Deep Learning Using Big Data Technologies

Andrew Mc Guinn   
 *CCT College Dublin*  
 *MSC Data Analytics*Word Count: 4124 words   
sba24458@student.cct.ie

*Abstract*— Stock prediction is a critical yet challenging task in financial markets due to inherent volatility and complexity of price movements. This study explores the integration of Hadoop, Apache Spark and deep learning techniques for predicting stock prices using historical data. Hadoop is utilized for distributed data storage, while Apache Spark enables large scale data preprocessing, distributed computing and efficient handling of vast datasets, ensuring scalability and performance improvement in training. Deep learning models, particularly recurrent neural networks (RNN’s) and long-short term memory (LSTM) networks, are implemented using TensorFlow, demonstrating their superior ability to capture temporal dependencies, showing superior performance to prior machine learning models. TensorFlow was utilized to develop learning models, including recurrent neural networks and long short-term memory networks. In this task python libraries are used to automatically download historical data used for forecasting with over 10 years of records being analyzed.

Keywords—RNN, LSTM, Hadoop, Apache Spark,

TensorFlow, Stock price prediction, distributed computing.

# Introduction

The ability to predict stock prices with accuracy is a crucial challenge in financial markets, where even minor fluctuations can have significant economic impacts. Stock prices are influenced by a number of factors, including markets trends, investor sentiment, economic indicators and historical price movements (Hall, 2024). Traditional statistical models often struggle to capture the complex, non-linear dependencies within financial time series data. In recent years ,deep learning models such as recurrent neural networks (RNN) and long-short term memory (LSTM) networks have demonstrated superior performance in time series forecasting (Datamount, 2019).

In the era of big data, the ability to process and analyse and analyse massive datasets efficiently has become crucial. Two of the most powerful frameworks that have revolutionized distributed computing are Apache Hadoop and Apache Spark. Hadoop with its Hadoop distributed file system (HDFS) and MapReduce processing model, laid the foundation for scalable fault-tolerant data storage and batch processing. However, its reliance on disk I/O made it less suitable for real-time and iterative computations (Mehta, 2024).

Apache Spark emerged as a game changer, offering an in-memory model that significantly accelerates data processing. Unlike Hadoops disk-based approach, Spark leverages resilient distributed datasets (RDD) and optimised directed acyclic graph (DAG) execution, making it up to 100x faster for iterative machine learning tasks. As big data frameworks continue to evolve, their synergy with deep learning has opened new frontiers in Sparks distributed computing capabilities, coupled with deep learning frameworks which enable efficient training and deployment of neural networks on massive datasets. The intersection of big data and deep learning has transformed industries enabling breakthroughs in natural language processing, image recognition and predictive analytics.

The primary objective of this research is to explore the optimisation of deep learning models for big data processing and analysis. This study aims to investigate how deep learning techniques can effectively integrated with big data frameworks like Apache Spark and Hadoop to enhance computational efficiency, scalability and real time data processing. By leveraging distributed computing and advanced deep learning architectures, this research seeks to develop strategies that improve model training speed, resource utilization and predictive accuracy when working with big datasets.

“How can deep learning models be optimized for efficient processing and analysis of big data using distributed computing frameworks like Apache Spark and Hadoop”

# Literature Review

## State of the Art

Traditional data processing relied heavily on legacy systems and relational databases, which were designed for structured data and transactional workloads. Relational Database Management systems (RDBMS), such as SQL, PostgreSQL and Oracle offered consistency and reliability but struggled with sheer volume, variety and velocity for big data (Robinson, 2023). These systems were built on centralised architectures, making them inefficient for handling distributed, high volume and unstructured data commonly found. To address these issues, NoSQL databases emerged, providing scalability and schema flexibility, however as data continued to grow exponentially, even NoSQL databases faced limitations in handling these massive processing and complex deep learning workloads .

The introduction of Hadoop revolutionised Big Data by providing a distributed storage and processing framework based on the MapReduce model. Hadoops HDFS (Hadoop distributed File System) allows large scale data storage across multiple nodes, ensuring fault tolerance and cost effective scalability. However its reliance on disk-based storage resulted in high latency, making it inefficient for real-time processing. To overcome this, Apache Spark emerged leveraging in memory computation to process data up to 100 times faster than MapReduce (Ojha, 2023) .

A key limitation of traditional Big Data frameworks is their inability to handle complex deep learning workloads efficiently. Hadoops disk-based storage system introduces latency in training deep learning models, whereas sparks memory intensive approach can become a constraint for extremely large datasets. This has led to hybrid architectures that combine distributed processing with GPU accelerated deep learning frameworks such as TensorFlowOnSpark enabling large scale model training.

Recent innovations have also focused on edge computing and federated learning to overcome centralised data processing limitations (Preeti, 2024). By distributing computation closer to data sources, these approaches enhance real-time analytics and reduce dependency on cloud infrastructure. Additionally, the integration of Graph Neural Networks (GNN’s) and Federated deep learning has enabled more scalable and decentralized big data processing, making it more possible to analyse to analyse interconnected datasets.

Traditional machine learning approaches struggle with sequential and time-dependent data, failing to capture long range dependencies in financial forecasting, speech recognition and sentiment analysis. RNN’s were introduced to model such temporal patterns, but they suffer from the vanishing gradient problem, making it difficult to retain long-term contextual information. To mitigate this. LSTM networks were introduced by Hochreiter and Schmidhuber (Sepp Hochreiter, 1997), revolutionised sequential modelling by incorporating gating mechanisms that regulate information flow. Despite their advancements, LSTM’s exhibit limited scalability in distributed environments, requiring high computational resources that make them less practical for real-time Big Data applications.

Recent innovations, such as transformer-based architectures (e.g., Bert, GPT-4 and T5) (Mousa, 2023) have surpassed LSTM’s in both performance and efficiency. Self-Attention mechanisms in Transformers eliminate the need for sequential processing, allowing for parallelised training on massive datasets- a crucial advantage in Big Data. While Transformers have been a significant breakthrough in sequential modelling, LSTM’s remain relevant for certain time-series applications.

## Research Methodologies and Critical evalutaion

The research (Sharma & Kaur, 2019) primarily evaluates the performance of two Big Data processing frameworks: Apache Hadoop’s MapReduce and Apache Spark, focusing on their efficiency and response time in handling queries on-disk and in-memory processing. Data preprocessing involved filtering and grouping large datasets, specifically taxi trip records, to prepare for query execution. The study highlights that while Hadoop’s MapReduce is more suited for batch processing tasks due to its reliance on disk operations, it struggles with real time query performance due to its higher latency. The key findings emphasize that Spark outperforms Hadoop in scenarios demanding low latency data access which is essential for time-sensitive applications.

This paper (Aung, 2024) focuses on predicting stock prices from 51 companies using a deep learning approach, specifically LSTM. The data spans from 2000 to 2021, totalling over 260,000 stock price data points. The evaluation of the prediction performance is done using Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) with the results showing that LSTM can predict stock prices with over 90% accuracy. The Big Data analytics section explains how deep learning models like LSTM can analyse large datasets in real-time, extracting useful patterns. While platforms like kaggle and Dataworld have their use cases, the data from these platforms may be incomplete or inconsistent, in regards to deeper actions such as stock splits, dividends or market events.

Deep learning models, particularly when trained on extensive datasets like the one presented in the research, are prone to overfitting. While LSTM’s excel at learning patterns from time-series data, they can also learn noise or irrelevant patterns, the metrics used would not be sufficient to assess overfitting, particularly in cases where the model performs well on the training data but poorly on unseen data.

The architecture in (Sushree Dasa, 2018) provides a comprehensive approach to real-time data processing for stock market prediction, incorporating large scale data management, sentiment analysis and RNN’s. The explanation of the Lambda architecture is well structured and the separation into batch, speed and serving layers is well explained. The paper effectively integrates several data processing and storage tools such as Twitter API, Apache Flume Spark and HDFS for data ingestion, processing and storage. The use of these tools ensures the model is both scalable and efficient for real time data processing.

The use of sentiment analysis as a method of assessing public opinion towards companies and relating it to stock prices is a ground breaking approach as it incorporates RNN’s ability to process sequential data helps in the temporal nature of twitter data. Overall the Lambda architecture, combined with Spark is designed to process massive amounts of data while ensuring fault-tolerance and scalability, however the paper lacks a detailed evaluation of model performance and error analysis, as well as more attention to comparing results among similar model architectures.

This paper (Indirman, 2023) provides a detailed understanding how to implement efficient resource management when working with distributed systems. The paper utilised Linear Regression (LR), Multiple Linear Regression (MLR) and LSTM for machine learning on both single node and multi-node clusters and once uses the metrics RMSE and MAPE for model execution time, that is, the time the model took to complete training and testing.

In the single node cluster configuration, the execution time for LR and MLR remained relatively similar, with LR performing faster due to its simpler model structure. However, the LSTM model showed significantly longer execution time due to its iterative training process. The execution tims for LR and MLR were not significantly impacted by core allocation, indicating that a single core could be sufficient for small datasets. However, LSTM demonstrated a significant decrease in execution time as the number of cores increased, though this improvement plateaued after 12 cores.

When comparing the multi node cluster with 2 nodes, it was observed that the execution time decreased as the number of cores increased. With 3 nodes the execution time for the Bitcoin price prediction showed a noticeable reduction with a 6 core allocation, suggesting better efficiency with the additional node. The RMSE values were generally lower for LSTM models across all configurations, demonstrating their superior prediction quality compared to LR and MLR. The MAPE values did not show significant changes based on the number of cores, which suggests that the quality of the prediction was largely unaffected by parallelisation.

The research does not benefit from Sparks distributed computing resources as LSTM’s are inherently sequential due to their dependence on previous time-step outputs, which limits parallelisation. The paper focuses on resource allocation but overlooks the lack of support for LSTM in Spark’s MLlib. The paper may benefit from a more detailed exploration of alternative deep learning frameworks and models, as well as a critical discussion on the trade-offs between prediction quality and computation time. Additionally, further exploration of ways to optimize the execution and hyperparameters of the LSTM could improve overall performance.

In (Sutradhar, et al., 2021) employed LSTM networks for stock price prediction, it lacks consideration for big data scalability, real- time processing and alternative deep learning architectures. The study uses RMSE as the sole performance measure, but it is scale dependent and does not always provide a clear picture of model accuracy, other evaluation metrics such as Mean Absolute Error (MAE) or even MAPE should be included to assess different aspects of model performance. Hyperparameter selection is not well explained, while the batch size of 64 and 256 and the number of epochs of 100 is mentioned, there is no justification for these choices. The paper gives a clear and comprehensive overview of RNN’s and LSTM’s architecture and it discusses the ADAM optimiser as well as emphasis on regularisation techniques for overfitting.

The use of vanilla RNN’s, LSTM’s and GRU’s by (Hussain, et al., 2018) is justified well by explaining the architectures as well as the vanishing gradient problem experienced by RNN’s. The use of a Spark cluster for parallel processing and scaling the models for handling large data volumes, showcasing an efficient way of processing Big Data. The selection of hyperparameters and optimisation via Sparks distributed nature is a great experimental design choice. Hyperparameter tuning revealed that a 3-layer, 20 node architecture yielded optimal performance for all models which can reduce training time.

The paper (Sarma, 2023) explores the integration of multiple data sources shows an understanding of Big Data., the paper uses DataBricks community platform with 15.3 GB of memory and 2 cores . This setup is efficient for small datasets but efficiency can vary when dealing with unstructured data for sentiment analysis. The usage of Apache Spark and Pyspark for parallel processing of large datasets is critical for Big Data, the paper does not elaborate on the data preprocessing steps. While the paper acknowledges advanced deep learning models, there is no justification for choice of models or hyperparameter tuning, the paper uses k-fold cross validation and Gridsearch CV for model optimisation but lacks any depth into these areas. The paper employs a wide range of evaluation metrics giving it optimal feedback of the models and implementation.

Th researchers in (Mohapatra, et al., 2019) leveraged real-time adaptive cryptocurrency price prediction leveraging Twitter sentimental analysis. Built on Apache Spark, it claims to efficiently handle high-volume financial data while ensuring fault tolerance and persistence. By utilizing Vader for sentiment analysis and an online approach for model adaptation, the system aims to improve predictive accuracy, firstly the reliance on Twitter data may introduce bias, as influential tweets do not always reflect actual market trends and it also lack contextual understanding, potentially misinterpreting sarcasm or complex financial discussions. Additionally, Sparks RDD based architecture enhances fault tolerance and scalability, it may be optimal for real time processing compared to alternative frameworks, as well as the study also lacks a robust benchmark against alternative models.

This paper (Mehendale, 2019) presents a compelling discussion on the orle of deep learning in processing big data for financial data. The integration of machine learning (ML) and distributed computing frameworks, such as Apache Spark and Kafka, demonstrates an understanding of scalability challenges in financial markets. Additionally, the paper acknowledges the advantages of deep learning in detecting complex, non-linear patterns within financial data.

However, the paper lacks specificity in its discussion of deep learning models. While it references deep learning techniques, it does not justify why certain architectures (e.g., LSTMs, GRUs, Transformers) are preferred over traditional ML methods like Random Forest or XGBoost. A benchmarking analysis comparing different approaches would strengthen its claims. Furthermore, deep learning models require large amounts of labeled data, which is often a limitation in financial applications. The paper could address this challenge by discussing techniques such as transfer learning or self-supervised learning.

This paper (Aditham, et al., 2017) addresses the growing concern of security in Big Data platforms, highlighting the challenges of preventing cyberattacks while managing vast datasets. It emphasizes the traditional security measures used in Big Data, such as access control, encryption and activity logging, but critiques their inadequacy in protecting against insider and control flow attacks. The approach is innovative, the paper proposes a novel approach for predicting attacks within big data systems by leveraging LSTM’s. By monitoring memory patterns of datanodes, the proposed method uses LSTM’s to predict abnormal behaviour that may signal potential threats. The discussion of insider threats as primary security risks is valid, but the paper could explore other potential vulnerabilities. Despite these limitations, the proposed method provides a promising direction for using deep learning in the realm of big data security.

# Methods

## Methodology for data acquisition

Tiingo provides access to financial data via its API, which is available to registered users. The free plan is designed for non-commercial use, making it suitable for small scale academic projects, the conditions are outline below.

A close-up of a data

AI-generated content may be incorrect.

Figure :Data License

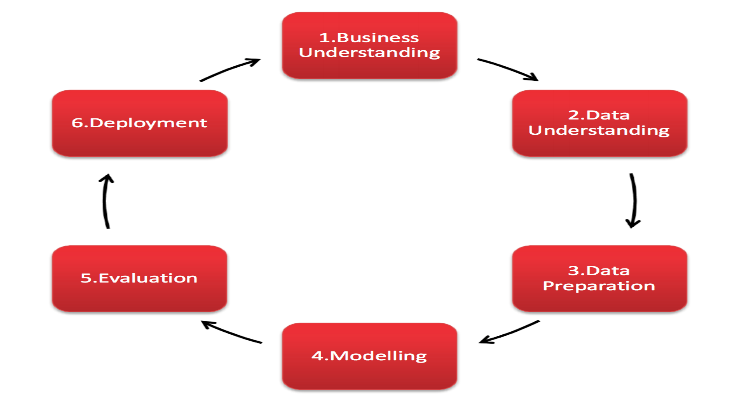


Figure : CRISP Framework

Applying the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework will guide the project in a structured way. This framework allows for clear steps, starting from understanding the problem, preparing the data, building models and evaluating the results.

## Methodology for data storage and processing

In the modern data ecosystem, the storage and management of Big Data require scalable, fault tolerant and high-performance systems. Legacy approaches such as traditional databases are often ill-suited for Big Data, which tends to involve vast volumes, high velocity and variety (Petrova, 2025). For instance, RDBMS cannot efficiently handle large datasets or real time streaming data, as they are not designed to scale horizontally. In contrast, modern Big Data systems leverage distributed storage (HDFS) and distributed frameworks e.g. Spark, which enable the parallel processing of data across clusters of machines.

The dataset is uploaded and stored to HDFS as a CSV file with 128MB blocks on my local cluster for processing, the distributed storage solution allows for redundancy and fault tolerance, meaning that if a node fails, other copies of the data are available for processing. Managing large datasets on HDFS enables efficient parallel processing and storage, however, this approach introduces challenges in terms of data consistency, synchronization and latency, which must be carefully managed to ensure efficient operations.

Once the data was uploaded to HDFS, SparkContext was initiated through the driver program, this enabled parallel processing of the data in local mode on the cluster. In this setup, Spark will utilize all available CPU cores indicated by ‘local[\*]’, there is no cluster manager such as YARN or in this configuration, instead Spark runs in a single JVM instance on the local machine, leveraging the machines resources for parallel execution. The Spark UI provides runtime monitoring and job execution details, it can be accessed through ‘http://localhost:4040’.

In optimising the configuration of Spark for this dataset, key adjustments are made to enhance performance, particularly in memory management and parallelism. The executor and driver memory were set to 2GB to ensure adequate resources while avoiding memory overflow. Allowing two cores per executor allows better resource utilization and can handle multiple tasks simultaneously. By setting the shuffle partitions to 50, which aims to optimise the shuffling process during stage transitions reducing the overhead. The default parallelism is set to 50, enhancing the degree of parallel execution across the spark cluster. For further optimisation, the Spark dataset was cached to avoid re-reading the data from disk for every part of EDA. The schema defines the data types for each column, such as DateType, ensuring proper handling of the data, descriptive statistics were conducted to gain valuable insights into the data, outlier analysis as well as other processes such as checking missing values was conducted, no missing values were found, the data was scaled between -1 and 1 through MinMaxScaler, the data was scaled after splitting to avoid data leakage (Sutradhar, et al., 2021).

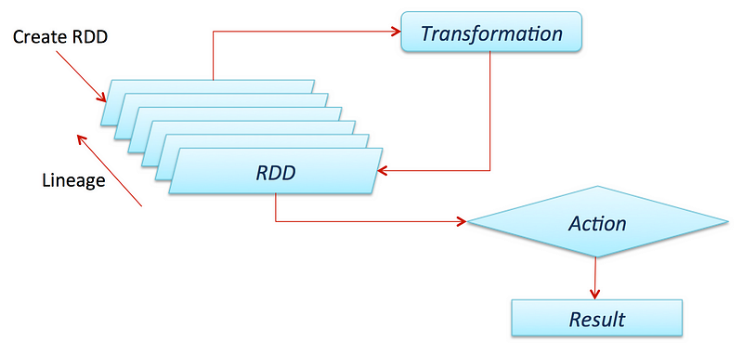


Figure : RDD process

In the project, Resilient distributed datasets (RDD) are a fundamental data structure, designed to be fault-tolerant, distributed and immutable, the process is outlined in figure 3. By converting the dataset to an RDD, operations such as calculating daily price changes in stock prices using the MAP and FILTER functions are enabled. The MAP function allows for element-wise transformations, while the FILTER function is used to isolate high-volume trading days seen in figure 4. These transformations are lazy, meaning they do not execute until an action such as ‘collect’ triggers the computation. This delay optimises performance and allows Spark to build a Directed Acyclic Graph (DAG) of stages as seen above titled ‘lineage’, these processes can be altered and adapted for increased data size.

A graph of a number of high treading days

AI-generated content may be incorrect.

Figure : Top 10 high volume days

## Methodology for Deep Learning and results

\*\*The original methodology was to use sparks MLLIB but due to technical difficulties, this was not possible, so Deep learning was conducted in a Windows machine using TensorFlow\*\*

A graph showing a line graph

AI-generated content may be incorrect.

Figure :Stock Close price

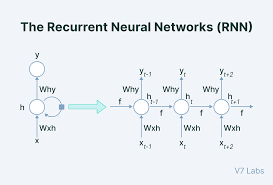


Figure :RNN Architecture

The first deep learning model introduced is the simple recurrent neural network (RNN). The architecture consists of

a hidden state layer, which allows the model to retain information from previous steps, and weights that govern the connections between the input, hidden state and output. This model has three RNN layers and 50 neurons, as the dataset is small in dimensions, a smaller number of neurons per layer is chosen to avoid overfitting (Hussain, et al., 2018).In each RNN layer, the default tanh activation function is applied. This function helps in controlling the range of the output, ensuring the network stays within a bound range (-1,1) and introducing non-linearity, as well as effectively regulating the gradients during backpropagation. Tanh has been traditionally used in RNN’s due to its ability to manage long term dependencies. Tanh is zero-centered, which means the output is symmetric around the origin, this is often considered an advantage because it can help the learning algorithm converge faster, in contrast to ReLU (Rectified Linear Unit), the zero centered nature of tanh prevents bias during optimization (All, 2024).

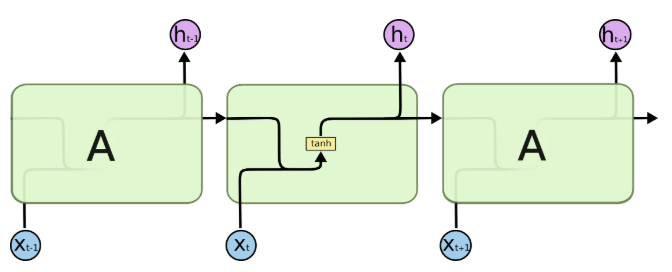


Figure :tanh function

The model is trained using the Mean Squared Error (MSE) (1), a standard choice for regression tasks, while MSE effectively penalizes large errors, Mean Absolute error (MAE) (2) could be considered as an alternative which is useful for distributions with outliers. The Adam optimizer utilized by (Sutradhar, et al., 2021) is employed to update weights efficiently, leveraging adaptive learning rates to balance convergence speed.

(1)

(2)

Moderate hyperparameter tuning was implemented through batch size and number of Epochs, the time step of 100 ensures the model captures sufficient historical context. Figure 8 displays training and test loss.

A graph with blue lines and orange lines

AI-generated content may be incorrect.

Figure : Loss Chart for RNN

Figure 8 shows that while the loss is low, the model exhibits signs of overfitting, to address this, the next model implemented must increase in complexity.

A graph with blue and orange lines

AI-generated content may be incorrect.

Figure :RNN predictions

Figures 9 and 10 display the predictions attached to the original dataset.

A graph with blue lines

AI-generated content may be incorrect.

Figure :RNN predictions

The architecture of the model Long-short term memory (LSTM) is made of 4 main parts, the memory cell, uses the sigmoid and tanh activation functions, tanh regulates gradient flow, preventing exploding values, while Sigmoid gates control the flow of information deciding what information to retain. The sigmoid function used in the Forget gate outputs values between 0 and 1, values closer to 0 mean less information is retained and vice versa, calculated through point-wise operations.

The input and output gate both use the sigmoid function, determining which new information should be added to the memory and what part of the cell state should be output as the hidden state ht, seen in Figure 11.

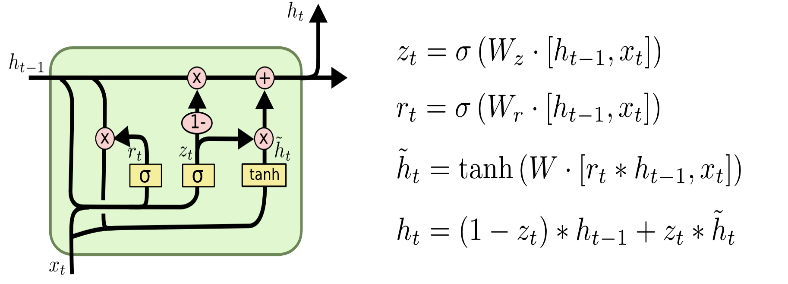


Figure :LSTM Architecture

Two LSTMs were implemented sequentially, both models share similarities, such as the use of the Adam optimizer, MSE and the dense layer with one output neuron. The first model is simpler with less neurons, less layers and no form of regularization to address the overfitting in RNN, the increase in model complexity was the first address to overfitting. Dropout layers were added to the LSTM2 with a dropout rate of 20%. Dropout is a widely used technique that randomly sets a fraction of the layers input units to zero during training. This regularization method reduces the risk of overfitting by promoting generalization on unseen data, potentially making the second model more suitable for cases where the dataset is prone to noise.

A graph of loss and loss of data

AI-generated content may be incorrect.

Figure :LSTM1 Loss

**A graph with a line and a line

AI-generated content may be incorrect.**

Figure :LSTM predictions

# Results

Figures 8,12 and 14 display the loss for training and validation, the RNN displayed a moderate reduction in training loss from 5.3361e-04 to 3.6184e-04 this represents roughly a 30% reduction in training loss over the 20 epochs it was run, validation loss slightly increased for the model from 9.5427e-05 to 3.6184e-04 indicating potential overfitting. This behavior led to the choice of the LSTM, as LSTM’s increased complexity and capability to better handle long-term dependencies can help address overfitting and improve regularization. LSTM1, despite running for 50 epochs, showed the smallest loss reduction, suggesting that it struggled to improve with time, for its validation loss, it experiences the biggest reduction from 2.5876e-04 to 1.2476e-04. The final LSTM model ran for 50 epoch and had a loss of 8.2892e-04 which decreased to 5.8307e-04 and for validation loss of 1.6909e-04 which didn’t improve much.

A graph of loss and value

AI-generated content may be incorrect.

Figure :LSTM2 Loss

A graph with blue line and white text

AI-generated content may be incorrect.

A graph showing the number of days

AI-generated content may be incorrect.

# Conclusions and Future work

This project explored the use of deep learning models, specifically RNN’s and LSTM’s, for stock price prediction using financial data. While the models showed good success, challenges like overfitting were addressed by increasing the model complexity and applying regularization techniques. Future work includes optimizing the models through extensive hyperparameter tuning, expanding the dataset size and introducing other models such as the gated recurrent unit (GRU). To enhance scalability, live streaming of financial data using Apache Kafka could be implemented. As well as changing from a local cluster to a larger, distributed environment will help handle the with increased size.

# Bibliography

Aditham, S., Ranganathan, N. & Katkoori, S., 2017. *LSTM-Based Memory Profiling for Predicting Data Attacks in Distributed Big Data Systems,* Lake Buena Vista: ieeexplore .

All, M., 2024. *https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks.* [Online]   
Available at: https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks  
[Accessed 23 03 2025].

Aung, T. T., 2024. *Big Data Analytics with Stock Market Price Prediction using Long Short-Term Memory Neural Network,* Myanmar: reaserchgate .

Datamount, 2019. *medium.com.* [Online]   
Available at: https://medium.com/@datamount/you-can-blend-apache-spark-and-tensorflow-to-build-potential-deep-learning-solutions-9298e9fe8f6c  
[Accessed 20 03 2025].

Hall, M., 2024. *investopedia.* [Online]   
Available at: https://www.investopedia.com/ask/answers/100314/what-are-key-factors-cause-market-go-and-down.asp  
[Accessed 20 03 2025].

Hussain, L. et al., 2018. *Forecasting Time Series Stock Data using Deep Learning Technique in a Distributed Computing Environment,* Greater Noida, India: ieeexplore.

Indirman, M. D. C., 2023. *Distributed Machine Learning using HDFS and Apache Spark for,* Mataram: E3S Web of Conferences 465, 02058 (2023).

Mehendale, P., 2019. *Survey on Real-Time Data Processing in Finance Using Machine Learning Techniques,* Troy: Research Gate .

Mehta, A., 2024. *appinventiv.com.* [Online]   
Available at: https://appinventiv.com/blog/spark-vs-hadoop-big-data-frameworks/  
[Accessed 20 03 2024].

Mohapatra, S., Ahmed, N. & Alencar, P., 2019. *KryptoOracle: A Real-Time Cryptocurrency Price Prediction Platform Using Twitter Sentiments,* Los Angeles : ieeexplore.

Mousa, W., 2023. *https://medium.com/@waleedmousa975/a-beginners-guide-to-gpt-bert-and-t5-how-these-language-models-work-and-how-to-use-them-b93397f104ad.* [Online]   
Available at: medium  
[Accessed 21 03 2025].

Ojha, B., 2023. *medium.* [Online]   
Available at: https://medium.com/@bibhushabibhs/a-beginners-guide-to-apache-spark-introduction-and-key-concepts-930bd98df404  
[Accessed 21 03 2025].

Petrova, B., 2025. *https://www.revealbi.io/blog/legacy-systems-vs-modern-embedded-analytics.* [Online]   
Available at: https://www.revealbi.io/blog/legacy-systems-vs-modern-embedded-analytics  
[Accessed 26 03 2025].

Preeti, 2024. *https://medium.com/@preeti.rana.ai/edge-ai-and-federated-learning-transforming-data-processing-at-the-edge-aa50aec29f22.* [Online]   
Available at: https://medium.com/@preeti.rana.ai/edge-ai-and-federated-learning-transforming-data-processing-at-the-edge-aa50aec29f22  
[Accessed 21 03 2025].

Reddy, K. R. et al., 2022. *Stock Market Prediction Using Recurrent Neural Network,* Bhopal: ieeexplore.

Robinson, S., 2023. *techtarget.* [Online]   
Available at: https://www.techtarget.com/searchdatamanagement/definition/5-Vs-of-big-data  
[Accessed 21 03 2025].

Sarma, S. L. V. V. D., 2023. *Stock market analysis with the usage of machine learning and deep learning algorithms,* Guntur: Bulletin of Electrical Engineering and Informatics.

Sepp Hochreiter, J. S., 1997. *Long Short-Term Memory,* Munchen, Germany: s.n.

Sharma, M. & Kaur, J., 2019. *A Comparative Study of Big Data Processing: Hadoop vs. Spark,* New Delhi, India: ieeexplore.

Sushree Dasa, R. K. B. ,. M. k., 2018. Real-Time Sentiment Analysis of Twitter Streaming data for Stock. *science direct ,* p. 9.

Sutradhar, K. et al., 2021. *Stock Market Prediction using Recurrent Neural Network’s LSTM Architecture,* New York: ieeexplore.