Delft University of Technology, Bachelor Seminar of Computer Science and Engineering

Improving the performance of Recurrent Neural Networks for time series prediction by combining Long Short-Term Memory and Attention Long Short-Term Memory

Aaron van Diepen

Abstract

Recurrent neural networks (RNNs) used in time series prediction are still not perfect in their predictions and improvements can still be made in the area. Most recently transformers have led to great improvements in the field of RNNs, however transformers can not be used on time series data, because the architecture of transformers does not account for the flow of time and would use future data to predict past events. This research aims to further improve the performance of machine learning models on time-series prediction. It attempts to do so by implementing a new neural network model based on the multi-head attention mechanism (used in transformers) and combining it with an already existing neural network model called long short term memory (LSTM). To test whether the newly implemented models have improved performance they are tested on a weather dataset and compared on their ability to correctly predict daily maximum temperatures. The final results however show that combining LSTM and ALSTM models does not results in an improved loss that is worth the extra instability that is added to the model and the extra computational cost that is needed to train the model.

1 Introduction

Recurrent neural networks (RNNs) are machine learning models which learn from experience to solve a variety of previously unsolvable problems. RNNs can use their internal state (memory) to process variable length sequences of inputs into variable length outputs. They are already often used by large tech-companies such as Google and Microsoft in order to improve image recognition, text translation and other knowledge-based reasoning problems.

Attention Long Short Term Memory (ALSTM) is a new model in the field of Recurrent Neural Networks, this new model will be thoroughly discussed in Chapter 2 of this paper. The performance of the model had never been analyzed before, therefore its possible value to machine learning is yet to be uncovered. The ALSTM is a modification of the already well established multi-head attention mechanism [1]. By modifying the existing multi-head attention mechanism ALSTMs can easily focus on the important parts of the input data while partially ignoring the less important input. This modification is done in order to prevent future input from being used to predict past values. In this paper the ALSTM will be further discussed and, together with the LSTM and different combinations of the LSTM and ALSTM, it will be evaluated on a time series dataset. Combining the ALSTM and LSTM may potentially further improve the accuracy of the output for these kinds of problems.

The aim of this research is to find out whether a combination of Long Short Term Memory with Attention Long Short Term Memory improves accuracy for time-series classification using Machine Learning as compared to using either separate from each other.

As previously shown in other research papers, such as [2]–[4] combining multiple machine learning models can improve accuracy. By undertaking this project I hope to analyse whether the extra time spend training a complexer model is worth running a combination of an LSTM and ALSTM in order to achieve higher accuracy outcomes for machine learning tasks.

In this work three different forms of combining the LSTM and ALSTM models are proposed, in which their performance is either combined by running them in parallel or by feeding one networks output as an additional input into the other network. These resulting models (as well as the basic LSTM and ALSTM) are applied to a time series problem in which a daily maximum temperature is predicted. After which the accuracy of the results are compared in different ways.

2 Background

In this section the background for the in this project implemented attention long short-term memory (ALSTM) as well as the already wide spread long short-term memory (LSTM) will be discussed in detail.

2.1 RNN

RNNs are a recurrent network consisting of multiple cells, they process sequential information by passing through information about the previous inputs as an extra input to the next cell. This data is then used as extra information for predicting the next output.

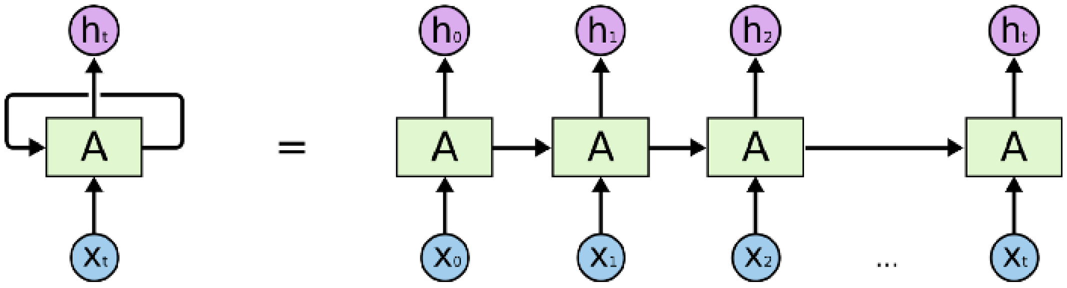

Figure 1: Graphical overview of the way RNNs process input [5]

2.2 LSTM

The already existing architecture used in this project is the LSTM, which was first introduced in 1997 by Hochreiter and Schmidhuber [6]. It is a form of RNN that uses a memory vector and a hidden state in order to transmit data to the next cell. It can be mathematically explained according to Formula 1, where the hidden state for input t is defined as h_t and the memory vector for input t is defined as c_t. It can also be explained as a graphical overview shown in Figure 2. It can be simply explained as being a combination of 3 gates, namely the forget gate, input gate and output gate. The forget gate scales the previous cell state based on hidden state, input and a learned bias. After "forgetting" part of the previous cell state the input gate adds a new value to the cell state again based on the hidden state, input and a two additional learned biases. Finally each cell calculates its new hidden state based on the newly calculated cell state, hidden state, input and a fourth learned bias. These gate calculations are repeated for every cell used in the full network to output the h_t for each input X_t.

$$\begin{aligned} &h_t := g_O(x_t, h_{t-1}) \odot \sigma_H(c_t), \quad h_0 := 0, \quad c_0 := 0 \\ &c_t := g_F(x_t, h_{t-1}) \odot c_{t-1} + g_I(x_t, h_{t-1}) \odot g_C(x_t, h_{t-1}) \\ &g_z(x, h) := \sigma_z(\Theta_z x + U_z h + b_z), \quad \forall z \in \{I, O, F, C\} \end{aligned} \tag{1}$$

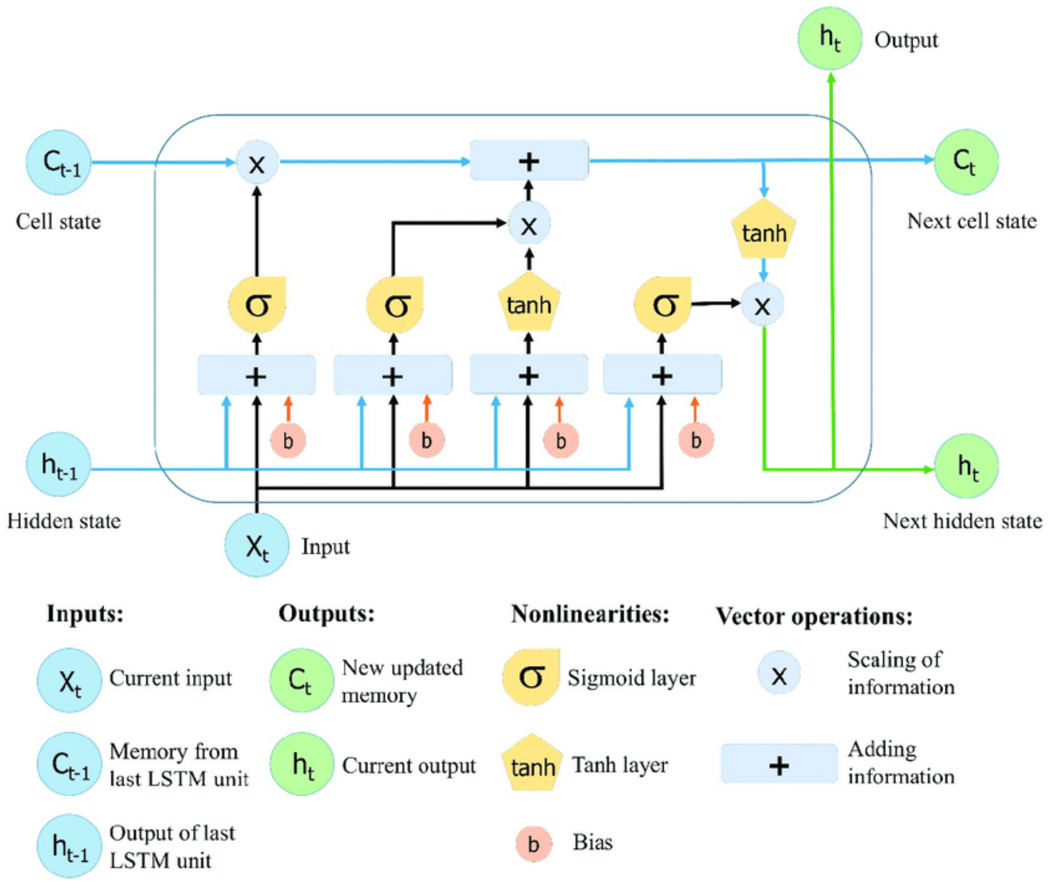

Figure 2: Graphical overview of internal mathematical formulas of a single LSTM cell [7]

2.3 Multi-Head Attention

In order to introduce the ALSTM first multi-head attention has to be introduced. Multi-Head Attention (MHA) is based on the attention mechanism, specifically Scaled Dot-Product

Attention, in which keys and queries are mapped to an output based on a weighted sum of the values. The weight assigned to each value is determined by the scaled dot-product of the query with all the keys. MHA simply runs through scaled dot-product attention multiple times. The outputs of these scaled dot-product attention networks are simply concatenated and linearly combined to form the output of the MHA network as shown in Figure 3. The final equations used in MHA are shown in Equation 2, where W^k represents the normalized weight matrix, Q the matrix representing the queries, K the matrix representing the keys, V the matrix representing the values and V' is the final output of the model.

$$\begin{aligned} &V' := \sigma_V(\textstyle\sum_{k=1}^{m'} W^k \sigma_C(V\Theta^k)) \\ &W^k := S^k \oslash (S^k 11^\intercal) \\ &S^k := exp(\tfrac{1}{\sqrt{d}} V Q^k K^k V^\intercal) \end{aligned} \tag{2}$$

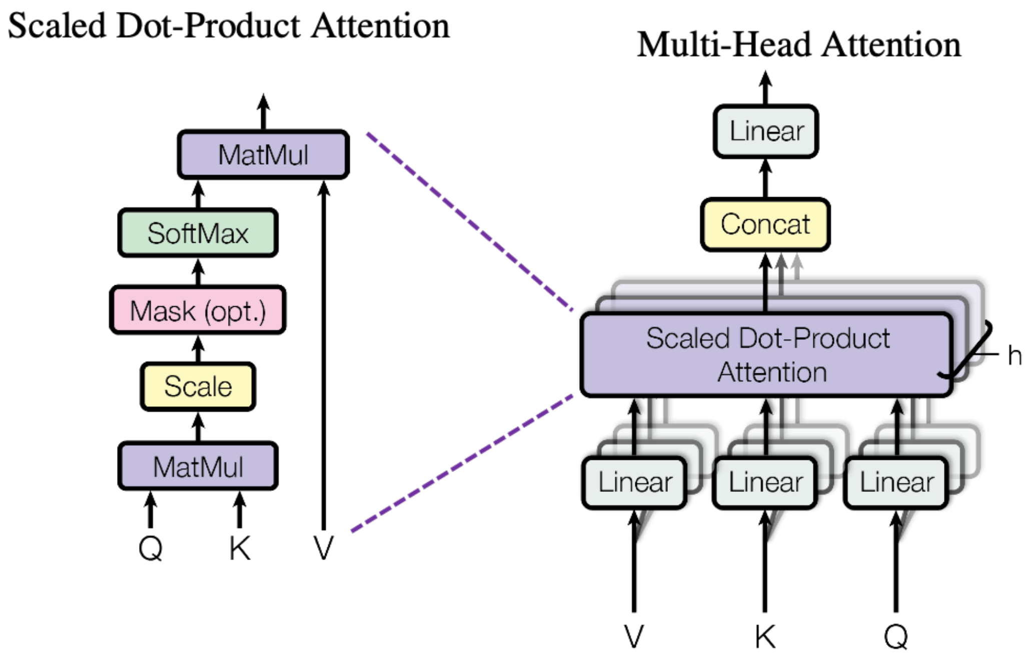

Figure 3: Graphical overview of internal formulas of multi-head attention [8]

2.4 ALSTM

The ALSTM is based on a simplification of the long short-term memory, that when simplified starts to closely resemble an attention network. The main difference between the ALSTM and standard multi-head attention layers is a multiplication by a lower-triangular matrix of the weight matrix, this multiplication removes the upper triangular weight matrix in order to preserve the sequentially of the input. The modified equations used for the ALSTM are shown in Equation 3, in which V has been replaced by X in order to show the correlation to x_t used in the notation for the LSTM to represent the input.

$$\begin{aligned}H &:= \textstyle\sum_{k=1}^{m'} W^k \sigma_C(X\Theta^k)\\ W^k &:= S^k \oslash (S^k 1 1^T)\\ S^k &:= exp(\tfrac{1}{\sqrt{d}} X Q^k K^k X^T) \odot L)\end{aligned} \tag{3}$$

The potential improvement for the ALSTM would be to allow for better access to the prior time series, thus this new architecture should perform better than the LSTM.

3 Methodology

In this section all relevant concepts, theory and models will be described and discussed.

An existing LSTM implementation will be used, as provided by PyTorch. An ALSTM compatible with this library, in order to keep consistency while testing, has been implemented. The implementation of the ALSTM network will be made available on gitlab in order for the experiments to be reproducible.

After implementing the ALSTM, the LSTM and ALSTM are combined in 3 different ways. The first one will be with the two models in parallel fed into a linear layer resulting in a single prediction in the time-series, as shown in Figure 4. The other two will be with the two models in series, with the features and the result of the first model as an input for the second model. One with the LSTM followed by the ALSTM (as shown in Figure 5) and one with the ALSTM followed by the LSTM (as shown in Figure 6).

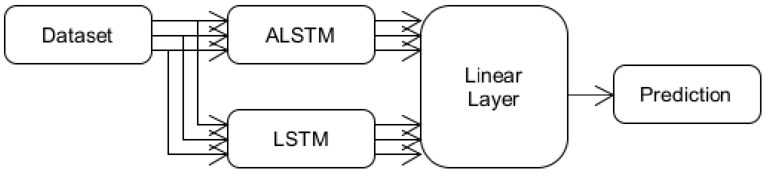

Figure 4: Combination 1

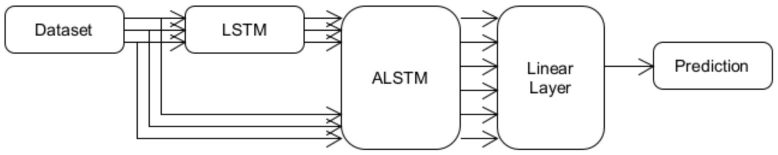

Figure 5: Combination 2

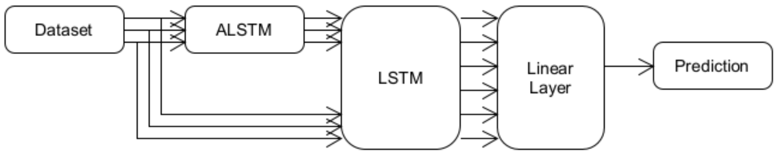

Figure 6: Combination 3

The models will be evaluated multiple times on data split using a time series split in order to achieve a more robust evaluation of the machine learning models. A time series split is the equivalent of k-fold cross validation but adapted in such a way that the training data is never ahead of the testing data as shown in Figure 7. It is a resampling procedure used to evaluate machine learning models on a limited data sample. A number of 5 splits will be used in order to achieve an optimal balance between computational cost and bias as recommended by Rodríguez, Pérez, and Lozano [9]. All models will be trained on each split selected, and there will be no communication of values between splits.

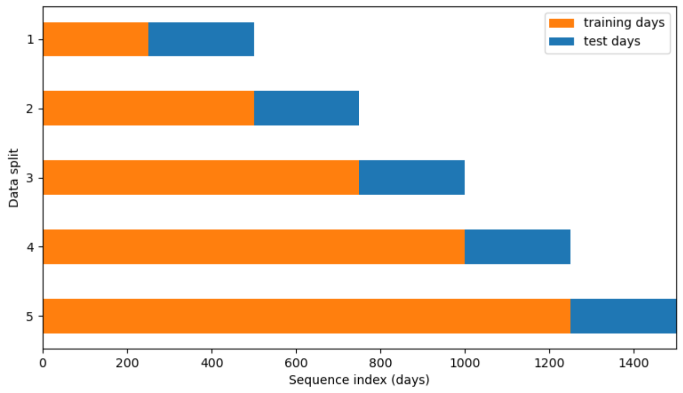

Figure 7: The 5 training and test data splits used for training, this method is also called time series split

All models will be trained in Python on the Ozone Level Detection Data Set from the UCI Machine Learning Repository [10]. Preproccesing will need to be done on the dataset in order to be able to run all models on it, for each model the same preproccesed dataset will be used. After the 4th year the dataset contains a lot of gaps in the data, therefore it was opted for to only use the more accurate first 1500 days of data in this research. This data still contained a few gaps which were filled up by propagating the last valid observation forward. The models will be trained on predicting maximum daily temperature of the next day in the series given all previous values in the series. The error will be calculated using the mean squared error, and the optimizer used to train the model is ADAM, which is a replacement optimization algorithm for stochastic gradient descent. All models will use a total of 2 hidden layers which should give a rough understanding of how well each model performs while keeping computational complexity low.

Model	Split 1	Split 2	Split 3	Split 4	Split 5
LSTM	2e-5	2e-5	2e-5	2e-5	2e-5
ALSTM	1e-3	5e-4	2e-5	1e-3	5e-3
Combination 1	4e-5	1e-4	1e-3	5e-5	1e-3
Combination 2	1e-6	2e-5	5e-5	6e-3	1e-2
Combination 3	4e-6	5e-6	2e-6	2e-6	7e-6

Table 1: Models and their specific learning rates selected to converge between 1000-3000 epochs

Because all models differ in their convergence speed it was decided to have a custom learn-rate for each model in order to clearly display how well the models could perform in best case scenario's. The learn-rate was picked such that the model reaches its minimum test loss within the first 3000 epochs. The final learning rates used for each model are displayed in Table 1.

4 Results

This section displays the results of the different neural architectures as described in the previous chapter.

Table 2 shows the time each models took per split for training 3000 epochs as well as showing the total time each model took to train across all five splits while running in a gpu accelerated runtime on google colab it is intended to give a rough estimate of how long the models take to train.

Model	Split 1	Split 2	Split 3	Split 4	Split 5	Total
LSTM	80	110	154	201	247	792
ALSTM	37	97	180	310	469	1093
Combination 1	99	196	325	485	679	1784
Combination 2	165	376	670	1049	1522	3782
Combination 3	99	195	322	487	677	1780

Table 2: The time it took each model to process 3000 training cycles in seconds (on a gpu accelerated runtime in google colab)

Figures 8-16 show the training and test losses over the training cycles (epochs) together with their minimum test loss achieved and a graph showing the prediction of the model with the overall lowest achieved test-loss at their respective epochs. The first image is intended to display the way in which the models converge and give an overview of how the models compare to each other.

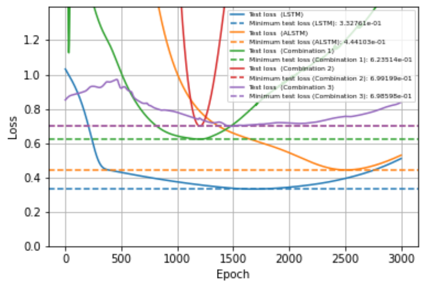

Figure 8: Testing loss graphs for all models on split 1 and their minimum loss achieved

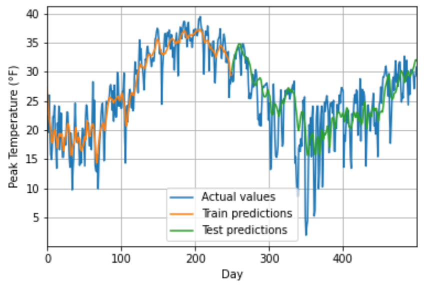

Figure 9: Prediction results of the LSTM (best of split 1)

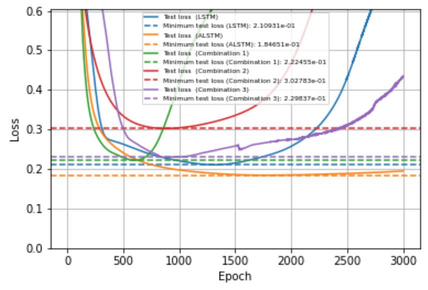

Figure 10: Testing loss graphs for all models on split 2 and their minimum loss achieved

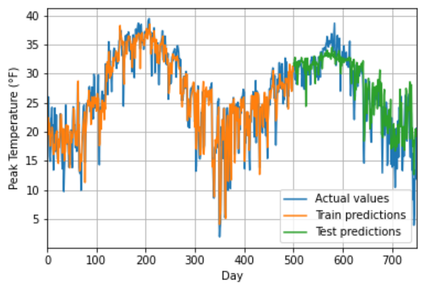

Figure 11: Prediction results of the ALSTM (best of split 2)

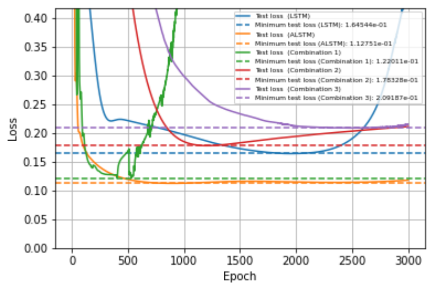

Figure 12: Testing loss graphs for all models on split 3 and their minimum loss achieved

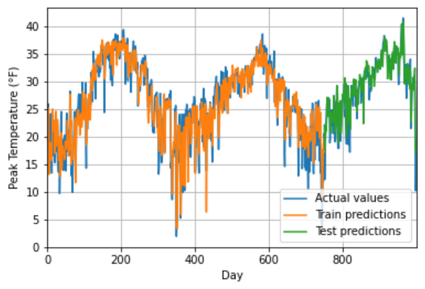

Figure 13: Prediction results of the ALSTM (best of split 3)

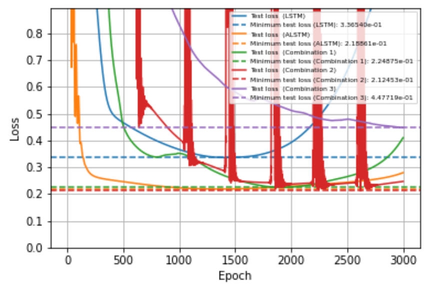

Figure 14: Testing loss graphs for all models on split 4 and their minimum loss achieved

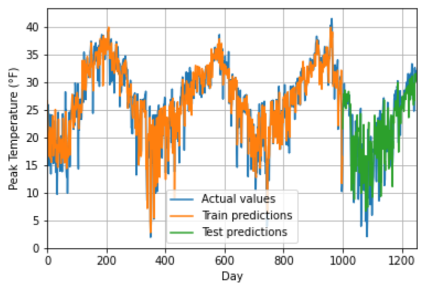

Figure 15: Prediction results of Combination 2 (best of split 4)

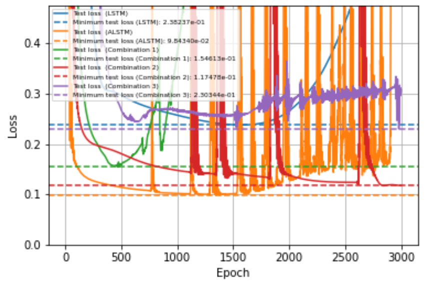

Figure 16: Testing loss graphs for all models on split 5 and their minimum loss achieved

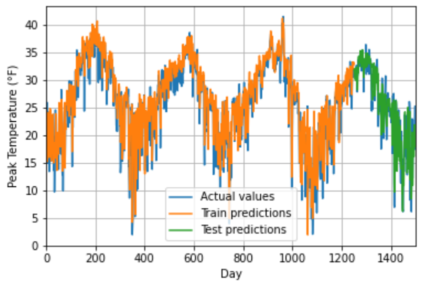

Figure 17: Prediction results of the ALSTM (best of split 5)

Table 2 shows the epoch at which each model reached their lowest test loss for the different splits. Its purpose is to give a rough understanding of how the learning rates of Table 1 were selected.

Model	Split 1	Split 2	Split 3	Split 4	Split 5
LSTM	1675	1315	1949	1405	1574
ALSTM	2510	1780	901	1349	1347
Combination 1	1199	600	404	1914	470
Combination 2	1201	879	1206	2620	2844
Combination 3	1487	858	2506	2999	2979

Table 3: The epoch at which each model reached their lowest test-losses.

Finally table 4 shows the minimum test-losses achieved after 3000 epochs of training the models for each split, as well as the average minimum test-loss for each model across all splits. In order to give an overview of best performing model across all splits.

Model	Split 1	Split 2	Split 3	Split 4	Split 5	Average
LSTM	3.33E-01	2.11E-01	1.65E-01	3.37E-01	2.38E-01	2.57E-01
ALSTM	4.44E-01	1.85E-01	1.13E-01	2.19E-01	9.84E-02	2.12E-01
Combination 1	6.24E-01	2.22E-01	1.22E-01	2.25E-01	1.55E-01	2.69E-01
Combination 2	6.99E-01	3.03E-01	1.78E-01	2.13E-01	1.17E-01	3.02E-01
Combination 3	6.99E-01	2.30E-01	2.09E-01	4.48E+00	2.30E-01	1.17E+00

Table 4: Models and their minimum test-losses across all different splits

5 Responsible Research

There are a few ethical questions to be asked when working with machine learning models, most of them speculating about what the projects can be used for. One of the main concerns is that the final results of training the models have no clear algorithm associated with them that makes easily understandable decisions. Therefore a model used to make actual predictions might make an inexplicable mistake that can have big consequences when fully relied upon in an actual use-case. This risk should be taken into account when relying on a machine learning solution to a problem.

"Garbage in, garbage out" (GIGO) should also be accounted for when one of the models is used in a real life situation. This concept is in principle about the data that the model is trained on and reinforces how users should keep in mind that their input data should be kept clean of any unwanted biases to prevent the model from adapting these biases.

The method should be fully reproducible as all details needed in order to reproduce the analysis (such as how many splits are used, what preprocessing has been done to the data, how the attention long short term memory has been implemented and how the combinations were created) are included in this document.

The dataset used for training the models is freely available for download from https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection [10].

6 Discussion

In this section the results will be compared in order to get to a conclusion on whether training a combination of the LSTM and ALSTM is worth the additional time spend training a model (that is more complicated) in order to potentially lower losses for time-series prediction.

As can be seen in table 2 the ALSTM is quicker than the LSTM at training on small sequences, which is likely due to the level of parallelisation the ALSTM is capable of that is better utilised on small sequences. The series combinations of the models (combinations 2 & 3) also take more time than the parallel combination of the models (combination 1).

Overall the models took a long time to train, therefore the learning-rate has not been fully tweaked optimally. The results from the different models could thus still be improved a lot by further tweaking of the learn-rates.

As can be clearly seen in figures 16 the ALSTM does not really converge but instead oscillates by slowly learning then making a short step in the wrong direction and then quickly correcting that mistake. This result can most likely be explained by the ALSTM not being able to store sequences in its memory cells like the LSTM can, therefore further training can easily decrease the performance of the model by wrongly adjusting a weight by a small amount. Which can be easily undone by further training resulting in big spikes.

As seen in table 4 the new architecture called ALSTM achieves a better minimum loss on average than the LSTM. However when trained on a small training sequence, as seen for split 1 in Table 4, the LSTM performs better than the ALSTM. Which is likely caused by the fact that a shorter sequence of prior data can more easily be represented by the cell state used in the LSTM and thus smaller advantage is gained by the ALSTM having access to all prior time steps.

The combination in which the ALSTM and the LSTM are used in parallel (combination 1) has the best performance among the combinations. This combination most likely has the best performance because the linear layer has the clearest correlation between input and output. As well as having the best performance this combination is the fastest to execute because both models can be executed in parallel, instead of either having to wait for the other to finish processing.

Overall because the training times are so increased for the combinations of the architectures as well as gaining no additional performance improvements when looking at total loss for test predictions as well as the learn-rates being harder to tweak, we can conclude that the increased difficulty of combining the LSTM model with the new ALSTM model is not worth the effort and time spend on training and tweaking values. This is most likely due to the LSTM and ALSTM being too close in the type of calculations they perform, such that combining them does not offer any valuable new way for the models to learn the relations between input and output.

7 Conclusions and Future Work

Overall possibly valuable insight has been achieved on the combination of different machine learning models. It shows that in the case of combining alstm and lstm it does not actually offer a lot of extra value compared to the extra costs and complexity of the model.

Further work could be done tweaking learning-rates in order to possibly still get slight reductions in the test-losses for the combinations of the LSTM and ALSTM, however seeing as any loss reduction would not likely be very high this might not be worth the effort unless no other potential ways to decrease test-losses for RNN's can be thought of.

Other possible work that could be done is research into whether the ALSTM can be adjusted such that it can be executed in a traditional, sequential way, such that predicting single next values in a sequence can be done without recalculating all previous values.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *ArXiv*, vol. 1409, Sep. 2014.

[2] X. Li, L. Wang, Y. Xin, Y. Yang, Q. Tang, and Y. Chen, "Automated software vulnerability detection based on hybrid neural network", English, *Applied Sciences (Switzerland)*, vol. 11, no. 7, 2021. [Online]. Available: `www.scopus.com`.

[3] T. Gu, X. Zhao, W. B. Barbazuk, and J. .-. Lee, "Mitar: A hybrid deep learning-based approach for predicting mirna targets", English, *BMC Bioinformatics*, vol. 22, no. 1, 2021. [Online]. Available: `www.scopus.com`.

[4] Y. Qiu, H. .-. Yang, S. Lu, and W. Chen, "A novel hybrid model based on recurrent neural networks for stock market timing", English, *Soft Computing*, vol. 24, no. 20, pp. 15 273–15 290, 2020, Cited By :2. [Online]. Available: `www.scopus.com`.

[5] P. Radhakrishnan, *Introduction to recurrent neural network*, Aug. 2017. [Online]. Available: `https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3`.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: `10.1162/neco.1997.9.8.1735`.

[7] X. H. Le, H. Ho, G. Lee, and S. Jung, "Application of long short-term memory (lstm) neural network for flood forecasting", *Water*, vol. 11, p. 1387, Jul. 2019. DOI: `10.3390/w11071387`.

[8] Y. Tamura, *Multi-head attention mechanism: Âqueriesâ, âkeysâ, and âvalues,â over and over again*, Apr. 2021. [Online]. Available: `https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/`.

[9] J. Rodríguez, A. Pérez, and J. Lozano, “Sensitivity analysis of k-fold cross validation in prediction error estimation”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 569–575, Apr. 2010.

[10] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: `http://archive.ics.uci.edu/ml`.