Attention LSTM - "A comparison of ALSTM and LSTM performance in airline passenger forecast"

Sina Sen 4821629

[1]TU Delft

Abstract

The introduction of Attention Long Short Term Memory (ALSTM) produces an alternative to Long Short Term Memory (LSTM) by aiming to optimize information passing via removing the complexity of the cells in LSTM. In this work, the results and comparison of the performance of LSTM algorithms versus ALSTM architectures is assessed through the forecast of airline passenger numbers over months. The results are analyzed through qualitative and quantitative data. The hypothesis made in this paper is that ALSTM will perform better than the LSTM, and the results shows that the hypothesis was correct under some circumstances, although by a small margin.

1 Introduction

Long Short Term Memory machines [3] are recurrent neural network architectures used mostly with time series data set for the purpose of prediction, forecast and classification [6]. Over years, they have been the industry standard when it comes to forecasting, prediction, and classification tasks of time series data. However, in LSTM architectures, as information passed from one time step to another has to pass through all other time steps, this may lead to loss of performance in LSTMs. A potential solution, or improvement, that is being researched is essentially the Attention LSTM. Multi headed self-attention layers can potentially be used to increase the performance of LSTMs and speed up the learning process. [7] These attention mechanisms are similar to cognitive attention, as they are believed to focus on important parts of the time series data and compute its task based more on relevant information. [7] In other words, they optimize the information passing process. In this research, how the freshly introduced architecture of ALSTM performs for flight passenger forecast, compared to normal LSTMs is evaluated. The evaluation metrics used are test loss, defined as Root Mean Square Error (RMSE) to see how well each model learns, and qualitative visualizations to see how well each one of the two architecture performs.

Main question that is researched in this project is: "How well does ALSTM perform in flight passenger forecast compared to the normal LSTM?". Breaking down the main question, some subquestions are:

- "How to construct ALSTM from LSTM?", "How to apply ALSTM into a prediction/regression task?"
- "How does LSTM and ALSTM perform at the task of flight delay prediction in terms of accuracy?"
- "How does ALSTM perform at the task of flight delay prediction in terms of accuracy?"
- "Which metrics to use for evaluation and comparison of both the LSTM and the ALSTM?"
- "Under which conditions does ALSTM perform more accurately than LSTM and vice versa?"

The hypothesis is that ALSTM will perform better compared to LSTM in the forecast task of flight passenger numbers. LSTMs are designed to use the past data to produce forecasts for the future, or predict an outcome of steps of time ahead. [3]. However, due to its complex architecture in terms of amount of mathematical operations and information passing from one cell to another, LSTMs can suffer from loss of relevant information. ALSTM aims to tackle this problem through parallel processing, unlike sequential processing of LSTMs, and through its simplified architecture via attention mechanisms[7]. This is the main reason underlying the hypothesis. By choosing this particular project and research question, it is aimed to provide a better alternative to LSTMs as potential outcome of this research is a useful tool for important real life cases, such as weather prediction, stock market prediction, natural language processing, and many more. Furthermore, it is believed that this work can contribute to the better development of science and indirectly help lives on the cases that are have mentioned.

Through numerous experiments conducted, it is found out that, while LSTM has superior performance in some cases, ALSTM slightly performs better compared to LSTM under several scenarios that will be discussed later in the paper. More future work should be put into the research of ALSTMs as there may be performance improvements regarding the architecture, implementation and test scenarios to find out suitable uses cases for ALSTM.

2 Background Information

To understand why and how ALSTM is expected to perform better, it is vital to introduce the previous architectures on time series forecasting. The most important one is the LSTM architecture, which is compared to ALSTM in this research. Recurrent neural networks (RNN) are the building blocks of LSTM, which uses RNN architecture as its basis and introduces new concepts on top of it.

2.1 RNN Architecture

Long Short Term Memory architectures are often used with time series forecast and prediction problems. They are built on top of recurrent neural networks (RNN), which are often utilized for NLP problems like text generation and speech recognition [6]. RNNs process sequential information and its output on each time step depends on previous computations. [2] This allows RNNs to communicate with, or have a memory of, past information. [2]

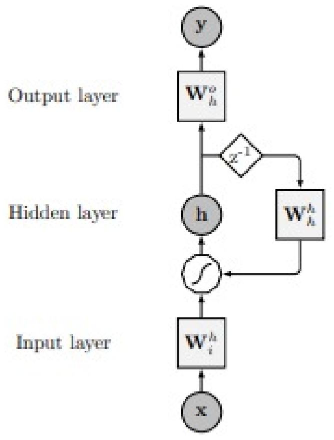

Figure 1: Broad representation of a RNN architecture. Dark circles: x: input node, h: hidden node, y: output node. Squares of W_i^h and W_h^h represent weight matrices of input, hidden and output weights. The weights are optimized in the training process via gradient descent.

[2]

While RNNs achieve forecasting the future on the short term, they can neglect the context behind the input and the information passed through long sequences and long term dependencies. The main reason this occurs is because of *vanishing and exploding gradient*, which refers to the rapid increase of the gradient during the training . [1] [5] To tackle this problem, along with increasing the longevity of architecture's memory in terms of understanding the context of long sequences of input, LSTM was built. [3]

2.2 LSTM Architecture

LSTMs are built to improve the performance and solve the problems of RNNs which are vanishing gradient and understanding the context of long term dependencies. LSTMs essentially achieve this through a newly introduced method, the cell state. Information passes through the cell state at each step and information can be modified during the passes. This, importantly, allows LSTM to remember or forget pieces of information along the training. Without a deep mathematical explanation, LSTM architecture can be described through the following states and gates:

- Forget gate: As the name suggest, forget gate's function is to determine whether or not to forget information that are no longer relevant for the context. The gate takes the hidden state of the previous cell $h_t - 1$ and the input x_t at that time step. Then the encoding of the $h_t - 1$ and x_t is multiplied by the weights and the bias is added. After that, sigmoid function is applied on this value and an output vector including 0s or 1s is calculated. This vector is transmitted to the cell state, telling cell state to forget the information piece if its corresponding position is a 0, and keep it if it is a 1. Finally, this output is multiplied with the previous cell state $C_t - 1$
- Input gate: This gate creates an extra piece of information that is also transferred to the cell state, after a series of operations is done. Firstly, similarly to forget gate, input gate follows the same process to understand which values should be transferred to the cell state. Then a vector that has all the possible values from the encoding of $h_t - 1$ and x_t is created through a tanh function that outputs values from -1 to 1. Lastly, these two vectors are multiplied, then added to the cell state.
- Output gate: Fits the values of cell state into a vector of values of -1 to 1 by using a tanh function. Then, again similarly to the forget gate, uses a sigmoid function to choose which values to keep to be outputted. Then, the result of the sigmoid and tanh is multiplied and fed into the next time step cell as the hidden state h_i.
- Cell state: Uses the information from the previous cell state, and gets additional input from forget and input gates, and passes its information to the next cell state, while also contributing to the next hidden state $h + i$.

Figure 2 and Figure 3 display this architecture visually. The part with the first sigmoid operation represents the forget gate, the calculation of the second sigmoid, the first tanh and multiplication of these two represents the input gate, and the last sigmoid function and tanh function's multiplication represents the output gate. Figure 3 and Figure 4 give a more abstract overview on the operations under LSTM architecture.

Within this research, it is believed that the performance of LSTM's information transmission can be further increased. Although LSTMs have shown a drastic improvement compared to RNNs [3], they still have some limitations that can be further overcome. Due to LSTM's complex architecture and the necessity of information passing from one step to another and through lots of different intermediary steps, LSTMs might result in a loss of performance in terms of the ability to assess distant past inputs and to learn patterns. This abovementioned optimization is not an improvement on LSTM, but rather an alternative approach to using time series using Multi Headed Attention Layers.

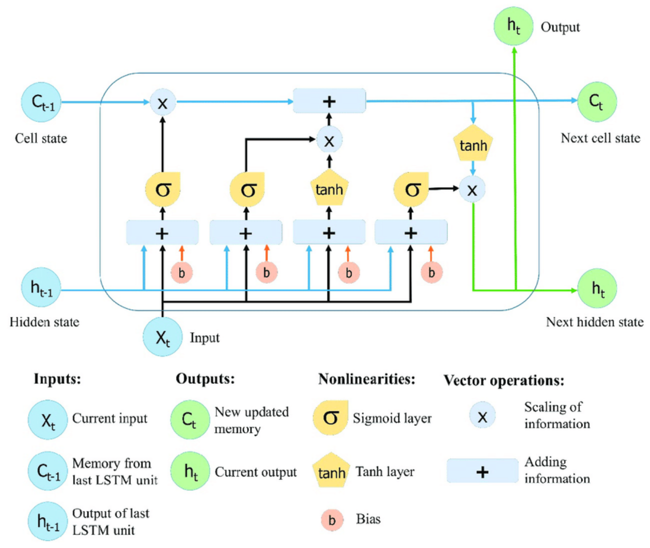

Figure 2: LSTM Architecture
[4]

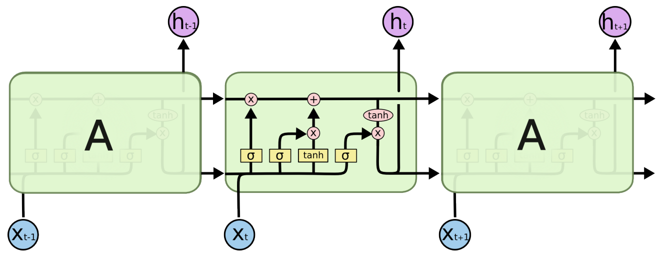

Figure 3: LSTM Architecture on a more abstract level
[4]

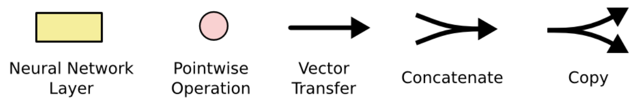

Figure 4: Notation on Figure 3
[4]

2.3 ALSTM Architecture

Attention mechanism has become a vital part of sequence modeling and forecast models in various tasks, as they enable modelling of dependencies and patterns while not regarding their distance in the input or output. [9] The core idea of ALSTM is to unpack the sequence of hidden states so that none of the gate function depend on the hidden state.

The newly introduced ALSTM architecture is designed to improve the information transmission of LSTMs. This architecture enables parallel processing of the input data, thus speeding up the learning process and removing the necessity to process time series data in order. It is aimed to have better recognition of important features and parts of the input, thus extracting more context relevant information and better information transmission for long dependencies. Furthermore, as LSTMs pass information one by one from one cell to another, the information transmitted is vulnerable in terms of the potential to forget important information from time steps before. This may not be a problem for short sequences, but for long sequences it is vital not to discount these information during the forecast. Instead, ALSTM allows the information tranmission directly from each of the previous time steps. This can potentially lead to better recognition of the patterns and information passing between time steps, while also respecting causality.

Attention Mechanism
It is stated by [7] that "An attention mechanism can be explained as mapping a query and a set of key-value pairs (k, v) to an output, where the query, keys, values, and output are all vectors." [7]

The output is a computation of the weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. [7]

Scaled Dot Product Attention
Another introduced concept on [7] is the scaled dot product attention. The input includes queries Q and keys K of dimension d_k, and values V of dimension dv. Attention is computed by computing the dot products of the query set with all keys and dividing each by the square root of dk. Lastly, a softmax function is applied to compute the weights of the values. [7]

$$Attention(Q, K, V) = softmax(\frac{Q^k K}{\sqrt{d_k}})V$$

[7]

Multi-Headed Attention (MHA)
Scaled Dot Product Attention can be used to perform attention in parallel. This can be achieved by linearly projecting the queries, keys and values h to d_k, d_k and d_v dimensions, respectively.[7] Then, the attention can be computed in parallel for the projected inputs, which results in d_v dimensional output values. Finally, these values are concatenated and once again projected.

$$MultiHead(Q, K, V) = concat(head_1, .., head_n)W^O$$

$$Head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

[7] The projections are parameter matrices $W_i^Q \ \epsilon \ R^{dmodel*d_k}$, $W_i^V \ \epsilon \ R^{dmodel*d_v}$, and $W^O \ \epsilon R^{dmodel*hd_v}$.

Figure 5 and Figure 6 are visual representations.

Attention LSTM
ALSTM architecture introduces the concept of reformulating LSTM as a self attention network and builds its architecture on top of the concept of multi-headed self attention. The cell state and the forget gates are ignored in this architecture. The input gate corresponds to d = 1 dimensional keys in the MHA, and the output gates become the corresponding queries. The causality is respected by multiplying the weights with the lower triangular matrix. It is expected that increasing the dimension d of the messages, ALSTM can yield better message passing between time steps.

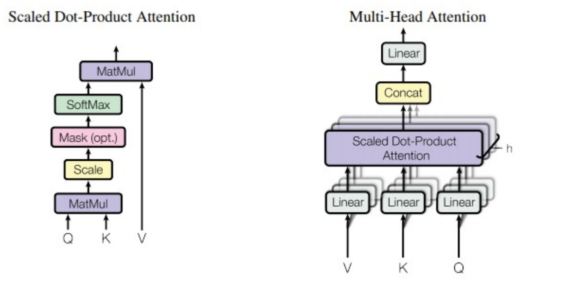

Figure 5: (Left) Scaled Dot Product Attention (Right) Multi Head Attention Architectures

[7]

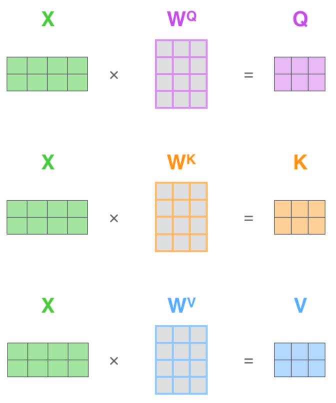

Figure 6: The calculation of Q, K, V matrices

The mathematical representations of what is discussed above are as follows:

$$S^k := exp(\frac{1}{\sqrt{d}} X Q^k K^k X^T) \odot L \qquad (1)$$

L is the lower triangular matrix. To respect the causality between the time steps, L is multiplied in Equation (1). $\odot$ is the point-wise Hadamard product. After that, the weight matrix W and hidden message H are computed according to the following equations:

$$W^k := S^k \oslash (S^k 1 1^T) \qquad (2)$$

$$H := \sum_{k=1}^{m'} \sigma_C(X \Theta^k) \qquad (3)$$

Here σ_C is a hyperbolic tangent (tanh) function that is often assumed as identity function, and $\oslash$ denotes a point-wise division.

3 Methodology

3.1 Implementation of ALSTM and Tools

The ALSTM model is implemented using Python 3.8 and PyTorch library developed by Facebook. The LSTM implementation used in this research is also default LSTM class of PyTorch. Matplotlib library is used visualize the results. Pandas and Scikit-learn is used to preprocess the data and fit it into a time series.

3.2 Dataset

The dataset used to compare the performance of LSTM with ALSTM is the airline passenger numbers data published open source by Github user jbrownlee is used. The dataset is a rather simple one, with month and passengers as its two columns. The dataset contains a span of 12 years, so 144 months. This data set is also one of the PyTorch's default data sets. Thus, it has been used in many machine learning projects and its validity has been proven.

The first month on the dataset is January of 1949, whereas the last month is the December of 1960. The data is in chronological order so there is no need to reorder the data.

The only other column that the dataset contains is the number of passengers. The data is visualized in the figure below.

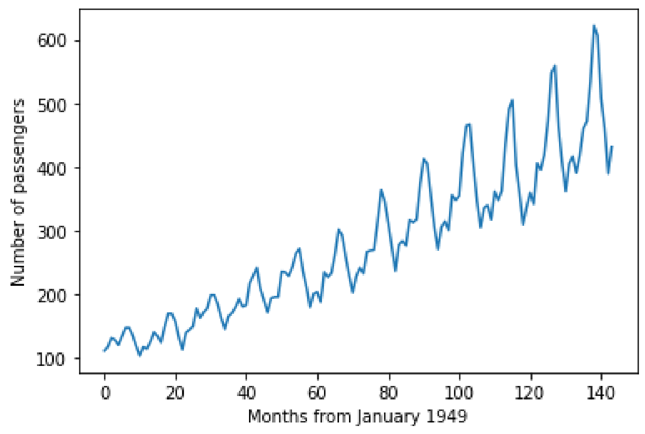

Figure 7: The number of passengers through months

Loading data into time series format
All the data from the time steps 0 to t-1 is fed to the model to forecast the values at time t so that forecast at time t will be based on all previous time steps, rather than just the previous one. This is done essentially to favor ALSTM's behaviour of being able to pass on information from earlier time steps and recognizing long term patterns and dependencies.

However, for comparison purposes, a sliding window approach is also used alternatively with a window length of 4 to see how ALSTM performs under a windowed approach.

3.3 Forecasting Task

The main goal of the forecasting task is to predict how many passengers will fly with the airline at the next time step. For this research, time step intervals used are months, as the data set used was convenient for this type of forecast. The result of this forecasting task may potentially be used in many real life areas like the analysis of customer behavior, or the improvement sales and marketing strategies (as the exact same algorithm can be applied to sales forecast instead of flight passanger forecast). While it is believed that, as there are no other features used other than the date and the number of passengers, the results produced in this research has no bias in terms of features. However, the data set used is extremely old and simple that the forecasting done in this paper may not perfectly reflect the forecasting of current time airlines, and may potentially perform worse than it does in this paper.

3.4 Performance Metrics

- Qualitative:
  - Root Mean Squared Error: This is a frequently used metric to assess how well the model performs and outputs the quality of the prediction
  - Average and Standard Deviation of Test Loss: The average and standard deviation of 5 different runs will be taken, to asses the performance of both models on a bigger scale. If the average test losses are more than two standard deviations separated, it can be said that one model is better than the other one.
- Quantitative:
  - Visual Forecast: With the visual forecast of ”Forecasted versus Actual” values, one can see how each architecture performed on a less abstract metric. After all, loss functions give us numbers whereas on a visual forecast it can be seen how accurate and close each architecture performs.

4 Experimental Setup and Results

The experiment that is conducted to reflect the performance of ALSTM and LSTM is:

- Monthly forecast experiment: with this experiment the total number of passangers of the current time step, which is the current month, is evaluated considering all previous time steps (months)

As input, time steps and their corresponding values of number of passengers are taken. And as output, the forecasted number of passengers for the current time step, is computed.

All of the following computations are averaged on 5 different runs to reduce variance. Furthermore, root mean squared error function is used to compute the loss. The graphs computed with loss values are all test losses.

Data pre-processing is done to fit the data in between 0 and 1 before feeding it to the models. During the training, adaptive learning rates are used for better learnings of the models. Then hyperparameter optimization is applied to figure out what values of learning rate and number of epochs does it work the best with. After this optimization, the hyperparameters determined that work the best with ALSTM are:

- Learning Rate: Starting from 10^-5, and adjusting it down through training to 10^-7
- Number of Epochs: 5000, after 5 thousand, the loss starts to rise back up, thus potentially leading to an overfit. That is why the value of 5000 is used.
- Message passing: A message passing value of 3 is used. It was observed that ALSTM performs the best on the value 3 and the worst on value 1.

Firstly the experiments are decided on the following metric; in 5 different runs with the training to test ratios of %90, %80, %75, %70, %60, the average and standard deviation of the test losses will be determined, and compared. Learning decay is used to adjust the learning rate during the training for better learning, and gradient clipping is used to prevent jumps in losses and in gradient descent. As the optimizer for the training of the models, Adam optimizer is used, with RMSE as the loss function.

Then, the models are trained with different number of epochs and learning rates to optimize these hyperparameters.

The goal of the first experiment with different splits is to realize under which splits does ALSTM perform better than LSTM, and how do their average performance through all splits compare.

On Table 1 and Figure 9, the average best test loss through 5 run on each of the 5 splits is displayed. It can be seen that ALSTM has an advantage, in terms of loss, for all splits except 0.90. This is promising for our hypothesis. As the training set gets smaller, LSTM starts to learn worse, and ALSTM learns better. Thus, the difference between the losses of two models increases. This may lead to a realization that ALSTM learns better on smaller training sets, and may start slightly overfitting after a threshold. On the other hand, for LSTM, there is a drastic difference of 0.015 between its best and worst loss across the splits. This may mean that compared to ALSTM, LSTM needs more data to be more accurate in terms of the test loss. This is also parallel with the claim made in this paper that ALSTM can potentially transmit information better than LSTM.

Train split	LSTM Test Loss Average	ALSTM Test Loss Average
0.90	0.099	0.100
0.80	0.105	0.098
0.75	0.106	0.095
0.70	0.108	0.096
0.60	0.114	0.097

Table 1: Average test loss through different training splits

The averages turned out that ALSTM performs better, but a statistical test of averages and standard deviations of all 5 splits is a good metric to compare both.

Looking at Table 2, it is seen that the average loss of ALSTM in 5 different splits is less than that of LSTM. The difference between two averages equals 0.09. Also, LSTM's standard deviation is 2 times worse than that of ALSTM. The difference of the averages between two models are separated two standard deviations away from each other, for both LSTM's and ALSTM's standard deviation. Thus, ALSTM has performed better than LSTM on this experiment.

Method	Mean	Standard Deviation
LSTM	0.106	0.004
ALSTM	0.097	0.002

Table 2: Mean and St. Dev of both methods through 5 different splits

Figure 8 visualises the learning process of the models on different splits. Although the end result of test loss is not distinguishable, it can be stated that, although less accurate, the LSTM model has learned faster than those of ALSTM. Indeed, after around 1500 epochs, LSTM's loss seems to converge, and increase slightly. The reason behind this can be the eagerness for overfitting for LSTM, which has been stated by other scientific works before [8].

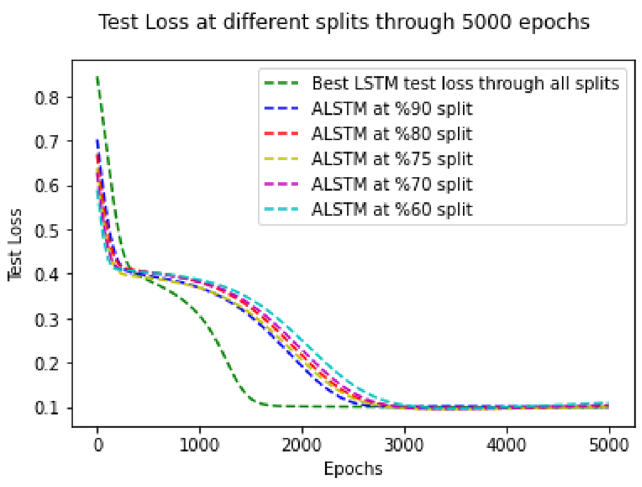

Figure 8: Test loss at different splits through 5000 epochs

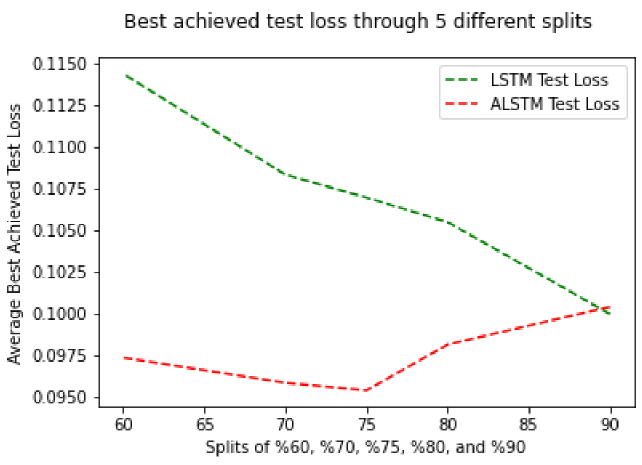

Figure 9: Best achieved test loss through 5 different splits

On Table 1, it is obersvable that the ALSTM performed the best on the split of %75. To have another qualitative metric, Figure 10 displays, just like a stock price chart, how both models forecast the number of passengers on a %75 split. This is displayed on Figure 10. Although the forecasts of LSTM and ALSTM are extremely close, ALSTM is closer to the original line (blue) on some parts.

Sliding Window Approach

In this paper's general approach, the data from time steps 0 to $t_i - 1$ is used to forecast the outcome of time step t_i. However, sliding window approach is a commonly used method in LSTM architecture to convert a data set into a time series.

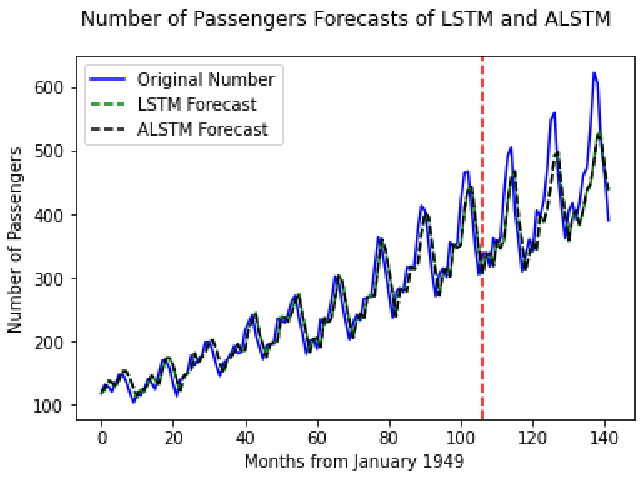

Figure 10: Passenger Number Forecasts of LSTM and ALSTM

In the windowed approach, based on sequence length s. The outcome of the current time step t_i is forecasted based on the time steps $t_(i - s)$ to $t_i - 1$.

In this approach, the results were drastically different. LSTM performed nearly two times better than ALSTM. These can be displayed on the tables and Table 3, Figure 11, and Figure 12.

Train split	LSTM Test Loss Average	ALSTM Test Loss Average
0.90	0.088	0.154
0.80	0.091	0.151
0.75	0.093	0.148
0.70	0.101	0.148
0.60	0.109	0.145

Table 3: Average test loss through different training splits under sliding window approach

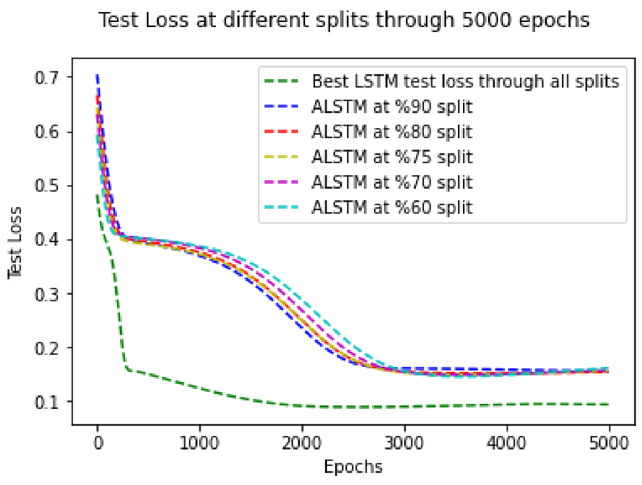

Figure 11: Best achieved test loss through 5 different splits under sliding window approach

Contrary to Figure 10, there is a distinct seeable advantage

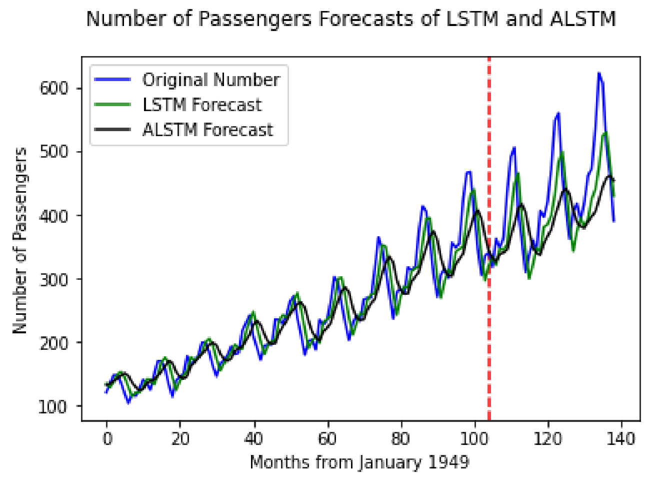

Figure 12: Passenger Number Forecasts of LSTM and ALSTM under sliding window approach

of LSTM over ALSTM on Figure 12. This is both due to a slight performance increase of LSTM, and a drastic performance decrease of ALSTM.

Also, LSTM's performance at %75 split that is displayed on Table 3, is even better than ALSTM's performance in the non-windowed approach. This is reasonable in the sense that, the sequntial, non windowed approach should favor ALSTM on paper due to its ability to pass information between all time steps, whereas LSTM architecture only allows for one by one transmission.

5 Responsible Research

5.1 Research Integrity

During this research, university resources like library articles, and books were used. There was no use of pirated research papers. As this was a research that did not need any human on animal data, no one was harmed in that sense. Public health and safety, is considered when producing the results of the research. And it is safe to say that this research does not harm public health or safety in any way as it just uses flight passenger data. During the research process, no copyrighted work is used without referencing or permission. None of the data is copied from anywhere, and everything is produced by the researcher.

5.2 Reproducibility

The results produced in this project can easily be reproduced by following the train split ratios specified in the experimentation section. If a researcher has the implementations of an LSTM and an ALSTM, they can use these numbers to derive similar results from the same data. The experimentation process is specified step by step.

6 Discussion

In the Experimentation section, the yielded results have shown that ALSTM has an advantage over LSTM in the scope of this project. However, LSTM performs much better under a sliding windowed approach, which only uses previous $t_(i-s)$ time steps to predict t_i.

6.1 Performance Comparison in the Non-Windowed Approach

The new ALSTM architecture to allow information transmission from all time steps 0 to t_i. Thus, to see if that works as expected in practice, the method to convert the data set into a time series, was not the sliding window approach, which is common in time series forecasting tasks. Instead, a fully sequential method is used. What this means is that, the forecast of time step t_i was done based on all of the time steps 0 to $t_i - 1$. This is called the non-windowed approach in this paper. In a sense, this approach can be thought of as a sliding window of sequence length increasing by 1 each time step so that the window contains all previous time steps.

In the non windowed approach, it is seen that ALSTM has a big advantage over LSTMs. This adheres to the hypothesis made in this research. The average of ALSTM for different splits for was 2 standard deviations seperated, or smaller, than the average of LSTM. This suggests that it is indeed a reasonable claim to conclude ALSTM performs better than LSTM in this approach.

However, on the quantitative analysis side, the difference in the forecasted values of ALSTM and LSTM is so tiny that the two lines on Figure 10 are not distinguishable. This raises the question on if the advantage gained by ALSTM is significant enough, and leads to the idea of ALSTM model and architecture implemented in this project can be improved further so that the accuracy of ALSTM will become more superior to that of LSTM. As can be seen on Figure 10, ALSTM performed extremely good for the training set, but the forecasts on the test set still have room for improvement. Also, Figure 8 suggests that ALSTM's learning was slower compared to LSTM, in terms of the reduction in the test loss.

Overall, this experiment showed that ALSTM has superior performance to LSTM's performance, but ALSTM architecture should be further improved to make this difference more significant.

6.2 Performance Comparison in the Sliding Window Approach

While this is not the main focus of this research, the sliding window approach is also utilized to convey how ALSTM makes a difference when the data is in the structure that time step t_i can gather information from all previous timesteps.

This thinking turned out to be correct, as LSTM has shown significant advantage over ALSTM in terms of performance, and LSTM's performance reduced by almost %100, while LSTM performance increased slightly. It is important to note that, if the best test loss scores of both windowed and non windowed approaches are compared, LSTM's test loss of 0.088 on Table 3 is better than ALSTM's loss of 0.095 on Table 2. Also, on Figure 12 it can be seen that the performance deficit between LSTM and ALSTM in the sliding window approach is much higher, in the favor of LSTM architecture.

7 Future Work and Conclusion

7.1 Future Work

In this research, ALSTM's performance is only tested within a small dataset over a monthly forecasting task. Because of this, the results of this research may not reflect the ability of ALSTM in general, and more research should be put into it to discover its capabilities within other scopes and domains of machine learning, and within other real life cases.

Also, the ALSTM implementation done as well as its architecture can be improved further so that ALSTM can perform a bit better not only under sliding window approach, but also on non-windowed approach in a sense that it will be able to forecast much more accurately. Lastly, more work on finding out scenarios which ALSTM performs better than its alternatives can be done, either by faster training, or higher performance.

7.2 Conclusion

In this paper, the performance of Attention Long Short Term Memory (ALSTM) and Long Short Term Memory (LSTM) architectures are compared in the scenario of forecasting the passenger numbers. ALSTM is designed to have better information transmission compared to LSTM due to its ability to pass information from all previous time steps to the current time step. That is why the hypothesis made in this research, "ALSTM will perform better compared to LSTM in the forecast task of flight passenger numbers" was reasonable on paper with a correctly structured time series data set. After experiments, it was seen that the hypothesis held, and the research was successful in the sense that capabilities of ALSTM versus LSTM were discovered. Although ALSTM's performance was better in terms of the average loss, this improvement was not super high.

To further prove that our hypothesis held and was reasonable, a sliding window approach was also tried alternatively. It is observed that because this approach uses limited data, it also limits ALSTM's ability to transmit information from all previous time steps, which is the main design goal behind ALSTM's architecture. Thus, ALSTM performed drastically worse under this approach, making LSTM a better alternative to use under a sliding window approach.

From this paper, it can be concluded that within the scope of this project, ALSTM performed better than LSTM, but there is still room for improvement.

References

[1] Simard P. Bengio, Y. and P. Frasconi. *Learning long-term dependencies with gradient descent is difficult*, 1994.

[2] Maiorino E. Rizzi A. Bianchi F., Kampffmeyer M. *Recurrent Neural Network Architectures*.

[3] S. Hochreiter and Schmidhuber. *Long short-term memory*, 1735–1780, 1997.

[4] Sungho Jung. *Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting*, 2019.

[5] Y.Bengio R. Pascanu, T. Mikolov. *On the difficulty of training recurrent neural networks*, 2013.

[6] J.; Sankar S.; Barfett J.; Colak E. Salehinejad, H.; Baarbe and S Valaee. *Recent advances in recurrent neural networks*, 2018.

[7] N.; Uszkoreit J.; Jones L.; Gomez A. N.; Kaiser L. Vaswani A.; Shazeer, N.; Parmar and Polosukhin. *Attention is all you need. In Advances in neural information processing systems*, 2017.

[8] Oriol Vinyals Wojciech Zaremba, Ilya Sutskever. *Recurrent Neural Network Regularization*, 2015.

[9] Luong Hoang Yoon Kim, Carl Denton and Alexander M. Rush. *Structured attention networks. In International Conference on Learning Representations*, 2017.