INSA Summer Research Fellowship Program
2023
8 Weeks Report

Application number:	ENGS2204
Summer Research Fellow:	Srinithi S L
College:	National Institute of Technology, Tiruchirappalli
Academic Discipline:	Electrical and Electronics Engineering
Personal Email :	slsrinithi1912@gmail.com
College Email:	107121104@nitt.edu
Date Performed:	May 23, 2023 to October 13, 2023
Guide:	Professor R. Ganesh, IPR-Gandhinagar

Acknowledgements

I am deeply grateful to Dr. Pawandeep Kaur, a Postdoctoral fellow at the Weizmann Institute of Science, Israel, for her valuable inputs, expert advice, and constructive feedback. Her mentorship and scholarly contributions have significantly enriched this report.

Turbulence prediction with Recurrent Neural Networks

October 13, 2023

1 Introduction

Turbulence modeling and prediction aims to develop more accurate and efficient techniques for understanding and forecasting turbulence. By gaining a deeper understanding of the underlying dynamics and structures of turbulent flows, researchers can improve the design and optimization of various engineering systems. In the context of the specific research paper titled "Recurrent neural networks and Koopman-based frameworks for temporal predictions in a low-order model of turbulence," **Eivazi et al. 2021**[3], the authors investigate the application of recurrent neural networks and Koopman-based frameworks for predicting the temporal behavior of turbulence in a low-order model. In the research project, LSTM and RNNs play a significant role in capturing the temporal dynamics of turbulent flows. By leveraging their ability to model sequential data and capture long-term dependencies, LSTM and RNNs enhance the accuracy and efficiency of turbulence predictions, offering valuable insights into turbulence evolution and enabling better control of turbulent flow systems.

2 Predictions with LSTM

The most basic form of neural network is the multilayer perceptron (MLP), which consists of interconnected layers of nodes or neurons. MLPs are commonly used in practice, but their main drawback is that they are designed for point prediction rather than time-series prediction, which requires a more context-aware approach. However, MLPs serve as a solid benchmark in machine learning applications and help highlight the need for more sophisticated network architectures. In a previous study **Srinivasan et al. 2019**[4], the accuracy of MLP predictions was evaluated using the nine-equation model by **Moehlis et al. (2004)**[5], where various network architectures were used to predict the time evolution of the nine coefficients. RNN(Figure 1) is a type of neural network designed to process sequential data, making it well-suited for time series forecasting. It utilizes feedback connections that allow information to be passed from one time step to the next, enabling the network to capture temporal dependencies and patterns in the data.

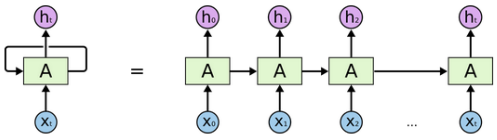


Figure 1: Recurrent Neural Networks. Figure source: **colah2015**.

LSTM (Long Short-Term Memory) (Figure 2)[6] is a type of recurrent neural network (RNN) architecture that is specifically designed to address the vanishing gradient problem and capture long-term dependencies in sequential data. LSTM networks consist of memory cells that store information over time, along with gating mechanisms that control the flow of information into, out of, and within the cells.

The key components of an LSTM cell are:
1. Input Gate (i_t): The input gate controls how much new information is allowed to enter the memory cell. It selectively updates the cell state based on the input and the previous cell state.
2. Forget Gate (f_t): The forget gate controls the amount of old information to forget from the memory cell. It decides which information is no longer relevant and should be discarded.
3. Output Gate (o_t): The output gate regulates the flow of information from the memory cell to the output of the LSTM unit. It determines how much of the current cell state should be exposed as the output.
4. Cell State (c_t): The cell state is responsible for storing and propagating information over time. It acts as the long-term memory of the LSTM unit, allowing the network to capture and remember important patterns or dependencies in the sequential data.
5. Hidden State (h_t): The hidden state is the output of the LSTM cell. It contains information about the current time step and is used for making predictions or passing information to subsequent LSTM cells in a sequence.

Mathematically, the LSTM cell equations can be represented as follows:
1. Input Gate Equation:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$

2. Forget Gate Equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

3. Output Gate Equation:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3}$$

4. Cell State Update Equation:

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$

5. Cell State Equation:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{5}$$

6. Hidden State Equation:

$$h_t = o_t \cdot \tanh(c_t) \tag{6}$$

In these equations, x_t represents the input at time step t, h_{t-1} represents the hidden state at the previous time step, and σ and tanh are the sigmoid and hyperbolic tangent activation functions, respectively. W_i, W_f, W_o, W_c and b_i, b_f, b_o, b_c are learnable parameters (weights and biases) of the LSTM cell.

These equations, with their respective equation numbers, allow the LSTM cell to perform computations on the input sequence, update its internal state (cell state), and pass information to the next time step, enabling the network to capture long-term dependencies in the data[9].

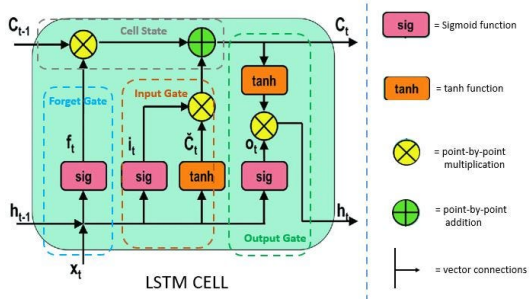


Figure 2: Long Short Term Memory cell. Source: **analyticsvidhya2022**.

3 Gated Recurrent Units (GRU)

Gated Recurrent Units (GRU) (Figure 3) are a type of recurrent neural network (RNN) architecture that, like LSTM, is designed to capture long-term dependencies in sequential data. GRUs are known for their simpler architecture compared to LSTM while still being effective in many sequence modeling tasks [2].
The key components of a GRU cell are:

a. Reset Gate (r_t): The reset gate controls which information from the previous state should be discarded or reset. It helps the network to decide how much of the previous state information is relevant for the current time step.

b. Update Gate (z_t): The update gate regulates the combination of the new input with the previous state. It controls the flow of information from the current input to the current state and helps the model to determine how much of the new state should be used.

c. Candidate Activation ($\tilde{h}_t$): The candidate activation computes a new candidate state based on the current input and the previous state information, as determined by the reset gate r_t.

d. Hidden State (h_t): The hidden state is the output of the GRU cell. It combines the previous hidden state with the candidate activation to form the new state for the current time step.

Mathematically, the GRU cell equations can be represented as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{1}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t]) \tag{3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{4}$$

In these equations, x_t represents the input at time step t, h_{t-1} represents the hidden state at the previous time step, and σ is the sigmoid activation function, while tanh is the hyperbolic tangent activation function. $\odot$ represents element-wise multiplication. W_z, W_r, and W_h are learnable parameters (weights) of the GRU cell.
GRUs are particularly useful when you want a simpler recurrent architecture that still captures essential sequential patterns and dependencies in your data.

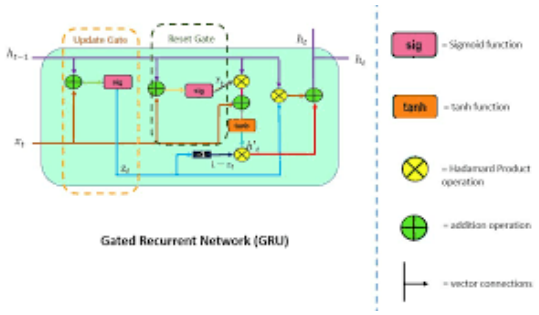


Figure 3: Gated Recurrent Unit. Source: **Pluralsight**.

4 The Setup and Methodology

In the pursuit of developing robust and accurate turbulence prediction models, the foundation lies in the quality and diversity of the training data. Using python, 9 coupled nonlinear Ordinary Differential Equations (Eq.1 of Srinivasan et al. 2019 paper)[4] are solved to generate a single time series of 4000 time steps and 9 coefficients (Figure 4). These ODEs encapsulate the behavior of nine coefficients governing various aspects of turbulent flow. The key to achieving diversity in the dataset lies in systematically varying a single parameter within these ODEs. By doing so, we create 100 unique time series datasets, each with a distinct parameter configuration. This systematic variation ensures that the datasets cover a broad spectrum of turbulent behaviors, ranging from subtle variations to more pronounced changes.

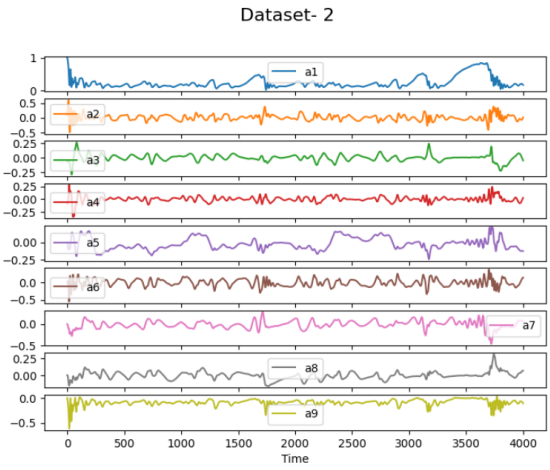


Figure 4: Single time series of 4000 time steps and 9 coefficients.

Besides the coefficient data, the calculation of in-plane and out-of-plane velocity fields (Figure 5) at each time step provides a deeper understanding of the fluid dynamics associated with the coefficient dynamics. These velocity fields are crucial for comprehending the underlying physics and for interpreting the model's predictions in terms of their impact on the actual flow behavior.

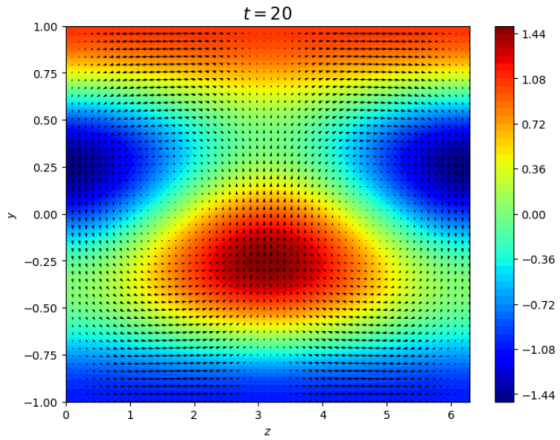


Figure 5: In-plane and out-of-plane velocity fields at time step t=20

To train and evaluate a predictive model, the dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing. For the model to make predictions on this time series data, the TensorFlow library (**TF**)[8] provides a useful function called "TimeseriesGenerator." This function employs the windowing method, which generates input-output pairs from the time series data. The windowing method creates a sliding window of fixed length and slides it along the time

series, generating subsequences of data as inputs and corresponding targets. In this project, we employ data windowing techniques[8] to facilitate both single time step prediction and multiple time step predictions. Specifically, we use a window size of 500 previous time steps as input to the model. For single time step prediction, the model predicts the value of the next time step. For multiple time step prediction, the model forecasts the values for the subsequent 32 time steps. These predictions are made by sliding the window along the time series data and generating input-output pairs for training and evaluation.

5 Model Architecture with LSTM

The model architecture (Figure 6) consists of three layers: one input layer, one LSTM layer, and one output layer. Here is a detailed explanation of each layer used in our project:

a. The *Input Layer* has 9 input nodes, which corresponds to the 9 coefficients of the time series data. The input layer receives a sequence of 500 previous time steps (datapoints) as input, where each time step consists of the 9 coefficients.

b. The *LSTM Layer* consists of 90 LSTM units or cells. Each LSTM unit is responsible for capturing and processing the temporal dependencies within the sequence of 500 previous time steps. The LSTM layer's architecture allows it to remember and utilize information from earlier time steps, capturing the long-term dependencies in the data. The LSTM layer performs computations on the input sequence, updates its internal state, and passes information to the next time step.

c. The *Output Layer* The output layer has 9 nodes, corresponding to the 9 coefficients being predicted. This layer takes the output from the LSTM layer and generates predictions for the next time step values of the 9 coefficients. Each node in the output layer represents a specific coefficient, and the values produced by these nodes are the predicted values for the respective coefficients.

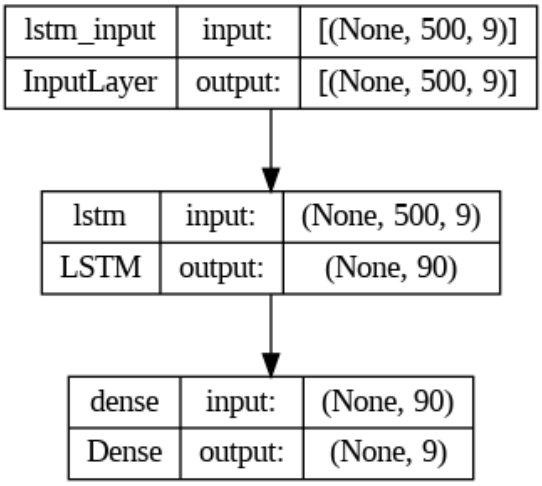


Figure 6: Model Architecture.

The total number of trainable parameters in the model architecture is the sum of the parameters in each layer and equals to 36819, where the number of trainable parameters in LSTM layer and Output Layer are 36000 and 819 respectively.

In the *LSTM layer*: The input-to-hidden connections have 9 input nodes and 90 LSTM units, with each connection having 4 parameters (weight and bias for input, forget, output, and cell state). The hidden-to-hidden connections have 90 LSTM units and each connection has 4 parameters.

$$\text{Total parameters in the LSTM layer} = 9\times90\times4+90\times90\times4 = 36000 \tag{7}$$

In the *Output layer*: The LSTM layer has 90 LSTM units connected to 9 output nodes. Each connection has 1 parameter. Additionally, each output node has its own bias parameter.

$$\text{Total parameters in the Output layer} = 90 \times 9 + 9 = 819 \tag{8}$$

Therefore, the correct total number of trainable parameters in the model is:

$$\text{Total number of trainable parameters} = 36000 + 819 = 36819 \tag{9}$$

The specific model architecture of a single layer LSTM is chosen for turbulence prediction due to its capability to handle sequential time series data, capture long-term dependencies, and facilitate efficient training with a manageable number of parameters[9]. The single layer design strikes a balance between model complexity and computational efficiency, making it suitable for the available dataset size and computational resources. Additionally, the LSTM's memory cells and gating mechanisms enable the model to effectively learn temporal patterns, leading to accurate turbulence predictions[4].

6 Results and Analysis

The model's training process involves minimizing the mean squared error loss (see Eq.10) during each training iteration. The model's weights are adjusted using the Adam optimizer[7], which calculates the gradients based on the loss function and updates the model parameters to minimize the loss (figure 7 & figure 8). The training progresses through the specified number of epochs, with the aim of optimizing the model's performance and reducing the mean squared error between the predicted and actual values of the time series. The training process employs the Mean Absolute Error (MAE) see Eq.11) metric to gauge the model's performance. The early stopping callback compares the validation loss between epochs and stops training if the loss does not improve for the specified number of epochs (patience). Early stopping helps prevent over-fitting and improves training efficiency by stopping the process when further training does not lead to significant improvements in the validation loss.

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10}$$

$$\text{Mean Absolute Error} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{11}$$

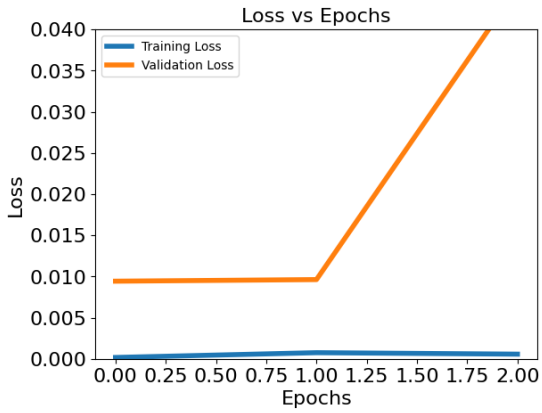


Figure 7: Continuous Error Reduction of the Single Time-Step Model with Epoch.

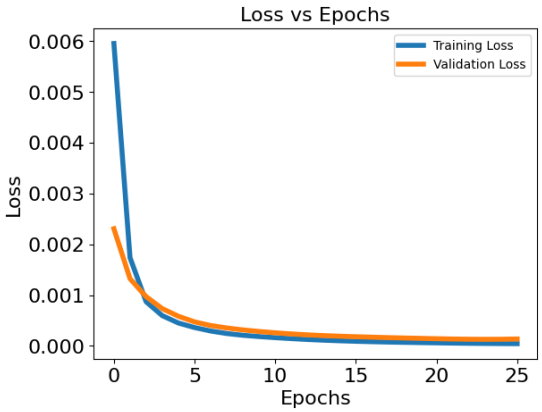


Figure 8: Continuous Error Reduction of the Multistep Time-Step Model with Epoch.

However, when encountering a scenario where the validation loss for the single-step model rises (Figure 4) while the training loss continues to decline (as seen in Figure 4), this signals clear overfitting. Overfitting occurs when the model becomes too focused on training data specifics, impeding its ability to capture fundamental patterns applicable to unseen or test data. To address this overfitting challenge, a savvy approach is employed. It harnesses ensemble methods and trains the model using a diverse set of 100 time series data. This entails creating multiple models, each with distinct configurations, and leveraging them to produce predictions. These individual predictions are then combined through averaging, effectively curbing variance and bolstering the model's overall generalization capacity. This ensemble technique results in more robust and dependable turbulence predictions.

Single-step predictions and multi-step predictions are two different approaches used in time series forecasting.

a. *Single-step predictions:* In this approach, the model is trained to predict the value of the next time step given the input sequence of previous time steps. The model takes a fixed window of historical data as input and predicts the value at the very next time step. After making the prediction, the window slides by one time step, and the process is repeated for the next prediction. This method is suitable when we are interested in short-term predictions and immediate future values.

b. *Multi-step predictions:* Multi-step predictions involve forecasting multiple future time steps ahead of the current time step. Instead of predicting just the next value, the model is trained to forecast a sequence of values into the future. The input sequence contains a fixed window of historical data, and the model predicts the values for several future time steps in a single pass. This approach is useful for long-term forecasting, where we want to understand the trends and behavior of the time series over an extended period.

The trained models for both the single-step and multi-step predictions are evaluated using the test data, and the error metrics are obtained: mean squared error (MSE) and mean absolute error (MAE) (Table 1).

Prediction Model	**Time Series Data**	**Number of Epochs**	**MSE(%)**	**MAE(%)**
Multi time-step model	1	6	0.014	0.833
Multi time-step ensemble model	50	6	0.000399	0.153765
Multi time-step ensemble model	100	6	0.000151	0.093676
Single time-step model	1	4	6.257	19.192
Single time-step ensemble model	50	4	0.075225	2.238164
Single time-step ensemble model	100	4	0.050211	1.896842

Table 1: Evaluation of Trained Models for Single-Step and Multi-Step Predictions using Test Data

Graphs are generated to display the predicted values for the nine coefficients in the test dataset. The graphs illustrate the forecasted time series data and include predictions for both single-step and multi-step predictions. A total of 300 timesteps are covered in the plots.

The comparative analysis between Single Time Step Prediction (depicted in Figure 9) and Single Time Step Ensemble Prediction (represented in Figure 10) reveals a striking contrast. The Single Time Step Ensemble Prediction method emerges as a remarkably robust and precise forecasting technique, demonstrating superior accuracy and reliability when juxtaposed with the traditional Single Time Step Prediction.Moreover, a more profound exploration of the graphical representations in Figure 10 and Figure 12 uncovers a compelling revelation. Figure 12, showcasing Multi Time Step Predictions, presents a considerably more comprehensive portrayal of the future trajectory of the time series in comparison to single time step predictions(Figure 10).

Multi Time Step Predictions, exemplified in Figure 12, impart a smooth and coherent pattern over the forecasted horizon, adeptly encapsulating the underlying trends and intricate dynamics inherent within the dataset. In contrast, Figure 10's single time step predictions may exhibit heightened volatility and short-term fluctuations, as they are primarily concerned with forecasting a single time step ahead.

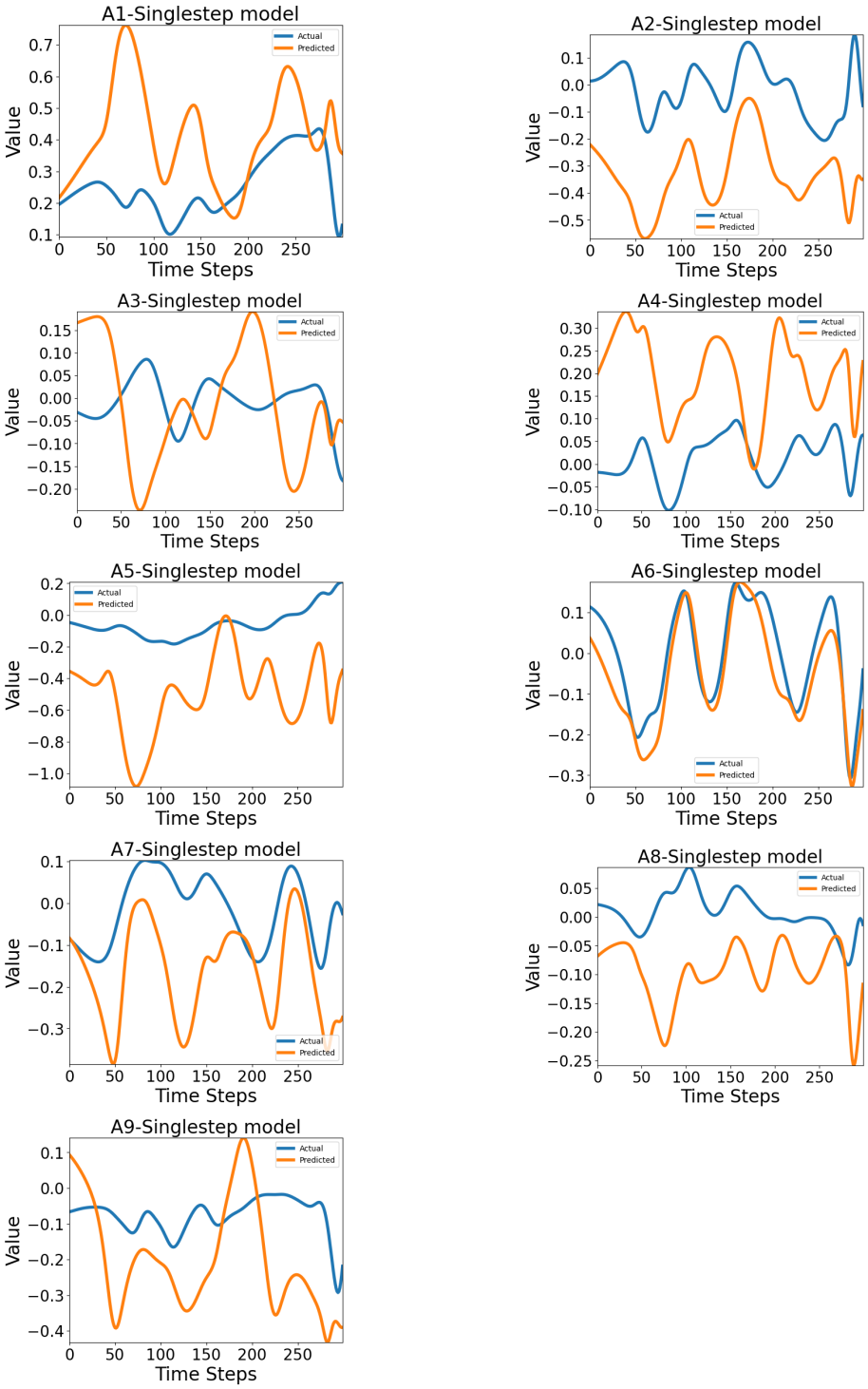


Figure 9: Single time-step predictions for for 1 Timeseries Data each of 9 coefficients

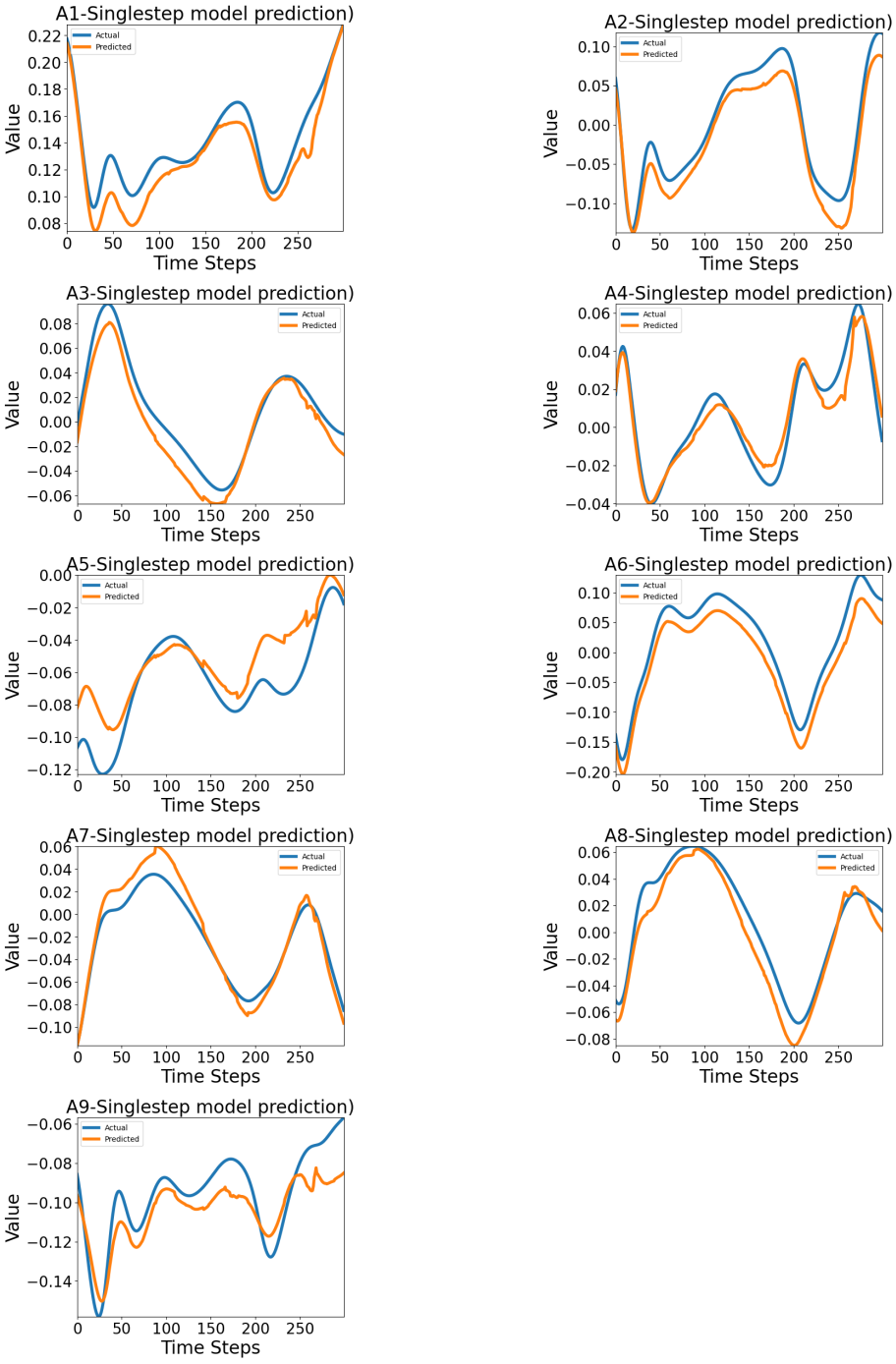


Figure 10: Single time-step ensemble predictions for 100 Timeseries Data for each of 9 coefficients

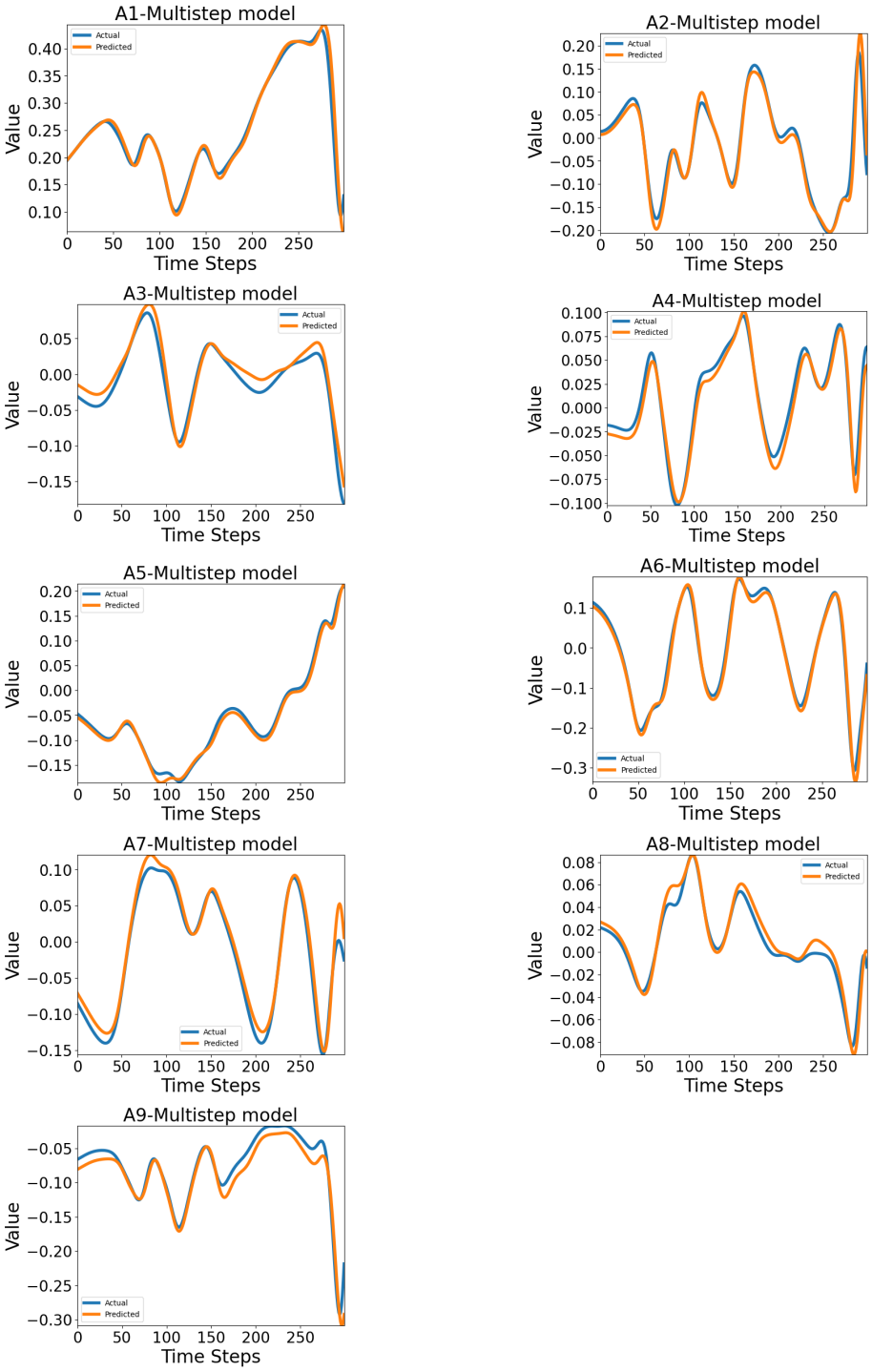


Figure 11: Multi time-step predictions for 1 Timeseries Data for each of 9 coefficients

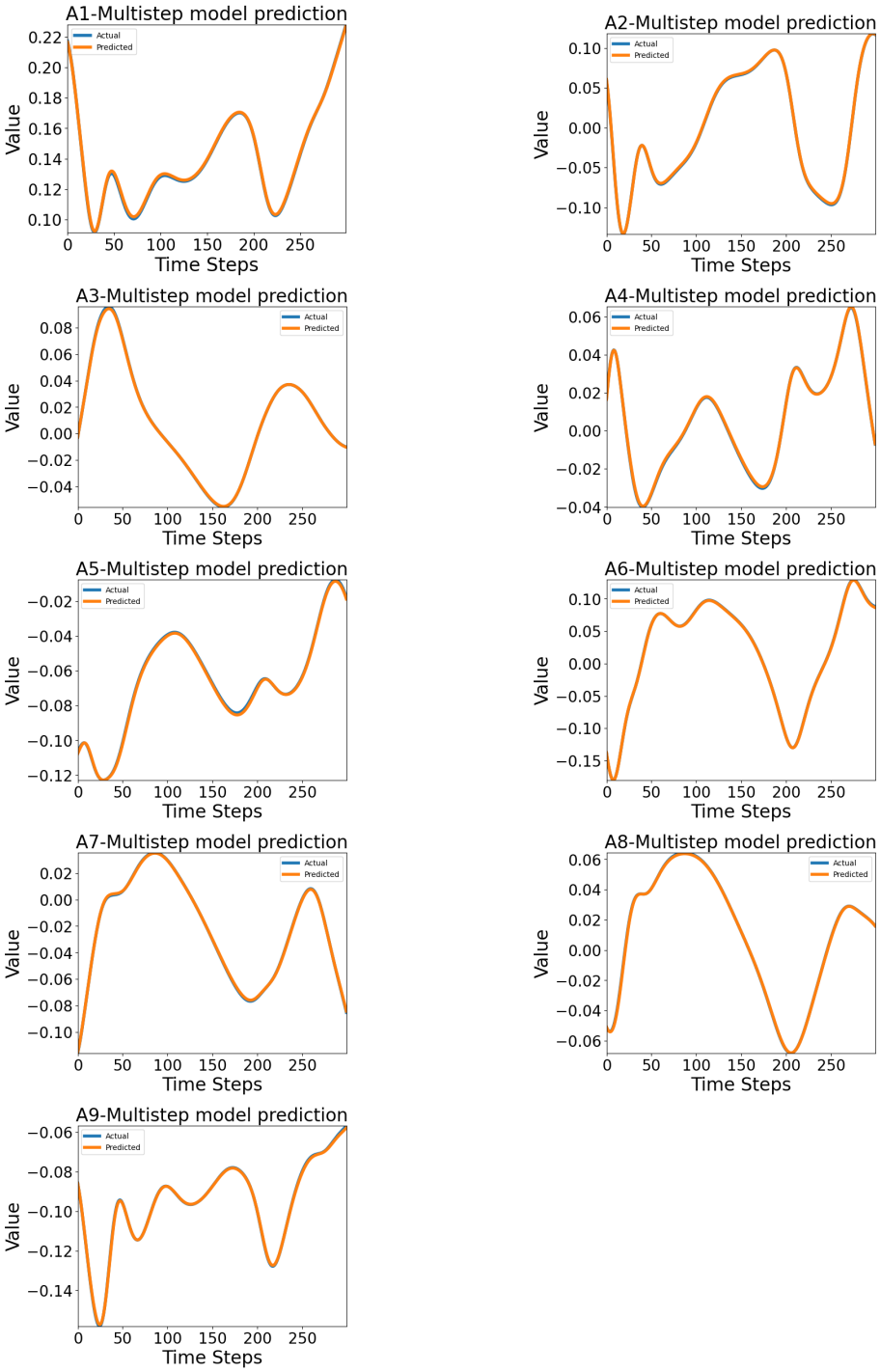


Figure 12: Multi time-step ensemble predictions for 100 Timeseries Data for each of 9 coefficients

This analytical nuance underscores the distinctive advantage of Multi Time Step Predictions, as they possess the unique capability to unveil and model intricate temporal dependencies and correlations concealed within the time series data. Consequently, Multi Time Step Predictions emerge as an indispensable tool for forecasting scenarios involving complex, dynamic, and interconnected data phenomena, making them highly pertinent and valuable in the realm of forecasting and prediction.

7 Predictions with Gated Recurrent units

The architectural transition from LSTM (Long Short-Term Memory) to GRU (Gated Recurrent Unit) in the model represents a strategic choice that balances efficiency and effectiveness in processing sequential data. This model architecture, depicted in Figure 13, comprises three pivotal layers: an input layer, a GRU layer, and an output layer. This choice is motivated by the observation that GRUs offer a similar capability to capture long-range dependencies in sequential data, but with fewer parameters.

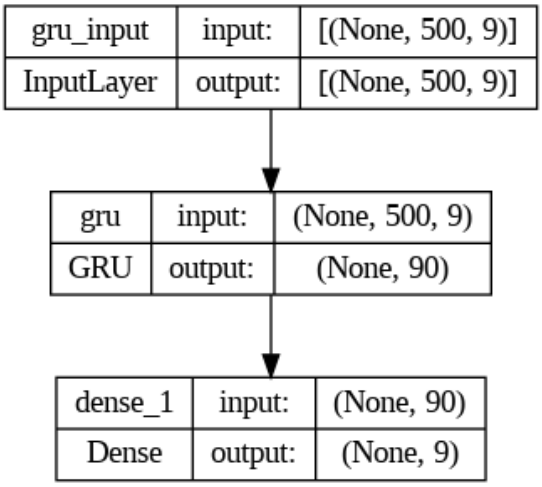


Figure 13: Model Architecture with GRU.

In the *GRU layer*: The input-to-hidden connections have 9 input nodes and 90 GRU units, with each connection having 3 parameters (weight and bias for update and reset gates). The hidden-to-hidden connections have 90 GRU units, and each connection also has 3 parameters.

$$\text{Total parameters in the GRU layer} = 9\times90\times3+90\times90\times3 = 24300 \tag{7}$$

In the *Output layer*: The GRU layer has 90 GRU units connected to 9 output nodes. Each connection has 1 parameter. Additionally, each output node has its bias parameter.

$$\text{Total parameters in the Output layer} = 90 \times 9 + 9 = 819 \quad (8)$$

Therefore, the correct total number of trainable parameters in the model is:

$$\text{Total number of trainable parameters} = 24300 + 819 = 25119 \quad (9)$$

The trained models for multi-step predictions are evaluated using the test data, and the error metrics are obtained: mean squared error (MSE) and mean absolute error (MAE) (Table 2).

Prediction Model	**Time Series Data**	**Number of Epochs**	**LSTM MSE (%)**	**LSTM MAE (%)**	**GRU MSE (%)**	**GRU MAE (%)**
Multi time-step model	1	6	0.01400	0.83300	0.00448	0.47268
Multi time-step ensemble model	50	6	0.00039	0.15376	0.00015	0.08359
Multi time-step ensemble model	100	6	0.00015	0.09367	0.00006	0.05590

Table 2: Performance Metrics for LSTM and GRU Prediction Models

By opting for GRUs over LSTMs, the model can strike a favorable balance between computational efficiency and predictive power. Fewer parameters lead to a smaller memory footprint, reduced computational cost during training and inference, and potentially faster convergence. This is particularly valuable in scenarios where computational resources are constrained or real-time predictions are required.

As shown in Figure 14, the model's predictions closely follow the actual ground truth values for several time steps into the future.It's important to note that while GRUs offer advantages in terms of parameter efficiency, the choice between LSTM and GRU should also consider the specific characteristics of the dataset and the complexity of the task at hand. In some cases, LSTMs may still outperform GRUs when dealing with exceptionally long sequences or when the problem demands a more complex memory structure.

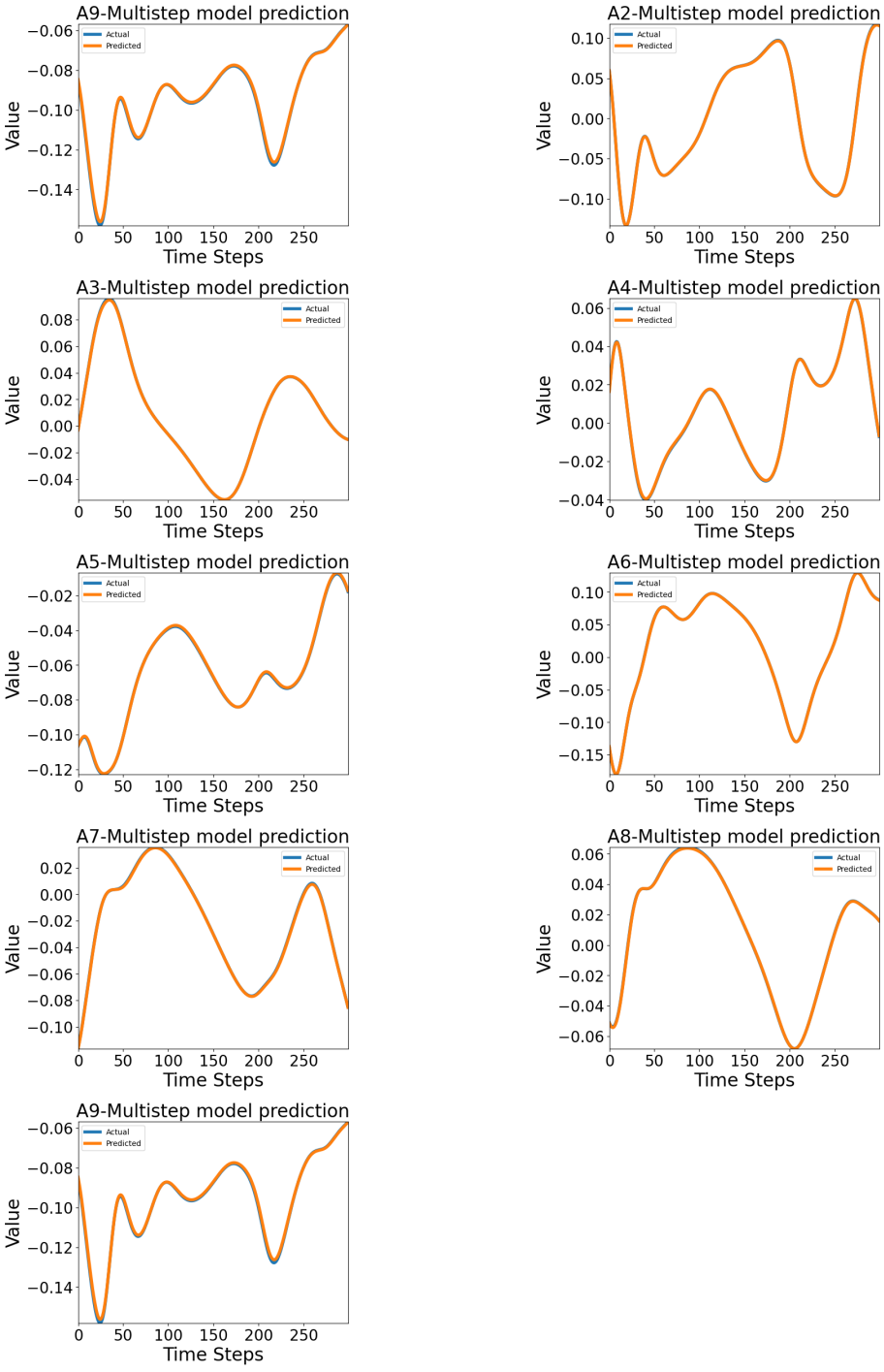
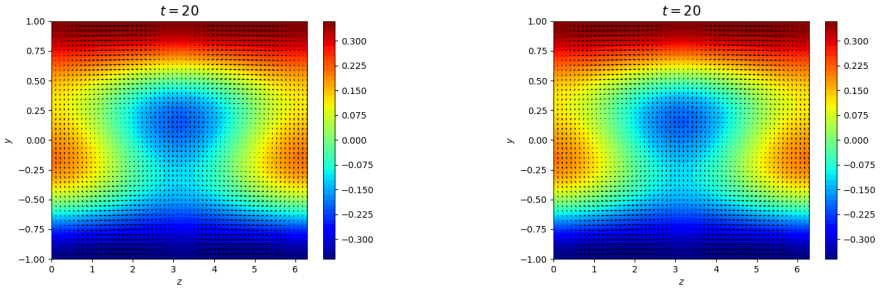


Figure 14: Multi time-step predictions for 100 Timeseries Data for each of 9 coefficients with GRU



(a) In-plane velocity field at time step $t = 20$ (LSTM).

(b) In-plane velocity field at time step $t = 20$ (GRU).

Figure 15: 2D Velocity Profile at time step $t = 20$ for LSTM and GRU models.

In comparing the velocity profiles reconstructed from the LSTM and GRU models for predicting turbulent shear flows, it's striking to note that they exhibit a remarkable degree of similarity. Both profiles share fundamental characteristics, such as capturing the overall flow patterns, major velocity fluctuations, and key features of the turbulent flow. This striking similarity underscores the effectiveness of both LSTM and GRU architectures in this particular forecasting task.

While there are nuanced differences, such as the level of smoothness and response time to changes in the input data, these distinctions are often subtle and may not significantly impact the overall quality of the predictions. In practice, this near-identical performance suggests that, for many applications in turbulent flow prediction, the choice between LSTM and GRU may be driven more by computational efficiency and model preference than substantial differences in prediction accuracy. Researchers and practitioners can leverage this insight to select the architecture that best aligns with their specific computational resources and operational requirements.

8 Conclusion

In this study, we have explored the application of recurrent neural networks (RNNs), specifically long short-term memory (LSTM) networks, in predicting turbulent shear flows. Our findings underscore the significant potential of these deep learning architectures in capturing and forecasting the intricate dynamics of turbulent phenomena. LSTM networks, with their memory-retaining capabilities, have proven to be adept at modeling the complex, time-evolving behavior of turbulent shear flows. The ability to discern long-range dependencies in the data has enabled LSTM networks to outperform traditional methods in point prediction tasks. This research contributes to the body of evidence supporting the efficacy of deep learning in fluid mechanics and engineering applications.

Moreover, the introduction of ensemble predictions has emerged as a game-changer

in turbulent flow forecasting. Ensemble techniques, which amalgamate predictions from multiple models, have demonstrated remarkable improvements in accuracy compared to individual models. This underscores the importance of harnessing the collective intelligence of diverse models to enhance the reliability of predictions in turbulent shear flows.

Furthermore, our investigation has paved the way for future research endeavors. The incorporation of advanced neural network architectures, such as Gated Recurrent Units (GRUs), presents exciting opportunities for further improving predictive performance while reducing computational complexity.

9 Future Scope

The research on Multi Time Step Predictions presents several exciting avenues for future exploration. One promising avenue is to improve the predictive accuracy by integrating advanced recurrent neural networks (RNNs) like Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) variations. The variations of LSTM and GRU, such as bidirectional, stacked, or attention-augmented models , provide additional flexibility and power for handling complex sequences (**AttentionSeq2Seq**)[14]. By selecting the appropriate architecture and fine-tuning hyperparameters, you can tailor these RNNs to your specific problem, achieving state-of-the-art results in various domains. These sophisticated RNN architectures can potentially make predictions more accurate and reliable. Extending the research to include multimodal data sources can provide valuable insights (**MITLecture5**)[13]. Combining data from various sources and disciplines can lead to a more comprehensive understanding of the phenomena being predicted. Explore data augmentation techniques (**DMQA**)[12], such as introducing noise, perturbations, or transformations to the training dataset, to enhance the model's ability to generalize and handle variations in real-world turbulence data. Exploring the use of Generative Adversarial Networks (GANs) for generating time series data is an interesting area (**TimeGAN2019**)[11]. GANs can potentially help create synthetic data that can be used for training and testing prediction models. Leveraging GANs (**DGGAN2023**)[10]for time series data generation opens exciting possibilities for improving the data-driven modeling process, from enhancing dataset quality to advancing predictive model performance and stress-testing. This area of research promises to make a significant contribution to the field of predictive analytics and decision-making.

References

[1] Analytics Vidhya. (2022). Tutorial on RNN, LSTM & GRU with Implementation. *Analytics Vidhya*. Retrieved from https://www.analyticsvidhya.com/blog/2022/01/tutorial-on-rnn-lstm-gru-with-implementation/

[2] Pluralsight. LSTM versus GRU Units in RNN. *Pluralsight*. Retrieved from https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn

[3] Eivazi and Guastoni, Luca and Schlatter, Philipp and Azizpour, Hossein. (2021). Recurrent neural networks and Koopman-based frameworks for temporal predictions in a low-order model of turbulence. *International Journal of Heat and Fluid Flow*, 90, 14. doi: 10.1016/j.ijheatfluidflow.2021.108816

[4] P. A. Srinivasan and L. Guastoni, H. Azizpour and P. Schlatter and R. Vinuesa1. (2019). Predictions of turbulent shear flows using deep neural networks. *PHYSICAL REVIEW FLUIDS*, 4, 15. doi: 2469-990X/2019/4(5)/054603(15)

[5] Jeff Moehlis, Holger Faisst, and Bruno Eckhardt, "A low-dimensional model for turbulent shear flows," New Journal of Physics, vol. 6, p. 56, January 2004, doi: 10.1088/1367-2630/6/1/056.

[6] Colah's Blog. (2015). Understanding LSTM Networks. *colah's blog*. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] Towards Data Science. (2021). Tutorial on LSTM: A Computational Perspective. *towards data science*. Retrieved from https://towardsdatascience.com/tutorial-on-lstm-a-computational-perspective-f3417442c2cd

[8] TensorFlow: An open-source deep learning framework developed by Google. It provides a flexible ecosystem of tools, libraries, and resources for building and deploying machine learning models. Available at: https://www.tensorflow.org/

[9] An Introduction to RNN and LSTM [Video file]. (2020, Oct 8). Retrieved from YouTube: https://youtu.be/Mdp5pAKNNW4

[10] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, Vyas Sekar. *Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions*. *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3419394.3423643

[11] Jinsung Yoon, Daniel Jarrett, Mihaela van der Schaar. *Time-series Generative Adversarial Networks*. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

[12] [Open DMQA Seminar] Augmentation of Time Series Data. *YouTube Video*. Available at: https://youtu.be/OmGjcOo1eN0?si=53WIi4hZxDCbh0VU.

[13] MIT 6.S191 Lecture 5 Multimodal Deep Learning. *YouTube Video*. Available at: https://youtu.be/6QewMQT4iMM?si=i2A_QucJsNY7zsGf.

[14] Attention for RNN Seq2Seq Models. *YouTube Video*. Available at: https://youtu.be/B3uws4cLcFw?si=N1PxUN8mga8Paqsj.

End of the Report