.3 and for RNNs in Section 2.2.4.6. Their Achilles heel for statistical methods appears to be the lack of adequate memory and representation whereas whilst current RNNs overcome this, they suffer from inherent gradient-based limitations of the back-propagation learning algorithm. In addition, the current RNN models are either over-simplified such as the SRN or overly complex like the ESN such that the temporal and spatial features are not accurately modelled. Thus, they suffer from a rapid loss of information as time increases and inputs become more distant. These limitations motivate the need for more suitable and simplified models that adequately capture, represent and preserve temporal information overtime.

The MRN, first developed in 1994 [208] and then discovered again in 2010 [20] appears to offer an appropriate balance between i) a sophisticated memory mechanism, that is dynamically sensitive to both historical and recent events, and ii) computational simplicity, such that the MRN retains the use of well-understood algorithms such as back-propagation through time [200]. Therefore, this research seeks to identify whether MRNs from the simple recurrent class of networks can be optimized and competitively applied for a range of time-series forecasting tasks when evaluated against current state-of-the-art models whilst mitigating current limitations and obtaining improved prediction accuracies.

Chapter 3

The Multi-recurrent Neural Network

This chapter presents the proposed methodology for time-series forecasting explored in this thesis. The Multi-recurrent Neural Network will be the focus of this research and in particular, how it can be extended and optimised to overcome some of its existing limitations. The chapter begins with an in-depth explanation of the MRN's structure, particularly its memory banks and ratios. In addition, the training process of the MRN is described along with the following forecasting methodologies; sliding window, forecast horizon and ensemble approach. The chapter concludes with a summary of the current research till date regarding the MRN and its suitability for time-series applications, and data used in this thesis is presented.

3.1 Multi-recurrent Neural Network (MRN)

Given the computational and architectural limitations of popular models used for time-series forecasting, it is clear that more sophisticated models are required for enhanced usability and performance. Ulbricht [208] proposed the Multi-recurrent Neural Network (MRN), a class of RNN *(using classical neurons[1])* that utilises

[1] A standard or *classical* artificial neuron is one that has a set of weighted inputs, calculates the net input and passes through a linear or non-linear activation function to determine its output. This will be the standard model of a neuron used throughout the thesis. However,

a combination of repeated memory banks of varying strengths. These memory banks consist of feedback activations from either the input, hidden or output layers and the memory banks themselves. The memory banks enable the MRN to store historical information in different forms which are then used for prediction. They are configured with layer-link ratios *(which determine the proportion of the new layer activations that are stored)* and self-link ratios*(which determine the proportion of the previous memory retained)*. The ratio between these links determines whether we have a **sluggish state-based memory bank** *(where the historical information changes slowly)*, **rapid memory bank** *(where the historical information changes often, as new information is presented to the network)* or **stable memory bank** *(where historical and current information are retained at the same rate)*. It is this state-based memory configuration that forms the context of the network for each new input received. Such combination of different degrees of recurrency enables rigid learning *(of the task structure)* and flexible learning *(of spatio-temporal features as new information is presented)* that effectively captures both variant and invariant properties of the time-series [208]. As shown in Equation 3.1 - Equation 3.3, the memory composition is determined by the ratios in a memory bank type *(either input, hidden or output)*, and these ratios along with the total number of memory banks for each type are key hyper-parameters. The MRN's dynamic memory structure enables it to both 'forget' and 'retain' knowledge, which is the catalyst for enhanced performance [208]. In addition, the MRN's memory structure enables the formation of a more sophisticated history, which is particularly required when solving long-term dependency problems [46, 200].

3.1.1 Architecture of the MRN

The architecture of the MRN as described by Ulbricht [208] is illustrated in Figure 3.1 showing the flow of information through the network layers, employing the following feedback *(which determine the memory bank composition at time, t, fed to the hidden layer at time, $t+1$)*:

note there are alternative neuronal models that are analoguous to those of biological neurons *(for example, spiking neurons)* - these are considered beyond the scope of this thesis.

1. Layer-recurrency: the output of the network's layer(s) copied to a respective memory bank at time, t. The layer-level recurrency comprises the:

   - *Input layer recurrency*: the input layer's output at time, t is copied to the relevant memory bank.
   - *Hidden layer recurrency*: the hidden layer's output at time, t is copied to the relevant memory bank.
   - *Output layer recurrency*: the output layer's output at time, t is copied to the relevant memory bank.

2. Self-recurrency: the outputs of the memory banks *(referred to as either state units or context units)* at time t, that is retained.

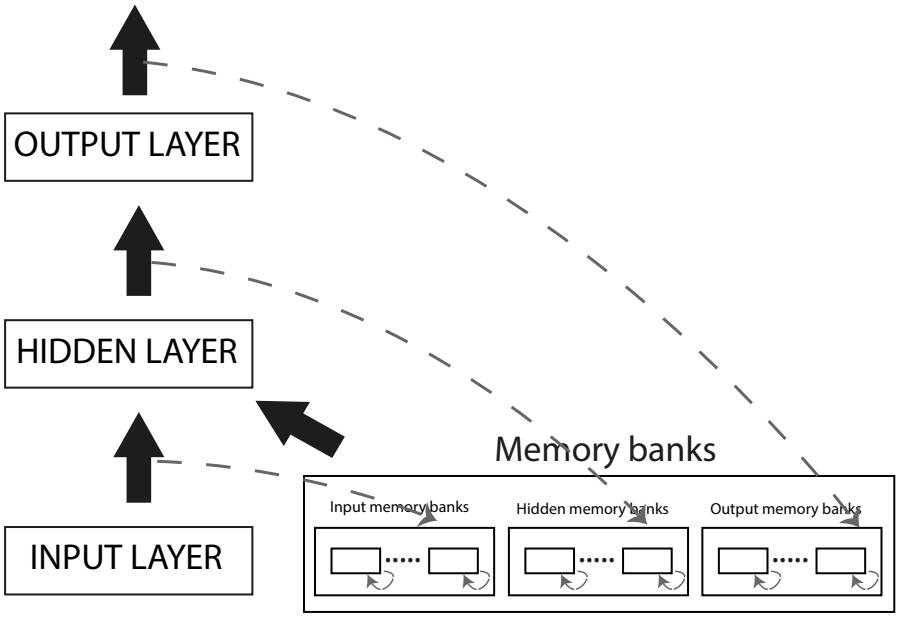


Figure 3.1: The architecture of the MRN

3.1.1.1 Memory architecture

The memory bank compositions for each layer are derived using the calculated layer-link and self-link ratios. A key assumption for the ratios in a given memory bank type *(input, hidden, output)* is that the ratio values are between 0 and 1. Therefore, for each memory bank type, the layer-link ratios are calculated (using the allocated number of memory banks for that type, n_M) as: $\frac{i}{n_M}$ for $i = n_M, ..., 2, 1$. The self-link ratios are then calculated as: $1 - \frac{i}{n_M}$ for $i = n_M, ..., 2, 1$. For example, a network with 4 output memory banks gives the following layer-link ratios: $\frac{4}{4}(1), \frac{3}{4}(0.75), \frac{2}{4}(0.5), \frac{1}{4}(0.25)$ and the following self-link ratios: $1 - \frac{4}{4}(0), 1 - \frac{3}{4}(0.25), 1 - \frac{2}{4}(0.5), 1 - \frac{1}{4}(0.75)$.

Figure 3.2 illustrates the derivation of the 4 output memory banks at time t to be employed at time $t + 1$. As shown, the layer-link ratios determine how much of the output values of the output layer at time t are copied and the self-link ratios determine how much of the respective memory bank values at time t are retained.

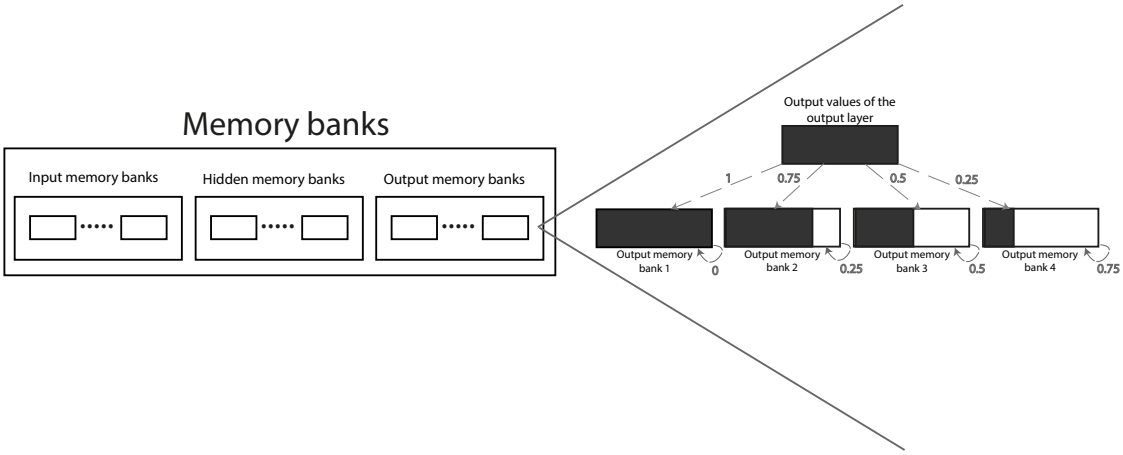


Figure 3.2: The derivation of the memory banks

The 4 memory banks all hold different properties *(of the past inputs)*. Output memory bank 1 holds only recent information and is the most flexible output memory bank, whereas output memory bank 2 holds mostly recent information and some historical information from the distance past. Output memory bank 3 holds an equal proportion of recent information and information from the distance past. Output memory bank 4 is the most rigid memory bank, it holds the greatest

proportion of information from the distance past and some recent information. Note: the MRN has only three types of memory banks (input, hidden and output), and a memory bank type can have a different number of memory banks from the other types. For example, an MRN may have an input memory bank with 3 copies; a hidden memory bank with 5 copies and an output memory bank with 10 copies *(denoted as [3, 5, 10])*.

3.1.1.2 Memory bank configuration

The number of memory banks for each type determines the memory configuration, Ulbricht [208] carried out experiments with a limited number of memory bank combinations, employing either 0, 1 or 4 memory bank(s) for each memory bank type. The memory bank combinations she employed were: [4, 4, 4], [4, 0, 0], [1, 0, 0], [0, 4, 4], [0, 0, 4], [0, 0, 1], [0, 4, 0], [0, 1, 0], [0, 0, 0]. For her experiments, the MRN employing 4 memory banks for each memory bank type ([4, 4, 4]) obtained the best performance.

In this research, the MRN memory bank configuration is explored further and more memory bank combinations are employed to understand the impact on the MRN's performance. More specifically, due to computational limitations and the large search space, the memory order *(that is the maximum number of memory banks employed for each memory bank type)* is 4 *(similar to Ulbricht [208])*. For any given set of hyper-parameters, each memory bank type is allowed to employ either 0, 2, 3 or 4 memory bank(s). To calculate the total number of possible combinations, the memory order for each memory type *(that is the input, the hidden and the output)* is multiplied, thus giving $4*4*4 = 64$. The first possible memory bank combination employs no memory bank for each memory bank type [0, 0, 0], this represents a standard recurrent network and is thus excluded. Therefore, a total of 63 combinations (models) are trained for a given set of hyper-parameters. *Note: the memory bank types do not employ just 1 memory bank as this mimics the SRN. More specifically, the calculated layer-link ratio is 1 and the self-link ratio is 0, which is identical to the SRN and has no flexibility.*

3.1.1.3 Brain dynamics

Interestingly, the MRN mimics the memory mechanism of the human brain *(which utilises different types of memories)* by combining multiple types of memory banks, which is what its superiority is attributed to [200]. The Brain Institute at The University of Queensland highlighted the two main types of memories in the brain; short-term and long-term memories [89]. Both types of memories are essential for the proper working of the brain and body [40]. The study of Henry Molaison, a patient who had his hippocampus surgically removed to treat his epilepsy, furthered highlighted that the brain utilised multiple types of memory [89]. More specifically, short-term memories enable the remembering of a small amount of information for a limited period of time while the long-term memory which is broadly categorised into two, explicit *(conscious)* and implicit *(unconscious)* is a vast store of knowledge and a record of prior events [89].

3.1.2 Training in the MRN

Training in the MRN occurs by forward pass and back-propagation through time. During the forward pass, the inputs are fed to the network (through the input layer), which along with the memory *(randomly initialised)* and the input layer biases produce an input layer output which is then fed to the hidden layer. The outputs of the hidden layer are then fed along with the hidden layer biases to the output layer to produce a network output. The layer outputs along with the current memory are copied using the ratios and the memory is then updated.

After a given number of forward passes, back-propagation (of the error) through time occurs. During this phase, the learnable parameters *(the weights and biases)* in the network are updated in line with the signal of the data to produce more accurate outputs. The two-stage process above describes the learning in the network, the forward pass and back-propagation are repeated until a desired error is reached or for a given number of epochs.

3.1.2.1 Forward Pass

During the forward pass, net outputs for each layer are calculated *(using the weights and biases)* which are then passed through an activation function *(where*

applicable) as information moves through the network *(Note: weights are randomly initialised in a given range)* [34]. The MRN's forward pass functions are given as follows. Given the input units at time $t-1$, I_{t-1} and the input memory at time $t-1$, M_{t-1_i}, the input memory at time t, M_{t_i} is calculated as:

$$M_{t_i} = (\frac{1}{n_i} \times I_{t-1}) + ((1 - \frac{1}{n_i}) \times M_{t-1_i}) \tag{3.1}$$

where n_i is the total number of input memory banks. The hidden units at time $t-1$, H_{t-1} and the hidden memory at time $t-1$, M_{t-1_h} are given. Thus, the hidden memory at time t, M_{t_h} is calculated as:

$$M_{t_h} = (\frac{1}{n_h} \times H_{t-1}) + ((1 - \frac{1}{n_h}) \times M_{t-1_h}) \tag{3.2}$$

where n_h is the total number of hidden memory banks. The output units at time $t-1$, O_{t-1} and the output memory at time $t-1$, M_{t-1_o} are given. Thus, the output memory at time t, M_{t_o} is calculated as:

$$M_{t_o} = (\frac{1}{n_o} \times O_{t-1}) + ((1 - \frac{1}{n_o}) \times M_{t-1_o}) \tag{3.3}$$

where n_o is the total number of output memory banks. Given the respective weights from the input and memory layer to the hidden layer, W_{i_h}, $W_{M_{ih}}$, $W_{M_{hh}}$, $W_{M_{oh}}$ and the input layer biases, b_i, the net hidden outputs at time t are calculated as:

$$\hat{H}_t = \sum W_{ih} I_t + \sum W_{M_{ih}} M_{t_i} + \sum W_{M_{hh}} M_{t_h} + \sum W_{M_{oh}} M_{t_o} + b_i \tag{3.4}$$

The outputs of the hidden layer at time t, H_t, are derived using the formula below:

$$H_t = f(\hat{H}_t) \tag{3.5}$$

where f is the chosen activation function. *(Note: for the experiments in this thesis sigmoid;* $\frac{1}{1+e^{-x}}$ *is employed).* The hidden outputs at time t, H_t, the hidden to output layer weights, W_{ho} and the hidden layer biases, b_h are given. Thus, the

outputs for the output layer at time t, O_t are calculated as:

$$O_t = \sum W_{ho} H_t + b_h \tag{3.6}$$

3.1.2.2 Back-propagation Through Time (BPTT)

The MRN was trained with *Back-propagation Through Time (BPTT)*, (an extension of the classical Back-propagation (BP)) to account for time-dependency as the network is *'unfolded'* through time *(as presented by [219])*. In particular, BP and BPTT have the same underlying algorithm, however, BPTT particularly 'unfolds' the network over a given number of time steps. This way the copy of the network which is fed forward (through the memory) during the forward pass is accounted for during the error calculation. The main aim of this phase is to adapt the learnable parameters temporally (given the interdependence between the network outputs over time).

For experiments, a decaying learning rate is used and recalculated after each epoch (using the total number of epochs, n_e and the current learning rate, L) as:

$$L = L - \frac{L}{n_e} \tag{3.7}$$

The error is defined using the Root-Mean Squared Error (RMSE) (given the total number of patterns, n) as:

$$E = \sqrt{\sum_{i=1}^{n} (O_i - A_i)^2} \tag{3.8}$$

To determine how much the error changes with respect to the learnable parameters (the weights, W and biases, b), partial derivatives, ($\frac{\partial E}{\partial W}$ and $\frac{\partial E}{\partial b}$) are employed. The MRN's back pass functions are as follows. Given the error at time t, E_t, the network outputs at time t, O_t and the actual output at time t, A_t, the output layer deltas at time t, D_{O_t} are calculated as:

$$D_{O_t} = \frac{\partial E}{\partial O} = (O_t - A_t) \tag{3.9}$$

The hidden layer to output layer weights, W_{ho} and hidden layer biases, b_h are updated as follows:

$$\begin{aligned} W_{ho} &= D_{O_t} * \frac{\partial O}{\partial W_{ho}} = D_{O_t} * H_t \\ b_h &= D_{O_t} * \frac{\partial O}{\partial b_h} = D_{O_t} * 1 = D_{O_t} \end{aligned} \tag{3.10}$$

The hidden layer delta at time t, D_{H_t} is calculated as:

$$\begin{aligned} D_{H_t} &= \frac{\partial E}{\partial O}\frac{\partial O}{\partial H}\frac{\partial H}{\partial \hat{H}} + Mem_{t+1} \\ &= (O_t - A_t) * W_{ho} * f^{-1}(\hat{H}) + Mem_{t+1} \end{aligned} \tag{3.11}$$

where f^{-1} is the inverse of the chosen activation function and Mem_{t+1} is the sum of the hidden layer deltas from the succeeding time step multiplied by the respective memory weights. The input layer to hidden layer weights, W_{ih}, input layer biases, b_i and the memory weights $W_{M_{kh}}$ are updated as follows:

$$\begin{aligned} W_{ih} &= D_{H_t} * \frac{\partial \hat{H}}{\partial W_{ih}} = D_{H_t} * I_t \\ b_i &= D_{H_t} * \frac{\partial \hat{H}}{\partial b_i} = D_{H_t} * 1 = D_{H_t} \\ W_{M_{kh}} &= D_{H_t} * \frac{\partial \hat{H}}{\partial W_{M_{kh}}} = D_{H_t} * M_{t_k} \text{ for } k = i, h, o \end{aligned} \tag{3.12}$$

where i, h and o mean input, hidden and output.

3.2 Forecasting Methodology

The forecasting methodology describes how the data is represented, manipulated and presented to the MRN during training and how forecasts are derived.

3.2.1 Sliding Window

A number of researchers such as [85, 110, 122, 169, 175] have demonstrated improved learning and enhanced performance when employing a sliding window

with varying ANNs. For example, Krystalakos *et al.* [110] demonstrated that the models that used a sliding window technique performed better on multi-state devices for Online Energy Disaggregation. In addition, Liu *et al.* [122] employed a Bayesian-Gaussian neural network driven by sliding window data for on-line modelling of a hydraulic turbine system. They showed that the sliding window enabled the quick capturing of changes in the HTS characteristics and the model obtained a high prediction accuracy.

Given the success with and input processing benefits offered by a sliding window, it is employed with the MRN. A sliding window technique with a shift factor of 1 is used to generate the input sequences for the model to process. These sequences, also known as temporal input windows, consist of a given number of previous input/output observation pairs along with the current observation. They are used as inputs to the model at any given time step to obtain a prediction. The input patterns in a window are presented sequentially to the model. The length of the input window is problem dependent and empirically established for each of the individual models evaluated. Figure 3.3 presents an example of the sliding window technique with 50 input observations, a window size of 3 and a shift factor of 1.

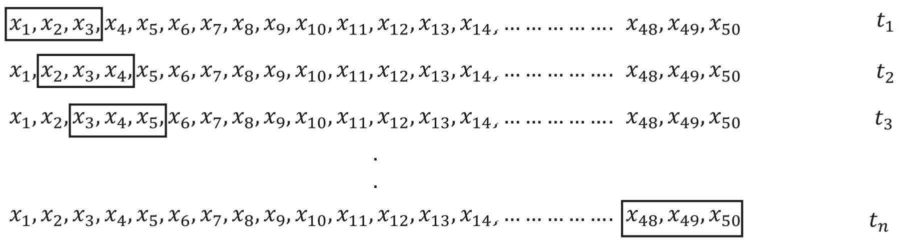


Figure 3.3: Example of the sliding window technique

When the data is generated in this way, the models are treated as finite memory models due to the fixed length of the window and the state memory is reset at the beginning of each window. For each time step, a sequence is fed to the network as inputs and the network reads the inputs sequentially (i.e. one pattern at a time) *(see Algorithm 1)*. The calculations for the feed forward weight updates in Equation 3.10 and Equation 3.12 are modified for information processing with

a sliding window as follows:

$$
\begin{aligned}
S_{j(j+1)_t} &= \sum_{1}^{n_s}(-L * (D_{(j+1)_t} * j_t)) + (F * S_{j(j+1)_{t-1}}) \\
s_j &= \sum_{1}^{n_s}(-L * D_{(j+1)_t}) + (F * s_{j_{t-1}}) \\
W_{j(j+1)} &= \frac{1}{n_s} * S_{j(j+1)_t} \\
b_j &= \frac{1}{n_s} * s_{j_t}
\end{aligned}
\tag{3.13}
$$

where $j, k = i, h, o$ refer to the input, hidden and output layers *(Note: $j+1$ indicates the succeeding layer to j)*, F is the momentum term[1] and n_s is the window size. The calculation for the memory weight updates in Equation 3.12 is modified for information processing with a sliding window as follows:

$$
\begin{aligned}
S_{M_{kh_t}} &= \sum_{1}^{n_s}(-L * (D_{H_t} * M_{t_k})) + (F * S_{M_{kh_{t-1}}}) \\
W_{M_{kh}} &= \frac{1}{n_s} * S_{M_{kh_t}}
\end{aligned}
\tag{3.14}
$$

where $k = i, h, o$ mean input, hidden and output.

3.2.2 Forecast Horizon

The desired output for any given time step is determined by the forecast horizon, that is, how many time steps ahead a prediction is made for the given target variable. For example, for 10 input-output observations, inputs; $\mathbf{X} = (x_1, x_2, ..., x_{10})$ outputs; $\mathbf{y} = (y_1, y_2, ..., y_{10})$, for pattern x_1, predicting for a forecast horizon of 1, 3 and 5 *(step(s) ahead)* gives the following output; y_2, y_4 and y_6.

[1] The momentum determines how much previous changes influence the current direction of change for the parameters.

Algorithm 1 Forward and Back propagation in the MRN with sliding window

procedure WINDOWS(x, y, s, h) ▷ s: window size, h: forecast horizon
  <initialise empty arrays to store input-output windows>
  for <all the input-output pairs> **do**
    <store the next s input-output pairs with horizon, h>
    <shift by a factor of 1 and repeat till the last s input-output pairs>
  end for
  return X, Y ▷ Input-Output windows
end procedure
procedure LEARNING(X, Y, l_r, m) ▷ l_r: learning rate, m: momentum
  <initialise random starting weights, W and biases, b>
  for <all the input windows> **do**
    <initialise empty arrays to store calculations>
    <initialise weight change vectors>
    for <for each window> **do**
      <do a forward pass>
      <store calculations>
    end for
    for <for each window> **do**
      <back propagation>
      <store calculations>
    end for
    <update learnable parameters using l_r and m>
  end for
  return W, b ▷ The learnt weights and biases
end procedure

3.2.3 Ensemble approach

Ensemble approaches have been widely applied to neural networks to enhance their performance. The most common of these is cross-validation, Lin *et al.* [121] applied cross-validation for the Radial Basis Function neural network which improved generalisation performance. Similarly, Zhang and Wu [237] discussed the importance of cross validation and employed it to prevent over-fitting. Other popular ensemble approaches include; Bootstrap Aggregation (bagging), Boosting, Stacked Generalization (stacking), Model Averaging and Weighted Average.

In particular, researchers such as Opitz *et al.* [148] and Refaeilzadeh *et al.* [162]

stated that the model averaging ensemble is preferred over other techniques such as cross validation, bagging or boosting which are computational expensive with lengthy training times and large variance. In addition, Binner *et al.* [20] showed that using model averaging provides more reliable results, is computationally inexpensive and improves performance. Model forecasts of the MRN are determined by a model averaging ensemble approach, where a given number of models are trained and the average of the predictions are used as the final prediction and to assess performance. *(Note: the number of models is empirically established).*

3.2.4 Methodological constraints

The MRN, similar to other RNNs, employs iterative training procedures and as such requires retraining/refitting as new data becomes available which is time-consuming unlike statistical methods such as Linear Regression that employ one-shot learning, which is fast and efficient.

3.3 Computational requirement

Williams and Zipser [222] showed that to store m-dimensional input and n-dimensional activity of a network over, h time steps *(epochs)* and with the number of target values, $t \leqslant nh$ requires $(m+n)h$ space therefore BPTT has a space complexity of $O((m+n)h)$. The time complexity of the BPTT as presented by [222] is $O(w_U H + w_A h)$ where w_U is the number of non-zero weights between units and w_A is the number of adjustable parameters.

Employing a sliding window further increases the number of calculations by the number of input windows (m_w) multiplied by the input window size (m_s). This increase in the number of calculations has a negligible effect on time complexity as the increase is by a constant, however, the space complexity increases. The MRN with a sliding window employing the BPTT algorithm, therefore has a space complexity of $O(((m_w * m_s) + n)h)$ and time complexity of $O(w_U H + w_A h)$ [222].

3.4 Time-series application with the MRN

To date, very few researchers have employed the MRN for time-series forecasting. Ulbricht [208] undertook a comparative study and exploited various memory bank combinations to identify which would enhance performance. The results from the study demonstrated that the MRN which combined all types of memory feedback *(input, hidden and output layer memories)* with self-recurrent feedback loops of different strengths obtained the best results. More specifically, Ulbricht showed that this memory combination enabled the formation of short-term memories with different properties. Thus, extending the neural network properties which often only have long-term memories and this enhanced performance [208]. In particular, Ulbricht showed that the hidden-layer feedback enabled the network to utilise information from the previous days [208]. This, coupled with output and input feedback, provided superior performance compared to other memory bank combinations for traffic forecasting [208]. The outcomes of the study were used for the installation of a highway check point tool.

In the study, 9 neural networks with different memory combinations were employed, these are:

1. The MRN with 4 input, 4 hidden and 4 output memories *(4I 4H 4O)*
2. An RNN with 4 input layer memories *(4I)*
3. An RNN with 1 input layer memory *(1I)*
4. An RNN with 4 hidden and 4 output memories *(4H 4O)*
5. An RNN with 4 output layer memories *(4O)*
6. An RNN with 1 output layer memory *(1O)*
7. An RNN with 4 hidden layer memories *(4H)*
8. An RNN with 1 hidden layer memory *(1H)*
9. Feedforward network *(without memory, None)*

Figure 3.4 shows the RMSE of the models presented in her paper, as shown the MRN *(which employed 4 input, 4 hidden and 4 output memory banks)* performed best. Ulbricht's works highlighted the importance of different feedback loops for time-series modelling and forecasting. In particular, this work demonstrated the versatility of neural networks (as they are not only suitable for long-term tasks but also for short-term tasks). More specifically, the MRN's memory is adaptable and as such further justifying the exploration of this network.

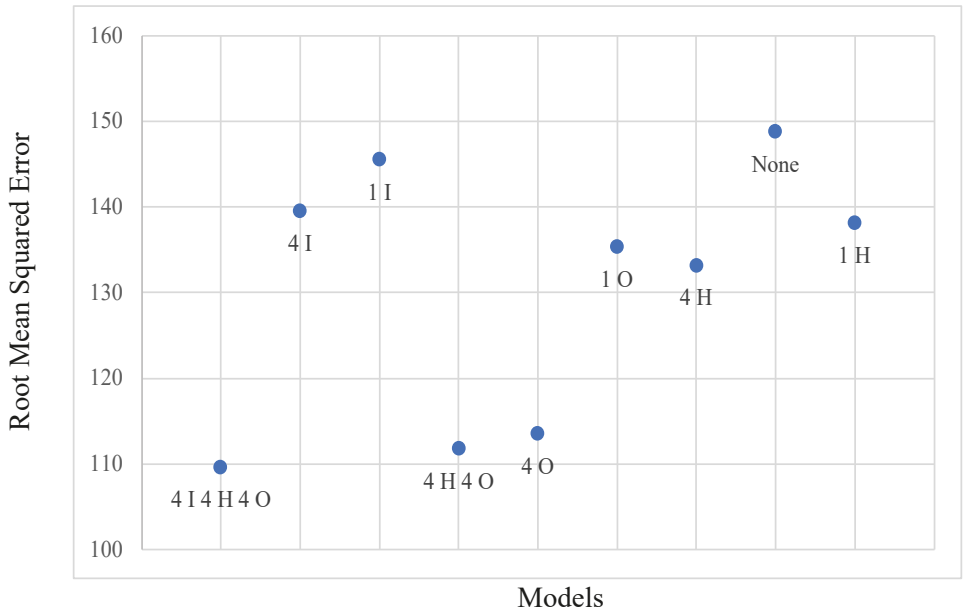


Figure 3.4: Root Mean Squared Error of the presented models in [208]

Over a decade later, Binner *et al.* [20] applied the MRN for inflation forecasting. Tepper *et al.* [200] and Shertil [183] for grammar induction through training the MRN to perform the next word prediction task. Tepper *et al.* [200] adapted the MRN and utilised just hidden and output feedback with noise injection in the input layer. In particular, Tepper *et al.* [200] demonstrated how the underlying representations formed by the MRN were superior to those formed by the SRN, NARX and ESN for a complex grammar induction task. Thus, providing evidence that the MRN is able to more robustly model complex time-series data and improve prediction accuracy. Figure 3.5 shows the performance of the MRN, SRN and ESN on randomly generated long sequences; the MRN consistently outperformed all the other networks [200]. *Note: Unique set refers to those test*

cases that did not appear in the training set and indicates the pure generalisation ability, as seen the MRN is far superior to the other models.

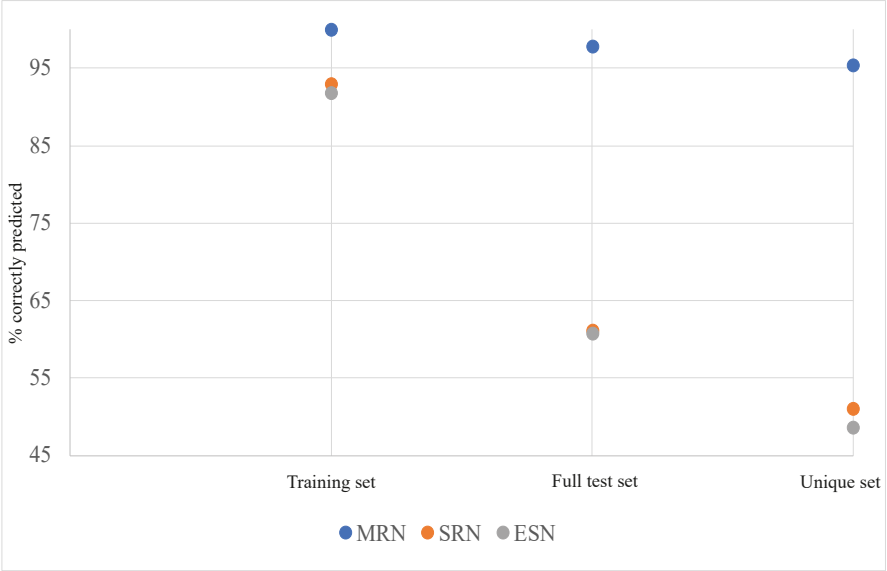


Figure 3.5: Network performance (% correctly predicted training and test) in [200]

Despite the success achieved with the MRN, there has been a very limited number of studies and therefore applications with the MRN. Today, there are increased data complexity as a result of structural shifts and variance, which require more dynamic and robust models for better modelling. Ulbricht's findings using (varying strengths of memory banks) over two decades ago are well-suited to model such complexity, as this type of graded feedback enables the network to account for many aspects of the data, thus, simplifying the task of capturing data properties. This research seeks to i) explore and exploit the MRN further and ii) mitigate the inherent limitations of the MRN by extending its architecture for enhanced time-series processing whilst maintaining simplicity to learn a problem. For the purpose of this research, the MRN as presented in this chapter is also referred to as the *standard MRN*.

3.5 Data

In this section, the datasets used for the experiments undertaken in this research are presented. Four experiments are carried out, namely Business cycle prediction, Oil price prediction, M3 competition prediction and Covid-19 forecasting.

3.5.1 Business cycle data

Giusto and Piger [67] used four monthly coincident series collected from February 1967 to July 2013 to 'nowcast' business cycle phases *(periods of expansion or recession)*. Understanding and identifying business cycle points is important for economic planning. These four series used commonly known as '*the big four economic indicators*' are:

1. **Non-farm payroll employment**: this is a monthly statistic which represents workers in the United States, specifically excluding proprietors, private household employees, unpaid volunteers, farm employees, and the unincorporated self-employed. [145].

2. **Industrial production index**: this "measures the real output for all facilities located in the United States manufacturing, mining, and electric and gas utilities (excluding those in U.S. territories)" [146].

3. **Real personal income excluding transfer receipts**: this "measure is deflated by the implicit price deflator for personal consumption expenditures." [147].

4. **Real manufacturing and trade sales**: this measures "the combined changes in business sales and end-of-month inventories for domestic retail trade, wholesale trade and manufacturers' activities". [209].

The series are obtained from the Federal Reserve Economic Data website: http://fred.stlouisfed.org/. The out-sample data is between October 1976 to July 2013, a period with five complete National Bureau of Economic Research (NBER) recession phases.

3.5.1.1 Data Preprocessing

The data consists of continuous-valued variables only and each variable was transformed using basic standardisation (based on the mean and standard deviation) of the variable values in the training set. This ensures all variables are of the same scale and transformed into a normal variable to ensure minimal measurement error. Muralidharan [135] showed that training the network with standardised data yields better results.

Three datasets for the real-time classification of NBER points were created. The first dataset comprises the four growth variables used in [67], the second comprises the change of direction[1] (COD) of each growth variable (thus, there are 4 input variables) and the third comprises the growth variables and their change of direction (thus, there are 8 input variables). (Note: the change of direction is not standardised).

3.5.2 Oil price data

Approximately, 99 million barrels of petroleum were consumed daily in 2018 [210]. Crude oil is therefore arguably the world's most important commodity, it is particularly key in ensuring nations are able to meet their energy demands [13]. Furthermore, oil prices have a massive effect on the price of other commodities and heavily influence macroeconomic projections for gross domestic product and inflation [56, 76].

Five (5) key indicators were used to predict monthly crude oil prices from July 1969 to March 2015, consisting of 549 observations (see Table. 3.1). Monthly data was preferred compared to daily data due to less noise and as such less obscurity of the signal, which is essential for learning in the network.

3.5.2.1 Data Pre-processing

The change of direction as described in Section 3.5.1.1 was calculated for the five variables in Table 3.1 and included as features for the oil price prediction

[1] This is the difference in magnitude between any given variable at time, t and the same variable at time, t - 1 which informs the direction (-1: negative change, 0: no change, 1: positive change).

Table 3.1: Key indicator variables and data source.

Variable	Data source
Real WTI Crude Oil Price, Month-End Prices	Energy Information Administration, BLS MacroTrends Data Download
Gold Fixing Price 10:30 A.M. (London time) in London Bullion Market, based in U.S. Dollars Average daily price	DataStream database
US External Trade: Goods, Deflator/Unit Value of Imports NADJ	
US Unemployment Rate SADJ	
US average weekly hours - Total Private Nonfarm VOLA	

task. The data was divided into training and testing sets, the training data accounted for 75% of the data and the remaining 25% of the data was out-sample testing. In particular, the in-sample data (pre-financial crisis) was from July 1969 to December 2005 and the out-sample data was from January 2006 to February 2015. The standardised variables and their change of direction are used as inputs at all time-steps for modelling, thus there are 10 input variables.

3.5.3 M3 competition data

The M3 competition data is a widely used data benchmark for assessing the ability of time-series forecasting models. The M3-Competition data consists of 3003 time-series which include data from various sectors (micro, industry, macro, finance, demographic and other) and different time intervals (yearly, quarterly, monthly and other).

Ten monthly series are randomly selected from the following sectors: Micro, Macro & Industry, the length of these series are between 126 and 144, and the last 18 observations for each series are used as out-sample observations to assess the performance. The 10 series are *N1807*, *N1908*, *N1918*, *N2012*, *N2144*, *N2150*, *N2158*, *N2159*, *N2516* and *N2521*.

3.5.3.1 Data Pre-processing

The data is standardised as explained in Section 3.5.1.1 using the mean and standard deviation. The change of direction or other pre-processing techniques were not utilised for comparability with published techniques.

3.5.4 Covid-19 data

Two series are collected to predict the number of Covid-19 *(confirmed and death)* cases in the USA, the experimental setup is similar to that in [182]. The data is obtained from Centers for Disease Control and Prevention, U.S. Department of Health and Human Services[1]. Daily data of the confirmed cases from the 7^{th} of February 2020 to the 7^{th} of July 2020 and 26^{th} of Feb to the 7^{th} of July 2020 for the death cases were used for this experimentation. The data was divided into training and testing sets, training data accounted for 80%[2] of the data and the remaining 20% was out-sample testing.

3.5.4.1 Data Pre-processing

The two series were transformed using *MinMaxScaler*[3], such that the data is scaled between 0 and 1 on the training set [182]. Transformation of the data is important to deal with data inconsistency and variance [182]. The training set is scaled as follows:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.15}$$

where x_{min} and x_{max} are the minimum and maximum values of the feature in the training set and x is the feature value.

3.6 Conclusion

Limitations of current techniques as presented in Chapter 2 support the need to identify and exploit other paradigms that can effectively model and predict time-

[1] https://www.cdc.gov/

[2] 80% of which specially to train and 20% to validate

[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

series data. Subsequently, the Multi-recurrent Neural Network was presented in this chapter positioned within a forecasting methodology that employs a fixed-length sliding window, forecast horizons of variable length and an ensemble-based approach for time-series processing.

In this chapter, the MRN's architecture was thoroughly presented, particularly how its memory structure enables better information processing. It was argued that employing multiple memory banks with various connection strengths provides a more powerful means of latching onto historical information *(that is, information from the distant past and near past)* at various levels of gradation to inform optimal decision-making. The data for the applications in this thesis are also presented along with the pre-processing techniques employed.

The MRN was introduced over two decades ago and remains largely under utilised, particularly due to the shift to more complex mechanisms, which are believed to be more suitable. However, these complex mechanisms present key limitations, which the MRN addresses. It appears to be a strong candidate for alleviating the issues associated with traditional gradient descent based models *(vanishing gradients)* whilst maintaining the architectural simplicity of Elman's SRN and the Jordan network. This research will therefore investigate the MRN further with a view to introducing a number of innovations that will overcome inherent limitations. The next chapter will evaluate the MRN on a range of real-world time-series problems to better understand its processing abilities and limitations, which will inform further extensions and optimisations of this particular approach.

Chapter 4

The Multi-recurrent Network: a comparative analysis

This chapter seeks to investigate the suitability of the MRN for time-series processing and its comparative performance. More specifically, the computational adequacy of the MRN for complex time-series forecasting will be evaluated and compared to the current state-of-the-art forecasting models. The MRN is applied to the following real-world problems from the time-series domain: Business cycle prediction, Oil price prediction, M3 Competition prediction and Covid-19 forecasting *(of varying complexity)*, to assess its performance.

4.1 Background

ANNs have been applied and are well-suited to modelling and predicting across various problems in the time-series domain, for example, financial forecasting [149], natural language processing [161], biological data mining [126], image analysis [3], anomaly detection [228], strategic game playing [188] and disease detection [143]. In particular, their superiority is attributed to their non-linearity and universal function approximation abilities as summarised in literature review presented in Chapter 2.

A number of authors have specifically applied a range of RNNs *(due to their memory mechanism)*, such as the SRN and current state-of-the-art, the LSTM,

presenting their superiority over traditional statistical and feed-forward ANN models. However, despite much success with these techniques, a number of modelling and prediction limitations have been identified. For example, the SRN and Jordan network overly simplify the memory mechanism, favouring the most recent state-based response over any historical response. Thus, resulting in vanishing gradients of historical yet important input observations and limiting their predictive ability. LSTMs on the other hand, employ a complex gating mechanism to alleviate the vanishing gradient problem, however, this complex gating mechanism leads to a significant increase in the number of trainable parameters. In addition, LSTMs do not possess a comprehensive memory mechanism and as such the model is unable to appropriately latch onto important information. Thus, limiting their ability to accurately model the underlying signal and their performance.

Researchers such as Dorffner [46], Tepper *et al.* [200], Danihelka *et al.* [42], Yu *et al.* [232] and Ulbricht [208] highlighted the importance of memory mechanisms, particularly as they enable the capturing of temporal dependencies. Such capturing is crucial for identifying latent interactions between relevant feature variables, thereby providing insights into behaviour dynamics [74]. The evaluation of current models for time-series processing highlighted the need for and importance of dynamic state-based models with appropriate memory mechanisms (as summarised in Chapter 2). Therefore, in this chapter, the performance of the MRN is evaluated on real data (of progressively increasing complexity) in different domains. The performance of the MRN is then compared to current state-of-the-art models to identify whether its internal memory mechanism and simple architecture offer any superiority and benefits.

4.2 Results and Analysis

The MRN is applied to the four time-series tasks; *Business cycle prediction, Oil price prediction, M3 Competition prediction* and *Covid-19 forecasting* using the datasets presented in Section 3.5. The performance of the MRN is compared to current state-of-the-art models, and the results are presented, analysed and discussed in this section.

4.2.1 Business cycle prediction

Giusto and Piger [67] proposed the Learning Vector Quantisation[1] model (LVQ) to identify turning points in real time[2]. In addition, they presented evidence with Monte Carlo to prove the possible benefits of an LVQ over a misspecified parametric statistical model. They demonstrated that the model had competitive performance over common alternatives. Thereafter, they implemented the LVQ and published their findings in [68].

The performance of the MRN is compared to the LVQ model, and also with the LSTM and Support Vector Machine (SVM)[3]. The Matthew Correlation Coefficient (MCC) is used to assess the performance of the models.

The Matthew Correlation Coefficient (MCC) is calculated as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.1}$$

where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

MRN: The MRN employed was described in Chapter 3; experimentation with a memory order of 4 (for the memory bank combinations) and window sizes of [20, 40, 60, 80] was undertaken to identify the best MRN.

LSTM: The LSTM used a dropout regularisation technique to reduce over-fitting. For experimentation, dropout values in [0, 0.9], units of [10, 15, 20, 25, 40] and two optimisers (Adam and Stochastic Gradient Descent (SGD)) were employed to obtain the best model.

SVM: For experimentation with the SVM, three types of kernels (sigmoid, polynomial and RBF) and a combination of hyper-parameters (coefficient, degree and

[1] Learning Vector Quantisation "is an adaptive learning algorithm in which the locations of the codebook vectors are established through adjustments of decreasing magnitude" [67].

[2] Due to its computationally simple structure and its superiority compared to a mis-specified parametric Bayesian Classifier.

[3] Support Vector Machine creates a hyper-plane in an n-dimensional space that groups data points distinctly: https://scikit-learn.org/stable/modules/svm.html

gamma for the relevant kernel) were employed to obtain the best SVM.

Giusto and Piger [67] nowcasted U.S. business cycle turning points (i.e. the forecast horizon was 't + 0', as the turning points occur) and this is particularly significant as the NBER's Business Cycle Dating Committee historically confirm turning points *after* they occur. In addition, different pre-processing techniques are applied to the model to identify the impact on the models. *Note: the LVQ models in [67] obtained a highest MCC of 0.48.*

In this section, the prediction task is taken a step further as predictions are made for 't + 1' steps ahead (that is a month ahead). Table 4.1 shows the results for all the experiments carried out and as seen the MRN performed better than the other models including the LVQ proposed by [67] for the three datasets.

Table 4.1: Comparison of models for the NBER turning points prediction task

Model	Optimser	MCC		
		Growth	COD	Growth & COD
LVQ	Clustering	0.578	0.558	0.574
LSTM	Adam	0.715	0.68	0.645
	SGD[1]	0.605	0.63	0.655
SVM	Sigmoid	0.13	0.635	0.627
	Poly[2]	0.57	0.416	0.49
	RBF[3]	0.396	0.395	0.276
MRN	BPTT[4]	**0.787**	**0.79**	**0.799**

In particular, using the change of direction of the growth variable significantly improved the performance of the SVM with a sigmoid kernel whilst other models had relatively the same or slightly worse performance. This improvement for the SVM can be attributed to its effective handling and processing of binary variables for which it is historically acclaimed. Discretisation with SVMs encourages faster, easier and more effective identification of decision boundaries required to separate

[1] Back-propagation through time (see Chapter 3.1.2.2)
[2] Stochastic Gradient Descent
[3] Polynomial kernel
[4] Radial Basis Function kernel

the classes. Similarly, with the third dataset (the growth variables and the change of direction), most models had a similar or slightly worse performance. Overall, the MRN provides the best results for this task. This experiment is particularly indicative that different models require different data transformations depending on the internal processing of the architecture to provide meaningful insight and harness the model's dynamics.

Table 4.2: Memory bank & window size for the best MRN model

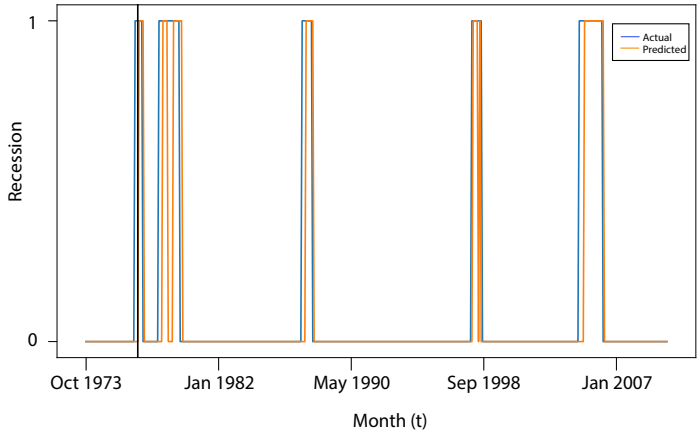
Dataset	Memory banks *([input, hidden, output])*	Window size	Trainable Parameters
Growth	[4, 2, 0]	80	1241
COD	[2, 3, 2]	80	1521
Growth & COD	[4, 0, 3]	80	901

The MRN performed best with the third dataset, Growth & COD variables, which is indicative of the usefulness of discretisation for time-series processing to enhance performance. In particular, the ability of the MRN to employ different memory combinations, encourages appropriate selection and utilisation of historical data to effectively inform predictions. Interestingly, the third dataset required less trainable parameters than the models that used the first or second dataset, underpinning the discovery of enhanced performance incorporating the change of direction (see Table 4.2).

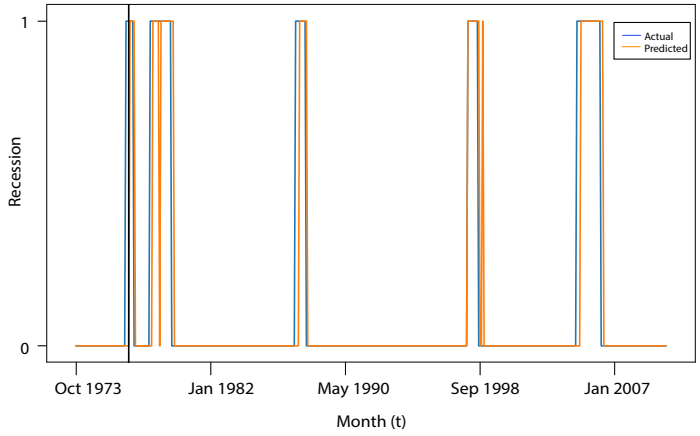
In particular, the graphs for the NBER turning points prediction task are shown in Figure 4.1. All three models have a lag when predicting the first, second and fourth recession. The model applied to the third dataset very closely maps the third recession, unlike the models that employed the first and second datasets. Overall, training the MRN with the three datasets provides meaningful results, highlighting its suitability for time-series processing.

4.2.2 Oil price prediction

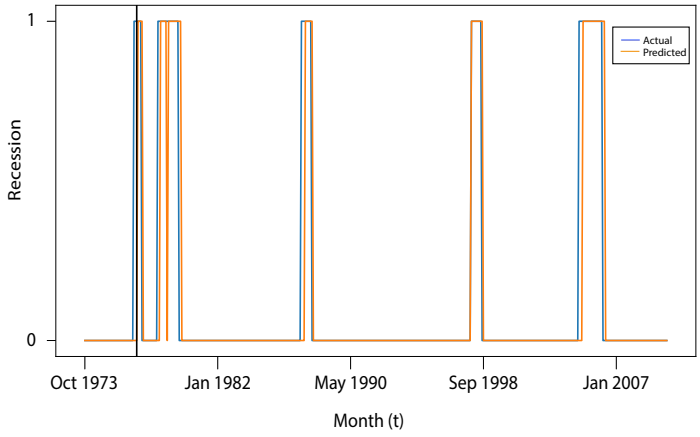
Four models (Random Walk, Jordan, SRN and MRN) were applied to the crude oil price prediction task and comparative results are presented. The MRN's per-



(a) Growth



(b) COD



(c) Growth and COD

Figure 4.1: Best MRN Models

formance is compared to models within its class *(that is, networks employing simple recurrency)* and the best of these models is then compared with the currently accepted state-of-the-art LSTM model. *Note: Tepper et al. [200] cogently demonstrated that MRN outperformed the SRN ESN and NARX, the work in this section builds on this finding.*

Random Walk (RW): The *random walk* refers to a model where the best forecast for the volatility of the next time step is the volatility of the current time step, and it indicates the efficiency of the oil market [168]. For a RW model, the best prediction of volatility for any time in the future is the current volatility. The RW model is used as a benchmark.

Simple Recurrent Network model class: The SRN[1], *Jordan network*[2] and the *MRN*[3] are applied for the crude oil prediction task. These three models used the sliding window approach presented in Chapter 3.2.1 and had the following hyper-parameters: 20 hidden units, an initial learning rate of 0.01 and a momentum of 0.9999.

LSTM: The *LSTM* is trained with 20 units and employs a sigmoid activation function. Experiments with two optimisers; SGD and Adam, and three dropouts [0.1, 0.4, 0.7] are undertaken.

Table 4.3 - Table 4.6 presents the RMSE of the models (RW, Jordan network, SRN, MRN) at 4 different horizons (1, 3, 6, 12 *(months)*) with 4 different window sizes (60, 120, 240, 300). The Improvement over Random Walk (IORW) for the models are also presented in the tables. The best window sizes for each model at the different horizons are highlighted in black and the overall best models across all the models are highlighted in blue. As can be seen, the MRN outperforms all the other models, supporting the claims of its superior performance. This supe-

[1]The SRN used is as described in [52], the previous hidden state along with the current observations are fed as inputs at any given time to the network.

[2]The Jordan network used is as described in [46], the previous output state along with the current observations are fed as inputs at any given time to the network.

[3]The MRN is described in Chapter 3 where multiple feedbacks are utilised.

riority becomes clearer as predictions are made further in time and is attributed to its memory mechanism which encourages strong information latching.

Table 4.3: Best RMSE scores for the Oil price prediction task of the RW for different horizons *(1, 3, 6, 12)*

	t+1	t+3	t+6	t+12
RMSE	3.56	8.02	12.32	14.11

Table 4.4: Best RMSE scores for the Oil price prediction task of the Jordan network for different horizons *(1, 3, 6, 12)*

	t+1		t+3		t+6		t+12	
Window size	RMSE	IORW	RMSE	IORW	RMSE	IORW	RMSE	IORW
60	0.363	0.898	**0.694**	**0.913**	0.995	0.92	1.572	0.89
120	0.339	0.9	0.714	0.911	0.967	0.922	**0.920**	**0.93**
240	0.331	0.91	0.719	0.91	0.996	0.92	1.040	0.926
300	**0.33**	**0.91**	0.713	0.911	**0.947**	**0.923**	0.925	0.93

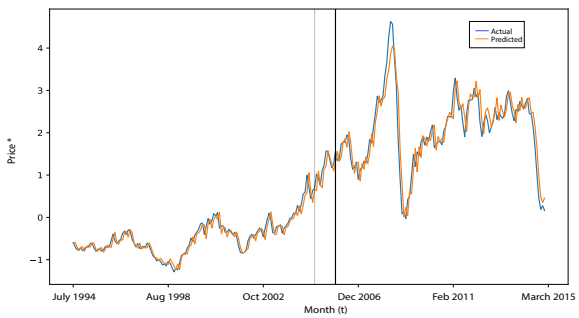
Table 4.5: Best RMSE scores for the Oil price prediction task of the SRN for different horizons *(1, 3, 6, 12)*

	t+1		t+3		t+6		t+12	
Window size	RMSE	IORW	RMSE	IORW	RMSE	IORW	RMSE	IORW
60	0.337	0.91	**0.7**	**0.91**	**1.061**	**0.91**	1.86	0.87
120	0.335	0.91	**0.7**	**0.91**	1.22	0.9	1.488	0.89
240	**0.332**	**0.91**	0.861	0.89	1.57	0.87	**1.409**	**0.9**
300	0.334	0.91	0.774	0.9	1.33	0.89	1.627	0.88

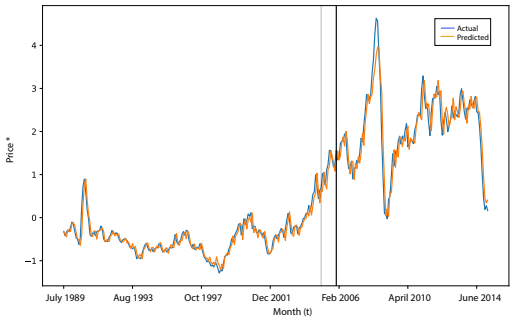
The Jordan network outperformed the SRN, suggesting that output feedbacks are more important than hidden feedbacks for the Oil price prediction task. As expected the models achieved the best prediction accuracy for the shortest horizon, 't + 1' (one-month ahead) as it has the least volatility compared to the other horizons which are further ahead in time. *Note: the best models have different window sizes, thus the in-sample periods shown will differ, however, the out-sample is the same for all models.*

Table 4.6: Best RMSE scores for the Oil price prediction task of the MRN for different horizons *(1, 3, 6, 12)*

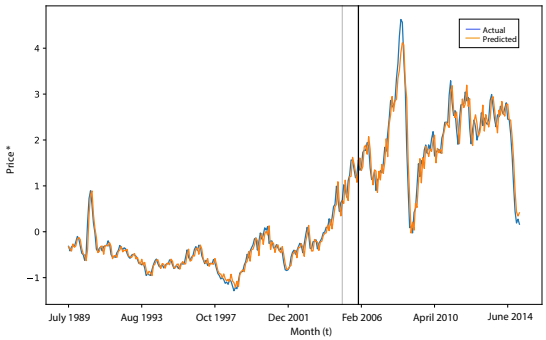
	t+1		t+3		t+6		t+12	
Window size	RMSE	IORW	RMSE	IORW	RMSE	IORW	RMSE	IORW
60	0.33	0.91	**0.693**	**0.914**	0.976	0.921	**0.892**	**0.937**
120	0.326	0.91	0.697	0.913	0.985	0.92	0.942	0.933
240	**0.321**	**0.91**	0.701	0.912	**0.864**	**0.93**	0.974	0.931
300	0.322	0.91	0.747	0.91	0.973	0.921	0.978	0.931



(a) Jordan



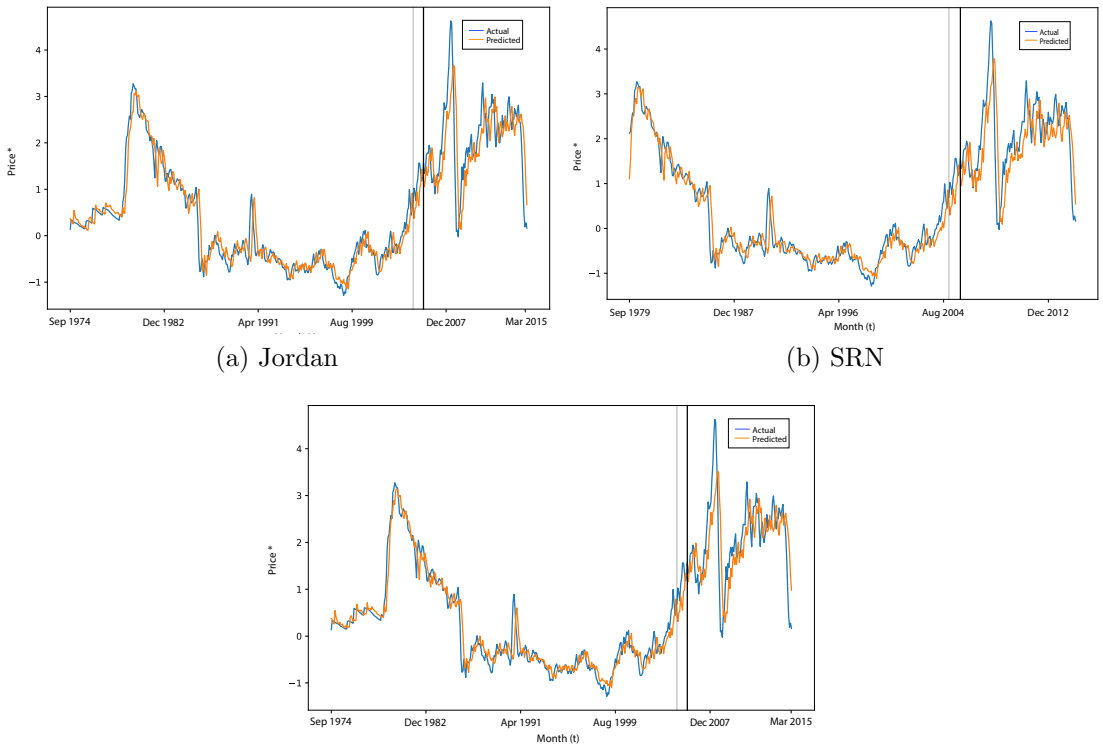
(b) SRN



(c) MRN

Figure 4.2: Best Models for 't + 1'

Figure 4.2 shows the predictions for the best models and the observed values for 't + 1' prediction task and as seen, all the models appear to closely follow the signal. From Table 4.3 - Table 4.6, the errors for the best SRN, Jordan & MRN models are very similar. All three models demonstrated strong predictive abilities for 't + 1' oil price prediction task.



(a) Jordan

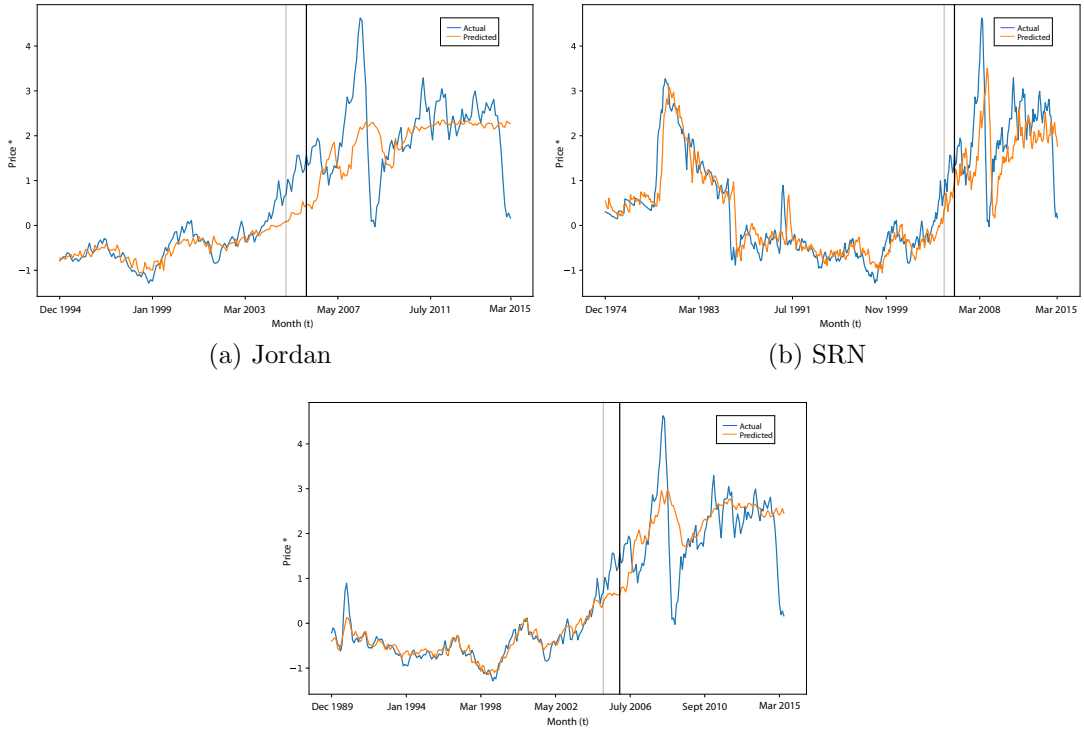
(b) SRN

(c) MRN

Figure 4.3: Best Models for 't + 3'

Figure 4.3 visualises the predictions for the best models and observations for 't + 3' prediction task. Similar to the 't + 1' prediction task, all the models appear to closely follow the signal, however, the predictions made are in a slightly smaller range than that of the observed values. All three models are strong candidates for the 't + 3' oil price prediction task.

Figure 4.4 visualises the predictions for the best models and the observed values for 't + 6' predictions. The MRN performed best, it appears to provide predictions by smoothing out using a weighted average. At most of the peaks, the MRN predicts values close to the observed, it however appears to miss most of the troughs in the signal. The Jordan network had a lower error than the SRN, however it appears to have difficulty mapping the signal. It smooths out similar to the MRN, however, it 'over smooths' such that it obtains averaged values within a small range for the predictions. The predictions are very close to most



(a) Jordan

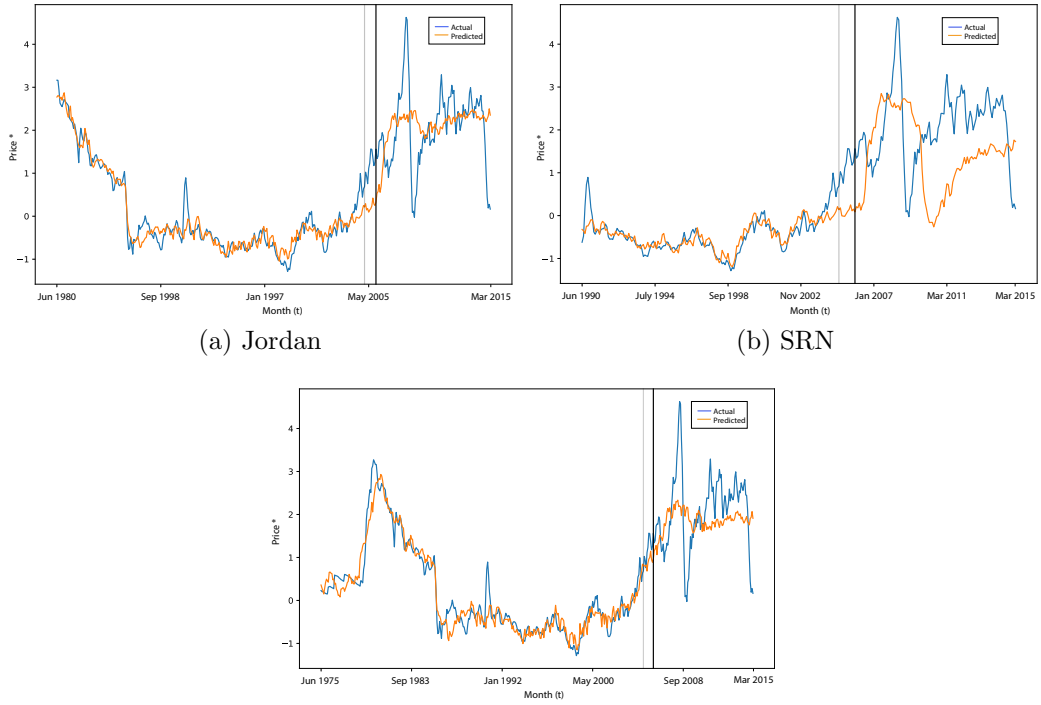
(b) SRN

(c) MRN

Figure 4.4: Best Models for 't + 6'

of the troughs, and it misses all the peaks. Finally, the SRN appears to follow the trend of the signal with a noticeable lag. In particular, obtaining predictions after they occur offers little to no benefit, thus limiting the suitability of the SRN for mid-term forecasting (such as 't + 6', *(6 months ahead)* prediction tasks). The results for 't + 6' prediction indicate that it is a volatile period, and mapping the signal can be difficult, given the number of changes that occur within the time-frame *(for example, the financial crisis or Covid-19 and the rapid change and impact on the economy between 3 - 6 months).*

Figure 4.5 presents the predicted values for the best models and the observed values for 't + 12' predictions and as seen, all the models appear to have difficulty mapping the signal. The Jordan network has a lower error for 't + 12' prediction than 't + 6' prediction. Similar to the 't + 6' prediction task, the MRN and the Jordan network appear to smooth out the values to make predictions, identifying some peaks and troughs of the signal, however, on a significantly smaller scale.



(a) Jordan

(b) SRN

(c) MRN

Figure 4.5: Best Models for 't + 12'

The SRN appears to have lagged predictions, as with the 't + 6' prediction task. The results from 't + 6' and 't + 12' indicate that the SRN is not suitable for long-term forecasting, supporting Tepper's [200] claim of limited processing and 'forgetting'. The MRN on the other hand, although it smooths out for 't + 6' and 't + 12' provides the best results.

Table 4.7: Hyper-parameters for the best MRN model

Horizon	Memory banks *([input, hidden, output])*	Window size	RMSE	Trainable parameters
1	[0, 4, 2]	240	0.321	1640
3	[0, 4, 4]	60	0.693	1680
6	[4, 3, 4]	240	0.864	2321
12	[0, 0, 3]	60	0.892	301

Table 4.7 presents the key hyper-parameters and the total number of adjustable parameters for the best MRN models. It is specifically noted that the memory bank combination varies for the different horizons. This flexibility is believed to encourage more stable formations of representations and better predictions.

The MRN is then compared to the LSTM and as shown in Table 4.8 outperformed the LSTM in all but one prediction task *('t + 3')*. In particular, the MRN requires significantly fewer adjustable parameters than the LSTM to learn the problem. The MRN appears to offer a simpler and more effective alternative to more complex (processing and computational) models.

Table 4.8: Comparative RMSE results of the MRN and LSTM *(with the number of trainable parameters shown in parentheses)*

Model	't + 1'	't + 3'	't + 6'	't + 12'
MRN	0.321 *(1,640)*	0.693 *(1,680)*	0.864 *(2,321)*	0.892 *(301)*
LSTM	0.448 *(5,781)*	0.688 *(5,781)*	0.932 *(9,061)*	1.161 *(9,061)*

The results show that the MRN is more persistently able to capture the temporal dependencies in crude oil prices within the evolving oil market. The enhanced performance with the MRN is attributed to the varying degrees of embedded memory within the network. In addition, these predictions appear to provide early indication of the 2008 financial crisis as the parameters in the MRN, are tuned up to December 2004, highlighting the MRN's ability for early warning indication.

4.2.3 M3 competition prediction

A number of models are applied to the M3 competition dataset. The forecasts for the initial 24 models employed are available at: https://forecasters.org/resources/time-series-data/m3-competition/. The MRN is compared to the best 5 of these models. Experiments with varying memory banks, 10 hidden

units and window sizes of [10, 40] are employed to obtain the best MRN. Table 4.9 shows the RMSEs for each model for the 10 randomly selected series. In addition, similar to the work presented by [119], a simple average of the all the RMSEs for each model is used to identify the best model and is presented in the table.

Table 4.9: Comparative RMSE results of six models applied to the M3 competition data

Model		MRN	Theta	Forecast Pro	ForcX	PP-Autocast	Dampen
Series	N2516	**187.1**	882.9	920.7	920.6	919.3	919.6
	N2521	2017.3	2028.5	2008.5	2018.7	2011	2011.7
	N1807	268.3	250.6	478.1	424.1	299.0	456.2
	N1908	453.9	362.1	317	313.1	333.5	373.4
	N2012	517.8	297.6	220.5	229.2	360	383
	N2159	521.9	478.5	466.5	494.4	509.3	437.8
	N2158	651.1	489.6	485.4	510	512.7	468.6
	N2150	141.6	171.7	82.2	128.4	141.2	183.5
	N2144	**538**	1091.5	1182.7	1181.7	1182.9	1182.8
	N1918	206.7	143.3	157.6	157.5	194	137.6
Average		**550.4**	619.6	631.9	637.8	646.3	655.4

As shown in Table 4.9, the MRN performs best for 2 of the series and overall, it has the most consistent results, obtaining the overall best average RMSE. All the models had the lowest RMSEs for series *N1918* and *N2150*, indicating the models could accurately model and map the signal, while the models had the highest RMSEs for series *N2144* and *N2521*, indicating the models had difficulties modelling and mapping the signal. The average RMSEs are indicative of the overall/summarised performance of the models, however, they are affected by relatively small or large sample values *(for example: RMSE scores for series N2521)* and may hide disparities.

Therefore, a non-parametric statistical hypothesis test, *The Friedman test*, is conducted. The model ranks are calculated and using Equation 4.2, the chi-square and F value are calculated to identify whether the models are significantly different *(see Table 4.10 for rankings)*. The null hypothesis H_0: The models are equal, that is the models are not significantly different and the alternate hypothesis H_A: The models are not equal, that is the models are significantly

different.

$$X_F^2 = \frac{12N}{k(k+1)}[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}]$$
$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2} \tag{4.2}$$

where X_F^2 is the chi-square, k = number of models, N = number of series and R_j is the sum of the ranks for the j^{th} series.

Table 4.10: RMSE ranking for the six models applied to the M3 competition data

Model		MRN	Theta	Forecast Pro	ForcX	PP-Autocast	Dampen
Series	N2516	**1**	2	6	5	3	4
	N2521	4	6	1	5	2	3
	N1807	2	1	6	4	3	5
	N1908	6	4	2	1	3	5
	N2012	6	3	1	2	4	5
	N2159	6	3	2	4	5	1
	N2158	6	3	2	4	5	1
	N2150	4	5	1	2	3	6
	N2144	**1**	2	4	3	6	5
	N1918	6	2	4	3	5	1
Average Rank		4.2	3.1	2.9	3.3	3.9	3.6

In order to investigate the hypothesis, the calculated F_F value 3.49 is compared to a known critical value of 11.07 *(obtained using the chi-squared table with an alpha level of* 0.05 *and* 5 *degrees of freedom)*. The calculated F_F value is less than the critical value, thus the null hypothesis, H_0, for the Friedman test is accepted, that is the models are not significantly different. In particular, as seen in Table 4.10, the MRN has the lowest overall rank suggesting it is the lowest performing model differing from the results obtained using the overall average. The Friedman test, as with many other non-parametric test, is less restrictive, relaxing the assumptions of the data. However, this may lead to less reliable results when the variability differs and as seen from Table 4.9 the variability between the models differ. In addition, the Friedman test is not strong particularly with

smaller sample sizes [220]. Iman and Davenport [88] derived a better statistic as they demonstrated the Friedman test is "undesirably conservative". The test also does not account for when the models do not learn and this is particularly crucial as one of the aims of this research is to identify a suitable model across different problem domains rather than in specific domains. It is evident that more testing techniques need to be conducted, thus the reader is therefore directed to Chapter 8.4 where future work on model forecast evaluation is presented.

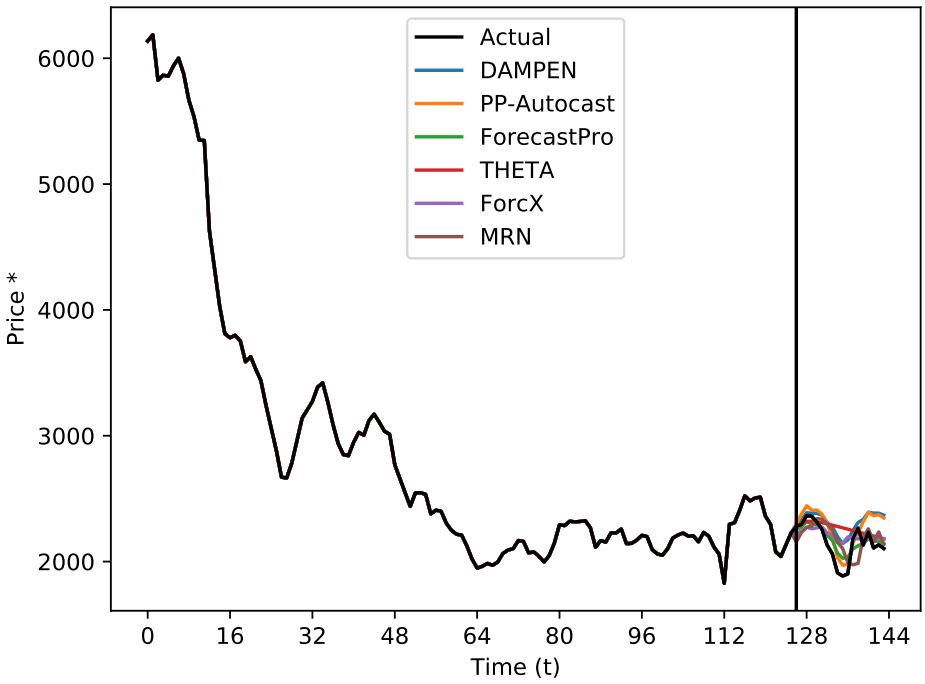


Figure 4.6: Series N2150

The model predictions and the observations for these four series are visualised in Figure 4.6 - Figure 4.9. Series *N2150* is shown in Figure 4.6, all the models appear to closely follow the series trend. Particularly, the PP-Autocast and Forecast Pro appear to make predictions that follow the signal, while the MRN appears to make predictions (including the trough) a few time-steps after it has occurred. The THETA appears to predict a straight line with a downward trend whilst the remaining models appear to follow the signal, although mapping the

trough on a smaller scale. The PP-Autocast and DAMPEN appear to make significantly different predictions to the observed towards the end of the series, overall the Forecast Pro's has the lowest score.

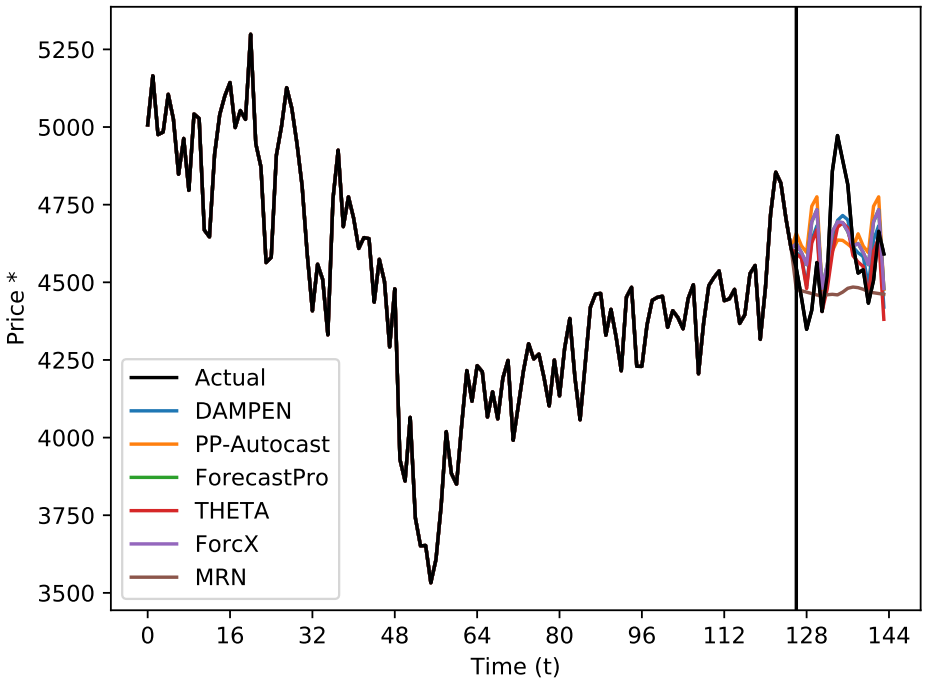


Figure 4.7: Series N1918

Series *N1918* shown in Figure 4.7 is more volatile than series 2150. For this series, all the models outperform the MRN. They provide predictions on a larger scale for the first and last peaks and on a smaller scale for the second peak. The MRN however appears to predict values that are smoothed and averaged.

Most models appear to have difficulty modelling series *N2144* and *N2521*. In particular, most of the models appear to predict a straight line for series *N2144* and could not identify the drastic drop in price. The MRN is the only model that could map the signal and provide useful predictions. Similarly, most models except the MRN predict a straight line for series *N2521* as seen in Figure 4.9, which is very volatile. The MRN is the only model that appears to map the trend, however on a much smaller scale than the observed values.

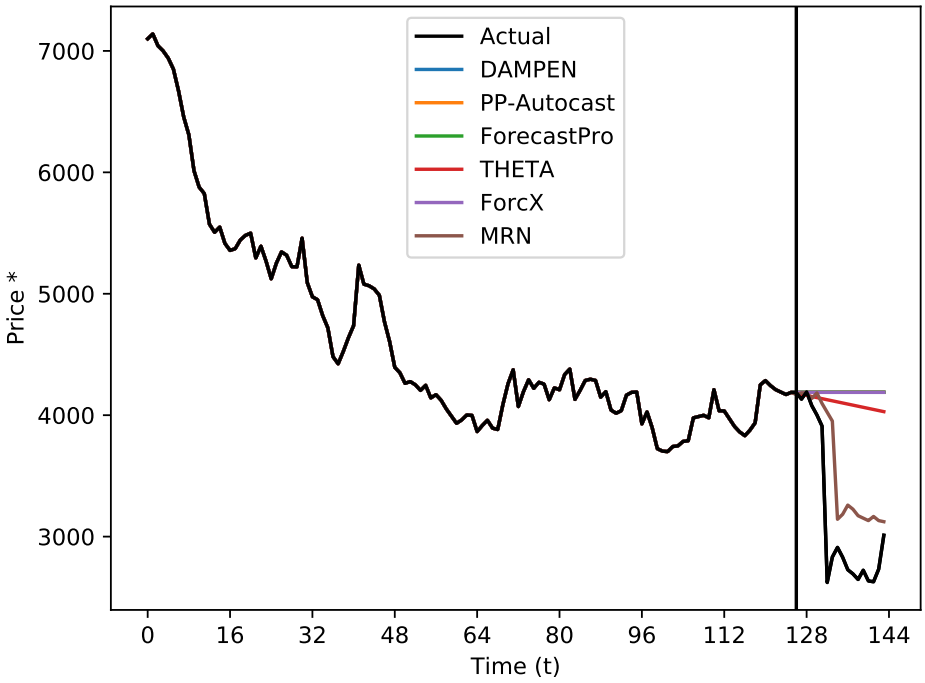


Figure 4.8: Series N2144

These series reflect real-world dynamics which can be volatile and chaotic. Although, the Friedman test indicates no significant improvement with the MRN, the MRN showed consistency with accurate predictions and learnt the mapping despite having the highest RMSE for some series. In particular, for two of the series, *N2144* and *N2521* were the other models had difficulties learning, the MRN was able to model and map the signal, specifically due to its state-based mechanism, enabling the modelling of different states available in the series. In conclusion, the M3 series are a notable benchmark for time-series forecasting, the results in this section along with the application of the MRN to other domains point to its suitability for time-series processing across a number of different fields and domains unlike traditional techniques or other ML techniques.

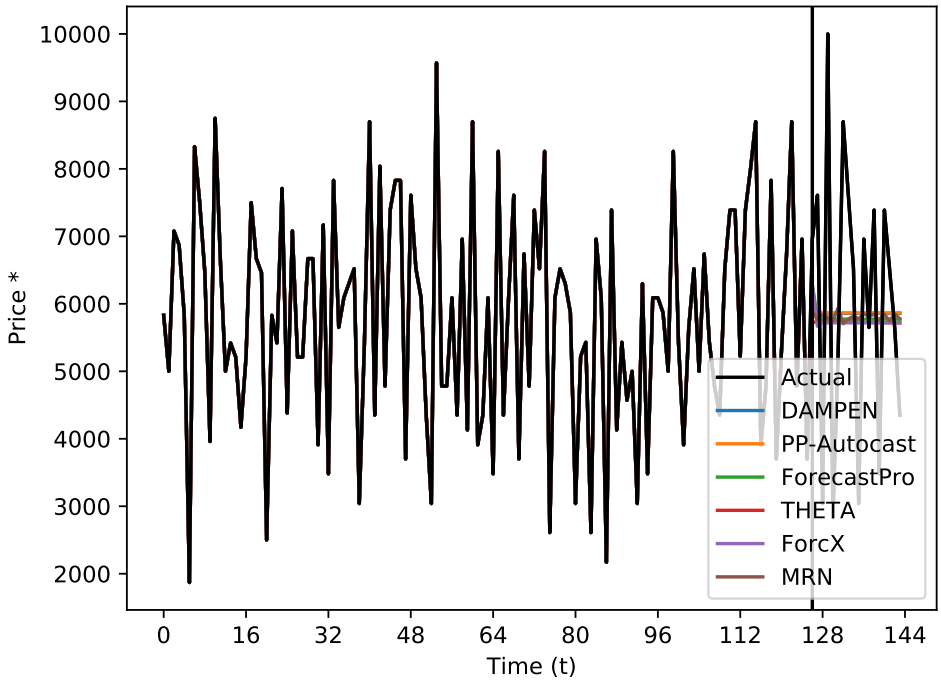


Figure 4.9: Series N2521

4.2.4 Covid-19 forecasting

The World Health Organisation (WHO) declared Covid-19 outbreak a pandemic on the 11th of March 2020 [41]. From the 11th of March to date, the outbreak has spread rapidly to many countries and has had a drastic impact on our day-to-day lives. A number of researchers in an attempt to understand the virus particularly the rate at which it spreads have applied various forecasting tools to forecast the future cases *(death or confirmed)*.

Data from a number of countries (such as India, Canada, United States) that were largely impacted by Covid-19 have recently made available. Covid-19 Data for the United States has been selected for analysis due the veracity of the data provided, the US being the world's largest economy and technologically advanced nation and it has been heavily impacted by Covid-19 in terms the rate of infection and number of deaths. In addition, the LSTM has been systematically

applied to this data set and will therefore act as a high-quality benchmark for the MRN [10, 12, 31, 203]. In addition, there is no universally agreed metric used by researchers to evaluate the performance of their models with respect to others and unfortunately, this precludes direct comparisons. As both the MRN and LSTM will be optimised for the Covid-19 data in this chapter and the same evaluation metrics used, it will offer insight into relative performance of each model and ascertain which model is more efficacious. The reader is therefore directed to Chapter 8 future work for a brief discussion on how the important Covid-19 study could be extended to include other countries.

The MRN is applied for the task and then compared to the LSTM. The models are assessed using the Mean Absolute Percentage Error (MAPE), calculated as follows:

$$M = \frac{1}{n} \sum \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{4.3}$$

where y_t is the actual outcome, $\hat{y}_t$ is the predicted outcome and n is the total number of outcomes.

MRN: The MRN employed is as described in Chapter 3. It employed 20 hidden units and experiments with varying memory banks and 5 window sizes [15, 20, 25, 35, 40] were undertaken to identify the best MRN.

LSTM: The LSTM employed a sigmoid activation function and a dropout regularisation technique to reduce over-fitting. For experimentation, 2 units [20, 100], 3 dropout values [0.1, 0.4, 0.7], 3 batch sizes [20, 50, 100] and three optimisers (RMSprop, Adam and SGD) were employed to obtain the best model.

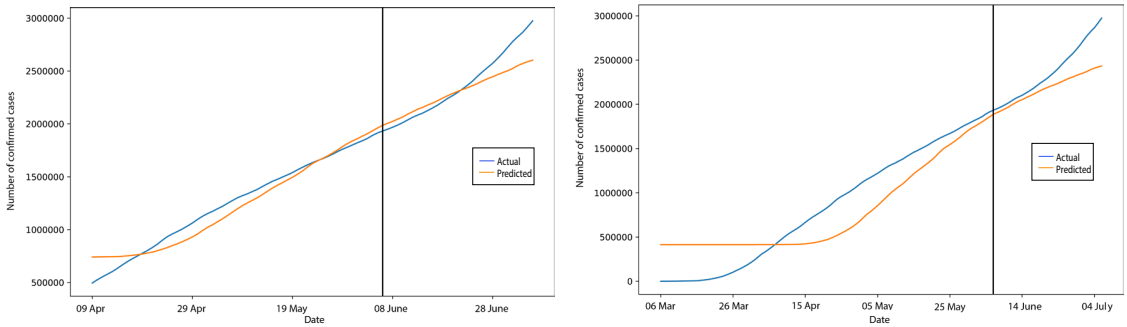
Table 4.11 presents the results for the best MRN and LSTM models for Covid-19 forecasting and as seen the MRN outperformed the LSTM.

Further experiments are conducted with the LSTM and as seen increasing the number of epochs and the units leads to significant improvements. Despite this improvement, the MRN still outperformed the LSTM.

The predicted values of the MRN and LSTM along with the observed values for the confirmed cases are presented in Figure 4.10a and Figure 4.10b. The MRN

Table 4.11: Comparative MAPE results of the MRN and LSTM for Covid-19 forecasting of confirmed and death cases in the USA

Model (epochs)	Hidden units	Confirmed cases	Death cases
MRN (500)	20	**4.11**	**0.3**
LSTM (500)	20	7.03	22.67
LSTM (1000)	20	6.998	7.62
LSTM (1000)	100	5.14	1.33



(a) MRN (b) LSTM

Figure 4.10: Confirmed cases

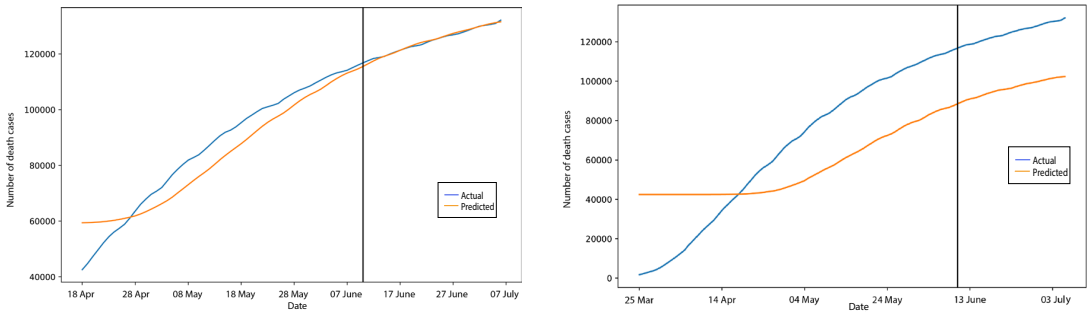
appears to follow the trend of the confirmed cases closely, it does however miss the sharp increase from mid-late June. The LSTM on the other hand, appears to map the trend of the signal, albeit not as well as the MRN and also misses the sharp increase in confirmed cases from mid-late June[1].

Figure 4.11a and Figure 4.11b present the predicted values of the MRN and LSTM along with the observed values for the death cases. The MRN appears to have learnt the underlying signal and predicts values for the death cases similar to those observed. The LSTM on the other hand, loosely follows the trend of the death cases although on a smaller scale[2].

Interestingly, the MRN, albeit a much simpler and less computationally in-

[1]Note: Both models have the same training and test set, however the number of windows available to the MRN is dependent on the window size, which determines the number of observations available; the MRN starts predicting from early April.

[2]Note: Both models have the same training and test set, however the number of windows available to the MRN is dependent on the window size, which determines the number of observations available; the MRN starts predicting from mid-April.



(a) MRN (b) LSTM

Figure 4.11: Death cases

tensive model class, outperformed the LSTM. More specifically, the MRN requires much fewer adjustable parameters than the LSTM for this problem. The MRN's memory mechanism enables it to 'remember' different spatio-temporal relationships within the signal and provides a descriptive overview of the information available. The results are indicative of the strong predictive abilities of the MRN for time-series forecasting and future work includes extending the MRN for Covid-19 forecasting for other countries and comparing its performance with notable papers.

4.3 Discussion

There has been a shift in recent years from simple recurrent networks *(such as SRNs and Jordan networks)* to more complex recurrent networks *(such as ESNs and LSTMs)*, particularly due to the vanishing gradient problem, which limits processing of time-dependence variables over a long period of time. However, in this chapter, the MRN, a type of simple recurrent network, demonstrates strong competencies for time-series processing and forecasting. It specifically employs multiple memory banks to encourage strong information latching. The results in this chapter particularly indicate that the shift to more complex mechanisms is premature and simpler recurrent networks such as the MRN are suitable and can provide superior predictive abilities compared to more complex methods currently employed for time-series analysis.

4.3.1 MRN suitability

The MRN demonstrated persistent superiority with time-series data and consistently outperformed current state-art-of-the models as seen in Section 4.2.1 - Section 4.2.2 and Section 4.2.3 - Section 4.2.4. As pointed out from the literature review in Chapter 2, the results in these Sections demonstrated the importance of an appropriate memory mechanism coupled with the right level of complexity to encourage and ensure effective time-series modelling and forecasting.

4.3.2 Computational complexity

Experiments presented in this section demonstrate that the MRN specifically required a significantly lower number of parameters than the LSTM *(current state-of-the-art)*. As stated in Section 3.3, the MRN employed the BPTT as presented in Section 3.1.2.2 and has a space complexity of $O(((m_w * m_s) + n)h)$. While, the LSTM models employed in this chapter utilised the following optimisations: SGD, Adam and RMSprop. SGD is the basis for all these optimisers and its space complexity is $O(o^4 + o^2r^2)$, where o and r are the output and regressor space *respectively* [53]. The gating mechanism of the LSTM particularly increases the complexity. More specifically, the MRN proves a worthy competitor as it not only obtains superior performance but also requires a lower number of parameters to obtain this enhanced performance.

4.3.3 Memory bank search space

The MRN is a flexible paradigm such that it can employ as many memory banks as needed for a specific task. While this sophistication is a key attribute for its superiority, as the memory order increases the number of possible combinations exponentially increases. For example, a memory order: of 4 requires $4^3 = 64$ models, of 5 requires $5^3 = 125$ models, of 6 requires $6^3 = 216$ models, of 7 requires $7^3 = 343$ models to obtain the best model and of 8 requires $8^3 = 512$ models. Excluding the memory bank combination [0, 0, 0], 63, 124, 215, 342, 511 model combinations are required for training to obtain the best model. As seen, the search space for the memory bank configuration is large, thus, the memory

order *(that is the maximum number of memory banks employed in each memory bank type)* is restricted to 4, as stated in Section 3.1.1.2. However, where more memory banks for each memory bank type are required, restricting the number of memory banks limits performance.

The results presented in this chapter demonstrated the advantages of the memory bank architecture and its suitability for modelling complex time-series. In particular, Ulbricht [208] raised the following questions for exploration with the MRN, *"Are there methods to improve the quality of the memories?"* and *"What techniques can be implemented to properly handle high-dimensionality in the MRN?"*. The MRN will be investigated in the following chapters, with a view to developing innovations that address the limitations highlighted, positioning the MRN for effective modelling of more complex time-series data.

4.4 Conclusion

In this chapter, the MRN was applied to a number of time-series forecasting tasks, to access its suitability and relative performance. The MRN was compared to state-of-the-art ML techniques such as the LSTMs, SVMs and SRNs to benchmark its performance. The experiments confirmed that the MRN is indeed a worthy competitor when compared to other state-of-the-art techniques and worthy of further exploration and optimisation.

In particular, the memory architecture of the MRN is shown to endow and enable effective spatio-temporal processing that encourages the identification of patterns and signal in the data. The architecture of the MRN provides more flexibility for the network to 'select' relevant information through the training process as parameters are updated. The variety of history strength enables the network to exploit the data, providing a more informative 'averaged' overview of the underlying signal, thus, encouraging better and more accurate predictions and enhancing performance. In addition, the MRN offers a more computationally effective paradigm than more complex models such as the LSTM, thus giving rise to better accessibility and harnessing the MRN for day-to-day usage and utilisation within low-end devices. Therefore, further exploration of the MRN's

architecture is undertaken in the following chapters.

Chapter 5

Time Sensitivity in the Multi Recurrent Network

This chapter introduces a novel extension to the MRN, to overcome two inherent learning limitations associated with the rigidity of its hidden layer. More specifically, time sensitivity is introduced within the hidden layer, to overcome the problem of catastrophic interference by partitioning the hidden units such that they are only active at specific time intervals, thereby reducing the temporal loading on each hidden unit. In addition, the approach enables the sets of units to hold activations for longer periods of time, which in turn provides a means to mitigate the vanishing gradient problem. The main objective is to specifically enhance the MRN as it outperformed alternative algorithms in most instances as shown in Chapter 4. Therefore, this extension to the MRN model is only compared to the standard MRN across four of the different problem domains introduced in Chapter 3, in order to understand the impact of these innovations on the behavioural dynamics of the MRN.

5.1 Background

Chapter 2 gave a critical account of recurrent neural networks, including the MRN, and revealed the need for more efficacious memory mechanisms for capturing temporal dependencies across time, thus ensuring effective time-series pro-

cessing. The MRN has been discussed and demonstrated to be a powerful dynamic modelling tool with a unique memory mechanism, which enables enhanced information processing and signal extraction. The MRN has shown strong abilities to capture the latent signal in non-linear time-series data, improve learning and achieve better accuracy *(compared to the SRN, NARX, ESN, LSTM, LVQ, SVM and Bayesian networks)*. As stated earlier, the MRN is not immune to the gradient descent problem and catastrophic interference typical of neural network architectures, such as MLPs and SRNs, that use shared weights and units. In particular, the MRN does not specifically partition and allocate its resources *(the adaptable parameters)* to learn the different *(short- and long-term)* prediction tasks, rendering it susceptible to the gradient descent problem albeit to a lower degree.

To illustrate the memory limitations of SRNs, that is their inability to simultaneously learn both the short- and long- term tasks, which in turn induces the problem of catastrophic interference and vanishing gradient, O'Connell [144] applied an SRN to the Embedded Reber Grammar (ERG)[1]. This highlighted the SRN's inability to separate various levels of embeddings generated from sub-grammars related across time. A sentence example analogous to this issue represented by the ERG is presented by Tepper *et al.* [200] using the subject-auxiliary verb agreement further discussed in Section 5.1.1. The short-term task is to predict the next word within a sentence given the current word, and the long-term task is to predict the agreement between words in a sentence that have been separated by intervening clauses (generated using some sub-grammar). Consider this example:

- S1: The dog that chased the cat it saw is very playful
- S2: The dogs that chased the cat they saw are very playful

For the above example, in order to correctly predict the auxiliary verb *('is' or 'are')*, the network must retain a representation of the subject's plurality *('dog' or 'dogs')*, as it traverses through the embedded clauses or grammar *('that*

[1]The Embedded Reber Grammar is an extension of the 'standard' Reber Grammar, it is a useful problem for RNNs to solve, as it helps to understand how well they model long-term dependencies (see [144]).

chased the cat'). Cleeremans *et al.* [35] highlighted the need for hidden unit components to maintain such activations through the sub-grammar. Additionally, O'Connell [144] emphasised the need for appropriate resource allocation to ensure both prediction tasks are completed.

Over the last few decades, researchers have identified that significantly enhanced performance can be obtained by either extending the architecture of ANNs *directly* (for example, incorporating self-learning, self-organising, internal decays or attentive nodes) or *indirectly*, by augmenting their architecture with other techniques (for example, by using wavelet or fuzzy learning). Therefore, in this chapter, an extension to mitigate the weaknesses of the MRN will be proposed and investigated. The two learning limitations to be addressed are; *gradient descent learning problem* and *catastrophic interference of hidden states due to all temporal features being super-imposed onto a single homogeneous hidden feature layer*. To resolve this issue, periodically attentive units are introduced within the hidden layer of the MRN.

5.1.1 Gradient Descent Learning Problem & Catastrophic Interference

RNNs are known to suffer significantly from the gradient descent problem. The *gradient descent problem* can be simply characterised by either i) the rapid decay of gradient information essential for learning, as it propagates either through the hidden layers of the ANN or as it propagates through time, when linearly processing sequential inputs; or ii) the exponential increase in gradient information *(explosion)* during training due to the 'excessive multiplication' of the long-term components over multiple time-steps [15, 153].

In addition, RNNs also suffer from *'catastrophic interference'* such that all the resources *(adaptable parameters)* within the network respond simultaneously to input information at every time period. Thus, these networks are unable to preserve resources for the long-term dependency learning once the short-term dependency learning has occurred [144]. *Catastrophic interference* within ANNs is also observed when the models are required to learn additional information after an initial training period. In such situations of continual incremental learning,

the network forgets knowledge acquired from the previously learned tasks (or data distributions), due to the use of shared hidden units [38]. Thus, forcing a superposition of different task information across the hidden weights and units, thereby causing 'forgetfulness' and 'confusion' as the representation of one new pattern affects the representation of all other patterns previously stored [105,191].

Researchers have sought to address these problems by shifting from simpler recurrent networks to more complex networks such as LSTMs and Gated Recurrent Units. The LSTM and its embedded gating mechanism were developed to deal specifically with vanishing gradients, by employing three gates *(forget, input, output)* to update and control the states of the cell. The forget gate informs what information should not be forgotten by the network. However, as presented in Section 2.2.4.5 the LSTM's modelling and predictive abilities are limited as they i) have limited representative abilities due to the lack of an appropriate memory mechanism [42], ii) are not easily adaptable [30] and iii) are inefficient for long-term dependency task [113]. In addition, [170] highlighted the 'over-complexity' of LSTMs. A systematic review on different LSTM cells carried out by [232] pointed to the need for 'external' memory to strengthen memory capacity.

Similarly, Cho *et al.* [33] introduced the Gated Recurrent Units (GRUs) *(a simpler LSTM variation)* in 2014, to tackle the vanishing gradient problem alongside reducing the network complexity (presented in the LSTM). GRUs optimise the LSTM's structure by combining the three LSTM gate units *(input, output and forget gate)* into two gate units *(reset and update gate)*. This enables the network to learn to skip 'irrelevant' temporary information. Thus, reducing the network complexity, enabling information dependency over a longer time to be maintained *(as redundant information is continuously discarded, and the hidden state can be more effectively utilised to store key information dependencies)* [216]. However, Wang *et al.* [216] pointed out some limitations with GRUs such as slow convergence rate, low learning efficiency and inability to discard enough redundant information in one screening for complex states of time series data.

Rather than resorting to complex gating mechanisms and thus, the use of multiple activation functions within the memory layer, O'Connell [144] sought to deal with this problem by employing Periodically Attentive (PA) units to extend the temporal capacity of SRNs. O'Connell [144] defined PA units as "units that

receive input at some regular interval". This is different from standard information processing within the SRN where hidden units receive input information at every time-step.

The ERG example (see Figure 5.1) is a notable benchmark and has been exploited by a number of researchers such as Hochreiter and Schmidhuber [84] with the LSTM and more recently Tepper *et al.* [200] with an MRN, ESN and NARX, Wang *et al.* [212] with an Auxiliary Memory Unit RNN, to assess the robustness of these networks and their ability to capture long-term dependency *(discussed in Section 2.2.4.7)*.

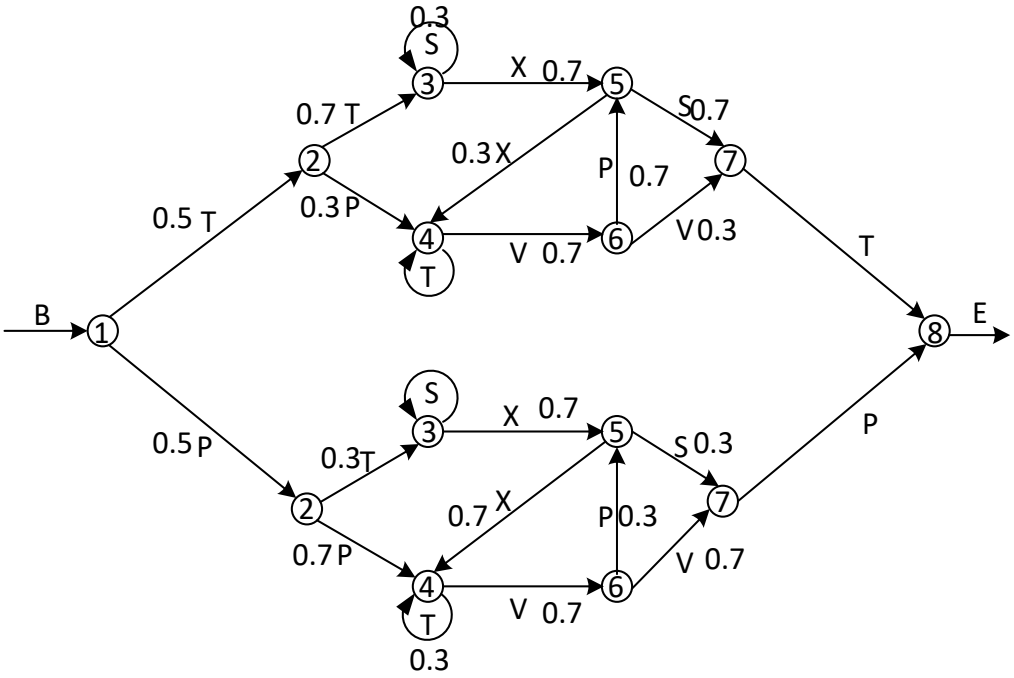


Figure 5.1: The ERG as presented in [200]

For the long-term prediction task of the ERG, the network must maintain the hidden activations caused by the indicator symbol *(the symbol between ① and ②)*, to correctly predict the penultimate symbol *(the symbol between ⑦ and ⑧)* [144]. In order for this to happen, the weights from the context layer *(which holds the memory)* to the hidden layer must be relatively large compared to the weights between the input and hidden layer [144]. This enables a stronger

role of previous hidden activations than current inputs for determining the next hidden activations [144]. However, weight initialisations are very small and as such it is unlikely that hidden activations will reflect the indicator symbol [144]. In addition, the error generated earlier in training will only reflect the short-term error *(to remember the last two symbols at any given time-steps)* [144]. Given the recurrence of the short-term error, the resources of the network are predominately utilised for the short-term task [144]. Similarly, Sentence S1 and S2 presented in Section 5.1 can also be explained using the ERG. Consider the top string of the ERG as the path for the singular subject and the bottom string as the path for the plural subject, such that the indicator symbol *(the subject)* must be remembered over a given number of time-steps to predict the correct verb to exit the path.

The PA units within the hidden layer of O'Connell's SRN were configured to receive inputs at regular time intervals, such that the network is forced to allocate different groups of hidden units to different units of time, thereby reducing the computational and representational demand on each hidden unit. [144]. As a result, even if the network initially learns the short-term prediction task, some units will still be available to learn the long-term prediction task as they will not have been fully allocated as short-term predictors and are therefore available to learn more distant temporal dependencies [144]. Section 5.3 describes how PA units can be effectively embedded within an MRN.

5.2 Incorporating Periodically Attentive (PA) Hidden Units into the MRN

The conventional hidden layer of the MRN is replaced with a hidden layer that consists of both PA units and non-PA units as shown in Figure 5.2. The MRN employing this modified hidden layer is referred to as the *Periodically Attentive MRN (PA-MRN)*. The PA hidden units defined as *Time Attentive Units (TAUs)* periodically receive input patterns from an input sequence based on a specific unit of time[1]. While the non-PA hidden units are defined as *Holistic Time Attentive Units (H-TAUs)*, they are similar to the hidden units of a standard MRN that

[1]Note: the time intervals are determined by the total number of phases.

receive input stimuli at every time-step. These H-TAUs are always active and serve to integrate holistic temporal information.

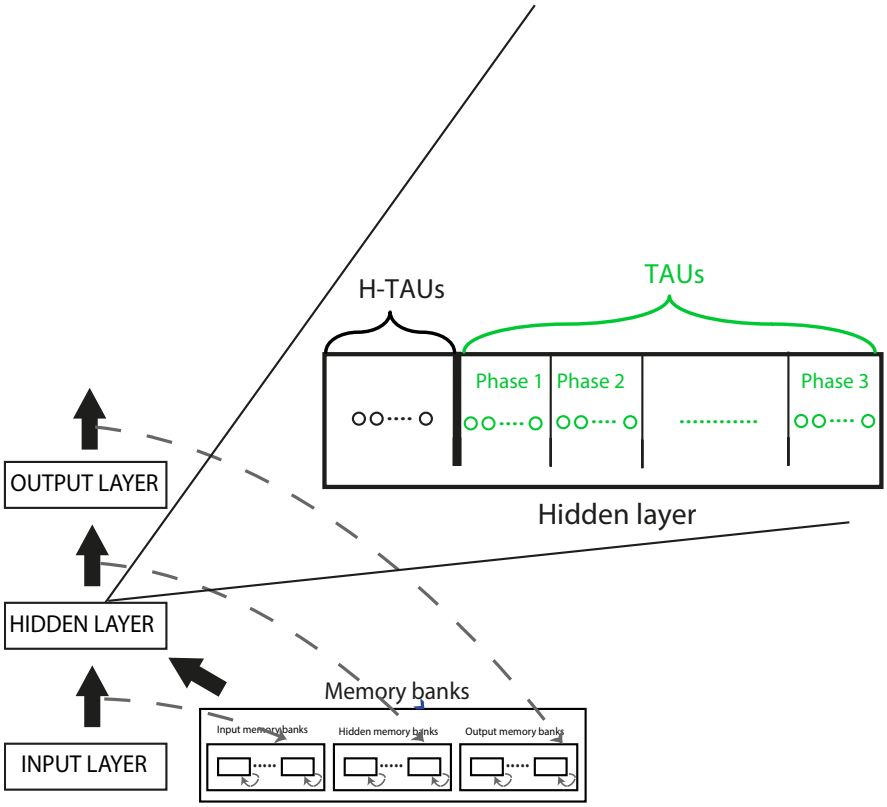


Figure 5.2: The architecture of the self-learning MRN

Employing TAUs reduces the superposition of information encoded by each unit, as the TAUs do not respond at every time-step of a given input sequence, but rather at regular time intervals. An input sequence *(as defined in Section 3.2.1)* of known length is divided into phases, the number of TAUs are then split into the phases[1]. The number of TAUs in each phase is obtained as follows:

$$\text{TAUs per phase} = \frac{\text{total number of TAUs}}{\text{total number of phases}} \tag{5.1}$$

[1]Note: experiments are run with different phases (which inform the time interval) to identify the best for each task. For example, employing 5 phases indicates each phase receive infromation every 5^{th} pattern.

For a given pattern, only the TAUs in the active phase respond, whereas the TAUs across the inactive phases do not respond but maintain the activations from when their respective phases were last active. The total number of phases informs the time attentive sensitivity, which determines how often a phase becomes active to receive input stimuli.

Table 5.1: Example of Periodically Attentive Units (for a window of size 12)

input pattern	Phase 1	Phase 2	Phase 3	H-TAU
start of sequence	-	-	-	-
1	p_1	-	-	p_1
2	^	p_2	-	p_2
3	^	^	p_3	p_3
4	p_4	^	^	p_4
5	^	p_5	^	p_5
6	^	^	p_6	p_6
7	p_7	^	^	p_7
8	^	p_8	^	p_8
9	^	^	p_9	p_9
10	p_{10}	^	^	p_{10}
11	^	p_{11}	^	p_{11}
12	^	^	p_{12}	p_{12}

Consider an input sequence of 12 patterns, where the TAUs are divided into 3 phases *(that is, a time attentive sensitivity of 3)* and 2 H-TAUs. Thus, employing 20 hidden units, 2 of which are H-TAUs gives 18 TAUs for the model. The TAUs are then split into the 3 phases, which gives 6 TAUs per phase. The 2 H-TAUs are responsive at every time-step, whereas the TAUs in each phase are active every 3^{rd} time-step.

Table 5.1 describes the information processing of the input sequence *(Note: p_t refers to the pattern, − indicates the beginning of a new sequence, where the units are set to a known initial value determined by the mid-point of the hidden unit activation function and (^) indicates that the TAUs in that phase remain fixed/unchanged).*

At the beginning of every new input sequence, the hidden *(PA and non-PA)* units are reset to the mid-point of the activation function. Next, the model

receives the 1^{st} input pattern, p_1 and phase 1 is active. The TAUs in phase 1 and the H-TAUs respond to the pattern. The other phases *(2 & 3)* remain inactive and their TAUs are unchanged *(as the activation mid-point)*. Then the model receives the 2^{nd} input pattern, p_2, phase 2 is active and the TAUs in phase 2 respond to the pattern likewise the H-TAUs. The other phases *(1 & 3)* remain inactive and their TAUs are unchanged *(the TAUs in phase 1 remain unchanged from the stimuli received from pattern 1, while the TAUs in phase 3 are unchanged as the activation mid-point)*. For the 3^{rd} input pattern, p_3, phase 3 is active and the TAUs in phase 3 and the H-TAUs respond to the pattern while the TAUs in the inactive phases (1 & 2) remain unchanged. The process is repeated for all patterns in the sequence such that each phase becomes active at every 3^{rd} pattern, allowing the TAUs in that phase to respond to the pattern while the TAUs in the inactive phases remain unchanged. After all the patterns in the sequence have been processed, the weights and biases are updated using the BPTT as described in Section 3.1.2.2.

5.2.1 PA units to tackle catastrophic interference and vanishing gradient problem

Catastrophic Interference results in the severe exponential loss of learned information caused by the representational overlap in the network [38]. This highlights a problem inherent in ANNs, limited learning capacity, such that learning new information once the capacity is reached leads to interference, which affects the network's ability to 'remember' stored information [38]. In particular, for time-series tasks, temporal dependencies between the current time and the past *(near past & distant past)* could have multiple sub-tasks; for example, long-term, short-term or medium-term. Employing TAUs split into phases *(which respond at regular time intervals)* within the hidden layer, induces, encourages and enables the network to learn different representations in each phase. Thus, preventing simultaneous learning of multiple tasks/sub-tasks *(as would occur in the standard MRN)* and as a result, mitigating catastrophic interference during learning.

Moreover, when a phase is active, the information stored and the representations learnt in other phases are 'frozen' until they are active. The phases become

active periodically and as such the TAUs in the phases can only be updated periodically. Therefore, enabling the TAUs in each phase to retain their learned representations for longer periods of time. This minimises the rate at which information decays and temporal dependencies over longer periods of time can be maintained and utilised for learning, thus, mitigating the vanishing gradient problem.

5.3 Results & Analysis

In this section, the MRN extension is applied to the four of the time-series tasks (Business cycle prediction, Oil price prediction, M3 Sales prediction and Covid-19 forecasting) presented in Chapter 3 and their performance is compared to the standard MRN[1]. The following metrics are used to evaluate the model for the different tasks, Matthew Correlation Coefficient (MCC) *(for Business cycle prediction)*, Root-Mean Squared Error (RMSE) *(for Oil price and M3 Competition prediction)* and Mean Absolute Percentage Error (MAPE) *(for Covid-19 forecasting)*. The results are presented along with the percentage Improvement over the standard MRN (IoMRN). The IoMRN is calculated as follows:

$$IoMRN = \frac{\text{PA-MRN score - MRN score}}{\text{MRN score}} \tag{5.2}$$

The memory order (for the memory banks) employed for the standard MRN is 4 (as stated in Section 3.1.1.2). As explained, the total number of memory bank combination trained is determined by the memory order. To determine the best values for the PA-MRN specific hyper-parameters *(H-TAUs, TAUs and phases)*, preliminary experiments employing different sets of hyper-parameters are conducted. Thus, keeping the number of PA-MRN models required for training to a minimum. Appendix A details the results of the preliminary experiments conducted to identify the best values of the PA-MRN specific hyper-parameters, which inform the experiments undertaken in this section.

Note: one of the key benefits offered by the MRN over state-of-the-art LSTM is its ability to learn a task with significantly less adjustable parameters (see Table

[1]The MRN as presented in Chapter 3

4.7). Therefore, for the purpose of this research (obtaining efficient models for effective time-series processing), the proposed MRN variant, the PA-MRN, is trained with no more than 30 hidden units (10 additional units more than the standard MRN).

5.3.1 Business cycle prediction

The PA-MRN is endowed with a hidden layer of periodically attentive units as described above. The hyper-parameters associated with the best MRN models for the business cycle prediction task are used as the initial starting point for the PA-MRN (see preliminary experiments in Appendix A.1). Table 5.2 presents the best models and their associated PA-MRN specific hyper-parameters for each dataset *(Growth, COD, Growth & COD)*. Experiments for this section are then conducted with the best hyper-parameter sets for each dataset.

Table 5.2: Best hyper-parameters for the PA-MRN

Dataset	Units	MCC	Window size	H-TAUs	TAUs	Phases
Growth	20	0.736	20	2	18	5
COD		0.758	20	10	10	5
Growth & COD		0.769	20	2	18	10

In Table 5.3 the best MCC scores for the standard MRN and PA-MRN models are presented and as seen from the table, the standard MRN outperformed the PA-MRN for all three datasets.

Table 5.3: Best MCC score for the NBER turning points prediction task

Model	Units	Dataset		
		Growth	COD	Growth & COD
MRN	20	0.787	0.79	0.799
PA-MRN	20	0.732	0.746	0.794

From Table 5.2, the best models for the Growth, COD and Growth & COD datasets allocate a minimum of 3, 2 and 1 TAU(s) per phase *respectively*. To understand whether the TAUs in each phase are sufficient to learn the task, the

PA-MRN models are trained with 10 additional units so that more TAUs are available for allocation to the phases.

Preliminary experiments are first conducted for the PA-MRN models with 10 additional units (see Appendix A.1). Table 5.4 presents the hyper-parameters associated with the best models employing 10 additional units.

Table 5.4: Best hyper-parameters for the PA-MRN

Dataset	Units	MCC	Window size	H-TAUs	TAUs	Phases
Growth	30	0.747	20	0	30	5
COD		0.747	80	2	28	10
Growth & COD		0.792	20	10	20	5

Table 5.5 presents the MCC score for the best standard MRN and PA-MRN *(with 20 & 30 units)* models. The PA-MRN models employing 10 additional units *(that is 30 units in total)*, perform better than the PA-MRN models employing 20 units for the first two datasets *(Growth & COD)* and maintains the performance for the third dataset.

Table 5.5: Best MCC score for the NBER turning points prediction task

Model	Units	Dataset		
		Growth	COD	Growth & COD
MRN	20	0.787	0.79	0.799
PA-MRN	20	0.732	0.746	0.794
	30	0.751	0.757	0.794

As seen from the hyper-parameters in Table 5.4, the best models for the Growth, COD and Growth & COD datasets allocate a minimum of 6, 2 and 4 TAUs per phase *respectively*. It appears that a critical component for the PA-MRN models to learn the NBER turning points task is a sufficient number of TAUs supplied to each phase.

Despite the underperformance with the PA-MRN models compared to the standard MRN models (see Table 5.6, the PA-MRN models in general performed best with a smaller window size of 20. This highlights the ability for these models to learn temporal dependencies with limited data while still obtaining good

Table 5.6: Improvement over the standard MRN (IoMRN) *(%)*

Dataset	Growth	COD	Growth & COD
MRN	0.787	0.79	0.799
PA-MRN	0.751	0.757	0.793
IoMRN	-4.57%	-4.18%	-0.75%

results. In particular, the results indicate that providing sufficient TAUs in each phase is key for this model variant.

5.3.2 Oil price prediction

The PA-MRN model is applied to the oil price prediction task, to understand whether employing TAUs which enforce and inform resource allocation encourages better learning and enhances performance. Preliminary experiments are conducted to identify the hyper-parameters[1] associated with the best models for each horizon *(1, 3, 6, 12) (see Appendix A.2 for the results of all the models)*. The results for the best models are presented in Table 5.7.

Table 5.7: Best hyper-parameters for the PA-MRN

Horizon	Units	RMSE	Window size	H-TAUs	TAUs	Phases
1	20	0.378	60	10	10	3
3		0.699	120	10	10	6
6		0.92	120	2	18	5
12		1.141	120	2	18	3

The hyper-parameters of the best models are then used to train the PA-MRN for the oil price prediction task and the results for the best models are presented in Table 5.8.

For the PA-MRN models employing 20 units, all but one of the best models have a minimum of 3 TAUs in each phase. To understand whether the TAUs per phase are sufficient to learn the task, the PA-MRN models are trained further with 10 additional units (that is a total of 30 units), so that there are more TAUs available for allocation. The best models for the experiments conducted

[1]Note: the hyper-parameters associated with the best standard MRN models are employed as a starting point to train the PA-MRN.

Table 5.8: Best RMSE scores for oil price prediction

Model	Ratios	Horizon			
		1	3	6	12
MRN	20	0.321	0.693	0.864	0.892
PA-MRN	20	0.329	**0.679**	0.883	0.995

for the PA-MRN models with 10 additional units are presented in Table 5.9 *(see Appendix A.2 for the results of all the models).*

Table 5.9: Best hyper-parameters for the PA-MRN

Horizon	Units	RMSE	Window size	H-TAUs	TAUs	Phases
1	30	0.429	60	10	20	3
3		0.674	120	0	30	5
6		0.922	120	10	20	6
12		1.2	120	2	28	6

The best PA-MRN models *(with 30 units)* for each horizon employ a minimum of 3 TAUs in each phase. More specifically, the best PA-MRN models for horizon of 1, 3, 6 and 12 employ a minimum of 6, 6, 3 and 4 TAUs per phase *respectively.* These hyper-parameters are then used to train the PA-MRN with 10 additional units for each horizon and the result for the best models are presented in Table 5.10.

Table 5.10: Best RMSE scores for oil price prediction

Model	Ratios	Horizon			
		1	3	6	12
MRN	20	0.321	0.693	0.864	0.892
PA-MRN	20	0.329	**0.679**	0.883	0.995
	30	0.334	0.705	0.883	1.058

As shown in the table, the PA-MRN offers an improvement in one instance *(an horizon of 3).* Employing (10) additional units with the PA-MRN for this task leads to a slight drop in performance for the remaining horizons which is indicative of over-fitting. It appears for the oil price prediction task, 3 TAUs are sufficient. Results in this section and in Section 5.3.1 highlight the need for an

appropriate number of TAUs per phase, as undersupplying or oversupplying the number of TAUs in each phase affects the performance of the models.

Table 5.11: Improvement over the standard MRN (IoMRN) *(%)*

Horizon	1	3	6	12
MRN	0.321	0.693	0.864	0.892
PA-MRN	0.329	0.679	0.883	0.995
IoMRN	-2.49%	2.02%	-2.2%	-11.6%

As shown in Table 5.7 and Table 5.9, for an horizon of 1, the best models employed a window size of 60 while for the remaining horizons *(3, 6, 12)*, the best models employed a window size of 120. Similar to the results obtained in Section 5.3.1, the PA-MRN models performed best with the smaller window size compared to the standard MRN models *(see Table 4.7 for the best window size for the standard MRN)* which is well suited for problems with limited data.

5.3.3 M3 competition prediction

The PA-MRN is applied to the 10 random series from the M3 competition data *(presented in Section 3.5.3)* and benchmarked against the standard MRN to assess its performance. Results from Section 5.3.1 and Section 5.3.2 highlight the need for sufficient TAUs in each phase. Thus, given the models employ 10 hidden units, the PA-MRN models are trained with these PA specific hyper-parameters, two H-TAUs [0, 2] and one phase [3], thus the models can have a minimum of either 2 or 3 TAUs per phase. Increasing the number of H-TAUs or phases reduces the number of TAUs allocated to each phase, therefore the parameter search space with the H-TAUs and phases is restricted to those aforementioned.

The results for the best models are presented in Table 5.12 where the PA-MRN models performed better than the standard MRN models for six series. However, for the remaining four series, particularly *N1908*, there is a significant decline in performance with the PA-MRN. Interestingly, for series *N1908*, the alternate models presented in Table 4.9 performed better than the MRN, suggesting the standard MRN and PA-MRN have difficulties mapping this series.

To identify whether the PA-MRN had sufficient TAUs, the models are given

Table 5.12: Best RMSE scores for M3 sales prediction

Model	Units	H-TAUs	Series				
			N2516	N2521	N1807	N1908	N2012
MRN	10	10	187.1	2017.3	268.3	453.9	517.8
PA MRN	10	0	196.9	**2009.6**	279.2	644.1	573.6
		2	201.6	2011.5	294	622.8	555.6

Model	Units	H-TAUs	Series				
			N2159	N2158	N2150	N2144	N1918
MRN	10	10	521.9	651.1	141.6	538	206.7
PA MRN	10	0	492.7	538.3	**129.8**	**432.1**	197
		2	**492.5**	**534.3**	134.5	441.4	**172.7**

4/5 TAUs per phase, thus increasing the total number of units to 15. The results of the best models with 15 units are presented in Table 5.13. Employing more TAUs enhanced the performance for six series. For three of the remaining four series, there was a slight drop in performance and finally for series *N1908*, there was a significant drop in performance.

Table 5.13: Best RMSE scores for M3 sales prediction

Model	Units	Series				
		N2516	N2521	N1807	N1908	N2012
PA MRN	10	**196.9**	**2009.6**	279.2	**622.8**	555.6
	15	199.3	2013.3	**256.8**	669.5	**533.4**

Model	Units	Series				
		N2159	N2158	N2150	N2144	N1918
PA MRN	10	492.5	534.3	**129.8**	432.1	172.7
	15	**488.1**	**531.3**	131.1	**422.5**	**162.7**

Figure 5.3 presents the predictions of the best standard MRN and PA-MRN models for series *N1918*, where the overall best PA-MRN model had the best improvement. The MRN has difficulty mapping the signal and appears to predict values that are smoothed out and averaged. The PA-MRN on the other hand, appears to offer more meaningful predictions than the MRN. It maps the signal,

albeit on a much smaller scale and in advance of the third trough.

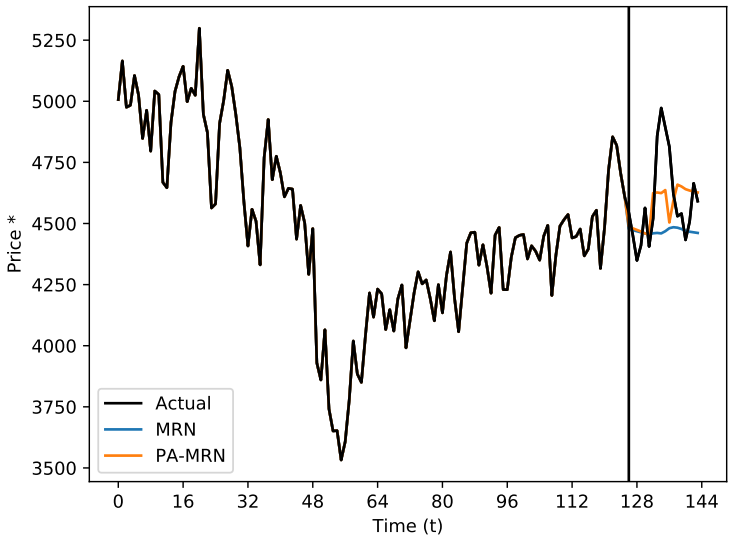


Figure 5.3: Series N1918

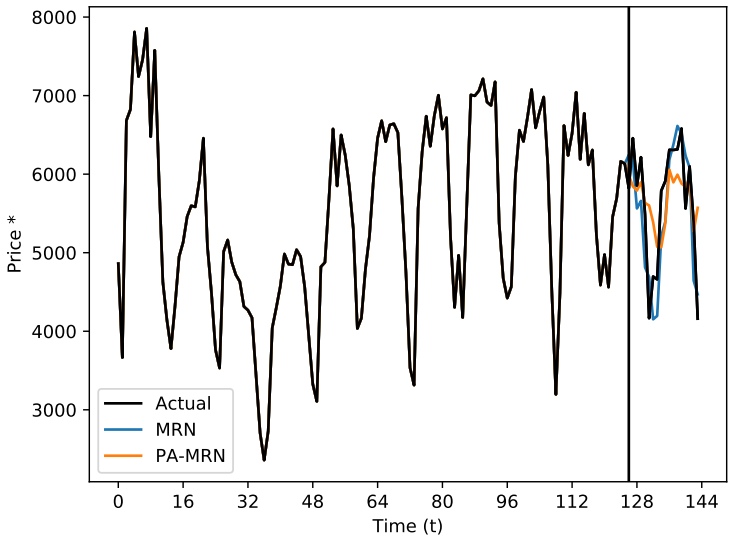


Figure 5.4: Series N1908

Figure 5.4 presents the predictions of the best standard MRN and PA-MRN

models for series *N1908*, where the overall best PA-MRN model had the worst performance. As seen from the figure, the PA-MRN appears to map the signal for the first trough but on a smaller scale, and it appears to identify the peak but does not map it as well as the standard MRN.

Table 5.14: Improvement over the standard MRN (IoMRN) *(%)*

Series	N2516	N2521	N1807	N1908	N2012
MRN	187.1	2017.3	268.3	453.9	517.8
PA-MRN	196.9	2009.6	256.8	622.8	533.4
IoMRN	-5.24%	0.38%	4.29%	-37.2%	-3.01%

Series	N2159	N2158	N2150	N2144	N1918	Average
MRN	521.9	651.1	141.6	538	206.7	550.36
PA-MRN	488.1	531.3	129.8	422.5	162.7	535.39
IoMRN	6.48%	18.4%	8.33%	21.5%	21.3%	-

The PA-MRN models offer significant improvement *(up to 21.5%)* over the standard MRN models as seen from Table 5.14 for seven of the ten series. However, for the remaining three series, there was a decline in performance, in particular, for series *N1908*, there is a significant decline of 37.2%. As shown in Table 5.14, the average for all the series is calculated *(as in [119])* and as seen the PA-MRN has a lower overall average than the MRN.Nonetheless, the PA-MRN model demonstrates strong abilities and offers improvement over the standard MRN in most cases for the M3 prediction task.

A simple t-test is conducted as in Section 4.2.3, The null hypothesis H_0: The mean difference between the PA-MRN and the MRN is greater than or equal to zero ($H_0 : \bar{\mu_d} \geq 0$) that is the PA-MRN is not significantly different from the MRN. Thus, the alternate hypothesis H_A: The mean difference between the PA-MRN and the MRN is less than zero ($H_A : \bar{\mu_d} < 0$) that is the PA-MRN is not significantly different from the MRN.

The RMSE difference between PA-MRN and the MRN is calculated and the mean and the standard deviation is calculated from the sample of RMSE differences (shown in Table 5.15). Using a left-tail t-test, the critical value $t_{9,0.05}$ (with a degree of freedom of 9 at 5% significance level) = -1.8331, the null hypothesis,

Table 5.15: T-test for the models

RMSE difference	PA-MRN/MRN
Mean	-149.8
Sample standard deviation	80.1
t	-5.9

H_0 is rejected and thus the alternative hypothesis is accepted, concluding that the PA-MRN does enhance performance.

5.3.4 Covid-19 forecasting

In this section, the PA-MRN model is applied for the Covid-19 forecasting of confirmed and death cases in the United States of America. Preliminary experiments are run with a combination of hyper-parameters[1] (all the results are presented in Appendix A.3). The best hyper-parameters *(Confirmed, & Death)* from the preliminary experiments are presented in Table 5.16. Experiments for this section are then conducted with the best hyper-parameter for the cases.

Table 5.16: Best hyper-parameters for the PA-MRN

Cases	Units	MAPE	Window size	H-TAUs	TAUs	Phases
Confirmed	20	4.57	35	0	20	5
Death		0.66	25	0	20	10

Table 5.17 presents the results of the best models, as seen, the PA-MRN performed better than the standard MRN for the death cases. However, for the confirmed cases the standard MRN performed better than the PA-MRN. From Table 5.16, the best PA-MRN models employed roughly 4, 2 TAUs per phase for the confirmed and death cases *respectively*. The PA-MRN is trained with 10 additional units so that more TAUs can be allocated to the phases. Preliminary

[1]The hyper-parameters of the best MRN model for Covid-19 forecasting is used as the initial starting point for the PA-MRN

results were run to identify the best PA specific hyper-parameters and the results are presented in Table 5.18 *(see Appendix A.3 for all the results).*

Table 5.17: Best MAPE score for Covid-19 forecasting

Model	Hidden units	confirmed	death
MRN	20	4.11	0.3
PA-MRN	20	4.6	**0.26**

As seen from Table 5.18, the best PA-MRN with 30 units employed a minimum of 3, 4 TAUs per phase for the confirmed and death cases *respectively*. These hyper-parameters are then used to train the PA-MRN models. The results of the best models are presented in Table 5.19. Employing additional units, which provided more TAUs per phase, led to an improvement in performance for the confirmed cases. Whereas for the death cases, although employing additional units increased the number of TAUs per phase, there was a drop in performance, which was likely due to over-fitting.

Table 5.18: Best hyper-parameters for the PA-MRN

Cases	Units	MAPE	Window size	H-TAUs	TAUs	Phases
Confirmed	30	4.56	35	10	20	6
Death		0.81	25	2	28	6

Table 5.19: Best MAPE score for Covid-19 forecasting

Model	Hidden units	confirmed	death
MRN	20	4.11	0.3
PA-MRN	20	4.6	**0.26**
	30	**3.6**	0.58

For the confirmed cases, Figure 5.5 visualises the predictions of the best models. The PA-MRN appears to make slightly closer predictions to the observed values than the MRN from early June to mid-June. Both models miss the sharp spike in cases around mid-June, the standard MRN does however appear to have closer predictions to the observed number of cases from mid-June to late-June

Table 5.20: Improvement over the standard MRN (IoMRN) *(%)*

Series	Confirmed cases	Death cases
MRN	4.11	0.3
PA-MRN	4.01	0.26
IoMRN	2.43%	13.33%

than the PA-MRN. For the death cases, the predictions for the best standard MRN and PA-MRN models are shown in Figure 5.6. Both models appear to provide very similar predictions to the observed values and follow the underlying signal and trend of the death cases.

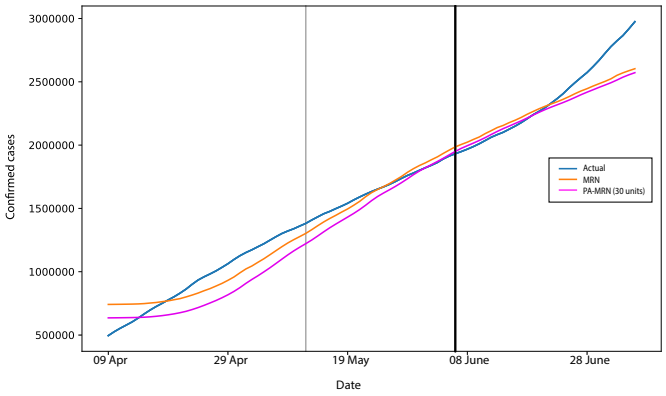


Figure 5.5: Best models for Covid-19 forecasting *(Confirmed cases)*

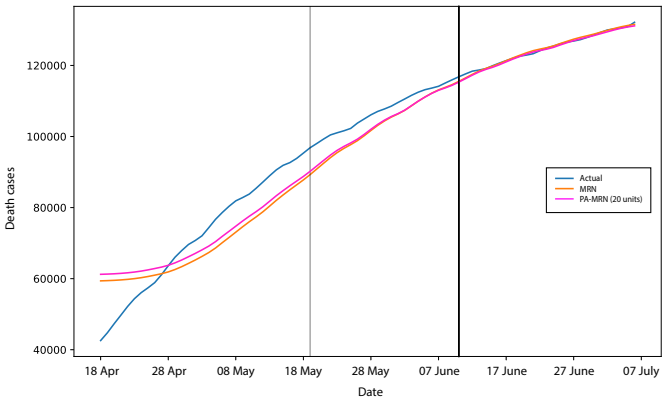


Figure 5.6: Best models for Covid-19 forecasting *(Death cases)*

The PA-MRN offers 2.4% - 13.33% improvement over the standard MRN

for the Covid-19 forecasting of confirmed and death cases as seen from Table 5.20. Overall, the PA-MRN model appears to enhance performance and offers an improvement over the standard MRN for Covid-19 forecasting.

5.4 Discussion

The MRN has been shown to add predictive value over other neural networks and traditional statistical models, it demonstrated superiority as seen from [20, 149, 183, 200, 208] and in Chapter 4. The superiority of the MRN has been further explored and enhanced in this chapter, by endowing it with periodically attentive units to address two key learning limitations; vanishing gradient descent and catastrophic interference, associated with most types of ANNs. A brief discussion is now given with a view to drawing conclusions across the experiments undertaken in this chapter.

5.4.1 Periodic attentiveness to mitigate gradient descent problem and catastrophic interference

The PA-MRN model offered some key advantages over the standard MRN model as indicated by the result obtained, *in general, it outperformed for the different tasks and problems)*. However, there were some cases where the PA-MRN offered no improvements, for example, the NBER turning points prediction task. Results indicated that there were likely insufficient units in each phase and increasing the total number of hidden units, which is directly related to the number of TAUs available for allocation in each phase, led to an improvement in performance. However, for the purpose of this research, to obtain computationally effective models and due to the large search space for the memory bank combinations, the number of hidden units for the PA-MRN was restricted *(to a maximum of 15 for the M3 Sales prediction & 30 for the remaining tasks)*.

To understand the benefits of the PA-MRN, the best standard MRN models are trained with 10 additional units, to understand whether the improvements gained with the PA-MRN can be simply obtained by increasing the number of hidden units, and the results are presented in Table 5.21 and Table 5.22.

Table 5.21: MRN models trained with additional hidden units

Dataset	Oil price				NBER			Covid-19	
Model	1	3	6	12	Growth	COD	G & COD	con.	death
PA-MRN	0.334	**0.68**	**0.883**	**0.995**	0.751	0.757	**0.793**	**4.01**	**0.26**
MRN *(30)*	0.319	0.71	1.24	1.13	0.775	0.763	0.78	4.73	4.92

As seen from Table 5.21, the MRN with additional units is outperformed by the PA-MRN in seven instances. This indicates that the MRN with additional units[1] is likely over-fitting and thus, a decline in performance is observed. Similarly, as seen from Table 5.22, the PA-MRN performs best for six of the ten series.

Table 5.22: MRN models trained with additional hidden units

	Series									
Model	2516	2521	1807	1908	2012	2159	2158	2150	2144	1918
PA-MRN	199.2	**2006.6**	336.9	656.5	525.2	**482.4**	**534.4**	**123.1**	**429**	**154.8**
MRN *(20)*	186.2	2015.6	261.5	470.3	468.9	576.8	619.3	145.7	508.7	205.6

In general, the results indicate that merely increasing the hidden units is not sufficient to enhance performance and extensions such as the PA-MRN introduced in this chapter are necessary to enhance performance. Experiments conducted conclusively demonstrated that in some cases the PA-MRN does generally offer improvement over the standard MRN, however, as seen in Section 5.4, an appropriate number of TAUs in each phase is necessary for the PA-MRN.

[1]Note: the models are trained using the hyper-parameters of the best models. NBER prediction task; learning rate: 0.09, momentum: 0.9999, epochs: 1000 *(see Table 4.2 for the best memory bank combination employed for each dataset along with best window size)*. Oil price prediction; learning rate: 0.01, momentum: 0.999, epochs: 1000 *(see Table 4.7 for the best memory bank combination and window size employed for each horizon)*. M3 sales prediction; learning rate: 0.01, momentum: 0.999, epochs: 1000 *(see Table B.1* for the best memory bank combination and window size employed for each series). Covid-19 forecasting; learning rate: 0.01, momentum: 0.999, epochs: 500 *(see Table B.2 for the best memory bank combination and window size employed for each series)*.

5.4.2 Search space for memory bank configuration

As described in Section 3.1.1.2, to obtain the best model for the standard MRN, a number of memory bank combinations are run, the total number of which is determined by the memory order. Due to the non-exhuastive search space for the memory bank combinations, the memory order (that is the maximum number of memory bank in each memory bank type), is constrained to 4. Similarly, to obtain the best PA-MRN models, the memory order which determines the total number of memory bank combinations required for training, is constrained to 4. More specifically, for a memory order of 4 *(that is each memory bank type can have 0, 2, 3 or 4 memory banks)*, for a given task, the total number of possible models combinations is $4^3 = 64$. Note: the memory bank configuration, [0, 0, 0], is not included *(as this represents a standard RNN model without a memory mechanism)*, thus, the total number of models required for training is 63. This presents a major limitation for tasks or problems which require a higher number of memory banks.

5.5 Conclusion

In this chapter, the MRN was extended and endowed with periodic attentive units and compared to the standard MRN. The MRN extension; *the PA-MRN* is applied to four time-series forecasting tasks, to assess whether it mitigates two key learning limitations of the standard MRN such that better learning and information latching is encouraged and as such performance is enhanced.

The results presented demonstrated that the PA-MRN can enhance performance in some cases. In particular, the PA-MRN models offered an improvement over the standard MRN of up to **21.5%** for the experiments undertaken. The experiments conclusively indicated that the MRN can be endowed and in some cases this endowment aided the mitigation of the limitations accounted with using gradient descent training (gradient descent problem and catastrophic interference).

However, the PA-MRN similar to the standard MRN and other machine learning techniques requires a lengthy hyperparameter tuning process (particularly for

the memory mechanism) to obtain an 'optimal' model. There is a non-exhuastive search space to be explored to identify the best models and as such the memory order is restricted. In the following chapter, an extension that addresses this architectural limitation of lengthy hyper-parameter tuning (specifically for the ratios in the memory) is presented.

5.6 Chapter contributions

This is the first work to explore and incorporate attentive mechanisms into the MRN. Specifically, the methodology in [144] is extended to the MRN to i) overcome the vanishing gradient problem by leaving unit representations from previous time steps active for longer (enabling further embedding within the memory banks) and ii) reduce catastrophic interference by dedicating hidden units to specific phases of time.

Chapter 6

Self-learning in the Multi-recurrent Network

In this chapter, a novel technique is introduced to mitigate an architectural limitation associated with the memory rigidity, so that the requirement of designer input for ratio hyper-parameters within the model can be eliminated. More specifically, self-learning of the recurrent link ratios are introduced, to inform the memory composition, by allowing the ratio of current and past information to be determined by the learning process rather than established *a priori*. This extension will be applied and compared to the standard MRN across four of the different problem domains introduced in Chapter 3 to understand the impact of these innovations on the behavioural dynamics of the MRN.

6.1 Background

Similar to many ANN variants, the MRN requires manual hyper-parameter tuning by the user. For example, Ulbricht's original MRN requires the designer to determine both the learning hyper-parameters (learning rate and momentum) and architectural hyper-parameters (number of hidden units, number of memory banks for each type and the weightings for both self- and layer-recurrency link ratios) [208]. Whilst there are hyper-parameter selection methods such as grid search [60, 120] and randomized searches of hyper-parameter spaces [18, 128], the

process of hyper-parameter tuning is very time-consuming. For this thesis, a technique would be sought to reduce the hyper-parameter tuning process associated with the layer-link ratios (which inform the memory composition) and possibly improve the memory quality, thus, enhancing the MRN's ability for time-series processing. Therefore, in this chapter, the general aim is to extend the standard MRN to specifically overcome the above-mentioned architectural limitation. To resolve this limitation, the architecture of the MRN will be extended by incorporating self-learning.

6.2 Self-Learning in ANNs

Authors such as Hartstein and Koch [79], Nguyen and Widrow [141, 142], Li *et al.* [118], Yasunaga *et al.* [230], Chen *et al.* [29] and Konecný *et al.* [107] have investigated and employed various self-learning schemes within ANNs achieving varying degrees of success.

Hartstein and Koch [79] published their work in 1989, which highlighted self-learning attributes within ANNs. They utilised the characteristics of a MOSFET[1] device which produces an output current from a linear function of an input voltage relative to a threshold voltage, to develop a more compact network [79]. Using the MOSFET characteristics, the output of each neuron is calculated as the summation of all the inputs relative to the respective learnt thresholds, which is then passed through a sigmoid function [79].

More specifically, the learnt thresholds are calculated as a function of the previous thresholds added to a proportion of the difference between the respective neuron outputs and neuron inputs minus a constant, these thresholds hold the memories of the network [79]. They then performed simulations in two phases with this type of neuron. The first phase was the learning phase, where thresholds were adjusted by a given number of random patterns. In the second phase, the learning was turned off and the memories were probed to identify their essential features, and how well the network learnt and performed on test patterns [79]. They found that the network was able to learn all the input patterns for a reasonable number of random patterns.

[1] Metal–Oxide–Semiconductor Field-Effect Transistor

They also demonstrated that the network had the ability to learn the inverse patterns as a result of network symmetry [79]. Their work highlighted the inherent self-learning and predictive abilities of artificial neural networks, particularly, as the network learns from the given neuron inputs and the thresholds. They viewed and demonstrated the concept of self-learning in their network as the dynamic learning ability of a network irrespective of its synaptic function. Hartstein and Koch [79] conclusively showed that ANNs in and of themselves are robust and possess innate self-learning abilities.

Similarly, Nguyen and Widrow [142] using The Truck Backer-Upper example demonstrated that their neural network can learn on its own accord and achieve nonlinear controller design to dock a truck. Ideally, when a truck driver is backing up a truck, a combination of forward and backward movements are required to successfully dock the truck. They set an objective for the system to successfully back the truck to the dock, *such that, the point (x trailer, y trailer) are aligned as closely as possible with the point (x dock, y dock)* using only backward movements [142]. The neural network employed is first trained to be an emulator of the truck and trailer dynamics, and then the neural network controller is trained to control the emulator. Results showed that the truck emulator had the ability to represent the trailer and truck, when 'jackknifed', in line or in any condition in between [142]. More specifically, the results highlighted the self-learning abilities of neural networks, without which, a substantial amount of user input and design would have been required [142]. Others such as Li *et al.* [118] demonstrated by incorporating active and self-learning to a deep Convolution Neural Network, the limitations associated with limited data are alleviated. The active- and self-learning worked in an iterative manner involving "(re)training the temporal-ensembling deep CNN and updating informatively unlabelled samples within pseudo-labels to the training dataset" [118]. They showed that through the self-learning process the model is able to query, subsequently filter and label samples by a multi-scale spatial constraint without additional labelling from experts. This enabled the training of only one model, which had the ability to learn the variability, thus reducing false alarms and increasing precision [118]. Konecný *et al.* [107] used a self-learning artificial neural network to classify the financial situation of enterprises (where the sets of objects of the particular classes are not

well-known). They demonstrated that employing a self-learning network negated the need for user input.

6.2.1 Self-learning and Recurrent Networks

Embedding self-learning mechanisms in ANNs has proven to be successful. More specifically, research on self-learning in RNNs appears to mainly occur in the 'forward' layers. For example, Bouchachia and Ortner [23] implemented the idea of self-learning in an RNN, Recursive Neural Network[1], by adopting a semi-supervised learning approach through self-learning for their dataset *(comprised of both labelled and unlabelled data)*, so that the network possesses the ability to actively pre-label data. They thus coerced the network to harness its structure for extracting and learning from the limited labelled data. The authors explained the approach as follows: *"Self-learning attributes are introduced such that the networks estimate a posterior probability function for each class, and assign an unlabelled pattern to the class for which the posterior probability of membership is the highest"* [23]. Using labelled and unlabelled data, their work demonstrated that RNNs possess self-learning abilities and can harness their structure to enhance performance [23].

More specifically, there remains limited work on self-learning in RNNs, particularly for their context/recurrent layer(s). As a result, the potential benefits achieved with self-learning, do not appear to be fully realised for RNNs. These layers are essential for effective processing of temporal data. They particularly serve as the network's 'memory', allowing the network to 'remember' past information. Thus, optimising these recurrent 'memory' layers where possible is paramount. In this chapter, endowing self-learning within the MRN's memory mechanism *(the key to its superiority)* is subsequently explored.

[1] "A neuron in the Recursive Neural Network is an extension of a neuron in a one-level Recurrent Neural Network, where instead of just considering the output of the neuron in the previous time step (like with the recurrent neuron), the recursive node gets signals from its subtrees." [23]

6.3 Introducing Ratio Learning in the MRN

In this section, a new self-learning approach is introduced whereby the MRN is extended by incorporating additional units, *Ratio Control Units (RCUs)*, that learn to determine optimal layer-recurrency link ratios for the memory banks. *(Note, the self-link ratios are calculated as described in Section 3.1.1.1.)* As presented in the subsections below, three techniques are proposed to determine memory composition and identify whether this can enhance performance. The MRN with self-learning mechanisms is referred to as the *Self-learning MRN (SL-MRN).*

6.3.1 Learning the layer-recurrency link ratios with additional hidden units *(SL-MRN 1)*

For the first approach, the RCUs are incorporated in the hidden layer as shown in Figure 6.1 *(the dashed lines represent the copying of the layers and the blue lines represent the updating of the relevant ratios).* Thus, the total number of units in the hidden layer of the proposed SL-MRN increases in proportion to the total number of memory banks in each type. *(Note: these RCUs are treated like the other hidden units).* Therefore, the SL-MRN naturally has a larger hidden layer than the standard MRN; this SL-MRN variant is referred to as *SL-MRN 1.*

Training in SL-MRN 1 is similar to that of the standard MRN *(as described in Section 3.1.2.1)*, it does however include additional calculations with the RCUs during the forward pass. After a pattern is passed forward, the net RCUs are calculated and passed through a sigmoid function like the other hidden units, *(this maintains the proportional assumption of the ratio values, ensuring values are fixed between 0 and 1).* The RCUs *(and subsequently the ratios)* are then updated based on the input stimuli received from the current pattern and memory, and are then used to update the memory, which is fed with the next pattern to the input layer[1]. The input, hidden and output memories at time t, M_{t_i}, M_{t_h} and M_{t_o} are updated *(such that Equation 3.1 - Equation 3.3 are modified)* as follows:

[1]Note: the ratios are readjusted during training and as new information is presented during testing.

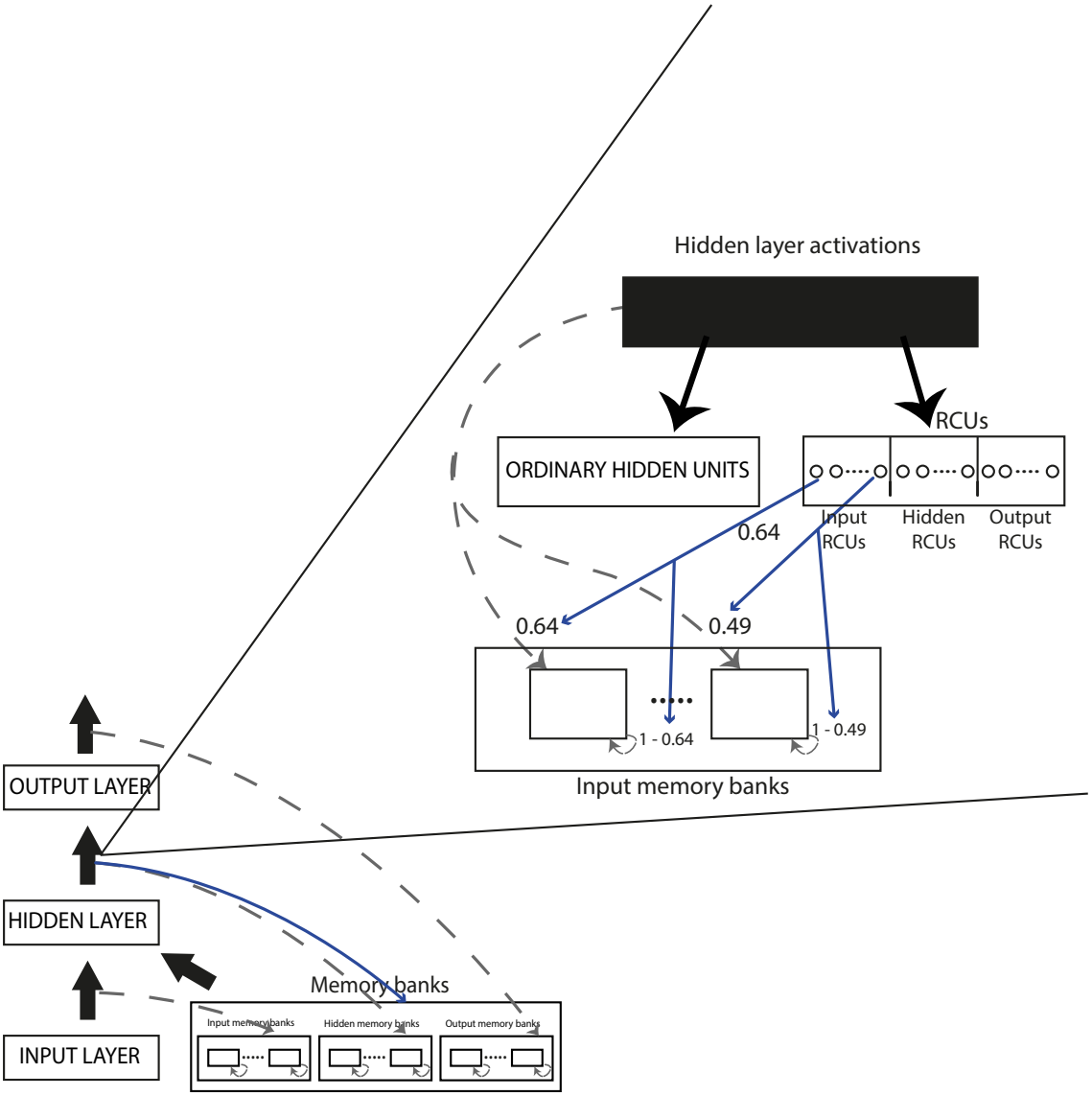


Figure 6.1: The memory architecture of the self-learning MRN with additional hidden units, called ratio control units, controlling the recurrent links for the input memory banks.

$$
\begin{aligned}
M_{t_i} &= (RCU_{t_i} \times I_{t-1}) + ((1 - RCU_{t_i}) \times M_{t-1_i}) \\
M_{t_h} &= (RCU_{t_h} \times H_{t-1}) + ((1 - RCU_{t_h}) \times M_{t-1_h}) \\
M_{t_o} &= (RCU_{t_o} \times O_{t-1}) + ((1 - RCU_{t_o}) \times M_{t-1_o})
\end{aligned}
\tag{6.1}
$$

where RCU_{t_i}, RCU_{t_h} and RCU_{t_o} are the RCUs at time t for the input, hidden and the output memory banks *respectively*. I_{t-1}, H_{t-1} and O_{t-1} are the outputs of the input, hidden and output layers at time $t-1$ *respectively*. M_{t-1_i}, M_{t-1_h} and M_{t-1_o} are the input, hidden and output memories at time $t-1$ *respectively*.

After the modified forward pass of an input sequence as described above, the network goes through the next phase of training, BPTT *(as described in Section 3.1.2.2)*. The network is trained for a given number of epochs, after which the learnt weights and biases are fixed and used in the testing phase.

6.3.2 Learning the layer-recurrency link ratios by incorporating ratio units *(SL-MRN 2)*

Rather than augmenting the MRN with additional units in the hidden layer, the second approach endows the MRN with a ratio layer, which holds the RCUs. The RCUs receive information from input patterns and the memory, and also contribute to the final output of the network[1], *as shown in Figure 6.2, where the dashed lines represent the copying of the layers and the blue line represents the updating of the relevant RCUs.* As shown in the *'zoomed in'* aspect of Figure 6.1, the RCUs are used to determine the weightings of the historical information from the network layers *(input, hidden & output)* and the memory layer. This SL-MRN variant is referred to as *SL-MRN 2*.

During training, a pattern from an input sequence is received, and the forward pass occurs as described in Section 3.1.2.1 with the addition of the RCUs calculation. The net RCUs at time t, $\hat{RCU}_t$, are calculated as follows:

$$\hat{RCU}_t = \sum W_{ir} I_t + \sum W_{M_{ir}} M_{t_i} + \sum W_{M_{hr}} M_{t_h} + \sum W_{M_{or}} M_{t_o} + b_r \qquad (6.2)$$

where W_{ir}, $W_{M_{ir}}$, $W_{M_{hr}}$, $W_{M_{or}}$ are the respective weights of the input, input memory, hidden memory and output memory layers to the ratio layer. b_{i_r} are the input layer biases (for the ratio layer), I_t are the outputs of the input layer at time t and M_{t_i}, M_{t_h} and M_{t_o} are the input, hidden and output memories at time t.

[1]Note: the ratios are only readjusted during training and do not change as new information is presented during testing

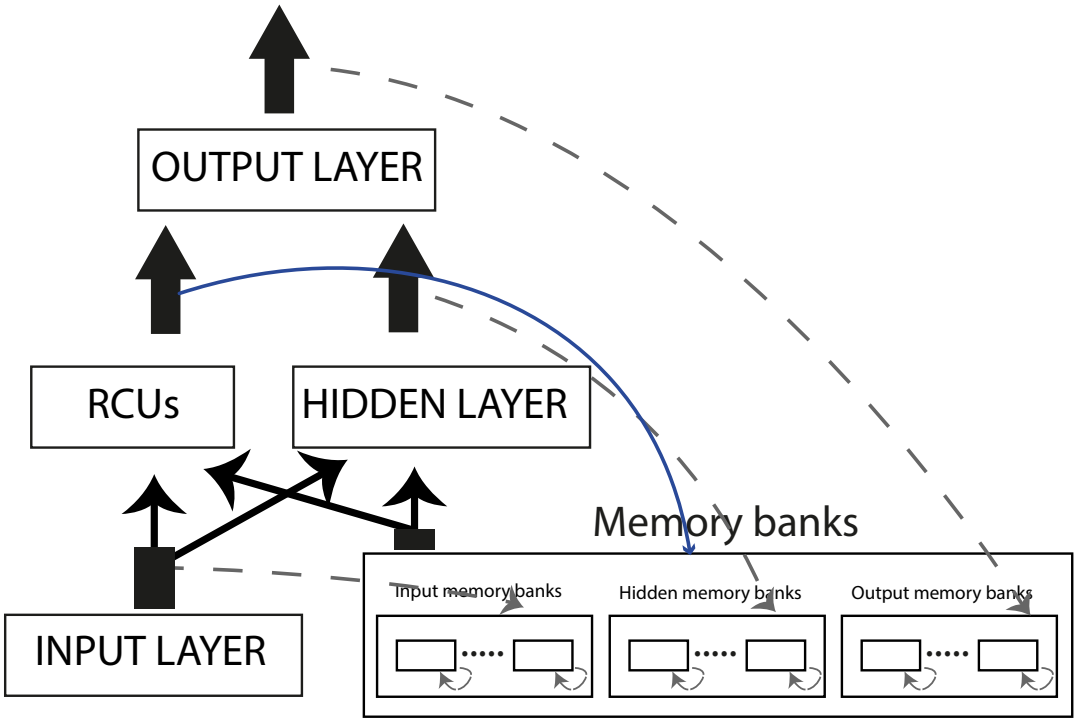


Figure 6.2: The architecture of the self-learning MRN

The RCUs at time t, RCU_t are derived by passing the net RCUs, $\hat{RCU}_t$ through the chosen activation function[1] for the ratio layer, f_r as follows:

$$RCU_t = f_r(\hat{RCU}_t) \tag{6.3}$$

The calculated RCUs at time t, RCU_t, are then used to update the memory *(as shown in Equation 6.1)* for the next pattern. The outputs of the output layer is then updated *(such that Equation 3.6 is modified)* as follows:

$$O_t = \sum W_{ho} H_t + \sum W_{ro} RCU_t + b_k \tag{6.4}$$

[1]Sigmoid activation function is employed for the experiments in this thesis to ensure the assumption of ratio values between 0 and 1 is maintained.

where W_{ho} and W_{ro} are the respective weights of the hidden and ratio layers to the output layer, b_k is the sum of the hidden and ratio layer biases and H_t are the outputs of the hidden layer at time t.

After the forward pass of the input sequence as described above, the next step is the BPTT with some modifications to include the new ratio layer. Similar to Equation 3.10 which shows the calculation to update the hidden layer to output layer weights and hidden layer biases, the ratio layer to output layer weights, W_{ro} and ratio layer biases, b_r are updated as follows:

$$\begin{aligned} W_{ro} &= D_{O_t} * \frac{\partial O}{\partial W_{ro}} = D_{O_t} * RCU_t \\ b_r &= D_{O_t} * \frac{\partial O}{\partial b_r} = D_{O_t} * 1 = D_{O_t} \end{aligned} \tag{6.5}$$

where D_{O_t} is the output layer deltas.

The ratio layer delta at time t, D_{RCU_t} is calculated as follows:

$$\begin{aligned} D_{RCU_t} &= \frac{\partial E}{\partial O}\frac{\partial O}{\partial RCU}\frac{\partial RCU}{\partial \hat{RCU}} + Mem_{t+1} \\ &= (O_t - A_t) * f_r^{-1}(\hat{RCU}) * W_{ro} + Mem_{t+1} \end{aligned} \tag{6.6}$$

where A_t is the actual output at time t and f_r^{-1} is the inverse of the chosen activation function.

The input layer to ratio layer weights, W_{ir}, input biases, b_r and the memory weights $W_{M_{kr}}$ are updated as follows:

$$\begin{aligned} W_{ir} &= D_{RCU_t} * \frac{\partial \hat{RCU}}{\partial W_{ir}} = D_{RCU_t} * I_t \\ b_{i_r} &= D_{RCU_t} * \frac{\partial \hat{RCU}}{\partial b_{i_r}} = D_{RCU_t} * 1 = D_{RCU_t} \\ W_{M_{kr}} &= D_{RCU_t} * \frac{\partial \hat{RCU}}{\partial W_{M_{kr}}} = D_{RCU_t} * M_{t_k} \text{ for } k = i, h, o \end{aligned} \tag{6.7}$$

where i, h and o mean input, hidden and output.

Employing the sliding window approach, Equation 6.5 and Equation 6.7 are modified in line with Equation 3.13 and Equation 3.14. The network is trained

for a given number of epochs, after which the learnt weights and biases are used in the testing phase.

6.3.3 Learning the layer-recurrency link ratios using the error gradients of the ratio units *(SL-MRN 3)*

The third and final approach derives RCU values from the error gradients of the units in the ratio layer *(incorporated in Section 6.3.2)*. The error calculations happen during the BPTT, therefore, the RCUs are only updated after a forward pass of an input sequence (unlike SL-MRN 1 and SL-MRN 2 that are updated after every pattern). Thus, the RCUs account for the collective temporal dependencies in a given window. This SL-MRN variant is referred to as *SL-MRN 3*.

After the forward pass, the errors are calculated using the modified BPTT described in Section 6.3.2 and the weights and biases in the network are updated. The errors for the ratio layer are calculated over the input sequence, so that the error is accrued for every pattern in the sequence. The RCUs are then calculated as follows:

$$\begin{aligned} T &= \sum_{t=1}^{n_s} (\tau * D_{RCU_t}) + (F * T_{t-1}) \\ RCU_t &= \frac{1}{n_s} * T_t \end{aligned} \tag{6.8}$$

where τ is the ratio momentum term, F is the momentum term and n_s is the length of the input sequence *(Note: T is an array to store the errors of the RCUs across the input sequence)*.

The RCUs are then rescaled between 0 and 1 for each memory bank type *(input, hidden, output)*, to maintain the ratio assumption of values between 0 and 1. They are then used to calculate the memory composition for the next window. The network is trained for a given number of epochs, after which the learnt weights and biases are used in the testing phase.

6.4 Results & Analysis

In this section, the SL-MRN is applied to the four of the time-series tasks *(Business cycle prediction, Oil price prediction, M3 Sales prediction and Covid-19 forecasting)* presented in Chapter 3 and their performance is compared to the standard MRN[1]. The following measures are used to evaluate the performance for each task: Matthew Correlation Coefficient (MCC) *(for Business cycle prediction)*, Root-Mean Squared Error (RMSE) *(for Oil price and M3 Competition prediction)* and Mean Absolute Percentage Error (MAPE) *(for Covid-19 forecasting)*. The results are presented along with the percentage Improvement over the standard MRN (IoMRN). The IoMRN is calculated as follows:

$$IoMRN = \frac{\text{SL-MRN score - MRN score}}{\text{MRN score}} \tag{6.9}$$

Similar to the standard MRN and the PA-MRN, the memory order for the memory bank types of the SL-MRN models is maintained at 4. Thus, as explained in Section 3.1.1.2, a total of 63 models is required for training with any given set of hyper-parameters to identify the best model.

The SL-MRN models build on and extend the MRN through additional units, more specifically, the way in which information is processed does not change *(unlike with the PA-MRN)*. Therefore, the hyper-parameters *(except memory bank configuration)* associated with the best standard MRN models[2] are utilised for the SL-MRN models. *(Note: this reduces the hyper-parameter search space drastically as the only tunable hyper-parameters for experimentation with the SL-MRN models are the memory banks)*. Employing additional units *(RCUs)* can lead to over-fitting, thus, the SL-MRN models are trained with 20 units as with the standard MRN and then with a lower number of units *(5 for the M3 Sales prediction & 15 for the remaining task)* to understand whether the models overfit and the impact on performance. In addition, the SL-MRN models are compared to the PA-MRN models, to understand which of these extensions offered better improvements for the tasks.

[1] The MRN as presented in Chapter 3

[2] see Table 4.2, Table 4.7 and Appendix B

6.4.1 Business cycle prediction

The SL-MRN extensions are applied to the NBER turning points prediction task and the results for the best models are presented in Table 6.1.

Table 6.1: Best MCC score for the NBER turning points prediction task

Model	Units	Dataset		
		Growth	COD	Growth & COD
MRN	20	0.787	0.79	**0.799**
SL-MRN 1	15	0.78	0.79	0.787
	20	0.775	**0.792**	**0.799**
SL-MRN 2	15	0.63	0.649	0.616
	20	0.629	0.701	0.573
SL-MRN 3	15	0.663	0.641	0.561
	20	0.639	0.627	0.618

As seen from the table, SL-MRN 1 performs significantly better than the other SL-MRN models. SL-MRN 2 and SL-MRN 3 obtained similar results for all three datasets, more specifically SL-MRN 3 obtained the highest score and performs better with the first and third dataset, while SL-MRN 2 performs better with the second dataset. SL-MRN 1 and the standard MRN attained similar performance on all three datasets. More specifically, the standard MRN outperforms for the first dataset, SL-MRN 1 outperforms for the second dataset and both models obtain the same MCC score for the third dataset. The best SL-MRN model offers marginal improvement over the standard MRN for the NBER turning points prediction task as seen from Table 6.2.

Table 6.2: Improvement over the standard MRN (IoMRN) *(%)*

Dataset	Growth	COD	Growth & COD
MRN	0.787	0.79	0.799
SL-MRN	0.78	0.792	0.799
IoMRN	-0.1%	0.2%	0%

In particular, as seen in Table 6.1, for the Growth dataset, the SL-MRN models performed best with 15 units; for the COD dataset, SL-MRN 3 performed best with 15 units whilst the remaining SL-MRN models performed best with 20

units and for the Growth & COD dataset, SL-MRN 2 performed best with 15 units whilst the remaining SL-MRN models performed best with 20 units. For this problem, there appears little scope for further optimisation as the standard MRN configuration appears to offer an optimum balance between historic and current information.

Table 6.3: Best MCC score for the MRN model extensions

Dataset	Growth	COD	Growth & COD
PA-MRN	0.751	0.757	0.793
SL-MRN	**0.78**	**0.792**	**0.799**

Furthermore, in Table 6.3, the best PA-MRN *(the model extension presented in Chapter 5)* and the overall best SL-MRN models are compared. The SL-MRN models performed better than the PA-MRN models for all datasets. The SL-MRN models provide comparatively similar results to the standard MRN and performed better than the PA-MRN for the NBER turning points prediction task.

6.4.2 Oil price prediction

The SL-MRN models are applied for the oil price prediction task to understand whether self-learning enhances performance and the results for the best models are presented in Table 6.4.

Table 6.4: Best RMSE scores for oil price prediction

Model	Units	Horizon			
		1	3	6	12
MRN	20	0.321	0.693	0.864	0.892
SL-MRN 1	15	0.32	0.689	0.964	**0.835**
	20	**0.318**	0.684	0.981	0.859
SL-MRN 2	15	0.321	0.682	0.901	0.9
	20	0.32	0.686	0.9	0.937
SL-MRN 3	15	0.32	**0.679**	0.935	0.956
	20	0.319	0.686	0.977	0.948

As seen from the Table 6.4, SL-MRN 1 performed best for an horizon of 1 and 12 while SL-MRN 3 performed best for an horizon of 3. The standard MRN

performed better than the SL-MRN models for an horizon of 6, the models appear to have difficulty predicting for this horizon as seen from Figure 6.5.

For an horizon of 1, the SL-MRN models performed best with 20 units; for an horizon of 3, SL-MRN 1 performed best with 20 units and the remaining SL-MRN models with 15 units; for an horizon of 6, SL-MRN 2 performed best with 20 units while the remaining with 15 units and for an horizon of 12, SL-MRN 3 performed with 20 units and the remaining models with 15 units.

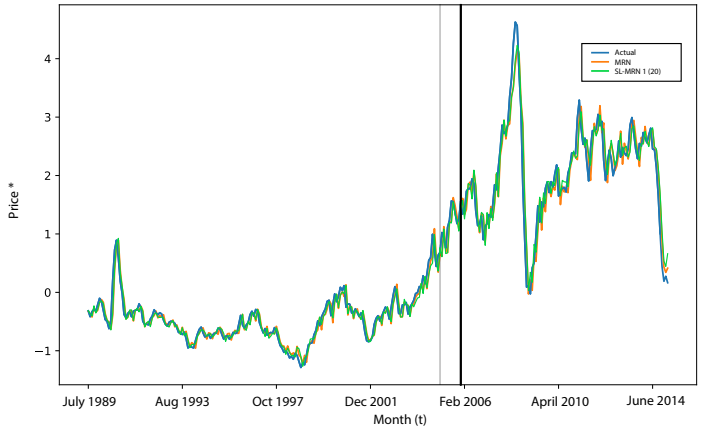


Figure 6.3: Best models for the oil price prediction task *(Horizon 1)*

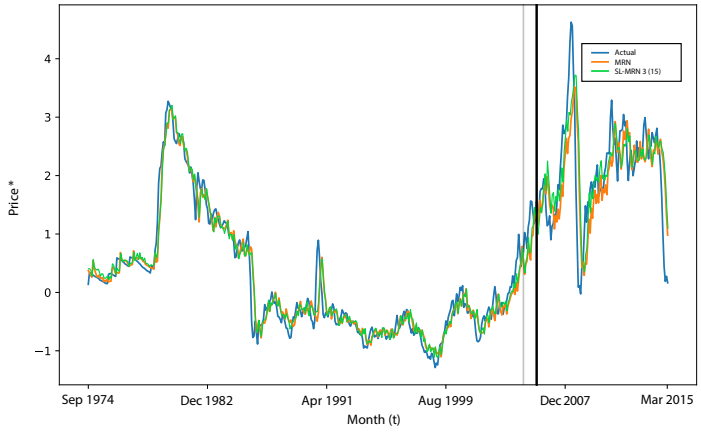


Figure 6.4: Best models for the oil price prediction task *(Horizon 3)*

Figure 6.3 visualizes the overall best models for an horizon of 1. All but one of the best SL-MRN models appear to offer a slight improvement over the standard MRN model. Figure 6.4 visualizes the overall best models for an horizon of 3. The SL-MRN predictions appear to be less erratic than the standard MRN model. As shown from Table 6.5, the SL-MRN models offer an improvement of up to 1% and 2% over the standard MRN models for an horizon of 1 and 3 *respectively*.

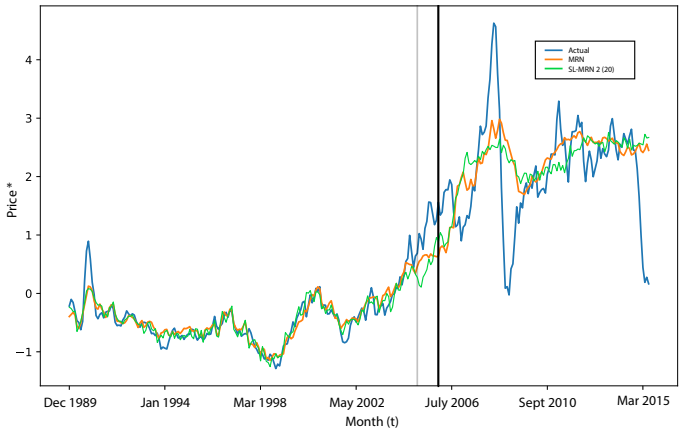


Figure 6.5: Best models for the oil price prediction task *(Horizon 6)*

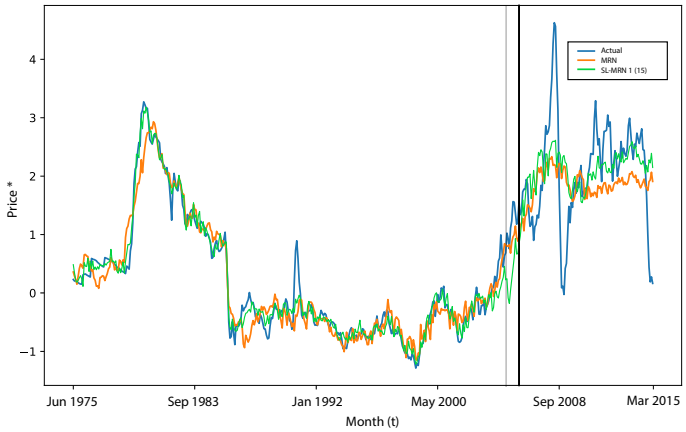


Figure 6.6: Best models for the oil price prediction task *(Horizon 12)*

Figure 6.5 visualizes the overall best models for an horizon of 6. Both the

standard MRN and SL-MRN models appear to predict the peak on a smaller scale and smooth out the succeeding predictions. All variants of the SL-MRN are outperformed by the standard MRN. Figure 6.6 visualizes the overall best models for an horizon of 12. The SL-MRN model appears to offer predictions closer to the observed values than the standard MRN.

Table 6.5: Improvement over the standard MRN (IoMRN) *(%)*

Horizon	1	3	6	12
MRN	0.321	0.693	0.864	0.892
SL-MRN	0.318	0.679	0.9	0.835
IoMRN	1%	2%	-4.17%	6%

As shown from Table 6.5, the SL-MRN model offers an improvement of up to 6% over the standard MRN models, for an horizon of 12. However, for an horizon of 6, there is a decline in performance of up to 4.17%. Interestingly, the overall best SL-MRN model obtained a better performance for 't + 12' than for 't + 6'. Similar to the standard MRN, the SL-MRN model had difficulties mapping an horizon of 6. This mapping is especially difficult for 't + 6', as numerous changes occur in a short period, and it appears the network may not have fully represented all these changes. Whereas for 't + 12', the network has a longer overview, encouraging it to fully represent these perturbations. More specifically, these perturbations affect the learning of not only the parameters but also the ratios in the SL-MRN and thus the memory composition, and as such led to a decline in performance.

Table 6.6: Best RMSE score for the MRN model extensions

Horizon	1	3	6	12
PA-MRN	0.329	**0.674**	**0.883**	0.995
SL-MRN	**0.318**	0.679	0.9	**0.835**

The performance of the best PA-MRN and the overall best SL-MRN models are compared and presented in Table 6.6. The PA-MRN models performed better than the SL-MRN models for two horizons *(3 & 6)* and was outperformed by the SL-MRN models for the remaining two horizons *(1 & 12)*. Overall, the PA-MRN models and the SL-MRN models offer an improvement over the standard MRN

up to 2.74%[1] (as seen from Table 5.11) and up to 6% *respectively*. The SL-MRN models offered an improvement over the standard MRN in most instances and demonstrated better predictive abilities than the PA-MRN for the Oil price prediction task.

6.4.3 M3 competition prediction

In this section, the SL-MRN models are applied to the 10 random series from the M3 data *(presented in Section 3.5.3)* and benchmarked against the standard MRN to assess its performance[2]. The results for the models are presented in Table 6.7.

Table 6.7: Best RMSE scores for M3 Sales prediction

Model *(Units)*		Series				
		N2516	N2521	N1807	N1908	N2012
MRN	(10)	187.1	2017.2	268.3	453.9	517.8
SL-MRN 1	(5)	187.1	2015.7	347.2	649.2	624.9
	(10)	186	2010.2	343	627.9	576.8
SL-MRN 2	(5)	181.9	2008.4	326.1	662	620.8
	(10)	181.2	**2005.8**	319.2	632.6	617.1
SL-MRN 3	(5)	181.9	2012.9	314.6	637.8	559.2
	(10)	**175.7**	2010.5	287.5	627.3	557.9

Model *(Units)*		Series				
		N2159	N2158	N2150	N2144	N1918
MRN	(10)	521.9	651.1	141.6	538	291
SL-MRN 1	(5)	**518.3**	570.9	**138.8**	512.5	**187.4**
	(10)	527.3	570.5	141.6	502.4	192.2
SL-MRN 2	(5)	523.4	578.6	140.8	506.2	187.9
	(10)	533.9	571.5	141.7	**491.6**	191.1
SL-MRN 3	(5)	526.6	**570.3**	138.9	530.8	187.7
	(10)	529.2	578.9	140.1	506.8	190.4

It can be seen that the SL-MRN models outperformed the standard MRN for

[1] wrong correct after viva

[2] The learning rate is lowered to 0.009 to aid learning

most of the series except three series *N1908*, *N1807* and *N2012*. In particular, SL-MRN 1 provided the best performance, outperforming on three of the series, while SL-MRN 2 and SL-MRN 3 outperformed on 2 series each. For 5 of the series *(N1807, N2159, N2158, N2150, N1918)*, the SL-MRN models performed best with 5 units and for the remaining 5 series the models performed best with 10 units.

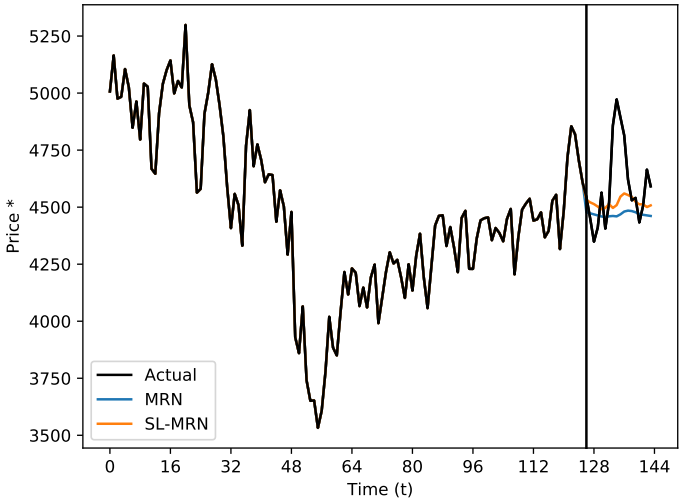


Figure 6.7: Series N1918

Figure 6.7 presents the predictions of the best standard MRN and SL-MRN models for series *N1918*, where the overall best SL-MRN model had the best improvement. It can be seen that the MRN appears to smooth out its predictions as does the SL-MRN, albeit to a lesser extent, offering better predictions than the MRN.

Figure 6.8 presents the predictions of the best standard MRN and SL-MRN models for series *N1908*, where the overall best SL-MRN model had the worst performance. The MRN and SL-MRN models appear to have very similar predictions, closely following the trend in the series. The SL-MRN does appear to have slightly more volatile predictions than the standard MRN, and predicts the peak on a smaller scale. The standard MRN outperformed the SL-MRN model.

In general, there is an improvement between 1% and 12.4% for most of the

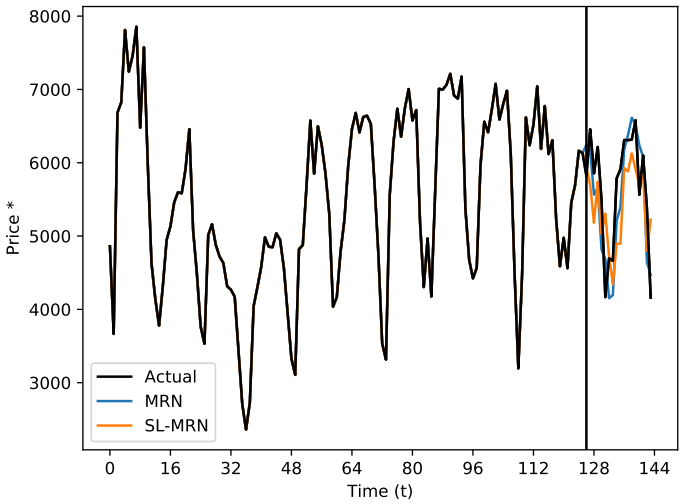


Figure 6.8: Series N1908

Table 6.8: Improvement over the standard MRN (IoMRN) *(%)*

Series	N2516	N2521	N1807	N1908	N2012
MRN	187.1	2017.2	268.3	453.9	517.8
SL-MRN	175.7	2005.8	287.5	627.3	557.9
IoMRN	6.1%	0.57%	-7.16%	-38.2%	-7.7%

Series	N2159	N2158	N2150	N2144	N1918
MRN	521.9	651.1	141.6	538	206.7
SL-MRN	518.3	570.3	138.8	491.6	187.4
IoMRN	0.69%	12.4%	2%	8.62%	9.34%

series except series *N1908*, *N1807* and *N2012* where there is an 38.2%, 7.16% and 7.7% deterioration in performance *respectively* (see Table 6.8). Interestingly, the standard MRN performed worst compared to other models on series *N1908* and *N1807* as seen from Table 4.9. This is indicative of why the SL-MRN did not provide an improvement, as the model on which it is built (the standard MRN), appears to have had significant difficulty mapping the signal.

The performance of the PA-MRN and the overall best SL-MRN for the M3 Sales prediction is compared and presented in Table 6.9, the PA-MRN performs

Table 6.9: Best RMSE score for the MRN model extensions

Series	N2516	N2521	N1807	N1908	N2012
PA-MRN	196.9	2009.6	**256.8**	**622.8**	**533.4**
SL-MRN	**175.7**	**2005.8**	287.5	627.3	557.9

Series	N2159	N2158	N2150	N2144	N1918
PA-MRN	**488.1**	**531.3**	**129.8**	**422.5**	**162.7**
SL-MRN	518.3	570.3	138.8	491.6	187.4

better than the SL-MRN for eight of the ten series. In general, the results presented in this section demonstrates that the SL-MRN model offers no significant improvement over the PA-MRN, but does however possess stronger predictive abilities than the standard MRN for the M3 Sales prediction task.

6.4.4 Covid-19 forecasting

In this section, the SL-MRN model is applied for the Covid-19 forecasting of confirmed and death cases in the United States of America, the results of the best models are presented in Table 6.10.

Table 6.10: Best MAPE score for Covid-19 forecasting

Model	Hidden units	Confirmed cases	Death cases
MRN	20	4.11	0.3
SL-MRN 1	15	4.43	0.33
	20	4.51	0.38
SL-MRN 2	15	**3.46**	0.49
	20	4.02	1.01
SL-MRN 3	15	3.8	15.4
	20	4.16	13.02

For the confirmed cases, SL-MRN 1 obtained higher scores than the standard MRN, thus, underperforming, while SL-MRN 2 obtained lower scores, thus, providing improved performance over the standard MRN. SL-MRN 3 obtained one score that was higher than the standard MRN and one that was lower than the

standard MRN. For the death cases, SL-MRN 1 obtained very similar scores to the standard MRN while SL-MRN 2 obtained higher scores than the standard MRN, thus, underperforming. SL-MRN 3 obtained significantly higher scores, indicating difficulty learning the underlying signal, thus, providing predictions significantly different from the observed values. In particular, SL-MRN 2 was the overall best for the confirmed cases, obtaining the lowest MAPE score and the standard MRN performed best for the death cases with the lowest MAPE score. In addition, in general, all the SL-MRN variants performed best with 15 units for the confirmed and death cases.

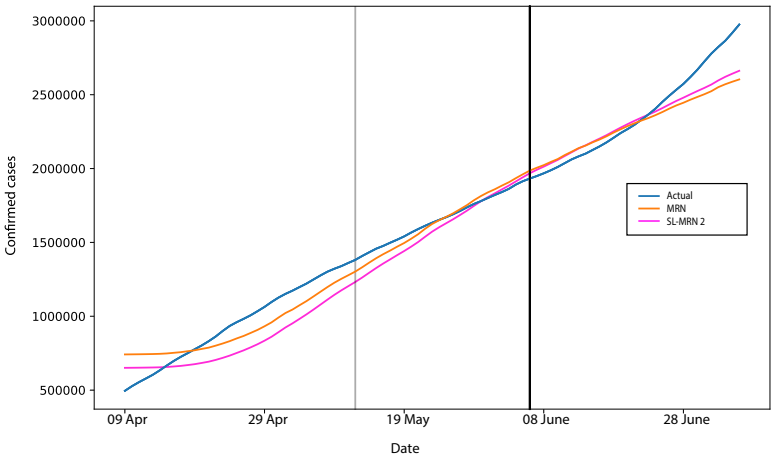


Figure 6.9: Best models for Covid-19 forecasting *(Confirmed cases)*

Figure 6.9 visualizes the overall best SL-MRN and MRN models for the confirmed cases. The SL-MRN appears to provide predictions closer to the observed values than the standard MRN. Figure 6.10 visualizes the overall best models for the death cases, both the standard MRN and SL-MRN appear to obtain very similar predictions which closely match the observed values. As shown in Table 6.11, for the confirmed cases, the SL-MRN offers significant improvement *(up to 16%)* and no improvement for the death cases.

The overall best SL-MRN and the best PA-MRN models are presented in Table 6.12. The PA-MRN is outperformed by the SL-MRN for the confirmed cases and performed better than the SL-MRN for the death cases. The SL-MRN offered an improvement of up to 16% while the PA-MRN offered an improvement of up to 13.33% for Covid-19 forecasting. Overall, the SL-MRN offers some

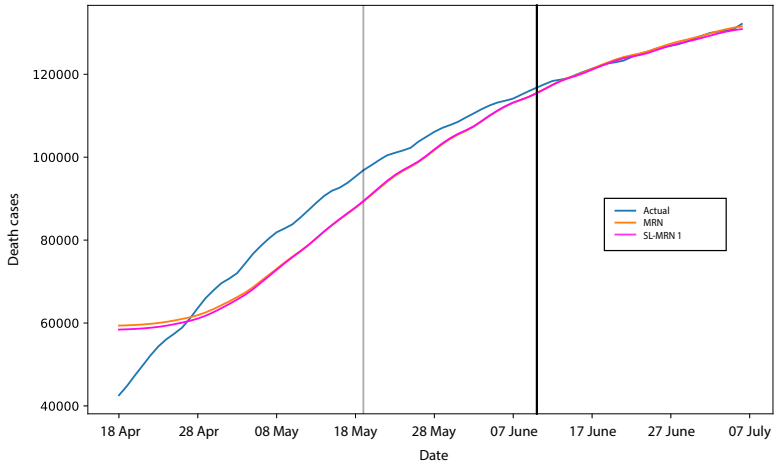


Figure 6.10: Best models for Covid-19 forecasting *(Death cases)*

Table 6.11: Improvement over the standard MRN (IoMRN) *(%)*

Series	Confirmed	Death
MRN	4.11	0.3
SL-MRN	**3.46**	0.33
IoMRN	16%	-10%

Table 6.12: Best RMSE score for the MRN model extensions

Series	Confirmed	Death
PA-MRN	4.01	**0.26**
SL-MRN	**3.46**	0.33

improvement for Covid-19 forecasting.

6.5 Discussion

A discussion of the experiments undertaken in this chapter is given to conclusively assess the SL-MRN and its predictive abilities.

6.5.1 Self-learning attributes to enhance memory quality and reduce user design input

Self-learning properties in ANNs have notably been employed to the feed-forward layers which has proven to be successful and experiments conducted in this chapter build on that. Particularly, given the importance of an appropriate memory mechanism for effective time-series processing, the self-learning concept is extended to the memory layer of the MRN, by incorporating RCUs to learn the appropriate layer-link ratios from the dataset for a given task.

Results presented demonstrate that in general, the MRN extension with self-learning, *the SL-MRN* has the ability to learn specific layer-link ratios for a given task, which enhanced performance, offering an improvement over the standard MRN. To understand the benefits of the SL-MRN, the best standard MRN models are trained with 10 additional hidden units to identify whether the improvements observed with the SL-MRN can be observed by simply increasing the number of hidden units. The standard MRN with additional units[1] is compared to the overall best SL-MRN models for all the tasks presented in Section 6.4, and the results are presented in Table 6.13 and Table 6.14.

As seen from Table 6.13, the SL-MRN performed best in all instances for the NBER turning points prediction task, Oil price prediction task and Covid-19 forecasting. For the M3 Sales prediction task, the SL-MRN performed best for seven of the ten series as seen in Table 6.14.

The results from Table 6.13 and Table 6.14 demonstrate that merely increasing the number of hidden units does not guarantee an improvement in performance, highlighting the need of the SL-MRN models to enhance performance. The SL-MRN models enable the incorporation of input data and memory to inform the ratios, which in turn determine memory bank composition. This appears to en-

[1]Note: the models are trained using the hyper-parameters of the best models. NBER prediction task; learning rate: 0.09, momentum: 0.9999, epochs: 1000 *(see Table 4.2 for the best memory bank combination employed for each dataset along with best window size).* Oil price prediction; learning rate: 0.01, momentum: 0.999, epochs: 1000 *(see Table 4.7 for the best memory bank combination and window size employed for each horizon).* M3 Sales prediction; learning rate: 0.01, momentum: 0.999, epochs: 1000 *(see Appendix B.1 for the best memory bank combination and window size employed for each series). Covid-19 forecasting; learning rate: 0.01, momentum: 0.999, epochs: 500 (see Appendix B.2 for the best memory bank combination and window size employed for each series).*

Table 6.13: MRN models trained with additional hidden units

Dataset	Oil price				NBER			Covid-19	
Model	1	3	6	12	Growth	COD	G & COD	con.	death
SL-MRN	**0.318**	**0.679**	**0.9**	**0.835**	**0.78**	**0.792**	**0.799**	**3.46**	**0.33**
MRN *(30)*	0.319	0.71	1.24	1.13	0.775	0.763	0.78	4.73	4.92

Table 6.14: MRN models trained with additional hidden units

	M3 Series									
Model	2516	2521	1807	1908	2012	2159	2158	2150	2144	1918
SL-MRN	**175.7**	**2005.8**	287.5	627.3	557.9	**518.3**	**570.3**	**138.8**	**491.6**	**187.4**
MRN *(20)*	186.2	2015.6	261.5	470.3	468.9	576.8	619.3	145.7	508.7	205.6

courage better information utilisation and processing, and in general led to an improvement in performance. The results in this chapter conclusively demonstrate that the SL-MRN models are more suitable for time-series processing.

6.5.2 Link ratios

One of the best models for each task presented in Section 6.4 is re-run with the associated hyper-parameters and the learnt ratios are presented.

6.5.2.1 Business cycle prediction

For the Business cycle prediction, SL-MRN 1 performed best with the third dataset which employed the memory bank configuration: $[3, 2, 4]$. The learnt ratios of the memory bank configuration for all the models in the ensemble are presented in Table 6.15.

The standard MRN with the memory bank configuration: $[3, 2, 4]$ employs the following ratios for the input: $[0.33, 0.67, 1]$, hidden: $[0.5, 1]$ and output: $[0.25, 0.5, 0.75, 1]$. The SL-MRN models in the ensemble as seen from Table 6.15 produced different ratios from the standard MRN. The ratio values produced are

Table 6.15: Learnt ratios for the best SL-MRN model for the NBER turning points prediction task

Model Ensemble	Input	Hidden	Output
Model 1	0.712, 0.654, 0.52	0.555, 0.645	0.589, 0.61, 0.651, 0.582
Model 2	0.576, 0.842, 0.548	0.537, 0.677	0.736, 0.335, 0.593, 0.72
Model 3	0.593, 0.612, 0.851	0.693, 0.597	0.641, 0.543, 0.59, 0.191
Model 4	0.664, 0.617, 0.463	0.64, 0.608	0.676, 0.383, 0.466, 0.693

between 0.45 and 0.85, 0.5 and 0.7, 0.1 and 0.75 for the input, hidden and output memory banks *respectively*. The models appear to utilise more stable and flexible memories for the input and hidden memories whilst employing a range of rigid to flexible memories for the output memories.

6.5.2.2 Oil price prediction

For the Oil price prediction task, the SL-MRN models underperformed for 't + 6' compared to the standard MRN. From the SL-MRN models, SL-MRN 2 performed best employing the memory bank configuration: $[3, 0, 4]$. The learnt ratios for the memory bank configuration are presented in Table 6.16.

The standard MRN with the memory bank configuration: $[3, 0, 4]$ employs the following ratios for the input: $[0.33, 0.67, 1]$, hidden: - and output: $[0.25, 0.5, 0.75, 1]$. Three of the four SL-MRN models *(Model 2, 3 & 4)* in the ensemble as seen from Table 6.16 produced learnt ratios between 0.6 and 1, these models appears to utilise more flexible memories. Model 1 produced learnt ratios between 0.4 and

Table 6.16: Learnt ratios for the best SL-MRN model for the Oil price prediction task *(Horizon of 6)*

Model Ensemble	Input	Hidden	Output
Model 1	1, 0.511, 0.43	-	0.683, 0.915, 0.437, 0.886
Model 2	0.626, 0.658, 0.687	-	0.637, 0.648, 0.624, 0.636
Model 3	0.635, 0.765, 0.751	-	0.761, 0.701, 0.714, 1
Model 4	0.772, 0.835, 0.737	-	0.741, 0.811, 0.728, 0.729

1, the model appears to utilise a combination of rigid to flexible memories.

6.5.2.3 M3 Competition prediction

For the M3 Sales prediction, the best improvement of 12.4% was obtained for series *N2158*. SL-MRN 3 performed best employing the memory bank configuration: $[0, 0, 2]$. The learnt ratios for the memory bank configuration are presented in Table 6.17.

Table 6.17: Learnt ratios for the best SL-MRN model for the M3 Sales prediction task

Model Ensemble	Input	Hidden	Output
Model 1	-	-	1, 0
Model 2	-	-	1, 0
Model 3	-	-	1, 0
Model 4	-	-	0, 1

The standard MRN with the memory bank configuration: $[0, 0, 2]$ employs the following ratios for the input: -, hidden: - and output: $[0.5, 1]$. The SL-MRN models in the ensemble as seen from Table 6.17 appear to utilise very flexible memories for the output memories.

6.5.2.4 Covid-19 forecasting

For Covid-19 forecasting, SL-MRN 2 performed best with a 16% improvement employing the memory bank configuration: $[2, 3, 2]$. The learnt ratios for the memory bank configuration are presented in Table 6.18.

Table 6.18: Learnt ratios for the best SL-MRN model for the Covid-19 forecasting

Model Ensemble	Input	Hidden	Output
Model 1	6.34e-06, 1.68e-06	5.4e-06, 6.3e-06, 2.94e-06	3e-06, 7.44e-06
Model 2	0.326, 0.219	0.217, 0.216, 0.218	0.263, 0.249
Model 3	0.197, 0.223,	0.281, 0.192, 0.208	0.279, 0.202
Model 3	0.229, 0.134	0.141, 0.183, 0.219	0.145, 0.136

The standard MRN with the memory bank configuration: $[2, 3, 2]$ employs the following ratios for the input: $[0.5, 1]$, hidden: $[0.33, 0.67, 1]$ and output: $[0.5, 1]$. The SL-MRN models in the ensemble as seen from Table 6.18 do not learn any ratios for all the memory bank types higher than 0.35 for this task, the model appears to prefer more rigid memories which hold on to historical information longer.

The learnt ratios of the best models are presented in Table 6.15 - Table 6.18, providing insight to memory bank composition preferred by the network *(as informed by the learnt ratios)* for the tasks. In one of the tasks, M3 Sales prediction,

the model produces ratios close to that of the standard MRN. However, for the remaining tasks, the models produce very different ratios compared to that of the standard MRN. Interestingly, in some cases, the models appear to learn similar ratios in a given memory bank type, which begs the question ***Do the models require multiple similar/replicated memories within a given memory bank type? And if they do not, how can the similar/replicated memories be removed from the network?***. This will be investigated in the following chapter.

6.5.3 The MRN extensions

In this chapter, the comparative performance of the overall best SL-MRN *(the MRN extension presented in this chapter)* and the best PA-MRN *(the MRN extension presented in Chapter 5)* is presented in Table 6.19. The SL-MRN models offer an improvement to the standard MRN across all the four different tasks, whilst the PA-MRN offered an improvement to the standard MRN for three of the four tasks.

Table 6.19: Improvement with MRN extensions

Task	NBER turning points	Oil price	M3 Sales	Covid-19
PA-MRN	-0.75%	2.02%	21.5%	13.33
SL-MRN	**0.2%**	**6%**	12.4%	**16%**

In particular, for the NBER turning points prediction, the PA-MRN offered no improvement over the standard MRN for any of the datasets. For the Oil price prediction, the PA-MRN offered an improvement over the standard MRN for one of the four horizons. It also offers an improvement over the standard MRN for most of the series for the M3 Sales prediction and, for both confirmed and death cases, for Covid-19 forecasting. Whereas, the SL-MRN offered an improvement over the standard MRN on one of the datasets and maintained performance for another dataset for the NBER turning points prediction. For the Oil price prediction, the SL-MRN offered an improvement over the standard MRN for three of the four horizons. The SL-MRN also offered an improvement

over the standard MRN for most of the series for the M3 Sales prediction and, for the confirmed cases, for Covid-19 forecasting. The SL-MRN appears to offer more consistent improvements across all four tasks than the PA-MRN, and thus, this research builds further on the SL-MRN extensions.

6.5.4 Search space for memory bank configuration

Similar to the standard MRN and the PA-MRN, the search space for the memory bank configuration is quite large, as a non-exhuastive number of combinations can be explored for a given task. As explained in Section 3.1.1.2, for any given set of hyper-parameters, the total number of models required for training is dependent on the memory order. Increasing the memory order exponentially increases the total number of models to be trained (as shown in Section 3.1.1.2). Thus, the search space for the memory bank configuration was constrained to 4, *(i.e a memory order of 4)*, for the experiments conducted up to and including this chapter. This restriction poses some limitations when employing the MRN variants, as the models can not be thoroughly explored and as such techniques to effectively and efficiently derive appropriate memory bank configurations for the MRN are required.

6.6 Conclusion

In this chapter, the MRN is endowed with self-learning and applied to four time-series forecasting tasks, to assess whether they mitigate a key architectural limitation with the standard MRN. More specifically, the model extension, *the SL-MRN* sought to identify whether the hyper-parameter tuning of ratios can be eliminated, through the self-learning of ratios which inform memory composition.

The SL-MRN, in general, enhanced the performance when compared to the standard MRN as seen from the results presented. In particular, the SL-MRN models offered an improvement over the standard MRN of up to **16%** in 13 of the 19 instances for the experiments undertaken. The experiments indicated that the MRN can be endowed to inform and enhance memory composition *(and thus quality)*, through incorporating RCUs to learn the layer-link ratios and as such

reducing user design input.

However, similar to the standard MRN and the PA-MRN presented in Chapter 5, the memory order for the memory bank types is constrained, due to the large space, to identify the memory bank configuration for a given task. More specifically, given the consistent improvement obtained with the SL-MRN, in the following chapter, a framework (incorporating the SL-MRN) will be proposed. This will seek to address the constraint on the memory order, by proposing a technique to efficiently derive 'good' models, without the need to train numerous memory combinations. The framework will also investigate the use of similar/replicated memory banks.

6.7 Major contributions

This is the first work presenting self-learning within the memory mechanisms of an RNN. The self-learning algorithm specifically builds on the architectural structure of the standard MRN by incorporating RCUs in the network *(either by augmenting the hidden layer or through a new ratio layer)*. The RCUs learn the layer-link ratios which inform the self-link ratios and both are used to copy current information and store historical information, which determine the memory composition. In the standard MRN, the link ratios are empirically established by the user. However, by employing the RCUs within the model, the hyper-parameter tuning process is reduced, due to the automatic learning of the ratios (encouraging the model to learn better/preferred ratios).

Chapter 7

Pruning the MRN

The MRN was assessed and compared to alternative state-of-the-art algorithms in preceding chapters, where it offered superior predictive abilities in many cases. Thereafter, two MRN extensions were presented and applied in Chapter 5 and Chapter 6, which offered improvements over the standard MRN. More specifically, the SL-MRN presented in Chapter 6 offered more consistent improvement over the standard MRN and the PA-MRN *(presented in Chapter 5)*. However, the MRN variants require a large search space to identify good memory bank configurations. As a result, in this work, the search space for the memory bank configuration was constrained and the memory bank order (that is the maximum number of memory banks allowed for each memory bank type) was restricted to 4, to reduce the requirement to training multiple memory combinations. In addition, employing the SL-MRN, which demonstrated the ability to learn the ratio hyper-parameters and inform memory composition, highlighted that for some tasks, similar ratios in a given memory bank type are learnt (by the network). Therefore, in this chapter, a framework is proposed to reduce the search space for the memory bank configuration, thus, eliminating the constraint on the memory order. In addition, the framework will seek to identify whether the model requires multiple replicated or similar memory banks or whether these are redundant and can be removed. The chapter begins with a brief overview of current pruning techniques employed within ANNs, including RNNs, and then the proposed pruning approach is presented.

7.1 Background

As with all RNNs, the size and computational complexity of the MRN is fundamentally based on the problem and how it is represented as a learning problem *(e.g. input and output representations, data quality and size)*. Assuming adequate data veracity and efficient representation of the problem, the complexity of the MRN is particularly dependent on the size of its internal sluggish state-based memory mechanism. Increasing the number of memory banks directly increases the network size *(the number of weights and biases)*, thus, increasing the computational and storage requirements. This increase also leads to an increased number of models required for training, to obtain the best model for a given set of hyper-parameters. For example, a memory order: of 4, 5, 6, 7 or 8 memory banks per memory bank type results in an exhaustive search space of $4^3 = 64$, $5^3 = 125$, $6^3 = 216$, $7^3 = 343$ or $8^3 = 512$ models *respectively*. As a result, subsequently increasing the time required to obtain an 'optimal' model.

Due to the number of models and possible architectural configurations required, and the search space to be explored, the experiments conducted thus far from Chapter 4 - Chapter 6 have been limited to a memory order of four *(4)* for each memory bank type. However, as the data complexity and feature space increases, more combinations of memory banks might be required, to fully capture the signal in the data. Thus, such a restriction on the memory order can limit the use and deployability of the MRN in other domains and for more complex data *(for example, patient data)*. This motivates the following question, *'Can a good model for a given problem be obtained in a more computationally efficient way?'*. To answer this question, this chapter seeks to explore and establish an effective framework to alleviate the difficulties associated with identifying good models.

Supposing the memory order did not have the above restriction, a 'good' number of memory banks required for the model to learn the problem can be identified through an exploratory training process. This is achieved by deliberately starting with an over parameterised model and then pruning based on the similarities of the learnt ratios. This drastically reduces the number of models required for training, as only one model with a 'large' enough search space is trained and then pruned. In addition, with a pruning algorithm, it would be

possible to remove additional parameters where necessary, informed by the ratio similarity. By employing this pruning algorithm, the restriction whereby the memory order is constrained to 4 can be relaxed, and a wider search space can be explored for any task and applications in different domains, thus ensuring the model is generalizable.

In Chapter 5 and 6, two extensions were proposed to mitigate key limitations inherent in the standard MRN, offering varying degrees of improvement to the standard MRN. *The SL-MRN* presented in Chapter 6 generally demonstrated superiority over the standard MRN and the extension presented in Chapter 5. In particular, *the SL-MRN* notably showed that there are more 'optimal' ratios that can be learnt by the model to inform the memory composition than user specified ratios, which typically led to an enhanced performance. However, and importantly, as with the standard MRN, both extensions suffer from the need to train multiple models, to explore the search space and as such limiting their use as the memory order increases.

Therefore, in this chapter, a framework which combines *the SL-MRN* and a pruning technique will be investigated and explored, to identify whether it mitigates the computational limitations *(in terms of search space)*, associated with model training and utilisation, such that an effective and generalizable model is obtained efficiently.

7.2 Pruning in Artificial Neural Systems

Pruning in artificial neural networks dates back to the late 1980s, initially inspired by synaptic pruning in biological systems [21]. The idea is to reduce the size of the network by removing unnecessary parameters, thus reducing the computational cost and increasing the scope for deployment. Barth *et al.* [140] carried out experiments to understand the impact of removing weights *(akin to synapses in biological neurons)* on the structure and behaviour of ANNs. She states "networks that are constructed through overabundance and then pruning are much more robust and efficient than networks that are constructed through other means" [172].

One of the earliest works on pruning in neural networks is by Yann Lecun *et*

al. [114] with magnitude based pruning. They presented a technique using second order derivatives, *Optimal Brain Damage (OBD)*, to remove unimportant weights (those whose deletion caused the least increase in the error). Hassibi and Stork [80] in 1992, developed the *Optimal Brain Surgeon (OBS)*, which computes the Hessian matrix *(relaxing the diagonality assumption in OBD)* to remove weights. They claimed the OBS performed better than the OBD and found that the test error increased when pruning with OBD and then retraining [80].

Augasta and Kathirvalavakuma [7] provided a survey of pruning algorithms proposed by researchers for feedforward neural networks, which they grouped into the following: Penalty term methods, Cross validation methods, Magnitude based methods, Mutual Information based methods, Evolutionary pruning methods, Sensitivity Analysis based methods and Significance based pruning methods. A number of these methods have been successfully deployed but not without limitations, the most notable issue being low computational efficacy *(for example, disproportionate increase in computation time)* [7]. Other limitations reported *inter-alia* included: i) the removal of 'small' yet important weights, which prevents saturation with the magnitude based methods of pruning, ii) the requirement for user specification for threshold and tuning parameters and iii) the inability of sensitivity analysis-based methods to detect all redundancy, as mutual independence assumed between inputs and hidden nodes. Augasta and Kathirvalavakuma [7] interestingly pointed out that real-world applications have a preference for simpler and more efficient methods. Thus, pointing to and warranting the need for simple yet computationally effective pruning techniques, to deal with high dimensionality and enhance usability.

There has been much work done exploring pruning techniques with feedforward networks (simple and deep), however, this is not the case for RNNs. Techniques proposed for feedforward networks have not successfully transferred to RNNs. This is particularly due to the impact of removing a memory-based neuron or weight causing multiple undesirable side effects, such as, feature dimension mismatch and as such invaliding the recurrent units [218]. Therefore, more care is needed for the pruning of RNNs.

Wen *et al.* [218] proposed a structured sparsity learning method for RNNs, and pruning occurred by penalising weight matrices through L0 regularization.

Results demonstrated that there was nearly 20× practical speed up achieved whilst maintaining performance. Others have implemented pruning techniques with RNNs such as; i) magnitude based algorithm: assessing the relevance of recurrent state neurons based on the magnitude of the incoming weights [66]; using an unsupervised noise-driven anti-Hebbian pruning rule [134], ii) structured pruning algorithm: removing whole columns of weight matrix [214], and iii) sensitivity based algorithm: measuring weight importance using *a posterior* probability [197]; using Jacobian Spectral Evaluation [234].

7.2.1 Knowledge gap

Most of the work on pruning has been largely focused on pruning feedforward networks, particularly deep feedforward networks, and there has been much success reported for these networks. However, there has been limited work in this area for RNNs, particularly, due to the increased model complexity and interconnectivity. Thus, identifying suitable and computationally effective approaches is not a trivial task. More specifically, most of the work for pruning RNNs, has been carried out for LSTMs [25, 78, 131, 199, 240] and GRUs [138, 235], such that other models have been largely neglected and this is arguably due to their network size. As a result, there is very limited to no work on pruning for comparatively simpler and smaller networks, such as SRNs, Jordan networks, ESNs or Hopfield networks. It should however be noted that although these networks are comparatively smaller, networks such as ESNs and MRNs can rapidly grow in size, due to their memory mechanism [208]. Furthermore, as with LSTMs and GRU-based models, the unnecessary connections in these networks can lead to over-fitting and limit their performance [208].

7.3 Methodology

In this section, a simple pruning technique based on the ratio similarity is introduced. The MRN is trained with a 'large' enough number of memory banks, to learn the problem and then pruned using a novel *one-shot* pruning algorithm. The main determinant of the network size for the MRN variants is the memory

bank mechanism. The flexibility of the MRN variants to employ varying combinations of memory banks, implies the network is able to grow as needed. More specifically, the layer- and self-link ratios for each memory bank type determine the composition of memory banks (i.e. how information (historical and new) is sorted and retained). In Chapter 6, the SL-MRN demonstrated that these ratios can be learnt during training, to determine the memory composition. Therefore, the pruning algorithm is applied to the SL-MRN models, and will be based on the similarity of these learnt ratios.

7.3.1 Pruning in the MRN based on ratio similarity

It is assumed the network is supplied with a specified number of memory banks for each memory type, which informs the search space. The question that arises is, if a particular memory bank type has more memory banks than required, *can the network encourage the additional (redundant) ratios of that memory bank type to converge to values close to each other, so they can be pruned?* For the purpose of pruning in this chapter, redundancy is defined as the similarity between ratios in a given memory bank type *(input, hidden, output)*. Figure 7.1 presents a model before and after pruning. Initially, the model has a memory bank configuration of [4, 4, 4], and it is trained and after pruning the resultant memory configuration is *[2, 2, 3]*.

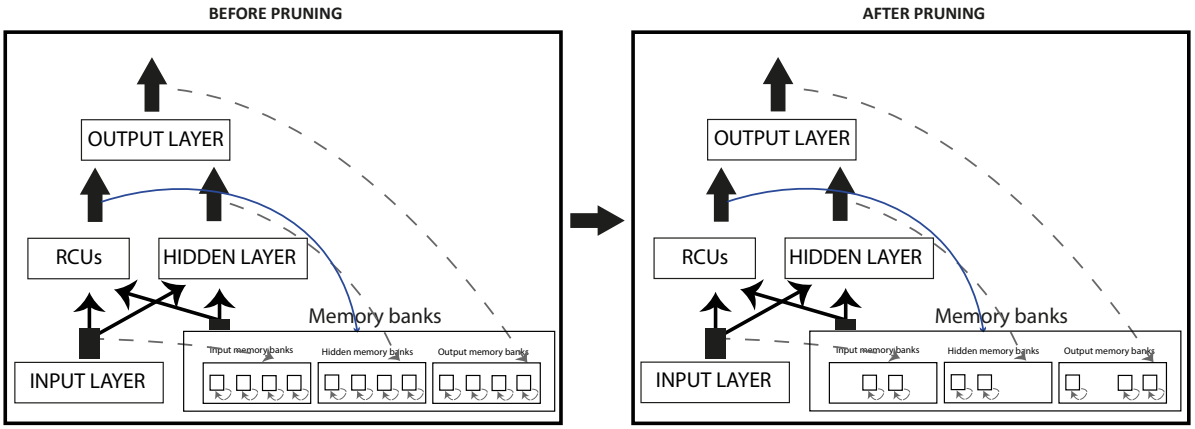


Figure 7.1: The SL-MRN model before and after pruning

The pruning approach is as follows. SL-MRN models are trained as described in Section 6.3. Afterwards, the pruning phase begins *(see Algorithm 2)*. For each memory bank type, a Similarity Index (SI) for the learnt layer-link ratios in a given memory bank type is calculated as follows:

$$SI_j = \frac{\sum_{i=1}^{n} e^{-(|r_j - r_i|)}}{n-1}, \text{ for i = 1, ..., n}, j \neq i \tag{7.1}$$

where n is the total number of layer-link ratios in that memory bank type.

The SI for a given ratio indicates how similar that ratio is to all the other ratios in the same memory bank type. The redundant ratios *(if any, along with their associated self-link ratios, memory banks and weights)* are pruned, and are excluded for information processing during testing. The SI for the remaining ratios is recalculated, and the pruning step is repeated if similar ratios exist. This process is repeated until the ratios in a given memory bank type are distinct enough for a given task. After the pruning phase, a new set of ratios *(and associated memory banks)* are used for the testing phase.

In theory, this pruning technique can enable the network to prune all the ratios based on similarity and thus, a reasonable stopping criterion is required. A key strength of the MRN variants is their ability to employ different types and combinations of memory. In particular, results from Chapter 4 - Chapter 6 demonstrated that different tasks *(even within the same domain)* may require different memory banks *(flexible, rigid)*. To determine whether the ratios should be pruned, the SI is compared to some probability threshold. The SI produces values in [0.35, 1], where 0.35 indicates very distinct and 1 indicates very similar. The mid-point of the SI values is roughly 0.65, thus, probabilty thresholds are chosen between 0.65 and 1, as the pruning algorithm is more concerned with similar ratios rather than distinct ones (which fall betewen 0 and 0.65) and thus, preventing 'over-pruning'. It should be noted that when the network prunes for a threshold probability, a new set of pruned ratios is obtained. To identify the best set of pruned ratios produced, each set of ratios obtained is used in the testing phase and the model with the best performance is selected.

Algorithm 2 Pruning Algorithm

```
1: procedure PRUNING(ratios R⃗, threshold T⃗)
2:     for <each probability threshold, t> do
3:         for <each memory bank type> do
4:             <calculate the similarity index, SI>
5:             while SI > threshold do
6:                 <remove ratios above t (and associated parameters)>
7:                 <recalculate the SI of remaining ratios>
8:             end while
9:         end for
10:    end for
11:    return ratios                                   ▷ pruned ratios
12: end procedure
```

It should be noted that although SL-MRN 1 presented in Chapter 6 obtained the best results in some instances, it is excluded from experimentation in this chapter. This is because supplying the network with a sufficiently large number of memory banks for SL-MRN 1, will drastically increase the number of units in the hidden layer, as the model augments its hidden layer with RCUs to represent each ratio. For example, employing a memory bank configuration of [4, 4, 4] or [8, 8, 8] leads to an increase of 12 or 24 units in the hidden layer. In particular, for the pruning technique proposed, the ratios (and their associated parameters) are pruned not the RCUs (which are augmented in the hidden layer). As a result, the final resulting model for SL-MRN 1 may be significantly larger and as such mitigates the possible benefits. The remaining two models, (SL-MRN 2 & SL-MRN 3), are employed as they have separate ratio layers and these units are not fed to the memory banks thus their structure is kept to a minimum.

7.4 Results & Analysis

The SL-MRN models with the proposed one-shot pruning algorithm are applied and evaluated for the following four time-series tasks: Business cycle prediction, Oil price prediction, M3 Sales prediction and Covid-19 forecasting. The models

are trained with 4 memory banks for each memory bank type*([4, 4, 4])* and then pruned where possible. For each task, the resultant models are presented, assessed, compared to the models without pruning and then evaluated.

7.4.1 Business cycle prediction

In this section, SL-MRN 2 and SL-MRN 3 with one-shot pruning are applied for the NBER turning points prediction task[1] and the results are presented in Table 7.1. SL-MRN 2 with pruning performs best with 20 units for two of the three series, while SL-MRN 3 performs best with 15 units for two of the three series. SL-MRN 3 with pruning performed the best overall, for all the series *(highlighted in blue)*.

Table 7.1: MCC Score for best SL-MRN models with pruning for NBER prediction task

		Dataset		
Model	Units	Growth	COD	Growth & COD
SL-MRN 2 with pruning	15	**0.78**	0.735	0.76
	20	0.77	**0.739**	**0.77**
SL-MRN 3 with pruning	15	**0.79**	**0.773**	0.75
	20	0.75	0.772	**0.772**

The best SL-MRN models with pruning are then compared to the best SL-MRN models without pruning and the results are presented in Table 7.2, as shown the SL-MRN models with pruning performed better than the SL-models without pruning *(highlighted in blue)*.

To understand the impact of the pruning algorithm, the MCC scores before and after pruning are compared and presented in Table 7.3. As seen from the table, employing the pruning technique with the models led to an improvement in performance for all but one instance, where the performance was maintained.

As earlier stated, all the SL-MRN models are trained with 4 memory banks for each type *([4, 4, 4)* and then pruned. The memory configuration after pruning

[1]The hyper-parameters, learning rate, momentum (lowered for better stability), initialisation values, hidden units associated with the SL-MRN models presented in Chapter 6 are employed.

Table 7.2: MCC Score for best SL-MRN models (with & without pruning) for NBER prediction task

		Dataset		
Model	Pruning	Growth	COD	Growth & COD
SL-MRN 2	without	0.63	0.701	0.616
	with	**0.78**	**0.739**	**0.77**
SL-MRN 3	without	0.663	0.641	0.618
	with	**0.79**	**0.773**	**0.772**

Table 7.3: MCC Score for best SL-MRN models with pruning for NBER prediction task

		Dataset		
Model	Pruning	Growth	COD	Growth & COD
SL-MRN 2 with pruning	before	0.734	0.713	0.732
	after	0.78 ↑	0.739 ↑	0.768 ↑
SL-MRN 3 with pruning	before	0.79	0.742	0.754
	after	0.79 -	0.773 ↑	0.772 ↑

the models in the ensemble for the best SL-MRN model with pruning is presented in Table 7.4. *It should be noted that each model results in different combinations dependent on the learnt ratios for an individual model.* For the growth series, the models in the ensemble employed all the memory banks provided and no pruning occurred. For the COD series and the Growth & COD series, one memory bank was pruned from most memory bank types.

Table 7.4: Memory bank configuration for best SL-MRN models after pruning for NBER prediction task

Growth	COD	Growth & COD
[4, 4, 4], [4, 4, 4], [4, 4, 4], [4, 4, 4]	[3, 3, 3], [4, 3, 3], [4, 3, 3], [4, 4, 4]	[3, 3, 3], [3, 4, 3], [3, 4, 4], [3, 3, 4]

Employing the pruning algorithm reduced the number of models required for training, as only one model with a maximum number of memory banks for each memory bank type is trained, and then pruned. This is a key advantage offered as the SL-MRN, without pruning, is run with multiple memory bank combinations

dependent on the memory order. More specifically, as stated for a memory order of 4, 63 memory combinations are required for training to obtain the best model. The SL-MRN models employed with the pruning algorithm not only reduced the number of models required for training but also enhanced performance of the SL-MRN models for the NBER turning points prediction task.

7.4.2 Oil price prediction

The SL-MRN models with the pruning technique are applied for the Oil price prediction at four different horizons *(1, 3, 6, 12)*[1]. The models are supplied with 4 memory banks for each type *[4, 4, 4]* and pruned, and the results are presented in Table 7.5. SL-MRN 3 with pruning performed best for an horizon of 12 months and for the remaining horizons, SL-MRN 2 with pruning performed best.

Table 7.5: RMSE Score for the best SL-MRN models with pruning for Oil price prediction

Model	Units	Horizon of 1	Horizon of 3	Horizon of 6	Horizon of 12
SL-MRN 2 with pruning	15	0.345	**0.697**	1.57	**1.26**
	20	**0.334**	0.699	**1.04**	1.27
SL-MRN 3 with pruning	15	**0.336**	**0.717**	**1.39**	1.37
	20	0.4	0.729	1.42	**1.07**

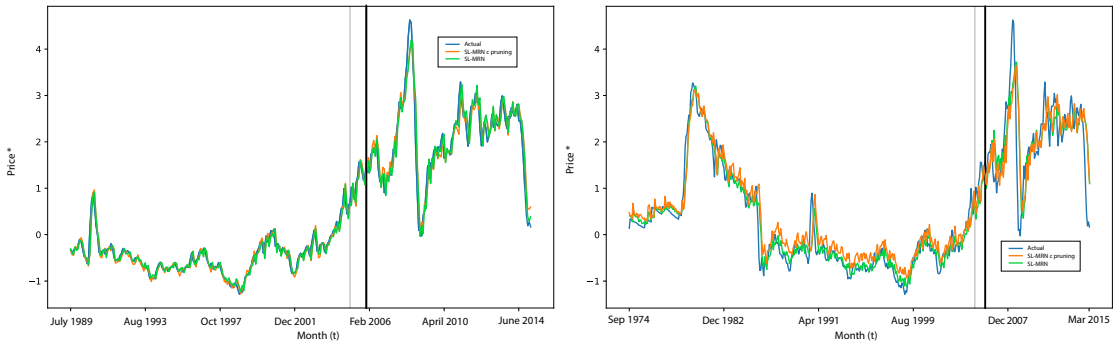
Table 7.6: RMSE Score for the best SL-MRN models (with & without pruning) for Oil price prediction

Model	pruning	Horizon of 1	Horizon of 3	Horizon of 6	Horizon of 12
SL-MRN 2	without	**0.321**	**0.684**	**0.9**	**0.9**
	with	0.334	0.697	1.04	1.26
SL-MRN 3	without	**0.32**	**0.679**	**0.935**	**0.95**
	with	0.336	0.717	1.39	1.07

The performance of the best SL-MRN models with pruning are then compared to the best SL-MRN models without pruning, and presented in Table 7.6. The

[1]The models are configured with the hyper-parameters associated with the SL-MRN models presented in Chapter 6 *(learning rate, momentum, initialisation values, hidden units)*.

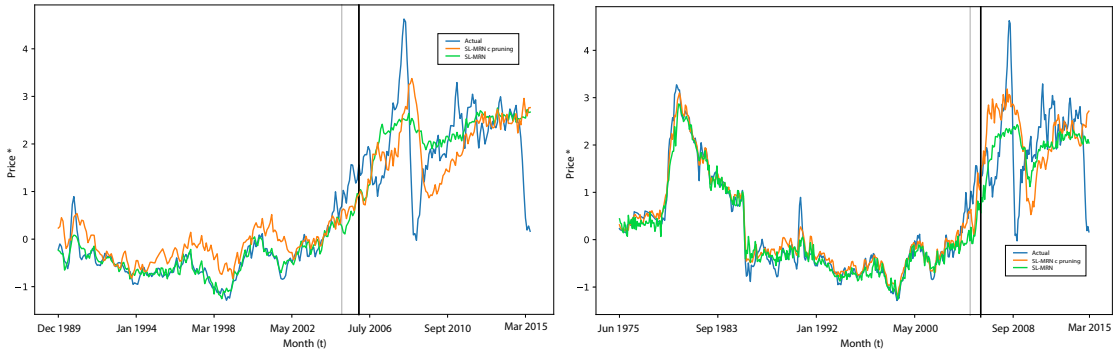
SL-MRN models without pruning performed better than those with pruning. The overall best SL-MRN models (with & without pruning) are visualised in Figure 7.2 for an horizon of 1 and 3 *respectively*. As shown from the figure, for the horizon of 1 and 3, the SL-MRN models with pruning and the SL-MRN without pruning obtain very similar predictions.



(a) Horizon of 1 (b) Horizon of 3

Figure 7.2: Overall best SL-MRN models for Oil price prediction

Figure 7.3 presents the overall best SL-models (with & without pruning) for an horizon of 6 and 12. For an horizon of 6 and 12, the models appears to map the signal albeit on a smaller scale following the trend. The SL-MRN without pruning appears to provide better predictions than the SL-MRN with pruning.



(a) Horizon of 1 (b) Horizon of 3

Figure 7.3: Overall best SL-MRN models for Oil price prediction

To understand the benefits of employing the pruning technique, the RMSE Score for the overall best SL-MRN models before and after pruning is presented

in Table 7.7, and as shown pruning the models enhanced the performance for all horizons. The memory bank configuration of the models in the ensemble after pruning is presented in Table 7.8.

Table 7.7: RMSE Score before and after pruning for the overall best SL-MRN models with pruning for Oil price prediction

pruning	Horizon of 1	Horizon of 3	Horizon of 6	Horizon of 12
before	0.35	0.77	1.34	1.08
after	0.33 ↓	0.697 ↓	1.04 ↓	1.07 ↓

For an horizon of 1, 3 and 6 months, in general two memory banks are pruned and for an horizon of 12 months, one or no memory bank is pruned. The SL-MRN models without pruning employed the memory bank combinations, for an horizon of 1: [0, 4, 4], an horizon of 3: [3, 3, 2], an horizon of 6: [3, 0, 4] and an horizon of 12: [0, 2, 2]. For an horizon of 3, the SL-MRN with pruning utilised a smaller memory configuration than the SL-MRN without pruning. For the remaining horizon, the SL-MRN without pruning has one memory bank type, where no memory banks are employed. However, the pruning technique prunes based on ratio similarity, and as such a minimum of 1 memory bank in each memory bank type is employed.

Table 7.8: Memory bank configuration for best SL-MRN models after pruning for Oil price prediction

Horizon of 1	Horizon of 3	Horizon of 6	Horizon of 12
[2, 2, 4], [2, 4, 2], [2, 2, 4], [2, 2, 2]	[2, 2, 2], [2, 2, 2], [2, 2, 2], [2, 2, 2]	[2, 3, 2], [2, 2, 2], [2, 2, 2], [2, 2, 4]	[4, 3, 4], [4, 3, 4], [4, 4, 4], [3, 4, 4]

The experiments in this section highlighted the limitations of the pruning technique, such that at least one memory bank is employed. For the Oil price prediction, there was a slight drop in performance as shown in Table 7.9. Nonetheless, employing the pruning algorithm with the SL-MRN models provided an efficient means to obtain a 'good' set of ratios, particularly, as only one model (with 4 memory banks for each memory bank type) was trained and pruned. The SL-MRN with pruning maintained performance with a maximum of 0.19% drop in

Table 7.9: Overall best SL-MRN models (for each series) for Oil price prediction

Series	SL-MRN with pruning	SL-MRN	% difference
Horizon of 1	0.334	0.32	-0.04
Horizon of 3	0.684	0.679	-0.01
Horizon of 6	295.4	287.5	-0.15
Horizon of 12	623.5	627.3	-0.19

the score, for the trainig of one 'large enough' pruned model rather than training 63 models (as with the SL-MRN models without pruning).

7.4.3 M3 competition prediction

The SL-MRN models with the pruning technique are applied to the 10 random series *(presented in Chapter 3)* of the M3 data for sales prediction[1]. The SL-MRN models are endowed with 4 memory banks for each memory bank type, then pruned, and the results are presented in Table 7.10, where SL-MRN 3 model with pruning performed best for six series and SL-MRN 2 with pruning performed best for the remaining four series. Note: the best model for each technique is highlighted in black and the overall best models in blue.

The performance of the SL-MRN models without pruning and the SL-MRN with pruning are then compared and presented in Table 7.11. The SL-MRN models without pruning performed best for six of the series, while the SL-MRN models with pruning performed best for the remaining four series. Note: the best model for each technique is highlighted in black and the overall best models in blue. Table 7.12 presents the scores of these best models before and after pruning and the resultant memory banks.

As seen from the table, no ratios were pruned for two series, *N2150* and *N1918*, which indicates that the model learnt ratios that were distinct. Interestingly, for series *N2521*, pruning the network led to a slight decrease in performance. For the remaining series, the models pruned the ratios and memory banks, indicating

[1]The models employed the hyper-parameters associated with the standard SL-MRN models *(learning rate, momentum, initialisation values, hidden units)*. The models are run with window sizes of 10 and 40, similar to the other MRN variants.

Table 7.10: Best RMSE scores of the SL-MRN models with pruning for M3 sales prediction

Model	Units	Series				
		N2516	N2521	N1807	N1908	N2012
SL-MRN 2 with pruning	5	**189.1**	2016.1	**429.4**	733.5	**538.4**
	10	198.9	**2014.5**	464.2	**693.4**	547.8
SL-MRN 3 with pruning	5	185.4	2016.6	**295.4**	**623.5**	**557.6**
	10	**177.9**	**2014.7**	347.5	643.5	567.2

Model	Units	Series				
		N2159	N2158	N2150	N2144	N1918
SL-MRN 2 with pruning	5	573.9	**543.9**	161.2	554.4	**208.4**
	10	**520.9**	555	**160.8**	**521.2**	212.5
SL-MRN 3 with pruning	5	**558.3**	**620.3**	**143.88**	**492.3**	**188.6**
	10	582.4	677.6	143.9	511.8	207.7

Table 7.11: Best RMSE scores for the best SL-MRN models (with & without pruning) for M3 sales prediction

		Series				
Model	pruning	N2516	N2521	N1807	N1908	N2012
SL-MRN 2	without	**181.2**	**2005.8**	**319.2**	**632.6**	617.1
	with	189.1	2014.5	429.4	693.4	**538.4**
SL-MRN 3	without	**175.7**	**2010.5**	**287.5**	627.3	557.9
	with	177.9	2014.7	295.4	**623.5**	**557.6**

		Series				
Model	pruning	N2159	N2158	N2150	N2144	N1918
SL-MRN 2	without	523.4	571.5	**140.8**	**491.6**	**187.9**
	with	**520.9**	**543.9**	160.8	521.2	208.4
SL-MRN 3	without	**526.6**	**570.3**	**138.9**	506.8	**187.7**
	with	558.3	620.3	143.88	**492.3**	188.6

some ratios were similar and removing them appears to enhance performance.

The overall best SL-MRN models with pruning for two series, *1807* and *1908*,

Table 7.12: RMSE scores before and after pruning for the M3 sales prediction

Series	before pruning	after pruning	Memory bank after pruning
2516	182.8	177.9 ↓	[3, 4, 4],[4, 4, 4], [4, 4, 4], [4, 4, 4]
2521	2013.3	2014.5 ↑	[1, 1, 1],[1, 1, 1], [1, 1, 1], [1, 1, 1]
1807	310.6	295.4 ↓	[3, 3, 4], [3, 4, 4], [4, 3, 4], [4, 3, 3]
1908	642.7	623.5 ↓	[4, 3, 3],[4, 3, 4], [4, 3, 4], [3, 4, 4]
2012	655.5	538.4 ↓	[1, 2, 4],[2, 1, 2], [1, 1, 1], [4, 2, 2]
2159	708.5	520.9 ↓	[1, 2, 2],[2, 2, 2], [2, 2, 1], [1, 1, 2]
2158	691.6	543.9 ↓	[1, 1, 1],[1, 1, 1],[1, 1, 1],[1, 1, 2]
2150	143.88	143.88 -	[4, 4, 4],[4, 4, 4], [4, 4, 4], [4, 4, 4]
2144	526.8	492.3 ↓	[3, 4, 4],[4, 3, 4], [4, 4, 4], [3, 4, 3]
1918	188.6	188.6 -	[4, 4, 4],[4, 4, 4], [4, 4, 4], [4, 4, 4]

where the SL-MRN models without pruning had difficulty predicting *(see Section 6.4.3)* are visualised in Figure 7.4 and Figure 7.5. Figure 7.4 presents series *1807*, both models appear to provide similar predictions, following the trend and achieving similar performance. Series *1908* is presented in Figure 7.5, where the SL-MRN models with pruning outperformed the SL-MRN models without pruning. Both models have similar predictions, however, the SL-MRN with pruning appears to provide predictions on a slightly higher scale than the SL-MRN models without pruning.

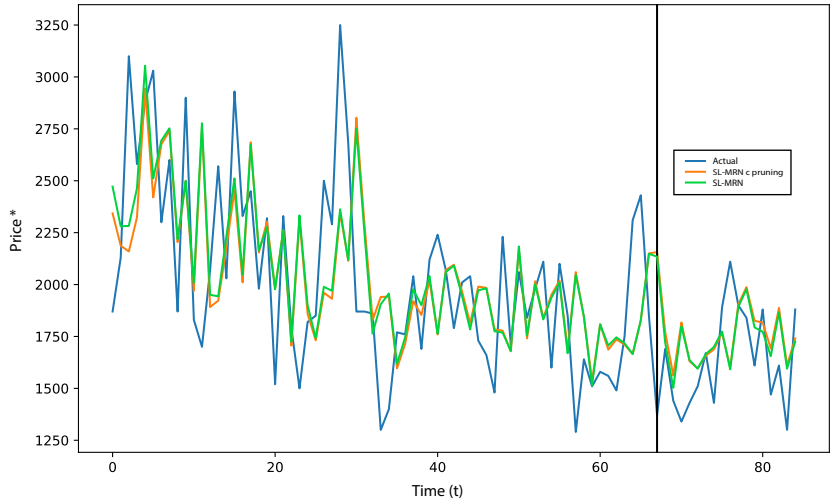


Figure 7.4: Overall best SL-MRN models for M3 Sales prediction *(N1807)*

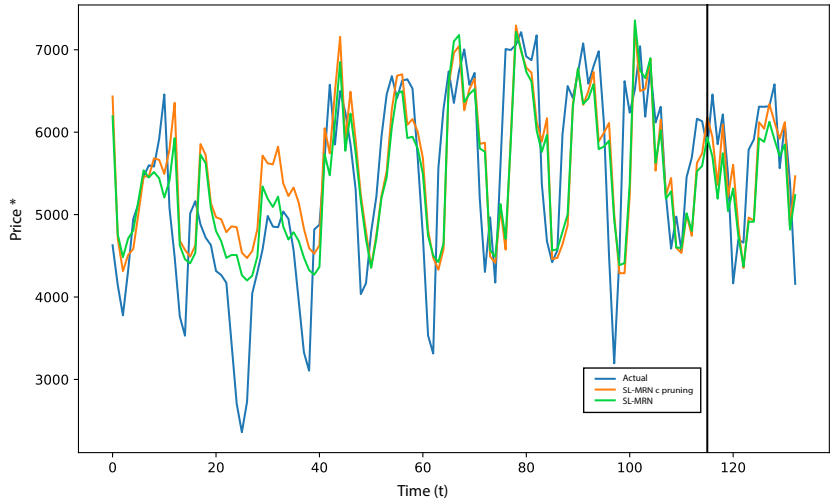


Figure 7.5: Overall best SL-MRN models for M3 Sales prediction *(N1908)*

The overall best models for the SL-MRN models are presented in Table 7.13. The SL-MRN model without pruning performed better than the SL-MRN model with pruning for six of the ten series. More importantly, employing the pruning technique enabled the training of only one model rather than 63 models (as with the SL-MRN). In addition, employing the pruning technique ensures performance is maintained with a maximum of 0.04% drop in the score and in some cases performance is improved up to 0.05%, whilst removing the redundant parameters. For the M3 Sales prediction task, the pruning technique enables the effective derivation of 'good' models.

7.4.4 Covid-19 forecasting

In this section, the SL-MRN models with one-shot pruning are applied for Covid-19 forecasting of confirmed and death cases in the United States[1] The SL-MRN models are trained with 4 memory banks for each memory bank type and pruned, and the results are presented in Table 7.14. SL-MRN 3 with pruning performed best for both the confirmed and death cases, while SL-MRN 2 with pruning appears to have difficulty mapping the signal and therefore obtains significantly

[1]The hyper-parameters *(learning rate, momentum, initialisation values, hidden units)* associated with the SL-MRN models are employed and the models are run with window size of 25 and 35 similar to the standard MRN and SL-MRN.

Table 7.13: Overall best SL-MRN models (for each series) for M3 sales prediction

Series	SL-MRN with pruning	SL-MRN	% difference
2516	177.9	175.7	-0.013
2521	2014.5	2005.8	-0.004
1807	295.4	287.5	-0.03
1908	623.5	627.3	0.01
2012	538.4	557.9	0.035
2159	520.9	523.4	0.005
2158	543.9	570.3	0.05
2150	143.88	138.9	-0.04
2144	492.3	491.6	-0.001
1918	188.6	187.7	-0.005

higher MAPE scores. The best score for each model is highlighted.

Table 7.14: MAPE Score for the best SL-MRN models with pruning for Covid-19 forecasting

Model	Units	Confirmed	Death
SL-MRN 2 with pruning	15	54.2	5.68
	20	**28.2**	**5.59**
SL-MRN 3 with pruning	15	**17.93**	43.5
	20	17.08	**8.18**

Table 7.15: MAPE Score for the best SL-MRN models (with & without pruning) for Covid-19 forecasting

Model	pruning	Confirmed	Death
SL-MRN 2	without	**3.46**	**0.49**
	with	28.2	5.59
SL-MRN 3	without	**3.8**	13
	with	17.08	**8.18**

As seen from Table 6.10, using SL-MRN 3 without pruning, the best score for the confirmed cases is *3.8*, which was obtained with a memory bank configuration of $[0, 3, 0]$ and for the death cases is *13*, which was obtained with a memory bank configuration of $[4, 4, 0]$. Employing the pruning algorithm for Covid-19

forecasting highlights its limitations; the pruning algorithm prunes based on ratio similarity, such that SL-MRN models will always have a non-zero minimum number of memory bank for each memory bank type *(input, hidden, output)*. The memory order for all the memory bank types is increased by 1, so the memory configuration becomes *[5, 5, 5]* and the models are trained to understand if this encourages better performance, the results are presented in Table 7.16.

Table 7.16: MAPE Score of the best SL-MRN models with pruning employing two memory configurations: [4, 4, 4] and [5, 5, 5]

Memory bank	[4, 4, 4]	[5, 5, 5]
Confirmed	5.59	5.24 ↓
Death	8.18	0.5 ↓

For the confirmed cases, there is a slight improvement in performance and for the death cases, employing an additional memory bank significantly improves the performance from *8.18* to *0.5*. The score before and after pruning, and the resultant memory bank configuration is presented in Table 7.17, and as seen the pruned models obtained better scores. The models in the ensemble prune no more than two memory banks for the different memory bank types.

Table 7.17: Overall best MAPE score for the best SL-MRN models before and after pruning for Covid-19 forecasting

Series	before pruning	after pruning	Memory bank after pruning
Confirmed	5.24	5.24 -	[5, 5, 5], [5, 5, 5], [5, 5, 5], [5, 5, 5]
Death	3.06	0.5 ↓	[3, 5, 4], [4, 5, 5], [5, 3, 5], [4, 3, 5]

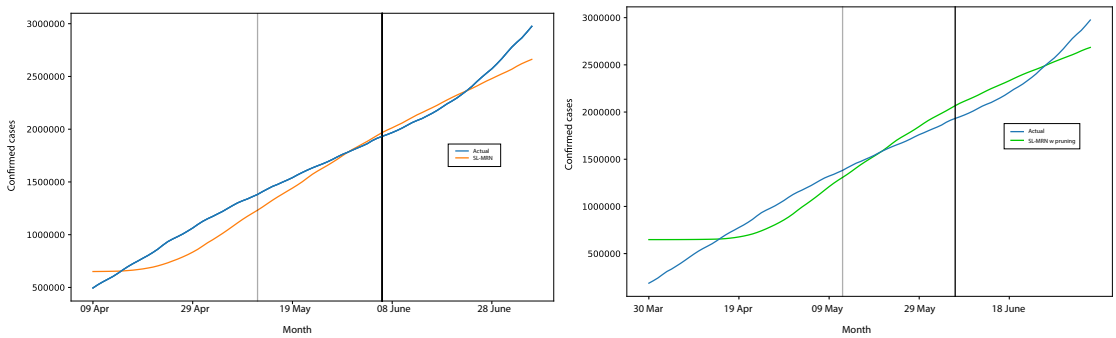
The overall best models for the SL-MRN with pruning and the SL-MRN without pruning are presented in Table 7.18. As seen from the table, the SL-MRN model with pruning obtained similar results to the SL-MRN model without pruning for the death cases however, for the confirmed cases, the performance drops when using the SL-MRN model with pruning.

The overall best SL-MRN models (with & without pruning) are visualised in Figure 7.6 and Figure 7.7. For the confirmed cases, the SL-MRN model without pruning appear to make predictions closer to the observed than the SL-MRN with pruning. *(Note: the SL-MRN model without pruning performed best with*

Table 7.18: Overall best MAPE score of the SL-MRN models for Covid-19 forecasting

Series	SL-MRN w pruning	SL-MRN	% difference
Confirmed	5.24	3.46	-0.5
Death	0.5	0.49	-0.02

a window size of 35 while SL-MRN models with pruning performed best with a window size of 25). For the death cases, both models provide similar predictions which are closely mapped to the observed values.



(a) SL-MRN without pruning (b) SL-MRN with pruning

Figure 7.6: Overall best SL-MRN models for Covid-19 forecasting *(Confirmed cases)*

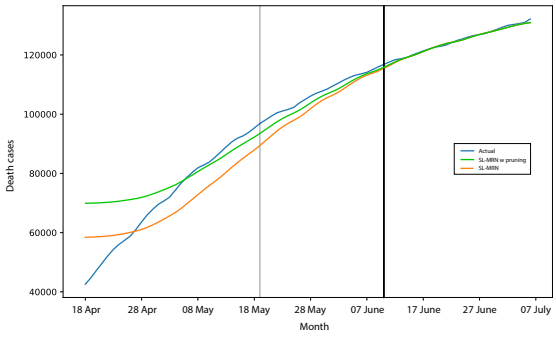


Figure 7.7: Overall best SL-MRN models for Covid-19 forecasting *(Death cases)*

The results in this section highlighted the need for the pruning algorithm to have an option to employ no memory banks if this is what is required. Nonethe-

less, the pruning technique employed provided an effective means to obtain a 'good' model whilst maintaining performance with a maximum of 0.5% drop in the score.

7.5 Discussion

In this section, the pruning technique employed is discussed further, and its performance is conclusively assessed in line with the limitation of the MRN variants.

7.5.1 One-shot pruning

The pruning technique presented is a one-shot pruning algorithm after training based on the ratio similarity. Ratios in a given memory bank type are pruned based on how similar they are to other ratios in the same memory bank type. Experiments demonstrated that the pruning technique is effective and offers two key computational benefits: firstly, *reduced number of models required for training*, as only one *'large'* model is required for training rather than multiple memory bank combinations, and secondly *compactness*, such that the number of ratios *and associated parameters (self-link ratios, memory banks & weights)* are reduced where possible.

However, experiments highlighted a limitation of the pruning algorithm, that is, the inability to completely eliminate a memory bank type, when it is not required. This is particularly due to the pruning condition, which is based on similarity. More specifically, where only one ratio for a memory bank type is left, the ratio can not be pruned as there are no ratios left for comparison. Experiments however demonstrated in such cases where a memory bank type is not required for the task, the self-learning attributes enable the network to still learn other ratios to satisfactorily model and predict for the task.

7.5.2 Ratios

To better understand the effectiveness of the pruning technique, the ratios for some models from the four tasks presented in Section 7.4 are discussed.

7.5.2.1 Business cycle prediction

SL-MRN 3 performed best for the NBER prediction task and the learnt pruned ratios are presented in Table 7.19. The models in the ensemble had the following memory bank configuration: Model 1 - $[4, 4, 4]$, Model 2 - $[4, 4, 4]$, Model 3 - $[4, 4, 4]$ and Model 4 - $[4, 4, 4]$.

Table 7.19: Learnt ratios for the overall best model for the NBER prediction task *(Growth dataset)*

Model Ensemble	Input	Hidden	Output
Model 1	0, 0.032 0.73, 1	0, 0.055, 0.74, 1	0, 0.087, 0.59, 1
Model 2	0, 0.26, 0.59, 1	0, 0.07, 0.95, 1	0, 0.056, 0.14, 1
Model 3	0, 0.026, 0.89, 1	0, 0.56, 0.65, 1	0, 0.2, 0.26, 1
Model 4	0, 0.047, 0.42, 1	0, 0.66, 0.67, 1	0, 0.41, 0.43, 1

All the models in the ensemble employed a memory configuration of $[4, 4, 4]$, that is, none of the ratios in all the memory bank types were pruned, and the model obtained the best performance employing all the memory banks.

7.5.2.2 Oil price prediction

SL-MRN 2 with pruning performed best for an horizon of 6 and SL-MRN 3 with pruning for an horizon of 12 for the Oil price prediction task. The best models in the ensemble for an horizon of 6 had the following memory bank configuration: Model 1 - $[2, 3, 2]$, Model 2 - $[2, 2, 2]$, Model 3 - $[2, 2, 2]$ and Model 4 - $[2, 2, 4]$ and for an horizon of 12, the best models had: Model 1 - $[4, 3, 4]$, Model 2 - $[4, 3, 4]$,

Model 3 - $[4, 4, 4]$ and Model 4 - $[3, 4, 4]$. The ratios before pruning for an horizon of 6 is presented in Table 7.20 and after pruning in Table 7.21.

Table 7.20: Ratios before pruning for the best SL-MRN model with pruning for the NBER turning points prediction task *(Horizon of 6)*

Model Ensemble	Input	Hidden	Output
Model 1	0.61, 0.62, 0.93, 0.97	$2x10^{-6}$, 0.62, 0.69, 0.94	0.61, 0.77, 0.86, 0.97
Model 2	0.16, 0.42, 0.61, 0.75	0.14, 0.62, 0.7, 0.98	0.61, 0.62, 0.9, 0.92
Model 3	0.49, 0.64, 0.69, 0.97	0.45, 0.59, 0.6, 0.8	0.48, 0.64, 0.65, 0.86
Model 4	0.57, 0.66, 0.7, 1	0.52, 0.58, 0.79, 0.81	0.013, 0.082, 0.72, 0.9

As shown from the table, the models performed best pruning the similar learnt ratios. The best model produced pruned ratios that had an SI greater than 0.65.

The ratios after pruning for an horizon of 12 are presented in Table 7.22. The model retained most of the ratios, and it appears the model does not find many ratios that are similar that need to be pruned. The best model pruned ratios with a SI above 0.85. More specifically, Model 1, Model 2 and Model 4 pruned one ratio (0.89, 0.84, 0.68) from the hidden, hidden and input memory bank *respectively*.

7.5.2.3 M3 Competition prediciton

SL-MRN 2 with pruning performed best with series N2158. The ratios before pruning are presented in Table 7.23 and after pruning in Table 7.24. The best models in the ensemble for series *N2158* had the following memory bank config-

Table 7.21: Ratios after pruning for the best SL-MRN model with pruning for the NBER turning points prediction task *(Horizon of 6)*

Model Ensemble	Input	Hidden	Output
Model 1	0.61, 0.97	$2x10^{-6}$, 0.62, 0.94	0.61, 0.97
Model 2	0.16, 0.75	0.14, 0.98	0.61, 0.92
Model 3	0.49, 0.97	0.45, 0.8	0.48, 0.86
Model 4	0.57, 1	0.52, 0.81	0.013, 0.082, 0.72, 0.9

Table 7.22: Ratios after pruning for the best SL-MRN model with pruning for the Oil price prediction task *(Horizon of 12)*

Model Ensemble	Input	Hidden	Output
Model 1	0, 0.27, 0.75, 1	0, 0.8, 1	0, 0.018, 0.78, 1
Model 2	0, 0.58, 0.95, 1	0, 0.71, 1	0, 0.11, 0.71, 1
Model 3	0, 0.23, 0.65, 1	0, 0.053, 0.38, 1	0, 0.46, 0.93, 1
Model 4	0, 0.68, 1	0, 0.28, 0.63, 1	0, 0.37, 0.73, 1

uration: Model 1 - $[1, 1, 1]$, Model 2 - $[1, 1, 1]$, Model 3 - $[1, 1, 1]$ and Model 4 - $[1, 1, 2]$.

As seen from the table, some models learnt very similar ratios within the

Table 7.23: Ratios before pruning for the best SL-MRN model with pruning for the M3 Sales prediction task *(Series N2158)*

Model Ensemble	Input	Hidden	Output
Model 1	0.48, 0.53, 0.56, 0.61	0.36, 0.43, 0.47, 0.6	0.58, 0.6, 0.6, 0.69
Model 2	0.56, 0.63, 0.65, 0.65	0.64, 0.64, 0.58, 0.78	0.44, 0.45, 0.61, 0.63
Model 3	0.47, 0.56, 0.63, 0.68	0.35, 0.49, 0.61, 0.62	0.4, 0.53, 0.57, 0.62
Model 4	0.43, 0.64, 0.65, 0.65	0.61, 0.64, 0.66, 0.77	0.4, 0.56, 0.69, 0.99

Table 7.24: Ratios after pruning for the best SL-MRN model with pruning for the M3 Sales prediction task *(Series N2158)*

Model Ensemble	Input	Hidden	Output
Model 1	0.61	0.36	0.58
Model 2	0.56	0.78	0.63
Model 3	0.68	0.35	0.4
Model 4	0.65	0.61	0.4, 0.99

same memory bank. The model performed best by pruning ratios within the same memory bank type with an SI greater than 0.7.

7.5.2.4 Covid-19 forecasting

SL-MRN 3 with pruning performed best for the death cases. The best models in the ensemble for the death cases had the following memory bank configuration: Model 1 - $[3, 5, 4]$, Model 2 - $[4, 5, 5]$, Model 3 - $[5, 3, 5]$ and Model 4 - $[4, 3, 5]$. The ratios before pruning are presented in Table 7.25 and after pruning in Table

7.26.

Table 7.25: Ratios before pruning for the best SL-MRN model with pruning for Covid-19 forecasting

Model Ensemble	Input	Hidden	Output
Model 1	0, 0.74, 0.86, 0.98, 1	0, 0.1, 0.57, 0.81, 1	0, 0.36, 0.43, 0.6, 1
Model 2	0, 0.27, 0.47, 0.57, 1	0, 0.074, 0.22, 0.87, 1	0, 0.09, 0.32, 0.63, 1
Model 3	0, 0.07, 0.13, 0.72, 1	0, 0.22, 0.24, 0.3, 1	0, 0.12, 0.83, 0.94, 1
Model 4	0, 0.12, 0.35, 0.48, 1	0, 0.74, 0.78, 0.84, 1	0, 0.2, 0.38, 0.89, 1

The models in the ensemble pruned no more than two ratios in any given memory bank type. The model performed best by pruning ratios with an SI above 0.7.

In this section, the pruned ratios produced for the best models are presented. It can be seen that the similarity between the ratios in a given memory bank type varied for the different tasks. In particular, for SL-MRN 3, the pruning technique appears to prune no more than one or two ratios within a memory bank type. This is due to the rescaling of RCUs, which inform ratio values *(as explained in Section 6.3.2)*. It appears rescaling as such encourages the model to produce somewhat distinct ratios, therefore during pruning the model is less likely to prune.

Table 7.26: Ratios after pruning for the best SL-MRN model with pruning for Covid-19 forecasting

Model Ensemble	Input	Hidden	Output
Model 1	0, 0.74, 1	0, 0.1, 0.57, 0.81, 1	0, 0.36, 0.6, 1
Model 2	0, 0.27, 0.57, 1	0, 0.074, 0.22, 0.87, 1	0, 0.09, 0.32, 0.63, 1
Model 3	0, 0.07, 0.13, 0.72, 1	0, 0.3, 1	0, 0.12, 0.83, 0.94, 1
Model 4	0, 0.12, 0.48, 1	0, 0.74, 1	0, 0.2, 0.38, 0.89, 1

7.6 Conclusion

In this chapter, a novel framework combining the SL-MRN (presented in Chapter 6) with a one-shot pruning algorithm (based on the learnt ratio similarity) is proposed. The framework was employed to mitigate the computational complexity associated with training the MRN variants *(a large search space for identify the memory bank configuration)*. Additionally, to identify whether the model requires multiple similar memory banks within a memory bank type. The SL-MRN models with the pruning technique are applied, assessed and evaluated for four time-series forecasting tasks, to understand its impact, and then compared to the SL-MRN models without the pruning technique.

The models are endowed with a large enough number of memory banks to learn the problem and then pruned based on the similarity of ratios in a given memory bank type. Employing the SL-MRN models with the one-shot pruning

technique provided a means to obtain 'good' models. The technique eliminated the training of multiple models in a exhaustive search space and removed redundant ratios (for each memory bank type). The experiments indicated that the pruning algorithm successfully enhanced computational efficacy whilst maintaining performance with a maximum of 0.5% drop in the score. However, a limitation was identified; that is a requirement of a non-zero number of memory banks for each memory bank type, which appeared to limit the robustness of the technique.

7.7 Major Contributions

A review of the literature presented in Chapter 2 suggested that an appropriate memory mechanism in RNNs is essential for effective time-series processing. In particular, results in Chapter 6 showed that the composition *(quality)* of the memory has a direct impact on performance. The work in this chapter, therefore, explored pruning mechanisms to effect better quality memories within the MRN whilst reducing the search space in order to find good memory configurations. A framework which combines the SL-MRN with a pruning technique based on the ratio similarity was presented. The pruning technique provided a means to i) eliminate the constraint on the memory order and reduce the search space for the memory bank combinations, as only one model is required for training rather than multiple models (dependent on the memory order), and ii) remove redundancy in the memory, by identifying similar memory banks in a memory bank type, thus reducing the network size where possible.

Chapter 8

Conclusion: Discussion and Future Work

8.1 Introduction

This chapter revisits the research questions presented in Chapter 1 and considers the extent to which these have been answered by this research. The research contributions of this thesis are summarised and evaluated with respect to their strengths and limitations. Future work and recommendations are subsequently presented to address the limitations highlighted and to further build on the insights provided in this thesis.

8.2 Thesis Summary and Contributions

This thesis presents a set of novel improvements to the MRN class of recurrent neural networks that alleviate some key learning and architectural weaknesses for dealing with time-series problems. The thesis sought to answer the following research questions:

1. *Does the MRN offer a robust alternative to current state-of-the-art models, such that it more effectively models and forecasts time-series data across a range of different problem domains and consistently provides superior performance?*

2. *Can the MRN class of models be extended and endowed to mitigate the inherent learning and architectural limitations and encourage more robust learning of both recent and historical information such that performance is enhanced?*

3. *How can the MRN be adapted to reduce model complexity, particularly, reducing the search space whilst minimising the impact on generalisation ability?*

The sections in this chapter, therefore, summarise the findings of each thesis chapter, with respect to how well the above research questions have been addressed. The research introduced i) periodic attentive units in the MRN to overcome the issue of vanishing gradient descent & catastrophic interferences, ii) self-learning attributes to overcome the issue of hyper-parameter tuning and inform memory composition and iii) a framework with a pruning algorithm to overcome the need for an exhaustive search process during the model fitting phase and identify the optimum memory configuration architecture for a given problem. The extensions are discussed in detail, together with how the results obtained provide new insights into time-series forecasting due to the MRN's optimised memory mechanism (summarized in Section 8.2.1 - 8.2.3).

8.2.1 Adequate computational complexity and appropriate memory mechanism

A critical review of current models and techniques presented in Chapter 2 identified two key criteria for effective time-series processing. The first criteria is *an appropriate memory mechanism, (which is sensitive to both historical and recent events)*, to enable more effective information latching, and enhance learning and generalisation performance. The second criteria is *minimal computational complexity*, such that the model is effective with reasonable training times, usability, transferability and deployment. More specifically, more complex models such as, LSTMs appear to be less adaptive, due to their complex gating mechanism and ESNs are more susceptible to noise leading to increased errors.

Over two decades ago, Ulbricht [208] undertook a study to extend the SRN model class, to include a variant that is endowed with different memory bank combinations, of historical input, hidden and output layer information. Her study demonstrated the importance of empirically establishing the most effective *weighted* input, hidden, output and memory layer feedback configurations to enhance performance. However, Ulbricht did not provide insight into how best to determine the optimum balance between historic and current information within the memory banks. In the model proposed by Ulbricht, the memory composition are determined by the layer- and self- links, which were manually fixed. This thesis therefore built on Ulbricht's work to: i) better identify the optimum memory bank configuration for a given problem; ii) enable the MRN to learn its own layer-link ratios and thus, determine how rigid or flexible its memories should be with minimum user input; and iii) reduce the search space for determining the optimum model complexity during the model fitting and selection phases.

The methodology presented by Ulbricht is coupled with a sliding window approach *(given its simplicity and input processing benefits)* and presented in Chapter 3. The methodology was assessed using a model averaging ensemble approach for any given forecast horizon. In particular, the MRN was applied to tasks and in domains where it has not been previously exploited and compared to state-of-the-art models *(LSTM, SVM, Decision Tree, and Gaussian Naive Bayes)*, in Chapter 4. The results demonstrated that the MRN is a suitable paradigm for effective time-series forecasting, and it offered superiority over current state-of-the-art techniques. In addition, the results highlighted the importance of the MRN's memory flexibility as the best models for the different tasks employed different memory bank combinations.

8.2.2 A model extension employing Periodic Attentive units

The MRN was endowed with periodic attentive units in Chapter 5. More specifically, the periodic units are partitioned such that they receive input stimuli periodically and not at every time-step. These periodic units coupled with the standard (non-periodic) units form the new hidden layer. This extension was developed to 1) prevent gradients from vanishing quickly as new inputs are en-

countered, and ii) minimise catastrophic interference caused by the temporal superposition of information across the hidden layer. By partitioning based on time, the network is encouraged to allocate information occurring at different points in time to different partitions of hidden units. The model extensions were applied and the performance of the extensions were assessed and compared to the standard MRN. The results demonstrated that there were instances where the model extension enhanced performance, indicating that employing periodic units mitigated the vanishing gradient problem and catastrophic interference. In particular, this extension highlighted the importance of partitioning units for different tasks within the MRN and the benefits offered are indicative of the possible benefits of extending this technique to other strong ANN candidate(s).

8.2.3 A novel model extension employing self-learning for the memory mechanism in the MRN

The MRN was endowed by employing RCUs to learn the layer-ratios, thus, reducing hyper-parameter tuning. In particular, employing RCUs to learn the ratios for the memory bank, encourages the network to determine its preferred memory composition rather than predetermined by the user. This simple yet effective technique not only highlighted the inherent strengths of the MRN through its self-learning abilities but also highlighted the importance of the memory composition in a network *(as identified from the critical review)*. In addition, it demonstrated that external and internal input stimuli can be used to learn the ratios which inform and store the different combinations of memories to be employed *(rather than empirically establishing them)*, which in general resulted in an enhanced performance. This simple approach is pivotal as it encourages a paradigm shift for how models are developed, in particular when incorporating memory mechanisms. It demonstrates the autonomy and strong abilities within ANNs to not only learn the underlying patterns but also architectural features *(for example, the MRN learning what types of memory to employ i.e. flexible, rigid, sluggish, stable and so on)*.

8.2.4 Pruning technique

In Chapter 7, a pruning algorithm was proposed to mitigate a computational limitation associated with the MRN. Due to the flexibility of the MRN's memory mechanism, a large search space must be explored to obtain optimal models for any given set of parameters. More specifically, as the memory order *(that is, the maximum number of memory banks for each memory bank type)* increases, the number of models to train rapidly increases. This is calculated using M^n , where M is the memory order and n is the total number of memory bank types. For example, a model with three memory bank types *(input, hidden and output)*, with a memory order: of 4 or 8 requires roughly $4^3 = 64$, $8^3 = 512$ models *respectively*, to exhaustively explore the search space and obtain the best model. Although, the MRN and its variants have proven to be worthy competitors, the large search space to obtain models may be considered to negate the benefits provided. Accordingly, a framework with a pruning technique was introduced to aid the efficient identification of a 'good' model.

The pruning algorithm is a one-shot algorithm based on similarity and was employed with the self-learning models presented in Chapter 6. Pruning occurs using a similarity index, which indicates how similar a ratio is to the remaining ratios in a given memory bank type. The pruning technique was presented with two of the SL-MRN models and applied, evaluated and compared to the two SL-MRN models without pruning, to assess the benefit of the pruning algorithm. The results demonstrated that employing the pruning technique provided a means to obtain 'good' models efficiently. In particular, the number of models required for training was reduced to one and the network ratios were automatically pruned *(along with associated parameters)* where necessary. Eliminating the training of numerous memory combinations is particularly important as the restrictions on the memory order can be removed, and the model can be exploited for any memory bank configuration needed. This framework highlighted more uses of the memory composition and how similarities within the memory can be used to filter out replication.

8.3 Discussion

Recurrent neural networks are known to possess the abilities to identify and map temporal signals. However, important signal information can disappear rapidly due to problems such as vanishing gradient and catastrophic interference as previously mentioned. In addition, RNN variants, such as the MRN, require additional user input to determine architectural hyper-parameter configurations. This led to the novel interventions with periodic attentive units, self-learning link ratios and one-shot pruning proposed by this research. This section discusses the strength and limitations arising from the experiments performed for each innovation and provides insight into how the issues encountered could be remedied.

The first main contribution, i.e. employing PA units to address the vanishing gradient problem and catastrophic interference, proved beneficial as seen from the experiments conducted in Chapter 5 where in some cases, there was an improvement over the standard MRN. However, for two of the four tasks, the PA-MRN offered little to no improvement to the standard MRN as highlighted in Section 5.4.1. In particular, experiments revealed that a sufficient number of TAUs is required when employing the PA-MRN, which in some cases may require a significant increase in the number of additional units. The PA-MRN models, however, are only trained further with 10 additional units, due to limited computational resources available for experimental activities and likewise the concomitant aim of this research, to identify a simple, small yet effective recurrent neural network-based paradigm for time- series processing.

The SL-MRN, the second extension, was presented in Chapter 6 to mitigate limitations with user imputation to determine architectural hyper-parameters. In general, the model consistently offered superior performance over the standard MRN for all tasks. In particular, it demonstrated the ability to identify suitable memory combinations without the need for user input, as the RCUs are adjusted in response to internal and external input stimuli. These incorporated RCUs, as explained in Chapter 6, determine the ratios, which inform the memory composition*(the model's superiority)*. In particular, the RCU values *(ratios)* are obtained from the net RCUs in the network, which are *(treated like other units and)* passed through a sigmoid activation function.

The sigmoid activation function is frequently used with ANNs *(to obtain unit activations within the hidden and output layers)*, as a result of some of its benefits such as being smooth and differentiable. It does, however, have some inherent limitations such as i) *squashing the unit activations* resulting in very similar RCU unit activations, and thus more homogeneous memory banks, and ii) *saturation of the units*; for large negative or positive values, the gradient for these values is almost zero and as such little learning takes place and the derived *RCU values* are of limited use due to the subsequent lack of variance. The consequence is that i) homogeneous RCUs and thus memory banks limit the observed performance as the superiority of these models is attributed to the different combinations and heterogeneity of memory banks *(sluggish, rigid and stable)* employed and ii) RCUs that are not optimal result in lower 'quality' memory bank composition which impedes learning and performance. Nonetheless, the SL-MRN demonstrated strong abilities for time-series processing and, in addition, offered more consistent improvements than the PA-MRN across the different tasks and domains.

The innovations and experimental results presented in Chapter 5 - Chapter 6 addressed key limitations of the standard MRN as discussed above. However, although the MRN's strength is its flexible memory mechanism, this introduces a key computational limitation, in that the MRN variants require the exploration of a large search space to obtain 'good' models. In order to address this limitation, a framework incorporating a one-shot pruning algorithm with the SL-MRN is introduced in Chapter 7. The pruning algorithm not only reduces the number of models required for training to one but also prunes redundant ratios and associated memory banks.

Results indicated that employing the pruning algorithm with the model, provided a means to efficiently obtain 'suitable' models without the cumbersome training of multiple memory combinations. Although, this technique successfully mitigates the limitations of the MRN variants, the pruning technique is limited. In particular, at least one memory bank is required for every memory bank type even though it may not be required for a given task *(which could lead to overfitting)*; this is due to the nature of the pruning algorithm *(which is based on similarity)*. Note: the whole technique is based on ratio selection, which may not be the only method of pruning effectively.

8.4 Future Work

This section identifies some possible directions for future research work beyond the contributions of this thesis, providing recommendations for improvements based on the discussion of the innovations in Section 8.3.

Firstly, to extend the current work presented in this thesis and to address the limitations discussed in Section 8.3, the following suggestions are presented:

- Extend the Covid-19 forecasting task to include other countries and comparison with other models.
- There appears to be a lack of universally agreed approaches to comparing the quality of forecasts generated by different models. Additional approaches, such as the Giacomini and Rossi test [64] and Diebold-Mariono test [44], should be explored to identify more reliable measures of forecast quality and thus enable researchers to reliably compare forecasts across different models.
- To mitigate the limitations of the sigmoid activation function, other activation functions could be employed *(for example: RELU, which abandons the upper bound on the positive gradient of a sigmoid function to allow more effective learning [137])* or an adaptive activation function can be developed *(as in [112])*. The adaptive activation function could employ a wider upper and lower range limit (unlike 0 to 1 as in sigmoid). By employing, an adaptive activation, net values passed through the function are not squashed and saturation is prevented, due to the wider transformation range. Thus, the network is encouraged to learn different types of ratios, preventing homogeneity of the ratios and providing optimal ratios.
- Extending the pruning technique such that the network not only prunes based on ratio similarity but also on memory bank importance, to identify whether a specific memory bank type is required. In particular, where a specific memory bank type is not required, the allocated memory bank(s) can be pruned. This could be achieved by storing and using the error gradients of the memory banks during training to identify the needed memory

bank types required for a given task and therefore the minimum non-zero requirement on memory bank is eliminated.

Building on the insights obtained from the results, future work could include identifying more appropriate memory bank compositions and model extension.

- Learning the self-link ratios

  The model extension proposed in this thesis for self-learning applies only to the layer-link ratios. More specifically, the MRN architecture assumes a connection between the self-link and layer-link ratios *(see 6.1)* thus imposing and forcing a link between the short-term and long-term memory encoding. Thus, to enhance memory composition, the self-link ratios which determine the proportion of the previous memory *(long-term memory)* retained, can be learnt by allocating RCUs to learn their values *(rather than deriving them from the layer-link ratios)*, thus providing the network more autonomy to determine what/how much information to be retained.

- Deep MRN and extensions to more complex domains

  Given the success with the MRN, a simpler class of recurrent neural networks, future work can focus on developing and implementing deep MRNs which employ multiple hidden layers with memory mechanism *(for one or more of the layers)*. Employing multiple hidden layers that collectively produce a hierarchical feature representation of the original input signal might be useful to map more complex representations [71]. In addition, the MRN can be further explored and extended with additional features *(such as convolutional layers, feature maps)* to extend its applicability to more domains *(for example; generate linguistic responses to changes in an image, describing the movement of an object within a scene or an action being performed by an agent within the image i.e. the network is dealing with link words to images)*. These extensions can also be explored for more complex, volatile and higher dimensional problem such as those in the healthcare domain for example; mortality prediction.

- Knowledge Extraction/Explainable AI

Future work could focus on endowing the MRN with explanatory power to provide better understanding of the model predictions and how the predictions are obtained. This could be achieved by mapping data to a set of binary or symbolic patterns and then for each variable, identifying value ranges for zero, positive small/med/high and negative small/med/high. Thereafter, a suitable knowledge/rule extraction method *(for example: Self-Organising Maps or Adaptive Resonance Theory with 7 cluster nodes [65])* could be investigated to form Finite State Machines or grammar discretizing behaviour of the system. The network could then be trained and optimised to learn the symbolic prediction task. Such techniques could be evaluated to review hidden unit state trajectories in response to training/test sequences, to identify localised clusters and transitions between clusters using either Principal Component Analysis as per [91] or SOM approaches as per [91]. The benefits of this approach would be i) to enhance user confidence and ii) to provide a deeper understanding of the underlying dependencies to better inform decision-making.

Publications

The following publications have been produced from the experiments and results obtained from this research:

Published papers

- O. Orojo, J. Tepper, T. M. McGinnity and M. Mahmud, "A Multi-recurrent Network for Crude Oil Price Prediction," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 2940-2945, doi: 10.1109/SSCI44817.2019.9002841.
- O. Orojo, J. Tepper, T. M. McGinnity and M. Mahmud, "Time sensitivity and self-organisation in Multi-recurrent Neural Networks," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-7, doi: 10.1109/IJCNN48605.2020.9206823.

To Appear in Proceedings

- Oluwatamilore Orojo, Jonathan Tepper, T M McGinnity and Mufti Mahmud. Sluggish state-based neural networks provide state-of-the-art forecasts of COVID-19 cases. Applied Intelligence and Informatics Conference. July 2021.

References

[1] Briggs A. and Sculpher M. An Introduction to Markov Modelling for Economic Evaluation. *PharmacoEconomics*, **13**[4]:397–409, 1998. 20

[2] Ratnadip Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *CoRR*, **abs/1302.6613**, 2013. 9, 10, 14

[3] Hafsa Moontari Ali, M Shamim Kaiser, and Mufti Mahmud. Application of convolutional neural network in segmenting brain regions from mri data. In *Proc. Brain Informatics*, pages 136–146. Springer, 2019. 63

[4] Andres M. Alonso. 4. Autoregressive, MA and ARMA processes 4.1. University Lecture, 2008. 13

[5] Muhammad Ardalani-Farsa and Saeed Zolfaghari. Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks. *Neurocomputing*, **73**[13-15]:2540–2553, aug 2010. 32

[6] Hamid Asgari, XiaoQi Chen, Mirko Morini, Michele Pinelli, Raazesh Sainudiin, Pier Ruggero Spina, and Mauro Venturini. NARX models for simulation of the start-up operation of a single-shaft gas turbine. *Applied Thermal Engineering*, **93**:368–376, January 2016. 28

[7] M.Gethsiyal Augasta and T. Kathirvalavakumar. Pruning algorithms of neural networks — a comparative study. *Central European Journal of Computer Science*, **3**:105–115, 2013. 146

[8] Alev Dilek Aydin and Seyma Caliskan Cavdar. Two Different Points of View through Artificial Intelligence and Vector Autoregressive Models for Ex Post and Ex Ante Forecasting. *Comput Intell Neurosci*, **2015**, 2015. 17

[9] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. In *New Advances in Machine Learning*, 2012. 23

[10] Seyed Mohammad Ayyoubzadeh, Seyed Mehdi Ayyoubzadeh, Hoda Zahedi, Mahnaz Ahmadi, and Sharareh R Niakan Kalhori. Predicting covid-19 incidence through analysis of google trends data in iran: Data mining and deep learning pilot study. *JMIR Public Health Surveill*, **6**[2]:e18828, 2020. 82

[11] Jan Babecký, Tomáš Havránek, Jakub Matějů, Marek Rusnák, Kateřina Šmídková, and Bořek Vašíček. Banking, debt, and currency crises in developed countries: Stylized facts and early warning indicators. *Journal of Financial Stability*, **15**:1–17, December 2014. 17

[12] Arko Barman. Time series analysis and forecasting of COVID-19 cases using LSTM and ARIMA models. *CoRR*, **abs/2006.13852**, 2020. 82

[13] Niaz Bashiri Behmiri and José Ramos Pires Manso. Crude Oil Price Forecasting Techniques: A Comprehensive Review of Literature. *SSRN Electron. J.*, 2013. 59

[14] Sebastián Basterrech. An empirical study of the l2-boost technique with echo state networks. *CoRR*, **abs/1501.00503**, January 2015. 33

[15] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**[2]:157–166, 1994. 36, 37, 38, 39, 90

[16] Yoshua Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3, 1993. 37

[17] El Mostafa Bentour. A ranking of VAR and structural models in forecasting. MPRA Paper 61502, University Library of Munich, Germany, January 2015. 17, 18

[18] J. Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, **13**:281–305, 2012. 113

[19] Filippo Maria Bianchi, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *CoRR*, **abs/1705.04378**, 2017. 34

[20] J.M. Binner, P. Tino, J. Tepper, R. Anderson, B. Jones, and G. Kendall. Does money matter in inflation forecasting? *Physica A*, **389**[21]:4793–4808, November 2010. 2, 38, 41, 54, 56, 109

[21] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? *CoRR*, **abs/2003.03033**, 2020. 145

[22] Mikael Bodén. A guide to recurrent neural networks and backpropagation. In *IN THE DALLAS PROJECT, SICS TECHNICAL REPORT T2002:03, SICS*, 2002. 27

[23] Abdelhamid Bouchachia and Alexander Ortner. Self-learning recursive neural networks for structured data classification. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 808–815, Beijing, China, July 2014. IEEE. 116

[24] John A Bullinaria. Recurrent Neural Networks,. University Lecture, 2015. 27

[25] Shijie Cao, C. Zhang, Zhuliang Yao, W. Xiao, L. Nie, D. Zhan, Yunxin Liu, Ming Wu, and L. Zhang. Efficient and effective sparse lstm on fpga with bank-balanced sparsity. *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019. 147

[26] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters, 2019. 21, 22

[27] Luis Fernando Chaves and Mercedes Pascual. Climate cycles and forecasts of cutaneous leishmaniasis, a nonstationary vector-borne disease. *PLoS Medicine*, **3**[8]:1320–1328, 2006. 13

[28] Luis Fernando Chaves and Mercedes Pascual. Comparing Models for Early Warning Systems of Neglected Tropical Diseases. *PLOS Neglected Tropical Diseases*, **1**[1]:e33, October 2007. 13

[29] Shan-Ben Chen, L. Wu, and Q. Wang. Self-learning fuzzy neural networks for control of uncertain systems with time delays. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, **27 1**:142–8, 1997. 114

[30] Yanhui Chen, Kaijian He, and Geoffrey K.F. Tso. Forecasting Crude Oil Prices: a Deep Learning based Model. *Procedia Comput. Sci.*, **122**:300–307, 2017. 35, 91

[31] Vinay Kumar Reddy Chimmula and L. Zhang. Time series forecasting of covid-19 transmission in canada using lstm networks. *Chaos, Solitons, and Fractals*, **135**:109864 – 109864, 2020. 82

[32] Daniel M. Chin, John F. Geweke, and Preston J. Miller. Predicting turning points. Staff Report 267, Federal Reserve Bank of Minneapolis, 2000. 15

[33] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, **abs/1406.1078**, 2014. 91

[34] Mirza Cilimkovic. Neural Networks and Back Propagation Algorithm. University Lecture. 48

[35] Axel Cleeremans, D. Servan-Schreiber, and James L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, **1**:372–381, 1989. 90

[36] Jacques J. F. Commandeur and S. J. Koopman. *An introduction to state space time series analysis*. Practical econometrics. Oxford University Press, Oxford ; New York, 2007. OCLC: ocn123374304. 16

[37] David A. Cook, Graeme Duke, Graeme K. Hart, David Pilcher, and Daniel Mullany. Review of the application of risk-adjusted charts to analyse mortality outcomes in critical care. *Crit Care Resusc*, **10**[3]:239–251, September 2008. 11

[38] R. Coop. *Mitigation of Catastrophic Interference in Neural Networks and Ensembles using a Fixed Expansion Layer*. PhD thesis, The University of Tennessee, 2013. 91, 96

[39] Tim G. Coulson, Daniel V. Mullany, Christopher M. Reid, Michael Bailey, and David Pilcher. Measuring the quality of perioperative care in cardiac surgery. *European Heart Journal - Quality of Care and Clinical Outcomes*, **3**[1]:11–19, January 2017. 12

[40] N. Cowan. What are the differences between long-term, short-term, and working memory? *Progress in brain research*, **169**:323–38, 2008. 47

[41] Domenico Cucinotta and Maurizio Vanelli. Who declares covid-19 a pandemic. *Acta Bio Medica : Atenei Parmensis*, **91**:157 – 160, 2020. 81

[42] Ivo Danihelka, Greg Wayne, Benigno Una, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. *33rd International Conference on Machine Learning, ICML 2016*, **4**:2929–2938, 2016. 35, 64, 91

[43] Tamal Datta Chaudhuri and Indranil Ghosh. Artificial Neural Network and Time Series Modeling Based Approach to Forecasting the Exchange Rate in a Multivariate Framework. *Journal of Insurance and Financial Management*, **1**:92–123, July 2016. 27

[44] Francis Diebold and Roberto Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, **13**[3]:253–63, 1995. 178

[45] Robert S. DiPietro, Nassir Navab, and Gregory D. Hager. Revisiting NARX recurrent neural networks for long-term dependencies. *CoRR*, **abs/1702.07805**, 2017. 28

[46] Georg Dorffner. Neural networks for time series processing. *Neural Network World*, **6**:447–468, 1996. 9, 10, 28, 29, 30, 32, 43, 64, 69

[47] Ke-Lin Du and M. N. S. Swamy. Recurrent Neural Networks. In *Neural Networks and Statistical Learning*, pages 337–353. Springer London, London, 2014. 27

[48] Grzegorz Dudek. Multilayer perceptron for GEFCom2014 probabilistic electricity price forecasting. *International Journal of Forecasting*, **32**[3]:1057–1060, 2016. 25

[49] Steven N Durlauf and Lawrence E Blume. *Macroeconometrics and Time Series Analysis*. Palgrave Macmillan, London, 2009 edition, 2009. 20

[50] R. Dybowski, P. Weller, R. Chang, and V. Gant. Prediction of outcome in critically ill patients using artificial neural network synthesised by genetic algorithm. *Lancet*, **347**[9009]:1146–1150, April 1996. 25

[51] W E Pofahl, Steven Walczak, E Rhone, and S D Izenberg. Use of an artificial neural network to predict length of stay in acute pancreatitis. *The American surgeon*, **64**:868–72, September 1998. 25

[52] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, **14**[2]:179–211, 1990. 2, 30, 31, 69

[53] Tolga Ergen and S. Kozat. Efficient online learning algorithms based on lstm neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, **29**:3772–3783, 2018. 85

[54] R J Frank, N Davey, and S P Hunt. Time Series Prediction and Neural Networks. University Lecture. 25

[55] Tak-Chung Fu. A review on time series data mining. *Eng. Appl. Artif. Intell.*, **24**:164–181, 2011. 1

[56] L. A. Gabralla, R. Jammazi, and A. Abraham. Oil price prediction using ensemble machine learning. In *Proc. ICCEEE 2013*, pages 674–679, August 2013. 59

[57] John W. Galbraith, Aman Ullah, and Victoria Zinde-Walsh. Estimation of the vector moving average model by vector autoregression. *Econometric Reviews*, **21**[2]:205–219, 2002. 17

[58] C. Gallicchio and A. Micheli. Echo state property of deep reservoir computing networks. *Cognitive Computation*, **9**:337–350, 2017. 33

[59] Wendong Ge, Jin-Won Huh, Yu Rang Park, Jae-Ho Lee, Young-Hak Kim, and Alexander Turchin. An Interpretable ICU Mortality Prediction Model Based on Logistic Regression and Recurrent Neural Networks with LSTM units. *AMIA Annu Symp Proc*, **2018**:460–469, December 2018. 35

[60] G. C. Siji George and B. Sumathi. Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction. *International Journal of Advanced Computer Science and Applications*, **11**, 2020. 113

[61] David Gerbing. Time Series Components. University Lecture, 2016. 9, 10

[62] Carlos Gershenson. Artificial neural networks for beginners. *CoRR*, **cs.NE/0308031**, 2003. 23

[63] Changiz Gholipour, Fakher Rahim, Abolghasem Fakhree, and Behrad Ziapour. Using an Artificial Neural Networks (ANNs) Model for Prediction of Intensive Care Unit (ICU) Outcome and Length of Stay at Hospital in Traumatic Patients. *J Clin Diagn Res*, **9**[4]:OC19–OC23, April 2015. 25

[64] Raffaella Giacomini and Barbara Rossi. Forecast comparisons in unstable environments. *Journal of Applied Econometrics*, **25**[4]:595–620, 2010. 178

[65] C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference. *Machine Learning*, **44**[1]:161–183, July 2001. 180

[66] C. Lee Giles and C. Omlin. Pruning recurrent neural networks for improved generalization performance. *IEEE transactions on neural networks*, **5 5**:848–51, 1994. 38, 147

[67] Andrea Giusto and Jeremy Piger. Nowcasting U.S. Business Cycle Turning Points with Vector Quantization. Working papers, Dalhousie University, Department of Economics, September 2013. 58, 59, 65, 66

[68] Andrea Giusto and Jeremy Piger. Identifying business cycle turning points in real time with vector quantization. *International Journal of Forecasting*, **33**[1]:174–184, 2017. 65

[69] Stephen M. Goldfeld and Richard E. Quandt. A Markov model for switching regressions. *Journal of Econometrics*, **1**[1]:3–15, March 1973. 19

[70] Elsy Gómez-Ramos and y Francisco Venegas-Martínez. A REVIEW OF ARTIFICIAL NEURAL NETWORKS: HOW WELL DO THEY PERFORM IN FORECASTING TIME SERIES? *Journal of Statistical Analysis*, **6**, 2013. 25

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 2, 179

[72] S. Gopalswamy, P. J. Tighe, and P. Rashidi. Deep recurrent neural networks for predicting intraoperative and postoperative outcomes and trends. In *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 361–364, February 2017. 35

[73] Jan Grandell. Time series analysis. University Lecture. 8, 9

[74] Umut Güçlü and Marcel A. J. van Gerven. Modeling the dynamics of human brain activity with recurrent neural networks. *Front. Comput. Neurosci.*, **11**, February 2017. arXiv: 1606.03071. 64

[75] Massimo Guidolin. Vector Autoregressive Moving Average (VARMA) Models 1 Foundations of Multivariate Time Series Analysis 1.1 Weak Stationarity of Multivariate Time Series, 2018. 17

[76] Manel Hamdi and Chaker Aloui. Forecasting crude oil price using artificial neural networks: A literature survey. *Econ. Bull.*, **35**[2]:1339–1359, 2015. 59

[77] James D. Hamilton. A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, **57**[2]:357–384, 1989. 19

[78] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Y. Li, Dongliang Xie, Hong Luo, Song Yao, Y. Wang, H. Yang, and W. Dally. Ese: Efficient speech recognition engine with sparse lstm on fpga. *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017. 147

[79] Allan Hartstein and R. H. Koch. A Self-Learning Neural Network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 769–776. Morgan-Kaufmann, 1989. 114, 115

[80] B. Hassibi and D. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1992. 146

[81] D. Haviv, Alexander Rivkind, and O. Barak. Understanding and controlling memory in recurrent neural networks. In *ICML*, 2019. 27

[82] Suzana E. Hikichi, Eduardo G. Salgado, and Luiz A. Beijo. Forecasting number of ISO 14001 certifications in the Americas using ARIMA models. *Journal of Cleaner Production*, **147**:242–253, mar 2017. 16

[83] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, **6**:107–116, 1998. 32

[84] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, **9**[8]:1735–1780, 1997. 2, 92

[85] H. S. Hota, Richa Handa, and A. Shrivas. Time series data prediction using sliding window based rbf neural network. *International Journal of Computational Intelligence Research*, **13**:1145 – 1156, 2017. 50

[86] Meng Hsuen Hsieh, Meng Ju Hsieh, Chin-Ming Chen, Chia-Chang Hsieh, Chien-Ming Chao, and Chih-Cheng Lai. Comparison of machine learning models for the prediction of mortality of patients with unplanned extubation in intensive care units. *Scientific Reports*, **8**[1]:17116, November 2018. 25

[87] Rob J. Hyndman. Moving Averages. In *International Encyclopedia of Statistical Science*, pages 866–869. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 11, 15

[88] Ronald L. Iman and James M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, **9**[6]:571–595, 1980. 78

[89] Queensland Brain Institute. Where are memories stored in the brain? https://qbi.uq.edu.au/brain-basics/memory/where-are-memories-stored. Accessed: 2021-03-17. 47

[90] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, **33**[4]:917–963, Mar 2019. 25

[91] Henrik Jacobsson, Stefan L. Frank, and Diego Federici. Automated Abstraction of Dynamic Neural Systems for Natural Language Processing. In *2007 International Joint Conference on Neural Networks*, pages 1446–1451, Orlando, FL, USA, August 2007. IEEE. 180

[92] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science, 2001. 32, 33

[93] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., USA, 1st edition, 1999. 27

[94] Doreen Jirak, Stephan Tietz, Hassan Ali, and Stefan Wermter. Echo state networks and long short-term memory for continuous gesture recognition: a comparative study. *Cognitive Computation*, 2020. 33

[95] Michael I. Jordan. Chapter 25 - Serial Order: A Parallel Distributed Processing Approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Advances in Psychology*, **121** of *Neural-Network Models of Cognition*, pages 471–495. North-Holland, January 1997. 28

[96] S. Jordan. Analysis and approximation of a jit production line. *Decision Sciences*, **19**:672–681, 1988. 2

[97] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proc. ICML 2015*, **37**, pages 2342–2350, 2015. 36

[98] M. Kaiser. Time-delay neural networks for control. *IFAC Proceedings Volumes*, **27**[14]:967–972, 1994. 26

[99] Deepak A. Kaji, John R. Zech, Jun S. Kim, Samuel K. Cho, Neha S. Dangayach, Anthony B. Costa, and Eric K. Oermann. An attention based deep learning model of clinical events in the intensive care unit. *PLoS One*, **14**[2], February 2019. 35

[100] C. Kang, C. Yong, X. Feng, X. Chuanzhi, S. Xiaofang, S. Dongkuo, D. Maosheng, Z. Jun, X. Li, S. Jiafeng, and L. Xiaoyu. Ultra-short-term wind power prediction and its application in early-warning system of power systems security and stability. In *2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)*, pages 682–687, July 2011. 15

[101] Z. Kasiran, Z. Ibrahim, and M. S. Mohd Ribuan. Mobile phone customers churn prediction using elman and Jordan Recurrent Neural Net-

work. In *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, pages 673–678, December 2012. 29

[102] Geoff Kenny, Aidan Meyler, and Terry Quinn. Forecasting Irish inflation using ARIMA models. Research Technical Papers 3/RT/98, Central Bank of Ireland, December 1998. 15

[103] Daniel Kent and Fathi M. Salem. Performance of three slim variants of the long short-term memory (LSTM) layer. *CoRR*, **abs/1901.00525**, 2019. 34, 35

[104] Sangyeon Kim and Myungjoo Kang. Financial series prediction using attention LSTM. *CoRR*, **abs/1902.10877**, 2019. 35

[105] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, **abs/1612.00796**, 2016. 91

[106] Nowrouz Kohzadi, Milton S. Boyd, Bahman Kermanshahi, and Iebeling Kaastra. A comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, **10**[2]:169–181, March 1996. 25

[107] V. Konecný, O. Trenz, and E. Svobodová. Classification of companies with the assistance of self-learning neural networks. *Agricultural Economics-zemedelska Ekonomika*, **56**:51–58, 2018. 114, 115

[108] Timo Koskela, Mikko Lehtokangas, Jukka Saarinen, and Kimmo Kaski. Time series prediction with multilayer perceptron, fir and elman neural networks. In *In Proceedings of the World Congress on Neural Networks*, pages 491–496. Press, 1996. 25

[109] David Kriesel. A Brief Introduction to Neural Networks. http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf, 2005. 23

[110] Odysseas Krystalakos, Christoforos Nalmpantis, and D. Vrakas. Sliding window approach for online energy disaggregation using artificial neural networks. *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018. 50, 51

[111] Chung-Ming Kuan. Lecture on the markov switching model. *Institute of Economics Academia Sinica*, 01 2002. 19

[112] Vladimír Kunc and Jiří Kléma. On transformative adaptive activation functions in neural networks for gene expression inference. *PLOS ONE*, **16**:e0243915, 01 2021. 178

[113] Xuan Hien Le, Hung Viet Ho, Giha Lee, and Sungho Jung. Application of Long Short-Term Memory (LSTM) neural network for flood forecasting. *Water (Switzerland)*, **11**[7], 2019. 35, 91

[114] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, **2**, pages 598–605. Morgan-Kaufmann, 1990. 146

[115] Anzy Lee, Zong Geem, and Kyung-Duck Suh. Determination of Optimal Initial Weights of an Artificial Neural Network by Using the Harmony Search Algorithm: Application to Breakwater Armor Stones. *Appl. Sci.*, **6**[6]:164, May 2016. 27

[116] Hsiu Yun Lee and Show Lin Chen. Why use Markov-switching models in exchange rate prediction? *Economic Modelling*, **23**[4]:662–668, 2006. 19

[117] Decai Li, Min Han, Senior Member, and Jun Wang. Chaotic Time Series Prediction Based on a Novel Robust Echo State Network. *Transactions on Neural Networks and Learning Systems*, **23**[5]:787–799, 2012. 33

[118] Y. Li, S. Martinis, and M. Wieland. Urban flood mapping with an active self-learning convolutional neural network based on terrasar-x intensity and interferometric coherence. *Isprs Journal of Photogrammetry and Remote Sensing*, **152**:178–191, 2019. 114, 115

[119] Yong Li, Richard Gault, and T. Martin McGinnity. Probabilistic, recurrent, fuzzy neural network for processing noisy time-series data. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10, 2021. 76, 105

[120] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for NAS. *CoRR*, **abs/1912.06059**, 2019. 113

[121] C.L. Lin, J.F. Wang, C.Y. Chen, C.W. Chen, and C.W. Yen. Improving the generalization performance of rbf neural networks using a linear regression technique. *Expert Systems with Applications*, **36**[10]:12049–12053, 2009. 53

[122] Y. Liu, Y. Fang, and Xue mei Zhu. Modeling of hydraulic turbine systems based on a bayesian-gaussian neural network driven by sliding window data. *Journal of Zhejiang University SCIENCE C*, **11**:56–62, 2009. 50, 51

[123] Hsin-min Lu, Daniel Zeng, and Hsinchun Chen. Bioterrorism Event Detection Based on the Markov Switching Model: A Simulated Anthrax Outbreak Study. *ISI*, pages 76–81, 2008. 19

[124] Hsin-min Lu, Daniel Zeng, and Hsinchun Chen. Prospective Infectious Disease Outbreak Detection Using Markov Switching Models. *Ieee Transactions on Knowledge and Data Engineering*, **22**[4]:565–577, 2010. 20

[125] Gilbert C.S. Lui, W.K. Li, Kenneth M.Y. Leung, Joseph H.W. Lee, and A.W. Jayawardena. Modelling algal blooms using vector autoregressive model with exogenous variables and long memory filter. *Ecological Modelling*, **200**[1-2]:130–138, January 2007. 17

[126] Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE trans. neural netw. learn. syst.*, **29**[6]:2063–2079, 2018. 63

[127] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning forecasting methods: Con-

cerns and ways forward. *PLoS ONE*, **13**[3]:e0194889, March 2018. 21, 22, 25

[128] R. Mantovani, A. L. Rossi, J. Vanschoren, B. Bischl, and A. Carvalho. Effectiveness of random search in svm hyper-parameter tuning. *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015. 113

[129] F. D. Marques, L. de F. Rodrigues de Souza, D. C. Rebolho, A. S. Caporali, and E. M. Belo. Application of Time-Delay Neural and Recurrent Neural Networks for the Identification of a Hingeless Helicopter Blade Flapping and Torsion Motions. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, **XXVII**[2]:97–103, 2005. 26

[130] José Maria P. Menezes and G. Barreto. Long-term time series prediction with the narx network: An empirical evaluation. *Neurocomputing*, **71**:3335–3343, 2008. 28

[131] Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. In *EMNLP*, 2017. 147

[132] Miami University. Lecture 4a : ARMA Model. University Lecture, 2014. 14

[133] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. *Machine learning: algorithms and applications*. CRC Press, Taylor & Francis Group, Boca Raton, 2017. 22

[134] Eli Moore and Rishidev Chaudhuri. Using noise to probe recurrent neural network structure and prune synapses, 2020. 147

[135] K Muralidharan. A note on transformation, standardization and normalization. *Int. J. Oper. Quant. Manage.*, **IX**[1 & 2]:116–122, 2010. 59

[136] Martin J Murphy and Sonja Dieterich. Comparative performance of linear and nonlinear neural networks to predict irregular breathing. *Phys. Med. Biol.*, **51**[22]:5903–5914, November 2006. 25

[137] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 178

[138] Sharan Narang, Gregory F. Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. *CoRR*, **abs/1704.05119**, 2017. 147

[139] Ganapathy S. Natarajan and Aishwarya Ashok. Multivariate forecasting of crude oil spot prices using neural networks. *CoRR*, **abs/1811.08963**, 2018. 27

[140] S. Navlakha, A. Barth, and Z. Bar-Joseph. Decreasing-rate pruning optimizes the construction of efficient and robust distributed networks. *PLoS Computational Biology*, **11**, 2015. 145

[141] D. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, **10**:18–23, 1990. 114

[142] D. Nguyen and B. Widrow. Truck backer-upper: an example of self-learning in neural networks. In *Defense, Security, and Sensing*, 1990. 114, 115

[143] Manan Binth Taj Noor, Nusrat Zerin Zenia, M Shamim Kaiser, Mufti Mahmud, and Shamim Al Mamun. Detecting neurodegenerative disease from mri: A brief review on a deep learning perspective. In *Proc. Brain Informatics*, pages 115–125. Springer, 2019. 63

[144] Thomas C O'Connell. *Using Periodically Attentive Units to Extend the Temporal Capacity of Simple Recurrent Networks*. Master's thesis, University at Albany, State University of New York, New York, NY, USA, 1995. 89, 90, 91, 92, 93, 112

[145] U.S. Bureau of Economic Analysis. All employees, total nonfarm (payems). https://fred.stlouisfed.org/series/PAYEMS. Accessed: 2021-02-04. 58

[146] U.S. Bureau of Economic Analysis. Industrial production: Total index (indpro). https://fred.stlouisfed.org/series/INDPRO. Accessed: 2021-02-04. 58

[147] U.S. Bureau of Economic Analysis. Industrial production: Total index (indpro). https://fred.stlouisfed.org/series/PIECTR. Accessed: 2021-02-04. 58

[148] Michael Opitz, Horst Possegger, and Horst Bischof. Efficient model averaging for deep neural networks. In *Asian Conference on Computer Vision (ACCV)*, 2016. 53

[149] Oluwatamilore Orojo, Jonathan Tepper, T. M. McGinnity, and Mufti Mahmud. A Multi-recurrent Network for Crude Oil Price Prediction. In *Proc. IEEE SSCI*, pages 2953–2958. IEEE, 2019. 2, 38, 63, 109

[150] Oluwatamilore Orojo, Jonathan Tepper, T. M. McGinnity, and Mufti Mahmud. Time sensitivity and self-organisation in multi-recurrent neural networks. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020. 2, 38

[151] Dimitrios V Paliouras. *COMPARING REGIME-SWITCHING MODELS IN TIME SERIES: LOGISTIC MIXTURES: vs. MARKOV SWITCHING*. Master's thesis, University of Maryland, 2007. 18, 19

[152] Ana Pano-Azucena, Esteban Tlelo-Cuautle, Sheldon Tan, Brisbane Ovilla-Martinez, and Luis de la Fraga. FPGA-Based Implementation of a Multilayer Perceptron Suitable for Chaotic Time Series Prediction. *Technologies*, **6**[4]:90, oct 2018. 25

[153] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. On the difficulty of training Recurrent Neural Networks. *CoRR*, **abs/1211.5063**, 2012. 90

[154] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, **2015-Janua**, pages 1–5, 2015. 26

[155] Florian Pelgrin. Lecture 2 : ARMA (p , q) models. University Lecture, 2012. 14

[156] Marcus B. Perry. The Weighted Moving Average Technique. In *Wiley Encyclopedia of Operations Research and Management Science*, page eorms0964. John Wiley & Sons, Inc., Hoboken, NJ, USA, February 2011. 11

[157] Biljana Petrevska. Predicting tourism demand by A.R.I.M.A. models. *Economic Research-Ekonomska Istrazivanja*, **30**[1]:939–950, 2017. 16

[158] Andreea-Cristina PETRICĂ, Stelian STANCU, and Alexandru TINDECHE. Limitation of ARIMA models in financial and monetary economics. *Theoretical and Applied Economics*, **0**[4(609), W]:19–42, Winter 2016. 21

[159] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of Biomedical Informatics*, **69**:218–229, May 2017. 35

[160] David V Pilcher, Toni Hoffman, Chris Thomas, David Ernest, and Graeme K Hart. Risk-adjusted continuous outcome monitoring with an EWMA chart: could it have detected excess mortality among intensive care patients at Bundaberg Base Hospital? *Critical Care and Resuscitation*, **12**[1]:6, 2010. 11, 12

[161] Gollam Rabby, Saiful Azad, Mufti Mahmud, Kamal Z. Zamli, and Mohammed Mostafizur Rahman. Teket: a tree-based unsupervised keyphrase extraction technique. *Cogn. Comput.*, 2020. doi: 10.1007/s12559-019-09706-3, [epub ahead of print]. 63

[162] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of Database Systems*, **532–538**:532–538, 01 2009. 53

[163] Prof. Gesine Reinert. 1 What are Time Series ? University Lecture, 2010. 16

[164] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, **65 6**:386–408, 1958. 24

[165] Stephen M. Ruffing and Ganes H.K. Venayagamoorthy. Short to medium range time series prediction of solar irradiance using an Echo State Network. *2009 15th International Conference on Intelligent System Applications to Power Systems, ISAP '09*, 2009. 33, 34

[166] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. 24

[167] Dede Ruslan, Rusiadi Rusiadi, Ade Novalina, and Annisa Ilmi Faried Lubis. EARLY DETECTION OF THE FINANCIAL CRISIS OF DEVELOPING COUNTRIES. *International Journal of Civil Engineering and Technology*, page 15, 2018. 17

[168] Perry Sadorsky. Modeling and forecasting petroleum futures volatility. *Energy Econ.*, **28**[4]:467–488, July 2006. 69

[169] H. Saeed, Hang Wang, M. Peng, A. Hussain, and Amjad Nawaz. Online fault monitoring based on deep neural network & sliding window technique. *Progress in Nuclear Energy*, **121**:103236, 2020. 50

[170] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, **2015-August**:4580–4584, 2015. 36, 91

[171] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, **abs/1402.1128**, 2014. 35

[172] Jill Sakai. Core concept: How synaptic pruning shapes neural wiring during development and, possibly, in disease. *Proceedings of the National Academy of Sciences*, **117**:16096 – 16099, 2020. 145

[173] Hojjat Salehinejad, Julianne Baarbe, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *CoRR*, **abs/1801.01078**, 2018. 27, 37

[174] B. P. Salmon, J. C. Olivier, W. Kleynhans, K. J. Wessels, F. van den Bergh, and K. C. Steenkamp. The use of a Multilayer Perceptron for detecting new human settlements from a time series of MODIS images. *International Journal of Applied Earth Observation and Geoinformation*, **13**[6]:873–883, 2011. 25

[175] J. Sanders, S. Frank, R. Kudchadker, T. Bruno, and J. Ma. Development and clinical implementation of seednet: A sliding-window convolutional neural network for radioactive seed identification in mri-assisted radiosurgery (mars). *Magnetic Resonance in Medicine*, **81**:3888 – 3900, 2019. 50

[176] Smart A Sarpong. Modeling and Forecasting Maternal Mortality; an Application of ARIMA Models. *International Journal of Applied Science and Technology*, **3**[1]:10, 2013. 14, 15

[177] Hidefumi Sawai. TDNN-LR continuous speech recognition system using adaptive incremental TDNN training. In *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, **1**, pages 53–56, 1991. 26

[178] Murat H. Sazli. A brief review of feed-forward neural networks. *Communications, Faculty Of Science, University of Ankara*, pages 11–17, 2006. 24

[179] Ani Shabri and Ruhaidah Samsudin. Daily Crude Oil Price Forecasting Using Hybridizing Wavelet and Artificial Neural Network Model. *Mathematical Problems in Engineering*, **2014**:1–10, 2014. 27

[180] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: from theory to algorithms*. Cambridge University Press, New York, NY, USA, 2014. 22, 23

[181] Yuehjen E. Shao and Shih Chieh Lin. Using a time delay neural network approach to diagnose the out-of-control signals for a multivariate normal process with variance shifts. *Mathematics*, **7**[10], 2019. 26

[182] Sourabh Shastri, K. Singh, S. Kumar, P. Kour, and V. Mansotra. Time series forecasting of covid-19 using deep learning models: India-usa comparative case study. *Chaos, Solitons, and Fractals*, **140**:110227 – 110227, 2020. 61

[183] Mahmud Saad Shertil. *On the Induction of Temporal Structure by Recurrent Neural Networks*. PhD thesis, Nottingham Trent University, 2014. 2, 38, 56, 109

[184] Hon-Yi Shi, King-Teh Lee, Hao-Hsien Lee, Wen-Hsien Ho, Ding-Ping Sun, Jhi-Joung Wang, and Chong-Chi Chiu. Comparison of Artificial Neural Network and Logistic Regression Models for Predicting In-Hospital Mortality after Primary Liver Cancer Surgery. *PLOS ONE*, **7**[4]:e35781, April 2012. 25

[185] Zhiwei Shi and Min Han. Support vector echo-state machine for chaotic time-series prediction. *IEEE Transactions on Neural Networks*, **18**[2]:359–372, 2007. 33

[186] Robert. H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications with R Examples*, **19**. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, 3rd edition, 1964. 10

[187] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications: with R examples*. Springer texts in statistics. Springer, New York, 3rd ed edition, 2011. 9

[188] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, **529**[7587]:484, 2016. 63

[189] Sameer Singh. Noisy time-series prediction using pattern recognition techniques. *Computational Intelligence*, **16**, 2000. 15

[190] Yogesh Singh, Pradeep Kumar Bhatia, and Omprakash Sangwan. A REVIEW OF STUDIES ON MACHINE LEARNING TECHNIQUES. *International Journal of Computer Science and Security (IJCSS)*, **1**:15, 1999. 23

[191] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward training recurrent neural networks for lifelong learning. *Neural Computation*, **32**[1]:1–35, 2020. 91

[192] Q. Song. Robust Jordan network for nonlinear time series prediction. In *The 2011 International Joint Conference on Neural Networks*, pages 2542–2549, July 2011. 29

[193] Zhe Song, Yu Jiang, and Zijun Zhang. Short-term wind speed forecasting with Markov-switching model. *Applied Energy*, **130**:103–112, 2014. 20

[194] Simon Stevenson. A comparison of the forecasting ability of ARIMA models. *Journal of Property Investment & Finance*, **25**[3]:223–240, May 2007. 15

[195] James H. Stock and Mark W. Watson. Vector autoregressions. *Journal of Economic Perspectives*, **15**[4]:101–115, 2001. 18

[196] Putu Sugiartawan and Sri Hartati. Time series data prediction using elman recurrent neural network on tourist visits in tanah lot tourism object. *International Journal of Engineering and Advanced Technology*, **9**[1]:314–320, 2019. 31

[197] J. Sum, Lai-Wan Chan, Chi sing Leung, and G. Young. Extended kalman filterbased pruning method for recurrent neural networks. *Neural Computation*, **10**:1481–1505, 1998. 147

[198] Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *CoRR*, **abs/1804.11188**, 2018. 34

[199] Mao Tan, Siping Yuan, Shuaihu Li, Yongxin Su, Hui Li, and Feng biao He. Ultra-short-term industrial power demand forecasting using lstm based hybrid ensemble learning. *IEEE Transactions on Power Systems*, **35**:2937–2948, 2020. 147

[200] Jonathan A. Tepper, Mahmud S. Shertil, and Heather M. Powell. On the importance of sluggish state memory for learning long term dependency. *Knowl. Based Syst.*, **96**:104–114, March 2016. xix, 2, 32, 33, 34, 36, 38, 41, 43, 47, 56, 57, 64, 69, 74, 89, 92, 109

[201] The University of Pennsylvania. Introduction to ARMA Models Modeling paradigm. 14

[202] thisMatter.com. The random walk and the efficient market hypotheses. https://thismatter.com/money/investments/random-walk-efficient-market-hypotheses.htm. Accessed: 2021-02-08. 18

[203] Anuradha Tomar and Neeraj Gupta. Prediction for the spread of covid-19 in india and effectiveness of preventive measures. *The Science of the Total Environment*, **728**:138762 – 138762, 2020. 82

[204] Matthew H. Tong, Adam D. Bickett, Eric M. Christiansen, and Garrison W. Cottrell. Learning grammatical structure with Echo State Networks. *Neural Networks*, **20**[3]:424–432, April 2007. 33, 34

[205] Xiaojun Tong, Zhu Wang, and Haining Yu. A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. *Computer Physics Communications*, **180**[10]:1795–1801, oct 2009. 32

[206] Pei-Fang (Jennifer) Tsai, Po-Chia Chen, Yen-You Chen, Hao-Yuan Song, Hsiu-Mei Lin, Fu-Man Lin, and Qiou-Pieng Huang. Length of Hospital Stay Prediction at the Admission Stage for Cardiology Patients Using Artificial Neural Network. *Journal of Healthcare Engineering*, **2016**, 2016. 25

[207] UCLA. 3 ARIMA Models. University Lecture, 2011. 14

[208] Claudia Ulbricht. Multi-recurrent Networks for Traffic Forecasting. *Proceedings of the National Conference on Artificial Intelligence*, **2**:883–888, 1994. xix, 2, 32, 38, 41, 42, 43, 46, 55, 56, 64, 86, 109, 113, 147, 173

[209] Department of Commerce US Census Bureau. Manufacturing & trade inventories & sales. https://catalog.data.gov/dataset/manufacturing-trade-inventories-sales. Accessed: 2021-02-04. 58

[210] U.S. Energy Information Administration. Short-term energy outlook. https://www.eia.gov/outlooks/steo/report/global_oil.php. 59

[211] J. Wang. A process level network traffic prediction algorithm based on ARIMA model in smart substation. In *2013 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2013)*, pages 1–5, August 2013. 15

[212] Jianyong Wang, L. Zhang, Quan Guo, and Z. Yi. Recurrent neural networks with auxiliary memory units. *IEEE Transactions on Neural Networks and Learning Systems*, **29**:1652–1661, 2018. 92

[213] Jie Wang, Jun Wang, Wen Fang, and Hongli Niu. Financial Time Series Prediction Using Elman Recurrent Random Neural Networks, 2016. 31

[214] Shaorun Wang, P. Lin, Ruihan Hu, Hao Wang, J. He, Qijun Huang, and Sheng Chang. Acceleration of lstm with structured pruning method on fpga. *IEEE Access*, **7**:62930–62937, 2019. 147

[215] Wenbin Wang and Wenjuan Zhang. Early defect identification: application of statistical process control methods. *Journal of Quality in Maintenance Engineering*, **14**[3]:225–236, August 2008. 12, 13

[216] Xin Wang, Jiabing Xu, Wei Shi, and Jiarui Liu. OGRU: An optimized gated recurrent unit neural network. *Journal of Physics: Conference Series*, **1325**:012089, oct 2019. 91

[217] Mohd Arif Wani. Comparative Study of Back Propagation Learning Algorithms for Neural Networks. *J. Adv. Res. Comput. Sci. Softw. Eng.*, **3**[12]:6, 2013. 27

[218] Liangjiang Wen, Xueyang Zhang, Haoli Bai, and Zenglin Xu. Structured pruning of recurrent neural networks through neuron selection. *Neural networks : the official journal of the International Neural Network Society*, **123**:134–141, 2020. 146

[219] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, **78**[10]:1550–1560, October 1990. 49

[220] Elise Whitley and Jonathan Ball. Statistics review 6: Nonparametric methods. *Critical Care*, **6**:509 – 513, 2002. 78

[221] B. Widrow and M. E. Hoff. Adaptive switching circuits. Technical report, Standford University, 1960. 24

[222] Ronald J. Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity, 1995. 54

[223] G. David Williamson and Ginner Weatherby Hudson. A monitoring system for detecting aberrations in public health surveillance reports. *Statistics in Medicine*, **18**[23]:3283–3298, 1999. 15, 16

[224] A. Wysocki and M. Ławryńczuk. Jordan neural network for modelling and predictive control of dynamic systems. In *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 145–150, August 2015. 29, 30

[225] Hang Xie, Hao Tang, and Yu He Liao. Time series prediction based on narx neural networks: An advanced approach. In *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, **3**, pages 1275–1279, 2009. 28

[226] Tao Xiong, Yukun Bao, and Zhongyi Hu. Beyond one-step-ahead forecasting: Evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Econ.*, **40**:405–415, November 2013. 27

[227] A. H. Yaacob, I. K. T. Tan, S. F. Chien, and H. K. Tan. ARIMA Based Network Anomaly Detection. In *2010 Second International Conference on Communication Software and Networks*, pages 205–209, February 2010. 15

[228] Salisu Wada Yahaya, Ahmad Lotfi, and Mufti Mahmud. A consensus novelty detection ensemble approach for anomaly detection in activities of daily living. *Appl. Soft Comput.*, **83**:105613, 2019. 63

[229] R. Yasdi. Prediction of Road Traffic using a Neural Network Approach. *Neural Computing & Applications*, **8**[2]:135–142, May 1999. 29

[230] M. Yasunaga, N. Masuda, M. Yagyu, Asai Mitsuo, K. Shibata, Mitsuo Ooyama, M. Yamada, Takahiro Sakaguchi, and M. Hashimoto. A self-learning digital neural network using wafer-scale lsi. *IEEE Journal of Solid-state Circuits*, **28**:106–114, 1993. 114

[231] Lean Yu, Shouyang Wang, and Kin Keung Lai. Forecasting crude oil price with an EMD-based neural network ensemble learning paradigm. *Energy Econ.*, **30**[5]:2623–2635, September 2008. 27

[232] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, **31**[7]:1235–1270, 2019. 35, 64, 91

[233] Haoxian Zhang, Murat Ö Balaban, and José C. Principe. Improving pattern recognition of electronic nose data with time-delay neural networks. *Sensors and Actuators, B: Chemical*, **96**[1-2]:385–389, 2003. 26

[234] Matthew Shunshi Zhang and Bradly C. Stadie. One-shot pruning of recurrent neural networks by jacobian spectrum evaluation. *CoRR*, **abs/1912.00120**, 2019. 147

[235] Matthew Shunshi Zhang and Bradly C. Stadie. One-shot pruning of recurrent neural networks by jacobian spectrum evaluation. *CoRR*, **abs/1912.00120**, 2019. 147

[236] Mingda Zhang. Time Series : Autoregressive models AR, MA, ARMA, ARIMA. University Lecture, 2018. 11, 12, 13, 14, 15, 16

[237] Yudong Zhang and Lenan Wu. Crop classification by forward neural network with adaptive chaotic particle swarm optimization. *Sensors*, **11**[5]:4721–4743, 2011. 53

[238] Shisheng Zhong, Xiaolong Xie, Lin Lin, and Fang Wang. Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction. *Neurocomputing*, **238**:191–204, 2017. 33

[239] Lingling Zhou, Ping Zhao, Dongdong Wu, Cheng Cheng, and Hao Huang. Time series model for forecasting the number of new admission inpatients. *BMC Med Inform Decis Mak*, **18**, June 2018. 25

[240] Michael H. Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression, 2018. 147

[241] Yao Zhu, Xiaoliang Fan, J. Wu, Xiao Liu, J. Shi, and C. Wang. Predicting icu mortality by supervised bidirectional lstm networks. In *AIH@IJCAI*, 2018. 35

Appendices

A Preliminary results with the PA-MRN

Preliminary experiments are conducted for the PA-MRN models to identify the best parameters for the NBER prediction task, Oil price prediction task and Covid-19 forecasting. *Note: All models are run for 500 epochs and the memory bank combination associated with the best standard MRN is employed.*

A.1 Business cycle prediction

Preliminary experiments are conducted for the NBER turning points prediction task with a combination of hyperparameters for the PA-MRN. The PA-MRN employed the memory bank combination of the best MRN model for each dataset *(See Table 4.2 for the memory combination of the best MRN models).*

The PA-MRN models are trained with two hidden units [20, 30], three H-TAUs [0, 2, 10], four TAUs [3, 5, 6, 10] and three window sizes [20, 40, 80] to understand the impact on its performance.

A.1.1 Experiments with window size of 80

Table 1 and Table 2 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 80.

Table 1: PA-MRN results for different hyperparameter *(Window size: 80)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
20	0	20	3	0.177	0	0
	2	18		0.356	0	0
	10	10		0.217	0	0
	0	20	5	0.333	0	0
	2	18		0	0	0
	10	10		0.356	0	0
	0	20	6	0.509	0.539	0.125
	2	18		0.378	0	0.177
	10	10		0.217	0	0
	0	20	10	**0.688**	**0.742**	0.761
	2	18		0.697	0.715	0.751
	10	10		0.506	0.734	**0.734**

Table 2: PA-MRN results for different hyperparameter *(Window size: 80)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
30	0	30	3	0.507	0	0
	2	28		0.506	0	0
	10	20		0.523	0	0
	0	30	5	0.506	0	0.125
	2	28		0.571	0	0
	10	20		0.506	0	0
	0	30	6	0.684	0.695	0.493
	2	28		0.697	0	0.611
	10	20		0.436	0.492	0.115
	0	30	10	**0.708**	0.71	0.759
	2	28		0.688	**0.747**	**0.768**
	10	20		0.693	0.698	0.756

A.1.2 Experiments with window size of 40

Table 3 and Table 4 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 40.

Table 3: PA-MRN results for different hyperparameter *(Window size: 40)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
20	0	20	3	0	**0.642**	0.381
	2	18		0	0.023	**0.641**
	10	10		0	0	0.544
	0	20	5	0	0.394	0.48
	2	18		0	0	0.504
	10	10		0.251	0	0.596
	0	20	6	0.356	0.383	0.337
	2	18		0.217	0	0.339
	10	10		0.308	0	0.173
	0	20	10	**0.472**	0.508	0.442
	2	18		0.399	0.537	0.431
	10	10		0.356	0.511	0.403

Table 4: PA-MRN results for different hyperparameter *(Window size: 40)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
30	0	30	3	0.525	0.673	0.647
	2	28		0.251	0.659	**0.681**
	10	20		0.281	**0.704**	0.566
	0	30	5	0.457	0.448	0.474
	2	28		0.399	0	0.562
	10	20		0.419	0.522	0.635
	0	30	6	0.541	0.692	0.411
	2	28		0.419	0.445	0.472
	10	20		0.438	0.451	0.427
	0	30	10	**0.692**	0	0.437
	2	28		0.678	0.588	0.313
	10	20		0.657	0.288	0.356

A.1.3 Experiments with window size of 20

Table 5 and Table 6 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 20.

Table 5: PA-MRN results for different hyperparameter *(Window size: 20)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
20	0	20	3	0.735	0.707	0.707
	2	18		0.726	0.707	0.735
	10	10		0.655	0.68	0.688
	0	20	5	0.719	0.732	0.721
	2	18		**0.736**	0.732	0.125
	10	10		0.719	**0.758**	0.744
	0	20	6	0.641	0.733	0.751
	2	18		0.733	0.733	0.73
	10	10		0.714	0.745	0.754
	0	20	10	0.7	0.716	0.766
	2	18		0.722	0.681	**0.769**
	10	10		0.722	0.756	0.733

Table 6: PA-MRN results for different hyperparameter *(Window size: 20)*

Units	H-TAUs	TAUs	Phase	Dataset		
				Growth	COD	Growth & COD
30	0	30	3	0.735	0.721	0.735
	2	28		0.735	0.701	0.77
	10	20		0.735	0.333	0.747
	0	30	5	**0.747**	0.701	0.736
	2	28		0.72	0.694	0.757
	10	20		0.732	0.681	**0.792**
	0	30	6	0.716	0	0.726
	2	28		0.627	0.457	0.747
	10	20		0.723	0.356	0.783
	0	30	10	0.541	0.72	0.775
	2	28		0.641	0.711	0.763
	10	20		0.714	**0.745**	0.783

A.2 Oil price prediction task

Preliminary experiments are conducted for the Oil price prediction task with a combination of hyperparameters for the PA-MRN. The PA-MRN employed the memory bank combination of the best MRN model for each dataset *(See Table 4.7 for the memory combination of the best MRN models)*.

The PA-MRN models are trained with two hidden units [20, 30], three H-TAUs [0, 2, 10], four TAUs [3, 5, 6, 10] and four window sizes [60, 120, 240, 300] to understand the impact on its performance.

A.2.1 Experiments with window size of 60

Table 7 and Table 8 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 60.

Table 7: PA-MRN results for different hyperparameter *(Window size: 60)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
20	0	20	3	0.70	0.741	1.125	1.472
	2	18		0.533	0.76	1.08	**1.265**
	10	10		**0.378**	**0.713**	**1.077**	1.283
	0	20	5	0.605	2.195	2.196	1.851
	2	18		2.571	1.14	1.107	1.541
	10	10		0.388	1.472	1.088	1.645
	0	20	6	2.228	2.387	1.941	2.163
	2	18		1.694	1.619	1.412	2.128
	10	10		0.439	0.797	1.091	1.355
	0	20	10	2.059	2.153	1.618	2.15
	2	18		0.965	2.063	1.283	2.079
	10	10		0.429	0.733	1.112	1.361

Table 8: PA-MRN results for different hyperparameter *(Window size: 60)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
30	0	30	3	0.552	**0.718**	1.141	1.525
	2	28		0.715	0.84	1.188	1.874
	10	20		**0.429**	0.727	**1.086**	1.671
	0	30	5	2.306	1.76	2.249	2.163
	2	28		0.577	2.211	2.144	2.166
	10	20		0.47	0.722	1.09	**1.298**
	0	30	6	2.075	2.06	1.68	1.928
	2	28		1.824	2.219	1.393	2.143
	10	20		0.447	0.927	1.102	1.898
	0	30	10	2.293	2.198	1.835	2.164
	2	28		1.798	1.99	1.198	2.295
	10	20		0.632	1.106	1.121	1.778

A.2.2 Experiments with window size of 120

Table 9 and Table 10 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 120.

Table 9: PA-MRN results for different hyperparameter *(Window size: 120)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
20	0	20	3	0.662	0.742	0.922	1.155
	2	18		0.69	0.748	0.927	**1.141**
	10	10		**0.445**	0.735	0.961	1.184
	0	20	5	0.643	0.708	0.928	1.159
	2	18		0.663	0.732	**0.92**	1.162
	10	10		0.489	0.726	0.994	1.168
	0	20	6	0.643	0.731	0.936	1.172
	2	18		0.64	0.732	0.925	1.172
	10	10		0.463	**0.699**	0.996	1.244
	0	20	10	0.611	1.286	0.974	1.189
	2	18		0.715	0.715	0.956	1.162
	10	10		0.524	0.712	1.058	1.201

Table 10: PA-MRN results for different hyperparameter *(Window size: 120)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
30	0	30	3	0.652	0.744	0.948	1.208
	2	28		0.63	0.739	0.951	1.2
	10	20		0.537	0.749	0.98	1.218
	0	30	5	0.558	**0.674**	0.956	1.217
	2	28		0.636	0.735	0.95	1.225
	10	20		1.796	0.747	0.977	1.273
	0	30	6	0.633	0.887	0.963	1.201
	2	28		0.716	0.734	0.959	**1.197**
	10	20		0.684	0.689	**0.922**	1.225
	0	30	10	1.567	1.017	1.008	1.356
	2	28		0.684	0.688	1.009	1.205
	10	20		**0.531**	0.969	1.34	1.25

A.2.3 Experiments with window size of 240

Table 11 and Table 12 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 240.

Table 11: PA-MRN results for different hyperparameter *(Window size: 240)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
20	0	20	3	1.265	1.855	1.4	1.548
	2	18		1.438	2.134	1.432	1.486
	10	10		1.556	1.332	1.451	1.533
	0	20	5	**1.137**	1.175	1.455	1.558
	2	18		1.326	1.146	1.454	1.502
	10	10		1.267	**1.053**	1.4	1.497
	0	20	6	1.411	1.588	1.417	1.558
	2	18		1.379	1.341	1.4	1.495
	10	10		1.204	1.731	1.386	1.493
	0	20	10	1.275	1.258	**1.347**	1.597
	2	18		1.308	1.33	**1.347**	**1.427**
	10	10		1.276	1.477	1.431	1.559

Table 12: PA-MRN results for different hyperparameter *(Window size: 240)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
30	0	30	3	1.584	1.193	1.506	1.45
	2	28		1.008	1.825	1.624	1.45
	10	20		1.359	2.024	1.526	1.481
	0	30	5	1.119	1.504	1.523	1.524
	2	28		1.084	1.142	1.412	1.502
	10	20		1.583	1.517	**1.326**	1.368
	0	30	6	1.118	1.357	1.355	1.481
	2	28		1.469	1.389	1.415	1.429
	10	20		1.672	2.34	1.445	1.416
	0	30	10	1.249	1.318	1.695	1.451
	2	28		**0.693**	**0.942**	1.594	**1.349**
	10	20		0.714	2.826	1.357	1.605

A.2.4 Experiments with window size of 300

Table 13 and Table 14 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 300.

Table 13: PA-MRN results for different hyperparameter *(Window size: 300)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
20	0	20	3	**2.029**	2.058	1.883	2.058
	2	18		2.037	2.067	1.646	2.05
	10	10		2.05	2.064	1.907	2.066
	0	20	5	2.032	2.066	1.745	2.05
	2	18		2.052	2.052	1.38	2.058
	10	10		2.042	2.056	1.688	2.048
	0	20	6	2.044	2.055	1.639	2.05
	2	18		2.043	2.05	**1.176**	**2.043**
	10	10		2.046	**2.049**	1.692	2.056
	0	20	10	2.047	2.06	1.271	**2.043**
	2	18		2.045	2.057	1.218	2.044
	10	10		2.052	2.061	1.362	2.047

Table 14: PA-MRN results for different hyperparameter *(Window size: 300)*

Units	H-TAUs	TAUs	Phase	Horizon - 1	Horizon - 3	Horizon - 6	Horizon - 12
30	0	30	3	2.061	2.069	1.635	2.038
	2	28		**2.049**	2.064	**1.498**	2.041
	10	20		2.071	2.054	1.691	2.043
	0	30	5	2.075	2.06	1.645	2.038
	2	28		2.071	2.07	1.912	2.03
	10	20		2.063	2.064	1.749	2.034
	0	30	6	2.069	**2.042**	1.813	2.034
	2	28		2.068	2.07	1.828	2.041
	10	20		2.062	2.06	1.875	2.045
	0	30	10	2.07	2.058	1.774	2.032
	2	28		2.058	2.059	1.744	2.042
	10	20		2.062	2.065	1.76	**2.028**

A.3 Covid-19 forecasting

Preliminary experiments are conducted for Covid-19 forecasting with a combination of hyperparameters for the PA-MRN. The PA-MRN employed the memory bank combination of the best MRN model for each dataset; confirmed cases [0, 3, 0] and death cases [4, 4, 4].

The PA-MRN models are trained with two hidden units [20, 30], three H-TAUs [0, 2, 10], four TAUs [3, 5, 6, 10] and five window sizes [15, 25, 35, 45, 55] to understand the impact on its performance. The results for a window size of 45 and 55 are not presented, as the MAPE score is significantly large.

A.3.1 Experiments with window size of 15

Table 15 and Table 16 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 15.

Table 15: PA-MRN results for different hyperparameter *(Window size: 15)*

Units	H-TAUs	TAUs	Phase	confirmed	death
20	0	20	3	71.86	52.35
	2	18		71.37	52.05
	10	10		72.41	52.48
	0	20	5	71.53	52.21
	2	18		71.51	51.84
	10	10		71.51	52.61
	0	20	6	71.84	51.63
	2	18		**70.54**	51.65
	10	10		71.17	51.80
	0	20	10	70.71	51.52
	2	18		71.18	**51.23**
	10	10		72.13	52

Table 16: PA-MRN results for different hyperparameter *(Window size: 15)*

Units	H-TAUs	TAUs	Phase	confirmed	death
30	0	30	3	69.24	49.81
	2	28		69.4	50.87
	10	20		70.94	51.18
	0	30	5	70.21	50.33
	2	28		71.06	51.09
	10	20		70.41	49.72
	0	30	6	69.63	49.82
	2	28		70.16	**49.68**
	10	20		69.83	50.21
	0	30	10	69.42	50.51
	2	28		69.5	49.83
	10	20		**66.84**	49.97

A.3.2 Experiments with window size of 25

Table 17 and Table 18 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 25.

Table 17: PA-MRN results for different hyperparameter *(Window size: 25)*

Units	H-TAUs	TAUs	Phase	confirmed	death
20	0	20	3	58.44	8.67
	2	18		58.29	4.41
	10	10		60.34	2.39
	0	20	5	58.7	4.87
	2	18		58.58	5.5
	10	10		**57.22**	3.12
	0	20	6	60.27	3.86
	2	18		60.33	6.02
	10	10		58.18	3.87
	0	20	10	61.82	**0.66**
	2	18		59.76	2.75
	10	10		59	5.54

Table 18: PA-MRN results for different hyperparameter *(Window size: 25)*

Units	H-TAUs	TAUs	Phase	confirmed	death
30	0	30	3	58.98	6.97
	2	28		59.4	6.9
	10	20		57.78	7.27
	0	30	5	57.83	2.67
	2	28		**54.71**	5.93
	10	20		59.23	7.88
	0	30	6	58.58	4.76
	2	28		57.31	**0.81**
	10	20		57.61	4.07
	0	30	10	58.37	13.35
	2	28		59.84	2.68
	10	20		57.63	5.73

A.3.3 Experiments with window size of 35

Table 19 and Table 20 present the results of the PA-MRN models with 20 and 30 hidden units *respectively*, employing a combination of hyperparameters with a window size of 35.

The hyperparameters (H-TAUs, phase, window sizes) associated with the best

Table 19: PA-MRN results for different hyperparameter *(Window size: 35)*

Units	H-TAUs	TAUs	Phase	confirmed	death
20	0	20	3	5.87	2.73
	2	18		4.76	2.28
	10	10		5.28	2.75
	0	20	5	**4.57**	3.56
	2	18		5.36	0.97
	10	10		5.06	1
	0	20	6	4.88	0.93
	2	18		5.9	**0.91**
	10	10		4.71	1.91
	0	20	10	11.43	10.81
	2	18		5.64	1.73
	10	10		7.26	3.64

Table 20: PA-MRN results for different hyperparameter *(Window size: 35)*

Units	H-TAUs	TAUs	Phase	confirmed	death
30	0	30	3	7.02	9423.19
	2	28		7.97	4612.62
	10	20		6.56	3147.62
	0	30	5	4.71	39.87
	2	28		9.53	1298.07
	10	20		18.7	569.21
	0	30	6	5.08	2467.25
	2	28		4.68	258.73
	10	20		**4.56**	654.06
	0	30	10	5.31	1572.63
	2	28		6.85	476.69
	10	20		4.89	81.8

PA-MRN models from the preliminary experiments are used to conduct the experiments presented in Section 5.3.1, Section 5.3.2 and Section 5.3.4.

B Hyper-parameters for best MRN models

The hyper-parameters associated with the best standard MRN models for the M3 sales prediction & Covid-19 forecasting task are presented.

B.1 M3 Competition prediction

Table 21 presents the hyper-parameters of the best models for the M3 sales prediction.

Table 21: The best memory bank combinations and window sizes for the M3 Competition prediction

Series	Memory bank combination	Window size
N2516	[4, 4, 2]	40
N2521	[3, 2, 4]	10
N1807	[4, 0, 2]	40
N1908	[4, 4, 2]	40
N2012	[3, 3, 0]	40
N2159	[0, 3, 0]	40
N2158	[2, 0, 0]	40
N2150	[0, 0, 3]	40
N2144	[2, 0, 4]	10
N1918	[3, 4, 4]	10

B.2 Covid-19 forecasting

Table 22 presents the hyper-parameters of the best models for Covid19 forecasting.

Table 22: The best memory bank combinations and window sizes for Covid-19 forecasting

Series	Memory bank combination	Window size
Confirmed	[0, 3, 0]	35
Death	[4, 4, 4]	25