Essays on Time Series Analysis and Statistical Machine Learning

Von der Wirtschaftswissenschaftlichen Fakultät der
Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor(in) der Wirtschaftswissenschaften
— Doctor rerum politicarum —

genehmigte Dissertation

von

M.Sc. Johanna Meier
geboren am 02.06.1995 in Lemgo

2023

Referent: Prof. Dr. Philipp Sibbertsen, Leibniz Universität Hannover
Korreferent: Prof. Dr. Marcel Prokopczuk, Leibniz Universität Hannover
Tag der Promotion: 05.10.2023

Acknowledgements

First and foremost, I would like to thank my supervisor and co-author Prof. Dr. Philipp Sibbertsen for giving me the opportunity to earn my PhD degree at the Institute of Statistics as well as for his tireless support and guidance over the last three years. I would also like to express my gratitude to Prof. Dr. Marcel Prokopczuk for taking his time to be my second examiner, to Prof. Dr. Kay Blaufus for chairing my examination board, as well as to Dr. Ute Lohse for advising.

Further, I am grateful to my co-authors Dr. Pushpa Dissanayake, Teresa Flock, and Ziang Niu for countless inspiring discussions, and in particular to Dr. François-Xavier Briol for continuing to share his experience during my PhD journey.

Special thanks are due to my former and present colleagues for creating a pleasant working environment and their moral support in difficult times. I would like to particularly mention Teresa Flock and Vivien Less who accompanied me since day one of my PhD.

Lastly, I would like to recognise the unconditional help of my family: the continuous reassurance by my parents, the constant encouragement by my brother, and the emotional support by my fiancé. To all of you: my deepest gratitude.

ABSTRACT

This thesis encompasses three research articles contributing to the fields of time series analysis and statistical machine learning. Firstly, we develop a peaks-over-threshold approach, which captures both short- and long-term correlations in the underlying time series in order to model the clustering behaviour in high-threshold exceedances. The suggested model is motivated by and applied to oceanographic data. Secondly, we propose an efficient discrepancy-based inference approach for intractable generative models based on quasi-Monte Carlo methods. We demonstrate that this method substantially reduces the computational cost of estimating the model parameters in various applications of academic and practical interest. Thirdly, we suggest training methods for deep sequential models, which improve the forecast precision when facing structural breaks in the in-sample period. These mitigation strategies are examined in an extensive simulation study and utilised to forecast energy data. As the developed theory in this thesis is very versatile, it is applicable to a broad range of data types as well as research fields, and in particular to economic time series.

Keywords: Peaks-over-threshold · Extremal clustering · Long-range dependence · Intractable generative models · Discrepancy-based inference · Quasi-Monte Carlo · Deep sequential models · Structural breaks

CONTENTS

Contents i

List of Figures iii

List of Tables v

1 Introduction 1

2 Modelling Short- and Long-Term Dependencies of Clustered High-Threshold Exceedances in Significant Wave Heights 4

3 Discrepancy-Based Inference for Intractable Generative Models Using Quasi-Monte Carlo 5

4 Forecasting Facing Structural Breaks Using Deep Sequential Models 6

- 4.1 Introduction 6
- 4.2 Review of Deep Sequential Architectures 8
- 4.3 Fostering Forecast Robustness Facing Mean Shifts 10
  - 4.3.1 Multiple Mean Shift Model 10
  - 4.3.2 Effect of Mean Shifts on the Forecast Precision 10
  - 4.3.3 Weighted Loss Functions 12
    - 4.3.3.1 Applying Weighted Loss Functions to a Toy Example . . . 16
  - 4.3.4 Excluding Pre-Break Observations 17
- 4.4 Simulation Study 18
  - 4.4.1 Experimental Setup 18
    - 4.4.1.1 Data Generating Processes 19
    - 4.4.1.2 Mean Shift Scenarios 19
    - 4.4.1.3 Model Setups 20
  - 4.4.2 Weighted Loss Functions 20
    - 4.4.2.1 Setting 1 + Scenario 1 21
    - 4.4.2.2 Setting 1 + Scenario 2 22
    - 4.4.2.3 Setting 2 + Scenario 1 24
    - 4.4.2.4 Setting 2 + Scenario 2 25
  - 4.4.3 Excluding Pre-break Observations 27

        4.4.3.1 Setting 1 27
        4.4.3.2 Setting 2 28
4.5 Application to the Voltage of the German Electric Power Grid 29
    4.5.1 Data and Experimental Setup 29
    4.5.2 Results 31
        4.5.2.1 Best Performance 31
        4.5.2.2 Overall Performance 33
4.6 Conclusion 34
Appendix 35
    Appendix A Additional Results: Models 35
    Appendix B Additional Results: Simulation Study 37
    Appendix B Additional Results: Application 52

Bibliography **53**

List of Figures

4.1 Illustration of the effect of the break location on the prediction of deep sequential models trained with MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for RNN (blue), GRU (green), and LSTM (orange) architectures. The realisations of the DGP before and after the break point are given as reference as well as the naive prediction $\hat{y}_t = \overline{y}_{1:T}$. 11

4.2 Comparison of exponential, Rayleigh, Bartlett, Parzen and Tukey-Hanning weighting schemes with different choices of the parameter α for a sample size of $T = 1000$. 15

4.3 Illustration of the effect of training RNNs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$. 16

4.4 Illustration of the effect of varying the parameter α in the weighted loss when training RNNs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using MSE loss, for which $\gamma(t) = 1/T$. 17

4.5 Voltage of the electric power grid measured every second in volt over the course of a day at Zinkmattenstraße in Freiburg, Germany. Data is aggregated to measurements every minute by averaging. The dashed vertical lines represent the estimated break points and the red horizontal bars the 95% confidence intervals. 30

4.6 *Overall performance* - distribution of the RMSE on the test set for 100 trained RNN architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights. 33

A.1 Illustration of the effect of training LSTMs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$. 35

A.2 Illustration of the effect of training GRUs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$. 36

A.3 Illustration of the effect of varying the parameter α in the weighted loss when training LSTMs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$. 36

A.4 Illustration of the effect of varying the parameter α in the weighted loss when training GRUs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$. 37

C.1 *Overall performance* - distribution of the RMSE on the test set for 100 trained LSTM architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights. 52

C.2 *Overall performance* - distribution of the RMSE on the test set for 100 trained GRU architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights. 52

LIST OF TABLES

4.1 Considered weighting schemes for the weighted loss function defined in (4.5). . 14

4.2 $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$. 21

4.3 $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$. 23

4.4 $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$. 24

4.5 $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$ 26

4.6 $\mathrm{MSFE}_{\mathrm{PB}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 combined with Scenario 1. 27

4.7 $\mathrm{MSFE}_{\mathrm{PB}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 combined with Scenario 1. 28

4.8 Break points and corresponding 95% confidence intervals estimated using the Bai and Perron (1998, 2003) procedure. 30

4.9 *Best performance* - $\mathrm{MSFE}_i/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for the model with the smallest validation loss of all 100 trained models. 32

B.1 $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$. 37

B.2 $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 38

B.3 $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 38

B.4 $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 39

B.5 $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$. 40

B.6 $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 41

B.7 $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 42

B.8 $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 43

B.9 $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$. 44

B.10 $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 44
B.11 $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 45
B.12 $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. 45
B.13 $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ 46
B.14 $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$ 47
B.15 $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$ 48
B.16 $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$ 49
B.17 $\mathrm{MSFE}_{\mathrm{PB}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 combined with Scenario 2. 50
B.18 $\mathrm{MSFE}_{\mathrm{PB}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 combined with Scenario 2. 51

Chapter 1

Introduction

This thesis comprises three separate research articles contributing to the realm of time series analysis and statistical machine learning. While the first two articles solely focus on classical time series methods and the foundations of statistical machine learning respectively, the third article attempts to bridge the gap between both worlds. None of the articles appears to be related to econometric problems at first glance, but the developed theory is very versatile and therefore applicable to various data types and research fields, notably economics.

The first research article is concerned with classical time series methods in the context of extreme value theory (EVT): EVT is designed to analyse the occurrence of extreme events and thus aims at characterising the tail behaviour of a distribution. It addresses two key problems when estimating the tail of a distribution: (1) the tail distribution may differ from the distribution of the remaining data, and (2) rare events may not even be present in the data as a result of limited data availability (McNeil & Frey, 2000). One fundamental approach to EVT is the peaks-over-threshold (POT) method, which models extreme observations exceeding a particular high threshold (Ferreira & de Haan, 2015). The assumption of identically and independently distributed (iid) data underpinning the classical POT approach, however, is violated in many practical settings leading to high-threshold exceedances occurring in a clustered fashion. This clustering behaviour can arise from short- and long-term dependencies in the underlying time series. We develop a POT approach, which captures both types of serial correlation simultaneously, and thereby models the clustering behaviour of high-threshold exceedances. We apply our method to significant wave heights, which are frequently found to not only exhibit short-range but also long-range dependence (Cabrera & Rodríguez, 2011).

Of course, this research article is motivated by the dynamics in oceanographic data, but its range of possible applications is not restricted to this type of data: POT methods have also been found to be essential for financial risk management (Chavez-Demoulin et al., 2005; Herrera & Schipp, 2014) as well as insurance (Lee et al., 2012). Since financial data such as stock index returns (Bhardwaj & Swanson, 2006) or inflation rates (Durham et al., 2019) have recurrently been identified as being governed by not only short-term dynamics but also long-range dependence, our method could prove especially useful for these data types, as it extends existing approaches by considering short- and long-range correlations simultaneously instead of focusing on one of these dynamics alone.

The second research article addresses the foundations of statistical machine learning: statisticians and machine learning researchers develop ever more complex models. The model complexity poses a particular challenge for parameter estimation, since standard tools are no longer adequate. Standard inference approaches such as maximum likelihood estimation (Rossi, 2018) and Bayesian inference (Box & Tiao, 2011) rely on having access to a likelihood function representing the model of interest. However, if the modelling task becomes highly complex, it may no longer be possible to specify the corresponding likelihood function mathematically or it may be prohibitively expensive to compute it (Marin et al., 2012). The term *intractable generative models* refers to a class of statistical models, for which the likelihood function is unavailable, but it is possible to simulate new observations from the model given fixed parameter values (Cranmer et al., 2020). This class of models requires alternative inference approaches, which may be based on computing a discrepancy between the actual data and the data simulated from the generative model. In order to be able to handle high-dimensional or large-scale datasets, these approaches to parameter inference need to be scalable and computationally efficient. In this article, we develop methods, which are capable of significantly reducing the number of data points required to estimate the parameters of these generative models resulting in a substantial reduction in computational cost of the inference procedure. For this, we make use of quasi-Monte Carlo (QMC) point sets, which allow to generate a more diverse set of samples from the generative model compared to random sampling. Our theoretical results are verified in a simulation study and we demonstrate that our inference approach can efficiently train a variational autoencoder on the MNIST dataset to generate images of hand-written digits.

My contribution to this joint research project (co-first author) especially encompasses the conceptualisation and implementation of all numerical experiments required to demonstrate that our developed theory does not only work but is also relevant for practitioners. For this, I conceived three categories of experiments: simulations with simple generative models, for which either (1) all of our assumptions are satisfied or (2) particular assumptions are violated, as well as (3) the application of a complex generative model to a relevant dataset, which is of practical interest but still satisfies all assumptions. Since QMC methods rely on more restrictive assumptions than random sampling, the question of whether our methodology is robust to assumption violations is crucial for its relevance in practice. For the first experiment category, a uniform distribution is studied. Of course, this example does not require inference tools for intractable generative models, but it is a simple way of studying the sample complexity of QMC point sets in discrepency-based inference and by that serving as a proof of concept. For the second category of experiments, Gaussian, bivariate Beta and multivariate g-and-k distributions are chosen in order to cover univariate as well as multivariate (intractable) generative models. The violated assumptions include unbounded generators and rejection sampling algorithms proving a wide applicability of discrepancy-based inference using QMC while also complementing the literature which discusses the usability of QMC methods beyond their assumptions (see e.g. Owen, 2013). For the final experiment category, a generative neural network is examined, which is trained as the decoder network of a variational autoencoder constituting a popular choice for generating realistic looking images. The chosen MNIST dataset forms a benchmark dataset for deep learning algorithms.

While this article mainly focuses on academically motivated applications, the developed methods are not limited to those presented: intractable generative models also play a large role in econometrics and finance applications. One example is the class of stochastic volatility models (Kim et al., 1998), which belongs into the standard toolkit for modelling volatility in stock return time series. A further example are hidden Markov models, frequently used to reproduce the stylised facts of daily return series (Rydén et al., 1998). Moreover, in operations research, queuing models constitute yet another member of the class of intractable generative models. For instance, the M/G/1 queue model refers to a single-server fist come first serve queue with Markovian arrival times and a general distribution of service times, from which it is easy to sample, but it has an intractable likelihood (Fearnhead & Prangle, 2012).

The final research article is focusing on bridging the gap between classical time series analysis and deep learning: forecasting any economic or physical variable builds upon the assumption that the data generating process (DGP) is governed by constant parameters over the entire pre-forecast sampling period. Permanent shifts in the parameters of the GDP are referred to as *structural breaks*. If they occur in the pre-forecast period they are well known to lead to biased model parameter estimates causing forecast failure. For classical time series models such as autoregressive moving average (ARMA) processes, the consequences of structural breaks in the in-sample period are well-studied (cf. Clements & Hendry, 2006), so that a range of mitigation strategies were developed in response (e.g. Gardner, 2006; Pesaran & Pick, 2011). Despite the recent popularity of deep sequential models such as recurrent neural networks (RNNs), long-short-term-memory (LSTM) and gated-recurrent unit (GRU) architectures for time series forecasting (c.f. Lim & Zohren, 2021), there is little work addressing the effects of in-sample structural breaks on their forecast performance. For the particular problem of mean shifts, we therefore discuss the consequences for the forecast precision of deep sequential models and propose mitigation strategies to improve their forecast robustness. The mitigation strategies are evaluated in an extensive simulation study and their practical benefit demonstrated on energy data.

In this article, the developed methodology is applied to forecast the voltage of the German electric power grid, but the scope of application is not confined to energy data: since deep sequential architectures are well suited for modelling dependence structures in large datasets, economic variables sampled at high frequencies constitute a typical use case. Real interest rates for instance have been found to be subject to structural breaks (Rapach & Wohar, 2005). Similar results are known for European carbon prices (Alberola et al., 2008) as well as stock market (Andreou & Ghysels, 2002) and exchange rate volatilities (Rapach & Strauss, 2008). All of these financial variables are available at a daily or even higher frequency.

The remaining thesis is structured as follows. Chapter 2 presents the first research article on modelling short- and long-term dependencies of clustered high-threshold exceedances in significant wave heights. Chapter 3 is dedicated to the second research article on discrepancy-based inference for intractable generative models using quasi-Monte Carlo, while Chapter 4 is concerned with the final research article on forecasting facing structural breaks using deep sequential models.

CHAPTER 2

MODELLING SHORT- AND LONG-TERM DEPENDENCIES OF CLUSTERED HIGH-THRESHOLD EXCEEDANCES IN SIGNIFICANT WAVE HEIGHTS

Co-authored with Pushpa Dissanayake, Teresa Flock, and Philipp Sibbertsen.
Published in Mathematics *2021, 9(21), 2817.*

https://doi.org/10.3390/math9212817

CHAPTER 3

DISCREPANCY-BASED INFERENCE FOR INTRACTABLE GENERATIVE MODELS USING QUASI-MONTE CARLO

Co-authored with Ziang Niu and François-Xavier Briol.
Published in Electronic Journal of Statistics *2023, 17(1), 1411-1456.*

https://doi.org/10.1214/23-EJS2131

Chapter 4

Forecasting Facing Structural Breaks Using Deep Sequential Models

4.1 Introduction

Forecasting economic variables relies on the assumption that the parameters of the underlying data-generating process (DGP) remain constant over the entire pre-forecast sampling period. Structural breaks can be defined as permanent shifts in the parameters of the DGP, and if they occur in the pre-forecast period they can lead to biased model parameter estimates and cause forecast failures. For classical time series models, such as autoregressive moving average (ARMA) models, the consequences of structural breaks on parameter estimates and forecasts are well studied (cf. Clements & Hendry, 2006), and various mitigation strategies have been proposed: estimating the break location and excluding pre-break observations from estimation (Bai & Perron, 1998, 2003), exponential smoothing (ExpS) (Gardner, 2006), or forecast combinations (Pesaran & Pick, 2011) amongst others.

Yet, in recent years, deep sequential models have gained popularity for time series prediction. Originally developed for sequence modelling, they proved to be just as suitable for temporal forecasting applications. Even though this class of models is more difficult to interpret than classical time series models, the ability to capture complex relationships makes for a useful alternative when prioritising forecasting performance. See Lim and Zohren (2021) for a recent review on deep learning models for time series. Despite their popularity, there is little work on mitigating the effects of in-sample structural breaks on the forecasting performance of deep sequential models.

In the machine learning community, structural breaks fall into the category of distribution shift problems. Distribution shift is a collective term for situations, in which the statistical properties of the data change over time, i.e. the distribution shifts temporally. However, the data type is not limited to time series (Duan et al., 2023). The distribution shift problem is usually approached via domain adaptation (Tzeng et al., 2017) or domain generalisation (Wang et al., 2022), where the underlying idea is to learn common knowledge, which is transferable between domains in spite of disparities between distributions. In case of distribution shifts in time series, i.e. non-stationary time series, these methods are not easily transferable (Kim et al., 2021). The main challenge is that distributions may change constantly, but it is unknown how to best characterise the distributions in order to infer the common knowledge. Complex modelling solutions have been proposed such as AdaRNN (Du et al., 2021), characterising the worst-case distribution shift in the time series based on the principle of maximum entropy, or a hypernetwork-based framework (Duan et al., 2023), which jointly learns the time-varying distributions and corresponding forecasting models. Alternatively, feature engineering could be deployed for deep sequential models. This involves identifying time-varying features representing the factors, which cause the drift in parameters, and use them as exogenous covariates (Lim & Zohren, 2021). If the covariates carry sufficient information about the distribution shift, this approach allows the model to learn accurate conditional distributions based on these exogenous signals. All of these approaches have in common that (1) they are laborious to implement in practice, and (2) they are not specific to the problem of structural breaks, which imply permanent instead of constant changes of the parameters in the DGP.

In this work, we want to specifically improve the forecasts of popular deep sequential models such as recurrent neural networks (RNNs), long short-term memory (LSTM) and gated recurrent unit (GRU) architectures when facing in-sample structural breaks. When forecasting in the presence of structural breaks, we most certainly face a bias-variance trade-off: using a model estimation window including a structural break may lead to biased out-of-sample forecasts, but may simultaneously reduce the forecast variance due to the larger sample size (Tian & Anderson, 2014). Therefore, we want to assess how simple mitigation strategies can help balance the trade-off between forecast bias and error variance. For this, we focus on forecast failure caused by changes in the previous unconditional mean in the pre-forecast data, i.e. in-sample mean shifts.

de Boom et al. (2019) identify a loss, which decays at each time step, as a possible future research track for improving the performance of character-level RNNs in the context of natural language processing (NLP). We show that this is also a promising direction for fostering the robustness of deep sequential models when facing structural breaks. We aim at finding a suitable weighting scheme for the training loss, which reduces the detrimental effect of mean shifts, while retaining relevant information on the dependence structure of the underlying DGP. Moreover, we demonstrate that adopting a popular approach from the classical time series literature, where we estimate the break point and discard all pre-break observations from the training set, can successfully mitigate the effect of in-sample mean shifts in certain situations. In order to give recommendations as to when which approach can provide the largest forecast precision improvements, we conduct an extensive simulation study. We further illustrate the practical value of the proposed mitigation strategies using real-world data.

The paper is structured as follows. Section 2 reviews deep sequential architectures. Section 3 then investigates the robustness of these models against in-sample mean shifts and proposes mitigation strategies based on weighted loss functions as well as estimates of the break points. In section 4, we conduct an extensive simulation study, while section 5 applies the proposed methodology to the voltage of the German electric power grid. Section 6 concludes.

4.2 Review of Deep Sequential Architectures

Deep sequential models have historically been developed for sequence modelling, achieving strong results on a wide range of natural language processing tasks (Young et al., 2018). The class of deep sequential models typically encompasses RNNs and their more complex extensions. Since time series data has a natural interpretation as sequences of inputs and targets, various RNN-based architectures have been proposed in the context of temporal forecasting applications (Lim et al., 2020; Rangapuram et al., 2018; Salinas et al., 2020; Wang et al., 2019). The core characteristic of these models is an internal memory state representing a compact summary of past information, which is sequentially updated as new observations become available.

Generally, a recurrent network takes the input sequence $\{x_t\}_{t\geq 1}$ to output the sequence $\{z_t\}_{t\geq 1}$ in order to model the target sequence $\{y_t\}_{t\geq 1}$, where $x_t \in \mathbb{R}^q$, $z_t \in \mathbb{R}^p$, and $y_t \in \mathbb{R}^p$ with $p, q \in \mathbb{Z}$. We assume that the data-generating process (DGP) is of the following form:

$$y_t = z_t + u_t, \quad \text{for } t \in \mathbb{Z}, \tag{4.1}$$

where $\{u_t\}_{t\geq 1}$ is a sequence of p-dimensional independent and identically distributed (iid) random vectors. This corresponds to the additive error assumption utilised for popular loss functions measuring the distance between output and target sequences, such as ℓ_1, ℓ_2, Huber and quantile loss. Further, we assume that $x_t = y_t$, i.e. there is no exogenous input, which allows to analyse the forecast performance of the models for time series in their most simple form. Then, the one-step-ahead forecast of a recurrent network takes the form

$$z_{T+1} = f(y_{1:T}),$$

where $f(\cdot)$ is the prediction function learnt by the model. This prediction function usually consists of a series of nonlinear, recurrent layers. For an RNN-based architecture, the prediction z_{T+1} is based on the hidden internal state $h_t \in \mathbb{R}^q$ at time T, which captures past information, so that such a recurrent layer can be formulated as

$$z_{t+1} = g(W_{zh} h_t + b_z), \quad \text{for } t \in \{1, \ldots, T\},$$

where $g(\cdot)$ denotes an element-wise output function, and $W_{\cdot} \in \mathbb{R}^{p\times q}$ and $b_{\cdot} \in \mathbb{R}^p$ are the linear weights and biases of the network respectively. The RNN-variants typically differ in the way the hidden internal state evolves.

One of the simplest RNNs for time series prediction is the Elman RNN (Elman, 1990), for which the hidden internal state is updated according to

$$h_t = \lambda(W_{hh} h_{t-1} + W_{hy} y_{t-1} + b_h)$$

where $y_0 = 0$, $h_0 = 0$, and $\lambda(\cdot)$ is an element-wise activation function.

Considering longer ranging dependence structures, the application scope of simple RNN-variants can be limited due to drawing on an infinite history of data in the hidden state (Bengio et al., 1994; Hochreiter et al., 2001), which may give rise to vanishing or exploding gradients (Goodfellow et al., 2016). LSTMs (Hochreiter & Schmidhuber, 1997) improve the gradient flow within the network by introducing a cell state $c_t \in \mathbb{R}^q$, which preserves long-term information and is governed by the following gating operations:

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fy}y_{t-1} + b_f), \\ i_t &= \sigma(W_{ih}h_{t-1} + W_{iy}y_{t-1} + b_i), \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{oy}y_{t-1} + b_o), \end{aligned}$$

with $y_0 = 0$, $h_0 = 0$, $f_t, i_t, o_t \in \mathbb{R}^q$, and $\sigma(\cdot)$ denoting an element-wise sigmoid function. We refer to f_t as the forget gate, i_t as the input gate, o_t as the output gate. These gates alter both the hidden and the cell states as follows:

$$\begin{aligned} \tilde{c}_t &= \tanh(W_{ch}h_{t-1} + W_{cy}y_{t-1} + b_c), \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot c_{t-1}, \\ h_t &= o_t \odot \tanh(c_t), \end{aligned}$$

where $\tilde{c}_t \in \mathbb{R}^q$ denotes the candidate cell state, $\tanh(\cdot)$ the element-wise hyperbolic tangent function, and $\odot$ the element-wise (Hadamard) product.

The GRU (Cho et al., 2014) is a successor to LSTMs simplifying the gating mechanics, while retaining a similar performance. The gating operations controlling the information flow can be stated as

$$\begin{aligned} s_t &= \sigma(W_{sh}h_{t-1} + W_{sh}y_{t-1} + b_s), \\ r_t &= \sigma(W_{rh}h_{t-1} + W_{ry}y_{t-1} + b_r), \end{aligned}$$

with $y_0 = 0$, $h_0 = 0$, and $s_t, r_t \in \mathbb{R}^q$. s_t and r_t are referred to as the update and reset gate respectively, which modify the hidden state as below

$$\begin{aligned} \tilde{h}_t &= \tanh\left(W_{hr}(r_t \odot h_{t-1}) + W_{hy}y_{t-1}\right), \\ h_t &= s_t \odot \tilde{h}_t + (1 - s_t) \odot h_{t-1}, \end{aligned}$$

where $\tilde{h}_t \in \mathbb{R}^q$ is the candidate hidden state.

To train any of these RNN-variants, we need to select a suitable loss function for the parameter optimisation. The most popular choice in a regression setting is the mean squared error (MSE) loss:

$$l_{\text{MSE}} = \frac{1}{T}\|y_t - z_t\|_2^2, \quad \text{for } t \in \{1, \dots, T\}. \tag{4.2}$$

Alternatives such as ℓ_1, ℓ_2 or Huber loss can also be considered of course. The deep sequential models can then be trained using backpropagation through time combined with a suitable optimisation algorithm.

Following this brief review of deep sequential architectures, we will now investigate the robustness of these models against in-sample mean shifts and propose two different mitigation strategies.

4.3 Fostering Forecast Robustness Facing Mean Shifts

In this section, we will first specify the considered multiple mean shift model (Section 4.3.1) and then discuss how mean shifts in the training data influence the forecast precision of deep sequential models (Section 4.3.2). Afterwards, we propose mitigation strategies in the form of weighted loss functions (Section 4.3.3) and excluding pre-break observations (Section 4.3.4).

4.3.1 Multiple Mean Shift Model

In order to further discuss how the forecast of deep sequential models is affected by in-sample structural breaks, we need to specify the type of structural break process we would like to investigate. We consider a multiple mean shift model, in which the stochastic process $\{y_t\}_{t\geq 1}$ follows

$$y_t = \mu + \sum_{k=1}^{K} \beta_k \mathbb{1}(t^*_{k-1} \leq t < t^*_k) + \epsilon_t. \tag{4.3}$$

We assume that the sequence of regression means $\mu + \sum_{k=1}^{K} \beta_k \mathbb{1}(t^*_{k-1} \leq t < t^*_k)$ is deterministic and $|\mu_t| < \infty$ holds. Further, β_k is a coefficient determining the magnitude of the mean shift in regime k, $\mathbb{1}(t^*_{k-1} \leq t < t^*_k)$ denotes an indicator function, which depends on the location of the break point $t^*_k = \lfloor \tau^*_k T \rfloor$, where $\lfloor \cdot \rfloor$ represents the integer part of its argument. $\tau^*_k \in [0, 1]$ represents the break fraction, T the sample size, and we define $t^*_0 = 0$ and $t^*_K = T$, implying $\tau^*_0 = 0$ and $\tau^*_K = 1$. Finally, $\{\epsilon_t\}_{t\geq 1}$ is a sequence of error terms following some stationary stochastic process. In this notation, a time series with a single mean shift is characterised by two regimes, i.e. $K = 2$.

4.3.2 Effect of Mean Shifts on the Forecast Precision

Having specified the type of structural break scenario we want to investigate, we can now discuss how the forecast performance of deep sequential models is impacted by mean shifts in the training sample. Generally, deep sequential models are trained by minimising some defined forecast error. While model selection for traditional time series models, such as ARMA models, is based on sparsity considerations (e.g. information criteria penalising the number of parameters), deep sequential models are trained based on the same criterion used to evaluate the forecast accuracy. This is of course ideal for forecasting applications as long as the training sample is representative for the forecast period. But what happens if it is not representative due to the presence of a mean shift?

When using any standard regression loss, every prediction error is assigned an equal weight. The MSE loss in (4.2), for instance, attaches a weight of $\frac{1}{T}$ to each squared forecast error. Assuming a single structural break in the training set, this means that pre-break prediction errors and post-break prediction errors are weighted equally during training. How robust an out-of-sample forecast is against the presence of a structural break in the training data, therefore, depends on the ratio of pre-break to post-break observations. The larger the proportion of post-break observations, the less biased the out-of-sample forecast will be. This effect will now be illustrated using a toy example.

Toy Example We generate $T = 500$ data points from a constant DGP of the form

$$y_t = 0.5\mathbb{1}(t \geq t_1^*) \tag{4.4}$$

with $t_1^* = \lfloor \tau_1^* T \rfloor$ and different values of $\tau_1^* \in [0, 0.95]$. We reserve 10% of the data for out-of-sample prediction. Then, we train simple RNN, GRU, and LSTM architectures, consisting of a single hidden layer with $q = 10$ nodes each, for 50 epochs using the MSE loss and a batch size of 128 on the remaining training data in order to predict y_{t+1} from univariate input y_t. Since the DGP is constant, the out-of-sample forecast $\hat{y}_{T+l} = \hat{f}(y_{1:(T+l-1)})$ is constant as well for all $l \geq 1$.

Figure 4.1 reports the results of 1000 repetitions of this toy experiment. Each graphs depicts the mean of the out-of-sample forecast $\hat{y}_{T+1}$ over all repetitions, while the standard deviation of the out-of-sample forecast is represented by error bars. We observe that the larger τ_1^*, i.e. the larger the ratio of pre-break to post-break observations in the training data, the larger is the observed bias of the out-of-sample forecast. Hence, the equal weighting scheme appears to play a crucial role in the resulting forecast failure. However, the predictions still improve over the naive forecast $\hat{y}_{T+1} = \overline{y}_{1:T} = \frac{1}{T}\sum_{t=1}^{T} y_t$.

Figure 4.1 also reveals subtle differences between the considered deep sequential architectures: while RNNs exhibit the smallest bias, they also have the largest variation in the out-of-sample forecast for growing τ_1^*. The LSTM model, designed to capture longer ranging dependencies, is most affected by the mean shift, i.e. has the largest bias in the out-of-sample forecast for large τ_1^*. At the same time, the out-of-sample forecast exhibits

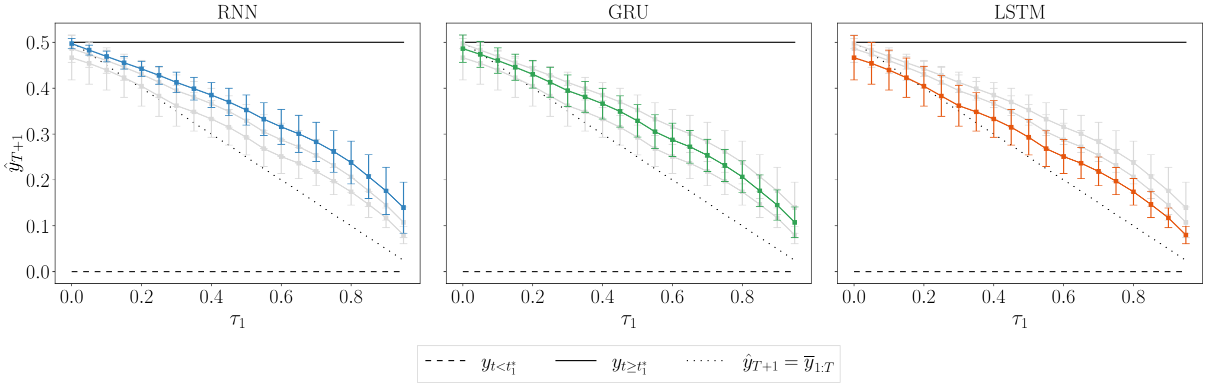

Figure 4.1: Illustration of the effect of the break location on the prediction of deep sequential models trained with MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for RNN (blue), GRU (green), and LSTM (orange) architectures. The realisations of the DGP before and after the break point are given as reference as well as the naive prediction $\hat{y}_t = \overline{y}_{1:T}$.

smaller variation, indicating that the use of LSTM cells provides more stability through controlling the information flow. The GRU model represents the middle ground between RNN and LSTM models in terms of out-of-sample forecast bias and variance, which can likely be attributed to a more (less) restrictive information flow compared to the RNN (LSTM) architecture.

4.3.3 Weighted Loss Functions

Having illustrated how mean shifts affect the forecast performance, we aim at identifying mitigation strategies for deep sequential models. We propose to move away from an equal weighting scheme and use a weighted loss function, for which the weight attached to the error decays at every time step. The final loss is then a linear combination of the weighted losses at every time step. For some loss function $\mathcal{L}(\cdot)$ and decay parameter $\gamma(t)$, we define the weighted loss as

$$l = \sum_{t=1}^{T} \gamma(t)\mathcal{L}(y_t, z_t). \tag{4.5}$$

The properties of this weighted loss function depend on the choices of $\mathcal{L}(\cdot)$ and $\gamma(t)$. While $\mathcal{L}(\cdot)$ controls how the prediction error magnitude is penalised, $\gamma(t)$ governs the rate, at which the influence of previous time steps decays. The linear combination in (4.5) ensures that the resulting gradient is scaled similarly to the loss function, implying that the contribution of the first time steps to the final loss is reduced.

Currently, the main interest in reducing the weight on early observations in the training data lies in improving the action anticipation in applications such as autonomous driving. Typically, the goal is to infer an action class before the action is fully observed (cf. Chan et al., 2017). The underlying idea is to create a weighted loss function encouraging an early action recognition by giving smaller penalties to classification errors based on partial video sequences, which consist of only few frames as opposed to the full sequence. Jain et al. (2016) introduced the idea of an exponentially weighted softmax loss, relying on weights of the form $\gamma(t) = e^{-(T-t)}$. Aliakbarian et al. (2017) proposed to combine this weighting scheme with a cross-entropy loss and compared it to the use of linear weights $\gamma(t) = \frac{t}{T}$. Considering an exponential decay, Tonutti et al. (2019) use a constant discount factor of 0.9 for an exponentially weighted cross-entropy loss, i.e. weights defined as $\gamma(t) = e^{-0.9(T-t)}$, but, thereby, they do not leverage the advantages of tuning this parameter. For large T, it is necessary to have control over the discount factor, to avoid pushing the weights to zero too early. A related approach was proposed by Bai et al. (2018), but focused on dealing with irregularly annotated temporal image sequences. Instead of choosing a constant discount factor, the weights depend on the distance to the last annotated image. Thus, there is no attempt to find a weighting scheme for the whole sequence of training data.

We now propose a range of weighting schemes for the weighted loss function defined in (4.5), which have different properties and may therefore be suitable for the deployment in a variety of mean shift scenarios.

Exponential Weights First, we consider an exponential decay. The decay parameter in (4.5) is then defined as

$$\gamma_{\text{RAY}}(t) = e^{-\alpha(T-t)},$$

where $\alpha \geq 0$. For $\alpha = 0$, we recover an equal weighting scheme. The discount factor α controls the strength of the weight reduction over time: a larger α discounts earlier prediction errors more strongly. Having control over the strength of the discounting extends the ideas of Aliakbarian et al. (2017), Jain et al. (2016), and Tonutti et al. (2019).

Rayleigh Weights Further, we consider weights based on the Rayleigh distribution. The Rayleigh model has previously been employed in epidemiology (Wallinga & Teunis, 2004) and general network diffusion applications (Ding et al., 2015; Gomez-Rodriguez et al., 2011) for its usability as a time-decaying kernel. For the loss in (4.5), we use Rayleigh weights expressed as

$$\gamma_{\text{RAY}}(t) = e^{-\frac{1}{2}\alpha(T-t)^2},$$

with $\alpha \geq 0$. For $\alpha = 0$, we retrieve a loss function with equal weights. Generally, the larger the value of α, the smaller is the weight on past time steps.

Bartlett Weights Next, we consider weights derived from the Bartlett kernel, which is popular in the literature concerning heteroskedasticity and autocorrelation consistent (HAC) covariance matrix estimation (Newey & West, 1987). This choice of weights represents a linear decay of previous time steps. Therefore, we extend the idea of a linear weighting scheme proposed in Aliakbarian et al. (2017) to time series data, but add an additional parameter: for the weighted loss definition in (4.5), the Bartlett weights are defined using

$$\gamma_{\text{BAR}}(t) = \begin{cases} 1 - \frac{T-t}{\alpha} & \text{for } \frac{T-t}{\alpha} \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.6}$$

where $0 < \alpha \leq T$ can be referred to as the lag truncation parameter, since lags of order $\frac{T-t}{\alpha} > 1$ are assigned zero weight. For $\alpha = T$, no truncation is taking place. This weighting scheme is different from the previously discussed exponential and Rayleigh weights, since we do not only control how strongly past observations are down-weighted, but how many early observations we choose to ignore entirely.

Parzen Weights Moreover, we examine weights based on the Parzen kernel, which has been considered in the HAC estimation literature (Gallant, 1987, p.533). For the weighted loss function in (4.5), we define the Parzen weights by

$$\gamma_{\text{PAR}}(t) = \begin{cases} 1 - 6\left(\frac{T-t}{\alpha}\right)^2 + 6\left(\frac{T-t}{\alpha}\right)^3 & \text{for } 0 \leq \frac{T-t}{\alpha} \leq \frac{1}{2}, \\ 2\left(1 - \frac{T-t}{\alpha}\right)^3 & \text{for } \frac{1}{2} \leq \frac{T-t}{\alpha} \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

where $0 < \alpha \leq T$ serves as a lag truncation parameter. For $\alpha = T$, the weights are not truncated. Again, this weighting scheme allows to control the strength of the discounting as well as how many early observations are ignored when calculating the loss.

Tukey-Hanning Weights Lastly, we derive weights from the Tukey-Hanning kernel. This kernel is frequently used in the HAC covariance estimation literature (Andrews, 1991) as well as in volatility estimation (Barndorff-Nielsen et al., 2008). For the weighted loss function in (4.5), we define the Tukey-Hanning weights by

$$\gamma_{\textsc{Tuk}}(t) = \begin{cases} \left(1 + \cos\left(\pi \frac{T-t}{\alpha}\right)\right)/2 & \text{for } \frac{T-t}{\alpha} \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

where $0 < \alpha \leq T$ serves once more as a lag truncation parameter. Choosing $\alpha = T$, results in no truncation. Thus, these weights regulate the strength of the discounting of past observations and also allow for control over how many early observations are discarded when calculating the loss.

Weight Comparison Table 4.1 summarises the considered weighting schemes for the weighted loss functions. Figure 4.2 provides an illustration of the weighting schemes with different choices of α for a sample size of $T = 1000$. The characteristics of the different weights are obvious: while α changes the functional form of the weight decay for the exponential and Rayleigh weights, it additionally controls how many early observations obtain a weight of exactly zero in case of Bartlett, Parzen and Tukey-Hanning weights.

Loss Function Choice All weighting schemes could obviously be applied in combination with any loss function $\mathcal{L}(\cdot)$. The choice of the error measure depends on the desired

Table 4.1: Considered weighting schemes for the weighted loss function defined in (4.5).

Weights	$\gamma(t)$	
Exponential (Exp)	$e^{-\alpha(T-t)}$	
Rayleigh (Ray)	$e^{-\frac{1}{2}\alpha(T-t)^2}$	
Bartlett (Bar)	$1 - \frac{T-t}{\alpha}$ $0,$	$\frac{T-t}{\alpha} \leq 1$ otherwise

Parzen (Par)	$1 - 6\left(\frac{T-t}{\alpha}\right)^2 + 6\left(\frac{T-t}{\alpha}\right)^3$
	$2\left(1 - \frac{T-t}{\alpha}\right)^3$
	0
		for $0 \leq \frac{T-t}{\alpha} \leq \frac{1}{2}$
		for $\frac{1}{2} \leq \frac{T-t}{\alpha} \leq 1$
		otherwise

Tukey-Hanning (Tuk)	$\left(1 + \cos\left(\pi \frac{T-t}{\alpha}\right)\right)/2$ 0	for $\frac{T-t}{\alpha} \leq 1$ otherwise

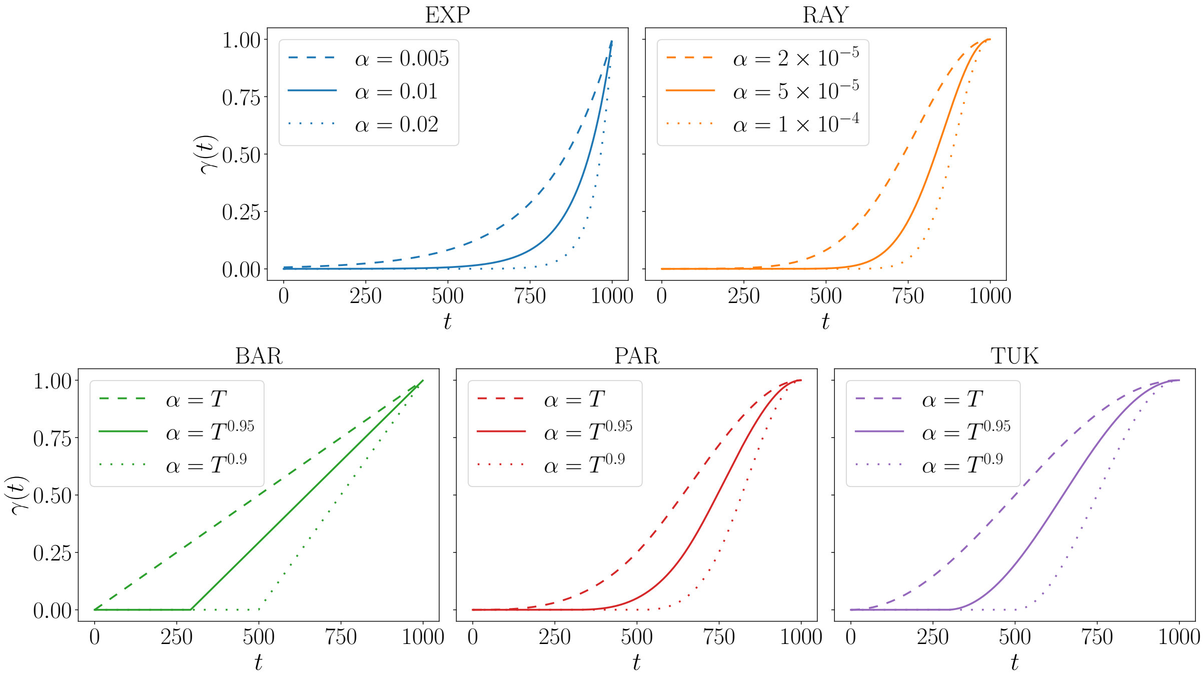

Figure 4.2: Comparison of exponential, Rayleigh, Bartlett, Parzen and Tukey-Hanning weighting schemes with different choices of the parameter α for a sample size of $T = 1000$.

properties of the loss function. In the following, we will focus on the squared prediction error, i.e. $\mathcal{L}(y_t, z_t) = \|y_t - z_t\|_2^2$. This is a natural choice if we want to evaluate and compare model performances using the mean squared forecast error (MSFE). By selecting the squared prediction error, we prioritise penalising large errors more than small errors.

Parameter Choice The favourable effect of using weighted loss functions in the presence of mean shifts in the training data will largely depend on the choice of the parameter α for a given weighting scheme. Finding an optimal α analytically is infeasible since the highly non-linear deep sequential models have no closed form solution. A simple yet effective way of performing parameter selection is to deploy a validation set approach: using the model forecasts on the validation set, we can compare the performance of different choices for α. We select that value for α to obtain predictions on the test data, which provides the smallest validation loss. Here, it is not desirable to discount the weight of past observations, so that we replace the weights used for training the deep sequential model with equal weights, i.e. $\gamma(t) = 1/T$, to calculate the validation loss. Thus, as validation loss, we consider the MSE loss. Assuming that the mean shift does not occur in the validation set, this approach is superior to other popular routes for hyperparameter tuning: a k-fold cross-validation for example would suffer from biased model forecast introduced not only by the presence of mean shifts in the data used for training but also in the held-out data.

4.3.3.1 Applying Weighted Loss Functions to a Toy Example

Using a simple DGP, we aim at demonstrating that the application of weighted loss functions can indeed improve the forecast performance of deep sequential models when facing mean shifts. Therefore, we consider the same setting and DGP as used in Figure 4.1, i.e. the GDP given in Equation (4.4). Figure 4.3 reports the mean prediction of an RNN which is trained using a weighted square loss based on exponential weights with $\alpha = 0.01$, Rayleigh weights with $\alpha = 5 \times 10^{-5}$ as well as Bartlett, Parzen, and Tukey-Hanning weights with $\alpha = T^{0.95}$ respectively. We observe that the forecasts based on the weighted square loss improve significantly over the predictions with the MSE loss. The bias-reduction is especially pronounced for the mid-range of considered break fractions τ_1. For the extreme choice of $\tau_1 = 0.95$, we can only observe a slight improvement, which appears sensible given a selection of parameters, that result in moderately discounting weights. Figures A.2 and A.1 in Appendix A report similar findings for GRU and LSTM architectures respectively. However, we can additionally note that the usage of a weighted loss function does increase the forecast variance for GRUs and LSTMs.

How does the choice of the weight parameter α influence these results? Figure 4.4

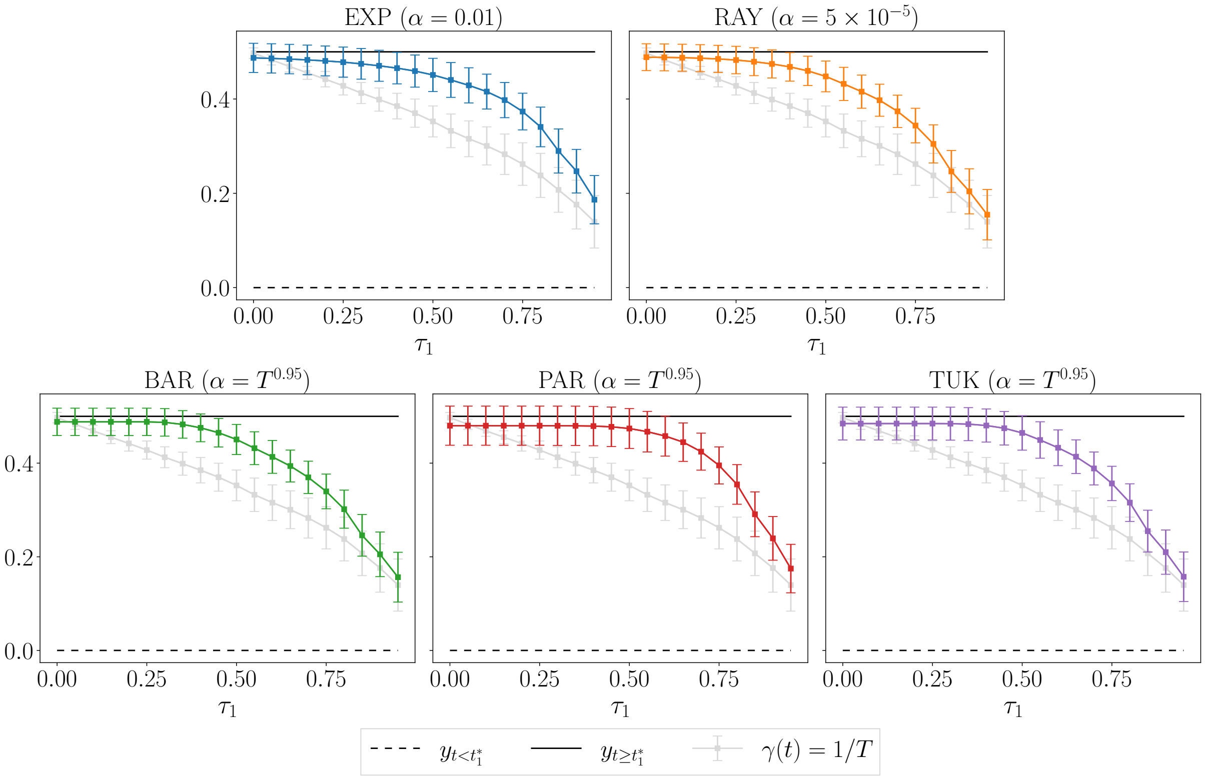

Figure 4.3: Illustration of the effect of training RNNs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$.

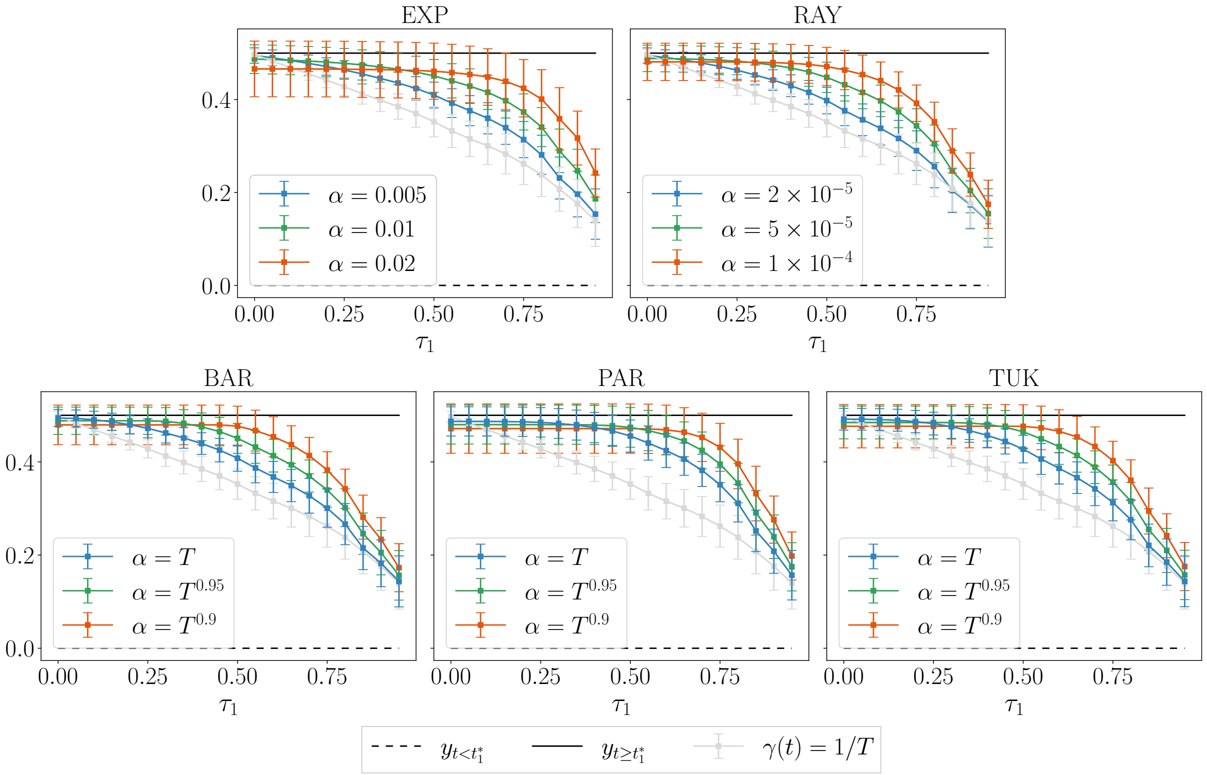

Figure 4.4: Illustration of the effect of varying the parameter α in the weighted loss when training RNNs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using MSE loss, for which $\gamma(t) = 1/T$.

compares different choices of α for the various weighting schemes for the RNN in this toy setting. For all weighting schemes, the choice of α has a decisive effect. The stronger the discounting of the weighting scheme, i.e. the larger α, the less biased is the RNN forecast for $\tau > 0.25$. This improvement comes at the expense of a larger forecast variance and an increased forecast bias for small break fractions. Similar results can be observed for GRU and LSTM architectures in the respective Figures A.4 and A.3 in Appendix A. These findings highlight the importance of a careful weight parameter choice depending on the data at hand using for instance the proposed validation set approach.

Furthermore, the results above leave room for an alternative approach in situations, in which the break point appears early in the sample. In that case, we might trade-off reducing the forecast variance by using more samples to capture the underlying dependency structure in favour of reducing the estimation bias by excluding pre-break observations.

4.3.4 Excluding Pre-Break Observations

As an alternative to the use of weighted loss functions, we consider estimating the break location and excluding the pre-break observations from the training set to reduce the estimation bias. As deep sequential models are generally data-hungry, this mitigation strategy may increase the forecast variance due to the reduced sample size and is thus only useful as long as we can retain enough post-break observations to still get a precise model estimate. Therefore, we aim at developing recommendations on when this strategy may be beneficial. In the following, we refer to this strategy as the PB strategy, which is short for *post-break* sample.

One option to estimate the location of multiple structural changes is the sequential testing procedure developed by Bai and Perron (1998, 2003). In a multiple linear regression framework, Bai and Perron (1998, 2003) estimate the break locations for a specified number of breaks by finding the data partition, which minimises the sum of squared residuals. To determine the correct number of breaks, they sequentially test the null hypothesis of l breaks against the alternative of $l+1$ breaks using a sup Wald-type statistic. Building on the estimated break locations for l and $l+1$ breaks, the test statistic is based on determining whether the $l+1$ break model can provide a significantly smaller minimal sum of squared residuals than the l break model. This process can be repeated by sequentially increasing l until the test fails to reject the null hypothesis of no additional breaks. For technical details, we refer the reader to Bai and Perron (1998, 2003). After identifying all break locations, all observations before the final break point can be discarded. The remaining observations are then used to train the deep sequential model of interest.

As of now, we are not aware of any work using this strategy to train deep sequential architectures. However, Lin and Zhang (2022) apply a related strategy to forecast the carbon price: the de-noised component of a wavelet transform as well as estimated break points and lagged observations are used as input for an LSTM network. Here, the estimated break points are used as exogenous input variables, rather than shortening the time series based on this information.

4.4 Simulation Study

In this section, we put the proposed mitigation strategies to the test. We investigate how weighted loss functions and the PB strategy perform under different DGPs and mean shift scenarios. The simulation study is implemented in Python and builds upon the `PyTorch` (Paszke et al., 2019), `scikit-learn` (Buitinck et al., 2013), and `statsmodels` (Seabold & Perktold, 2010) libraries. The code can be found at

https://github.com/johannnamr/
Forecasting-Facing-Structural-Breaks-Using-Deep-Sequential-Models.

4.4.1 Experimental Setup

Before presenting the relevant simulations results, we first discuss all considered experimental setups.

4.4.1.1 Data Generating Processes

Our experimental setup comprises two different situations: we assume that the DGP is represented by the mean shift model in (4.1), where the stationary stochastic process ϵ_t follows either (1) an autoregressive (AR) model (cf. Hamilton, 2020) or (2) an ARMA model (Box et al., 2016). In Setting 1, we particularly focus on the effect of different strengths of autocorrelation, while Setting 2 increases the complexity of the DGP by adding a moving average (MA) component. Thus, we analyse the effect of the proposed mitigation strategies in a simple yet informative simulation setup.

Setting 1 As DGP, we consider the following stationary AR process of order 1:

$$(1 - \phi L)\epsilon_t = \mu + u_t, \tag{4.9}$$

where L is the lag operator, $|\phi| < 1$, $\mu \in \mathbb{R}$, and u_t is standard normal white noise. We select an AR(1) process with $\mu = 0$ and $\phi \in \{0.1; 0.4; 0.7; -0.4\}$. This parameter choice covers different levels of persistence. We do not consider a situation close to non-stationarity such as $\phi = 0.99$, since we found the deep sequential model forecasts to break down in that case.

Setting 2 Here, we consider a stationary ARMA model of order $(1, 1)$ as DGP, which has the following form:

$$(1 - \phi L)\epsilon_t = \mu + (1 + \theta L)u_t, \tag{4.10}$$

where L is the lag operator, $|\phi| < 1$, $\theta \in \mathbb{R}$, $\mu \in \mathbb{R}$, and u_t is standard normal white noise. We select an ARMA$(1, 1)$ process with $\mu = 0$, $\phi \in \{0.4; -0.4\}$ and $\theta \in \{0.3; -0.3\}$. This parameter choice can illustrate the effect of an the additional MA term compared to Setting 1.

For every parameter combination, we simulate a sample of 1000 observations and hold back the most recent 10% of observations for the test set and the previous 5% for the validation set. Further, we repeat each experiment 500 times. A larger number of repetitions is infeasible due to the enormous run times of repeated optimisations for model training. Before introducing any breaks, the entire time series is scaled to values within the interval $[-1, 1]$ using maximum absolute scaling, i.e. $x_{scaled} = \frac{x}{\max(x)}$. This scaling induces no shifts, but allows for the values of error measures to be comparable across simulation settings, since the time series now have similar magnitudes.

4.4.1.2 Mean Shift Scenarios

For each DGP, our experimental setup considers two different mean shift scenarios: we assume (1) a single mean shift, and (2) multiple mean shifts. In Scenario 1, we reduce the challenge of forecasting facing mean shifts in the training sample to the simplest situation possible. Scenario 2 adds complexity in two different ways: an additional mean shift either increases the distortion further, or reverts to the original mean.

Scenario 1 In this scenario, we consider a single mean shift implying two regimes such that $K = 2$. For every simulation setting, we investigate combinations of break sizes $\beta = \beta_1 \in \{\text{sd}(\epsilon_t)/2,\ \text{sd}(\epsilon_t),\ 2 \cdot \text{sd}(\epsilon_t)\}$ and break locations $\tau^* = \tau_1^* \in \{0.2, 0.5, 0.8\}$. The choices of break locations correspond to three interesting situations: for $\tau_1^* = 0.2$, the mean shift occurs in the beginning of the training set, whereas for $\tau_1^* = 0.8$ the mean shifts towards the end of the training set. If $\tau_1^* = 0.5$, the mean shift occurs in the middle of the training sample.

Scenario 2 This scenario investigates the forecast performance of deep sequential models when facing multiple mean shifts. We consider the Settings 1 and 2 with two mean breaks, i.e. three regimes implying $K = 3$. We focus on the break fractions $\tau^* = (\tau_1^*, \tau_2^*) \in \{(0.2, 0.5), (0.5, 0.8)\}$ and the situation of an increasing break as well as of a mean reversion with break sizes $\beta_k \in \{\text{sd}(\epsilon_t)/2, \text{sd}(\epsilon_t), 2 \cdot \text{sd}(\epsilon_t)\}$. For an increasing break, we have $\beta_1 = \beta_2$ and for a mean reversion $\beta_1 = -\beta_2$.

4.4.1.3 Model Setups

Based on the simulated time series, we aim at generating one-step-ahead forecasts for the deep sequential architectures. For training the models, we use the Adam algorithm (Kingma & Ba, 2017) with learning rate 10^{-3}, weight decay 10^{-6}, and batch size 256 for optimisation. We stop the optimisation when the loss function drops by less than 10^{-5} or has been increasing for 100 steps, or when the optimisation procedure has reached 500 steps in total. As our final model, we select the deep sequential model, which was found to have the smallest loss on the validation set. When training the deep sequential models using weighted loss functions, we deploy the validation set approach described in Section 4.3.3 to choose the most suitable parameter α among three pre-determined options. We use the very same options for α illustrated in Figure 4.2, so that we consider $\alpha \in \{0.005, 0.01, 0.02\}$ for exponential weights, $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ for Rayleigh weights, and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$ for the kernel-based Bartlett, Parzen and Tukey-Hanning weights. Since we consider a univariate setting, we have $p = 1$ for all models. For all deep sequential models, we consider a single-layer architecture with $q = 10$ nodes in the hidden layer and train them using some weighted loss function as defined in (4.5), which we base on the squared prediction error. The reference for evaluating the proposed mitigation strategies is the MSE loss, for which $\gamma(t) = \frac{1}{T}$.

4.4.2 Weighted Loss Functions

Having discussed the experimental setup, we now explore the effect on the forecast quality of utilising weighted loss functions for training the deep sequential models. We experiment with different weights and choices of the parameter α resulting in various discounting schemes of past observations.

4.4.2.1 Setting 1 + Scenario 1

We begin by examining the forecast precision of the deep sequential models trained using a weighted loss in Setting 1 and Scenario 1, where data is simulated from an AR(1) process with different choices of ϕ and is subject to a single mean shift. Table 4.2 presents the ratio between the MSFE for the deep sequential models when trained using an exponentially weighted loss function (MSFE_{EXP}) and the MSFE in case of equal weights ($\text{MSFE}_{\text{EQUAL}}$). Results are reported for various break sizes β and locations τ^*. The simulation results for Rayleigh, Bartlett, Parzen, and Tukey-Hanning weighting schemes can be found in the respective Tables B.1, B.2, B.3, and B.4 in Appendix B.1.

For the exponential weighting scheme, we find that the benefits depend on the situation at hand. For all models, we observe that the MSFE increases compared to the MSE loss for small breaks ($\beta = \text{sd}(\epsilon_t)/2$) occurring early in the sample ($\tau^* = 0.2$). However, the larger the break and the later it appears, the larger is the improvement in terms of MSFE. Surprisingly, the benefits diminish with increasing autocorrelation, but are way larger for situations with negative autocorrelation, i.e. when comparing the cases $\phi = 0.4$ and $\phi = -0.4$. A reason for this is possibly that we investigate upward mean shifts ($\beta > 0$) implying that, for equal weights, the modelled negative autocorrelation is underestimating the true autocorrelation strength when facing a mean shift. Between the model architectures, only slight differences can be observed: across most parameter combinations, the RNN benefits slightly more from the exponentially weighted loss than the LSTM and GRU architectures. This is likely due to the more complex structure of LSTM and GRU models, which is constructed to retain more of the past information.

Comparing the simulation results between weighting schemes, we note that exponential and Rayleigh weights perform similarly well with slightly larger MSFE improvements for the exponential weighting scheme. For mean shifts appearing early or in the middle of the training set, i.e. $\tau^* = 0.2$ and $\tau^* = 0.5$, Bartlett, Parzen and Tukey-Hanning weights

Table 4.2: $\text{MSFE}_{\text{EXP}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$.

	β	$\text{sd}(\epsilon_t)/2$			$\text{sd}(\epsilon_t)$			$2 \cdot \text{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	1.0103	0.9918	0.9728	0.9973	0.9515	0.9201	0.9656	0.8992	0.8488
	0.4	0.9964	0.9736	0.9546	0.9828	0.9271	0.8854	0.9366	0.8604	0.8239
	0.7	1.0071	0.9937	0.9887	0.9994	0.9688	0.9393	0.9748	0.9133	0.8857
	-0.4	0.986	0.9137	0.8437	0.9255	0.7963	0.7602	0.8042	0.717	0.7407
LSTM	0.1	1.01	0.9929	0.9765	0.999	0.9574	0.9224	0.9642	0.914	0.8721
	0.4	1.0109	0.9884	0.9646	0.9924	0.932	0.8944	0.9551	0.8796	0.842
	0.7	1.0052	1.0009	0.9844	1.0005	0.9698	0.957	0.9836	0.9389	0.8974
	-0.4	0.9868	0.9117	0.8464	0.9301	0.8066	0.7633	0.8153	0.7347	0.8025
GRU	0.1	1.0067	0.9937	0.9752	0.9992	0.9522	0.9303	0.9646	0.9123	0.8835
	0.4	1.0074	0.9884	0.96	0.9907	0.9346	0.8899	0.9597	0.8762	0.833
	0.7	1.0093	1.0018	0.9875	0.9971	0.9741	0.9452	0.9804	0.9283	0.8897
	-0.4	0.9887	0.9137	0.8492	0.9282	0.8055	0.7673	0.8104	0.7275	0.7906

provide larger precision gains in terms of MSFE than exponential and Rayleigh weights, but the picture is reversed for breaks appearing late in the sample ($\tau^* = 0.8$). The differences in MSFE gains between these two groups of weights grow with an increasing amount of autocorrelation in the data. Further, this difference is largest for Bartlett weights.

These results can be explained by the shape of the weights as illustrated in Figure 4.2 in Section 4.3.3. Exponential and Rayleigh weights discount early observations more strongly than the kernel-inspired Bartlett, Parzen and Tukey-Hanning weights and therefore have benefits if the mean shift occurs later in the training sample. Compared to Parzen and Tukey-Hanning weighting schemes, the Bartlett scheme applies larger weights to earlier observations due to its linearity. Hence, Bartlett weights are best suited for situations where the break manifests itself early in the training set since it is desirable to not loose too much information from early data points, especially if the time series is highly correlated.

In general, we can conclude from these simulation results that the use of weighted loss functions is a suitable mitigation strategy for an autoregressive time series. The particular choice of the weighting scheme however, strongly depends on the present mean shift scenario.

4.4.2.2 Setting 1 + Scenario 2

Next, we examine the forecast precision of deep sequential models trained using a weighted loss in Setting 1 and Scenario 2, where data is simulated from an AR(1) process with different choices of ϕ and is subject to two either increasing or reverting mean shifts. Table 4.3 presents the ratio $\text{MSFE}_{\text{EXP}}/\text{MSFE}_{\text{EQUAL}}$ for various break sizes β and locations τ^* for an exponential weights. The results for Rayleigh, Bartlett, Parzen, and Tukey-Hanning weighting schemes can be found in the respective Tables B.5, B.6, B.7, and B.8 in Appendix B.2.

For an increasing break ($\beta_1 = \beta_2$), we note that the use a weighted loss with exponential weight decay is advantageous for all model architectures in all simulation settings. Generally, we find that the precision gain when using the exponential weighting scheme increases with increasing break size and decreasing number of observations after the final mean shift. In the moderate autocorrelation case, i.e. for $\phi = 0.4$, we find larger MSFE improvements than in the mild ($\phi = 0.1$) and strong ($\phi = 0.7$) autocorrelation cases. The most obvious forecast precision gains can be recorded in the negative autocorrelation case with $\phi = -0.4$ for all models, which is likely due to the considered positive mean shifts. Similar to Scenario 1, we precision benefits of the exponentially weighted loss tend to be larger for RNN than for LSTM or GRU models.

For a reverting break($\beta_1 = -\beta_2$), we find mixed results when using the exponential weighting scheme: while we observe benefits in the negative autocorrelation case across all settings and models, we only note more precise forecasts for medium ($\beta = \text{sd}(\epsilon_t)$) and large ($\beta = 2 \cdot \text{sd}(\epsilon_t)$) break sizes in mildly or moderately autocorrelated observations. For small break sizes, the bias induced by the data section, which is subject to a shifted mean, is not large enough to require the use of a weighted loss function. In the case of a highly correlated time series, the exponential weight decay discards too much information from

Table 4.3: $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	ϕ/τ^*	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.9779	0.9327	0.9181	0.8798	0.8738	0.8336
	0.4	0.9597	0.9093	0.8945	0.838	0.8345	0.8147
	0.7	0.9851	0.9637	0.9446	0.9049	0.8793	0.8355
	-0.4	0.8552	0.7747	0.7563	0.7039	0.6852	0.7327
LSTM	0.1	0.9783	0.9466	0.9279	0.8872	0.9178	0.8703
	0.4	0.9681	0.9134	0.9036	0.8473	0.8773	0.8437
	0.7	0.994	0.9615	0.9559	0.9104	0.9473	0.916
	-0.4	0.8532	0.7927	0.7653	0.737	0.7282	0.7828
GRU	0.1	0.9771	0.9397	0.9325	0.8634	0.9013	0.8721
	0.4	0.9673	0.9187	0.9085	0.8568	0.8682	0.8388
	0.7	0.9892	0.961	0.9573	0.9162	0.9277	0.8847
	-0.4	0.8541	0.7778	0.7652	0.7174	0.7225	0.7716
$\beta_1 = -\beta_2$							
RNN	0.1	1.0053	1.0099	0.9866	0.9889	0.9353	0.9396
	0.4	0.9915	0.999	0.9662	0.981	0.9066	0.9246
	0.7	1.0032	1.0101	0.9905	1.0051	0.9571	0.9739
	-0.4	0.9647	0.9784	0.8757	0.8945	0.7657	0.854
LSTM	0.1	1.0034	1.0086	0.9835	0.9866	0.9457	0.9502
	0.4	1.0036	1.0071	0.9758	0.9922	0.9252	0.9451
	0.7	1.0035	1.0146	0.997	1.0057	0.9661	0.9846
	-0.4	0.9688	0.9829	0.8732	0.9002	0.7838	0.8706
GRU	0.1	1.0061	1.0072	0.9835	0.9828	0.9452	0.9476
	0.4	1.0071	1.0039	0.9761	0.9902	0.925	0.9483
	0.7	1.0055	1.0075	0.9961	1.013	0.9647	0.9881
	-0.4	0.9693	0.9827	0.8732	0.9004	0.7781	0.8713

the observations prior to the first mean shift, so that for $\tau^* = (0.5, 0.8)$, the section with a shifted mean still highly affects the forecast precision. This diminishes the advantages of using an exponentially weighted loss function unless the break size is large.

When comparing the different weighting schemes, we note that exponential, Rayleigh and Parzen weights provide the largest MSFE improvements in the increasing break case. For $\tau^* = (0.2, 0.5)$, Rayleigh weights have advantages over Parzen and exponential weights, whereas for $\tau^* = (0.5, 0.8)$, an exponential weighting scheme outperforms Rayleigh and Parzen weights. These results can likely be explained by exponential, Rayleigh and Parzen weights dropping off more quickly than Bartlett and Tukey-Hanning weights, which appears to be advantageous if we have two mean shifts pointing into the same direction. Since the Parzen weighting scheme preserves slightly more information from earlier data points than exponential and Rayleigh weights, it is less well suited to situations where the final break occurs late in the training sample implying that the period with the deviating mean obtains too much weight during model training. In case of a reverting break, exponential and Rayleigh weights can yield larger precision gains than the Parzen weighting scheme, while all three are superior to Bartlett and Tukey-Hanning weights. The difference between weighted loss functions based on Parzen and exponential or Rayleigh weights can again be explained by Parzen weights keeping slighlty more information from earlier data points.

These simulation results indicate that a weighted loss function with a strong weight decay such as exponential, Rayleigh or Parzen weights is a suitable mitigation strategy when we have multiple mean shifts pointing into the same direction, i.e. they deviate further and further from the original mean. When considering mean shifts, which revert their direction, picture is less clear-cut. Here, the size of the mean shift appears to have the largest impact, such that the use of a weighted loss function is most beneficial for large break sizes.

4.4.2.3 Setting 2 + Scenario 1

We now shift to Setting 2, where the time series is simulated from an ARMA$(1, 1)$ process with different values for the parameters ϕ and θ and is subject to a single mean shift. Table 4.4 reports the ratio $\text{MSFE}_{\text{EXP}}/\text{MSFE}_{\text{EQUAL}}$ for various break sizes β and locations τ^* in case of an exponential weighting scheme. The simulation results for Rayleigh, Bartlett, Parzen, and Tukey-Hanning weighting schemes can be found in the respective Tables B.9, B.10, B.11, and B.12 in Appendix B.3.

For the exponential weighting scheme, we find general patterns which are similar to the Scenario 1 results when the DGP is an AR(1) process, i.e. Setting 1: the forecast precision gains increase with increasing break size and a later break location as well as for negative autocorrelation. However, the effectiveness of deploying an exponentially weighted loss function differs between Setting 1 and 2. For $\phi = 0.4$, adding an MA component, regardless of whether the MA parameter has a positive or negative sign,

Table 4.4: $\text{MSFE}_{\text{EXP}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$.

	β	$\text{sd}(\epsilon_t)/2$			$\text{sd}(\epsilon_t)$			$2 \cdot \text{sd}(\epsilon_t)$		
	$(\phi, \theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0074	0.9943	0.9846	0.9983	0.9639	0.9126	0.9746	0.874	0.8154
	(0.4,-0.3)	1.0106	0.9907	0.9644	0.9981	0.9426	0.9088	0.9658	0.8979	0.8637
	(-0.4,0.3)	0.999	0.976	0.9475	0.981	0.9241	0.8681	0.9327	0.8473	0.8121
	(-0.4,-0.3)	0.9736	0.8537	0.7209	0.8766	0.6476	0.5679	0.6322	0.5426	0.5147
LSTM	(0.4,0.3)	1.005	0.9942	0.9811	0.9955	0.9649	0.9095	0.972	0.8984	0.8235
	(0.4,-0.3)	1.0083	0.9893	0.9677	0.996	0.9507	0.9075	0.9649	0.9048	0.8662
	(-0.4,0.3)	1.0033	0.9781	0.9488	0.9828	0.931	0.8978	0.9483	0.878	0.8404
	(-0.4,-0.3)	0.9742	0.8502	0.7148	0.8856	0.6414	0.5844	0.6602	0.5583	0.5504
GRU	(0.4,0.3)	1.0035	0.9919	0.9786	0.994	0.9594	0.9159	0.9682	0.8903	0.8212
	(0.4,-0.3)	1.0079	0.9882	0.9692	0.9987	0.949	0.9159	0.964	0.8976	0.8648
	(-0.4,0.3)	1.001	0.977	0.9544	0.981	0.9274	0.8837	0.9369	0.8801	0.8278
	(-0.4,-0.3)	0.9733	0.8496	0.719	0.8816	0.6437	0.5786	0.6557	0.5503	0.5473

reduces the MSFE improvements obtained from using exponential weights compared to Setting 1. For $\phi = -0.4$, we note the same result for a positive MA component ($\theta = 0.3$), but in case of a negative MA coefficient ($\theta = -0.3$), we observe large forecast precision gains especially for large break sizes. The effectiveness reduction is likely due to the increased complexity of the DGP, which requires more information in order for the deep sequential models to produce adequate forecasts. If both AR and MA components have negative coefficients, the enlarged MSFE improvements can be traced back to the positive direction of the mean shift resulting in more severely biased forecasts when using an equally weighted loss. All of these observations are stable across the different model architectures.

Comparing the different considered weighting schemes, we find that Tukey-Hanning weights provide the largest MSFE gains for an early break ($\tau^* = 0.2$) followed by Bartlett weights. Both types of weights represent different types of decay (nonlinear and linear respectively), but the have in common that they perform the weakest discounting of past observations compared to the other considered weights (see Figure 4.2). This is beneficial here since the more complex dependence structure in the simulated time series requires more data to appropriately train the deep sequential models. For a middle break ($\tau^* = 0.5$), Rayleigh and Parzen weighting schemes yield the biggest forecast precision improvements indicating that they provide the best balance between down-weighting pre-break observations and retaining most post-break information. In case of a late break ($\tau^* = 0.8$), an exponentially weighted loss achieves the largest MSFE improvements as in this setup, most of the information from earlier observations is rendered irrelevant. From these results, it becomes clear that the choice of a suitable weighting scheme strongly depends on composition of the time series under investigation.

4.4.2.4 Setting 2 + Scenario 2

Lastly, we consider Setting 2 with a double mean shift. Table 4.5 records the ratio $\mathrm{MSFE}_{\mathrm{EXP}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ for various break sizes β and locations τ^* in case of an exponential weighting scheme. The results for Rayleigh, Bartlett, Parzen, and Tukey-Hanning weighting schemes can be found in the respective Tables B.13, B.14, B.15, and B.16 in Appendix B.4.

For an exponentially weighted loss, we again find similar patterns as for Scenario 2 when the DGP is represented by an AR(1) process (Setting 1): we observe no benefit in terms of MSFE improvement for an early final break and a small break size, while there are growing forecast precision gains for a late final break point and increasing break size as well as for the negative autocorrelation cases. Yet the effectiveness of utilising exponential weights is reduced by adding an MA component in both increasing and reverting break situations since the added complexity in the DGP demands for more data to generate precise forecasts with deep sequential models. An exception is the combination of a negative AR and MA coefficient where we observe much larger forecast precision improvements in the increasing and reverting break cases, again possibly reasoned by the positive mean shift direction. These simulation results appear to be stable across the different model architectures.

Table 4.5: $\text{MSFE}_{\text{EXP}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with exponential weight decay and $\alpha \in \{0.005, 0.01, 0.02\}$

	β	$\text{sd}(\epsilon_t)/2$		$\text{sd}(\epsilon_t)$		$2 \cdot \text{sd}(\epsilon_t)$	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_2 = \beta_2$							
RNN	(0.4,0.3)	0.9828	0.9387	0.9234	0.8618	0.8445	0.798
	(0.4,-0.3)	0.9712	0.9329	0.9206	0.8812	0.861	0.8252
	(-0.4,0.3)	0.9558	0.9002	0.8854	0.8274	0.8343	0.8167
	(-0.4,-0.3)	0.7529	0.605	0.5871	0.5323	0.532	0.6065
LSTM	(0.4,0.3)	0.9759	0.942	0.9369	0.8663	0.9142	0.8593
	(0.4,-0.3)	0.9768	0.9377	0.9311	0.8943	0.9145	0.8715
	(-0.4,0.3)	0.9594	0.9156	0.9062	0.8591	0.8917	0.8495
	(-0.4,-0.3)	0.7501	0.6101	0.5934	0.5788	0.5708	0.6341
GRU	(0.4,0.3)	0.9776	0.9392	0.9358	0.868	0.8881	0.8358
	(0.4,-0.3)	0.979	0.9343	0.9299	0.8717	0.9028	0.8623
	(-0.4,0.3)	0.9578	0.9101	0.9038	0.8465	0.8766	0.8481
	(-0.4,-0.3)	0.7503	0.6088	0.5873	0.5557	0.5524	0.6441
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	1.0034	1.0113	0.9899	0.9989	0.9449	0.9526
	(0.4,-0.3)	1.0029	1.0091	0.9784	0.9866	0.9315	0.9346
	(-0.4,0.3)	0.9948	0.9997	0.9627	0.9728	0.907	0.9212
	(-0.4,-0.3)	0.9404	0.9533	0.7824	0.8095	0.5923	0.7445
LSTM	(0.4,0.3)	0.9999	1.0068	0.9891	0.9971	0.9521	0.9631
	(0.4,-0.3)	1.0026	1.0051	0.9806	0.9836	0.9475	0.9443
	(-0.4,0.3)	0.9947	0.9997	0.968	0.9767	0.9252	0.9357
	(-0.4,-0.3)	0.9434	0.9543	0.79	0.8177	0.6138	0.7178
GRU	(0.4,0.3)	0.999	1.0096	0.9891	0.9997	0.9469	0.9623
	(0.4,-0.3)	1.0027	1.003	0.982	0.9871	0.9522	0.945
	(-0.4,0.3)	0.9937	0.9989	0.9633	0.9745	0.9118	0.9268
	(-0.4,-0.3)	0.9428	0.9544	0.7885	0.8162	0.6079	0.7261

When comparing the different weighting schemes, we note for both an increasing and a reverting break that if $\tau = (0.2, 0.5)$, i.e. in case of an early final break, Tukey-Hanning and exponential weights give the smallest benefits in terms of MSFE, while Rayleigh and Parzen provide the largest forecast precision improvements. In this situation, Tukey-Hanning weights include too much information from the observations prior to the final break, whereas the exponential weighting scheme does not keep track of a long enough information history by discounting the observations too strongly (see Figure 4.2). In contrast, Rayleigh and Parzen weights strike an appropriate balance in this situation. For $\tau = (0.5, 0.8)$, Bartlett and Tukey-Hanning weights provide the smallest MSFE gains since too much information prior to the final break point is conserved, while the weighting schemes with the strongest discounting of past observations, exponential and Rayleigh weights, achieve the largest forecast precision improvements. From these simulation results, we can once more conclude that there is not one best weight type, but the choice is very much dependent on the mean shift scenario at hand.

4.4.3 Excluding Pre-break Observations

We investigate whether excluding the pre-break observations from the training data can improve the forecast performance of deep sequential models. Since the data is simulated, the break points are known for all simulation setups. To purely analyse the effect of using a reduced dataset, we use this knowledge instead of applying the sequential testing procedure of Bai and Perron (1998, 2003), thereby excluding the uncertainty associated with estimating break points. Additionally, we do not distinguish between single and multiple mean shifts: we simply reduce the dataset to the observations after the most recent break location, regardless of how many mean shifts occurred before.

4.4.3.1 Setting 1

We begin by considering Setting 1, i.e. an AR(1) process with different values for ϕ. Table 4.6 reports the ratio between the MSFE using the PB strategy and the MSFE when the models are trained on the whole dataset using the MSE loss for various break points τ^* and break sizes β in Scenario 1.

We observe that the MSFE improves over the reference in all cases apart from the GRU architecture if $\phi = 0.4$ and we have a small and early occurring mean shift. The larger the break size β, and the later the break appears, the more beneficial it is to use the PB strategy. For an increasing autocorrelation, the reported MSFE ratio decreases and when comparing the $\phi = 0.4$ and the $\phi = -0.4$ cases, we find additional improvements for the negative autocorrelation setting. From these results, we conclude that the PB strategy is a useful way of dealing with in-sample structural breaks when training deep sequential models.

However, it is important to note that when investigating the MSFE_{PB} of RNN, LSTM and GRU models instead of the reported ratio (results not reported here), we find that the MSFE_{PB} decreases with stronger autocorrelation in the simulated time series and increases with a more recent break point. This indicates that the deep sequential architectures are

Table 4.6: $\text{MSFE}_{\text{PB}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 1 combined with Scenario 1.

	β	$\text{sd}(\epsilon_t)/2$			$\text{sd}(\epsilon_t)$			$2 \cdot \text{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	0.9967	0.9813	0.9634	0.9843	0.9407	0.9085	0.9527	0.8857	0.8378
	0.4	0.9945	0.9783	0.9576	0.9824	0.9314	0.882	0.936	0.8588	0.8156
	0.7	0.9951	0.9883	0.9909	0.9905	0.9592	0.9266	0.9692	0.899	0.8731
	-0.4	0.9807	0.9086	0.8369	0.9199	0.7885	0.7427	0.7977	0.709	0.6699
LSTM	0.1	0.9981	0.9822	0.966	0.9877	0.9468	0.9114	0.9546	0.9002	0.861
	0.4	0.9953	0.9817	0.9592	0.9823	0.9284	0.8805	0.9454	0.8684	0.8209
	0.7	0.9936	0.9898	0.9803	0.9882	0.9587	0.934	0.9694	0.9228	0.8783
	-0.4	0.9805	0.9049	0.8356	0.9231	0.7973	0.7435	0.808	0.7245	0.7246
GRU	0.1	0.9958	0.9821	0.9653	0.9877	0.9406	0.919	0.9535	0.8976	0.8733
	0.4	1.0001	0.9766	0.9528	0.9827	0.9247	0.8772	0.9497	0.8602	0.8125
	0.7	0.9979	0.9926	0.9843	0.9886	0.9644	0.9314	0.9692	0.9139	0.8742
	-0.4	0.9833	0.9071	0.8386	0.9219	0.7964	0.7477	0.803	0.7179	0.7131

more data-hungry than traditional time series models and it could be detrimental for the forecast performance to disregard large proportions of the training data entirely. The use of weighted loss functions could therefore be a promising alternative strategy in particular settings.

Table B.17 in Appendix B.5 reports the ratio $\text{MSFE}_{\text{PB}}/\text{MSFE}_{\text{EQUAL}}$ for Scenario 2. As expected, we observe similar results to Scenario 1 in case of an increasing double mean shift. For a reverting mean shift, we only find advantages if the break is large or if the final break occurs earlier in the sample. In this case, it is obvious that the success of the PB strategy strongly depends on how much data is left post-break.

Compared to the use of weighted loss functions as mitigating strategy for in-sample mean shifts, the PB strategy provides larger MSFE improvements in both Scenario 1 and 2 if the (final) mean shift occurs earlier in the sample. This is clearly related to the fact that deep sequential models need a relatively large dataset to generate precise forecasts. Deploying weighted loss functions caters to this requirement better than discarding considerable chunks of the time series entirely.

4.4.3.2 Setting 2

We now investigate how the PB strategy performs if additional complexity is added to the underlying DGP. We therefore present the simulation results for Setting 2, where the data is generated by an ARMA(1,1) process with different parameter values for the AR and MA components for Scenarios 1 (Table 4.7) and 2 (Table B.18 in Appendix B.5). Again, all results are presented as the ratio between MSFE_{PB} and the $\text{MSFE}_{\text{EQUAL}}$.

For Scenario 1, we find for all model architectures that the forecast precision improves in all cases apart from the $(\phi, \theta) = (0.4, 0.3)$ situation with a small and early occurring mean shift. Generally, we observe similar results for this scenario as in Setting 1, where the DGP is an AR(1) process: adding an MA component reduces the effectiveness of the PB strategy due to the added complexity demanding more data to train the deep

Table 4.7: $\text{MSFE}_{\text{PB}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 combined with Scenario 1.

	β	$\text{sd}(\epsilon_t)/2$			$\text{sd}(\epsilon_t)$			$2 \cdot \text{sd}(\epsilon_t)$		
	$(\phi, \theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0005	0.9876	0.9812	0.9915	0.9574	0.9094	0.968	0.8681	0.8126
	(0.4,-0.3)	0.9963	0.981	0.9558	0.984	0.9333	0.9008	0.9521	0.8891	0.856
	(-0.4,0.3)	0.9893	0.9658	0.9401	0.9714	0.9144	0.8613	0.9236	0.8384	0.8058
	(-0.4,-0.3)	0.9702	0.8485	0.712	0.8735	0.6436	0.5609	0.63	0.5393	0.5083
LSTM	(0.4,0.3)	1.0013	0.9895	0.9781	0.9918	0.9604	0.9067	0.9684	0.8942	0.8209
	(0.4,-0.3)	0.9981	0.9793	0.9588	0.9859	0.941	0.8992	0.9552	0.8956	0.8582
	(-0.4,0.3)	0.9936	0.9701	0.9399	0.9732	0.9234	0.8893	0.9391	0.8708	0.8325
	(-0.4,-0.3)	0.9703	0.8443	0.7062	0.8821	0.6369	0.5775	0.6575	0.5544	0.5439
GRU	(0.4,0.3)	1.0003	0.9871	0.9729	0.9908	0.9547	0.9105	0.9651	0.8859	0.8164
	(0.4,-0.3)	0.9962	0.9787	0.963	0.9871	0.9399	0.9101	0.9528	0.889	0.8593
	(-0.4,0.3)	0.9905	0.9688	0.9475	0.9708	0.9196	0.8774	0.9272	0.8727	0.8218
	(-0.4,-0.3)	0.9695	0.8443	0.7104	0.8781	0.6397	0.5717	0.6532	0.5469	0.5408

sequential models. Again, an exception is constituted by augmenting the negative autocorrelation case with a negative MA coefficient due to the positive direction of the mean shift. These observations similarly hold for RNN, LSTM and GRU architectures. Compared to deploying the mitigating strategy of weighted loss functions, the PB strategy provides more forecast precision gains across all simulation settings of Scenario 1.

For Scenario 2, we observe general result patterns for the PB strategy, which are similar to the ones in Setting 1. Across all model architectures, the additional MA component has the same effect as described for Scenario 1. Compared to the use of weighted loss functions, we once more find advantages in terms of larger MSFE improvements for the PB strategy if the final mean shift occurs early in the sample, while we detect disadvantages for late final breaks. Therefore, we can summarise that the usefulness of the PB strategy strongly depends on the location of the (final) mean shift and even more so if the complexity of the dependency structure in the time series of interest increases.

4.5 Application to the Voltage of the German Electric Power Grid

We now aim at forecasting real-world data, which are subject to mean shifts. This application is implemented in R and Python using the `strucchange` (Zeileis et al., 2002) and `PyTorch` (Paszke et al., 2019) libraries. The code can also be found at

https://github.com/johannnamr/Forecasting-Facing-Structural-Breaks-Using-Deep-Sequential-Models.

4.5.1 Data and Experimental Setup

We aim at forecasting the voltage of the electric power grid measured every second over the course of a day at Zinkmattenstraße in Freiburg, Germany. We consider four different dates and aggregate the data to measurements every minute by averaging in order to obtain four datasets with 1440 observations each. All data is provided by the Fraunhofer Institute for Solar Energy Systems via their online platform *Energy Charts* (Fraunhofer Institute for Solar Energy Systems, 2023).

For the dynamic operation of electric power grids, it is crucial to be able to accurately forecast parameters relevant to a smart grid such as power supply and demand as well as technical information on the stability of the system (Schäfer et al., 2015). The voltage of the power grid is one of the parameters providing indications for its stability. Fluctuations in the voltage occur due to local changes in power supply and demand, which may be large enough to shift the mean of the time series: Figure 4.5 illustrates the four time series and highlights the observations, at which the sequential procedure of Bai and Perron (1998, 2003) identifies a mean shift, and indicates the corresponding 95% confidence intervals. Table 4.8 summarises the results of the Bai and Perron (1998, 2003) procedure. We find that each of the four datasets contains four mean shifts at different points in time. Apart from the third break point in the datasets of the 18/09/2022 and 08/01/2023, all break point estimates exhibit narrow confidence intervals. Additional augmented Dickey-Fuller tests (Dickey & Fuller, 1979) on the resulting sections between the break points reject the null hypothesis of non-stationarity on a 10% level in most cases. Most importantly, the null is rejected for the final sections of all datasets. Therefore, the four voltage time series are well suited for the application of our developed mitigation strategies for the training of deep sequential models in the presence of mean shifts in the training data.

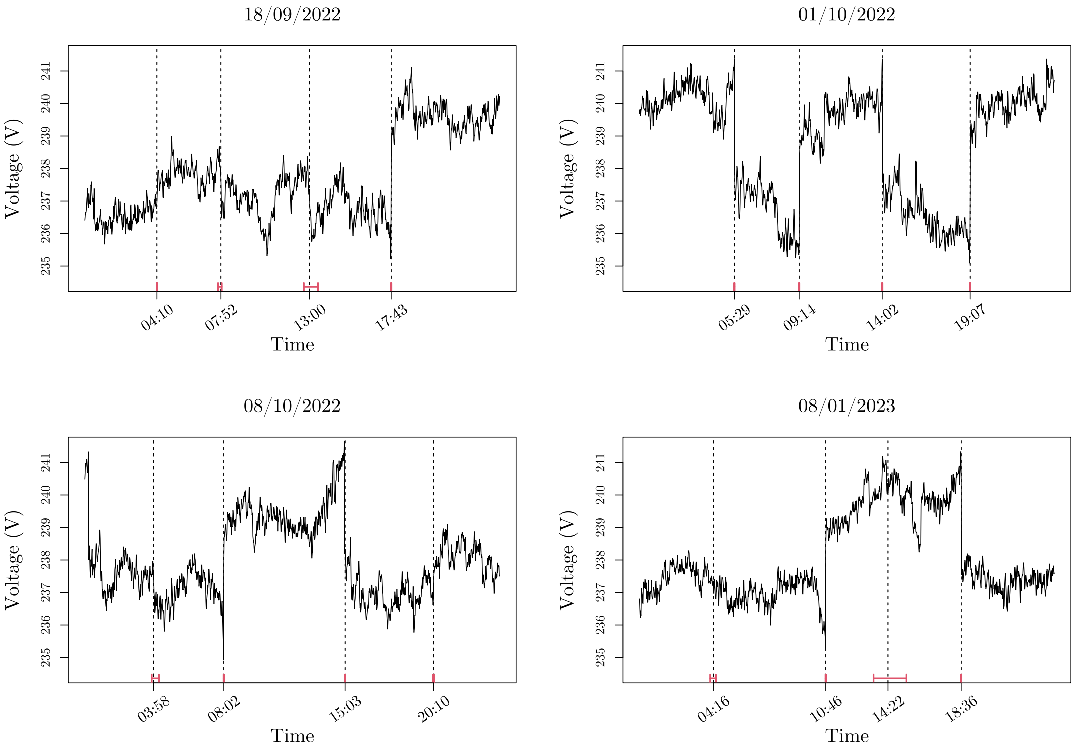

Figure 4.5: Voltage of the electric power grid measured every second in volt over the course of a day at Zinkmattenstraße in Freiburg, Germany. Data is aggregated to measurements every minute by averaging. The dashed vertical lines represent the estimated break points and the red horizontal bars the 95% confidence intervals.

Table 4.8: Break points and corresponding 95% confidence intervals estimated using the Bai and Perron (1998, 2003) procedure.

Data	Estimate	95%-CI
18/09/2022	04:10	04:09 , 04:11
	07:52	07:42 , 07:55
	13:00	12:40 , 13:29
	17:43	17:42 , 17:44
01/10/2022	05:29	05:28 , 05:30
	09:14	09:13 , 09:15
	14:02	14:01 , 14:03
	19:07	19:06 , 19:08

Data	Estimate	95%-CI
08/10/2022	03:58	03:52 , 04:17
	08:02	08:01 , 08:03
	15:03	15:02 , 15:04
	20:10	20:08 , 20:13
08/01/2023	04:16	04:05 , 04:25
	10:46	10:45 , 10:47
	14:22	13:32 , 15:26
	18:36	18:35 , 18:37

For the application, we split the voltage time series into training, validation, and test data: 90% of the data (1295 observations) are reserved for training, while the remaining 10% are split into $1/3$ validation (48 observations) and $2/3$ test data (96 observations). The final observation in the training set corresponds to a time stamp of 21:34, which is located beyond the final estimated break point for all time series. For model training, all time series are scaled to values within the interval $[0, 1]$ using min-max-scaling, i.e. $x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$. Further, we consider a univariate setting, i.e. $p = 1$, in which we generate one-step ahead forecasts from the input of previous observations.

For the optimisation, we choose the same parameters as in the simulations of Section 4.4: to train the deep sequential models, we use the Adam algorithm (Kingma & Ba, 2017) with learning rate 10^{-3}, weight decay 10^{-6}, and batch size 256 for optimisation. We stop the optimisation when the loss function drops by less than 10^{-5} or has been increasing for 100 steps, or when the optimisation procedure has reached 500 steps in total. The learned model is chosen to be the one with the smallest loss on the validation set. For all deep sequential models, we consider a single-layer architecture with $q = 10$ nodes in the hidden layer and train them using the MSE loss unless stated otherwise.

Since we are now working with real-life data instead of being in a well-controlled simulation setting, we make adjustments to our forecast evaluation methodology. This is to ensure that we provide the best possible voltage forecast given the class of deep sequential models: due to the non-convexity of the optimisation problem, we initialise with 100 different random seeds, resulting in 100 different trained models for each model architecture. While every single trained model represents a locally optimal model, the best trained model is close to a globally optimal model. We therefore report the *overall performance* of all 100 models, i.e. the distribution of their forecast metrics on the test set, as well as the *best performance* coming from the model with the most precise forecast on the validation set. When using weighted loss functions, we choose the most suitable value for the parameter α based on the validation MSE loss and repeat this process 100 times in order to evaluate best and overall performances. For exponential and Rayleigh weights, we use $\alpha \in \{0.005, 0.01, 0.002\}$ and $\alpha \in \{2\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}\}$ respectively, while for the kernel-based weights, i.e. Bartlett, Parzen and Tukey-Hanning weights, we choose $\alpha \in \{T, T^{0.95}, T^{0.9}\}$. To deploy the PB strategy, we cut off the training set at the final estimated break point for each dataset as reported in Table 4.8.

4.5.2 Results

4.5.2.1 Best Performance

Table 4.9 reports the best performance of all proposed mitigation strategies for all datasets and model architectures. Considering the dataset 18/09/2022, we find that all mitigation strategies improve over the equal weights training in terms of MSFE. An explanation can be derived from the illustration of this dataset in Figure 4.5: the first estimated mean shifts appear to be very small in size compared to the final break resulting in strongly biased weight estimates for models trained with a standard MSE loss.

Table 4.9: *Best performance* - $\text{MSFE}_i/\text{MSFE}_{\text{EQUAL}}$ reported for the model with the smallest validation loss of all 100 trained models.

		Weighted Loss					
Data	Model	Exp	Ray	Bar	Par	Tuk	PB
18/09/2022	RNN	0.9561	0.9602	0.9556	0.9555	0.9584	0.9505
	LSTM	0.9968	0.9793	0.9926	0.9907	0.9855	0.9696
	GRU	0.9747	0.9638	0.969	0.9736	0.9733	0.9551
01/10/2022	RNN	1.0	0.9721	0.9559	1.0086	0.9974	1.1744
	LSTM	1.0224	0.9992	1.0243	1.0198	1.0075	1.1839
	GRU	1.025	0.9825	1.0021	1.018	1.0095	1.1653
08/10/2022	RNN	1.0037	0.9533	0.914	0.9564	0.9153	1.0508
	LSTM	0.9364	0.9625	0.941	0.953	0.9533	1.3051
	GRU	0.9703	0.9522	0.9278	0.9982	0.9252	1.2049
08/01/2023	RNN	0.9002	0.7989	0.7811	0.7766	0.8461	0.7957
	LSTM	1.0162	0.9791	0.9172	0.9958	0.9921	0.9588
	GRU	0.9887	0.9261	0.8964	0.9939	0.9923	0.921

For dataset 01/10/2022, using Rayleigh weights slightly improves the MSFE over equal weights for all considered models. Exponential, Bartlett, Parzen, and Tukey-Hanning weighting schemes perform similarly to standard MSE loss training with MSFE records marginally below or above the reference. When using only post-break observations, the MSFE is up to 18% larger than when using equal weights. Referring to Figure 4.5, we observe that the mean in this dataset alternates between two largely different levels posing a setting which is very difficult to depict by any of the considered model architectures regardless of how the models were trained.

Focusing on dataset 08/10/2022, we find that the weighted loss strategy can provide MSFE improvements of up to 8.5% over the standard MSE loss (RNN architecture trained with Bartlett or Tukey-Hanning weights). However, the MSFE when using only post-break observations is reported as being up to 30.5% larger (LSTM architecture) than the equal weights reference. Examining Figure 4.5, we can conclude that down-weighting observations between the time stamps 06:02 and 15:03 exhibiting a much higher level than all others, confers an advantage to models trained with weighted loss functions. Further, there appear to be too few observations left after the final estimated break point at time stamp 20:10 to get precise weight estimates for models trained using the PB strategy.

For the last dataset, i.e. 08/01/2023, we observe that all mitigation strategies provide smaller MSFEs than the equal weights approach apart from the LSTM architecture trained with exponential weights. For RNNs, a MSFE reduction of over 20% is recorded (Rayleigh, Bartlett, and Parzen weighting schemes as well as PB strategy). Reflecting on the time series plot of the data in Figure 4.5, we note a similar situation to the previous dataset with the period between time stamps 10:46 and 18:36 being characterised by a much larger mean than the remaining observations.

Overall, we find that we are never worse off when choosing a weighted loss function over a standard MSE loss. Rayleigh weights emerge as being particularly recommendable since the MSFE is smaller compared to equal weights for all considered datasets and model architectures. Using only post-break observations turns out to be a viable strategy only if there is enough data left in the training set to learn the dependence structure.

4.5.2.2 Overall Performance

Figure 4.6 reports the overall performance for the RNN architecture. Overall datasets, the Rayleigh weighting scheme showcases the most consistent results of all mitigation strategies by having the smallest median root MSFE (RMSFE) of all considered model setups.

Considering dataset 18/09/2022 in particular, we find that all mitigation strategies outperform the standard MSE loss in the median. Approaches using a weighted loss, show an elevated variation in their RMSFE compared to equal weights, but even their largest RMSFE is still smaller than the smallest RMSFE recorded for the standard MSE loss training. When using only post-break observations, we note a large upward variation in the RMSFE.

For the dataset 01/10/2022, only exponential and Rayleigh weighting schemes can, on average, outperform a training using equal weights. The RMSFE of the PB strategy is always larger than for a standard MSE loss, which can certainly be explained by a training set, which is too short for the model to learn the dependence structure in the data.

When investigating dataset 08/10/2022, all weighted loss approaches apart from Tukey-Hanning weights report smaller RMSFE in mean and median than the standard MSE

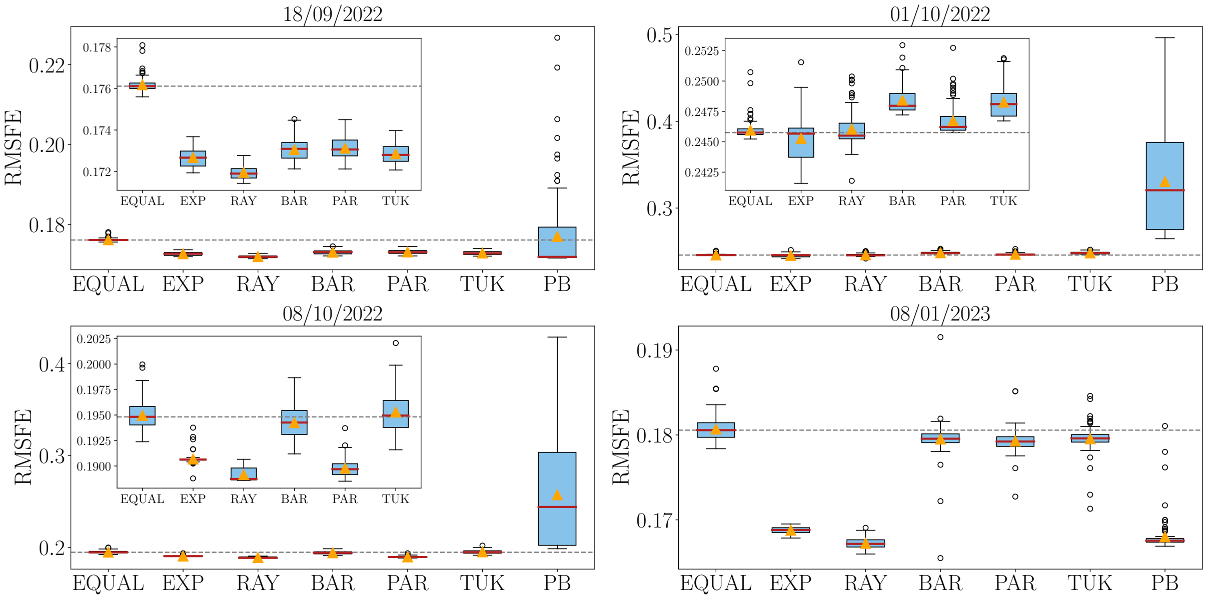

Figure 4.6: *Overall performance* - distribution of the RMSE on the test set for 100 trained RNN architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights.

loss. For exponential, Rayleigh and Parzen weighting schemes, all recorded RMSFEs are smaller than the average MSFE for equal weights. Training the RNN using only post-break observations leads to a larger RMSFE on average than when using the full dataset and an increased variation. Again, this is likely a result of a training data set, which is too small for the model to learn the relevant dependencies.

For the final dataset, i.e. 08/01/2023, we find that all mitigation strategies have a smaller RMSFE in mean and median than using the standard MSE loss. Notably, the RMSFE is particularly small for exponential and Rayleigh weighting schemes as well as the PB strategy.

Appendix C reports the overall performance for LSTM and GRU architectures in Figures C.1 and C.2 respectively. Generally, the observations coincide with the RNN results. The only exception occurs for the 01/10/2022 dataset when training LSTM models. Here, the standard MSE loss outperforms all mitigation strategies.

In summary, we note that in almost all settings it is sensible to consider one of the proposed mitigation strategies when the training data is found to contain mean shifts. Especially, exponential and Rayleigh weighting schemes can be recommended to forecast the voltage in the German electric power grid. Whether using only post-break observations is a viable option heavily depends on how much data remains after the final estimated break point in relation to how difficult it is to learn the underlying dependence structure.

4.6 Conclusion

In this work, we investigated the forecasting performance of deep sequential models in the presence of mean shifts in the training data. First, we reviewed RNN, LSTM and GRU architectures. Then, we presented the considered mean shift model, and discussed how the forecasts of deep sequential models are affected by mean shifts. Further, we proposed different strategies for fostering the forecast robustness of deep sequential models when facing mean shifts such as the deployment of weighted loss functions or estimating the break points and excluding pre-break observations from the training set. In an extensive simulation study, we evaluated the proposed mitigation strategies. Finally, this comparison was extended to a real-world application, in which we predicted the voltage of the German electric power grid.

We found that different mitigation strategies work better under different conditions: if there is enough data left after the final estimated break point such that the underlying DGP can be successfully characterised, discarding all previous observations in the training set can produce good forecasts. If the the final estimated mean shift occurs later in the training sample, however, the use of weighted loss functions to train the deep sequential models can be an appropriate mitigating strategy. The magnitude of the forecast precision improvements mostly depends on the size of the mean shift. Further, the choice of particular weights revolves around the particular structure of the time series of interest and is therefore difficult to generalise. In the application, we observed that, on average, we were never worse off when deploying weighted losses with exponential or Rayleigh weight decay than when using no mitigation strategy at all. Additionally, training the deep sequential architectures using these weighting schemes provided the possibility of large improvements in the forecast precision.

In our analyses, we only considered a fixed sequence of weight decay parameters for the weighted loss functions. A natural next step would be to develop suitable tuning strategies for selecting the most beneficial parameter value for a given time series. Furthermore, deep sequential architectures naturally lend themselves to multivariate settings, so that the proposed mitigation strategies could be investigated for multivariate forecasts. Having examined the impact of mean shifts in the training data, further investigations are required in order to analyse and possibly mitigate the effect of mean shifts in the validation or test set.

Appendix

Appendix A Additional Results: Models

This section reports additional results for section 4.2.

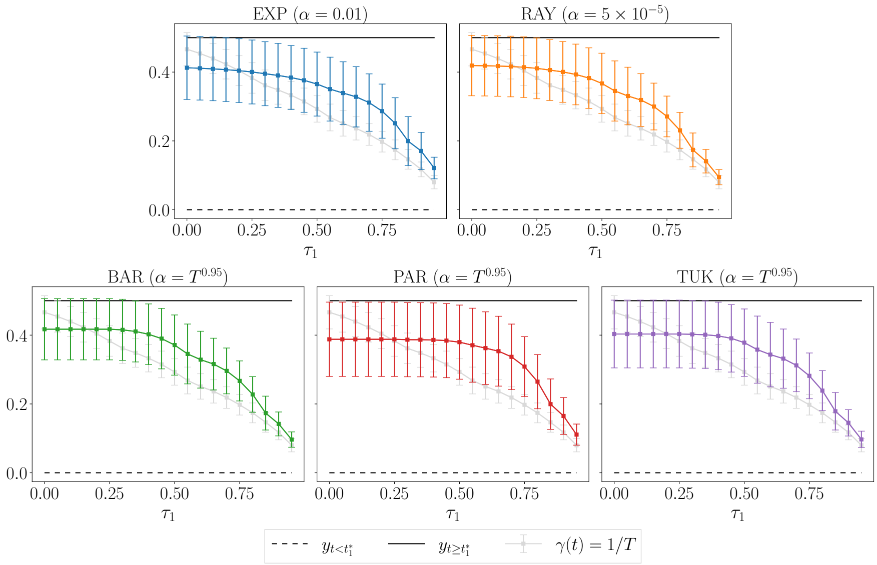

Figure A.1: Illustration of the effect of training LSTMs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$.

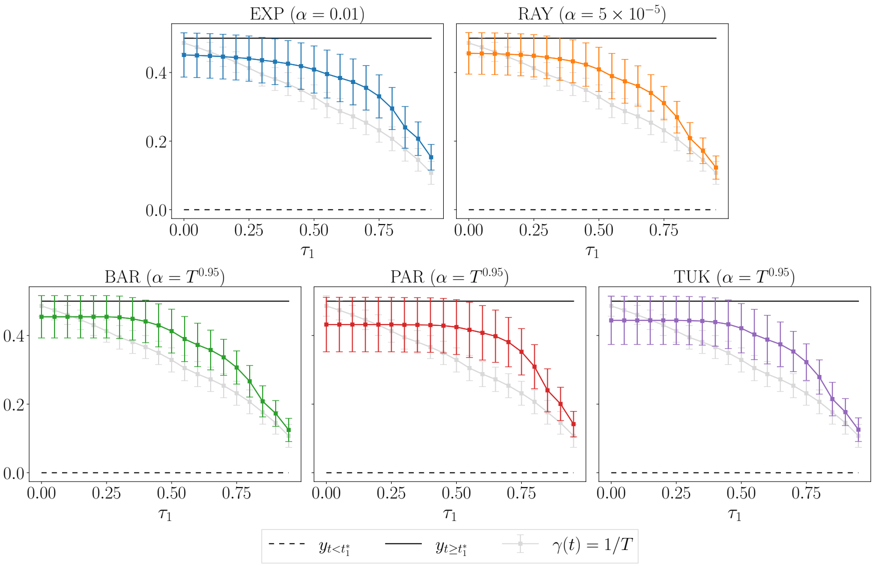

Figure A.2: Illustration of the effect of training GRUs with a weighted square loss compared to a standard MSE loss. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$.

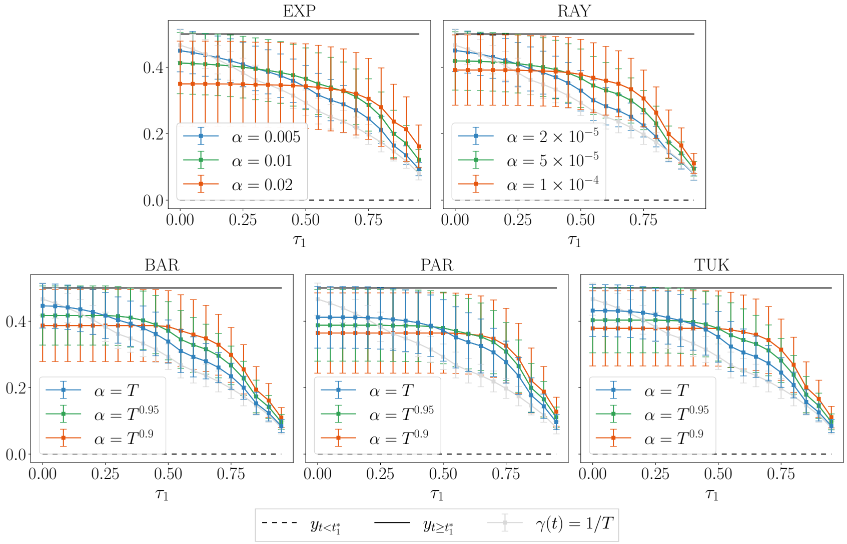

Figure A.3: Illustration of the effect of varying the parameter α in the weighted loss when training LSTMs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$.

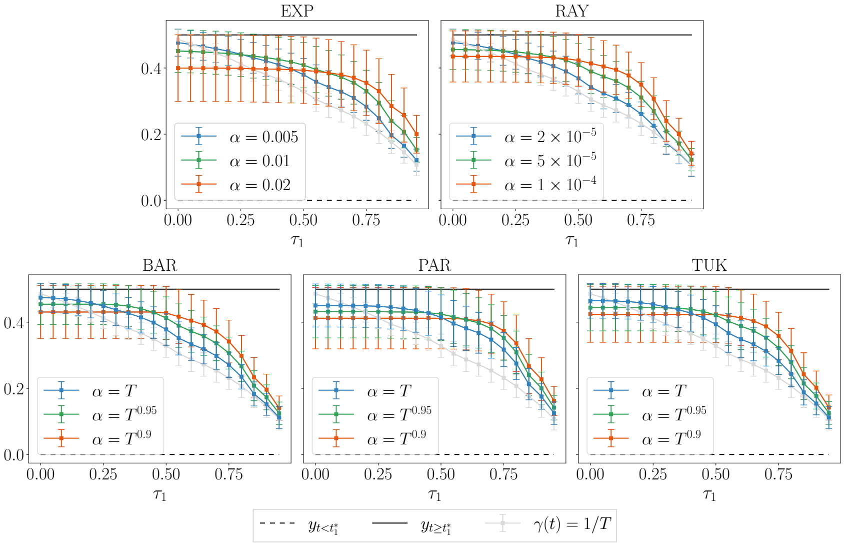

Figure A.4: Illustration of the effect of varying the parameter α in the weighted loss when training GRUs. The graphs depict the (constant) mean prediction over 1000 repetitions and its standard deviation as error bars for all weighting schemes with different values for α. The realisations of the DGP before and after the break point are given as reference as well as the prediction using an MSE loss, for which $\gamma(t) = 1/T$.

Appendix B Additional Results: Simulation Study

This section reports additional results for section 4.4.

B.1 Weighted Loss Functions: Setting 1 + Scenario 1

Table B.1: $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	1.0099	0.9924	0.9734	0.9982	0.9503	0.9213	0.9657	0.8973	0.8586
	0.4	0.9932	0.9759	0.9548	0.982	0.9252	0.8916	0.936	0.8574	0.8513
	0.7	1.0021	0.9941	0.9897	0.9971	0.9671	0.9375	0.9762	0.9068	0.901
	-0.4	0.9841	0.9121	0.8558	0.9244	0.7926	0.8098	0.8015	0.713	0.7872
LSTM	0.1	1.0082	0.9936	0.9754	0.9986	0.9548	0.9225	0.9655	0.9116	0.8796
	0.4	1.0109	0.9878	0.9689	0.9942	0.9322	0.9037	0.9561	0.8759	0.8712
	0.7	1.0024	0.9926	0.9821	1.0012	0.966	0.9502	0.9803	0.9358	0.905
	-0.4	0.9856	0.9089	0.8575	0.9272	0.8014	0.8186	0.8126	0.7309	0.861
GRU	0.1	1.0063	0.9929	0.9755	0.9998	0.9498	0.9314	0.9639	0.9096	0.8883
	0.4	1.0081	0.9864	0.9634	0.9909	0.9361	0.8985	0.9592	0.875	0.8597
	0.7	1.0049	0.9922	0.99	0.9948	0.9672	0.9481	0.9805	0.9206	0.9006
	-0.4	0.9887	0.9112	0.8602	0.9256	0.8011	0.8248	0.8071	0.7226	0.8455

Table B.2: $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	1.0082	0.9925	0.981	0.9963	0.9527	0.9306	0.9678	0.9004	0.8617
	0.4	0.9926	0.9725	0.9742	0.9798	0.9318	0.9389	0.9344	0.8622	0.9053
	0.7	0.9964	0.9929	1.0032	0.9914	0.9669	0.9729	0.9773	0.9139	0.9839
	-0.4	0.9841	0.9136	0.9268	0.9237	0.7939	0.8733	0.8012	0.7156	0.7981
LSTM	0.1	1.0081	0.9921	0.9787	0.9971	0.9585	0.9295	0.9669	0.9129	0.8809
	0.4	1.0072	0.9859	0.9902	0.9918	0.9354	0.9581	0.9542	0.8846	0.9346
	0.7	0.9962	0.9973	0.9899	0.9942	0.9719	0.98	0.9803	0.9436	0.9548
	-0.4	0.9825	0.9107	0.9373	0.9253	0.8019	0.8823	0.813	0.7311	0.8678
GRU	0.1	1.0052	0.9919	0.9798	0.9991	0.9529	0.939	0.9663	0.9113	0.8917
	0.4	1.0084	0.9863	0.9823	0.9882	0.9391	0.9482	0.9587	0.8817	0.9117
	0.7	1.0034	0.9954	1.003	0.9967	0.9753	0.9778	0.977	0.9356	0.9762
	-0.4	0.9856	0.9135	0.9379	0.9243	0.8019	0.8879	0.8062	0.7229	0.852

Table B.3: $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	1.0093	0.991	0.9761	0.9966	0.95	0.9283	0.9644	0.8983	0.8629
	0.4	0.9948	0.9716	0.9627	0.9826	0.9272	0.912	0.9347	0.8587	0.8885
	0.7	1.002	0.9925	0.9924	0.9952	0.9656	0.9486	0.9713	0.9101	0.9332
	-0.4	0.9862	0.9115	0.8849	0.9249	0.7927	0.8592	0.8015	0.7129	0.7935
LSTM	0.1	1.0087	0.9914	0.9773	0.9978	0.9554	0.929	0.964	0.9127	0.8833
	0.4	1.0116	0.9857	0.9767	0.9937	0.9315	0.9257	0.9544	0.8761	0.9133
	0.7	0.9992	0.9935	0.9846	0.9931	0.9675	0.9607	0.977	0.9361	0.9263
	-0.4	0.9838	0.9091	0.8885	0.9268	0.802	0.8778	0.8109	0.7294	0.8659
GRU	0.1	1.0062	0.9927	0.9769	0.9988	0.9508	0.9334	0.9638	0.9099	0.8939
	0.4	1.0074	0.99	0.9691	0.9951	0.9355	0.9211	0.9572	0.8776	0.8955
	0.7	1.0057	0.9944	0.9897	0.9926	0.9668	0.9628	0.977	0.9226	0.9326
	-0.4	0.987	0.9118	0.8914	0.9247	0.8013	0.8768	0.8059	0.7218	0.858

Table B.4: $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 1 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	ϕ/τ^*	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	0.1	1.0096	0.9921	0.9777	0.9967	0.9529	0.9312	0.9676	0.898	0.8659
	0.4	0.9919	0.9726	0.972	0.9826	0.9258	0.9361	0.9335	0.8595	0.904
	0.7	0.9975	0.9973	0.9966	0.9942	0.9669	0.9694	0.971	0.9143	0.9735
	-0.4	0.9841	0.9132	0.9162	0.9225	0.7931	0.8755	0.8017	0.712	0.8
LSTM	0.1	1.0086	0.9909	0.9804	0.999	0.9577	0.9276	0.9644	0.9129	0.879
	0.4	1.0089	0.9869	0.9878	0.9902	0.936	0.9526	0.9563	0.8813	0.9303
	0.7	1.0013	0.9974	0.9931	0.9915	0.9693	0.9791	0.9783	0.9437	0.9575
	-0.4	0.9834	0.9094	0.9255	0.925	0.8011	0.886	0.8122	0.7284	0.867
GRU	0.1	1.0052	0.9913	0.9782	0.9984	0.9526	0.9368	0.9644	0.9107	0.8924
	0.4	1.0063	0.9891	0.9773	0.9895	0.9388	0.9389	0.9562	0.8763	0.9092
	0.7	1.0026	0.9976	0.9985	0.9922	0.9724	0.9771	0.9701	0.931	0.9571
	-0.4	0.9857	0.9123	0.9246	0.9246	0.8006	0.8887	0.806	0.7205	0.85

B.2 Weighted Loss Functions: Setting 1 + Scenario 2

Table B.5: $\mathrm{MSFE}_{\mathrm{RAY}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	ϕ/τ^*	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.977	0.9335	0.9165	0.8834	0.8711	0.8421
	0.4	0.9589	0.9125	0.8919	0.8459	0.8288	0.838
	0.7	0.9797	0.9578	0.9382	0.9023	0.8715	0.8464
	-0.4	0.8528	0.7832	0.7541	0.7509	0.6811	0.7782
LSTM	0.1	0.9776	0.9449	0.9251	0.8912	0.9148	0.877
	0.4	0.9684	0.9132	0.9027	0.8538	0.8682	0.8703
	0.7	0.9873	0.9661	0.9523	0.914	0.938	0.9197
	-0.4	0.8496	0.8032	0.7604	0.7903	0.7229	0.838
GRU	0.1	0.9768	0.9385	0.9295	0.8657	0.8999	0.8804
	0.4	0.967	0.9226	0.909	0.8626	0.862	0.8697
	0.7	0.9826	0.9593	0.9539	0.916	0.9184	0.8871
	-0.4	0.8511	0.7864	0.7606	0.768	0.7187	0.8274
$\beta_1 = -\beta_2$							
RNN	0.1	1.0061	1.013	0.9866	0.9929	0.9347	0.9505
	0.4	0.9942	1.0003	0.9664	0.9889	0.9058	0.9556
	0.7	1.0011	1.0073	0.9875	1.0095	0.9479	0.9849
	-0.4	0.9633	0.9899	0.8726	0.9534	0.762	0.9051
LSTM	0.1	1.0031	1.0078	0.984	0.9879	0.9427	0.9596
	0.4	1.0044	1.0116	0.9758	1.0021	0.9223	0.9777
	0.7	0.9985	1.0087	0.9924	1.0097	0.9619	0.9897
	-0.4	0.9655	0.9968	0.8695	0.9626	0.7796	0.9322
GRU	0.1	1.0047	1.0082	0.9833	0.9867	0.9433	0.9535
	0.4	1.0063	1.01	0.9729	1.0017	0.924	0.9781
	0.7	1.0103	1.0053	0.9897	1.0159	0.9588	0.9936
	-0.4	0.9667	0.9952	0.8677	0.9634	0.7735	0.9278

Table B.6: $\text{MSFE}_{\text{BAR}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\text{sd}(\epsilon_t)/2$		$\text{sd}(\epsilon_t)$		$2 \cdot \text{sd}(\epsilon_t)$	
	ϕ/τ^*	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.9778	0.9402	0.9176	0.897	0.8732	0.8495
	0.4	0.9594	0.9321	0.897	0.8883	0.8344	0.8939
	0.7	0.9865	0.9757	0.9496	0.9342	0.8831	0.9277
	-0.4	0.854	0.8572	0.7545	0.8147	0.684	0.7902
LSTM	0.1	0.9779	0.9485	0.9272	0.8976	0.9144	0.8757
	0.4	0.969	0.9356	0.9079	0.9078	0.8772	0.9384
	0.7	0.9927	0.9775	0.9652	0.9379	0.9495	0.978
	-0.4	0.8513	0.8868	0.7604	0.8543	0.7233	0.8464
GRU	0.1	0.9769	0.9424	0.9324	0.8729	0.8997	0.8791
	0.4	0.9682	0.9441	0.9106	0.9138	0.8678	0.9195
	0.7	0.983	0.9692	0.96	0.9423	0.9282	0.9504
	-0.4	0.8525	0.8635	0.7606	0.8364	0.7185	0.8335
$\beta_1 = -\beta_2$							
RNN	0.1	1.0053	1.014	0.9889	1.0012	0.9361	0.9518
	0.4	0.9915	1.01	0.9696	1.0272	0.9105	1.009
	0.7	1.0032	1.0065	0.9918	1.028	0.9583	1.0634
	-0.4	0.9648	1.0453	0.8737	1.0125	0.7642	0.9129
LSTM	0.1	1.0018	1.0113	0.9853	0.9938	0.9458	0.9589
	0.4	0.9996	1.0212	0.9774	1.0507	0.9287	1.045
	0.7	0.9993	1.0077	0.9972	1.0203	0.9719	1.0379
	-0.4	0.9674	1.0567	0.8698	1.0344	0.7802	0.9372
GRU	0.1	1.005	1.0088	0.9837	0.9919	0.945	0.9553
	0.4	1.002	1.0227	0.9792	1.0424	0.9327	1.0367
	0.7	1.001	1.0097	0.9931	1.0286	0.9691	1.0463
	-0.4	0.9678	1.051	0.8692	1.0281	0.773	0.9368

Table B.7: $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	ϕ/τ^*	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.9761	0.9365	0.917	0.8877	0.8719	0.8464
	0.4	0.9565	0.9207	0.8934	0.8626	0.8314	0.872
	0.7	0.9815	0.9673	0.9438	0.9135	0.8745	0.8809
	-0.4	0.8527	0.8107	0.754	0.7968	0.6814	0.7894
LSTM	0.1	0.9768	0.9471	0.9257	0.8944	0.9147	0.8794
	0.4	0.9677	0.9195	0.9031	0.8742	0.8734	0.9163
	0.7	0.9882	0.9705	0.954	0.9265	0.9415	0.9426
	-0.4	0.8498	0.8339	0.7607	0.8517	0.7224	0.8495
GRU	0.1	0.9758	0.9408	0.9302	0.8698	0.8995	0.8807
	0.4	0.9702	0.9253	0.9099	0.885	0.8647	0.9007
	0.7	0.9833	0.9628	0.9549	0.9235	0.9208	0.9271
	-0.4	0.8509	0.8183	0.7607	0.8247	0.7169	0.8336
$\beta_1 = -\beta_2$							
RNN	0.1	1.0063	1.0137	0.987	0.9976	0.9348	0.9523
	0.4	0.9902	1.0071	0.9683	1.0138	0.9061	0.9942
	0.7	0.9997	1.0123	0.9914	1.0159	0.9507	1.0294
	-0.4	0.9636	1.0251	0.8724	1.0119	0.7618	0.9154
LSTM	0.1	1.0025	1.0093	0.9836	0.9933	0.9442	0.9614
	0.4	1.0009	1.0179	0.9729	1.0286	0.923	1.021
	0.7	1.0021	1.0149	0.9916	1.0235	0.9648	1.0238
	-0.4	0.9651	1.0331	0.8695	1.0332	0.7782	0.9412
GRU	0.1	1.0042	1.007	0.9819	0.9917	0.9438	0.9549
	0.4	1.0037	1.0143	0.9765	1.0254	0.93	1.0205
	0.7	1.0065	1.0049	0.9927	1.0247	0.9641	1.0288
	-0.4	0.9663	1.0286	0.869	1.0261	0.7718	0.9323

Table B.8: $\text{MSFE}_{\text{TUK}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 1 and Scenario 2 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\text{sd}(\epsilon_t)/2$		$\text{sd}(\epsilon_t)$		$2 \cdot \text{sd}(\epsilon_t)$	
	ϕ/τ^*	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.9777	0.9391	0.9174	0.8924	0.872	0.846
	0.4	0.9604	0.9297	0.8947	0.8854	0.8316	0.8854
	0.7	0.9825	0.9729	0.9456	0.9286	0.8825	0.9162
	-0.4	0.8544	0.8432	0.7536	0.8139	0.68	0.7908
LSTM	0.1	0.9772	0.9489	0.9283	0.896	0.915	0.8751
	0.4	0.9673	0.9333	0.9052	0.8958	0.8746	0.9341
	0.7	0.9889	0.9733	0.9573	0.9339	0.9498	0.9675
	-0.4	0.8502	0.8677	0.7595	0.8526	0.7208	0.8435
GRU	0.1	0.9765	0.9435	0.9331	0.8716	0.8998	0.8794
	0.4	0.9682	0.9392	0.913	0.9044	0.8661	0.9208
	0.7	0.9824	0.9699	0.9564	0.9328	0.9303	0.9387
	-0.4	0.8518	0.8518	0.7602	0.8317	0.7153	0.84
$\beta_1 = -\beta_2$							
RNN	0.1	1.0048	1.0159	0.9885	1.0002	0.9357	0.9544
	0.4	0.9891	1.0109	0.9679	1.0326	0.9068	1.0173
	0.7	0.9985	1.0121	0.9893	1.0272	0.9555	1.0651
	-0.4	0.9641	1.0543	0.8718	1.0188	0.7611	0.9207
LSTM	0.1	1.0016	1.0134	0.9859	0.9933	0.945	0.9604
	0.4	1.0015	1.0248	0.976	1.0544	0.9265	1.0476
	0.7	1.0015	1.0141	0.9968	1.0275	0.9714	1.0453
	-0.4	0.9662	1.0689	0.8683	1.0384	0.7774	0.9388
GRU	0.1	1.0051	1.0111	0.9838	0.9946	0.9453	0.9592
	0.4	1.0038	1.021	0.9766	1.048	0.9293	1.042
	0.7	1.0019	1.0126	0.991	1.0391	0.9719	1.0553
	-0.4	0.9677	1.0609	0.8687	1.0345	0.7697	0.9381

B.3 Weighted Loss Functions: Setting 2 + Scenario 1

Table B.9: $\text{MSFE}_{\text{RAY}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$.

	β	sd(ϵ_t)/2			sd(ϵ_t)			$2 \cdot$ sd(ϵ_t)		
	$(\phi, \theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0065	0.9933	0.987	0.9974	0.963	0.9148	0.9737	0.8731	0.8174
	(0.4,-0.3)	1.0095	0.989	0.963	0.997	0.9409	0.9076	0.9647	0.8964	0.8625
	(-0.4,0.3)	1.0	0.9742	0.9503	0.982	0.9224	0.8707	0.9337	0.8458	0.8145
	(-0.4,-0.3)	0.9724	0.8507	0.7396	0.8755	0.6453	0.5827	0.6315	0.5407	0.528
LSTM	(0.4,0.3)	1.0038	0.9924	0.9809	0.9943	0.9632	0.9094	0.9709	0.8967	0.8233
	(0.4,-0.3)	1.0066	0.9898	0.9678	0.9943	0.9511	0.9076	0.9633	0.9052	0.8663
	(-0.4,0.3)	1.0022	0.9737	0.9513	0.9816	0.9268	0.9001	0.9472	0.8741	0.8426
	(-0.4,-0.3)	0.9735	0.847	0.7319	0.885	0.639	0.5984	0.6597	0.5562	0.5636
GRU	(0.4,0.3)	1.0035	0.9929	0.9759	0.994	0.9603	0.9133	0.9682	0.8911	0.8188
	(0.4,-0.3)	1.0069	0.9878	0.9692	0.9977	0.9487	0.9159	0.9631	0.8973	0.8648
	(-0.4,0.3)	0.9999	0.9747	0.9557	0.98	0.9252	0.885	0.9359	0.878	0.8289
	(-0.4,-0.3)	0.9724	0.8468	0.7386	0.8808	0.6415	0.5944	0.6552	0.5485	0.5622

Table B.10: $\text{MSFE}_{\text{BAR}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	sd(ϵ_t)/2			sd(ϵ_t)			$2 \cdot$ sd(ϵ_t)		
	$(\phi, \theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0045	0.9958	0.9993	0.9954	0.9654	0.9262	0.9718	0.8753	0.8276
	(0.4,-0.3)	1.0072	0.9876	0.9697	0.9948	0.9397	0.9138	0.9626	0.8951	0.8684
	(-0.4,0.3)	0.9989	0.9736	0.9608	0.981	0.9219	0.8802	0.9326	0.8452	0.8235
	(-0.4,-0.3)	0.972	0.8515	0.8845	0.8751	0.6459	0.6969	0.6312	0.5412	0.6315
LSTM	(0.4,0.3)	1.0028	0.9941	0.993	0.9933	0.9649	0.9206	0.9699	0.8983	0.8335
	(0.4,-0.3)	1.007	0.9891	0.9716	0.9947	0.9504	0.9112	0.9637	0.9046	0.8697
	(-0.4,0.3)	1.0003	0.9758	0.9595	0.9798	0.9288	0.9079	0.9455	0.876	0.8499
	(-0.4,-0.3)	0.9723	0.8478	0.8815	0.8839	0.6396	0.7208	0.6589	0.5567	0.6789
GRU	(0.4,0.3)	1.0016	0.9926	0.9908	0.9922	0.9601	0.9273	0.9664	0.8909	0.8314
	(0.4,-0.3)	1.0048	0.9884	0.9717	0.9956	0.9492	0.9183	0.9611	0.8978	0.867
	(-0.4,0.3)	0.9984	0.9755	0.9652	0.9785	0.926	0.8937	0.9345	0.8788	0.8371
	(-0.4,-0.3)	0.9708	0.8475	0.8836	0.8793	0.6421	0.7111	0.6541	0.549	0.6726

Table B.11: $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	$(\phi,\theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0048	0.9918	0.9885	0.9957	0.9615	0.9162	0.9721	0.8718	0.8186
	(0.4,-0.3)	1.0101	0.9876	0.9677	0.9977	0.9397	0.912	0.9653	0.8951	0.8667
	(-0.4,0.3)	0.9968	0.9756	0.9556	0.9789	0.9237	0.8755	0.9307	0.8469	0.819
	(-0.4,-0.3)	0.9715	0.851	0.7918	0.8747	0.6455	0.6238	0.6309	0.5409	0.5653
LSTM	(0.4,0.3)	1.0031	0.9922	0.9874	0.9936	0.963	0.9154	0.9701	0.8966	0.8288
	(0.4,-0.3)	1.0067	0.9895	0.9688	0.9944	0.9508	0.9085	0.9634	0.9049	0.8671
	(-0.4,0.3)	1.0001	0.9768	0.9551	0.9797	0.9297	0.9037	0.9453	0.8768	0.8459
	(-0.4,-0.3)	0.9726	0.8477	0.7849	0.8841	0.6395	0.6418	0.659	0.5566	0.6044
GRU	(0.4,0.3)	1.001	0.9896	0.9815	0.9916	0.9572	0.9186	0.9658	0.8882	0.8236
	(0.4,-0.3)	1.0067	0.9857	0.9726	0.9975	0.9466	0.9191	0.9629	0.8953	0.8678
	(-0.4,0.3)	0.9989	0.9738	0.9605	0.979	0.9244	0.8894	0.935	0.8772	0.8331
	(-0.4,-0.3)	0.9717	0.847	0.7926	0.8801	0.6418	0.6379	0.6546	0.5487	0.6034

Table B.12: $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 1 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$.

	β	$\mathrm{sd}(\epsilon_t)/2$			$\mathrm{sd}(\epsilon_t)$			$2 \cdot \mathrm{sd}(\epsilon_t)$		
	$(\phi,\theta)/\tau^*$	0.2	0.5	0.8	0.2	0.5	0.8	0.2	0.5	0.8
RNN	(0.4,0.3)	1.0029	0.9935	1.001	0.9938	0.9632	0.9277	0.9703	0.8733	0.8289
	(0.4,-0.3)	1.006	0.9917	0.9704	0.9936	0.9436	0.9145	0.9614	0.8989	0.869
	(-0.4,0.3)	0.9982	0.9733	0.9596	0.9802	0.9215	0.8791	0.932	0.8449	0.8224
	(-0.4,-0.3)	0.9712	0.8513	0.8584	0.8744	0.6458	0.6763	0.6307	0.5411	0.6129
LSTM	(0.4,0.3)	1.0024	0.9942	0.9928	0.9929	0.9649	0.9204	0.9695	0.8984	0.8333
	(0.4,-0.3)	1.0038	0.9888	0.9722	0.9915	0.9502	0.9117	0.9606	0.9044	0.8702
	(-0.4,0.3)	0.9996	0.9767	0.9598	0.9792	0.9297	0.9082	0.9448	0.8768	0.8501
	(-0.4,-0.3)	0.9716	0.8474	0.8532	0.8833	0.6393	0.6976	0.6584	0.5564	0.657
GRU	(0.4,0.3)	1.0017	0.9923	0.9901	0.9923	0.9598	0.9266	0.9665	0.8907	0.8308
	(0.4,-0.3)	1.0064	0.987	0.9716	0.9972	0.9478	0.9182	0.9626	0.8965	0.8669
	(-0.4,0.3)	0.9984	0.9757	0.9657	0.9785	0.9261	0.8942	0.9345	0.8789	0.8376
	(-0.4,-0.3)	0.9713	0.8476	0.8577	0.8797	0.6422	0.6903	0.6544	0.549	0.6529

B.4 Weighted Loss Functions: Setting 2 + Scenario 2

Table B.13: $\text{MSFE}_{\text{RAY}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Rayleigh weight decay and $\alpha \in \{2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$

	β	$\text{sd}(\epsilon_t)/2$		$\text{sd}(\epsilon_t)$		$2 \cdot \text{sd}(\epsilon_t)$	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	(0.4,0.3)	0.9816	0.9397	0.9219	0.8654	0.8397	0.8239
	(0.4,-0.3)	0.9738	0.9321	0.9176	0.8858	0.861	0.8332
	(-0.4,0.3)	0.9545	0.9029	0.8836	0.8317	0.8323	0.8246
	(-0.4,-0.3)	0.7498	0.6213	0.5861	0.6052	0.5294	0.7312
LSTM	(0.4,0.3)	0.9747	0.9399	0.9327	0.8684	0.9076	0.8805
	(0.4,-0.3)	0.9747	0.9374	0.9263	0.8969	0.911	0.8772
	(-0.4,0.3)	0.956	0.917	0.9031	0.8649	0.8884	0.8562
	(-0.4,-0.3)	0.7467	0.6257	0.5917	0.6519	0.5672	0.8188
GRU	(0.4,0.3)	0.9758	0.941	0.9312	0.8705	0.8828	0.8557
	(0.4,-0.3)	0.9791	0.9347	0.9262	0.8737	0.8979	0.8654
	(-0.4,0.3)	0.9558	0.9139	0.9008	0.854	0.8729	0.8558
	(-0.4,-0.3)	0.747	0.6241	0.5856	0.6264	0.55	0.8201
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	1.0025	1.0117	0.9867	1.0049	0.9421	0.9787
	(0.4,-0.3)	1.0042	1.0068	0.9777	0.9897	0.9302	0.9394
	(-0.4,0.3)	0.9923	1.0038	0.9609	0.9793	0.9055	0.9322
	(-0.4,-0.3)	0.9377	0.979	0.78	0.9202	0.5896	0.8988
LSTM	(0.4,0.3)	0.9992	1.0087	0.9852	1.0019	0.9491	0.9833
	(0.4,-0.3)	1.0018	1.0053	0.9799	0.9854	0.9458	0.9508
	(-0.4,0.3)	0.9934	1.0017	0.9653	0.9813	0.9215	0.9393
	(-0.4,-0.3)	0.941	0.9785	0.7872	0.9222	0.6115	0.9252
GRU	(0.4,0.3)	0.9987	1.0123	0.9875	1.0034	0.9425	0.9865
	(0.4,-0.3)	1.0042	1.0042	0.9807	0.9923	0.9505	0.9481
	(-0.4,0.3)	0.9916	1.0008	0.9604	0.9811	0.9094	0.9295
	(-0.4,-0.3)	0.9397	0.9785	0.7858	0.9239	0.6055	0.919

Table B.14: $\mathrm{MSFE}_{\mathrm{BAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Bartlett weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	(0.4,0.3)	0.9833	0.9519	0.9253	0.9134	0.8425	0.9297
	(0.4,-0.3)	0.9708	0.9417	0.9216	0.8916	0.86	0.8386
	(-0.4,0.3)	0.9546	0.9154	0.8844	0.8405	0.8337	0.8352
	(-0.4,-0.3)	0.7513	0.75	0.5854	0.7717	0.534	0.757
LSTM	(0.4,0.3)	0.9778	0.9527	0.9388	0.9178	0.9125	0.9677
	(0.4,-0.3)	0.9748	0.944	0.9314	0.9001	0.9111	0.8739
	(-0.4,0.3)	0.9569	0.9264	0.9036	0.8687	0.8891	0.8579
	(-0.4,-0.3)	0.7468	0.7609	0.591	0.8654	0.5698	0.8337
GRU	(0.4,0.3)	0.9776	0.9586	0.9403	0.9183	0.8873	0.9576
	(0.4,-0.3)	0.9787	0.94	0.9315	0.8798	0.9007	0.8721
	(-0.4,0.3)	0.9561	0.9223	0.9039	0.8612	0.8741	0.8654
	(-0.4,-0.3)	0.748	0.7576	0.5851	0.818	0.5531	0.8409
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	1.0024	1.0135	0.991	1.0366	0.9487	1.0735
	(0.4,-0.3)	1.0022	1.0099	0.9781	0.9982	0.936	0.9452
	(-0.4,0.3)	0.9927	1.01	0.9632	0.9879	0.9075	0.9354
	(-0.4,-0.3)	0.9394	1.0999	0.7796	1.1254	0.5947	0.9089
LSTM	(0.4,0.3)	0.9975	1.0126	0.9909	1.0307	0.9545	1.0611
	(0.4,-0.3)	1.0	1.0068	0.9835	0.9931	0.9498	0.9472
	(-0.4,0.3)	0.9926	1.0102	0.9684	0.9867	0.9229	0.9409
	(-0.4,-0.3)	0.9428	1.102	0.7861	1.1868	0.6148	0.9406
GRU	(0.4,0.3)	0.998	1.0175	0.9919	1.0379	0.9503	1.0665
	(0.4,-0.3)	1.004	1.0099	0.9835	0.9966	0.9514	0.9521
	(-0.4,0.3)	0.9921	1.0086	0.9633	0.9875	0.9107	0.9375
	(-0.4,-0.3)	0.9415	1.1039	0.7848	1.1685	0.6086	0.9401

Table B.15: $\mathrm{MSFE}_{\mathrm{PAR}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Parzen weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	(0.4,0.3)	0.9809	0.9449	0.9224	0.8809	0.8432	0.8772
	(0.4,-0.3)	0.9698	0.9357	0.9202	0.8888	0.8601	0.8328
	(-0.4,0.3)	0.955	0.9072	0.8839	0.8394	0.8317	0.8302
	(-0.4,-0.3)	0.7501	0.6663	0.585	0.7145	0.5295	0.7477
LSTM	(0.4,0.3)	0.9741	0.9485	0.9323	0.8822	0.9117	0.9179
	(0.4,-0.3)	0.9747	0.9404	0.9251	0.8997	0.91	0.8744
	(-0.4,0.3)	0.9558	0.922	0.9038	0.8666	0.8885	0.8547
	(-0.4,-0.3)	0.7468	0.6694	0.5904	0.7907	0.5667	0.8315
GRU	(0.4,0.3)	0.9758	0.9438	0.9312	0.8892	0.8843	0.9045
	(0.4,-0.3)	0.9777	0.9358	0.9289	0.8766	0.9009	0.869
	(-0.4,0.3)	0.9567	0.9172	0.8999	0.855	0.8719	0.8592
	(-0.4,-0.3)	0.7473	0.6686	0.5846	0.7576	0.55	0.8387
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	1.0012	1.0161	0.9835	1.023	0.9451	1.032
	(0.4,-0.3)	1.0034	1.0102	0.9787	0.9941	0.9315	0.9414
	(-0.4,0.3)	0.9965	1.0066	0.9632	0.9869	0.906	0.9299
	(-0.4,-0.3)	0.9381	1.0494	0.7789	1.0783	0.5894	0.9083
LSTM	(0.4,0.3)	0.999	1.0128	0.9878	1.0191	0.9502	1.0273
	(0.4,-0.3)	0.9994	1.0074	0.982	0.9904	0.9475	0.9522
	(-0.4,0.3)	0.9917	1.0068	0.9658	0.9854	0.922	0.9415
	(-0.4,-0.3)	0.9415	1.0459	0.7862	1.1176	0.6111	0.9395
GRU	(0.4,0.3)	1.0001	1.0167	0.9882	1.0208	0.9469	1.0375
	(0.4,-0.3)	1.0036	1.0079	0.9825	0.9931	0.9502	0.9497
	(-0.4,0.3)	0.9926	1.0077	0.9616	0.988	0.9085	0.9365
	(-0.4,-0.3)	0.9405	1.0512	0.7845	1.1109	0.605	0.9327

Table B.16: $\mathrm{MSFE}_{\mathrm{TUK}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 and Scenario 2 using a weighted loss function with Tukey-Hanning weight decay and $\alpha \in \{T, T^{0.95}, T^{0.9}\}$

	β	sd(ϵ_t)/2		sd(ϵ_t)		$2 \cdot$ sd(ϵ_t)	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	(0.4,0.3)	0.9826	0.9538	0.9233	0.9044	0.8451	0.9163
	(0.4,-0.3)	0.9714	0.9375	0.9216	0.8943	0.8604	0.8378
	(-0.4,0.3)	0.9561	0.9135	0.8835	0.8406	0.832	0.8358
	(-0.4,-0.3)	0.7505	0.7217	0.5855	0.7698	0.5286	0.7509
LSTM	(0.4,0.3)	0.9763	0.9532	0.9372	0.9062	0.9141	0.9566
	(0.4,-0.3)	0.9751	0.9401	0.9312	0.9001	0.911	0.8756
	(-0.4,0.3)	0.9571	0.9251	0.9041	0.8686	0.8886	0.8648
	(-0.4,-0.3)	0.7468	0.7299	0.5905	0.8603	0.5652	0.843
GRU	(0.4,0.3)	0.9764	0.9522	0.9361	0.9087	0.8894	0.9484
	(0.4,-0.3)	0.9774	0.936	0.9321	0.8793	0.8984	0.8705
	(-0.4,0.3)	0.9578	0.9216	0.9025	0.8632	0.8727	0.8584
	(-0.4,-0.3)	0.7475	0.7286	0.5848	0.821	0.549	0.8402
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	1.0044	1.0192	0.9916	1.0404	0.9488	1.0795
	(0.4,-0.3)	1.0035	1.0123	0.9803	1.001	0.9339	0.9423
	(-0.4,0.3)	0.9921	1.0118	0.9616	0.9894	0.9057	0.9287
	(-0.4,-0.3)	0.9388	1.1233	0.779	1.1466	0.5884	0.9101
LSTM	(0.4,0.3)	0.9975	1.0157	0.9901	1.0365	0.9541	1.0663
	(0.4,-0.3)	0.9995	1.0083	0.9846	0.9916	0.948	0.9458
	(-0.4,0.3)	0.9933	1.0111	0.9667	0.9858	0.9225	0.9444
	(-0.4,-0.3)	0.9421	1.134	0.7865	1.1972	0.6103	0.9392
GRU	(0.4,0.3)	0.9992	1.0211	0.9914	1.044	0.9492	1.0797
	(0.4,-0.3)	1.0042	1.0109	0.9832	0.9944	0.9523	0.9535
	(-0.4,0.3)	0.9925	1.0099	0.9632	0.9893	0.9086	0.9349
	(-0.4,-0.3)	0.9415	1.1299	0.7845	1.1794	0.6038	0.9289

B.5 Excluding pre-break observations

Table B.17: $\text{MSFE}_{\text{PB}}/\text{MSFE}_{\text{EQUAL}}$ reported for Setting 1 combined with Scenario 2.

	β	sd(ϵ_t)/2		sd(ϵ_t)		$2 \cdot$ sd(ϵ_t)	
	$\phi\backslash\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	0.1	0.9649	0.9427	0.904	0.8888	0.858	0.8393
	0.4	0.9619	0.9344	0.8955	0.8535	0.8297	0.8227
	0.7	0.9773	0.9607	0.9357	0.8926	0.8637	0.8235
	-0.4	0.8507	0.8747	0.7509	0.7842	0.6784	0.7542
LSTM	0.1	0.969	0.9567	0.9175	0.8979	0.9033	0.8772
	0.4	0.9555	0.9254	0.8916	0.8521	0.8557	0.8446
	0.7	0.9794	0.9582	0.9434	0.8948	0.926	0.8939
	-0.4	0.8473	0.8962	0.7569	0.8217	0.7182	0.8082
GRU	0.1	0.9674	0.952	0.9207	0.875	0.8861	0.8819
	0.4	0.9625	0.9323	0.9024	0.8622	0.8528	0.8412
	0.7	0.9824	0.9567	0.951	0.9038	0.914	0.8648
	-0.4	0.8477	0.8798	0.7562	0.8002	0.7126	0.7972
$\beta_1 = -\beta_2$							
RNN	0.1	0.9931	1.0216	0.9738	0.9968	0.9199	0.9453
	0.4	0.9934	1.0247	0.969	1.0011	0.9042	0.9359
	0.7	0.994	1.0103	0.9825	0.9973	0.9407	0.9602
	-0.4	0.9611	1.1066	0.8686	0.9977	0.7588	0.8836
LSTM	0.1	0.9937	1.0214	0.9752	0.9986	0.9326	0.9592
	0.4	0.9912	1.0261	0.9628	1.0025	0.9071	0.9428
	0.7	0.9917	1.0106	0.9818	0.9947	0.9503	0.9637
	-0.4	0.963	1.1116	0.8657	1.0024	0.7744	0.9025
GRU	0.1	0.9941	1.0225	0.9723	0.9984	0.9312	0.9577
	0.4	0.9989	1.0247	0.9694	1.0031	0.9174	0.948
	0.7	1.0017	1.0084	0.9887	1.0021	0.9565	0.9681
	-0.4	0.9627	1.1118	0.8642	1.0041	0.7671	0.9002

Table B.18: $\mathrm{MSFE}_{\mathrm{PB}}/\mathrm{MSFE}_{\mathrm{EQUAL}}$ reported for Setting 2 combined with Scenario 2.

	β	$\mathrm{sd}(\epsilon_t)/2$		$\mathrm{sd}(\epsilon_t)$		$2 \cdot \mathrm{sd}(\epsilon_t)$	
	$(\phi,\theta)/\tau^*$	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)	(0.2,0.5)	(0.5,0.8)
$\beta_1 = \beta_2$							
RNN	(0.4,0.3)	0.9781	0.9445	0.918	0.862	0.8336	0.7966
	(0.4,-0.3)	0.9615	0.9509	0.9102	0.8956	0.8484	0.8361
	(-0.4,0.3)	0.9465	0.9315	0.8749	0.8515	0.8217	0.8338
	(-0.4,-0.3)	0.748	0.7653	0.5835	0.6557	0.5276	0.6292
LSTM	(0.4,0.3)	0.9717	0.9459	0.9281	0.8667	0.9001	0.854
	(0.4,-0.3)	0.9667	0.953	0.9192	0.9063	0.8964	0.8789
	(-0.4,0.3)	0.9512	0.9418	0.8971	0.8798	0.8788	0.8642
	(-0.4,-0.3)	0.7436	0.7646	0.5882	0.7085	0.5633	0.6859
GRU	(0.4,0.3)	0.975	0.9443	0.9298	0.8665	0.8757	0.8304
	(0.4,-0.3)	0.9681	0.9492	0.9188	0.8825	0.8849	0.8695
	(-0.4,0.3)	0.9502	0.941	0.8936	0.8711	0.8629	0.866
	(-0.4,-0.3)	0.7447	0.7649	0.5827	0.6807	0.5471	0.689
$\beta_1 = -\beta_2$							
RNN	(0.4,0.3)	0.9995	1.0178	0.9829	1.0004	0.9363	0.9453
	(0.4,-0.3)	0.9937	1.0268	0.9684	1.0024	0.9194	0.946
	(-0.4,0.3)	0.9849	1.035	0.9525	1.0021	0.8943	0.9398
	(-0.4,-0.3)	0.9353	1.2049	0.7766	0.9968	0.5874	0.7761
LSTM	(0.4,0.3)	0.9947	1.0158	0.981	0.999	0.9404	0.9584
	(0.4,-0.3)	0.9929	1.0204	0.9723	0.9973	0.9318	0.9514
	(-0.4,0.3)	0.9872	1.031	0.959	1.0013	0.912	0.9532
	(-0.4,-0.3)	0.9379	1.1962	0.7827	1.0014	0.6077	0.7772
GRU	(0.4,0.3)	0.996	1.0168	0.9841	0.9987	0.9368	0.9552
	(0.4,-0.3)	0.9941	1.0214	0.9714	1.001	0.9369	0.9527
	(-0.4,0.3)	0.9856	1.034	0.9543	1.0041	0.8992	0.9468
	(-0.4,-0.3)	0.9372	1.1992	0.7815	1.0013	0.6022	0.7787

Appendix C Additional Results: Application

This section reports additional results for section 4.5. In particular, we show the overall performance of LSTM and GRU architectures on the voltage measurements in the electric power grid in Germany.

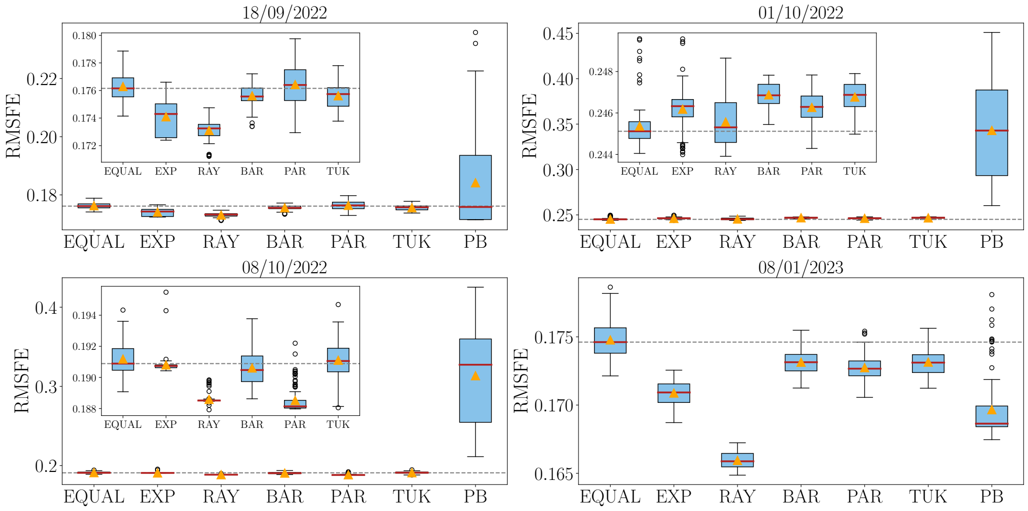

Figure C.1: *Overall performance* - distribution of the RMSE on the test set for 100 trained LSTM architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights.

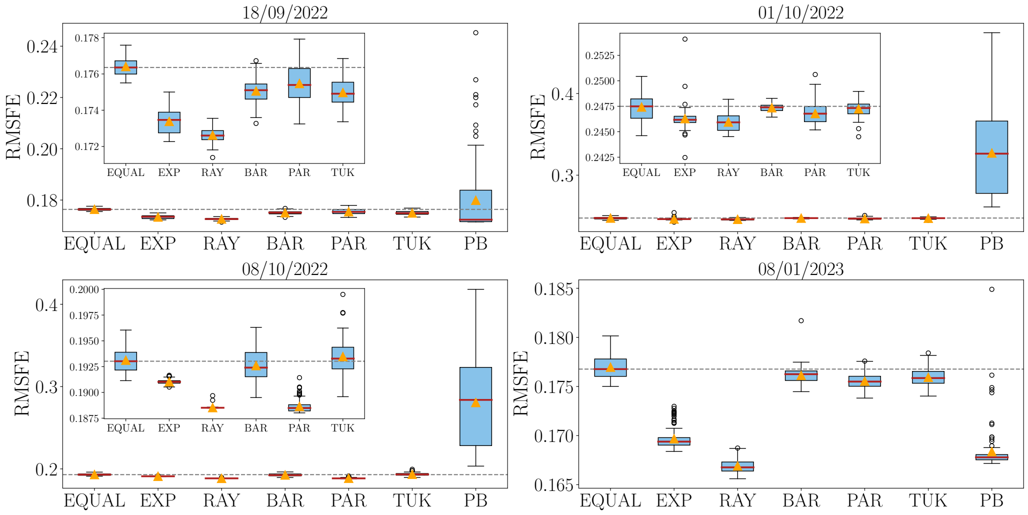

Figure C.2: *Overall performance* - distribution of the RMSE on the test set for 100 trained GRU architectures. The red bars provide the sample median and the orange triangles the sample mean. The grey dashed horizontal line corresponds to the median RMSFE when using equal weights.

BIBLIOGRAPHY

Alberola, E., Chevallier, J., & Chèze, B. (2008). Price drivers and structural breaks in European carbon prices 2005–2007. *Energy Policy*, *36*(2), 787–797.

Aliakbarian, M. S., Saleh, F. S., Salzmann, M., Fernando, B., Petersson, L., & Andersson, L. (2017). Encouraging LSTMs to anticipate actions very early, 280–289.

Andreou, E., & Ghysels, E. (2002). Detecting multiple breaks in financial market volatility dynamics. *Journal of Applied Econometrics*, *17*(5), 579–600.

Andrews, D. W. K. (1991). Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica*, *59*(3), 817.

Bai, J., & Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, *66*(1), 47.

Bai, J., & Perron, P. (2003). Critical values for multiple structural change tests. *The Econometrics Journal*, *6*(1), 72–78.

Bai, W., Suzuki, H., Qin, C., Tarroni, G., Oktay, O., Matthews, P. M., & Rueckert, D. (2018). Recurrent neural networks for aortic image sequence segmentation with sparse annotations, 586–594.

Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., & Shephard, N. (2008). Designing realized kernels to measure the ex post variation of equity prices in the presence of noise. *Econometrica*, *76*(6), 1481–1536.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bhardwaj, G., & Swanson, N. R. (2006). An empirical investigation of the usefulness of ARFIMA models for predicting macroeconomic and financial time series. *Journal of Econometrics*, *131*(1-2), 539–578.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time series analysis: Forecasting and control* (Fifth edition). Wiley; EBSCO Industries.

Box, G. E., & Tiao, G. C. (2011). *Bayesian inference in statistical analysis.* John Wiley & Sons.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Cabrera, L., & Rodríguez, G. (2011). Detrended fluctuation analysis of significant wave height time series. In C. A. Brebbia, G. Benassai, & G. R. Rodriguez (Eds.), *Coastal processes ii* (pp. 333–341). WIT PressSouthampton, UK.

Chan, F.-H., Chen, Y.-T., Xiang, Y., & Sun, M. (2017). Anticipating accidents in dashcam videos, 136–153.

Chavez-Demoulin, V., Davison, A. C., & McNeil, A. J. (2005). Estimating value-at-risk: A point process approach. *Quantitative Finance*, *5*(2), 227–234.

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). *On the properties of neural machine translation: Encoder-decoder approaches*. arXiv.

Clements, M. P., & Hendry, D. F. (2006). Chapter 12: Forecasting with breaks. In G. Elliott, C. W. J. Granger, & A. Timmermann (Eds.), *Handbook of economic forecasting* (pp. 605–657). Elsevier North Holland.

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, *117*(48).

de Boom, C., Demeester, T., & Dhoedt, B. (2019). Character-level recurrent neural networks in practice: Comparing training and sampling schemes. *Neural Computing and Applications*, *31*(8), 4001–4017.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, *74*(366a), 427–431.

Ding, W., Shang, Y., Guo, L., Hu, X., Yan, R., & He, T. (2015). Video popularity prediction by sentiment propagation via implicit network. In J. Bailey (Ed.), *Proceedings of the 24th acm international on conference on information and knowledge management* (pp. 1621–1630). ACM.

Du, Y., Wang, J., Feng, W., Pan, S., Qin, T., Xu, R., & Wang, C. (2021). AdaRNN: Adaptive learning and forecasting for time series. In G. Demartini (Ed.), *Proceedings of the 30th acm international conference on information & knowledge management* (pp. 402–411). Association for Computing Machinery.

Duan, W., He, X., Zhou, L., Thiele, L., & Rao, H. (2023). Combating distribution shift for accurate time series forecasting via hypernetworks. *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, 900–907.

Durham, G., Geweke, J., Porter-Hudak, S., & Sowell, F. (2019). Bayesian inference for ARFIMA models. *Journal of Time Series Analysis*, *40*(4), 388–410.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Fearnhead, P., & Prangle, D. (2012). Constructing summary statistics for approximate Bayesian computation: Semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *74*(3), 419–474.

Ferreira, A., & de Haan, L. (2015). On the block maxima method in extreme value theory: PWM estimators. *The Annals of Statistics*, *43*(1).

Fraunhofer Institute for Solar Energy Systems. (2023). Energy charts. Retrieved March 29, 2023, from www.energy-charts.info

Gallant, A. R. (1987). *Nonlinear statistical models*. Wiley.

Gardner, E. S. (2006). Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting*, *22*(4), 637–666.

Gomez-Rodriguez, M., Balduzzi, D., & Schölkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 561–568.

Goodfellow, I., Courville, A., & Bengio, Y. (2016). *Deep learning*. The MIT Press.

Hamilton, J. D. (2020). *Time series analysis*. Princeton University Press.

Herrera, R., & Schipp, B. (2014). Statistics of extreme events in risk management: The impact of the subprime and global financial crisis on the German stock market. *The North American Journal of Economics and Finance*, *29*, 218–238.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidthuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In J. F. Kolen & S. C. Kremer (Eds.), *A field guide to dynamical recurrent networks*. IEEE Press; IEEE Xplore.

Jain, A., Singh, A., Koppula, H. S., Soh, S., & Saxena, A. (2016). Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In A. Okamura & A. Menciassi (Eds.), *2016 ieee international conference on robotics and automation, stockholm, sweden, may 16th-21st* (pp. 3118–3125). IEEE.

Kim, S., Shephard, N., & Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, *65*(3), 361–393.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., & Choo, J. (2021). Reversible instance normalization for accurate time-series forecasting against distribution shift. *International Conference on Learning Representations*.

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*. arXiv.

Lee, D., Li, W. K., & Wong, T. S. T. (2012). Modeling insurance claims via a mixture exponential model combined with peaks-over-threshold approach. *Insurance: Mathematics and Economics*, *51*(3), 538–550.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, *379*(2194).

Lim, B., Zohren, S., & Roberts, S. (2020). Recurrent neural filters: learning independent Bayesian filtering steps for time series prediction. *2020 International Joint Conference on Neural Networks (IJCNN)*.

Lin, B., & Zhang, C. (2022). Forecasting carbon price in the European carbon market: The role of structural changes. *Process Safety and Environmental Protection*, *166*, 341–354.

Marin, J.-M., Pudlo, P., Robert, C. P., & Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, *22*(6), 1167–1180.

McNeil, A. J., & Frey, R. (2000). Estimation of tail-related risk measures for heteroscedastic financial time series: An extreme value approach. *Journal of Empirical Finance*, *7*(3-4), 271–300.

Newey, W. K., & West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, *55*(3), 703.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples.*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Pesaran, M. H., & Pick, A. (2011). Forecast combination across estimation windows. *Journal of Business & Economic Statistics*, *29*(2), 307–318.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems.* Curran Associates, Inc.

Rapach, D. E., & Strauss, J. K. (2008). Structural breaks and GARCH models of exchange rate volatility. *Journal of Applied Econometrics*, *23*(1), 65–90.

Rapach, D. E., & Wohar, M. E. (2005). Regime changes in international real interest rates: Are they a monetary phenomenon? *Journal of Money, Credit and Banking*, *37*(5), 887–906.

Rossi, R. J. (2018). *Mathematical statistics: An introduction to likelihood based inference.* John Wiley & Sons.

Rydén, T., Teräsvirta, T., & Åsbrink, S. (1998). Stylized facts of daily return series and the hidden Markov model. *Journal of Applied Econometrics*, *13*(3), 217–244.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191.

Schäfer, B., Matthiae, M., Timme, M., & Witthaut, D. (2015). Decentral smart grid control. *New Journal of Physics*, *17*(1).

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference.*

Tian, J., & Anderson, H. M. (2014). Forecast combinations under structural break uncertainty. *International Journal of Forecasting*, *30*(1), 161–175.

Tonutti, M., Ruffaldi, E., Cattaneo, A., & Avizzano, C. A. (2019). Robust and subject-independent driving manoeuvre anticipation through domain-adversarial recurrent neural networks. *Robotics and Autonomous Systems*, *115*, 162–173.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Wallinga, J., & Teunis, P. (2004). Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American journal of epidemiology*, *160*(6), 509–516.

Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., & Yu, P. (2022). Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 1.

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., & Januschowski, T. (2019). Deep factors for forecasting. *International Conference on Machine Learning*, 6607–6617.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75.

Zeileis, A., Leisch, F., Hornik, K., & Kleiber, C. (2002). strucchange : An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, *7*(2).