FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Pastcasting: improved forecasting of the future by correcting the past

André Carlos Almeida Baptista



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Soares

Second Supervisor: Yassine Baghoussi

Company Supervisor: Miguel Arantes

August 3, 2020

Pastcasting: improved forecasting of the future by correcting the past

André Carlos Almeida Baptista

Mestrado Integrado em Engenharia Informática e Computação

August 3, 2020

Abstract

Forecasting refers to the process of anticipating the future by leveraging information from the past. That information is usually represented in the form of time series, which are collections of observations made through time. In retail, time series consisting of past sales data are used to predict future demand. The forecasting accuracy depends on the quality of data, which might be affected by the occurrence of anomalies. Stock shortages are an example of this: they cause lower amounts of sales even when the demand remains the same. Therefore, using the anomalous sales data to perform forecasts will likely result in a underestimation of demand.

We address the general problem of anomalies affecting data quality by *pastcasting*: predicting how data should have been in the past in order for it to better explain the future. We propose *Pastprop*, an adaptation of the backpropagation algorithm used by LSTM that assigns part of the responsibility for errors to the training data and changes it accordingly. Three variants of Pastprop were developed and tested on artificial and benchmark data, as well as on a case study from the retail domain. We evaluate the ability of Pastprop to reconstruct anomalies; its effects on forecasting accuracy; and the benefits of using the corrected data with other forecasting methods. The results indicate that the proposed method is not only able to reconstruct data that were affected by anomalies but also to improve forecasting accuracy when compared to a standard LSTM. Additionally, ARIMA benefited from learning with data corrected by our method.

Keywords: time series, forecasting, pastcasting, backpropagation, LSTM, anomaly correction, data quality

Resumo

Previsão é o processo de antecipar o futuro tirando partido de informação sobre o passado. Essa informação é, geralmente, representada sob a forma de séries temporais, que são observações de uma certa variável feitas ao longo do tempo. No retalho, séries temporais sobre as vendas registadas no passado são usadas para prever a procura futura. A precisão das previsões depende da qualidade dos dados que, por sua vez, podem ser afetados pela ocorrência de anomalias. Um exemplo disso são as falhas de stock, que resultam numa menor quantidade de vendas mesmo quando a procura se tinha mantido. Assim sendo, o uso de dados anómalos sobre vendas passadas para realizar previsões pode levar a uma subestimação da procura futura.

Abordamos o problema da ocorrência de anomalias em séries temporais e os seus efeitos nas previsões através da técnica de *pastcasting*: prever como os dados deveriam ter sido no passado para que expliquem melhor o futuro. Propomos o método *Pastprop*, uma adaptação do algoritmo de backpropagation usado pela LSTM que atribui parte da responsabilidade pelos erros aos dados de treino e altera-os em conformidade. Desenvolvemos três versões de Pastprop e testámo-las em dados gerados artificialmente, em dados de referência e ainda num caso de estudo do domínio do retalho. Avaliamos o método quanto à sua capacidade de reconstruir anomalias; aos seus efeitos na qualidade das previsões; e ainda quanto aos possíveis benefícios de usar os dados corrigidos noutros métodos de previsão. Os resultados indicam que o método proposto não só é capaz de reconstrutir dados que foram afetados por anomalias como também de melhorar as previsões comparativamente a uma LSTM padrão. Adicionalmente, o método ARIMA beneficiou em ser treinado com dados corrigidos por Pastprop.

Keywords: séries temporais, previsão, pastcasting, backpropagation, LSTM, correção de anomalias, qualidade de dados

Agradecimentos

Este trabalho foi parcialmente apoiado pelo SONAE IM.LAB@FEUP, através de um projeto de investigação de M.Sc. financiado pela Inovretail.

Gostaria de agradecer ao Professor Carlos Soares pelas excelentes oportunidades que me proporcionou e por confiar em mim para desenvolver este projeto; pelo entusiasmo que demonstrou em todos os momentos e pelo gosto em discutir ideias; pelas suas orientações durante este percurso. Ao Miguel Arantes gostaria de agradecer a forma como me recebeu na empresa, procurando sempre que me sentisse bem e integrasse da melhor maneira; agradeço a sua total disponibilidade para esclarecimento de dúvidas e a ajuda preciosa que me prestou ao longo destes meses. Ao Yassine Baghoussi faço um agradecimento em inglês: thank you for all your contributions, your great suggestions and knowledgeable insights, as well as the practical help you've provided me with. Irei lembrar com saudade as nossas reuniões, presenciais e remotas. Estendo os meus agradecimentos ao João Guichard e a todo o pessoal da Inovretail.

Agradeço aos meus pais e ao meu irmão. À minha mãe e ao meu pai, que eu adoro e que me apoiaram sempre em tudo na vida. Ao meu irmão, de quem eu me orgulho profundamente. Agradeço à minha família, aos meus avós, tios e primos por estarem sempre ao meu lado. Aos meus amigos, pela força que me dão. A todos eles dedico este trabalho.

André Carlos Almeida Baptista

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 2
- 1.3 Goals . . . 2

2 Related Work **5**
- 2.1 Pastcasting . . . 5
  - 2.1.1 Similar approaches . . . 6
- 2.2 Time series forecasting . . . 7
  - 2.2.1 Demand forecasting in retail . . . 8
- 2.3 Neural Networks . . . 9
  - 2.3.1 Perceptron . . . 9
  - 2.3.2 Architecture . . . 10
  - 2.3.3 Forward pass . . . 10
  - 2.3.4 Backpropagation . . . 11
  - 2.3.5 Recurrent Neural Networks . . . 12
  - 2.3.6 Long Short-Term Memory . . . 13

3 Pastprop **19**
- 3.1 Calculating deltas . . . 19
- 3.2 Pastprop variants . . . 21
  - 3.2.1 Regular Pastprop . . . 21
  - 3.2.2 Progressive Pastprop . . . 22
  - 3.2.3 Selective Pastprop . . . 22
- 3.3 Important notes . . . 23
- 3.4 Hyperparameters . . . 24

4 Experimental setup **25**
- 4.1 Anomaly generation . . . 25
- 4.2 Datasets . . . 26
  - 4.2.1 Artificial data . . . 26
  - 4.2.2 Benchmark data . . . 27
  - 4.2.3 Retail case study data . . . 28
- 4.3 Methodology . . . 28
  - 4.3.1 Baseline algorithms . . . 28
  - 4.3.2 Reconstruction Ability and Outside Loss . . . 29

5 Results **31**
5.1 Reconstructing the past 31
5.1.1 Artificial data 31
5.1.2 Benchmark and case study data 34
5.2 Forecasting the future while reconstructing the past 36
5.2.1 Forecasting on test data 36
5.2.2 Benchmark and case study data 38
5.3 Forecasting the future with a reconstructed past 43
5.3.1 Corrections over anomalous data 43
5.3.2 Corrections over original data 45

6 Conclusions **49**

References **51**

List of Figures

2.1 Architecture of an LSTM block. 13

3.1 Simplified illustration of how Regular Pastprop works. The green box is a sample and the blue box is the respective label. The green dashed box contains the deltas for that sample. 22
3.2 Simplified illustration of how Progressive Pastprop works. The green box is a sample and the blue box is the respective label. The green dashed box contains the deltas for that sample. The deltas are divided by the corresponding factor at the bottom. 23

4.1 Examples of anomalies with different levels of magnitude. From left to right: level 0, level 25 and level 50. The blue lines are the original series and the orange lines are the anomalous series. Please note the different graph scales. 26
4.2 Example to visualize anomaly reconstruction. Blue is the original series, orange is the anomaly and green is the corrected series. The symbol a illustrates the difference between the corrections and the original data; b illustrates the difference between the anomaly and the original data. 29

5.1 Proof of concept: Selective Pastprop on the sinusoidal series trained for 1000 epochs (50 waiting epochs) with a data correction rate of 1e−1. Blue is the original series, orange is the anomaly and green is the corrected series. 32
5.2 Best examples of anomaly reconstruction achieved with the (1000; 1e−2) hyperparameter combination out of the experiments on benchmark and case study data. Blue lines are the original series, orange are the anomalies and green are the corrected series. 34
5.3 Corrections (green) made to one of the M5 time series that was affected by an anomaly of magnitude level 50 at the middle position (orange). The corrections were obtained through Progressive Pastprop while using the (1000; 1e−2) hyperparameter combination. This image is an expanded view of the top left graph in figure 5.2. 35
5.4 Illustration of the forecasting mechanism on test data of Pastprop and LSTM. The rectangles represent the time steps of the time series. Green rectangles denote input samples while blue rectangles indicate what is being forecasted. The numbers indicate the horizon at which the corresponding values are predicted. 37
5.5 Top: LSTM forecast on electricity data that was affected by an anomaly at middle of training data. Bottom: Progressive Pastprop forecast on the same data with the hyperparameters (50; 1e−1). Blue are the test data and orange are the predictions. The improvement on forecasting accuracy was of 66.7%. 40

5.6 Corrections (orange) made to an M5 series with no anomalies (blue) with Selective Pastprop and (200; 1e−1). When provided as input data to ARIMA, this series was the best at improving forecasting accuracy in comparison to the original series, as shown in Figure 5.7. 47
5.7 At the top: forecast (on test data) using ARIMA with a M5 series at 7 day horizon. At the bottom: forecast on the same test data but giving ARIMA the corrected training series shown in Figure 5.6. The bottom forecast was 33% more accurate in terms of MSE. Blue are the real test data and orange are the predictions. 48

List of Tables

5.1 Anomaly reconstruction ability (AR) and outside loss (OL) on artificial data, averaged by combination of epochs and data correction rate. The anomalies occurred at the middle of data. 33

5.2 Average outside losses aggregated by anomaly position on artificial data. The experiments excluded (1000; 1e−2), (200; 1e−1) and (1000; 1e−1). 33

5.3 Anomaly reconstruction ability (AR) and outside loss (OL) on benchmark and case study data, averaged by combination of epochs and data correction rate. The anomalies occurred at the middle of data. 35

5.4 Forecasting accuracies (MSE) at 1, 7 and 28 day horizons obtained by LSTM, Pastprop, ARIMA and ETS on univariate benchmark data (3 M5 series and 3 electricity series). Each series was affected by an anomaly at the middle of training data. Results were aggregated by combination of epochs and data correction rate. 39

5.5 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by each Pastprop variant in relation to LSTM. This information can be inferred from table 5.4 when considering only the hyperparameters (50; 1e−2) and (50; 1e−1). 41

5.6 Forecasting accuracies (MSE) at 1, 7 and 14 day horizons obtained by LSTM, Pastprop, ARIMA and BATS on all benchmark and case study data. The series were not affected by manually inserted anomalies. Results were aggregated by combination of epochs and data correction rate. The maximum horizon of 14 days was imposed by the case study. 42

5.7 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by each Pastprop variant in relation to LSTM. This information can be inferred from table 5.6 when considering only the hyperparameters (50; 1e−2) and (50; 1e−1). 42

5.8 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by ARIMA when using data corrected by each Pastprop variant vs. ARIMA with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were affected by anomalies of magnitude levels 0 and 50 at the middle position. The horizon of the forecasts was of 7 days. 44

5.9 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by LSTM when using data corrected by each Pastprop variant vs. LSTM with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were affected by anomalies of magnitude levels 0 and 50 at the middle position. The horizon of the forecasts was of 7 days. 45

5.10 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by ARIMA when using data corrected by each Pastprop variant vs. ARIMA with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were not affected by anomalies. The horizon of the forecasts was of 7 days. 46
5.11 Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by LSTM when using data corrected by each Pastprop variant vs. LSTM with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were not affected by anomalies. The horizon of the forecasts was of 7 days. 47

Abbreviations

NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ARIMA	Autoregressive Integrated Moving Average
MSE	Mean Squared Error

Chapter 1

Introduction

In general, forecasts are used as a means of supporting decision making. Successful decisions are dependent on accurate forecasts. Likewise, the accuracy of forecasts is dependent on the quality of available data. Therefore, enhancing data quality is a relevant direction when attempting to improve forecasting results.

The following sections describe the context, motivation and goals behind this work. Furthermore, the structure of the next chapters is presented.

1.1 Context

When dealing with forecasting problems, data is usually represented in the form of time series. Time series data can be defined as a collection of values observed sequentially through time within constant time intervals [40][16]. Due to the nature of this type of data, each value in the series is typically related to other observations. That is, previous instances in the series are generally good predictors of subsequent values. Moreover, time series are prone to express other complex relationships between observations, such as trends and seasonality.

The occurrence of trends and seasonal patterns in sales data is very common in the retail domain. Trends dictate the popularity of products and their expected sales growth. In fashion retail, for example, new trends may be originated in quite unpredictable and inexplicable manner. Seasonality can also affect consumer behaviour. Due to necessity, some items have increased demand at specific times of the year, such as warm clothes during winter. Additionally, calendar events such as Easter, Black Friday and Christmas have huge impacts across multiple types of retail businesses. Furthermore, other factors such as promotions, pricing and competitors' behaviour [47], as well as marketing actions and the economic environment [37] can greatly influence the retailer's sales. The combination of all these variables makes demand forecasting in retail a challenging task.

Forecasting models support crucial business activities such as inventory planning and cost optimization, supply management, staff scheduling, competitive pricing and promotion planning [14]. These activities are ultimately linked to the retailer's ability to generate profit. For this reason, demand forecasting has been recognized as a key process in retail for a long time. In earlier days, companies would solely rely on the knowledge and opinions of experts to predict future demand. Eventually, expert based approaches were no longer satisfactory due to the increasing volumes of data involved. Statistical and machine learning methods took their place and have since been used extensively in this area [60, 19, 21, 48, 12, 46, 47]. More recently, deep learning techniques have also been applied to time series forecasting due to their potential of modeling complex dynamics in long and multivariate time series [37, 62, 29, 53, 13, 14, 8, 52, 58, 54].

1.2 Motivation

A common concern to any forecasting method is the quality of the available data. In retail, past sales data are used to estimate future demand. However, it should be noted that sales may sometimes be unrepresentative of demand. For example, when a product goes out of stock for some time period, the registered sales for that product will not take into account the number of lost sales opportunities. This is a case of *censored observation* [43]. In other words, the observed value of sales is limited by the available stock and the real demand is outside the measurable range. This type of anomaly effectively decreases the quality of input data. Another example of anomalous data is the temporary closing of a store for some extraordinary reason. During that time period there will be no recorded sales. Therefore, those zero values will probably be bad predictors of demand when the store reopens. Finally, an example of what would not be considered an anomaly is the sales increase during the Christmas season. The reasoning behind this is that, despite constituting an outlier, the phenomenon is expected and likely helpful at explaining future time series values.

1.3 Goals

Motivated by these issues, we develop a concrete solution that address the general problem of *pastcasting* [35, 7] - to change past data in a way that makes it a better predictor of the future. We propose *Pastprop*, which consists of modifying the backpropagation algorithm of an LSTM implementation with the goal of correcting anomalies in data and reducing their effect on forecasting accuracy. The idea is to make the learning process estimate the contribution of training data to the errors in the predictions. In other words, responsibility for errors should be shared not only among the network's weights but also with data.

The following are the main contributions of our work: (1) novel idea of adapting backpropagation for the purpose of data anomalies correction as a part of the learning process; (2) three variants of Pastprop; (3) empirical study of the approach on artificially generated data, benchmark datasets and a case study from the retail domain.

We identify the following as the main research questions of our empirical study:

- How effective is Pastprop at reconstructing anomalies in data?
- Can Pastprop improve time series forecasting accuracy?
- Do other algorithms benefit from using the series corrected by Pastprop as training data?

The remainder of the document is organized as described here. Chapter 2 provides an overview of the Related Work and is divided into 3 sections. Section 2.1 presents how pastcasting and other related methodologies have been used in other works. Section 2.2 presents a literature review on time series forecasting, with a focus on its application to retail. Section 2.3 covers core background knowledge related to Neural Networks and LSTM. Chapter 3 defines the proposed solution, explaining the implementation of the three Pastprop variants. Finally, chapter 4 addresses the experimental setup that was designed to perform the empirical study and its results are discussed in chapter 5.

Chapter 2

Related Work

This chapter provides background information and a literature review on the fundamental research topics that are related with this work. Initially, the concept of pastcasting is presented. Then, time series forecasting is discussed, as well as its application to the retail domain. Finally, the essential concepts of neural networks are explained, with a particular focus being given to the backpropagation algorithm and LSTM.

2.1 Pastcasting

When it comes to supporting decision making and planning activities, predicting the future holds great value for individuals, organizations and businesses. However, forecasting is not the only valuable process involving temporal data. In fact, other methods such as backcasting, pastcasting and recasting have the potential to increase overall problem understanding and help achieve better results [7]. These methodologies have been comprehensively explained as well as studied in the context of urban planning in [7]. Here, a brief overview of these concepts is provided so that we can clearly distinguish them.

Backcasting is mostly defined as planning method [49, 32] and has been the topic of a considerable amount of studies. A simple research on the Engineering Village database shows that backcasting has been predominantly featured in works related to sustainable development, strategic planning and government policies. Despite being labeled as a method for the most part, that definition has also been questioned and broadened to more a of a "general approach" [22]. The idea of backcasting is inverse to that of forecasting in the sense that it takes a future state to predict the present. The usefulness of it comes from starting with a desirable future state and then assessing the intermediate steps required to achieve that future [7], such as what investments should be made in a society to achieve sustainability [32], for example.

Recasting is a type of analysis which starts from a previous state and predicts the current state, making use of forecasting techniques [7]. Therefore, the differences between the projected and

the real current state are exposed. This technique can be used to investigate what variables the model failed to account for, and how the model can be calibrated accordingly [7]. This type of analysis seems to be exclusive to long term planning contexts. However, a parallel could be drawn between the idea of recasting and the way machine learning based forecasting algorithms perfect their models over past data.

Pastcasting starts by considering a real or fictional state in the present and attempts to predict a past state. This strategy can be applied to better understand what caused the current situation or to explore alternative courses of action that could have lead to a more ideal present scenario [7]. While this definition is more aligned with the context of planning, slightly different notions of pastcasting have also received application in other fields. Predicting growth rates of macroeconomic indicators such as the Gross Domestic Product (GDP) is the most notable example of this [35]. Interesting challenges are presented when studying these indicators due to the nature of the data involved. Estimates of GDP are released at a low frequency (quarterly) and with considerable delay relatively to the end of the time period they respect to [35]. This causes the need for using higher frequency variables when predicting GDP. Also, an important issue that arises is the unavailability of GDP estimates for previous quarters for some duration after a new quarter begins. This is where pastcasting can be applied. In this scope, pastcasting is defined as the process of predicting past, yet to be available data [35].

Finally, nowcasting should also be mentioned. It is essentially the same process as forecasting, but at very short horizons. This terminology is frequently used in two major fields: meteorology and economics. In fact, nowcasting has been the focus of multiple studies regarding GDP indicators (unlike pastcasting which has only appeared in the aforementioned work [35]). The idea is to estimate GDP for the undergoing quarter. The process is repeated as new information on the higher frequency variables becomes available during the quarter. Another characteristic of GDP data is the fact that previous releases are periodically updated and revised. The use of the most recently available data has shown to be beneficial when nowcasting [9, 35].

We conclude that the presented concepts are applicable to different research areas and can be considered to be general rather than well defined methodologies. Out of all of them, pastcasting is the one that more closely describes the method we propose in chapter 3.

2.1.1 Similar approaches

We highlight three works that share common objectives with our proposed method as well as with pastcasting in general, while also making use of similar techniques. In [44], two recurrent neural networks were used to make predictions in both directions, which allowed them to detect and correct anomalies in data. In [23], recurrent variants of Generative Adversarial Networks (GAN) were used to generate realistic time series. The method was applied to medical data and it was shown that the generated series had practical value to be used in supervised training tasks. GAN is also used in [61]. They perform a missing data imputation task by having the generator predict the missing values and the discriminator guess which were real or imputed values.

2.2 Time series forecasting

Time series data is defined as a "collection of large number of data values within a uniform time interval" [40]. The same concept can be described as a "collection of observations made sequentially through time" [16], which broadens the definition. However, in most literature, such as [24] and [25], "high dimensionality" is pointed as a characteristic of this type of data.

Time series are susceptible to multiple types of analysis and data mining tasks. Those include: similarity measuring (which supports other tasks), indexing, clustering, classification, prediction, segmentation, visualization, summarization, rule discovery and motif discovery [24, 25]. Forecasting, also referred to as prediction, is the focus of this review.

Formally, a time series is denoted by a sequence of values $x_1, x_2, ... x_N$ where the indexes refer to time steps and N is the length of the time series. The last element, x_N, often corresponds to the most recent observation that is available for the variable represented in the time series. When forecasting, the goal is to predict the future at a certain horizon h. Forecasts might include just the predicted value at that horizon, $\hat{x}_{N+h}$, or even all predicted values up to that point if h is specified as a maximum horizon ($\hat{x}_{N+1}, \hat{x}_{N+2}, ... \hat{x}_{N+h}$). Nevertheless, as highlighted in [16], it is important to mention at which time the forecast was performed, which is missing in this notation. Therefore, the forecast performed at time step N (that is, using information up to that point) of the value exactly at horizon h is denoted by $\hat{x}_N(h)$ [16].

Some of the most referenced forecasting methods are based on the Autoregressive Integrated Moving Average (ARIMA) family of models, firstly introduced in 1970 [27]. Exponential smoothing is another popular class of methods which originated in the late 1950s [15, 6, 60]. These two classes are considered to be the most commonly used in times series forecasting [34]. They have been a staple in the literature for a long time, such that newly proposed approaches often try to improve on their results. Both ARIMA and exponential smoothing are categorized as linear techniques, since forecasted values are "linear functions of past observations" [41]. An example of non linear techniques are the methods that derive from the Generalized Autoregressive Conditional Heteroscedastic (GARCH) family [12]. All classes of methods presented until now are known as traditional or statistical forecasting approaches. These are usually only prepared to work with univariate time series. On the other hand, machine learning methods are capable of incorporating additional, "explanatory" [16] time series to help forecast the main variable. Within this category, popular methods include Random Forest Regression (RFR), Extreme Gradient Boosting (XGB), Support Vector Regression (SVR), Extreme Learning Machines (ELM) and Bayesian Neural Networks [29, 54, 47, 10]. Recently, with the increase in popularity of deep learning, these techniques have also been applied to time series forecasting. Long Short-Term Memory (LSTM) is a type of recurrent neural network that has been frequently used for these purposes [37, 62, 29, 53, 13, 14, 8, 52, 58] and has shown to outperform traditional methods, including ARIMA [54].

2.2.1 Demand forecasting in retail

There is a wide range of domains where time series forecasting is applied. Some examples are economic planning, sales forecasting, inventory control, budgeting and financial risk management [16]. A collection of relevant studies regarding demand forecasting in the retail domain will be mentioned here.

[28] proposed a method for multivariate forecasting that integrated data preparation and preprocessing, variable selection and optimization and a forecasting module. The approach was tested in several datasets from the retail space. It was concluded that irrelevant input variables were effectively discarded, which improved forecasting accuracy.

[33] studied the value of competitive information, regarding price and promotions, in the forecasting accuracy of individual product sales. The proposed method included variable selection methods to identify the "most relevant competitive explanatory variables" [33]. The experiments showed that competitive information at the UPC level is more valuable for sales forecasting than the same information at the focal product level. Regarding a similar research field, [38] explored the effects of "intra- and inter-categorical promotional information" in sales forecasting of individual products. The dimensionality problem that arises from this type of information is addressed. The method proposed by [38] is shown to improve forecasting accuracy while reducing dimensionality of the promotional information. It is also concluded that intra-category information provides greater improvements in accuracy when compared to inter-category. This suggests that the conclusions from [38] are aligned with those of [33].

[55] conducted a study on demand forecasting for new items in the scope of fashion retail. A set of machine learning and neural network methods were used to model which item's attributes were responsible for increasing demand. The results showed that tree based models outperformed deep learning methods. It is also noted that sales forecasting for new fashion items is still a quite unexplored field.

[36] marks the first research to use clustering techniques for "combining forecasts from multiple items" in retail. Two methods, "simple clustering" and "weighted clustering", were proposed in [36]. The sales forecasts for item clusters were shown to substantially outperform individual item forecasts. More recently, [18] explored various combinations of clustering and machine learning techniques. When applied to computer retailing data, the best performing forecasting model was the one that combined a GHSOM clustering technique with an ELM network.

[19] provides a comparison between linear and nonlinear models for sales forecasting in retail. A particular focus is put on the modeling of seasonal behaviour within time series. Nonlinear methods were concluded to be the best for "modeling retail sales movement" [19]. Another important conclusion was that previously adjusting the seasonality of data is beneficial for the forecasting accuracy of neural networks [19].

2.2.1.1 Censored demand estimation

In retail, the ability to forecast demand is essential. This process is also referred to as "sales forecasting" in the literature and throughout this paper. However, a distinction between the two terms - sales and demand - is important for the issue presented here. The amount of sales a retailer makes during some time period can be objectively measured. On the other hand, the demand cannot as it relates to the total amount of potential sales. Those include the actual sales and the sales that would be made if the conditions were ideal. In such scenarios, the observed sales would be equal to the demand. Even though defining the concept of "ideal conditions" in the context of retail is probably very hard, we are more interested in identifying situations where conditions are clearly not ideal. In such cases, the observed sales are usually considerably lower than the actual existing demand. The presented issue relates to the general topic of *censored observations*. An analysis of censored data within time series can be found in [43].

The problematic of demand forecasting in the presence of censored sales data has been addressed in the literature. Followingly is a brief overview of these works. [45] tackles the problem through the use of ensemble methods. In the study, machine learning methods are combined while accounting and not accounting for data censoring. [11] studied the issue while using sales data of the same products from multiple retail stores. They proposed a method which includes modeling the problem with a "latent variable" and using "matrix completion" related techniques [11]. Their results argue that the demand forecasting error would tend to zero as the number of considered stores and timestamps increased to infinity. [42] addresses the problem while considering that the retailer's stock levels might not be accurately known at all times. They concluded that ignoring this possibility in the presence of censored sales data usually leads to a "systematic underestimation of demand" [42].

2.3 Neural Networks

An artificial neural network, commonly reffered to as neural network, is defined as "an abstract computational model designed to solve a variety of supervised and unsupervised learning tasks" [57]. The concept of neural network is inspired by the functioning of the human brain. According to the historical background provided in [57], the early origins of this idea are credited to McCulloch and Pitts, back in 1943. Hebb pushed the work forward in 1949. In 1959, Rosenblatt proposed the perceptron model, which is still the basic structure of artificial neurons used in present day NNs.

2.3.1 Perceptron

A perceptron can be described as a processing unit which takes multiple inputs and produces a single output. The inputs consist of a "set of weighted edges" [57]. Additionally, perceptrons take a separate input called bias, which is not weighted [56]. The processing required to produce the output value can be divided into two main steps. Firstly, the weighted sum of the inputs is

computed and the bias is added to the obtained value. Secondly, the result of the first step goes through an activation function, which produces the output [57][56].

When considering a single perceptron architecture, the output should always be boolean [56]. The activation function compares its received input to a threshold value. Then it produces one of two distinct values based on that comparison, signaling the perceptron's activation or deactivation [57][56]. This simple architecture can be used to represent linearly separable functions such as the AND, OR, NAND and NOR logic gates [56].

To learn higher complexity functions, multiple perceptrons with non-boolean activation functions can be used. In these architectures, the term neuron is used to denominate each processing unit instead. Examples of activation functions with continuous output would be the linear and *sigmoid* functions [57]. The linear function simply returns the input it receives. The sigmoid function is able to "squash" the input domain and produce an output ranging from 0 to 1 [56].

2.3.2 Architecture

The general architecture of a neural network consists of an input layer, one or more hidden layers and an output layer. The input layer is a set of nodes that merely hold the input data and don't perform any operations [39], unlike neurons. Each node is usually associated with one of the data's features. However, more than one node may be required to represent vector-like features. The input layer is connected to the neurons of the first hidden layer through weighted edges. Hidden layers are also connected via weighted edges both with each other and with the output layer.

We have by now discussed both the high-level structure of NNs and the functioning of their low-level units, called neurons. We should now seek to understand the learning process behind NNs, which is divided into two main stages: forward pass and backpropagation.

2.3.3 Forward pass

In a supervised learning setting, such as a classification or regression problem, each input example of the training data should be paired with its expected output. The features of the input example are given to the nodes in the input layer. These values are passed through the weighted edges to the neurons of the first hidden layer. Each of those neurons produces an output based on the sum of weighted inputs they received, a bias factor and their activation function, as explained previously. The produced outputs are then passed to the following hidden layer, which works in the same way. Eventually, the last hidden layer passes its outputs to the output layer.

The dimension of the output layer is based on the type of problem at hand. For example, in a regression problem where we wanted to predict a scalar variable, a single neuron in the output layer would suffice. The value of the output neuron after the forward pass would be the predicted value. In case of multiple target values, multiple output neurons should be used instead. Forecasting with neural networks could be implemented in this way. In classification problems, if we wanted to classify examples into one of three classes, three output neurons would be appropriate. The

expected outputs could be represented in a *one-hot* encoded way. The correct class would be associated with a value of 1 and the others with 0. After the forward pass, the values produced by the output neurons would represent the probability of the input example belonging to each class. The probabilities across output neurons should add up to 100%. A *softmax* activation function for the output neurons could be used to achieve this behaviour.

2.3.4 Backpropagation

We've just discussed how a forward pass is processed and some common ways in which output layers can be set up to support supervised learning tasks. What is still unexplained is how neural networks learn to correctly map inputs to outputs. The edges' weights are the learnable parameters that allow this to happen. The weights are assigned some initial (potentially random) values which are then updated according the network's learning algorithm.

The backpropagation algorithm is the most commonly used learning algorithm in neural networks. After the forward pass, the predictions are compared to the expected values through some error function. The global error of the network is computed by taking into account the individual error of all output neurons [56]. The derivative of the global error with respect to the weights connecting the last hidden layer to the output layer is calculated. This is called a gradient function. Those weights are then updated based on the gradient function and a learning rate constant. The weights are therefore changed in a direction that attempts to minimize their contribution to the global error. This is known as a gradient descent algorithm.

The calculations required to obtain the gradient function involve using the chain rule of derivatives. Before explaining them in further detail, let's first define the necessary variable names. A similar notation to the one presented in [56] will be used.

- E - global error in the considered forward pass
- h - hidden neuron
- o - output neuron
- y_n - activation value of neuron n
- s_n - sum of weighted inputs received by neuron n
- $W_{[n,m]}$ - weights connecting the previous layer to neuron m. n represents all the predecessors of m
- η - learning rate

$\Delta W_{[h,o]}$ is the delta to be added to the weights connecting the last hidden layer to an output neuron o. The formula to calculate this delta is the following [56]:

$$\Delta W_{[h,o]} = -\eta \frac{\partial E}{\partial W_{[h,o]}} \tag{2.1}$$

Using the chain rule of derivatives the formula is expanded to [56]:

$$\Delta W_{[h,o]} = -\eta \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial s_o} \frac{\partial s_o}{\partial W_{[h,o]}} \tag{2.2}$$

This delta is calculated for every output neuron and the respective weights are updated. However, only the weights connecting the last hidden layer to the output layer have been updated. To update the weights from previous layers, the error has to be propagated backwards. Considering the existence of only one hidden layer, we are now interested in the contribution of the weights between the input layer and the hidden layer to the global error. This requires some additional calculations, because we can not simply compare the hidden neuron's activation values with the final expected values. The following expression gives the delta to be added to the weights connecting the inputs to a given hidden neuron h [56]:

$$\Delta W_{[i,h]} = -\eta \sum_o \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial s_o} \frac{\partial s_o}{\partial y_h} \frac{\partial y_h}{\partial s_h} \frac{\partial s_h}{\partial W_{[i,h]}} \tag{2.3}$$

After all the weights are updated, this stage is completed. The network is ready to receive another training example and repeat the whole process.

2.3.5 Recurrent Neural Networks

Recurrent Neural Networks (RNN) use modified versions of the previously described neural network architecture, which is also known as the *feedforward* architecture. The difference resides in the recurrent connections which make RNNs more suitable to learn from sequential data, such as time series. The basic idea behind them is to not only consider the current input but also the network's hidden state produced at the previous time step [17]. Because of this, RNNs attempt to model and remember dependencies from the whole sequence they receive.

The introduction of these recursive mechanisms meant that the aforementioned backpropagation formulas were not directly applicable to RNNs. The most notable learning algorithms for RNNs are Backpropagation Through Time (BPTT) [59] and Real-Time Recurrent Learning (RTRL) [63]. The former, the most commonly used out of the two, is an adaptation of the backpropagation algorithm which essentially unfolds the network into a *feedforward* structure and performs the calculations from there [56]. However, there are major issues that severely hinder the ability of RNNs to learn long dependencies in time series. In fact, they are not effective at learning relationships in data more than 5 to 10 timesteps apart [56]. This is caused by the vanishing and exploding gradient problems, presented by Sepp Hochreiter in 1991 [30]. Basically, the error signals computed during backpropagation may become too small or increasingly big with every iteration. In the case of vanishing gradients, the weight updates become insignificant, and the learning therefore stagnates. With exploding gradients, the weight updates become unstable, which also leads to bad results.

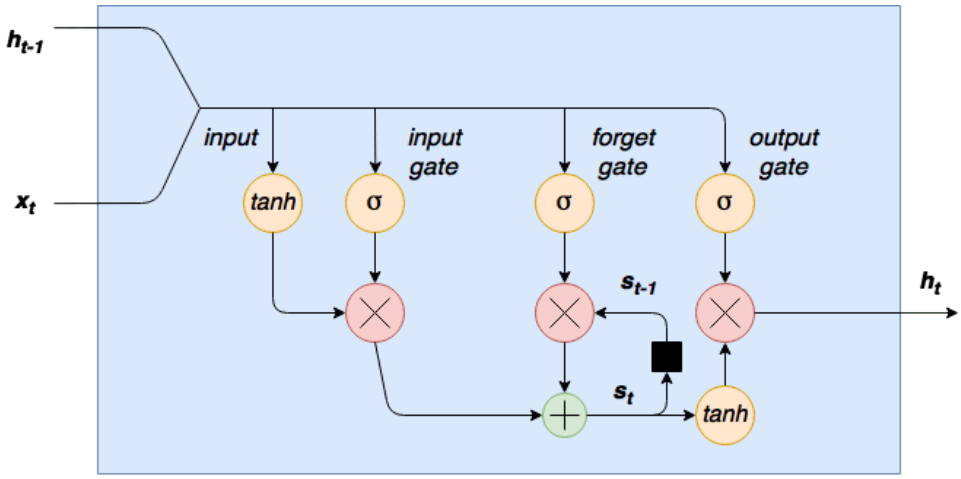


Figure 2.1: Architecture of an LSTM block.

Source: http://adventuresinmachinelearning.com/keras-lstm-tutorial/ (accessed at 10/02/2020)

2.3.6 Long Short-Term Memory

In 1997, a solution for this problem was proposed in the form of a novel RNN architecture denominated "Long Short-Term Memory" (LSTM) [31]. Alongside this architecture, a modified gradient descent algorithm was also introduced. The main difference of this learning algorithm from BPTT and RTRL is that the error flow is enforced to be constant throughout the network's internal states [31]. The breakthroughs, both in architecture and learning algorithm, allow LSTMs to learn relationships in data that could be as distant as 1000 timesteps [31].

The LSTM architecture is composed of blocks[1], each of them containing an input gate, output gate and memory cell. A gate is set of processing units with *sigmoid* activation functions whose inputs are scaled by weights. There is also an input modulation gate (hereby reffered to as modulation gate) which contains a *tanh* activation function instead. The memory cell holds hidden information resulting from previous inputs. The modulation gate learns to scale input features according to their importance and then squashes them between -1 and 1. Moreover, the purpose of the input gate is to regulate which of those features should used to updated the cell's state. The output gate filters which information from the cell should be passed to the next block. A forget gate was later proposed [26] to learn which data should be discarded from memory even before updating it with new information from the current input.

The learning mechanism of an LSTM is also divided into a forward and backward pass. These stages will be explained in accordance to a Python implementation of the LSTM architecture [2]. Furthermore, multiple papers [17, 26, 56, 51] with detailed explanations on this matter were also consulted.

[1] What is denoted here as *block* is often reffered to as LSTM *cell* in the literature. The term block is adopted to avoid confusion with the memory cell. Also, in the literature, the processing of a new input sample corresponds to a new time step. Here, however, a new input sample is associated with a new block. The term time step is reserved to address each value in a time series.

2.3.6.1 Forward pass

In the forward pass, the current input is concatenated with the previous hidden output and a bias factor in the same vector, resulting in the hidden input. The hidden input is transported to all the gates, being scaled by their weights and subjected to their activation functions. The cell state is calculated by adding two parcels. The first is the product of the modulation gate's activation and the input gate's activation. The second parcel is the product of the previous cell state and the forget gate's activation. To obtain the hidden output, the cell state is first squashed by the *tanh* function and then multiplied by the output gate's activation. Finally, passing the hidden output through an output layer of weights produces the predicted values. Those can be compared to the expected outputs to obtain an error measure. Both the cell state and the hidden output are passed to the next block, which will receive a new input sample and repeat the process.

A summary of the variables and expressions used in the forward pass is presented. The t subscripts indicate the LSTM blocks which the variables refer to.

- X - values in the input sample
- b - bias
- hin - hidden input
- $hout$ - hidden output
- $pred$ - predicted values
- $true$ - groundtruth values
- $cell$ - memory cell state
- W_i - weights of the input gate
- W_f - weights of the forget gate
- W_o - weights of the output gate
- W_g - weights of the modulation gate
- W_z - weights of the output layer
- i - activation of the input gate
- f - activation of the forget gate
- o - activation of the output gate
- g - activation of the modulation gate

$$hin_t = \{X_t, hout_{t-1}, b\} \tag{2.4}$$

$$\begin{aligned} W_i &= \{W_{xi}, W_{hi}, W_{bi}\} \\ W_f &= \{W_{xf}, W_{hf}, W_{bf}\} \\ W_o &= \{W_{xo}, W_{ho}, W_{bo}\} \\ W_g &= \{W_{xg}, W_{hg}, W_{bg}\} \end{aligned} \tag{2.5}$$

$$\begin{aligned} i_t &= \sigma(hin_t W_i) \\ f_t &= \sigma(hin_t W_f) \\ o_t &= \sigma(hin_t W_o) \\ g_t &= \tanh(hin_t W_g) \end{aligned} \tag{2.6}$$

$$cell_t = g_t i_t + cell_{t-1} f_t \tag{2.7}$$

$$hout_t = \tanh(cell_t) o_t \tag{2.8}$$

$$pred_t = hout_t W_z \tag{2.9}$$

2.3.6.2 Backward pass

There is an important parameter - the *batch size* - to decide how many input samples should be processed before starting the backward pass. In other words, the *batch size* determines the amount of blocks in the LSTM architecture. Another thing to keep in mind is that, during the forward pass, all weights are shared between blocks.

The goal of the backward pass is to minimize the sum of errors obtained across all blocks during the forward pass:

$$\begin{aligned} &\min L \\ &L = \sum_t Error_t \end{aligned} \tag{2.10}$$

For that purpose, an error function must be defined. We consider the following function [17], where the error at block t is:

$$Error_t = \frac{1}{2}(pred_t - true_t)^2 \tag{2.11}$$

It should be noted that *pred* and *true* are vectors (and not scalars) which contain, respectively, the predicted and expected values at a certain block. Therefore, the resulting *Error* is also a vector of the same length. Hence, the product operations in the formulas below are implied to be *point-wise* multiplications.

With an error measure defined, the first objective is to obtain the gradient responsible for modifying the output layer's weights. At each block t, we derive the error with respect to the output layer's weights by resorting to the chain rule of derivatives:

$$\frac{\mathrm{d}L}{\mathrm{d}W_z} = \sum_t \frac{\partial Error_t}{\partial W_z} = \sum_t \frac{\partial Error_t}{\partial pred_t} \frac{\partial pred_t}{\partial W_z} \tag{2.12}$$

$$\frac{\partial Error_t}{\partial pred_t} = pred_t - true_t \tag{2.13}$$

$$\frac{\partial pred_t}{\partial W_z} = hout_t \tag{2.14}$$

The next objective is to calculate the derivative of the total error with respect to the output gate's activation at each block. In the intermediate steps, we must take into consideration that the hidden outputs affect the global error both at the block they are calculated in and at the next block, since they are a part of their hidden input (Eq. 2.4). Equation 2.17 is a special case of the global error derivative with respect to the hidden output that is only applicable at the last block T, while Eq. 2.20 expresses that derivative at any block but the last.

$$\frac{\partial L}{\partial o_t} = \frac{\partial L}{\partial hout_t}\frac{\partial hout_t}{\partial o_t} \tag{2.15}$$

$$\frac{\partial hout_t}{\partial o_t} = \tanh(cell_t) \tag{2.16}$$

$$\frac{\partial L}{\partial hout_T} = \frac{\partial Error_T}{\partial hout_T} = \frac{\partial Error_T}{\partial pred_T}\frac{\partial pred_T}{\partial hout_T} \tag{2.17}$$

$$\frac{\partial Error_T}{\partial pred_T} = pred_T - true_T \tag{2.18}$$

$$\frac{\partial pred_T}{\partial hout_T} = W_z \tag{2.19}$$

$$\frac{\partial L}{\partial hout_{t-1}} = \frac{\partial Error_{t-1}}{\partial hout_{t-1}} + \frac{\partial Error_t}{\partial hout_{t-1}} \tag{2.20}$$

$$\frac{\partial Error_{t-1}}{\partial hout_{t-1}} = (pred_{t-1} - true_{t-1})W_z \tag{2.21}$$

$$\frac{\partial Error_t}{\partial hout_{t-1}} = \frac{\partial Error_t}{\partial hin_t}[hout_{t-1}] \tag{2.22}$$

What is meant in equation 2.22 is that, since $hout_{t-1}$ is a part of the hin_t vector (Eq. 2.4), we can subset $\frac{\partial Error_t}{\partial hin_t}$ to just the part that refers to $hout_{t-1}$. The calculations required to obtain $\frac{\partial Error_t}{\partial hin_t}$ will be displayed in Chapter 3 since that derivative will also be used in our proposed method.

Moving on, the derivative of the total error with respect to the cell state at each block is calculated as follows:

$$\frac{\partial L}{\partial cell_{t-1}} = \frac{\partial Error_{t-1}}{\partial cell_{t-1}} + \frac{\partial Error_t}{\partial cell_{t-1}} \tag{2.23}$$

$$\frac{\partial Error_{t-1}}{\partial cell_{t-1}} = \frac{\partial Error_{t-1}}{\partial \tanh(cell_{t-1})}\frac{\partial \tanh(cell_{t-1})}{\partial cell_{t-1}} \tag{2.24}$$

$$\frac{\partial \tanh(cell_{t-1})}{\partial cell_{t-1}} = \text{sech}^2(cell_{t-1}) = 1 - \tanh^2(cell_{t-1}) \tag{2.25}$$

$$\frac{\partial Error_{t-1}}{\partial \tanh(cell_{t-1})} = \frac{\partial Error_{t-1}}{\partial hout_{t-1}} \frac{\partial hout_{t-1}}{\partial \tanh(cell_{t-1})} \tag{2.26}$$

$$\frac{\partial Error_{t-1}}{\partial hout_{t-1}} = (pred_{t-1} - true_{t-1})W_z \tag{2.27}$$

$$\frac{\partial hout_{t-1}}{\partial \tanh(cell_{t-1})} = o_{t-1} \tag{2.28}$$

$$\frac{\partial Error_t}{\partial cell_{t-1}} = \frac{\partial Error_t}{\partial cell_t} \frac{\partial cell_t}{\partial cell_{t-1}} \tag{2.29}$$

$$\frac{\partial cell_t}{\partial cell_{t-1}} = f_t \tag{2.30}$$

The derivatives of the total error with respect to the input, forget and modulation gates' activation at each block are simpler to obtain:

$$\frac{\partial L}{\partial i_t} = \frac{\partial Error_t}{\partial i_t} = \frac{\partial Error_t}{\partial cell_t} \frac{\partial cell_t}{\partial i_t} \tag{2.31}$$

$$\frac{\partial L}{\partial f_t} = \frac{\partial Error_t}{\partial f_t} = \frac{\partial Error_t}{\partial cell_t} \frac{\partial cell_t}{\partial f_t} \tag{2.32}$$

$$\frac{\partial L}{\partial g_t} = \frac{\partial Error_t}{\partial g_t} = \frac{\partial Error_t}{\partial cell_t} \frac{\partial cell_t}{\partial g_t} \tag{2.33}$$

$$\frac{\partial Error_t}{\partial cell_t} = (pred_t - true_t)W_z o_t(1 - \tanh^2(cell_t)) \tag{2.34}$$

$$\frac{\partial cell_t}{\partial i_t} = g_t \tag{2.35}$$

$$\frac{\partial cell}{\partial f} = cell_{t-1} \tag{2.36}$$

$$\frac{\partial cell_t}{\partial g_t} = i_t \tag{2.37}$$

The formulas presented until this point allow us to calculate the derivatives of the global error with respect to each gate's activation at any given block t ($\frac{\partial L}{\partial i_t}$, $\frac{\partial L}{\partial f_t}$, $\frac{\partial L}{\partial o_t}$, $\frac{\partial L}{\partial g_t}$). However, we are ultimately looking to derive the global error with respect to the gates' weights, so that we can

finally update them. Those expressions are followingly derived.

$$\frac{\mathrm{d}L}{\mathrm{d}W_k} = \sum_t \frac{\partial L}{\partial k_t}\frac{\partial k_t}{\partial W_k} \tag{2.38}$$

$$\frac{\partial k_t}{\partial W_k} = \frac{\partial k_t}{\partial (hin_t W_k)}\frac{\partial (hin_t W_k)}{\partial W_k} \tag{2.39}$$

$$\frac{\partial (hin_t W_k)}{\partial W_k} = hin_t \tag{2.40}$$

$$k \in \{i, f, o, g\} \tag{2.41}$$

The generic equations 2.38 to 2.40 above are common to all gates. However, the term $\frac{\partial k_t}{\partial (hin_t W_k)}$ in Eq. 2.39 is calculated differently in gates with either *sigmoid* or *tanh* activation functions. For the input, forget and output gates, that expression is the following:

$$\frac{\partial j_t}{\partial (hin_t W_j)} = \frac{\partial \sigma(hin_t W_j)}{\partial (hin_t W_j)} = \sigma(hin_t W_j)(1 - \sigma(hin_t W_j)) = j_t(1 - j_t) \tag{2.42}$$

$$j \in \{i, f, o\} \tag{2.43}$$

For the modulation gate, the expression is calculated as follows:

$$\frac{\partial g_t}{\partial (hin_t W_g)} = \frac{\partial \tanh(hin_t W_g)}{\partial hin_t W_g} = \mathrm{sech}^2(hin_t W_g) = 1 - \tanh^2(hin_t W_g) \tag{2.44}$$

We have now calculated all the terms required to obtain $\frac{\mathrm{d}L}{\mathrm{d}W_i}$, $\frac{\mathrm{d}L}{\mathrm{d}W_f}$, $\frac{\mathrm{d}L}{\mathrm{d}W_o}$, $\frac{\mathrm{d}L}{\mathrm{d}W_g}$ and also $\frac{\mathrm{d}L}{\mathrm{d}W_z}$ from earlier (Eq. 2.12). These expressions can simply be multiplied by a negative learning rate constant to finally obtain the deltas that should be added to the respective weights.

$$\Delta W_u = -\eta \frac{\mathrm{d}L}{\mathrm{d}W_u} \tag{2.45}$$

$$u \in \{i, f, o, g, z\} \tag{2.46}$$

This concludes the backward pass.

Chapter 3

Pastprop

We propose the idea of reworking the backpropagation algorithm so that error responsibility is extended to data. As such, similarly to the network weights, input data should be corrected in the direction that minimizes error. The magnitude of the changes should be proportional to the contribution of data to that error. Their purpose is to improve the overall quality of the training data. In particular, we expect them to be effective at reconstructing anomalies. The premise is that anomalies can be viewed as sections of data with high responsibility for errors. Therefore, they should theoretically be subjected to more significant changes. Since corrections are made while the LSTM is learning to model its weights, they have an effect on this modeling process. At the end of learning, Pastprop has two outputs: the weights of the network; a different version of the training data it received as input.

The issue that arises from the mismatch between sales and demand, mentioned in Section 1.2, is an example of a practical situation where Pastprop could receive application. However, it should be stressed that the method is generic and its goal is not to target that specific problem. Any time series can be given to Pastprop - even if it is not clearly affected by anomalies - and the corrections applied to it could still be advantageous while adjusting the forecasting model (the network weights). Moreover, the corrected time series outputted by Pastprop at the end of learning could also be a valuable asset. They can be provided as training data to other forecasting algorithms, which might be beneficial in comparison to the original versions of data.

The idea of Pastprop could potentially be applied to other neural network architectures, provided they also learn through backpropagation. However, our work is focused on LSTMs only since they are well suited for time series forecasting - our task of interest.

3.1 Calculating deltas

The key variables required to implement *Pastprop* are the deltas to be added to the time series. In a similar manner to how the gradients responsible for updating the weights are obtained in the

backward pass, we can also obtain a gradient for the LSTM's hidden input. At each block t in the LSTM, we consider how much the hidden input contributed for the error at that block (not the global error of the forward pass). This is the expression we are looking for:

$$\frac{\partial Error_t}{\partial hin_t}$$

Since the hidden input is a vector composed of the input sample, previous hidden output and a bias value (Eq. 2.4), we subset the gradient to just the part that respects to the input sample.

$$\frac{\partial Error_t}{\partial hin_t}[X_t]$$

The deltas to be added to the input samples are calculated from this gradient and the *data correction rate* constant, a Pastprop specific hyperparameter which serves as a learning rate for data instead of weights.

The following formulas describe how the derivative of error with respect to the hidden input can be calculated at any block t.

$$\begin{aligned}\frac{\partial Error_t}{\partial hin_t} = &\frac{\partial Error_t}{\partial (hin_t W_i)}\frac{\partial (hin_t W_i)}{\partial hin_t} \\ &+\frac{\partial Error_t}{\partial (hin_t W_f)}\frac{\partial (hin_t W_f)}{\partial hin_t} \\ &+\frac{\partial Error_t}{\partial (hin_t W_o)}\frac{\partial (hin_t W_o)}{\partial hin_t} \\ &+\frac{\partial Error_t}{\partial (hin_t W_g)}\frac{\partial (hin_t W_g)}{\partial hin_t}\end{aligned} \tag{3.1}$$

$$\frac{\partial Error_t}{\partial (hin_t W_i)} = \frac{\partial Error_t}{\partial i_t}\frac{\partial i_t}{\partial (hin_t W_i)} \tag{3.2}$$

$$\frac{\partial Error_t}{\partial (hin_t W_f)} = \frac{\partial Error_t}{\partial f_t}\frac{\partial f_t}{\partial (hin_t W_f)} \tag{3.3}$$

$$\frac{\partial Error_t}{\partial (hin_t W_o)} = \frac{\partial Error_t}{\partial o_t}\frac{\partial o_t}{\partial (hin_t W_o)} \tag{3.4}$$

$$\frac{\partial Error_t}{\partial (hin_t W_g)} = \frac{\partial Error_t}{\partial g_t}\frac{\partial g_t}{\partial (hin_t W_g)} \tag{3.5}$$

The derivatives $\frac{\partial i_t}{\partial (hin_t W_i)}$, $\frac{\partial f_t}{\partial (hin_t W_f)}$, $\frac{\partial o_t}{\partial (hin_t W_o)}$, $\frac{\partial g_t}{\partial (hin_t W_g)}$ were presented in equations 2.42 and 2.44. $\frac{\partial Error_t}{\partial i_t}$, $\frac{\partial Error_t}{\partial f_t}$, $\frac{\partial Error_t}{\partial g_t}$ can be obtained from equations 2.31 to 2.33, while $\frac{\partial Error_t}{\partial o_t}$ is now calculated:

$$\frac{\partial Error_t}{\partial o_t} = \frac{\partial Error_t}{\partial hout_t}\frac{\partial hout_t}{\partial o_t} = (pred_t - true_t) W_z \tanh(cell_t) \tag{3.6}$$

Finally, the only missing derivatives are trivially obtained:

$$\frac{\partial(hin_t W_i)}{\partial hin_t} = W_i \tag{3.7}$$

$$\frac{\partial(hin_t W_f)}{\partial hin_t} = W_f \tag{3.8}$$

$$\frac{\partial(hin_t W_o)}{\partial hin_t} = W_o \tag{3.9}$$

$$\frac{\partial(hin_t W_g)}{\partial hin_t} = W_g \tag{3.10}$$

3.2 Pastprop variants

The following subsections explain the functioning of three different Pastprop versions. Their implementations were supported by publicly available Python code regarding the LSTM architecture [2, 1]. For the purpose of easily distinguishing between normal LSTMs and LSTMs that include the Pastprop mechanism, we will simply be referring to these forecasting methods as LSTM and Pastprop, respectively. Also for the purpose of simplicity in explanation, it is assumed that Pastprop receives one univariate time series as training data. However, this is not a requirement of the method. Furthermore, Pastprop is categorized as a method of supervised learning, so it works with input samples and groundtruth labels. It will be assumed that those sequences always contain more than one value. During learning, the next input sample to be processed occurs just one time step later than the previous sample. At the end of training, the outputs of Pastprop are both the network's weights and the final corrected time series.

3.2.1 Regular Pastprop

In this version, corrections are applied to the whole training series - all at once - after each epoch has been completed. The deltas to be added to each data sample are calculated and stored until the end of the epoch. Right before starting the new epoch, the deltas and time series are added together. This means that every epoch deals with a different version of the data. The first epoch happens to work with the unaltered data. Therefore, if only one epoch was to be considered, Regular Pastprop would produce exactly the same weights as a normal LSTM (in case the initial weights were the same).

Figure 3.1 illustrates the described mechanism in a simplified way. We can see that most time steps have overlapping deltas associated to them. In those cases, an average of the corresponding deltas (not their sum) is used to build the corrections array. The average is preferred over the sum to make the amount of data corrections less dependent on the sample size. For example, in figure 3.1, the value -0.09 at time step 2 of the corrections array is the result of averaging -0.09, -0.08 and -0.10.

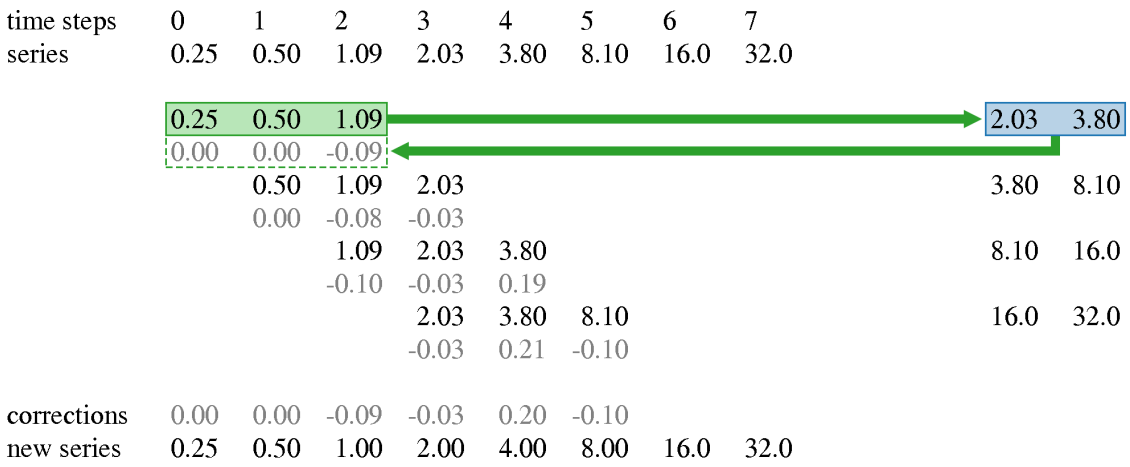


Figure 3.1: Simplified illustration of how Regular Pastprop works. The green box is a sample and the blue box is the respective label. The green dashed box contains the deltas for that sample.

3.2.2 Progressive Pastprop

This version differs from the Regular one in the sense that deltas are added to data over the course of epochs. As soon as they're obtained, deltas are first adjusted to compensate for overlapping and then added to samples. For example, if a time series value at a certain time step expects to be modified by 3 deltas over the course of an epoch, each of those deltas should be divided by 3 before addition. During learning, each sample takes into account the corrections made until that point to the series. Figure 3.2 illustrates these mechanisms. Contrarily to Regular Pastprop, in this version each epoch deals with progressively different time series. For this reason, the weights differ from those produced by an LSTM right from the first epoch.

3.2.3 Selective Pastprop

This variant is built upon Regular Pastprop and introduces two new features. The first is an hyperparameter that dictates the amount of initial epochs to wait before applying data corrections. Assuming the number of *waiting epochs* is n, the time series is only changed after the nth epoch is complete. The second feature is a thresholding mechanism. At the end of an epoch, when the corrections array is ready, the deltas in the array are ranked by the average of theirs and their neighbours' absolute value. Higher values translate to higher ranks. The highest ranked deltas prevail while the other ones are changed to zeros (which means they will have no effect on data). A threshold parameter is used to establish the amount of deltas to be preserved. Furthermore, the number of previous and subsequent time steps that define a delta's neighborhood is also required as a parameter. In case that number is zero, each delta is ranked solely by its own absolute value.

Keeping the time series unchanged for the first few epochs is an attempt at reducing the impact of the random weight initialization on data corrections. The calculation of deltas is dependent on weights, as evidenced in Section 3.1. This implies that deltas obtained early in the first epoch are

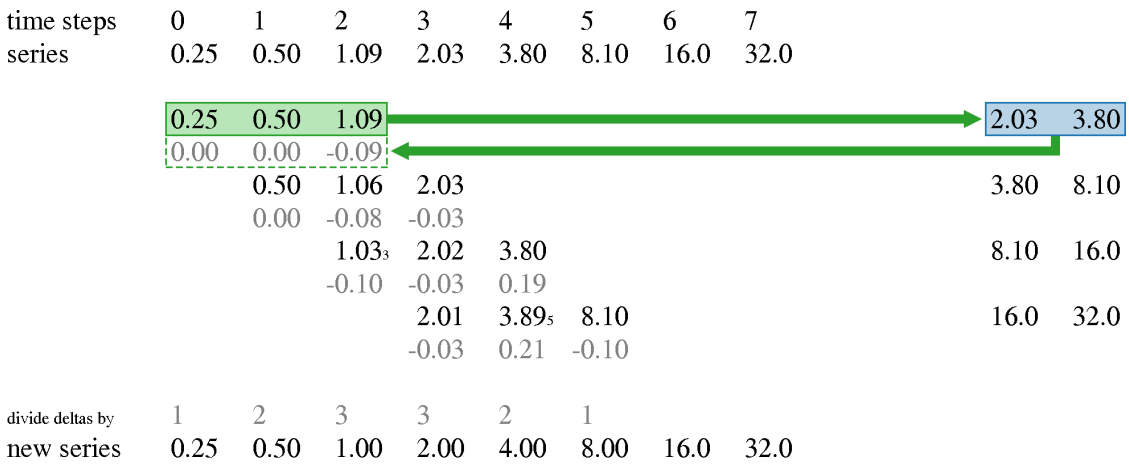


Figure 3.2: Simplified illustration of how Progressive Pastprop works. The green box is a sample and the blue box is the respective label. The green dashed box contains the deltas for that sample. The deltas are divided by the corresponding factor at the bottom.

partially based on close to random weights. Waiting for the completion of some initial epochs before applying corrections makes it more likely for the weights to reach a meaningful state.

The rationale behind the thresholding is to avoid unnecessary changes to data. Maintaining only the most significant deltas from the corrections array is an effort to achieve this. Moreover, the inclusion of neighbor deltas in the ranking system incentivizes the preservation of deltas associated with anomalous zones. In turn, single point anomalies are not as targeted by the system.

3.3 Important notes

A rule that applies to all Pastprop variants is that the groundtruth labels completely ignore all data corrections. They are always built from the original, unaltered time series. Hence, the targets based on which we would like to update weights and correct data are kept the same throughout the learning process.

Another thing is that the time series given to Pastprop for training cannot be corrected at the last few time steps. This is because the last training pair of input sample to groundtruth label uses the time series' last x time steps for the label, where x would be the *label size*. Therefore that part is not eligible for corrections, since deltas are always associated with input samples.

Regarding multivariate time series, the Pastprop implementations were programmed to only correct the main variable being forecasted and not the explanatory time series. Even though this was decided, it would be possible to do so and it could be interesting to study that in further work.

One last note is that the *batch size* has no direct influence on the amount of corrections. The deltas for each input sample are obtained from the gradient of the error at predicting their respective label. Therefore, no matter the number of input samples that are processed before starting a backward pass, the deltas for each input sample are always calculated individually. What does

change with the increase of the *batch size* are the updates to weights, which will happen less frequently. Despite that, all errors are still accounted for when updating weights, just in an aggregated instead of individual manner.

3.4 Hyperparameters

Finally, an overview of all the hyperparameters required for the Pastprop methods is presented. The LSTM related hyperparameters are the following:

- *sample size*: number of time steps considered in each sample
- *label size*: number of time steps considered in each label
- *hidden size*: number of processing units in each gate and output layer
- *batch size*: number of input samples to process in a forward pass before doing a backward pass (equal to the number of blocks in the LSTM architecture)
- *learning rate*: constant to regulate weight changes
- *epochs*: number of epochs to train the network for

The only Pastprop specific hyperparameter that is common to all implementations is the *data correction rate*, a constant to regulate data changes. The Selective version also needs the:

- *waiting epochs*: number of initial epochs to wait before applying corrections
- *threshold*: number of highest ranked deltas to keep on the corrections array
- *neighbor size*: number of previous and subsequent timesteps to consider when ranking deltas

Chapter 4

Experimental setup

This chapter addresses the setup that enabled our empirical study. We discuss the datasets that supported it. Basic characteristics of each time series such as their size and domain are presented, as well as the preprocessing steps that were applied to them. For each dataset, the hyperparameters that were tested in the corresponding experiments are revealed. The methodology behind the study and the metrics considered in our results are also explained.

Anomalies are important variables in our study. We start by explaining why and how they were inserted into data.

4.1 Anomaly generation

The goal of Pastprop is to improve the quality of time series, namely by correcting "incorrect" or "unuseful" data. There were no available datasets with clear information on which values were incorrect or not. Therefore, in a set of experiments, we systematically manipulated data so that we could test the method's ability to reconstruct back to the original series. Even though this manipulation is not entirely realistic, it allows us to understand the behaviour of the method better.

The properties considered in the generation of anomalies are listed below:

- *size*: number of timesteps that constitute the anomaly
- *magnitude*: indicates how different the anomaly is from data
- *position*: where the anomaly occurs in data

Magnitude Three different levels of magnitude were considered. The first, level 0, simply substitutes the data by a sequence of zeros. The other two levels function in a different way. The anomaly zone is divided into equally sized chunks. It is randomly chosen whether each chunk should add or subtract from the original data. The value of change at a given time step is the maximum between 0.1 and a percentage of the time series' original value. The possible percentages are 25 and 50 which correspond to the magnitude levels 25 and 50. The minimum absolute change value of 0.1, although arbitrary, was chosen in the context of data being normalized between 0 and 1. Figure 4.1 exemplifies how each of the three possible levels of magnitude may look like.

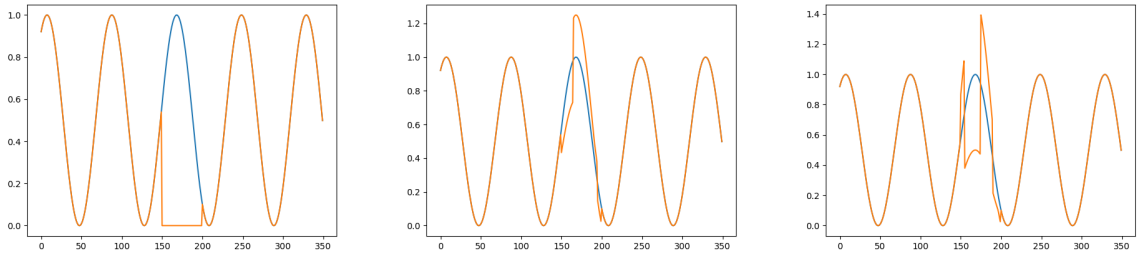


Figure 4.1: Examples of anomalies with different levels of magnitude. From left to right: level 0, level 25 and level 50. The blue lines are the original series and the orange lines are the anomalous series. Please note the different graph scales.

4.2 Datasets

The research was performed on 10 different time series, 3 of which were multivariate. The following subsections provide a description of how the data were obtained and what hyperparameters were tested with them. A few variables were kept constant throughout all experiments. For all presented datasets, the training data consisted of the first 70% of the time series and the last 30% were used for testing. Furthermore, the learning rate was always 1e−3 and the number of hidden units was always 20. Lastly, all series were normalized between 0 and 1.

4.2.1 Artificial data

The only artificially generated time series is a sinusoidal function with a total length of 500 time steps. The experimentation on this particular series was the most extensive in terms of exploring hyperparameter combinations and their effect on results. The following is the list of variables of interest and the possible values they were studied with:

- *epochs*: 50, 200, 1000 (respective *waiting epochs* were 10, 20, 50)
- *data correction rate*: 1e−3, 1e−2, 1e−1
- *anomaly size*: 25, 50, 100 time steps
- *anomaly position*: 5 different positions evenly spaced throughout the training data plus non-occurrence of anomaly
- *anomaly magnitude*: level 0, level 25, level 50
- *sample size*: 50 timesteps
- *label size*: 50 timesteps

Considering all possible combinations of these parameters and the definition of a single experiment presented in the Methodology Section (4.3), a total amount of 486 experiments were executed on the artificial data.

4.2.2 Benchmark data

A total of 7 benchmark time series were used, originated from 3 different publicly available datasets.

The M5 competition dataset [4] was processed with the goal of extracting 3 time series, each representing the daily unit sales of a specific product category (Hobbies, Foods and Household) registered in Walmart stores. The data was collected in the USA from January 2011 up to April 2016, resulting in 1913 consecutive days of unit sales information [4].

The next dataset was obtained from the UCI Machine Learning Repository [5]. It contains electricity consumption data of 370 different clients over a maximum period of 4 years, from January 1st of 2011 to January 1st of 2015. The consumption values were registered every 15 minutes in kilowatts (kW). The dataset was then aggregated by total daily electricity consumption. After that, 3 arbitrary clients were chosen from those whose data was available for the whole 4 year period. That left us with 3 time series with a length of 1462 time steps.

The last benchmark dataset used in the research was also gathered from the UCI Machine Learning Repository [5]. The dataset stores hourly traffic volumes data obtained from the Minnesota Department of Transportation from October 2012 up to September 2018. Besides that, weather information from OpenWeatherMap is available for the same period. A time series consisting of 1199 daily traffic volumes starting from June 2015 was extracted. Simultaneously, parallel time series such as the daily average temperature and the daily average percentage of cloud coverage were also extracted. Moreover, 6 binary time series were computed to encode the week day associated with each timestep (only 6 series are needed since a value of 0 in all of them encodes the seventh day). Yet another binary series marking the occurrence of locally celebrated holidays was used. Finally, two more auxiliary time series were created. One of them indicated if the month associated with the timestep was historically positively or negatively correlated with traffic volume. For example, because June had a positive 10% correlation with traffic volume, all observations during that month were marked with a 1. On the contrary, a value of 0 was attributed to all observations that happened in January, since it presented a negative 11% correlation. The same procedure was replicated for all months and the resulting auxiliary time series showed a positive correlation of 21% with traffic volume. This strategy was used to avoid the need of one extra time series per each month, therefore reducing the dimensionality of the data involved while still capturing some information regarding the effect of months on the main variable. The last explanatory time series was computed in a similar way but relative to the description of weather conditions at a given timestep, rather than the month at which the values were observed.

For all the benchmark data, the values tested for each hyperparameter were the following:

- *epochs*: 50, 200, 1000 (respective *waiting epochs* were 10, 20, 50)
- *data correction rate*: 1e−3, 1e−2, 1e−1
- *anomaly size*: 56 time steps
- *anomaly position*: middle of training data and non occurence of anomaly
- *anomaly magnitude*: level 0, level 50

- *sample size*: 28 timesteps
- *label size*: 28 timesteps

4.2.3 Retail case study data

We also used data from an international fashion retail case study that was provided by Inovretail. In order to protect the privacy of their client, the results of the experiments made on these data were aggregated with results related to other benchmark datasets.

Two main time series were considered, each consisting of daily unit sales of a popular article during a full year period. Along side with this data, auxiliary time series were used such as the daily stock levels, average temperature at the store's location and precipitation amounts. Similarly to before, 6 binary time series were used to represent the days of the week and another series indicated if each month was historically good or bad for sales.

The hyperparameters of the experiments done with these data were similar to the ones used for benchmark data. The only differences were in the sample and label sizes (14 time steps) and in the anomaly size (28 time steps).

4.3 Methodology

For a given time series and complete set of parameters, a single experiment consisted of performing multiple operations. The first step was to generate random initial weights. For the sake of fairness in comparison, all tested methods worked with these same starting weights. The methods were a simple LSTM, all three Pastprop variants and also three additional LSTMs. The latter made use of the series corrected by the Pastprop methods as training data. After execution, besides forecasting accuracies on test data, a few other results were registered. These included similarity measures between each corrected series and the original data, both on the anomaly zone (*reconstruction ability*) and outside of it (*outside loss*). With robustness in mind, the whole experimental procedure was repeated 5 times. This implies different weight initialization in each of the 5 runs. The final results attributed to a combination of time series and complete set of parameters were the average results of those runs.

4.3.1 Baseline algorithms

The procedure just described was the main methodology developed to conduct our study. However, additional experiments were needed to study all research questions. Other algorithms, traditionally used in forecasting tasks, were also considered:

- Autoregressive integrated moving average (ARIMA)
- Exponential smoothing state space model (ETS)
- Exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend and Seasonal components (BATS)

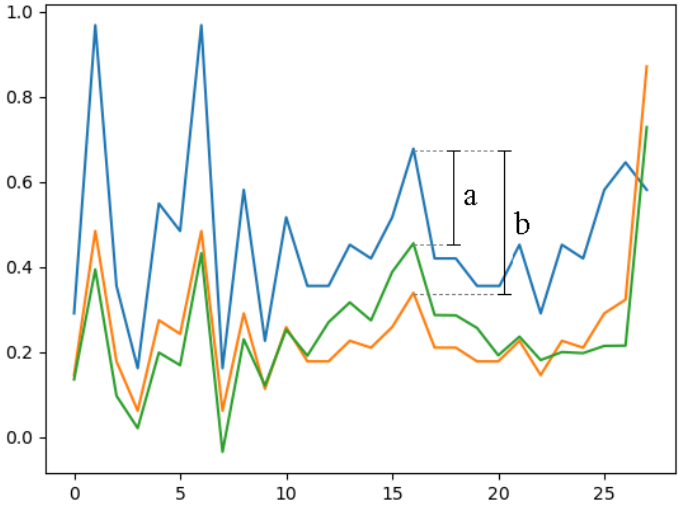


Figure 4.2: Example to visualize anomaly reconstruction. Blue is the original series, orange is the anomaly and green is the corrected series. The symbol *a* illustrates the difference between the corrections and the original data; *b* illustrates the difference between the anomaly and the original data.

The methods were executed by resorting to the R *forecast* package [3]. The respective functions used for ARIMA, ETS and BATS were *auto.arima*, *ets* and *bats*.

These baseline algorithms served two purposes in the empirical study: to act as references for the accuracies achieved by LSTM and Pastprop; to test if the corrected series outputted by Pastprop could benefit their predictive ability in comparison to the original series.

4.3.2 Reconstruction Ability and Outside Loss

The anomaly reconstruction ability was calculated by resorting to two variables: the MSE between the corrections and the original data; the MSE between the anomaly and the original data. The formula is obtained as follows:

$$\text{reconstruction ability} = 1 - \frac{\text{MSE}(corrections, original)}{\text{MSE}(anomaly, original)} \tag{4.1}$$

As such, a positive reconstruction ability indicates the corrections were able to push the anomalous zone closer to the original data. On the other hand, a negative value means the anomaly was accentuated even further.

The MSE between corrections and original data was also measured while excluding the anomalous part. This will be referred to as the "outside loss". Especially on the artificial data, where the series follows a well defined curve, there is no apparent reason to change data outside of the anomaly. Therefore we want this value to be as low as possible. When considering other types of data, this measure loses some of its meaning since corrections may be beneficial even outside of the anomaly.

Finally, both of theses measures - “reconstruction ability” and “outside loss” - can only be calculated in series that include manually inserted anomalies.

Chapter 5

Results

The results of our empirical study are showcased and discussed in this chapter. The analysis is organized in terms of the research questions and the types of data used in the experiments.

5.1 Reconstructing the past

In this section we investigate the research question: *How effective is Pastprop at reconstructing anomalies in data?*

5.1.1 Artificial data

5.1.1.1 Proof of concept

One experiment apart from the main empirical study was designed specifically to assess the ability of Pastprop to reconstruct an anomaly when given an highly favorable scenario to do so. This should give us an idea if the method is working as intended and if it has potential. The experiment was done using Selective Pastprop on the sinusoidal series with an anomaly occurring at the beginning of data. The difference from this experiment to other experiments on the same data is that the anomaly consists of a sequence of 0.5 values, which is roughly the average value of the original series in that zone. This makes it easier for the amount of corrections applied to reach the shape of the original series. The hyperparameters considered in the experiment are listed in figure 5.1, which displays the resulting anomaly reconstruction ability of 83.9%. The MSE value of the outside loss was of 0.01186. These results confirm that the deltas through which Pastprop is implemented have the effect on data that we expected them to have. Additionally, we can safely say the method has potential when it comes to reconstructing anomalies.

5.1.1.2 Main experiments

When considering only anomalies that occur at the middle of time series, the global results on artificial data indicate a positive reconstruction ability of roughly 10% when using the Selective Pastprop version. Meanwhile, the Regular and Progressive versions showed negative abilities of

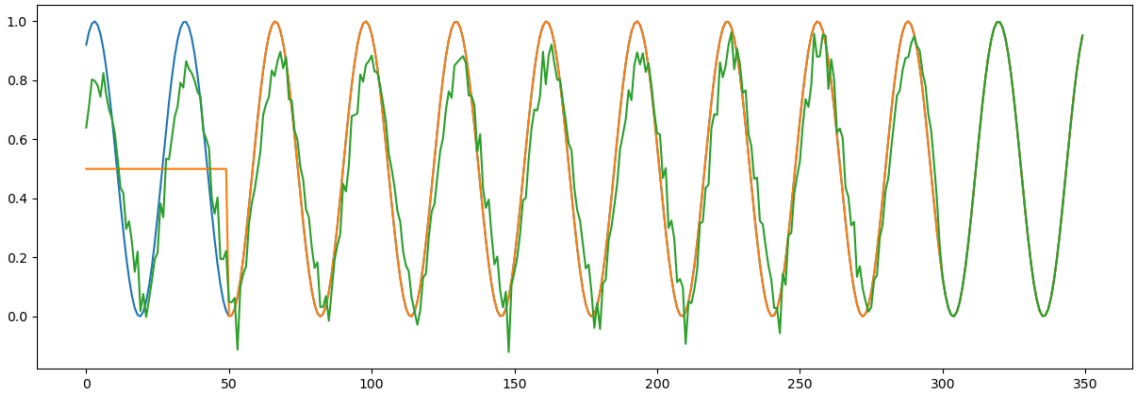


Figure 5.1: Proof of concept: Selective Pastprop on the sinusoidal series trained for 1000 epochs (50 waiting epochs) with a data correction rate of 1e−1. Blue is the original series, orange is the anomaly and green is the corrected series.

41% and 46%, respectively. However, the results differ if we order the experiments by the product of epochs and data correction rate and then ignore the ones at the higher end of this spectrum. When disconsidering the pairs (1000; 1e−2), (200; 1e−1), and (1000; 1e−1), the reconstruction ability of both the Regular and Progressive version increased to positive 32% while the Selective version's ability only slightly dropped to 9%. The average outside losses of the first two versions were of 0.007 while the Selective version was just 0.001. These results suggest that the Regular and Progressive versions are not tolerant of high number of epochs and data correction rate. Table 5.1 shows the obtained results aggregated by all 9 possible combinations of epochs and data correction rate. The rows are ordered by the product of those two hyperparameters, which makes it easier to visualize the "turning points" at which the reconstruction abilities start to get worse. We also see that the outside loss is directly proportional to the product of the hyperparameters.

Finally, to conclude the analysis on artificial data, we test the theory that outside losses should be higher the later anomalies occur in data. As mentioned previously, we experimented with the occurrence of anomalies at 5 different positions. The positions were evenly spaced along the training series, ending at the last possible time step that was eligible for corrections. With the anomaly occurring at the start of the series, that zone will be used as a sample to predict the unaltered data that comes next to it. However, the opposite does not happen: that zone will never be used as a label. When the anomaly occurs later, the anomaly zone will also be used as a label, meaning previous values will attempt to be corrected in a way that makes them better predictors of the anomaly. This causes unnecessary changes to data. Anomalies at the middle of series suffer from this issue, but at least they are used as samples as much as they are used as labels. The rightmost anomaly, however, is more frequently used as a label than a sample, which translates to more outside loss. Table 5.2 confirms this theory on the Regular and Progressive variants, as the outside losses increase from left to right.

Table 5.1: Anomaly reconstruction ability (AR) and outside loss (OL) on artificial data, averaged by combination of epochs and data correction rate. The anomalies occurred at the middle of data.

Hyperparameters	Regular		Progressive		Selective	
	AR	OL	AR	OL	AR	OL
(50; 1e−3)	4.30%	1.94e−5	4.50%	2.01e−5	0.49%	1.12e−6
(200; 1e−3)	15.21%	0.00020	15.54%	0.00022	2.09%	1.21e−5
(50; 1e−2)	33.47%	0.00157	34.16%	0.00160	4.75%	0.00012
(1000; 1e−3)	39.99%	0.00140	39.00%	0.00148	6.50%	0.00011
(200; 1e−2)	56.52%	0.00796	55.57%	0.00767	14.86%	0.00093
(50; 1e−1)	41.74%	0.02981	41.86%	0.03061	23.32%	0.00507
(1000; 1e−2)	-2.87%	0.04708	5.56%	0.04406	38.51%	0.00431
(200; 1e−1)	-102.81%	0.10129	-102.61%	0.10207	38.73%	0.02145
(1000; 1e−1)	-455.38%	0.24167	-509.62%	0.25250	-39.47%	0.08094

Table 5.2: Average outside losses aggregated by anomaly position on artificial data. The experiments excluded (1000; 1e−2), (200; 1e−1) and (1000; 1e−1).

	Leftmost		Middle		Rightmost
Regular	0.00326	0.00665	0.00683	0.00732	0.00742
Progressive	0.00348	0.00651	0.00693	0.00737	0.00819
Selective	0.00070	0.00125	0.00104	0.00119	0.00094

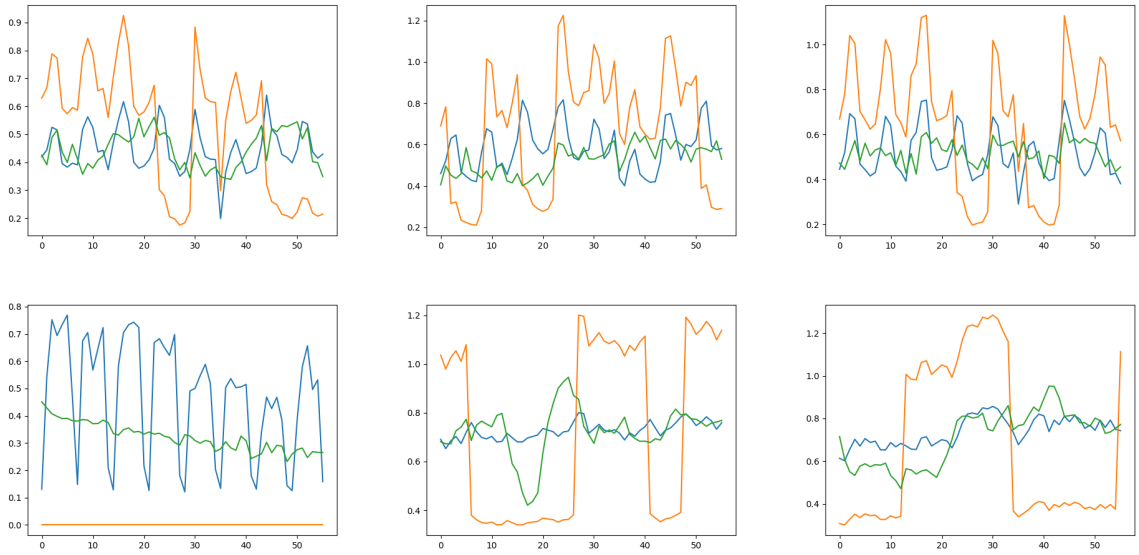


Figure 5.2: Best examples of anomaly reconstruction achieved with the (1000; 1e−2) hyperparameter combination out of the experiments on benchmark and case study data. Blue lines are the original series, orange are the anomalies and green are the corrected series.

5.1.2 Benchmark and case study data

The global averages of reconstruction ability across all experiments on these data, in which anomalies were always inserted at the middle position, were of: 1.7% in the Regular version; 5.2% in the Progressive version; 1.7% in the Selective version. The outside losses were around 0.01633 for both the Regular and Progressive variants and 0.00084 in the Selective one. When ignoring the experiments with data correction rate equal to 1e−1, we obtain the following anomaly reconstruction abilities: 13.3% in Regular Pastprop; 12.96% in Progressive Pastprop; 0.86% in Selective Pastprop. The corresponding outside losses were 0.00662, 0.00683 and 0.00023. Table 5.3 reveals the complete results aggregated by the combination of epochs and data correction rate.

The previous experiments on artificial data suggested that the Regular and Progressive variants were undermined by the usage of high number of epochs combined with high data correction rates. However, in the benchmark and case study experiments, the data correction rate alone seemed to be the deciding factor, since using a value of 1e−1 lead to negative reconstruction abilities. The best results were obtained with (1000; 1e−2). Figure 5.2 shows the greatest instances of anomaly reconstruction while using this combination of hyperparameters. To create context on what is happening outside the anomaly zones, figure 5.3 shows the full corrections to one of the series.

Furthermore, through table 5.3, we detect that the outside loss consistently grows with the product of the hyperparameters, just like what happened on the artificial data. More interestingly, that property is maintained even in the presence of negative reconstruction abilities. Lastly, we also note that the metrics obtained by Regular and Progressive Pastprop are fairly similar.

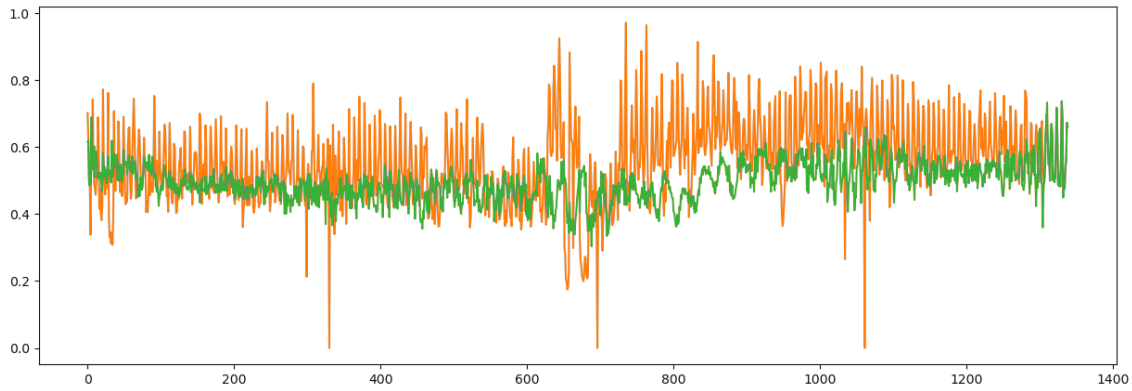


Figure 5.3: Corrections (green) made to one of the M5 time series that was affected by an anomaly of magnitude level 50 at the middle position (orange). The corrections were obtained through Progressive Pastprop while using the (1000; 1e−2) hyperparameter combination. This image is an expanded view of the top left graph in figure 5.2.

Table 5.3: Anomaly reconstruction ability (AR) and outside loss (OL) on benchmark and case study data, averaged by combination of epochs and data correction rate. The anomalies occurred at the middle of data.

Hyperparameters	Regular		Progressive		Selective	
	AR	OL	AR	OL	AR	OL
(50; 1e−3)	1.50%	2.53e−5	1.54%	2.66e−5	0.03%	1.27e−7
(200; 1e−3)	3.95%	0.00037	4.04%	0.00035	0.10%	2.01e−6
(50; 1e−2)	12.20%	0.00135	12.41%	0.00135	0.29%	1.33e−5
(1000; 1e−3)	13.33%	0.00326	12.33%	0.00324	0.44%	2.93e−5
(200; 1e−2)	19.16%	0.00893	20.42%	0.00912	0.70%	0.00015
(50; 1e−1)	-7.85%	0.02540	-4.20%	0.02635	2.61%	0.00076
(1000; 1e−2)	29.58%	0.02581	27.02%	0.02690	3.60%	0.00121
(200; 1e−1)	-58.64%	0.06547	-32.25%	0.06331	6.19%	0.00459
(1000; 1e−1)	-	-	-	-	-	-

Selective variant To understand why the Selective version showed less ability to reconstruct anomalies, a few things must be considered. After each epoch, only a subset of the corrections array is maintained, while the remaining deltas are changed to zeros. The threshold parameter controls how many deltas should be preserved, and in all experiments we set the threshold to be equal to the size of the anomaly. This is, in fact, a piece of information that is not available to the other Pastprop versions. Nonetheless, we give Selective Pastprop access to the anomaly size through the threshold parameter so that, at each epoch, it is has the possibility of choosing only the deltas that correspond to the anomaly zone in the series. This would be great if it happened every time, meaning that throughout the course of epochs only the anomaly zone would ever be changed (resulting in 0 outside loss). However, what ends up happening is that not all chosen deltas belong to the anomaly zone. Also, the chosen deltas vary from epoch to epoch, which makes it harder for the corrections to accumulate over time.

In the experiments, we measured the percentage (averaged across epochs) of preserved deltas that corresponded to the anomaly zone. This will be called the "percentage of aligned deltas". Considering all possible hyperparameters and anomalies occurring in the middle of series, that measure averaged 17.9% in the artificial data. Under the same conditions, that percentage was drastically different in the non artificial data, where it averaged just 0.1%. This clearly means that the mechanism for choosing what deltas to keep is not effective at recognizing the anomaly zones in series that are not well defined curves.

At last, to expand on the idea that it is easier to correct anomalies when they occur early in series, we check if the percentages of aligned deltas support this theory. We conclude that they do, since the averages (by anomaly position) of that measure on artificial data, starting from the leftmost anomalies to the rightmost, were the following: 41.6%, 23.3%, 17.9%, 14.2% and 18.8%.

5.2 Forecasting the future while reconstructing the past

In this section we investigate the research question: *Can Pastprop improve time series forecasting accuracy?*

To study this matter, the predictive abilities of each Pastprop variant on test data were compared to the ones obtained with LSTM, ARIMA, ETS and BATS. For all methods, the mechanisms to perform forecasts on test data are explained in the next two paragraphs.

5.2.1 Forecasting on test data

Pastprop and LSTM Both Pastprop and LSTM learn to model their weights on training data, which always corresponds to the first 70% of a given time series. After this, Pastprop and LSTM take the first *sample size* time steps of test data to forecast the next *label size* time steps. This means we are considering that all information is available up to *sample size* time steps into the test data. Then, we consider that one time step is advanced into the future, so that real information about it becomes available. Again, we use *sample size* time steps, including the new one that just became available, to forecast the next *label size* time steps. This process is repeated as shown

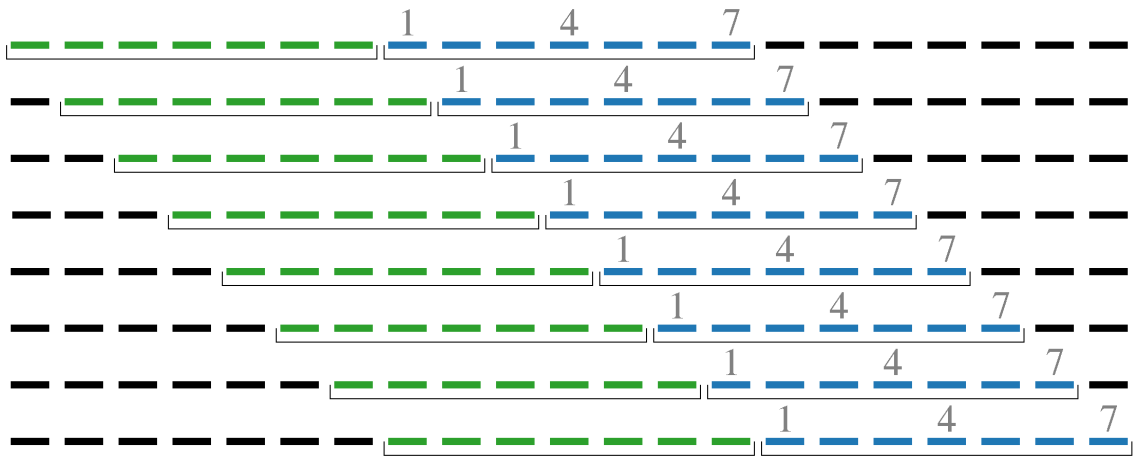


Figure 5.4: Illustration of the forecasting mechanism on test data of Pastprop and LSTM. The rectangles represent the time steps of the time series. Green rectangles denote input samples while blue rectangles indicate what is being forecasted. The numbers indicate the horizon at which the corresponding values are predicted.

in Figure 5.4. This means we perform several forecasts, most of which overlapping with other forecasts. What we do is build multiple series of predicted values, each one at a specific horizon. For example, we could build a series of predicted values at a 1 time step horizon, another series at 4, and another at 7, as illustrated in Figure 5.4. This allows us to study how the forecasting abilities of the methods vary with the considered horizons.

Baseline algorithms A similar mechanism was applied to obtain forecasts on test data with the baseline algorithms. However, for ARIMA, ETS and BATS to perform forecasts, they need to receive the training data as a whole (not in the format of sample to label pairings that are characteristic of supervised learning tasks). To start, the baseline algorithms received, as input data, the full training series plus the first *sample size* time steps (same amount used with LSTM and Pastprop) of the test series. Using that data, the next *label size* time steps were predicted. Then, a time step was advanced into the future. The baseline algorithms received input data with the same length as before. The only difference was that the newly available value was taken into account and the very first timestep from the training part of the series was discarded. This allowed the length of the input data given to baseline algorithms to remain constant throughout the process of forecasting on test data. The whole procedure was repeated by advancing more time steps into the future, one at a time. The last performed forecast was of the last *label size* timesteps of the test series. This strategy provided a scenario where the baseline algorithms had access to the same information and performed forecasts at the same timesteps than LSTM and Pastprop. Here, we also considered multiple series of forecasted values at specific horizons, in the same way as before. One final thing to note is that, out of the three baseline algorithms, only ARIMA and BATS can forecast with multivariate time series. Moreover, they need to know the values of the explanatory variables at the time steps they wish to predict (this is not the case with LSTM and Pastprop).

Since, in a real setting, it would be impossible to know those values beforehand, only explanatory variables regarding basic calendar information are provided to ARIMA an BATS.

5.2.2 Benchmark and case study data

We compare the forecasting abilities of Pastprop to LSTM, ARIMA and ETS at three different horizons: 1, 7 and 28 days. The considered measure of similarity between the forecasted values and the real values of the test series is MSE. Table 5.4 showcases this information. We can see that the accuracies of LSTM and the Pastprop variants can be generally grouped by number of epochs. A higher number of epochs always translated to worse results. The accuracies with 50 and 200 epochs were somewhat acceptable (even though worse than ARIMA and ETS), but using 1000 epochs produced extremely bad results. We believe this has to do with the fact that the specific implementation of LSTM we considered does not apply the *sigmoid* function at the output layer. Instead, the *hidden output* is simply multiplied by the output layer's weights to obtain the predicted values (Eq. 2.9). Since forecasts are not filtered by sigmoid, they are not restricted to any scale. In our case, due to the high number of epochs, the weights increased to a point that resulted in forecasts no longer being in the interval of 0 to 1, hence the low accuracies.

Reducing the scope of analysis to just the experiments with 50 epochs, where the aforementioned problem does not occur, we summarize the main findings:

- Horizon 7 displayed the best results. The accuracies at this horizon were, on average, 7% better compared to horizon 1 and 7.5% better compared to horizon 28. Consequentially, there was no substantial difference between forecasting accuracy at horizons 1 and 28.
- Considering (50; 1e$-$2) and (50; 1e$-$1) only, Regular and Progressive Pastprop were, on average, 9% and 13.3% better than LSTM, respectively. Selective Pastprop was 3.8% worse. More statistics on this can be found in Table 5.5.

We conclude that, with the just mentioned hyperparameters and the experimental conditions defined in Table 5.4, the Pastprop variants seem to improve the predictive ability of LSTM. Figure 5.5 is an example of that improvement achieving 66.7%. This type of gains are rare. However, it was more frequent for Pastprop to improve on LSTM than the opposite happening.

As to why ARIMA and ETS outperformed the other methods, there might be several reasons. More fine-tuning of the hyperparameters, such as the number of hidden units and learning rate, could be necessary. These were kept constant to prevent the total amount of experiments from exploding. Additionally, the base implementation of LSTM we used was the most "vanilla" possible so that implementing Pastprop would be more straightforward. Studies have shown the advantages of using deeper, stacked LSTM architectures in comparison to the more common shallow ones [20, 50].

Table 5.4: Forecasting accuracies (MSE) at 1, 7 and 28 day horizons obtained by LSTM, Pastprop, ARIMA and ETS on univariate benchmark data (3 M5 series and 3 electricity series). Each series was affected by an anomaly at the middle of training data. Results were aggregated by combination of epochs and data correction rate.

Horizon	Hyperparameters	LSTM	Regular	Progressive	Selective	ARIMA	ETS
1	(50; 1e−3)	0.02462	0.02775	0.02970	0.02790	0.01097	0.01351
	(50; 1e−2)	0.02729	0.03041	0.02458	0.02705		
	(50; 1e−1)	0.02615	0.02214	0.02303	0.03133		
	(200; 1e−3)	0.03994	0.06429	0.05774	0.05396		
	(200; 1e−2)	0.03665	0.03445	0.04312	0.05807		
	(200; 1e−1)	0.03930	0.04343	0.03777	0.03393		
	(1000; 1e−3)	2.06349	2.22513	6.98732	6.31422		
	(1000; 1e−2)	3.62515	1.99246	3.07272	7.89312		
7	(50; 1e−3)	0.02486	0.02396	0.02920	0.02722	0.01765	0.01679
	(50; 1e−2)	0.02836	0.02460	0.02526	0.02512		
	(50; 1e−1)	0.02186	0.01971	0.02050	0.02840		
	(200; 1e−3)	0.03810	0.07618	0.05311	0.04773		
	(200; 1e−2)	0.03349	0.03421	0.03418	0.05699		
	(200; 1e−1)	0.03457	0.04138	0.03591	0.03519		
	(1000; 1e−3)	2.20410	2.09030	7.33739	6.0169		
	(1000; 1e−2)	4.10815	1.62792	3.42391	7.81961		
28	(50; 1e−3)	0.02630	0.02928	0.02983	0.02906	0.01761	0.01753
	(50; 1e−2)	0.02986	0.02841	0.02815	0.02735		
	(50; 1e−1)	0.02881	0.02240	0.01876	0.02697		
	(200; 1e−3)	0.03965	0.07060	0.04814	0.05336		
	(200; 1e−2)	0.03697	0.03311	0.03415	0.04499		
	(200; 1e−1)	0.03448	0.04105	0.03739	0.03239		
	(1000; 1e−3)	1.97027	2.48150	8.14669	4.44908		
	(1000; 1e−2)	4.17214	1.65044	3.49904	6.89089		

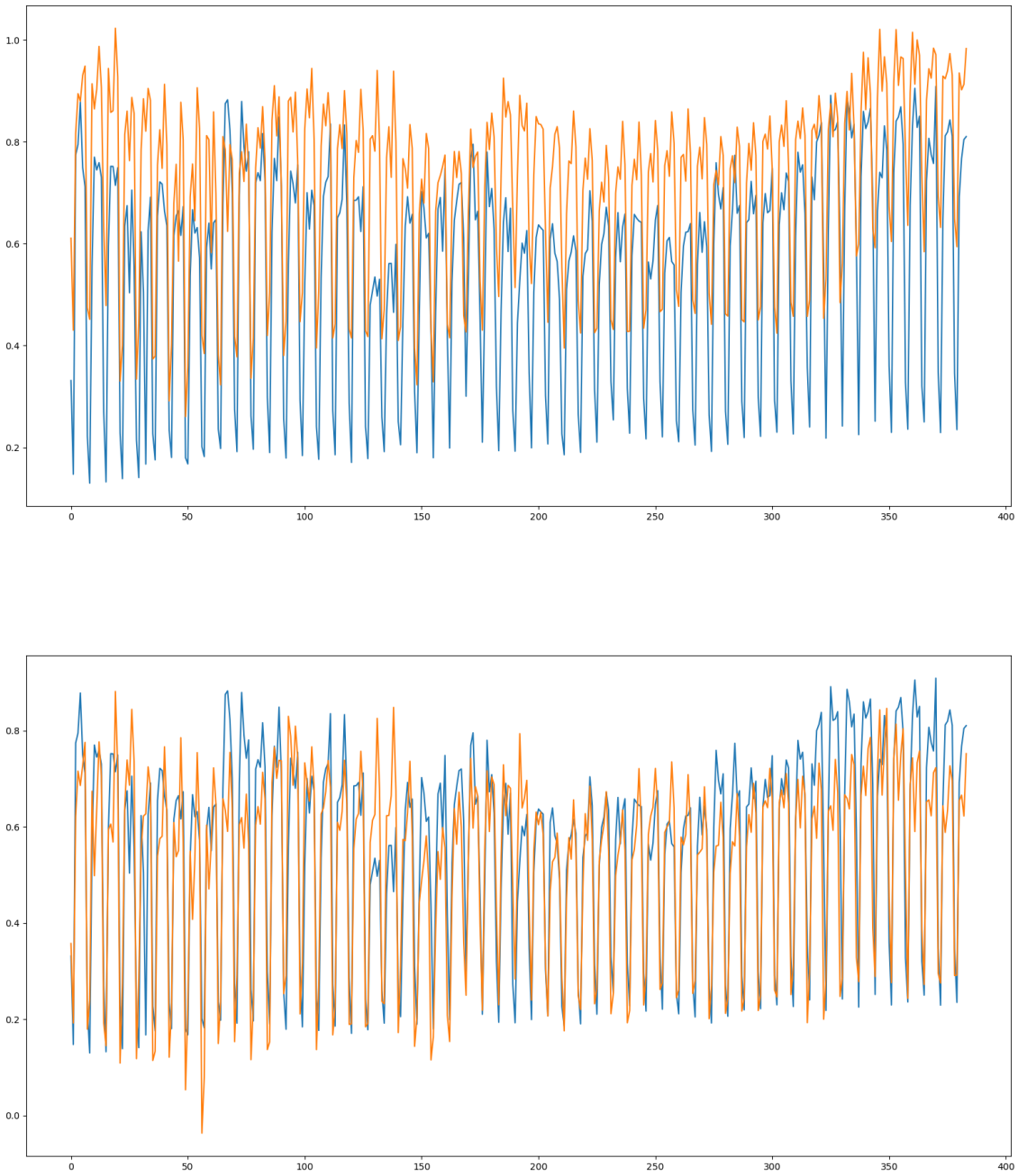


Figure 5.5: Top: LSTM forecast on electricity data that was affected by an anomaly at middle of training data. Bottom: Progressive Pastprop forecast on the same data with the hyperparameters (50; 1e−1). Blue are the test data and orange are the predictions. The improvement on forecasting accuracy was of 66.7%.

Table 5.5: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by each Pastprop variant in relation to LSTM. This information can be inferred from table 5.4 when considering only the hyperparameters (50; 1e−2) and (50; 1e−1).

	mean	sd	median	min	max
Regular	9.02%	11.57%	11.55%	-11.43%	22.25%
Progressive	13.27%	10.88%	10.43%	5.73%	34.88%
Selective	-3.77%	17.00%	3.63%	-29.92%	11.42%

Table 5.6: Forecasting accuracies (MSE) at 1, 7 and 14 day horizons obtained by LSTM, Pastprop, ARIMA and BATS on all benchmark and case study data. The series were not affected by manually inserted anomalies. Results were aggregated by combination of epochs and data correction rate. The maximum horizon of 14 days was imposed by the case study.

Horizon	Hyperparameters	LSTM	Regular	Progressive	Selective	ARIMA	BATS
1	(50; 1e−3)	0.03663	0.03558	0.03879	0.03703	0.01726	0.01932
	(50; 1e−2)	0.03684	0.03574	0.03426	0.03860		
	(50; 1e−1)	0.03358	0.03103	0.03272	0.03987		
7	(50; 1e−3)	0.03514	0.03520	0.03767	0.03589	0.0205	0.02225
	(50; 1e−2)	0.03771	0.03507	0.03375	0.03664		
	(50; 1e−1)	0.03420	0.03377	0.03303	0.04031		
14	(50; 1e−3)	0.03871	0.03619	0.03901	0.03649	0.02034	0.02307
	(50; 1e−2)	0.03931	0.03693	0.03522	0.03791		
	(50; 1e−1)	0.03627	0.03345	0.03359	0.03988		

The previously discussed experiments included the occurrence of anomalies at the middle of training data. We also want to find out if Pastprop can still improve forecasting accuracy when no anomalies are manually inserted in data. Table 5.6 contains information about these results. The findings are aligned with those on anomalous data. The best forecasting accuracies were also observed at a 7 day horizon and there were still MSE improvements when using Pastprop in comparison to LSTM. However, the gains were smaller. Table 5.7 displays detailed statistics on the accuracy gains of each Pastprop variant over LSTM.

Table 5.7: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by each Pastprop variant in relation to LSTM. This information can be inferred from table 5.6 when considering only the hyperparameters (50; 1e−2) and (50; 1e−1).

	mean	sd	median	min	max
Regular	5.44%	2.70%	6.53%	1.26%	7.78%
Progressive	6.88%	3.36%	7.20%	2.56%	10.50%
Selective	-7.49%	9.76%	-7.37%	-18.73%	3.56%

5.3 Forecasting the future with a reconstructed past

In this section we investigate the research question: *Do other algorithms benefit from using the series corrected by Pastprop as training data?*

We analyze the gains in forecasting accuracy between providing data corrected by Pastprop and non corrected data to standard LSTM and ARIMA. The strategy to perform this study was simple. First, we took a time series, provided it as input to LSTM and ARIMA and recorded the forecasting accuracies. The same data was given to the Pastprop methods so that we could obtain corrected time series. Finally, these series were provided to LSTM and ARIMA and the resulting accuracies were compared to those obtained earlier.

We divided the experiments into two groups. In the first group, we used data affected by manually inserted anomalies as a starting point. In the second group, the starting series were the original versions of data, without anomalies. For both groups of experiments we used the univariate benchmark time series. The anomalies generated for the first group were at the middle position with magnitude levels 0 and 50. The results of the first group are displayed in Table 5.8 and Table 5.9. For the second group, the results are in Table 5.10 and Table 5.11.

5.3.1 Corrections over anomalous data

Before discussing the results, we provide clarification on exactly how many experiments took place in the first group. Six univariate time series were used from the benchmark datasets. Since anomalies were applied, and two levels of magnitude were considered, that lead to 12 time series. These were passed to LSTM and ARIMA. The corresponding corrected time series were obtained during the main experiments. Each of them was given to Regular, Progressive and Selective Pastprop with a certain hyperparameter combination. Additionally, each of the Pastprop versions produced 5 corrected time series, since the main experiments consisted of 5 different runs. In short, there were 12 time series, and each of them had 15 corrected time series associated to them per hyperparameter combination. Considering 8 distinct hyperparameter combinations, the result was a total of 1440 corrected time series that were provided to ARIMA. Regarding LSTM, only 3 hyperparameter combinations were used, resulting in 540 corrected series.

Moving on, the results in Table 5.8 showed, on average, negative gains for almost all hyperparameter combinations and Pastprop versions. The few exceptions included Progressive Pastprop with (1000; 1e−3), which got a mean accuracy gain of 3.29% and a maximum of 24.57%. The Selective version with (1000; 1e−2) also obtained similar results. When observing the median gains throughout the table, we notice that accuracy gains above 1% were never the most frequent thing to happen. The standard deviation was also higher than the mean in all entries.

Advancing to Table 5.9, the results were not very convincing. There was a 16.79% average gain when using Regular Pastprop with (50; 1e−2), but there seems to be too much variability in the results to draw reliable conclusions.

Table 5.8: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by ARIMA when using data corrected by each Pastprop variant vs. ARIMA with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were affected by anomalies of magnitude levels 0 and 50 at the middle position. The horizon of the forecasts was of 7 days.

ARIMA (with anomaly)						
Hyperparameters	Pastprop	mean	std	median	min	max
(50; 1e−3)	Regular	-1.95%	7.51%	-3.18%	-9.33%	13.19%
	Progressive	-1.66%	7.58%	-3.15%	-8.67%	13.72%
	Selective	-1.87%	8.59%	-2.73%	-10.16%	15.60%
(50; 1e−2)	Regular	-2.95%	5.23%	-2.56%	-13.40%	4.38%
	Progressive	-1.51%	4.10%	-1.81%	-7.57%	4.05%
	Selective	0.22%	7.53%	-0.37%	-9.69%	22.79%
(50; 1e−1)	Regular	-2.51%	10.68%	-5.18%	-19.71%	19.40%
	Progressive	-1.67%	8.91%	-3.83%	-15.21%	18.04%
	Selective	-0.35%	8.01%	-0.22%	-14.25%	15.73%
(200; 1e−3)	Regular	-1.53%	5.17%	-0.39%	-14.00%	6.60%
	Progressive	-1.16%	5.49%	-0.37%	-13.31%	7.30%
	Selective	-1.01%	7.92%	-0.58%	-17.61%	19.68%
(200; 1e−2)	Regular	1.01%	10.8%	-1.09%	-13.77%	21.02%
	Progressive	0.28%	10.82%	-2.36%	-12.61%	22.32%
	Selective	-0.22%	7.28%	-1.27%	-7.02%	21.41%
(200; 1e−1)	Regular	-0.67%	10.55%	-0.79%	-18.00%	13.21%
	Progressive	0.53%	9.12%	-2.28%	-12.72%	16.23%
	Selective	0.84%	6.96%	0.46%	-8.73%	11.72%
(1000; 1e−3)	Regular	2.86%	11.98%	-1.01%	-13.80%	22.27%
	Progressive	3.29%	10.90%	0.56%	-8.94%	24.57%
	Selective	-2.05%	7.46%	-0.13%	-16.84%	15.58%
(1000; 1e−2)	Regular	-4.36%	13.46%	-3.57%	-25.73%	19.01%
	Progressive	-2.74%	12.17%	-3.93%	-21.47%	19.26%
	Selective	3.48%	8.17%	0.34%	-3.82%	20.90%

Table 5.9: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by LSTM when using data corrected by each Pastprop variant vs. LSTM with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were affected by anomalies of magnitude levels 0 and 50 at the middle position. The horizon of the forecasts was of 7 days.

LSTM (with anomaly)						
Hyperparameters	Pastprop	mean	std	median	min	max
(50; 1e−3)	Regular	-11.90%	42.18%	-0.32%	-115.84%	35.83%
	Progressive	-21.95%	42.89%	-9.14%	-107.63%	39.14%
	Selective	-14.81%	61.57%	4.26%	-209.20%	36.55%
(50; 1e−2)	Regular	16.79%	23.48%	13.41%	-17.51%	70.79%
	Progressive	-1.51%	4.10%	-1.81%	-7.57%	4.05%
	Selective	0.22%	7.53%	-0.37%	-9.69%	22.79%
(50; 1e−1)	Regular	-20.37%	52.86%	-9.64%	-124.28%	61.82%
	Progressive	-37.39%	86.86%	-14.62%	-246.36%	84.49%
	Selective	-21.19%	35.82%	-5.08%	-118.39%	8.61%

5.3.2 Corrections over original data

The same logic that applies to the total amount of experiments on the first group also applies here. The main difference is that, since no anomalies occur, the number of experiments is cut in half.

Table 5.10 revealed surprising results, with the average gains being positive in every occasion. The Selective variant was the best within every hyperparameter combination. Moreover, the average gain with that Pastprop version was higher than 10% with all hyperparameters except (50; 1e−3). The median followed the mean closely, which gives us more confidence to claim the consistency of the gains.

The forecast in which the data corrections provided to ARIMA resulted in the most gains relative to the original series can be seen in Figure 5.7. The corrected data responsible for that improvement can be seen in Figure 5.6. The shape of the predicted series obtained with the corrected data looks like it has "shrunk" (bottom graph in Figure 5.7). That phenomenon lead to 33% better accuracy (less MSE) likely because of the two extreme predictions in the top graph getting closer to their actual values in the bottom graph. Even though the forecast done with the original data might seem better adjusted to the real values, the forecast done with corrected data suffers less from failing to predict the exact time at which an extremely low value will occur.

Lastly, Table 5.11 demonstrated a majority of negative gains and too much variability, so we conclude that there were no benefits to be had in providing corrected data to LSTM.

Table 5.10: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by ARIMA when using data corrected by each Pastprop variant vs. ARIMA with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were not affected by anomalies. The horizon of the forecasts was of 7 days.

ARIMA (no anomaly)						
Hyperparameters	Pastprop	mean	std	median	min	max
(50; 1e−3)	Regular	3.20%	5.70%	0.24%	-2.17%	14.18%
	Progressive	4.16%	5.45%	2.96%	-2.53%	13.27%
	Selective	5.24%	6.74%	4.84%	-3.54%	14.96%
(50; 1e−2)	Regular	9.33%	8.30%	6.53%	-1.20%	20.32%
	Progressive	8.81%	8.59%	8.99%	-3.51%	21.55%
	Selective	12.32%	8.43%	12.08%	-2.82%	23.74%
(50; 1e−1)	Regular	1.35%	9.39%	-1.71%	-10.26%	16.28%
	Progressive	2.84%	9.40%	-2.33%	-5.32%	16.89%
	Selective	13.08%	8.99%	12.42%	-0.97%	26.20%
(200; 1e−3)	Regular	10.47%	9.26%	10.43%	-1.38%	24.20%
	Progressive	10.81%	9.04%	11.25%	-1.52%	23.79%
	Selective	12.01%	8.66%	12.31%	-3.34%	23.93%
(200; 1e−2)	Regular	6.26%	10.56%	1.21%	-5.43%	23.61%
	Progressive	5.19%	10.02%	1.95%	-7.07%	22.53%
	Selective	12.31%	8.70%	13.21%	-2.44%	23.98%
(200; 1e−1)	Regular	7.04%	8.74%	7.02%	-6.09%	21.70%
	Progressive	8.04%	12.60%	2.89%	-4.08%	27.93%
	Selective	13.25%	9.62%	9.52%	3.48%	29.54%
(1000; 1e−3)	Regular	10.97%	13.96%	8.74%	-6.28%	32.08%
	Progressive	12.73%	11.52%	10.56%	-0.33%	31.81%
	Selective	12.53%	8.47%	12.69%	-2.56%	24.18%
(1000; 1e−2)	Regular	7.88%	12.83%	6.49%	-6.71%	28.60%
	Progressive	6.07%	13.18%	-0.20%	-7.01%	26.82%
	Selective	11.35%	11.16%	10.43%	-0.85%	27.09%

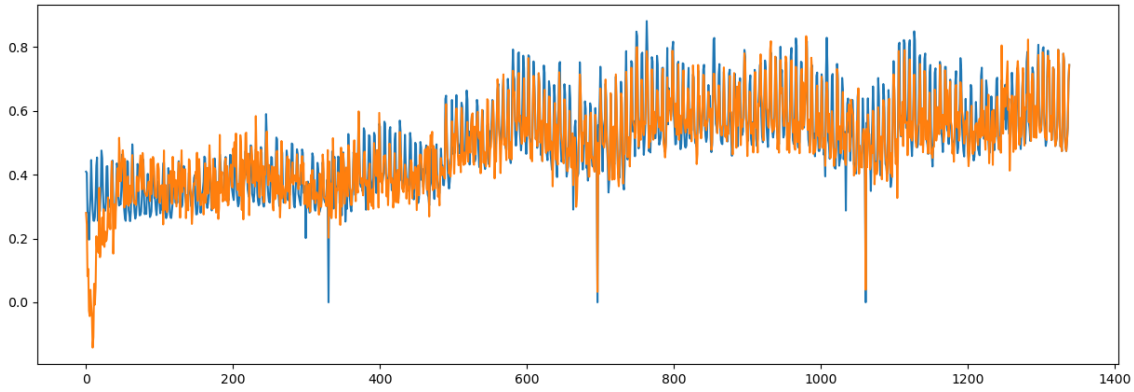


Figure 5.6: Corrections (orange) made to an M5 series with no anomalies (blue) with Selective Pastprop and (200; 1e−1). When provided as input data to ARIMA, this series was the best at improving forecasting accuracy in comparison to the original series, as shown in Figure 5.7.

Table 5.11: Statistics on the forecasting accuracy gains (percentage decreases of MSE) achieved by LSTM when using data corrected by each Pastprop variant vs. LSTM with the non corrected data. The experiments considered here were done on univariate benchmark data (3 M5 series and 3 electricity series) that were not affected by anomalies. The horizon of the forecasts was of 7 days.

LSTM (no anomaly)						
Hyperparameters	Pastprop	mean	std	median	min	max
(50; 1e−3)	Regular	-5.75%	32.83%	-0.36%	-94.25%	46.81%
	Progressive	-13.18%	19.48%	-6.62%	-60.40%	9.49%
	Selective	-7.70%	22.72%	-0.90%	-66.18%	18.65%
(50; 1e−2)	Regular	5.84%	26.19%	0.03%	-41.43%	69.26%
	Progressive	-2.83%	16.75%	-3.78%	-28.42%	42.37%
	Selective	3.66%	18.96%	3.10%	-25.38%	44.11%
(50; 1e−1)	Regular	-8.97%	28.26%	-20.32%	-40.81%	47.48%
	Progressive	-1.82%	36.26%	-3.96%	-57.17%	51.32%
	Selective	-12.85%	19.32%	-12.50%	-61.31%	19.13%

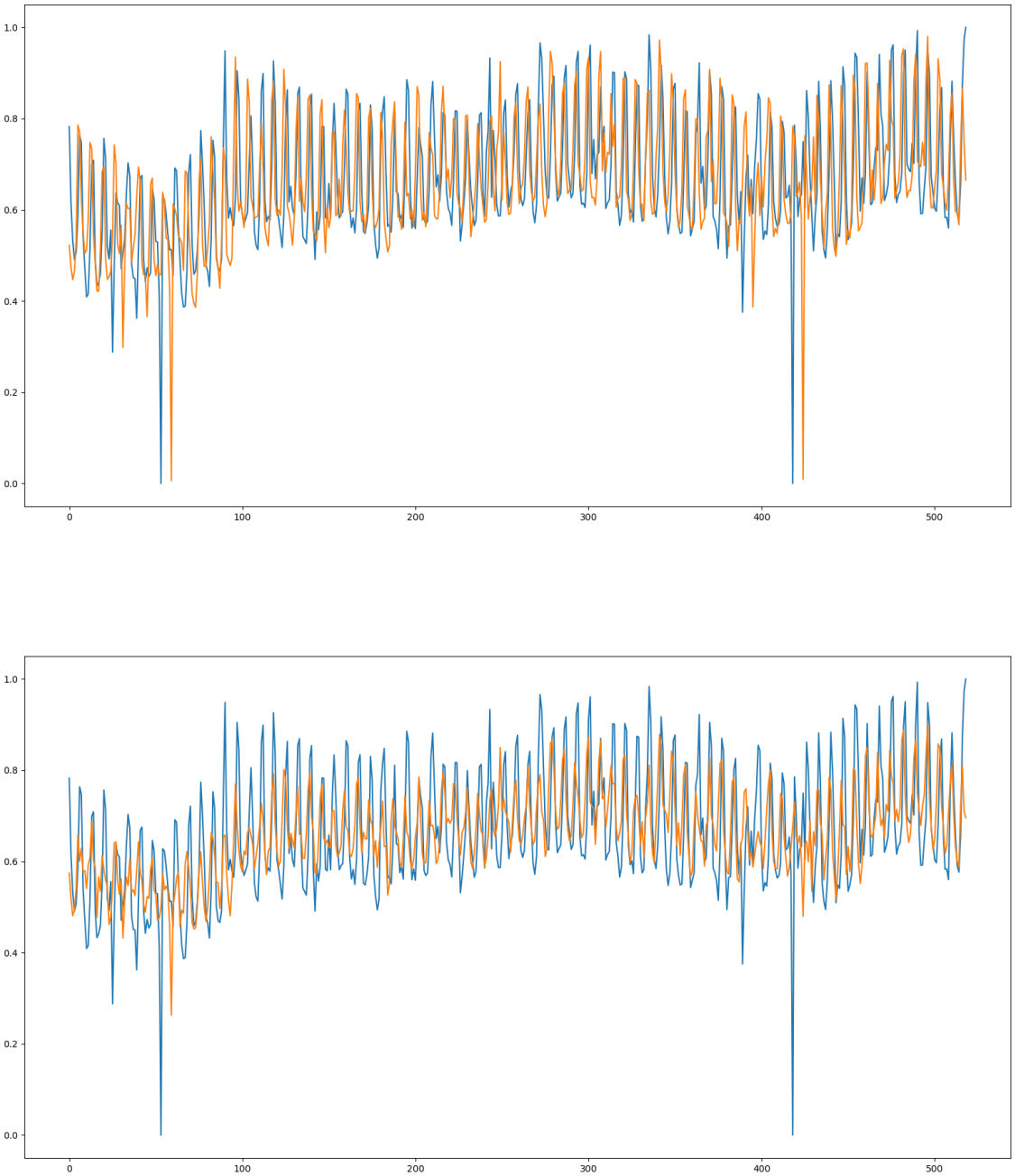


Figure 5.7: At the top: forecast (on test data) using ARIMA with a M5 series at 7 day horizon. At the bottom: forecast on the same test data but giving ARIMA the corrected training series shown in Figure 5.6. The bottom forecast was 33% more accurate in terms of MSE. Blue are the real test data and orange are the predictions.

Chapter 6

Conclusions

This work started with a brief literature review on pastcasting and time series forecasting. The basic concepts of neural networks and backpropagation were also introduced. Then, a more detailed explanation on the forward and backward pass of LSTM was presented. This allowed us to clearly define how Pastprop could be implemented while also providing theoretical context.

The occurence of anomalies in data motivated the development of Pastprop - an adaptation of the LSTM backpropagation algorithm which also updates data as a part of the learning process. The intention was for the corrections to be applied mostly on anomalous parts of data, that is, past values that weren't good predictors of the future. Pastprop can be thought of as a deep learning time series forecasting method which outputs a model (the network weights, like LSTM), and also outputs a new version of the training data that it received as input. We developed three different Pastprop variants and tested them on artificial time series, benchmark datasets and data from a case study provided by Inovretail.

We showed the potential of the Pastprop methods to reconstruct data anomalies that were previously generated and inserted in time series. In artificial data, the ability to reconstruct them was as high as 83.9% given some very specific conditions. In more general scenarios (but on that same type of data), the reconstructions were still able to achieve averages of 56.52% when using the best possible hyperparameters. In other time series that weren't as "well behaved" as sinusoidal functions, the best averages of reconstruction ability were of 29.58%. Although in certain experiments the corrections were able to resemble the original data at the anomaly zone in impressive manner, that usually came with the cost of too much changes outside the anomaly. The implementations of Pastprop were never given the information of where anomalies were located. The Selective variant was built with the finality of "guessing" the position of anomalies to avoid changes outside of those zones. However, that only proved to be effective on artificial data. Assuming the anomalies' positions were exactly known, the Pastprop methods could easily be adapted to receive that information as a parameter and to only correct those zones. That would be an highly favorable use case for Pastprop.

We also evaluated the direct effects of Pastprop on forecasting accuracy when compared to the standard LSTM it is based on. We concluded that accuracy gains did exist, and that those were slightly amplified when the training data was previously affected by anomalies. This suggests that the corrections made by Pastprop during learning have a positive impact on the modeling of the weights, and that the value of corrections increases when dealing with data that was affected by manually generated anomalies. This is to be expected. However, it is probably more exciting that Pastprop was still better than LSTM when dealing with data that was not previously manipulated.

Finally, we interpreted the series outputted by Pastprop as different versions of the same data that could potentially be used with other forecasting methods. We tested for accuracy gains in comparison to the original versions of data and found out that ARIMA consistently improved its forecasts when receiving non anomalous time series that were changed by Pastprop.

As future work, there are several ideas that could be interesting to explore. The idea of adapting the backpropagation algorithm is not limited by this LSTM architecture. The implementation could be generalized to stacked LSTM or even to other types of Recurrent Neural Networks such as the Gated Recurrent Unit (GRU). Other ideas related to the Pastprop method include applying corrections not only to the main time series but also to the explanatory variables. This could make sense, for example, when there is anomalous or missing information regarding precipitation amounts. Moreover, experimenting with applying corrections to the groundtruth labels during training could also be a direction to explore. Finally, the potential of Pastprop could be studied in the context of other supervised learning tasks that are not related to time series forecasting.

References

[1] A pure Python and Numpy implementation of an LSTM Network. https://github.com/AlphaDeeX/LSTM-Python. Accessed: 2020-06-20.

[2] An efficient, batched LSTM. https://gist.github.com/karpathy/587454dc0146a6ae21fc. Accessed: 2020-06-20.

[3] Forecasting Functions for Time Series and Linear Models. https://www.rdocumentation.org/packages/forecast. Accessed: 2020-07-02.

[4] The M5 Competition. https://mofc.unic.ac.cy/m5-competition/. Accessed: 2020-06-05.

[5] UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.php. Accessed: 2020-06-05.

[6] Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

[7] The role of multidirectional temporal analysis in scenario planning exercises and Planning Support Systems. *Computers, Environment and Urban Systems*, 64:91–102, 2017.

[8] Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2019.

[9] Knut Are Aastveit, Karsten R. Gerdrup, Anne Sofie Jore, and Leif Anders Thorsrud. Nowcasting GDP in real time: A density combination approach. *Journal of Business and Economic Statistics*, 32(1):48–68, 2014.

[10] Nesreen K. Ahmed, Amir F. Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5):594–621, 2010.

[11] Muhammad J Amjad and Devavrat Shah. Censored demand estimation in retail. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–28, 2017.

[12] Goodness C. Aye, Mehmet Balcilar, Rangan Gupta, and Anandamayee Majumdar. Forecasting aggregate retail sales: The case of South Africa. *International Journal of Production Economics*, 160:66–79, 2015.

[13] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140, 2020.

[14] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology. pages 1–16, 2019.

[15] Robert Goodell Brown. *Statistical forecasting for inventory control*. McGraw/Hill, 1959.

[16] Chris Chatfield. *Time-series forecasting*. Chapman and Hall/CRC, 2000.

[17] Gang Chen. A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation. pages 1–10, 2016.

[18] I. Fei Chen and Chi Jie Lu. Sales forecasting by combining clustering and machine-learning techniques for computer retailing. *Neural Computing and Applications*, 28(9):2633–2647, 2017.

[19] Ching Wu Chu and Guoqiang Peter Zhang. A comparative study of linear and nonlinear models for aggregate retail sales forecasting. *International Journal of Production Economics*, 86(3):217–231, 2003.

[20] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. pages 1–11, 2018.

[21] Claudimar Pereira Da Veiga, Cássia Rita Pereira Da Veiga, Anderson Catapan, Ubiratã Tortato, and Wesley Vieira Da Silva. Demand forecasting in food retail: A comparison between the Holt-Winters and ARIMA models. *WSEAS Transactions on Business and Economics*, 11(1):608–614, 2014.

[22] Karl H. Dreborg. Essence of backcasting. *Futures*, 28(9):813–828, 1996.

[23] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. 2017.

[24] Amin Fakhrazari and Hamid Vakilzadian. A survey on time series data mining. *IEEE International Conference on Electro Information Technology*, pages 476–481, 2017.

[25] Tak Chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[26] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. 1999.

[27] Michael Geurts, George E. P. Box, and Gwilym M. Jenkins. Time Series Analysis: Forecasting and Control. *Journal of Marketing Research*, 14(2):269, 1977.

[28] Z. X. Guo, W. K. Wong, and Min Li. A multivariate intelligent decision-making model for retail sales forecasting. *Decision Support Systems*, 55(1):247–255, 2013.

[29] Suleka Helmini, Nadheesh Jihan, Malith Jayasinghe, and Srinath Perera. Sales Forecasting Using Multivariate Long Short Term Memory Networks. *PeerJ Preprints*, (May):16, 2019.

[30] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1), 1991.

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

[32] J. Holmberg and K. H. Robert. Backcasting — a framework for strategic planning. *International Journal of Sustainable Development and World Ecology*, 7(4):291–308, 2000.

[33] Tao Huang, Robert Fildes, and Didier Soopramanien. The value of competitive information in forecasting FMCG retail product sales and the variable selection problem. *European Journal of Operational Research*, 237(2):738–748, 2014.

[34] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[35] Hyun Hak Kim and Norman R. Swanson. Methods for Pastcasting, Nowcasting and Forecasting Using Factor-MIDAS: With an Application to Korean GDP. *SSRN Electronic Journal*, 2017.

[36] Mahesh Kumar and Nitin R. Patel. Using clustering to improve sales forecasts in retail merchandising. *Annals of Operations Research*, 174(1):33–46, 2010.

[37] A. L.D. Loureiro, V. L. Miguéis, and Lucas F.M. da Silva. Exploring the use of deep neural networks for sales forecasting in fashion retail. *Decision Support Systems*, 114(January):81–93, 2018.

[38] Shaohui Ma, Robert Fildes, and Tao Huang. Demand forecasting with high dimensional data: The case of SKU retail sales forecasting with intra- and inter-category promotional information. *European Journal of Operational Research*, 249(1):245–257, 2016.

[39] Bohdan Macukow. Neural Networks – State of Art, Brief History, Basic Models and Architecture. In Khalid Saeed and Władysław Homenda, editors, *Computer Information Systems and Industrial Management*, pages 3–14, Cham, 2016. Springer International Publishing.

[40] G. Mahalakshmi, S. Sridevi, and S. Rajaram. A survey on forecasting of time series data. *2016 International Conference on Computing Technologies and Intelligent Data Engineering, ICCTIDE 2016*, pages 1–8, 2016.

[41] Soheila Mehrmolaei and Mohammad Reza Keyvanpour. Time series forecasting using improved ARIMA. *2016 Artificial Intelligence and Robotics, IRANOPEN 2016*, pages 92–97, 2016.

[42] Adam J Mersereau. Demand estimation from censored observations with inventory record inaccuracy. *Manufacturing & Service Operations Management*, 17(3):335–349, 2015.

[43] Nagham Muslim Mohammad. Censored Time Series Analysis. 2014.

[44] Timo Nolle, Alexander Seeliger, Nils Thoma, and Max Mühlhäuser. DeepAlign: Alignment-Based Process Anomaly Correction Using Recurrent Neural Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12127 LNCS:319–333, 2020.

[45] Evgeniy Ozhegov and Daria Teterina. The Ensemble Method for Censored Demand Prediction. *SSRN Electronic Journal*, 2019, 2018.

[46] Bohdan M. Pavlyshenko. Linear, machine learning and probabilistic approaches for time series analysis. *Proceedings of the 2016 IEEE 1st International Conference on Data Stream Mining and Processing, DSMP 2016*, (August):377–381, 2016.

[47] Bohdan M. Pavlyshenko. Machine-learning models for sales time series forecasting. *Data*, 4(1):1–11, 2019.

[48] Patrícia Ramos, Nicolau Santos, and Rui Rebelo. Performance of state space and ARIMA models for consumer retail sales forecasting. *Robotics and Computer-Integrated Manufacturing*, 34:151–163, 2015.

[49] John Bridger Robinson. Energy backcasting A proposed method of policy analysis. *Energy Policy*, 10(4):337–344, 1982.

[50] Alaa Sagheer and Mostafa Kotb. Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Scientific Reports*, 9(1):1–16, 2019.

[51] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404(March):1–43, 2020.

[52] Heng Shi, Minghao Xu, and Ran Li. Deep Learning for Household Load Forecasting-A Novel Pooling Deep RNN. *IEEE Transactions on Smart Grid*, 9(5):5271–5280, 2018.

[53] Yu-Sen Shih and Min-Huei Lin. A LSTM Approach for Sales Forecasting of Goods with Short-Term Demands in E-Commerce. In Ngoc Thanh Nguyen, Ford Lumban Gaol, Tzung-Pei Hong, and Bogdan Trawiński, editors, *Intelligent Information and Database Systems*, pages 244–256, Cham, 2019. Springer International Publishing.

[54] Sima Siami-Namini and Akbar Siami Namin. Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. pages 1–19, 2018.

[55] Pawan Kumar Singh, Yadunath Gupta, Nilpa Jha, and Aruna Rajan. Fashion Retail: Forecasting Demand for New Items. 2019.

[56] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. sep 2019.

[57] Pang-Ning Tan. *Neural Networks*, pages 2497–2501. Springer New York, New York, NY, 2018.

[58] Yuzhen Wang, Dan Chang, and Chaojin Zhou. The Study of a Sales Forecast Model Based on SA-LSTM The Study of a Sales Forecast Model Based on SA-LSTM. 2019.

[59] Paul J. Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[60] Peter R. Winters. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342, 1960.

[61] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. GAIN: Missing data imputation using generative adversarial nets. *35th International Conference on Machine Learning, ICML 2018*, 13:9042–9051, 2018.

[62] Quan Yu, Kesheng Wang, Jan Ola Strandhagen, and Yi Wang. Application of Long Short-Term Memory Neural Network to Sales Forecasting in Retail—A Case Study. In Kesheng Wang, Yi Wang, Jan Ola Strandhagen, and Tao Yu, editors, *Advanced Manufacturing and Automation VII*, pages 11–17, Singapore, 2018. Springer Singapore.

[63] David Zipser and Ronald J Williams. Gradient-Based Learning Algorithms for Recurrent Networks and Thei r Computational Complexity. *Back-propagation: Theory, Architectures and Applications*, pages 433–486, 1995.