Flow Field Reconstruction using Optimal Sensor Placement and Deep Learning

Bhavneet Bali, Mandar Tendolkar

Veermata Jijabai Technological Institute, Mumbai, India

Corresponding author: Bhavneet Bali, Email: brbali_b19@me.vjti.ac.in

Fluid flow reconstruction is a crucial task in engineering, environmental science, and fluid dynamics. Accurately predicting and understanding fluid flow patterns is essential for optimizing processes, designing efficient systems, and mitigating potential risks. Traditional methods often rely on complex mathematical models and computationally intensive simulations. However, recent advancements in deep learning and neural networks have shown promising results in tackling this problem. The present work proposes a novel approach to fluid flow reconstruction using neural networks, aiming to develop an efficient and accurate feed-forward neural network model capable of predicting fluid flow behaviour based on available data. An artificial neural network (ANN) is used to capture spatial dependencies in fluid flow data, learning to infer underlying flow dynamics. The model's robustness and generalization capabilities are ensured by carefully curating the dataset and incorporating appropriate data augmentation techniques. The results demonstrate the effectiveness of neural networks in fluid flow reconstruction, with significant improvements in prediction accuracy and efficiency compared to traditional methods. The ability to reconstruct fluid flow patterns accurately from limited or incomplete data has the potential to revolutionize various industries, enabling more informed decision-making, optimizing processes, and improving safety measures. This work contributes to the growing body of knowledge in deep learning for fluid dynamics and offers a promising avenue for further advancements in predicting and understanding fluid flow behaviour.

Keywords: Artificial Neural Network (ANN), Fluid flow reconstruction, Proper Orthogonal Decomposition (POD)

2023. In Satyasai Jagannath Nanda & Rajendra Prasad Yadav (eds.), *Data Science and Intelligent Computing Techniques*, 499–508. Computing & Intelligent Systems, SCRS, India. https://doi.org/10.56155/978-81-955020-2-8-46

1 Introduction

Fluid flow reconstruction is the process of predicting or estimating the behaviour of fluid flow, often from limited or incomplete data. It has numerous applications in areas such as aerospace engineering, and mechanical engineering. It involves training a neural network to learn the relationship between the data and then using the network to make predictions for new data. Fluid flow reconstruction techniques can vary depending on the type of flow being analyzed, the available measurement data and the desired level of accuracy. Some common techniques include using mathematical models such as Navier-Stokes equations, finite element or finite difference methods, and machine learning algorithms. The accuracy and reliability of the reconstructed flow field can depend on the quality and quantity of measurement data available as well as the chosen reconstruction technique.

An artificial neural network (ANN), also known as a neural network, is a computational model capable of processing information to tackle tasks such as classification and regression. It is the component of artificial intelligence inspired by the human brain and nervous system. Artificial neurons and the coding signals in an ANN are used to simulate the electrical activities of how the nervous system communicates. Each artificial neural neuron receives input signals, processes these signals, and generates an output signal. A general practice is to sum the weighted inputs, and then feed the sum value into a transfer function to judge if the value is qualified to be transferred to another neuron as an input. The simplest transfer function is to compare the sum value with the set threshold, while nonlinear functions allow more flexibility.

The prime objective of the present work is to develop a feed-forward neuralnetwork to learn the relationship between sensor measurements and the reduced-order state of the system. The work further aims at data-driven optimization of sensor placement to achieve accurate fluid flow reconstruction.

2 Literature Review

2.1 Surrogate Modelling

Surrogate modelling, also known as metamodeling or response surface modelling, is a machine-learning technique used to approximate and represent complex mathematical or computational models. It involves creating a simpler, computationally efficient model, called a surrogate model, that acts as a substitute for the original model. This approach is used for tasks like optimization, sensitivity analysis, and uncertainty quantification, providing similar outputs but with lower computational costs. The surrogate model is constructed using projection-based reduced-order modelling, where the high-dimensional system is compressed to a low-dimensional system and the dynamics of the low-dimensional system are modeled[5].

2.2 Physics-Guided Machine Learning

Physics Guided Machine Learning (PGML) is a framework that combines the strengths of physics-based models and machine learning to improve modelgeneralizability and interpretability. It embeds simplified physics-based models into neural network architecture, allowing the network to adhere to realistic constraints. PGML is flexible and applicable to various physical systems, incorporating physics during training and prediction stages. Despite its limitations, PGML addresseschallenges in capturing complex dependencies from data and addressing limitedscientific data and a lack of understanding of underlying physical processes. Byinjecting physics-based features into neural network layers, PGML provides amodular and generic approach to embed simplified theories or physics-based kernels. Overall, PGML has the potential to enhance the performance, interpretability, and generalizability of machine learning models in scientific domains, bridging the gap between data-driven approaches and physical understanding[1, 2].

2.3 Reduced Order Modelling

Reduced Order Modelling (ROM) focuses on precision in simulations while reducing computational effort. It offers accurate and realistic inter-mediate models that bridge the gap between simulations and experiments. ROM uses compression techniques, machine learning, and image processing to efficiently utilize existing models and experimental data. It enables real-time modelling, facilitates re-design, and overcomes challenges related to licensing, multi-scale considerations, and confidence inindustrial settings. Non-intrusive techniques, such as machine learning methods like LSTM and statistical approaches like GP modelling, provide effective solutions for optimal sensor placement in systems with unknown or insufficient equations, enabling accurate predictions and efficient data collection[2, 9].

2.4 Deep Learning

Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) are deep learning models used for complex pattern recognition and prediction tasks. ANN is inspired by the human brain's function and consists of input, hidden, and output layers. CNN, on the other hand, is designed for image-related tasks and mimics the connectivity pattern of the human brain's visual processing. Both models requirecareful hyperparameter selection and training procedures to prevent overfitting and ensure generalization. Techniques like regularization, dropout, ensembling, and early stopping can be employed to find optimal hyperparameters.CNN architectures for Kraichnan turbulence involve multiple hidden layers, filters, and activation functions. Post-processing is applied to the CNN's predicted source term for numerical stability before using it in the vorticity transport equation.

2.5 Data Set

We use the NOAA OI SST V2 analysis dataset for building our surrogate model. The analysis uses in situ (ship and buoy) and satellite SSTs plus SSTs simulated by the sea-ice cover. Before computing the analysis, the satellite data is adjusted for biases using the method of Reynolds and Reynolds and Marsico. This dataset consists of the weekly average sea surface temperature on a 1° latitude $\times$ 1° longitude global grid (180 $\times$ 360). The SST dataset exhibits a strong periodic structure due to seasonalfluctuations. Despite this seasonal periodicity, complex ocean dynamics lead to rich flow physics in this dataset. This dataset has been used in several recent studies on flow reconstruction, geophysical emulation, and dynamic mode decomposition. Here, we use the data from October 1981 to December 2000 (1000 snapshots) for building a surrogate model and the data from January 2001 to June 2018 (914 snapshots) for comparing the performance of the surrogate model for forecasting.

3 Methodology

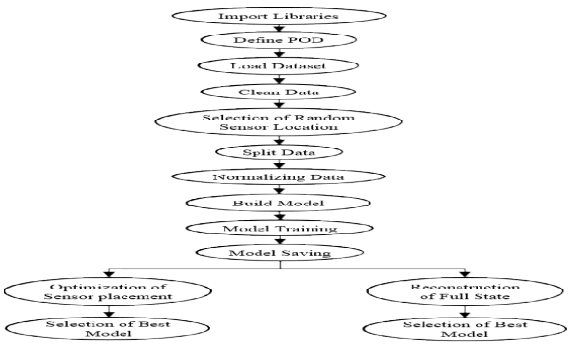

Figure 1. Algorithm

3.1 Data Preparation and POD Functions

POD is used for reduced-order modelling in dynamical systems, as it captures spatial and temporal variations of flow fields. PCA, on the other hand, is primarily used for stationary data and cannot capture time-dependent behaviour. POD outperforms PCA in fluid mechanics applications, such as transitional flow analysis, as demonstrated by [4].

After initializing some input parameters, the dataset of sea-surface temperature (SST) is loaded which is stored in an h5py file, and preprocessing of the dataset is done by reshaping it into a 3D array. Specifically, it reshapes the SST data from shape [number of samples, 180*360] to [number of samples, 180, 360] by first flipping the data along the vertical axis using 'np.flipud()' and then reshaping it using, 'reshape()' method. All the data points corresponding to the land area (NaN values) are removed and only keep the sea surface data points. This is done for all the samples in the dataset. The resulting data is stored in the variable `sst_masked`, which has the shape '[number_of_points, number_of_samples]'. Then the average sea surfacetemperatures and fluctuation of the sea surface temperatures are computed for a certain number of snapshots after which, the POD basis functions are computed using the fluctuation data. The number of basis functions retained is set. Relative Information Content (RIC) is a measure used to assess the amount of information carried by a particular feature or variable in a dataset, relative to the total amount of information in the dataset. First, the POD Modal Coefficients and POD BasisFunctions are calculated and split into training and test sets. A variable is defined that is of the shape (me,). This variable has to be initialized for every value of 'me'. The random.sample() function takes care of randomness in picking up the sensor locations. To make sure we can reproduce the same results we set the random.seed(10). Now that random sensors have been chosen, the sst_observations are split into training and test sets with the help of sst_fluc and ns_train_rom. We have 1000 samples for training. The input has features equal to the me value and the output has 25 values to predict.

The dataset has to be scaled and shuffled before feeding it to the neural network. Scaling the data before training a neural network is important to ensure that the model converges faster and more reliably, has better performance, is numerically stable, and is less prone to overfitting and shuffling the dataset before training a neural network helps to ensure that model is not biased towards specific samples or features, and helps to improve its ability to generalize to new data.

3.2 Model Training and Evaluation

A function called 'build_model' is defined which constructs a neural network model using the Keras library. The function begins by creating an empty sequential model using the models.Sequential() method from Keras. This model will serve as the base for building the neural network.Next, a dense layer with 40 neurons is added to the model, using the activationfunction specified during initialization. The input shape is assumed to be a 2D array where the first dimension represents the number of samples (or instances) and the second dimension represents the number of features (or inputs).

Next, a for loop is used to add the number of dense layers to the model, each with thenumber of neurons and the activation function specified during initialization. The loop also adds a final dense layer with the number of neurons equal to the number ofoutput classes.

After the loop is complete, the model.compile() method is called to compile the model for training. The optimizer argument specifies the optimization algorithm to be used during training, which in this case is Adam. The loss argument specifies the lossfunction to be used during training, which in this case is Mean Squared Error (MSE). The metrics argument specifies additional metrics to be evaluated during training, which in this case is Mean Absolute Error (MAE). Finally, the function returns the constructed neural network model which is trained for 500 epochs with a batch size of 16 and evaluates the model's performance on a validation set which is 25% of the training set. After training,

the validation Mean Absolute Error (MAE) is recorded and saved. This task is done for different values of 'me' and corresponding results are stored in a folder for later analysis.

3.3 Optimization of Sensor Placement

The optimized split is achieved by performing QR decomposition[7] of the matrix using the qr() function from the linalg module of the scipy library. The specific parameters used are mode='economic' and pivoting=True. mode='economic' specifies that the function should return only the essential parts of the QR decomposition, i.e., the Q and R matrices, rather than the full matrices. This can be more memory-efficient for large matrices. pivoting=True specifies that the function should use pivoting to improve the numerical stability of the decomposition. Pivoting means that the function reorders the rows of the matrix so that the largest diagonal element of R is first, then the next largest, and so on. This helps to minimize round-off errors and improve the accuracy of the decomposition.

The output of the function is a tuple (Q, R, pivot): Q is the orthogonal matrix in the QR decomposition and has orthonormal columns. R is the upper triangular matrix in the QR decomposition. Pivot is a vector of row indices that gives the permutation of the row that was used in the pivoting process. This can be used to reconstruct the original order of the rows if needed. First 'me' entries of the pivot are extracted and stored in a variable. These entries represent the row indices that were selected during pivoting, and they correspond to the most important 'me' columns of the matrix.

3.4 Reconstruction of Full State of System

The reconstructed Sea Surface Temperature (SST) data is obtained using the Proper Orthogonal Decomposition (POD) method. The variable `sst_masked` contains the original SST data with some missing values that were masked out. The reconstructed data is stored in 'sst_rec_test' as a two-dimensional array with dimensions `(not_nan_array.shape[0], at_test.shape[0])`. The missing values are filled with NaNs.

The variable `PHIw` contains the POD basis functions that were computed from the training data using the `POD` function. The `PODrec` function is used to compute the reconstructed SST data using the test data and the POD basis functions. The variable `sst_avg` contains the average SST computed from the training data.

The ground truth data collected from satellite observations is stored in the `ufom` variable. This data is used to compute the forecasting error analysis. The reconstructed SST data is reshaped into a three-dimensional array with dimensions `(sst_rec_test.shape[1], lat.shape[1], lon.shape[1])` using a loop over each time step `k` in `sst_rec_test`. Finally, the resulting three-dimensional array is stored in the `utest` variable.

To reconstruct predicted fluid flow fields using a POD-based model trained using deep learning first, the predicted coefficients of the POD basis functions are transformed back to the original scale of the data by multiplying by the standard deviation and adding the mean value. Then, the reconstructed fluid flow fields are obtained by applying the inverse POD transformation to the predicted coefficients. This is done using the `PODrec` function with the predicted coefficients `at_pred` and the POD basis functions `PHEw`.Finally, the reconstructed fluid flow fields are reshaped into a 3D array with dimensions corresponding to the latitude, longitude, and time axes. The reshaping is done using a for loop to iterate over the time dimension and reshape each time step using the `reshape` function. The resulting 3D array is stored in the variable `upred`.

4 Results and Discussions

4.1 Getting The Best Model For me = 10

Random Sensor Placement Starting with Random Sensor Placement (Figure 2), the Mean Absolute Error (MAE) is plotted against Epochs for two activation functions, namely 'Relu' and 'Tanh'. The minimum MAE value of 0.657 is observed for the model with 4 Layers and the 'Tanh' activation function as hyperparameters.

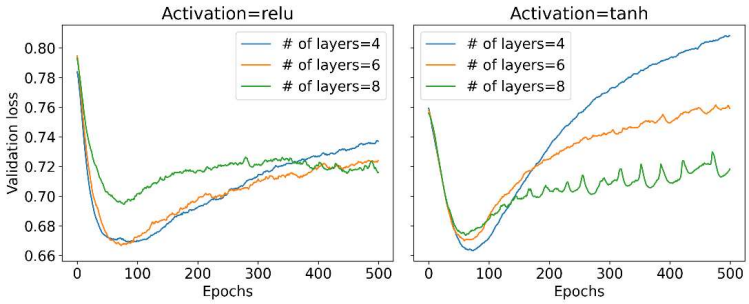

Figure 2. MAE comparison for Random Sensor Placement (me = 10)

Optimal Sensor Placement For Optimal Sensor Placement (Figure 3), the same MAE vs. Epochs plot is plotted. It is observed that a minimum MAE of 0.499, occurs for the model with 6 Layers and the 'Relu' activation function as hyperparameters. An improvement is observed after the optimization of the sensor placements.

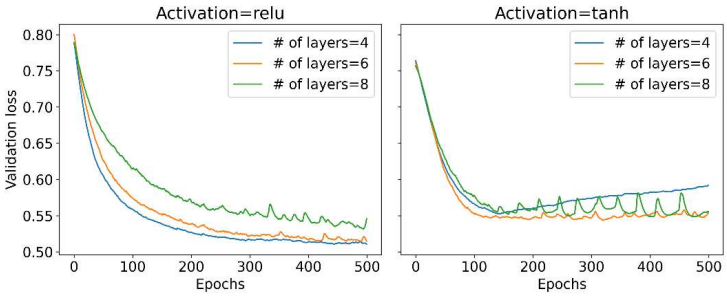

Figure 3. MAE comparison for Optimal Sensor Placement (me = 10)

4.2 Reconstruction of the Full State of the System

After obtaining the best parameters for each model, it is better to visualize the bigger picture with all the models compared side by side. Two important differences are visible by looking at Figure 4. It is observed that as the number of sensor observations increases, the MAE value starts to come down significantly in the case of Random Sensor Placement, but the effect of change in the number of sensor observations is not as significant in the case of Optimal Sensor Placement.

Also, optimisation of sensor placement leads to reaching a lower MAE faster ascompared to Random Sensor Placement. Furthermore, when 25 POD ModalCoefficients are chosen to retain, more than 96% of the information is preserved. A brief comparison is given below in Figure 5. The data is projected

onto thepre-calculated POD Basis Function to produce a reconstruction after the inputparameters are established. It is plotted next to the True Data representation for better comparison

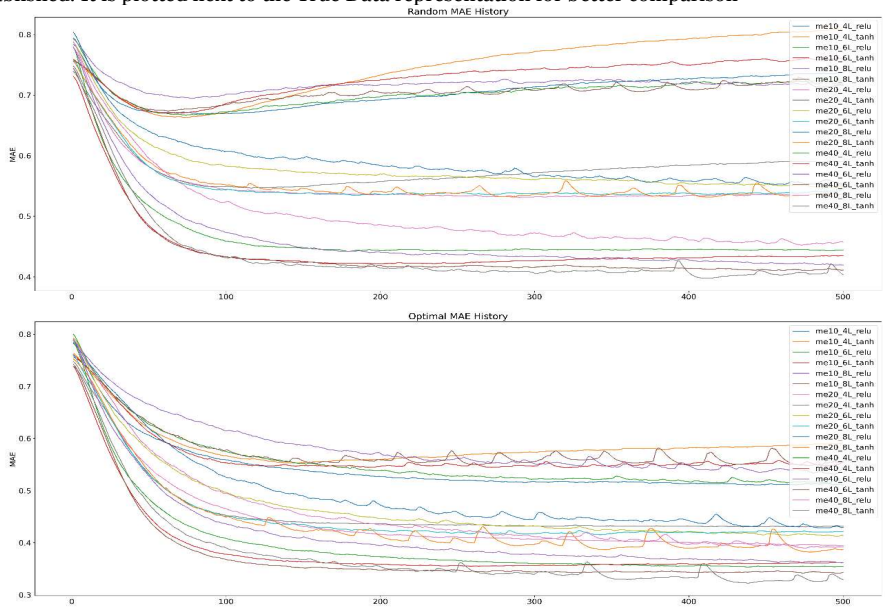

Figure 4. MAE Comparison of all the models

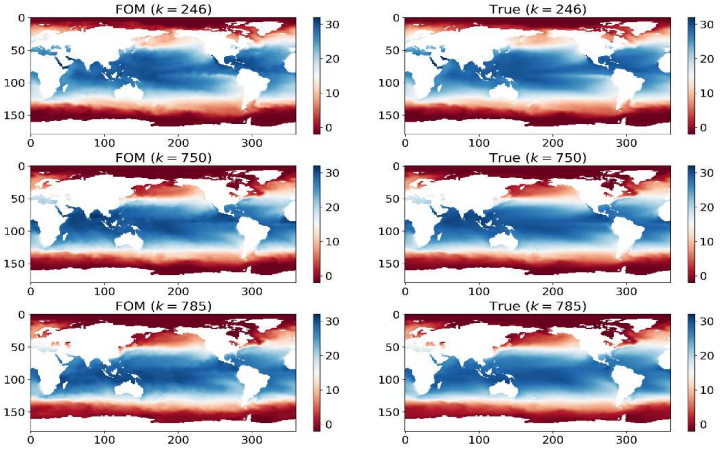

Figure 5. Ground truth data collected from satellite observations (FOM) alongside True projection of data onto pre-computed POD Basis Function (True)

4.3 Reconstruction of the Full State of the System Using Predicted Modal Coefficients

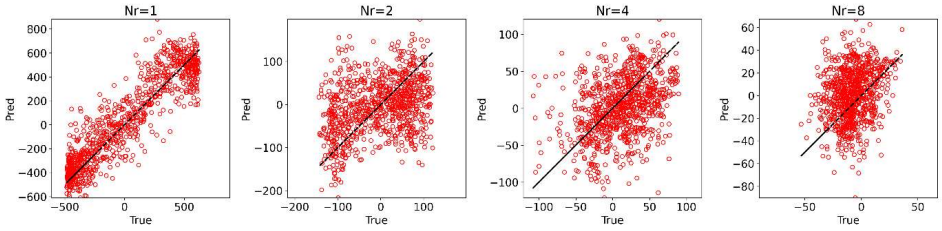

Figure 6. Correlation plot between true and predicted coefficients for Random Sensor Placement (me = 10)

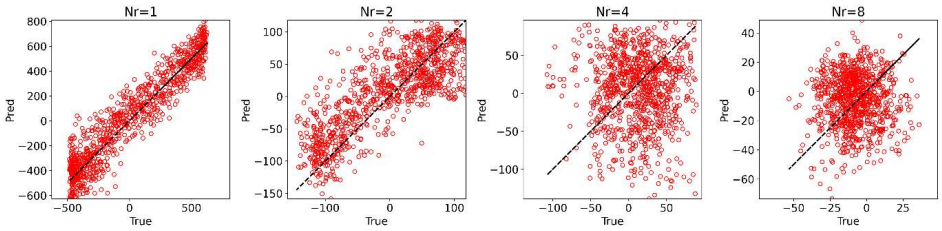

Figure7. Correlation plot between true and predicted coefficients for Optimal Sensor Placement (me = 10)

A correlation plot is plotted to examine relationships between preserved modalcoefficients (Figure 6 and Figure 7). The correlation declines for Random Sensor Placement after the first modal coefficient but remains somewhat constant for Optimal Sensor Placement up to the second modal coefficient. There is little substantial association between the two. The probability density function for predicted values shows a similar shape (Figure 8), but the improvement achieved by optimizing lowers the error range in Optimal Sensor Placement.

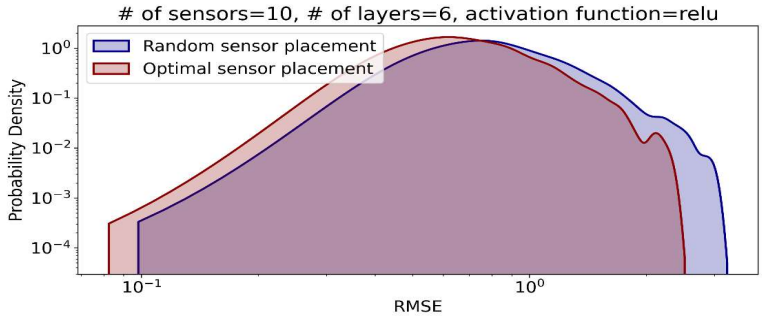

Figure8. Error comparison between Random and optimal Sensor Placement (me = 10)

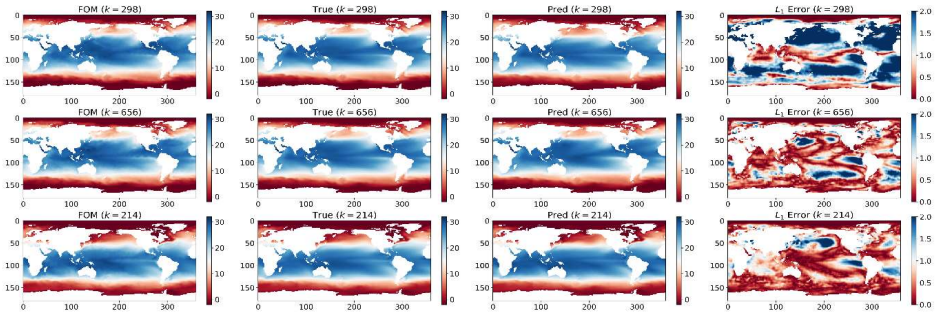

Figure 9. Reconstruction for Random Sensor Placement and me = 10

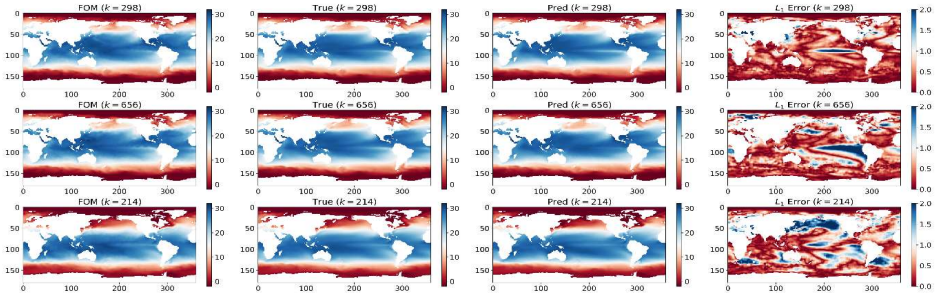

Figure 10. Reconstruction for Optimal Sensor Placement and me = 10

Concluding Remarks: Optimizing sensor placement significantly reduces errors and accelerates MAE convergence to low values. Random sensor placement increases correlation to higher modes, while optimal placement results in a weaker correlation to lower modes. However, optimizing sensor placement may not be worth the time and computational power, as Random sensor placement is better in some cases and the improvement observed after optimizing sensor locations is not significant enough to be considered worth the input (Figure 9 and Figure 10).The QR pivoting algorithm demonstrated higher efficiency for optimal sensor placement.

5 Conclusion

The work aims to develop a neural network to learn sensor measurements and reduced-order state relationships, enabling data-driven optimization of sensor placement for accurate fluid flow reconstruction. Proper Orthogonal Decomposition (POD) and Deep Neural Networks are employed to achieve maximum Relative Information Content (RIC) with 40 sensor observations.

Deep Neural network models for achieving a minimum value of mean absolute error (mae) for random sensor placements at different numbers of sensor observations (me) are:

1. 4 layers 40 neurons with activation function ‘tanh’ for me = 10.
2. 8 layers 40 neurons with activation function ‘tanh’ for me = 20.
3. 8 layers 40 neurons with activation function ‘tanh’ for me = 40.

Deep Neural network models for achieving a minimum value of mean absolute error (mae) for optimal sensor placements at different numbers of sensor observations (me) are:

1. 6 layers 40 neurons with activation function 'relu' for me = 10.
2. 8 layers 40 neurons with activation function 'tanh' for me = 20.
3. 8 layers 40 neurons with activation function 'tanh' for me = 40.

6 Acknowledgement

The authors would like to acknowledge Dr. Suraj Pawar, ex-research scholar, at Oklahoma State University, USA for his immense help at every stage of this work.

References

[1] Pawar, S., San, O., Aksoylu, B., Rasheed, A. and Kvamsdal, T., 2021. Physics-guided machine learning using simplified theories. Physics of Fluids, 33(1).

[2] Suraj Pawar, Physics-guided machine learning for turbulence closure and reduced-order modelling, PhD Dissertation, Oklahoma State University, USA, 2022

[3] Pawar, S. and San, O., 2022. Equation□Free Surrogate Modeling of Geophysical Flows at the Intersection of Machine Learning and Data Assimilation. Journal of Advances in Modeling Earth Systems, 14(11), p.e2022MS003170.

[4] Rowley, C.W., 2005. Model reduction for fluids, using balanced proper orthogonal decomposition. International Journal of Bifurcation and Chaos, 15(03), pp.997-1013.

[5] Alexander I. J. Forrester, András Sóbester, Andy J. Keane, Engineering Design viaSurrogate Modelling: A Practical Guide, 2008.

[6] Callaham, Jared L., Kazuki Maeda, and Steven L. Brunton. "Robust flow reconstruction from limited measurements via sparse representation." Physical Review Fluids 4, no. 10 (2019): 103907.

[7] Krause, A., Singh, A. and Guestrin, C., 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, 9(2).

[8] Berkooz, G., Holmes, P. and Lumley, J.L., 1993. The proper orthogonal decomposition in the analysis of turbulent flows. Annual review of fluid mechanics, 25(1), pp.539-575.

[9] Yu, J., Yan, C. and Guo, M., 2019. Non-intrusive reduced-order modelling for fluid problems: A brief review. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 233(16), pp.5896-5912.