Long short-term memory network: A Review

Mogage Nicolae – ICA 246/1

# Abstract

Time-series analysis is a way of analyzing a sequence of data points collected over an interval of time. This subject raises problems in the machine learning domain due to the incapability of the traditional methods to retain older information. In this manner, recurrent neuronal networks have been introduced. This newly discovered model was able to remember older information, but another problem appeared, the vanishing gradient. To solve it, long short-term memory networks were introduced, making use of 4 gates to choose which information to remember and which to forget. In this research we want to see how this method performs in various domains, so 4 datasets were selected, providing text, audio and tabular data. Each dataset underwent a series of feature selection in order to maintain only the necessary information. The model on which the experiments were done was a simple bidirectional 2-layered long short-term memory network, with a fully connected layer on top of it to convert the result to the necessary number of classes based on the problem’s output. The method shows promising results on both classification and regression problems, achieving better results on the latter one. This raises the question whether the LSTM are better to use for time-series prediction or sequence classification. Even though this study covers the fundamentals of the network and confirms the information that is now available, more thorough investigation is required to properly make conclusions about the functioning of this network in cases when the agent needs to preserve previous knowledge.

**Table of Contents**

[Abstract 1](#_Toc183462827)

[1. Introduction 2](#_Toc183462828)

[2. Long short-term memory 2](#_Toc183462829)

[3. Datasets 4](#_Toc183462830)

[Sentiment140 4](#_Toc183462831)

[UrbanSound8K 5](#_Toc183462832)

[Jena Climate 5](#_Toc183462833)

[Top 100 Cryptocurrencies Historical 6](#_Toc183462834)

[4. Experiments 7](#_Toc183462835)

[Setup 7](#_Toc183462836)

[Results 8](#_Toc183462837)

[Discussion 12](#_Toc183462838)

[5. Conclusions 12](#_Toc183462839)

[6. Bibliography 13](#_Toc183462840)

# Introduction

As deep learning technology evolved more and more, recurrent neural networks (RNN) also made important advancements. This includes the long short-term memory (LSTM) models which are a type of the recurrent networks. They specialize on sequential data and can reproduce better results by remembering details, unlike traditional methods. To better verify this concept, we are going to test this method on 4 different datasets. The topics we chose to study are weather forecast, cryptocurrency prediction, sentiment analysis on texts and sound recognition. In this way, having 2 regression problems and 2 classification ones can provide a clear picture of how this model performs.

The model used for all the experiments is a 2-layered bidirectional LSTM with a fully connected layer on top of it. Using the same configuration for all datasets helps us to better observe the generalization of this method. This can provide us with valuable information in which domains is better to use the long short-term memory networks or if the developer needs to look after other solutions.

This report details the used methodology, the experiments setup and their results, as well as a discussion of areas for future research. Understanding how this method works, and its limitations, can help us to better contribute to the machine learning research domain. Moreover, the survey can be used as a starting point on whether LSTM are the fit solution or not for one’s problem.

# Long short-term memory

In contrast to feedforward networks, this neural architecture excels at detecting patterns in data sequences due to its bidirectional communication. By introducing a loop within the hidden layer, it allows for reusing prior information, thereby extending the functionality of the neural network to consider previous input data [1]. Figure 2.1 illustrates a simplistic comparison between feedforward propagation and the use of recurrence; it is worth noting that while only one hidden layer is used, this can be extended to include multiple layers, which can lead to better results.



Figure 2.1: Visualizations of different architectures

These differences can also be expressed mathematically. The equation for the hidden layer variable in feedforward propagation is computed by multiplying the input data with a weight matrix and optionally adding a bias term. In the recurrent case, an additional term is included, involving the multiplication of the previously computed variable with a different weight matrix. Finally, an activation function is applied to transmit the information. This activation function is typically a sigmoid logistic function or a hyperbolic tangent, which prepares the gradients for backpropagation. Taking all these components into account, we derive equations 2.1 and 2.2.

2.1

2.2

For a deeper understanding, the recurrent network concept is detailed in Figure 2.2. Here, it can be observed that at time\_t , the variable is computed based on the input data , the previous value , and the model parameters , . The consistent use of these coefficients and their transfer between phases enable efficient processing of data sequences.



Figure 2.2: Unfolded RNN Diagram

However, during backpropagation, due to the derivative calculations involving matrix multiplications over very long sequences, the vanishing gradient problem may arise [2]. To address this issue, long short-term memory (LSTM) networks were introduced [3]. LSTMs add multiple activation functions, referred to as gates. These are called the input gate, forget gate, and output gate, with their computations described by equations 2.3, 2.4, and 2.5. Additionally, a candidate state for the next unit is needed, calculated using formula 2.6, with the next cell state determined by equation 2.7 [1, 2]. The differences between the units used by these architectures and the computation of gates are visually represented in Figure 2.3.

2.3

2.4

2.5

2.6

2.7



Figure 2.3: Architectures of Networks’ unit

The long short-term memory model we are going to use is already implemented in the PyTorch library. Each problem has its own input and hidden state sizes, but we add 2 layers and make it bidirectional over all datasets. On top of the LSTM, we add a fully connected layer to convert the output to the required number of classes based on the number of expected classes.

# Datasets

For an improved setup of the experiments, we made a short analysis of each dataset for a deeper understanding. This helped us to manually make the feature selection and visualize the data. The selected data covers multiple options in which the LSTM can be used. Included in these are tabular data, based on text or real numbers, or sounds inputs. Apart from the sound recognition and cryptocurrencies datasets, which has 80% train data and 20% test data, each dataset was split into training, validation, and test data, with percentages of 60%, 20%, and 20%, respectively. Details of each database will be discussed in the following subsections to motivate the choose of each one and their contribution.

## Sentiment140

This dataset was originally published in Stanford’s research [4], where they automatically classified the sentiment of twitter messages using distant supervision. They collected the data using the Twitter API [4], each tweet having at most 140-characters. This presents a unique challenge compared to formally composed texts due to the shorter nature. After a feature reduction [4], the dataset consists of 5 features, the id, date, username and text of the tweet, as well as a query term. The target is formed from the polarity of the tweet, where 0 means negative, 2 means neutral and 4 is for a positive tweet. In the Table 3.1 we can see an example for each target available in the final dataset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Date** | **Flag** | **User** | **Text** | **Target** |
| 1467811184 | Mon Apr 06 22:19:57 PDT 2009 | NO\_QUERY | \_ElleCTF\_ | my whole body feels itchy and like its on fire | 0 |
| 1960185231 | Fri May 29 07:33:36 PDT 2009 | NO\_QUERY | elorentzo | KAJ??? I ve loved you all my life!!! | 4 |

Table 3.1: Example from original dataset

We went even further and removed the ID, Date, Flag and User fields as they are not relevant to predict the connotation of the message. This is because, from the model’s perspective, ID represents a random number, Date does not bring an important influence on the text nature, Flag is the same for all entries, and User may bring unnecessary load on the input. Moreover, because the final dataset had only 0 and 4 as targets, we converted the 4 target values into 1. Therefore, we only remained with 2 columns, representing the text and the target. Emoticons are removed from the dataset as they are considered noisy labels and can result in wrong assumptions when doing the classification. For example, we can have a smiley emoji but the text to have negative connotations, confusing the model when it has to predict.

## UrbanSound8K

The second classification dataset we are using was specifically designed for urban sound classification [5]. The authors made the selection of the 10 classes based on an analysis of noise complaints filed through New York City [5]. The 10 sounds in UrbanSound are: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. These classes are covered in the 8732 labeled slices that were obtained from the full-length recordings, each one having a maximum of 4 seconds. This duration was chosen based on a listening test which found it sufficient from humans to identify environmental sounds with 82% accuracy [5].

The authors split all the audios into 10 folds, and to train the model, a 10-fold cross-validation must be done to precisely determine the accuracy of the model. This approach ensures that each class is proportionally represented in every fold. More details will be covered in the experiments chapter. This dataset brings a unique perspective on how LSTMs work with short sequences and how they behave on audios because of the various lengths it must cover and the multiple classes, complicating the learning process. Having said that, the dataset can be seen as an entry point for future research on urban sounds and can provide valuable information in this domain. In the Table 3.2 we can see the count for each class. The model will be biased to not predict the low count classes because there are less examples for them.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Count** | **Class** | **Count** |
| Dog bark | 1000 | Jackhammer | 1000 |
| Children playing | 1000 | Drilling | 1000 |
| Air conditioner | 1000 | Siren | 929 |
| Street music | 1000 | Car horn | 429 |
| Engine idling | 1000 | Gunshot | 373 |

Table 3.2: Distribution for each class

## Jena Climate

Having chosen a text and audio input datasets, we want to see how the LSTM performs also on tabular data. In order to better make a conclusion, we also need to check the method performance on regression problems. With this in mind, the first dataset selected covers the weather forecast topic. This dataset recorded the metrics at the Weather Station of the Max Planck Institute for Biogeochemistry in Jena, Germany. Jena Climate is made up of 15 features, such as: date-time; pressure; temperature in Celsius, Kelvin and Celsius relative to humidity; relative humidity; saturation vapor pressure; vapor pressure; vapor pressure deficit; specific humidity; water vapor concentration; airtight; wind speed; maximum wind speed; and wind direction in degrees. The data is covered from 1st January 2009 to 31st December 2016, being recorded every 10 minutes during these years.

Because we do not have a specific target, we can choose any of the features to serve as it. In this research, we went for the temperature, so we removed the two columns for the temperature in Kelvin and Celsius relative to humidity. The reasoning behind this is that we can better observe the behavior of the LSTM in time series data. The length of the sequence we opted for is 6, meaning we predict the temperature based on the last hour metrics. All columns are represented in real numbers, creating a high variety of values for each feature.

To better understand the data we are working with, we can see in the Figure 3.1 the distribution of each column we are working with. The dataset contained some outliers on the wind speed and maximum wind speed columns, where -9999 values were registered. We opted to remove the rows that contained these errors to avoid unexpected behavior of the model.

A group of blue and white graphs

Description automatically generated

Figure 3.1: Distribution of features

## Top 100 Cryptocurrencies Historical

The final dataset covers a quite interesting domain, the blockchain and its cryptocurrencies. Unlike the stocks market, where the prices do not change a lot from day to day, here the values are very volatile and can change a lot from day to day unpredictable. For this reason, very accurate models are yet to appear in the industry. We want to experiment how a simple LSTM can adapt to this sequence data and with what accuracy it can predict this regression problem.

The dataset contains, as the name suggest, 100 different crypto coins files. Each file is formed from the following fields: date, open, high, low, close, volume and currency. In the Table 3.3 we can see an example from the dataset based of the Elrond coin. We remove the date column because we do not shuffle the data to keep track of the time. The currency one is also omitted as USD is used in all the files. Furthermore, the statistics included in this database are recorded every day over multiple years, depending on when the coin started. In this case we chose the close price to predict based on the last 2 weeks of data. Because there are fewer entries available of more recent currencies, it is more difficult to project the prices. For research purposes, we only trained and evaluated the model on 10 different coins, older but as well as recent ones. The selection was done based on popularity, but also on the volatility of the currencies to better evaluate the method. This allows us to properly evaluate the model and to verify the capabilities of the long short-term memory networks in this domain.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Open** | **High** | **Low** | **Close** | **Volume** | **Currency** |
| 2020-09-22 | 8.85965746 | 9.06857681 | 8.46222591 | 8.72244834 | 5299690 | USD |

Table 3.3: Cryptocurrency dataset example

# Experiments

In the following sections we will cover the setup made for the problems, the results we got during and after training, as well as a discussion on the performance of the LSTM and future research.

## Setup

For the first experiment, we use a pretrained model developed by Google, Bidirectional Encoder Representations from Transformers (BERT) [6], to tokenize the text into tokens. We opted for the large, uncased variant because beside the first character of the sentence, there are not many upper-case letters. The maximum length of the tokenizer is set to 140 due to the nature of the tweets that cannot be over 140 characters. To ensure the same input size for all messages, we pad the message, and in case there are any longer texts, we truncate them to the set length. Special characters for beginning and end of sequence are also added with the help of the tokenizer.

In the second experiment, all the sounds are ensured to be a single channel (mono). If the audio has multiple channels (stereo), it averages the channels along the first dimension to produce a mono waveform. Afterwards, we extract 40 Mel-Frequency Cepstral Coefficients (MFCC) features from it. A higher value for the computed coefficients allows us to capture both low and high-frequency details, helping the model in distinguishing different sound classes. Following the same idea as before, we whether pad or truncate the MFCC to a fixed length. After making the average over all folds, we went with 325 for this size to retain as much information as possible from the longer sounds, but also maintain a possibility for the shorter ones to be predicted.

Next, for the regression problems because the sequences are not predefined, as in the classification case, we create them by iterating over the dataset and sticking them over a given period of time. In case of the weather is the last hour and for the crypto is the last 2 weeks. In the case of cryptocurrencies, because there is a lack of data, we transform the data to a [0, 1] interval by following the formula given by the equation 4.1. Furthermore, replacing min with 0 and max with 1, we remain with the formula 4.2. This results in a more robust training process of the model.

4.1

4.2

The loss function used for the first 2 experiments is the categorial cross entropy [7], where bigger discrepancies in outputs results in greater losses. The mathematical formula is given by the equation 4.3, where indicates whether example i belongs to class j, is the prediction for class j for example i, N is the number of examples, and C is the number of classes. Furthermore, in the regression case we use the mean squared error, found in the equation 4.4, where p are the predicted values, r are the actual values, and N is the number of examples. Moreover, adaptive moment estimation (Adam) [8], mathematically represented by the formula 4.5, is used as the optimization function in all cases. This chose was done because of the convergence speed of the function. Therefore, a fewer number of epochs is necessary to achieve almost the same result than using other methods like SGD or AdaGrad. However, the cost of this speed is that the result we obtain is not the best, this choice being done based on the available resources. To avoid the case where the model does not converge to a solution, we also introduced a scheduler to decrease the learning rate after a given interval of epochs, depending on the problem’s need.

4.3

4.4

4.5

Finally, the metrics for the classification case are accuracy and f1 score, as for the last 2 experiments, we use mean absolute error and the r2 score. These measurements allow us to properly evaluate the method and draw the correct conclusions. The mathematical formulas can be found below in the same order. True positive values are denoted by TP, false positives by FP, and false negatives by FN. The confusion matrix can be used to extract these values.

, where ,

,

The implementation of the datasets, the model and all the setup was done using PyTorch framework and scikit-learn library. These tools made the development easier in terms of design, training and evaluation processes, providing already defined options for the functions we used. During the training, the datasets were split into batches of 32 or 64, depending on the problem specifics. The initial learning rate set was 0.0001 for the Jena Climate dataset, and 0.001 for the others.

## Results

The training process for each dataset can be seen in the Figure 4.1. In the case of the sentiment analysis, we trained the model for fewer epics because the model does not show promising performance, even after multiple hyperparameter tuning attempts. The cross-entropy loss remained in the 68% value over all the epochs, proving the limits of a simple LSTM in this case. More about this will be covered in the discussion section. Regarding UrbanSound8k, as the authors suggested for accurate results, we performed 10-fold cross validation here, having 9 folds for training and 1 for testing the model. The average accuracy was computed by iterating over all of them. We carried out 10-fold cross validation, as the authors recommended for accurate findings, using 9 folds for training and 1 for testing the model. We iterated over each of them and calculated the average accuracy. Below is shown the train and test losses for every fold, each one following almost the same pattern.

|  |  |
| --- | --- |
| A graph with blue and orange lines  Description automatically generated   1. Sentiment140 | A graph of a train loss  Description automatically generated   1. UrbanSound8K |
| A graph with a line graph  Description automatically generated  (c) Jena Climate | A graph of a graph showing the amount of a train loss  Description automatically generated with medium confidence   1. Top 100 Cryptocurrencies |

Figure 4.1: Training losses

On the opposite end, the results obtained for the regression problems are promising. In the Jena Climate dataset, we can see in the Figure 4.1.c how the MSE loss rapidly decreases, getting in the end under 10%. We stopped the training process around the 25-epoch mark to avoid the overfit problem where a model performs really well on training data but does poorly on unseen cases. Furthermore, we do not have a validation set for the cryptocurrencies dataset because there is a lack of samples to work with. This allows us to properly train the model to extract the best performance out of it. The losses for all the coins we trained the model on are added in the same plot, allowing us to better observe which chain leads to better predictions. As seen, Bitcoin, Ethereum and Litecoin shows the most promising results, with a MSE below 0.0005. This happens because these coins have more entries, having a longer history in the market. On the other side, Elrond, Polkadot and Tether are the worst ones, with a MSE above 0.0013, with Polkadot having the highest one, 0.0025.

Based on the chosen metrics, accuracy with f1 score for classification, and mean absolute error with r2 score for regression, we obtained the results shown in Table 4.1. This performance proves the capabilities of this method in the time-series data domain. However, using a simple long short-term memory unit is far too simple for a machine learning model to be able to learn such heavy tasks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | Sentiment140 | UrbanSound8K | Jena Climate | Top 100 cryptocurrencies |
| **First metric** | 0.5507 | 0.3429 | 0.2104 | 0.0237 |
| **Second Metric** | 0.5502 | 0.3435 | -0.9070 | -1.6448 |

Table 4.1: Results

For a better comprehension of the results, we also created a confusion matrix and a plot for the actual and predicted value, depending on the problem’s type. The figures 4.2 and 4.3 show us the incapability of the method to retain the information on very short sequences and take decisions based on it. As for the regression case, figures 4.4 and 4.5 proves that even if the r2 score is negative, the actual value are really close to the truth, and they are following the same trend. In order to achieve the best results and make a correct assumption on the LSTM, hyperparameter tuning was done for each dataset. This includes finding an equilibrium between the start learning rate, when to decay it during training and batch size. This brought improvements up to 20% on the f1 and r2 scores.

|  |  |
| --- | --- |
| A diagram of a confusion matrix  Description automatically generated  Figure 4.2: Sentiment140 Confusion Matrix | A screenshot of a graph  Description automatically generated  Figure 4.3: UrbanSound8k Confusion Matrix |

A graph showing the temperature of a person

Description automatically generated with medium confidence

Figure 4.4: Jena Climate test plot

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Figure 4.5: Top 100 Cryptocurrencies plots

## Discussion

The results we obtained during the experiments provide valuable insights into the possibilities and limitations of the LSTM models over a various range of tasks. As previously seen, this method fails to learn when dealing with short sequences, not being able to capture the fine details. But when it comes to sequences with multiple features available, it succeeds in keeping track of the past events in order to make a more accurate prediction. This involves the fact that a simple LSTM lacks the representational power for tasks with complicated dependencies or noisy datasets. Depending on the data, normalization can improve the training process, as seen in our crypto experiment, where without the transformation of the samples the error rate would have been much higher. Another important addition which impacted the performance of the method is the learning rate scheduler, which allows us to change the learning rate during training. This has an impact on the backpropagation process, making sure the model is able to converge. Without it, the learning phase would have been taken longer because it would have been necessary to set a lower learning rate from the start.

What is yet to be proven is whether LSTMs perform better in the regression type of problems or in classification. Our experiments tend to demonstrate that this method is more suitable for the former one. Additionally, based on the observations we made, another trend that should be followed is whether this type of RNN is more suitable to use historical data to take a future decision or to learn information from a sequence to make an independent prediction. Nevertheless, the research can be taken even further for this machine learning algorithm which has a lot of possibilities. The key contribution of our research was the observation of the long short-term memory network’s behavior in multiple domains.

Future work could explore more advanced architectures that make use of the long short-term memory method. This would allow the model to overcome the observed limitations in handling short sequences and high variability. For example, hybrid models like attention-enhanced LSTMs or transformers can help in addressing these challenges. Moreover, extending this work to include transfer learning or ensemble methods may further enhance the robustness and generalizability of the results. Adaptation of external features or domain-specific augmentation could also help to improve the performance of the model. For a clearer picture of the behavior, additional metrics such as recall or mape can be explored. Data transformation techniques is also a subject that should be addressed when it comes to what can be improved in the future, allowing the model to assimilate the information faster.

# Conclusions

This research has provided valuable insights into the performance of long short-term memory networks across diverse datasets. By analyzing the LSTM model’s behavior in handling sequence and time-series data from multiple different sources, we highlighted both the strengths and the limitations of this architecture. The key findings indicate that the network performs really good in processing long-term dependencies, but it faces challenges when dealing with short sequences.

The experiments suggest that LSTMs are suitable for regression tasks, especially when predicting continuous variables over time. As seen in Figure 4.4, the Jena Climate demonstrated the network’s ability to efficiently predict and adapt to trends in tabular data, achieving a mean squared error below 10% by the 25th epoch. This proves the capability of the method to capture complex temporal relations if the dataset is large enough and well-processed. Similarly, the results we achieved in the cryptocurrencies experiment back up the hypothesis that the model is able to learn from highly volatile data. This is evidenced by the performance we got on well-represented coins like Bitcoin and Ethereum. The ability to normalize the inputs can significantly improve the training efficiency for some datasets, demonstrating the importance of preprocessing in achieving accurate predictions.

In contrast, LSTMs had a harder time regarding the classification tasks, where the model failed to distinguish patterns from short and noisy sequences. This resulted in the cross-entropy loss to decrease very slowly and achieve arbitrary results. In this way, the limitations of the network are shown in the contexts where deeper contextual understanding is required. Despite using a large tokenizer for the Sentiment140 or the 10-fold cross validation, the low accuracy we achieved suggests that a feature extraction or augmentation is needed in order to improve the overall performance.

To address the observed challenges, future research should explore advanced neuronal architectures that combine LSTMs with transformers in order to improve the capability to retain important information from shorter sequences. Another topic that can be addressed is the feature engineering, where fine transformations of the input data can impact the results of the model. Moreover, hyperparameter tuning can greatly impact the performance of the model, finding the equilibrium between the batch size, learning rate, optimizer and loss function used being an important step in developing properly models that can perform well in our day-to-day life.

In conclusion, this study shows that while long short-term memory networks are an effective model for many tasks, the context in which they are used can greatly impact the results. The problem’s domain and dataset characteristics needs to be properly evaluated in order to determine whether LSTMs are the suitable method. By addressing the mentioned problems in future research, applications may be able to make better use of the capability this method provides, leading to more accurate and scalable machine learning solutions, on which people can rely to use.

# Bibliography

|  |  |
| --- | --- |
| [1] | R. M. Schmidt, *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,* 2019. |
| [2] | G. Chen, *A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation,* 2018. |
| [3] | S. Hochreiter and J. Schmidhuber, *Long Short-term Memory,* 1997. |
| [4] | A. Go, R. Bhayani and L. Huang, *Twitter sentiment classification using distant supervision,* 2009. |
| [5] | J. Salamon, C. Jacoby and J. P. Bello, *A Dataset and Taxonomy for Urban Sound Research,* 2014. |
| [6] | J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding,* 2019. |
| [7] | A. Mao, M. Mohri and Y. Zhong, *Cross-Entropy Loss Functions: Theoretical Analysis and Applications,* 2023. |
| [8] | D. P. Kingma and J. L. Ba, *Adam: A Method for Stochastic Optimization,* 2017. |

This work is the result of my own activity, and I confirm I have neither given, nor received unauthorized assistance for this work.

I declare that I did not use generative AI or automated tools in the creation of content or drafting of this document.