An Experimental Study Investigating The Effects of Data Pre-processing Methods on Deep Learning Based Time Series Data Prediction

Mehmet Aktas * , Mustafa Bugra Yilmaz , Abdulkadir Karabacak

Posted Date: 25 October 2023

doi: 10.20944/preprints202310.1635.v1

Keywords: experimental study on time series prediction methods; investigation on time series model predictions; analysis of the effects of data pre-processing methods on time series prediction; long short-term memory models; stacked autoencoder; wavelet transformation

Article

An Experimental Study Investigating the Effects of Data Pre-Processing Methods on Deep Learning Based Time Series Data Prediction

Mustafa Bugra Yilmaz [1,†,‡], Abdulkadir Karabacak [2,‡] and Mehmet S. Aktas [1,‡,*]

[1] Computer Engineering Department, Yildiz Technical University; bugra.yilmaz@std.yildiz.edu.tr, aktas@yildiz.edu.tr
[2] Intellica Research and Development Center; abdulkadir.karabacak@intellica.net
* Correspondence: aktas@yildiz.edu.tr
‡ These authors contributed equally to this work.

Abstract: Time-series analysis is a widely used technique across various fields and industries, as it helps in understanding, predicting, and forecasting the behavior of data points over time. These fields include but are not limited to finance, economics, healthcare, transportation, etc. In the case of this paper, we have focused on finance. Predicting future values of financial time series offers several benefits. Accurate forecasts can help investors make better decisions about their investments. To predict future values, deep learning algorithms are commonly used since it is an effective method with complex data. In this study, we conduct a study that investigates the use of different data pre-processing techniques on deep learning algorithms in predicting the time series values. To conduct this experimental study, we utilize an open source software, which uses long short-term memory technique as the representative deep learning technique, published in GitHub software code repository platform. With this study, we investigate the effects of autoencoder and discrete wavelet transform data pre-processing techniques in time-series prediction. We discuss the details of the experimental study and report our results. The results show that time series prediction (using backtesting methodology) without any data pre-processing leads to 12.6% for mean absolute percentage error. The results also show that time series prediction with the data pre-processing techniques (Wavelet Transform and Stacked Autoencoder) leads to a 3.4% MAPE error rate.

Keywords: experimental study on time series prediction methods; investigation on time series model predictions; analysis of the effects of data pre-processing methods on time series prediction; long short-term memory models; stacked autoencoder; wavelet transformation

1. Introduction

Time-series analysis is a critical technique that is widely used across various fields and industries, including finance, economics, healthcare, transportation, and more. It provides a means to understand, predict, and forecast the behavior of data points over time, and is particularly useful in making decisions about future investments. In finance, for example, predicting the future values of financial time series is crucial for investors to make informed decisions about buying and selling stocks, bonds, and other financial instruments.

To predict future values accurately, deep learning algorithms have proven to be an effective method for handling complex data[1,2]. These techniques have been applied to time-series prediction with promising results, especially with the use of long short- term memory (LSTM) networks, which are specifically designed to handle sequential data[3]. However, despite the widespread use of deep learning methods for time-series analysis, there is a lack of studies that apply and investigate the effect of data pre-processing techniques on the accuracy of time-series predictions.

In this context, this study aims to fill this gap in the literature by investigating the effects of different data pre-processing techniques on the prediction performance of deeplearning methods for

 doi:10.20944/preprints202310.1635.v1

time-series predictions. Specifically, we examine the use of two data pre-processing techniques, stacked autoencoder and discrete wavelet transform, and their effects on the prediction accuracy of LSTM networks.

Autoencoders are neural networks that are trained to encode input data into a lower-dimensional representation while maintaining the input's essential features. This could help extract the important parts of the time-series data. Discrete wavelet transform is a mathematical technique that decomposes time-series data into different frequency bands, to help smooth out the possible outliers in the data, which are found quite often in time-series. There are many more types of deep learning and classical methods that could be used as pre-processing steps, in this study, we focus on the effects of stacked autoencoder (SAE) and descrete wavelet transformation (DWT).

The primary objective of this study is to conduct an analysis on how data pre-processing technique affects the prediction accuracy of the deep learning networks for time-series prediction. To do so, we use a representative deep learning technique, LSTM, and apply it to real-world financial data. We utilize an open source software, which uses long short-term memory technique as the representative deep learning technique, published in GitHub software code repository platform. We evaluate the performance of LSTM networks with different data pre-processing techniques, including stacked autoencoders, which use multiple sparse autoencoder networks, and discrete wavelet transform. We compared the results of each combination to see how these pre-processing methods affect LSTM prediction accurately.

In conclusion, this study contributes to the literature by investigating the effects of data pre-processing techniques for improving the accuracy of deep learning methods for time-series prediction. The results show that time series prediction (using backtesting methodology) without any data pre-processing leads to 12.6% for mean absolute percent-age error. The results also show that time series prediction with the data preprocessing techniques (Wavelet Transform and Stacked Autoencoder) leads to 3.4%. We report our results in the discussion section in detail.

The paper is organized as follows. In Section 2, fundamental concepts for this paper are explained and a literature review on the subject is given in detail. In Section 3, the methodology of the experimental study approach is discussed in detail. Section 4 contains the prototype and the experimental results. In Section 5, we conclude the paper and discuss possible future work.

2. Fundamental Concepts and Literature Review

In this section, we provide insights into the fundamental concepts utilized within the scope of this research, along with a comprehensive review of the existing literature pertinent to the field of study.

2.1. Fundamental Concepts

Deep Learning: Deep learning is a subset of machine learning, which in turn is a branch of artificial intelligence. It revolves around the concept of using artificial neural networks, inspired by the human brain, to process and learn from vast amounts of data. These networks consist of multiple layers of interconnected nodes, also known as neurons, which allow the model to process and extract increasingly complex features from the input data. The process of deep learning involves feeding the neural network a large amount of labeled data to train it. During this training phase, the network adjusts its weights and biases to minimize the error between its predictions and the true labels. This optimization is typically achieved through back-propagation, a technique used to adjust the parameters in the network based on the gradients of the loss function with respect to each weight. Once the neural network is trained, it can be used for various tasks such as time series prediction, image recognition, natural language processing, and speech recognition, among others. Deep learning models have proven to be highly effective in these domains due to their ability to learn intricate patterns and representations from raw data, enabling them to surpass traditional machine learning algorithms in numerous applications.

Backtesting: Backtesting is a technique used in finance to evaluate a trading strategy or investment model by applying it to historical data. It simulates trades based on the strategy, testing

 doi:10.20944/preprints202310.1635.v1

its performance and reliability, which helps to identify potential risks and optimize the strategy before deploying it in the real market.

2.2. Literature Review

There exists research studies indicating that deep learning algorithms, particularly long short-term memory (LSTM), are widely used for time-series prediction in various fields, such as finance, healthcare, and transportation. However, despite the popularity of these methods, there is a lack of research that examines the effect of data pre-processing techniques on the accuracy of time-series predictions.

One study that utilized LSTMs for time series prediction is the work by Zhang et al. who proposed a model called Attention-based LSTM (AT-LSTM) for financial time series forecasting. Their model uses attention mechanisms to selectively focus on relevant inputs, allowing for better feature representation and improved prediction accuracy [1]. Another paper that used only LSTM compared their method to random forest (RAF), a deep neural net (DNN), and a logistic regression classifier (LOG) and found that LSTM results in higher returns. But they did not use other networks for pre-processing for LSTM [3].

Another deep learning model that has been used for time series prediction is the Convolutional Neural Network (CNN) [2]. CNNs are commonly used for image processing tasks, but they have also been shown to be effective in time series prediction. In particular, CNNs are well-suited for univariate time series data, where the time series is treated as a one-dimensional signal. One study that used CNNs for time series prediction is the work by Borovykh et al. [2] for predicting future values of financial data. They have found that CNNs can be time-efficient and easy-to-implement alternatives for RNNs, and they can effectively learn the relations between the time series.

In addition to LSTMs and CNNs, other deep learning models such as Deep Belief Networks (DBNs) have also been used for time series prediction. DBNs are a type of generative model that can learn hierarchical representations of data. One study that utilized DBNs for time series prediction is the work by Kuremoto et al., who proposed a model with Restricted Boltzmann Machine and Deep Belief Network. Their model uses a DBN to learn the temporal dependencies in the data [4].

Based on these previous studies, we argue that there is a lack of comprehensive analysis on the effects of data pre-processing for LSTM based deep learning networks. Overall, deep learning approaches have shown great promise in time series prediction tasks. However, there are still challenges that need to be addressed such as the selection of appropriate architectures, the handling of missing data, and the development of models that can handle high-dimensional and heterogeneous time series data. Nonetheless, the studies discussed in this literature review provide a strong foundation for further exploration and development of deep learning approaches for time series prediction.

In this study, we investigate the effects of data pre-processing steps on time series prediction. To do this, we introduce a data analysis process. There are studies focusing on distributed computing techniques and web services [5–16]. Different from these studies, we mainly focus on time series prediction data analysis process and leave out the distributed computing aspects of it as out of scope. There are data encoding methods used to understand users' actions to model the data in different domains [17–21]. In this study, we investigate the effects of stacked autoencoder and wavelet transformation data preprocessing techniques on time series prediction. To show the effects of data pre-processing techniques, this study provided an implementation. There are studies analyzing the quality of software systems [22,23]. However, in this study, we leave out software quality analysis for future work.

3. Methodology

This study proposes an experimental study methodology that investigates the effects of data pre-processing techniques in time series prediction. The proposed experimental study methodology is illustrated in Figure 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2023 doi:10.20944/preprints202310.1635.v1

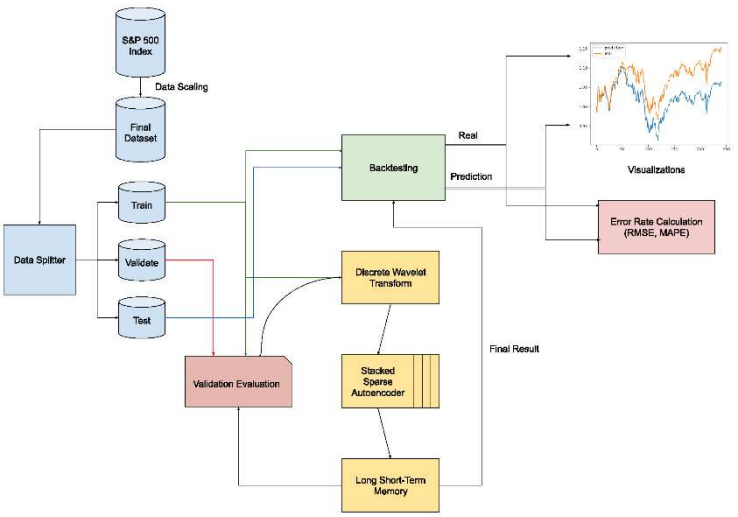

Figure 1. Proposed experimental study approach to investigate the effects of data pre-processing on time series prediction.

Below, we explain the details of the approach step by step in detail. The proposed approach implements a combination of three steps to predict the next step of a time series. 1st step is wavelet transform, which helps with getting rid of noise in the data. 2nd step is the stacked autoencoder, which, as the name implies, uses multiple autoencoders. Autoencoders help compress the data and reduce dimensions. And finally, 3rd step is the long short-term memory, which is a form of recurrent neural network.

It can capture and store information over long sequences, thus making it suitable for time-series predictions. Outputs of these steps are fed into the next one in order. Steps loop for every 3 months for data.

We discuss the details of the modules (Data Splitter module, Discrete Wavelet Transform module, Stacked Encoders Module, Long Short-Term Memory and Validation Module, Backtesting Module, Error Rate Calculation Module) used in the proposed experimental study below in detail.

3.1. Data Splitter Module

This module is responsible for splitting the data into three different data segments: train, validate, and test. The distribution is 90%, 10%, 10%, respectively. Data is split using library functions such as NumPy's split function.

3.2. Discrete Wavelet Transformation Module

This module is responsible for implementing the wavelet data transformation data pre-processing technique. The Wavelet Transform is a mathematical technique used for signal analysis, which enables the decomposition of a signal into different frequency components while preserving the time information. This is achieved by employing a set of basis functions, called wavelets, which are localized in both time and frequency. The Wavelet Transform provides a multi-resolution representation of the signal, making it particularly suitable for the analysis of non-stationary signals and revealing hidden patterns or features at various scales.

All 3 parts of the data are first deconstructed, then smoothed out against a threshold and reconstructed again. The output is then fed into the next algorithm, SAE or LSTM. To apply wavelet transforms to the validation data, we include all of the train data andappend past and current validation data. For the test, we include all of the training and validation data, then append the past and current test data.

3.3. Stacked Autoencoders Module

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2023 doi:10.20944/preprints202310.1635.v1

Stacked encoder is a type of unsupervised deep learning model, to effectively extract hierarchical features from the input data. Composed of multiple layers of encoders and decoders, the stacked autoencoder learns to reconstruct the input while minimizing the reconstruction error.

The data, depending on the model that is used, is given directly or by DWT, to the SAE. The type we use is sparse autoencoder, which improves the performance of the network for classification tasks[24]. A single network is built with 10 hidden layers. Then, 4 of these networks are used back to back. Then the results are fed into the LSTM network.

The 4 SAE networks are in train mode first. In the first step, the training data is fed into the network. And then, each next network uses the results of the previous networks.

After the training data is done, then the 4 networks are switched to evaluation mode. This mode has different properties than the training mode, e.g. dropout is disabled. Like with the train data, validate and test data are encoded in a chain of 4 networks. Just like the wavelet step, validate and test data will be used for evaluation, they will not be used in training.

3.4. Long Short-Term Memory and Validation Module

This module is responsible for modeling the time series data using deep learning techniques. Here, we use Long Short-Term Memory (LSTM) as the representative deep learning approach. LSTM is a specialized type of Recurrent Neural Network (RNN) architecture that effectively addresses the vanishing gradient problem[25], which is commonly encountered in traditional RNNs. This issue causes difficulties in learning long-range dependencies, hindering the network's performance when dealing with sequences. LSTM introduces memory cells, input gates, output gates, and forget gates that enable the network to retain and manage information over extended periods[26].

3.5. Backtesting Module

The aim is to do yearly backtesting. So, during LSTM training, since every step is 3 months, we accumulate data every 4 loops. After the training is done, we send these predicted and real yearly values to the backtesting function. This function compares each daily value in the predicted yearly values, against a threshold, which is 0 in this case, and simulates a buy/sell or does nothing depending on the comparison result. Each year's result is stored separately. When all years are done, it outputs the real and predicted values. These results are sent to create graphs and to calculate error rates.

3.6. Error Rate Calculation Module

We use Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) for the evaluation of the results. The backtesting results are separated year by year, and for each year, the error rates between the daily predicted and real values are calculated by the specified metrics.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(x_{\mathrm{i,pred}} - x_{\mathrm{i,real}})^2}{n}} \tag{1}$$

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|x_{\mathrm{i,real}} - x_{\mathrm{i,pred}}|}{x_{\mathrm{i,real}}} \tag{2}$$

In our study, by utilizing the software design and implementation illustrated in Figure 1, we conduct an experimental study to investigate the effects of data pre-processing techniques in time series prediction.

4. Prototype and Evaluation

In this section, we provide the details of the prototype application we developed to effectively investigate the impact of data preprocessing steps within our proposed business workflow, shown in Figure 1, for experimental study. Additionally, we elaborate on the specifics of the dataset

 doi:10.20944/preprints202310.1635.v1

employed for the experimental investigation and present an in-depth analysis of the research findings.

4.1. Prototype

To conduct this experimental study, we utilize an open source software, which uses long short-term memory technique as the representative deep learning technique, published in GitHub software code repository platform. The code is implemented in Python 3.9. JetBrains PyCharm was used to develop the code, which has very complex and well-developed auto-completion, analysis, etc. Numpy and Pandas were used for extended mathematical operations and helper data types, such as Dataframes. To use and implement deep neural networks, PyTorch was used. The code that we have based our project on was written by mlpanda on GitHub [27]. We designed our experimental study implementation using the aforementioned open-source project. We expanded on this project to be able to conduct this experimental study.

4.2. Dataset

For the experimental study, we used the same dataset introduced by the open-source time series prediction project that we utilized. The details of this dataset was initially discussed in [28]. Within this dataset, we utilize the S&P 500 Index data, which has 2080 days/rows and 21 features.

4.3. Test Design

In the test design, we apply backtesting as the testing methodology. Here, the data was split into 3 parts: training (90%), validation (10%), and testing (10%). Train, validate, and test data are prepared for LSTM. The target variable is changed to log return for train and normal return for validation and test data. Train data is shuffled. Train is for training the model, validation is used to oversee the performance of the model as the model is trained, and testing is for the final performance evaluation. Testing consists of 3-month periods, and it shifts by this window iteratively, for 6 years total. For evaluating the effects of data preprocessing techniques in time series prediction we used the following two evaluation metrics: RSME and MAPE.

Train data is given to the network, and during the training, validation data is used to measure the loss using MSE (Mean Squared Error). Then, the test data is given to this model. Output is collected and put into a list for every 4 iterations. Every iteration is approximately 3 months, so a year of data is collected. This is done 6 times. At the end, we have a list of lists that has 6 years of data, every year is separate.

After this step, the difference between these two prediction time series and real-time series are compared using the two evaluation metrics. We have chosen to use RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error). SciKit Learn library has these functions built-in, but they are simple enough that one can implement them themselves if no libraries are used.

4.4. Experimental Results

The results show that time series prediction (using backtesting methodology) without any data pre-processing leads to 12.6% for mean absolute percentage error. The results also show that time series prediction with the data preprocessing techniques (Wavelet Transform and Stacked Autoencoder) leads to 3.4% MAPE.

The results are illustrated in Table 1. These results indicate that the inclusion of SAE data pre-processing method improves LSTM prediction performance.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2023 doi:10.20944/preprints202310.1635.v1

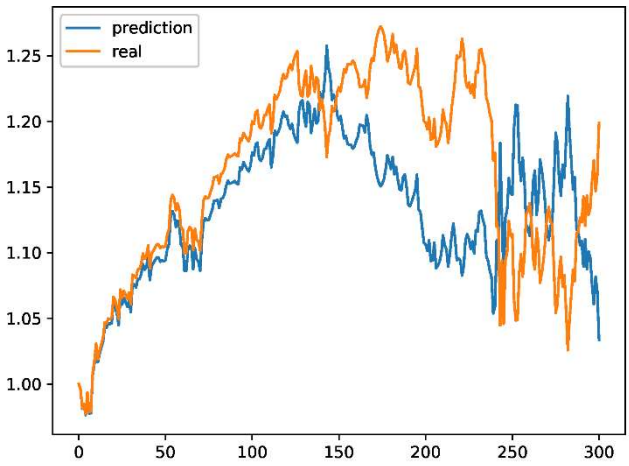

Figure 2. Year 1 for only LSTM.

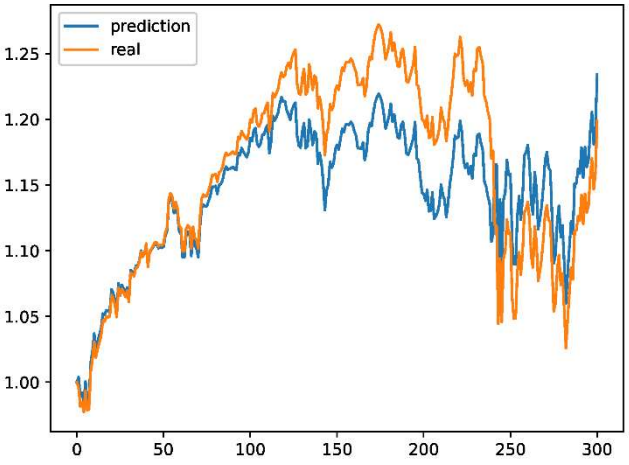

Figure 3. Year 1 for DWT-SAE-LSTM.

Table 1. Table for RMSE and MAPE metrics.

RMSE							
Algorithm	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Average
LSTM	0.174	0.116	0.227	0.165	0.095	0.030	0.134
DWT-LSTM	0.067	0.128	0.182	0.053	0.041	0.076	0.091
SAE-LSTM	0.040	0.014	0.054	0.069	0.031	0.035	0.040
DWT-SAE-LSTM	0.097	0.018	0.037	0.049	0.031	0.058	0.048
MAPE							
Algorithm	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Average
LSTM	0.147	0.103	0.223	0.163	0.095	0.024	0.126
DWT-LSTM	0.046	0.120	0.174	0.040	0.0301	0.061	0.079
SAE-LSTM	0.022	0.011	0.042	0.054	0.027	0.029	0.031
DWT-SAE-LSTM	0.049	0.013	0.028	0.038	0.027	0.050	0.034

The results indicate that stacked autoencoders have reduced the error rates from 12-13% range to 3-4%. Discrete wavelet transform, on its own, reduces the error rates to 8-9%.

5. Discussion

Overall, the results of the experimental study show that using any of the combinations of these methods seems to give better results compared to only using LSTM time series prediction on the dataset that we utilized.

On the dataset that we utilized, the results indicate that pre-processing can improve results for future time-series prediction. The choice of stacked autoencoder seems to be an effective one, due to its ability to extract relevant information, and also smooth out noise. Discrete wavelet transform did also reduce the error rates when used on its own because it reduces and smooths out noise in the data.

5.1. Threats to Validity of Results

For our experimental study, we utilized an open-source time series prediction software that is available on Git Hub for research. We used the software as-is without any modifications. We also utilized the dataset for investigating the time series dataset provided in the same GitHub repository. The validity of our experimental study is dependent on the open-source software and the dataset that we utilized. Although, in this study, we introduce a generic methodology on how one can investigate the effects of data pre-processing methods on time series prediction, our results are based on a specific open-source software and dataset. Hence, the experimental study can further be expanded by applying it to different time series prediction implementations and time series datasets.

6. Conclusion and Future Work

This paper investigated the effects of data preprocessing techniques on time series prediction. To do so, we utilized an LSTM-based time series prediction methodology and conduct an experimental study to demonstrate how the data pre-processing techniques affect the results. To illustrate the testing of our approach, we implemented a prototype im-plementation by leveraging an open-source time series prediction implementation available on a GitHub repository.

In summary, for the dataset that we used, the results indicate that time series prediction without any data pre-processing leads to 12.6% for mean absolute percentage error. The results also show that time series prediction with the data preprocessing techniques (Wavelet Transform and Stacked Autoencoder) leads to 3.4% MAPE.

There is much to do in the field of applying pre-processing for deep learning-based 297future value predictions. As for the future work, we plan on expanding this study by including additional techniques for data pre-processing, and investigate the best deep learning network parameters to get even more optimized results.

References

1. Zhang, X.; Liang, X.; Zhiyuli, A.; Zhang, S.; Xu, R.; Wu, B. AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction. *IOP Conference Series: Materials Science and Engineering* **2019**, *569*, 052037. https://doi.org/10.1088/1757-899X/569/5/ 052037.
2. Borovykh, A.; Bohte, S.; Oosterlee, C.W. Conditional Time Series Forecasting with Convolutional Neural Networks, 2018, [arXiv:stat.ML/1703.04691]. https://doi.org/10.48550/arXiv.1703.04691.
3. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* **2018**, *270*, 654–669. https://doi.org/10.1016/j.ejor.2017.11.054.
4. Kuremoto, T.; Kimura, S.; Kobayashi, K.; Obayashi, M. Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing* **2014**, *137*, 47–56. Advanced Intelligent Computing Theories and Methodologies, https://doi.org/10.1016/j.neucom.2013.03.047.
5. Aktas, M.S.; Fox, G.C.; Pierce, M.; Oh, S. XML metadata services. *Concurrency and Computation: Practice and Experience 20 (7), 801-823* **2008**.
6. Aktas, M.S.; Pierce, M. High-performance hybrid information service architecture. *Concurrency and Computation: Practice and Experience 22 (15), 2095-2123* **2010**.
7. Fox, G.C.; Aktas, M.S.; et al.. Real time streaming Data grid applications. *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements, 253-267, Springer* **2006**.
8. Tufek, A.; et al.. Provenance Collection Platform for the Weather Research and Forecasting Model. *2018 14th International Conference on Semantics, Knowledge and Grids* **2018**.
9. Aktas, M.S.; et al.. Fault tolerant high performance Information Services for dynamic collections of Grid and Web services. *Future Generation Computer Systems 23 (3), 317-337* **2007**.

10. Aydin, G.; et al.. Building and applying geographical information system Grids. *Concurrency and Computation: Practice and Experience 20 (14), 1653-1695* **2008**.
11. Aydin, G.; et al.. Algorithms and the Grid. *Computing and Visualization in Science 12, 115-124* **2009**.
12. Aktas, M.; et al.. ISERVO: Implementing the International Solid Earth Research Virtual Observatory by integrating computational grid and geographical information Web Services. *Computational Earthquake Physics: Simulations, Analysis and Infrastructure, Part II, 2281-2296* **2007**.
13. Pierce, M.; et al.. The QuakeSim project: Web services for managing geophysical data and applications. *Earthquakes: Simulations, Sources and Tsunamis, 635-651* **2008**.
14. Nacar, M.A.; et al.. VLab: collaborative Grid services and portals to support computational material science. *Concurrency and Computation: Practice and Experience 19 (12), 1717-1728* **2007**.
15. Dundar, B.; et al.. A Big Data Processing Framework for Self-Healing Internet of Things Applications. *2021 IEEE International Conference on Big Data (Big Data), 2353-2361* **2021**.
16. Aktas, M.S.; et al.. Implementing geographical information system grid services to support computational geophysics in a service-oriented environment. *NASA Earth-Sun System Technology Conference, University of Maryland, Adelphi, Maryland, 2005* **2005**.
17. Uygun, Y.; et al.. On the Large-scale Graph Data Processing for User Interface Testing in Big Data Science Projects. *2020 IEEE International Conference on Big Data (Big Data), 2049-2056* **2020**.
18. Olmezogullari, E.; Aktas, M.S. Pattern2Vec: Representation of clickstream data sequences for learning user navigational behavior. *Concurrency and Computation: Practice and Experience 34 (9)* **2022**.
19. Olmezogullari, E.; et al.. Representation of Click-Stream DataSequences for Learning User Navigational Behavior by Using Embeddings. *2020 IEEE International Conference on Big Data (Big Data), 3173-3179* **2020**.
20. Baeth, M.J.; Aktas, M.S. An approach to custom privacy policy violation detection problems using big social provenance data. *Concurrency and Computation: Practice and Experience 30 (21)* **2018**.
21. Baeth, M.J.; et al.. Detecting Misinformation in Social Networks Using Provenance Data. *2017 13th International Conference on Semantics, Knowledge and Grids* **2017**.
22. Sahinoglu, M.; et al.. Mobile Application Verification: A Systematic Mapping Study. *Computational Science and Its Applications- ICCSA 2015: 15th International Conference, Banff, AB, Canada, June 22-25, 2015, Proceedings, Part V 15, 2015* **2015**.
23. Kapdan, M.; et al..On the Structural Code Clone Detection Problem: A Survey and Software Metric Based Approach. *Computational Science and Its Applications–ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30–July 3, 2014, Proceedings, Part V 14* **2014**.
24. Makhzani, A.; Frey, B. k-Sparse Autoencoders. *International Conference on Learning Representations* **2014**, [arXiv:cs.LG/1312.5663]. https://doi.org/10.48550/arXiv.1312.5663.
25. Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München* **1991**, *91*.
26. Hochreiter, S.; Schmidhuber, J. LSTM can Solve Hard Long Time Lag Problems. In Proceedings of the Advances in Neural Information Processing Systems; Mozer, M.; Jordan, M.; Petsche, T., Eds. MIT Press, 1996, Vol. 9
27. mlpanda. DeepLearning_Financial, 2018.
28. Bao, W.; Yue, J.; Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE* **2017**, *12*, 1–24. https://doi.org/10.1371/journal.pone.0180944.