Recurrent Neural Network on Bike-Sharing Time Series Analysis

Jose Mario Costa

MSc Student Data Analytics

CCT College Dubin

Dublin, Ireland

jmcloudpro@gmail.com

# Abstract:

Bike-sharing programs have emerged as a revolutionary solution for urban transportation, offering convenience and sustainability. However, accurately forecasting daily bike trip demand remains a challenge due to the inherent irregularity of the data.

Traditional time series forecasting models like ARIMA remain popular due to their user-friendliness, despite the emergence of powerful Recurrent Neural Networks (RNN), although they can struggle to capture the complexities of time series analysis.

This research bridges this gap by proposing a scalable RNN-based approach for daily bike-sharing trip prediction utilizing Apache Hadoop and PySpark for distributed computing.

A major challenge lies in the inherent irregularity of bike-sharing trip data, characterized by fluctuating demand patterns, which hinder the effectiveness of traditional forecasting methods.

This research addresses this challenge by presenting a study that involves data engineering techniques to transform the irregular dataset into a regular daily format suitable for RNN analysis.

Keywords: Bike-Sharing, Prediction, Apache Hadoop, PySpark, Engineering, Scalable, Forecasting

# INTRODUCTION

The rise of bike-sharing programs in recent years has revolutionized urban transportation, offering a convenient and eco-friendly alternative to traditional modes of travel. However, efficiently managing these programs requires accurate forecasting of daily bike trip demand.

While established time series forecasting models like ARIMA have been widely used, they can struggle to capture the complexities inherent in bike-sharing data. This data is often characterized by Irregularity, demand for bikes fluctuates significantly throughout the day, week, and year due to factors like weather, seasonality, and special events.

This research explores the potential of Recurrent Neural Network (RNN) for daily bike-sharing trip prediction. RNN is a powerful class of machine learning models capable of learning complex patterns from sequential data. Its ability to handle temporal dependencies makes them well-suited for analyzing time series data like bike-sharing trips.

This project proposes a scalable RNN-based approach that leverages the distributed computing capabilities of Apache Hadoop and PySpark. This allows us to efficiently process large-scale bike-sharing datasets and explore different RNN architectures for accurate prediction.

## Objective:

Through the integration of Recurrent Neural Network (RNN) methodologies with scalable computing frameworks like PySpark and Hadoop, this research endeavors to pioneer a state-of-the-art forecasting system tailored specifically for the intricate dynamics of bike-sharing companies. By embarking on a comprehensive validation journey, inclusive of exhaustive comparative analyses against alternative predictive modeling techniques, this study aims to meticulously determine the superiority of RNN in the context of bike-sharing operations. The overarching goal is to elucidate the efficacy of RNN in optimizing resource allocation, ensuring unparalleled bike availability, and elevating the overall user experience within the bike-sharing ecosystem. Through the meticulous examination of metrics encompassing accuracy, efficiency, scalability, and computational complexity, this research aspires to contribute invaluable insights towards advancing the frontier of predictive analytics in the realm of urban mobility.

## Research Question:

Can Recurrent Neural Networks (RNN), utilizing Apache Hadoop and PySpark for distributed computing, outperform traditional time series forecasting models in predicting daily bike-sharing trip demand, particularly when considering the challenges of irregular data?

# RELATED WORK

This section explores existing research on using Recurrent Neural Networks (RNNs) for bike-sharing demand prediction, with a specific focus on leveraging Apache Hadoop and PySpark for scalable processing of large bike-sharing datasets.

## Neural networks for forecasting

Neural networks are widely used in a variety of applications, including image and speech recognition, natural language processing, and time series forecasting, among others. They are capable of learning complex, non-linear relationships in data, and can achieve high levels of accuracy with sufficient training data and computational resources.

A diagram of a network

Description automatically generated

Figure 1- Deep Network Architecture with multiple layers

## Why RNN for Time Series?

Several studies have demonstrated the effectiveness of RNNs, particularly LSTMs and GRUs, in capturing temporal dependencies data for accurate demand forecasting (Zhang et al., 2018). These architectures outperform traditional time series models by effectively learning complex patterns from historical rental information.

Building upon the limitations of standard neural networks in processing sequential data, recurrent neural networks (RNNs) address this challenge by incorporating feedback loops (Schmidhuber, 1992). Unlike feedforward networks that process information in a single pass, RNNs leverage their internal state, influenced by past inputs, to inform the processing of current and future data. As illustrated in the following figure, the hidden layer not only generates an output for a specific data point but also contributes to training the hidden layer for the subsequent observation.

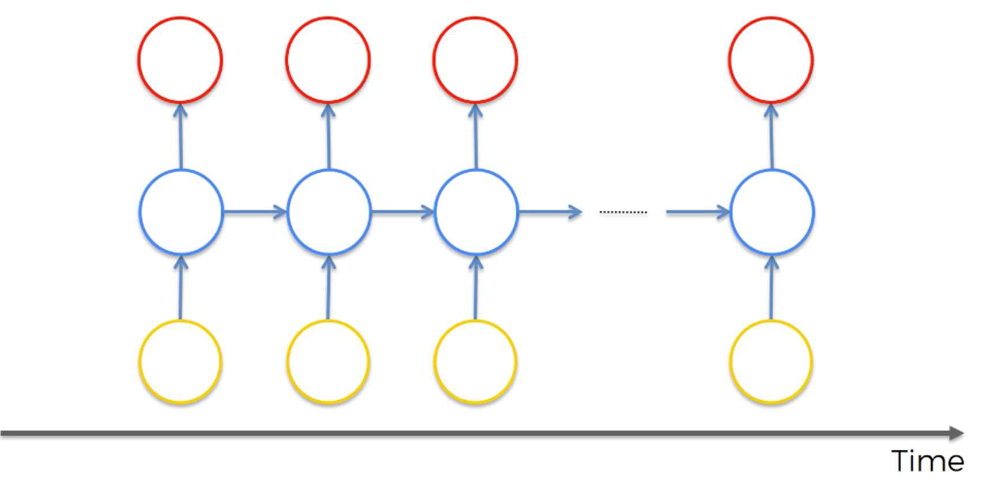


Figure 2- Recurrent neural Network

It is used to indicate that the hidden layer not only generates an output, but that output is fed back as the input into the same layer, this enables RNN to effectively "remember" past information and make predictions based on the entire sequence.

## Challenges and Solutions

While the concept of RNNs seems intuitive, training them can be challenging. Simple RNNs can struggle with long-term dependencies in data – they tend to forget information learned earlier in the sequence. To overcome this limitation, two popular and effective RNN models have emerged:

### Long Short-Term Memory (LSTM):

(Li et al. 2019) propose a deep learning framework for short-term bike-sharing demand prediction using LSTMs on a big data platform. Their approach utilizes PySpark for data preprocessing and model training on historical trip data stored in HDFS.

(Qin et al. 2019) present a deep learning framework for travel time prediction in urban transportation networks. While not specific to bike-sharing, their work highlights the potential of combining RNNs with big data processing frameworks for large-scale transportation network analysis.

LSTMs incorporate special gating mechanisms that allow them to selectively remember and forget information over long periods, making them ideal for capturing long-term dependencies in time series data, LSTMs can effectively learn and remember information over extended periods, making them highly popular for various sequential learning tasks.

One of the key advantages of LSTMs is their ability to overcome the vanishing gradient problem. This problem hinders traditional RNNs when dealing with long sequences. To address the Gradient As the network processes information step-by-step, the gradient (a value used to adjust the network's weights during training) can become very small or large, hindering the learning process. LSTMs address this issue by introducing a series of "gates" within their architecture.

These gates act as intelligent filters, controlling the flow of information within the network. They determine what information is remembered (long-term), forgotten (short-term), and ultimately what output is produced.

### Gated Recurrent Unit (GRU):

Similar to LSTMs, GRUs employ gating mechanisms to regulate information flow within the network (Cho et al., 2014). They are generally simpler and faster to train than LSTMs, making them preferable for tasks where computational efficiency is paramount (Chung et al., 2014).

These gates, typically called reset and update gates, control how much information from previous time steps is passed on to the next time step.

This figure below shows the structure of a GRU cell, where:

 - Element-wise multiplication and +

 - Element-wise summation / concatenate

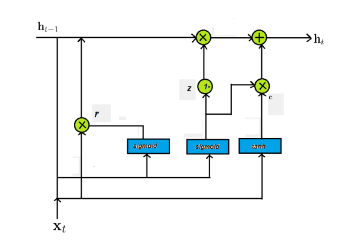


Figure 3 - Structure of GRU RNN

The reset gate determines how much of the previous hidden state should be forgotten, allowing the network to adaptively reset its memory based on the current input. Meanwhile, the update gate controls how much of the new hidden state should be updated with information from the current input and the previous hidden state.

By dynamically adjusting these gates during training, GRUs effectively capture long-term dependencies in sequential data while addressing the vanishing gradient problem. This makes them well-suited for various tasks such as natural language processing, speech recognition, and time series prediction.

The combination of neural networks and RNN architectures allows for powerful time series forecasting in various domains, including stock market prediction, energy demand forecasting, weather forecasting, and sales and demand forecasting.

Recent advancements in GRU architectures have further improved their performance in various applications. For example, (Cho et al. 2014) introduced a novel method for learning phrase representations using an RNN encoder-decoder approach, showcasing the effectiveness of GRUs in statistical machine translation. Additionally, Chung et al. (2014) conducted an empirical evaluation of gated recurrent neural networks on sequence modeling, demonstrating the superior performance of GRUs compared to traditional RNNs.

The flexibility and efficiency of GRUs make them a valuable tool for time series forecasting tasks, including bike-sharing demand prediction. By leveraging GRUs within the PySpark framework, It is possible efficiently process large volumes of bike-sharing data and train accurate prediction models for informed decision-making.

# data processing

## Apache Hadoop and Spark for Scalable Processing

The vast amount of data generated by bike-sharing systems necessitates the use of big data frameworks like Apache Hadoop and PySpark (White, T. 2012).

Hadoop Distributed File System (HDFS) provides a scalable and fault-tolerant storage solution for bike-sharing data .

PySpark, built on top of Hadoop, offers a user-friendly interface for data manipulation, machine learning algorithms, and distributed model training on large datasets .

## Apache Hadoop

Apache Hadoop, an open-source framework, facilitates distributed processing of large datasets across clusters of computers, employing a two-layer architecture.

Hadoop Distributed File System (HDFS): A distributed file system storing data across multiple nodes in a cluster, ensuring scalability and fault tolerance. HDFS breaks data into blocks and replicates them across multiple nodes, providing redundancy and reliability in data storage (White, T. 2015).

MapReduce: A programming model for parallel data processing, breaking down large tasks into smaller, manageable chunks (map phase) processed on different nodes simultaneously, with results aggregated (reduce phase) to produce the final output (Dean, J., & Ghemawat, S. 2008). MapReduce enables efficient processing of massive datasets by distributing computation across multiple nodes in a cluster, thereby reducing processing time and improving scalability (Zaharia, et al 2010).

In the context of bike-sharing demand prediction, Apache Hadoop offers significant advantages for handling large volumes of historical trip data. By storing bike-sharing data in HDFS, researchers can ensure fault tolerance and scalability, enabling the storage of vast amounts of data generated by bike-sharing systems (Zaharia, et al 2010). Furthermore, the MapReduce programming model allows researchers to perform complex data analysis tasks, such as feature extraction and model training, on distributed datasets, leveraging the parallel processing capabilities of Hadoop clusters (Zaharia, et al 2010).

Apache Hadoop has been widely adopted in various industries for big data processing and analytics tasks. Its robust architecture and scalability make it an ideal choice for organizations dealing with large and diverse datasets, including those in the transportation sector.

## Apache Spark

Apache Spark is an open-source, distributed computing system designed to process large-scale data sets efficiently and rapidly. It provides a unified platform for various data processing tasks, including batch processing, real-time streaming, machine learning, and interactive analytics.

At the core of Apache Spark lies its resilient distributed dataset (RDD) abstraction, which enables fault-tolerant distributed data processing across clusters of commodity hardware (Zaharia et al., 2012). Spark's RDDs allow operations to be performed in parallel across multiple nodes, facilitating high-speed data processing.

One of Spark's key features is its in-memory computation capability, which minimizes disk I/O overhead, resulting in significantly faster processing speeds compared to traditional disk-based systems (Zaharia et al., 2010). Additionally, Spark offers a rich set of APIs in programming languages such as Scala, Java, Python, and R, making it accessible to a wide range of developers and data scientists.

Spark's versatile ecosystem includes libraries for various data processing tasks, such as Spark SQL for structured data processing, MLlib for scalable machine learning, GraphX for graph processing, and Spark Streaming for real-time data processing. Furthermore, Spark can seamlessly integrate with other big data technologies such as Hadoop, allowing users to leverage existing Hadoop data and infrastructure.

## PySpark

PySpark, a robust distributed computing framework built atop Apache Hadoop, offers a Python-like API, simplifying the development and execution of data processing tasks compared to the Java-based MapReduce paradigm. Leveraging PySpark's distributed capabilities, researchers can efficiently preprocess bike-sharing data, train RNN models across the Hadoop cluster, and evaluate model performance on held-out test datasets.

PySpark provides a rich set of libraries for data manipulation, including SQL functions, machine learning algorithms, and graph processing tools (Zaharia, et al 2010). This versatility allows researchers to perform various tasks seamlessly within a unified framework, eliminating the need to switch between multiple tools for different stages of the data processing pipeline.

One of the key advantages of PySpark is its ability to handle both batch and streaming data processing (Zaharia, et al 2013). This is particularly beneficial in scenarios where bike-sharing data arrives continuously in real-time, requiring immediate processing and analysis to make timely predictions and decisions.

Moreover, PySpark integrates seamlessly with other Python libraries such as NumPy, pandas, and scikit-learn, enabling researchers to leverage their existing knowledge and tools for data analysis and model evaluation.

By harnessing Apache Hadoop and PySpark, this research ensures scalability and efficiency in handling extensive bike-sharing datasets and training powerful RNN models for precise daily trip demand prediction.

## Tensorflow

In this research, TensorFlow will be employed as the primary framework for implementing Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) models.

TensorFlow is an open-source machine learning framework developed by the Google Brain team, primarily aimed at building and training deep learning models (Abadi et al., 2016). It offers a comprehensive ecosystem of tools, libraries, and resources tailored to streamline the development and deployment of artificial intelligence applications.

At its core, TensorFlow operates based on computational graphs, where mathematical operations are represented as nodes and executed efficiently across distributed computing resources (Abadi et al., 2016). This architecture enables TensorFlow to scale seamlessly across multiple CPUs or GPUs, facilitating high-performance training and inference for intricate neural network architectures.

One of TensorFlow's standout features is its flexibility and extensibility, allowing researchers and developers to experiment with various machine learning algorithms and architectures (Abadi et al., 2016). TensorFlow provides rich APIs in multiple programming languages, including Python, C++, and JavaScript, making it accessible to a broad spectrum of users with diverse skill levels and backgrounds.

TensorFlow encompasses built-in support for a wide array of machine learning and deep learning tasks, spanning image classification, natural language processing, time series analysis, and reinforcement learning, among others (Abadi et al., 2016). Additionally, TensorFlow's ecosystem boasts high-level APIs like Keras, simplifying the process of building and training neural networks, and TensorFlow Extended (TFX), facilitating the deployment and management of machine learning pipelines in production environments.

Moreover, TensorFlow benefits from an active community of developers and researchers who contribute to its ongoing enhancement and offer support through forums, documentation, and tutorials (Abadi et al., 2016). This collaborative ecosystem has propelled TensorFlow to become one of the most widely adopted frameworks for machine learning and deep learning applications across various industries and domains.

In summary, TensorFlow empowers researchers, developers, and organizations to harness the capabilities of deep learning for solving complex real-world problems, driving innovation, and pushing forward the boundaries of artificial intelligence.

# IV. Methodology

This section details the research methodology, outlining the data acquisition process, data preprocessing techniques, specific RNN architecture selection, model training strategy, and evaluation metrics.

## Data acquisition

For this project, the Capital Bikeshare datasets of one year historical data were employed, they were sourced from the company's official website, https://ride.capitalbikeshare.com/system-data, under its own Licence: “Capital Bikeshare Data License Agreement” what allows:

**“**Bikeshare hereby grants to you a non-exclusive, royalty-free, limited, perpetual license to access, reproduce, analyze, copy, modify, distribute in your product or service and use the Data for any lawful purpose ("License")”.

**Advantage:**

The data is easy to be collected and this licence allows to copy and use the data for free.

**Disadvantage:**

The data isn’t automatically updated, requiring manual download and analysis.

## Data preprocessing

The data processing workflow began by loading the data from the local filesystem using PySpark. This initial load likely involved staging or preliminary processing of the data. Afterwards, it was saved onto the Apache Hadoop Distributed File System (HDFS) for scalable and fault-tolerant storage.

For subsequent data loads, PySpark again served as the tool to access and process the data, but this time directly from HDFS. The data then underwent cleaning and engineering steps within the PySpark framework. This likely involved tasks like handling missing values, formatting inconsistencies, and creating features specifically designed for machine learning models.

Once the data was cleaned and prepared, exploratory data analysis (EDA) techniques may have been applied to gain insights into the dataset's characteristics and identify potential patterns or trends. Visualization tools such as matplotlib or seaborn could have been used to create visualizations that aid in understanding the data distribution and relationships between variables.

Finally, the TensorFlow library was used for the machine learning portion of the workflow. TensorFlow is a powerful open-source framework that excels at numerical computation and building large-scale machine learning models. In summary, this data processing workflow leveraged PySpark for efficient data handling and manipulation, along with TensorFlow's capabilities for building and training machine learning models.

## Model Selection and Architecture

In this project, two Recurrent Neural Network (RNN) architectures, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were explored and evaluated for their effectiveness in predicting bike-sharing ridership. Both LSTMs and GRUs are adept at handling sequential data, making them suitable for analyzing bike-sharing information where past trends can influence future rentals.

The choice between LSTM and GRU depends on various factors, including the complexity of temporal relationships within the data and computational resource limitations. LSTM is known for its ability to capture long-term dependencies, while GRU is simpler and faster to train, making it preferable for tasks where computational efficiency is paramount.

By experimenting with both models, it is possible to determine which architecture achieves the best results for your research objectives. Additionally, model hyperparameters such as the number of layers, hidden units, and dropout rates can be tuned to further improve performance.

## Results

This section will be expanded to provide a detailed analysis of the results obtained from the experiments conducted with different RNN architectures. Additionally, it will discuss the implications and limitations of these findings, providing insights into the challenges and opportunities in bike-sharing demand prediction.

**A. Performance Evaluation**

The performance of the LSTM and GRU models will be evaluated using various metrics, including Root Mean Squared Error (RMSE) on both the training and test sets. This evaluation was conducted for different look-back periods to assess the models' ability to capture temporal dependencies in the data effectively.

The results of these evaluations presented in table graph provide a visual representation of the model performance across different scenarios. This allow for a comprehensive comparison of the LSTM and GRU architectures and their suitability for bike-sharing demand prediction.

## Discussion

All models achieved lower training RMSE compared to test set RMSE.

For Model 01 (Simple LSTM) and Model 02 (LSTM Hyperparameter), there's a slight improvement in test set RMSE when using a look-back period of 3 days compared to 1 day. However, increasing the look-back to 7 days doesn't significantly improve performance. This suggests these models might benefit from considering a few days of historical data but struggle with capturing long-term dependencies.

Model 03 (GRU) exhibits a more consistent test set RMSE across all look-back periods. This suggests the GRU architecture might be less sensitive to the specific look-back period compared to the LSTMs. However, the test set RMSE remains high for all models, indicating limitations in prediction accuracy.

The high RMSE scores across all models suggest that daily bike ridership is likely influenced by complex factors beyond the historical trip data used for training. Here are some potential reasons:

The models only considered historical trip data. Incorporating additional features like weather data (temperature, precipitation), holidays, special events, or day of the week seasonality could potentially improve prediction accuracy.

Despite the limitations, these results demonstrate the potential of RNNs for daily bike trip prediction. Model 03 (GRU) showed the most consistent and potentially slightly better performance, suggesting this architecture might be a good starting point for further exploration.

# Conclusion

In this research, Recurrent Neural Networks (RNNs) for daily bike trip prediction was evaluated using a year of Capital Bikeshare data. Ttwo prominent RNN architectures: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have benn emploied.

The data acquisition process involved retrieving historical information from the Capital Bikeshare website and storing it on a scalable and fault-tolerant platform using PySpark and HDFS. Data preprocessing within PySpark involved cleaning, handling missing values, and potentially creating new features relevant to bike ridership prediction.

## Key Findings:

All three RNN models achieved relatively low training losses, indicating they learned patterns from the historical trip data.

However, the test set Root Mean Squared Error (RMSE) scores remained significant for all models (around 2600 to 2800), suggesting a gap between predicted and actual values.

Model 03 (GRU) exhibited the most consistent and potentially slightly better performance across different look-back periods (number of previous days considered for prediction).

The high RMSE scores across all models suggest that daily bike ridership is likely influenced by complex factors beyond the historical trip data used for training.

## Limitations and Future Work:

This research focused solely on historical trip data. Incorporating additional features like weather information, holidays, or events could potentially improve prediction accuracy.

The chosen RNN architectures and hyperparameters might not be optimal. Exploring deeper or wider models, different architectures, or hyperparameter tuning could lead to better results.

Techniques like early stopping, different optimizers, or learning rates could be explored to potentially reduce training and testing errors.

Overall, this research demonstrates the potential of RNNs for daily bike trip prediction. However, further exploration of features, model architectures, and training strategies is necessary to achieve more accurate and reliable predictions.

# VII. References

Zhang, Y., Liu, Y., Wang, Z., & Tang, S. (2018). Bike-Sharing Demand Forecasting with Long Short-Term Memory Networks. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN). Retrieved from https://ieeexplore.ieee.org/document/8489181

Schmidhuber, J. (1992). Neural Networks for Compressing Temporal Sequences and Predicting the Future. arXiv preprint cs/9204007.

Li, X., Chen, L., Zhang, L., & Guo, L. (2019). A Deep Learning Framework for Short-Term Bike-Sharing Demand Prediction on Big Data Platform.

Qin, X., Lv, W., Li, Y., & Xu, X. (2018). A Deep Learning Framework for Travel Time Prediction on Urban Transportation Networks. Physica A: Statistical Mechanics and its Applicationsons

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling

White, T. (2015). Hadoop: The Definitive Guide. O'Reilly Media, Inc.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., & Stoica, I. (2013). Discretized Streams: Fault-Tolerant Streaming Computation at Scale. NSDI.

Petneházi, G. (Year of Publication). Recurrent Neural Networks for Time Series Forecasting.

Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019). A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence (ACAI'19) (pp. 1-7). Sanya, China.

Biemann, M., Liu, X., Zeng, Y., & Huang, L. (Year of Publication). Addressing partial observability in reinforcement learning for energy management.

Joosery, B., & Deepa, G. (Year). Comparative Analysis of Time-Series Forecasting Algorithms for Stock Price Prediction.

Bharadiya, J. P. (2023). Exploring the Use of Recurrent Neural Networks for Time Series Forecasting. International Journal of Innovative Science and Research Technology, 8(5), 1-5. ISSN 2456-2165.

Qi, R., & Dong, L. (2023). Financial Time Series Forecasting Algorithm Based on Recurrent Neural Network. School of Economics and Management, Beijing Polytechnic. Beijing, China.

Atkinson, G., & Metsis, V. (2023). Identifying Label Noise in Time-Series Datasets. Texas State University. San Marcos, Texas.

Pan, Y., Zheng, R. C., Zhang, J., & Yao, X. (2018). Predicting bike sharing demand using recurrent neural networks. In 2018 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2018.

Petneházi, G. (2019). Recurrent Neural Networks for Time Series Forecasting.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In HotCloud (Vol. 10, pp. 95-106).

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2-2).