AS4545: Undergraduate Research
Project Report

Deep Learning based RANS method for turbulence modelling.

Author: Rahul Chakwate (AE16B005)

Advisor: Dr. Vadlamani Nagabhushana Rao

June 3, 2020

Contents

1 Problem Statement

To develop a deep learning based Reynolds-averaged Navier–Stokes (RANS) turbulence model to predict Reynolds Stress Anisotropy tensor from the given high fidelity simulation data.

2 Introduction and Related Work

Reynolds Averaged Navier Stokes (RANS) models are widely used in industry to predict fluid flow. RANS models rely on turbulent transport instead of fully resolving the turbulent motion. Hence, they are much more computationally efficient as compared to the Direct Numerical Simulations (DNS) which try to resolve different scales of flow. However, RANS models can often be inaccurate in flow predictions because they depend on highly irregular turbulence empirical data. Earlier RANS models rely on the linear eddy viscosity model (LEVM) for their Reynolds stress closure. LEVM assumes linear relation between Reynolds stress and mean strain rate. However, these models provide inaccurate results in many flow cases including those with curvature separation or impingement. Hence, Craft et al.[1] proposed a non linear eddy viscosity model based on different combinations of products of rotation rate tensor and strain rate tensor. These did not show a significant improvement over linear models.

Recently, machine learning techniques are used to for Reynolds stress closure problem. Tracey et al[2] used non-parametric methods (Kernel Regression) to try to fit the error in RANS Reynolds stress predictions. This method exhibited limited generalization capability. In later work, Tracey et al. [2] used a neural net with single hidden layer to try to learn the Spalart-Allmaras turbulence model. Zhang et al.[3] also used neural networks to predict correction factor in turbulence term. These methods were also ineffective in predicting anisotropy tensor. Ling et al. [4] evaluated three different machine learning models: Support Vector Machines, Decision Trees with Adaboost and Random Forests. Random forests gave better results among the three methods. However, Galilean invariance cannot be enforced to predict anisotropy tensor by using the random forests.

Deep learning [5] is a branch of machine learning which uses deep neural networks to model highly complex functions which cannot be modelled by most other machine learning techniques. As the turbulence signal is a heavily complex signal, deep learning methods provide a better alternative for turbulence modelling. Tracey et al.[2] and Zhang et al. [3] used shallow neural networks with one or two hidden layers. Ling et al. [6] used rotation invariant tensors to model anisotropy tensor's eigenvalues which significantly improved the RANS modelling. Moghaddam1 et al. [7] makes use of deep learning algorithms via convolution neural networks along with data from direct numerical simulations to extract the optimal set of features that explain the evolution of turbulent flow statistics.

3 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance.

Motivated by these advancements, Julia ling et al.[8] came up with a tensor basis neural network which is based on invariant tensors.

3.1 Deep Neural Networks

A deep neural network consists of an input layer followed by several hidden layers followed by an output layer (Figure 1a). Each of the layers consists of many nodes. The nodes represent the features of that layer. The input layer will have as many input nodes as the number of features of the input. From one layer to another, the input to the layer l (X_l) is multiplied by the weights (W) of that layer and added with a bias term (b) which is further passed through a non-linear activation function (ϕ) which non-linearly scales the

input. The output of layer l is the input to the next hidden layer *l+1*. Mathematically the operation is as follows.

$$X_{l+1} = \phi(W^T X_l + b) \tag{1}$$

Here Rectufied Linear Unit (ReLU) is used as the non-linear activation function. It is defined as $\phi(x) = max(0, x)$. Its modified form leaky-ReLU is used in the model.

The neural network is trained using back propagation of gradients with gradient descent algorithm. The model is fitted to training data which trains the model to iteratively minimize the mean squared error between the true and predicted anisotropy tensor. A neural network has mainly three hyperparameters: the learning rate of training, number of hidden layers and the number of nodes in each layer. Optimum parameters were decided using the Bayesian hyperparameter optimization.

3.2 Tensor Basis Neural Network

A different form of neural network is used in this method instead of a simple neural network as shown in Figure 1b. Instead of directly formulating the stress anisotropy tensor (b) as a function of R and S, it enforces rotational invariance by formulating b as a combination based on isotropic tensors. Rotational invariance is a fundamental property of a fluid particle and it is necessary that any turbulence closure obeys it.

For the input tensors R and S, Pope [9] has derived relevant integrity basis of input tensors. He proves that an eddy viscosity model that is a function of only S and R can be expressed as a linear combination of 10 isotropic basis tensors:

$$b = \sum_{n=1}^{10} g^{(n)}(\lambda_1, ..., \lambda_5) T^{(n)} \tag{2}$$

Tensor b satisfying equation 2 satisfies Galilean invariance. These 10 tensors $T^{(1)}$,...,$T^{(10)}$ and 5 invariances λ_1,...,λ_5 are described in Pope [9]. The function $g^{(n)}(\lambda_1, ..., \lambda_5)$ is determined by the deep neural network TBNN.

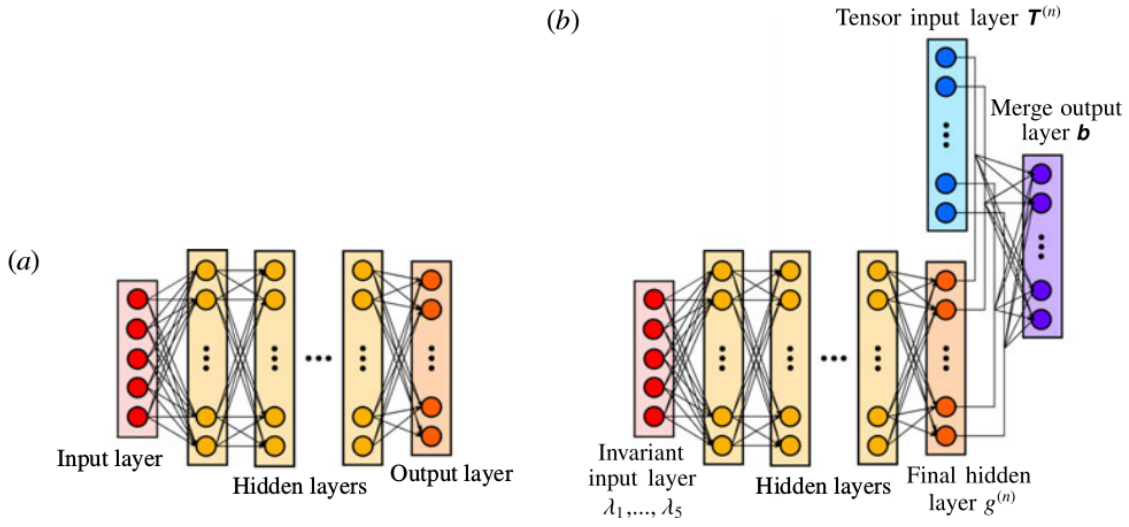

Figure 1: a) Simple Neural Network, b) Tensor Basis Neural Network.
Source: Julia ling et al.[8]

4 Neural Network Settings for the Experiemnts

Through Bayesian optimization, the following parameters were obtained as optimal parameters.

- number of hidden layers = 8,
- nodes per hidden layer = 30,
- Total number of weights = 6750
- activation function = Leaky-ReLU
- learning rate = $2.5 * 10^{-7}$.
- number of input nodes: 5 (λ's)
- number of output nodes: 10 (g's)
- loss function: RMSE Loss

$$RMSE = \sqrt{\frac{1}{9N_{data}} \sum_{n=1}^{N_{data}} \sum_{i=1}^{3} \sum_{j=1}^{3} (b_{ij,m,pred} - b_{ij,m,DNS})^2} \tag{3}$$

The 10 scalar outputs of the neural network are multiplied to corresponding 10 tensors (T). The sum of these 10 values gives the predicted (b) values. These predicted values are compared with true values using mean squared error loss function which is minimized by training the network.

5 Datasets

The following DNS datasets are publicly available for training the model.
Datasets:

- Channel Flow [10]
- Duct Flow [11]
- Flow around a Square Cylinder [12]

Detailed information about each dataset is as follows:

Channel Flow [10]:

Reynolds No.	**# data points**
180	56
395	64
590	176
Total	296

Table 1: Channel Flow Dataset

The dataset consists of DNS data of the flow across a channel at three different Reynolds number. The flow is symmetric across the X and Z axis. All values such as means and Reynolds stresses are same if the y value is same, i.e. same across the XZ plane. Hence, all the data boils down to the data on a single line. This line

Reynolds No.	# data points
300	16900
600	67600
900	150544
1200	268324
Total	503368

Table 2: Duct Flow Dataset

data is provided by the author at flow Reyonolds number of 180, 395 and 590 as shown in Table 1.

Duct Flow [11]

This dataset has DNS information of the fluid flow across a square duct at four different Reynolds number: 300,600,900 and 1200 (Table 2.). The flow is symmetric about the X axis.

Flow around a Square Cylinder [12]

Flow Reynolds No = 22000

Number of data points = 1498224

The dataset contains results from DNS of turbulent flow around a square cylinder at Re=22000. A constant velocity is imposed in the X direction. The flow is symmetric about the X-axis. A still from the dataset recording is given in Figure 2. below.

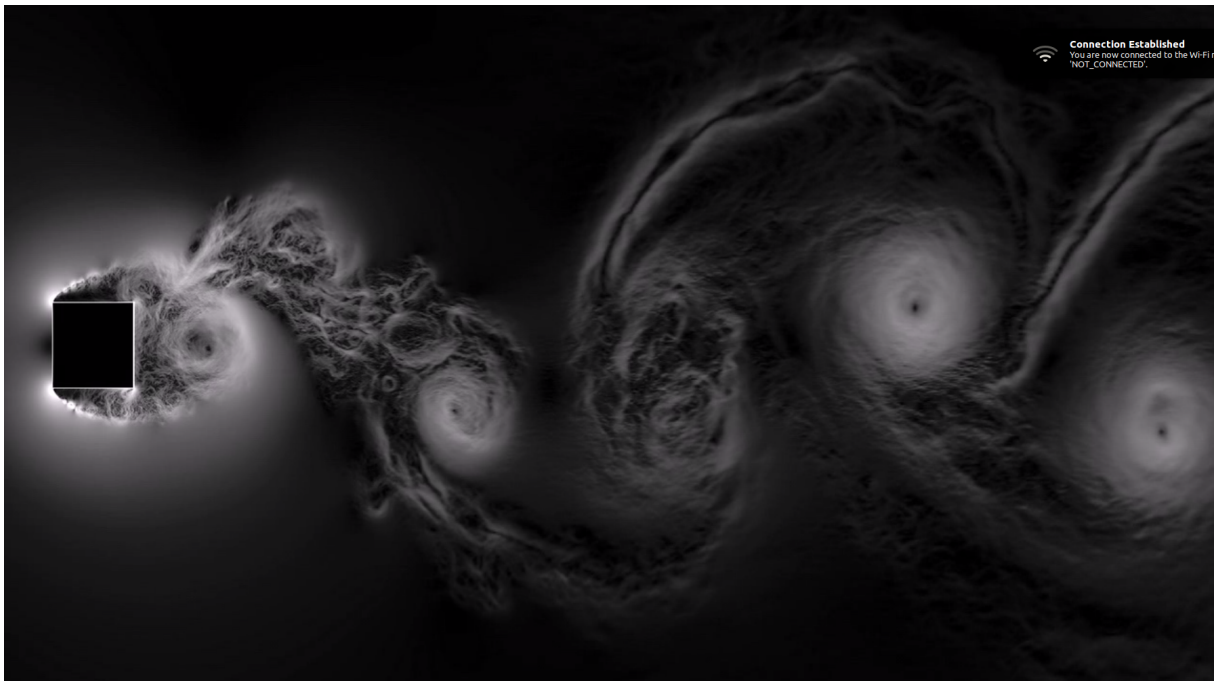

Figure 2: Flow around a Square Cylinder.
Source: [12]

6 Data Pre-processing

The above datasets contain raw DNS information such as coordinates, mean flow velocities, reynolds stresses and the dissipation factor. In order to convert the data into the required information, some calculations are required. Also, not all the data points are useful. Some of the data points are noisy and incorrect. Pre-processing is required in order to filter out these points.

In most of the datasets, the below variables are given. If they are not present, they are assumed zero. The notations have their standard meanings.

Given variables: x, y, z, $\overline{u}$, $\overline{v}$, $\overline{w}$, u'u', v'v', w'w', u'v', v'w', u'w', ϵ.

Required Variables: R, S, b(reynolds stress anisotropy tensor)

The following steps are adopted for data formulation and pre-processing:

1. Velocity gradients $\frac{dU_i}{dx_j}$ are obtained from U_i and X_j where $i, j \in (1, 2, 3)$. These gradients are obtained using *techplot* (CFD processing software).
2. Gradients matrix U_{ij} and its transpose U_{ji} are generated.
3. Rotation-rate tensor (R) and strain-rate tensor (S) are obtained by

$$S = \frac{1}{2} * \frac{k}{\epsilon} * (U_{ij} + U_{ji}) \tag{4}$$

   and

$$R = \frac{1}{2} * \frac{k}{\epsilon} * (U_{ij} - U_{ji}) \tag{5}$$

   where $k = u'u' + v'v' + w'w'$ is turbulent kinetic energy and ϵ the turbulent dissipation rate.
4. Ground truth anisotropy tensor (b) is obtained using

$$b_{ij} = \frac{\overline{u_i' u_j'}}{2k} - \frac{1}{3\delta_{ij}} \tag{6}$$

   where $\overline{u_i' u_j'}$ are reynolds stresses and δ_{ij} is dirac delta function.
5. Some of the data points are incorrect as they have $k < 0$ while turbulent kinetic energy must be a non-negetive number. Such points are filtered out from the training set.
6. Further, input to the neural network $\lambda's$ and T's are obtained from R and S using the transformations defined by Pope [9].

7 Experiments

7.1 Experiment on Channel Flow Dataset:

Figure 3. shows the performance of the TBNN model on channel flow dataset. It shows the plots of true and predicted Reynolds stress anisotropy tensor (b) against the data points. To be precise, the first 56 data points represent Re=180, 56-120 data points represent Re=395 and the points post 120 denotes Re=590.

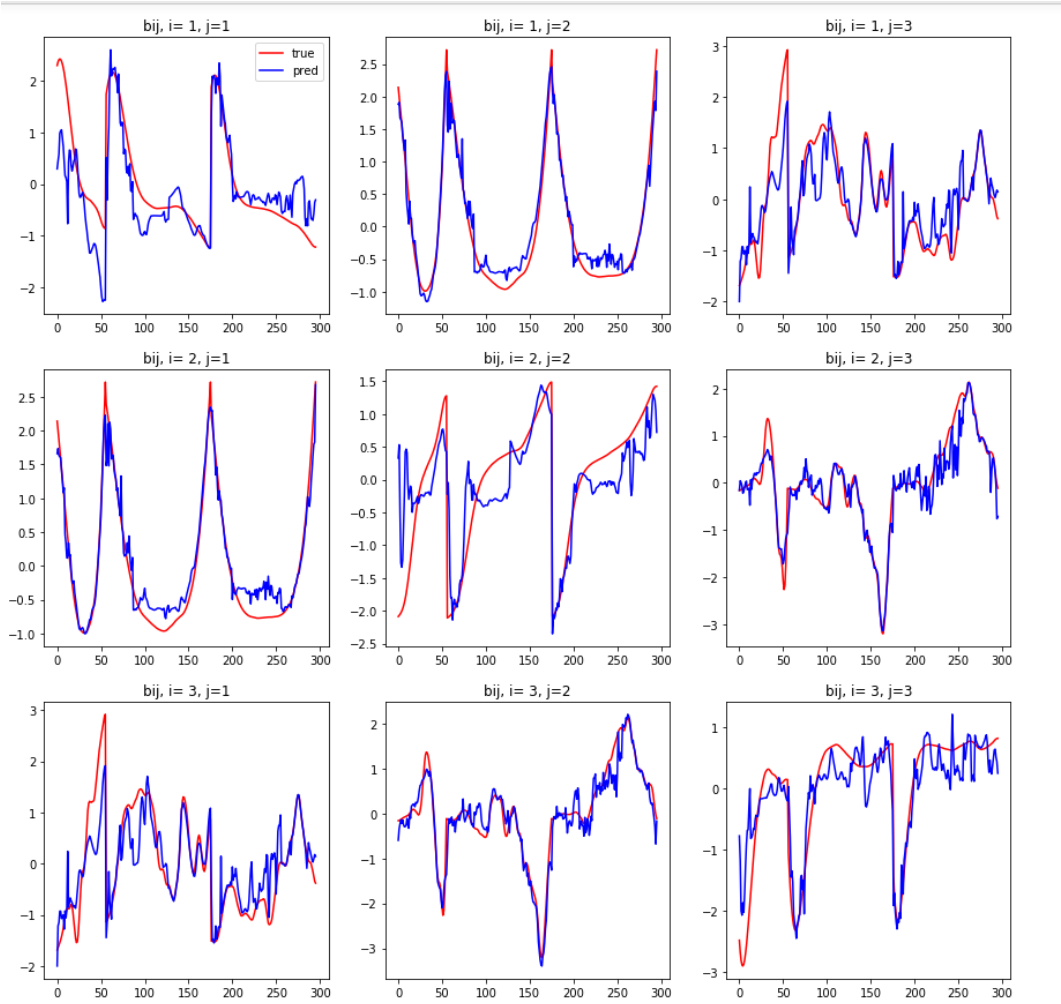

Figure 3: Results of TBNN model on channel flow dataset. Each of the 9 sub-figures shows true (red) and predicted (blue) values of b_{ij}.

Conclusion:

As seen in figure 3., the model fits well on the channel flow dataset. Though the number of data points are too less, the neural network tends to fit the points well. Many small fluctuations can be seen because the function (neural network) has a lot of parameters or degrees of freedom. The data is insufficient to train all the parameters which induces fluctuations. A lot more data is required to reduce the fluctuations.

7.2 Experiments on Ductflow Dataset

7.2.1 Experiment 1: Train and test on Re=300 dataset

In the first experiment, only Re=300 dataset was used to train and test the TBNN model. This experiment gives an idea of how well the neural network is able to model a large dataset like the ductflow Re=300 case with 16900 data points.

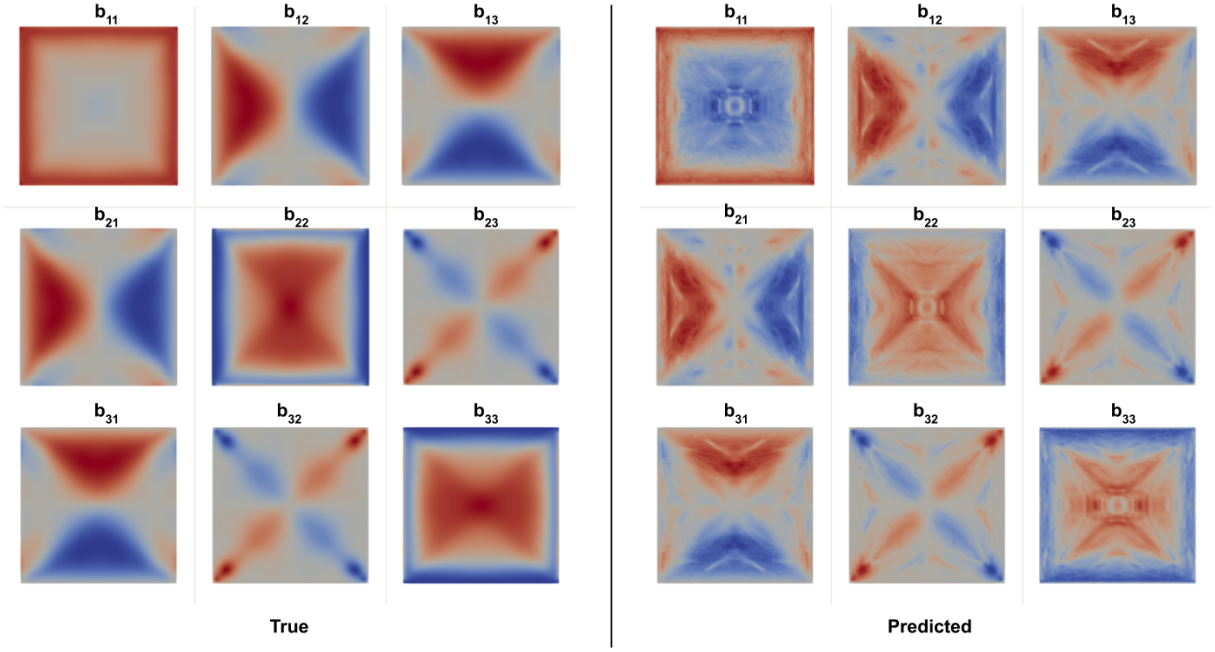

Figure 4: Results of TBNN model on Ductflow Re=300 dataset. Left: True b values obtained from DNS. Right: TBNN predicted values of b. The value increases from deep blue to deep red.

Comparison of the TBNN predictions with the ground truth values of b (DNS) is made in Figure 4. The error in the predictions is obtained using equation (3) as

$$\text{RMSE Loss} = 0.028$$

Conclusion: As seen in Figure 4., the predictions are fairly accurate with a lot of fluctuations in between the correct values. Visually the non-diagonal elements of b are predicted more accurately than the diagonal elements. The boundary predictions also seems to be fairly accurate. For diagonal elements, various patterns emerge in the predictions. The fluctuations can be smoothened (using a Gaussian kernel) to get more accurate predictions.

RMSE loss of 0.028 suggests that each of the values of b_{ij} is off by 0.028 on an average from its true value. Considering

$$b_{ij} \in (-\tfrac{1}{3}, \tfrac{2}{3}) \;\; \forall i \neq j \text{ and}$$
$$b_{ij} \in (-\tfrac{1}{2}, \tfrac{1}{2}) \;\; \forall i = j$$

the percentage error in predictions can be calculated as

$$\begin{aligned} \%\text{error} &= \frac{\text{RMSE Loss}}{\text{range of b}} * 100 \\ &= \tfrac{0.028}{1} * 100 \\ &= 2.8\% \end{aligned}$$

Hence, the predicted values of b are 2.8 % off by the true values which is a tolerable level of error.

7.2.2 Experiment 2: Points violating the constraints for Re=300

To ensure realizability, the following constraints are imposed on the predicted anisotropy tensor.

$$-\frac{1}{2} \leq b_{ij} \leq \frac{1}{2} \qquad \forall i = j \tag{7}$$

$$-\frac{1}{3} \leq b_{ij} \leq \frac{2}{3} \qquad \forall i \neq j \tag{8}$$

$$\varepsilon_1 \geq \frac{3|\varepsilon_2| - \varepsilon_2}{2} \tag{9}$$

$$\varepsilon_1 \leq \frac{1}{3} - \varepsilon_2 \tag{10}$$

Figure 5. shows the flagged points which violates one of the above constraints.

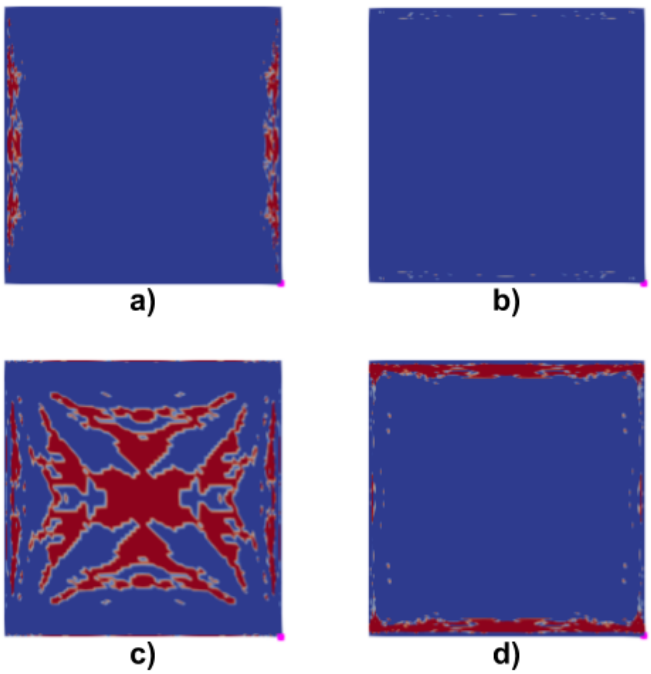

Figure 5: Points (marked in red) violating the constraints. Sub-figure a) Violates the b_{22} constraint given by equation (7). b) violates the b_{33} constraint given by equation (7). c) violates the eigenvalue constraint given by equation (9). d) violates the eigenvalue constraint given by equation (10).

Each of the subfigures (a), (b), (c), (d) in Figure 5. violates a different constraint. Figure 4(a) and 4(b) shows data points violating constraint equation (7) for b_{22} and b_{33} respectively. Figure 4(c) and 4(d) depicts points violating the eigenvalue constraint equations (9) and (10) respectively.

Conclusion: From Figure 5., one can infer that equation (9) is the constraint violated by the most number of points (fig. 4c) followed by equation (10) violated by a few points on the upper and lower boundaries. A few points on left and right boundaries violate constraint (7) for b_{22} while a negligible number of points

violate constraint (7) for b_{33}. The other constraints (such as b_{11} and equation (8)) are not at all violated by any of the points and hence they are not shown in the figure.

7.2.3 Experiment 3: Train and test individually for flows with different Reynolds number

To test the TBNN modelling ability with an increase in reynolds number, this experiment was conducted. Four different models A, B, C and D were trained individually on Re=300, 600, 900 and 1200 respectively. The results are compiled in Table 3.

Model	Test Case	# data points	# training iterations	RMSE Loss
A	Re=300	16900	10000	0.028
B	Re=600	67600	10000	0.034
C	Re=900	150544	10000	0.057
D	Re=1200	268324	10000	0.053

Table 3: Model Comparison for four different Reynolds numbers.

Conclusion: Table 3. shows the RMSE error for the four different test cases. Each of the models is trained for 10000 iterations. The number of data points increases from 16900 for Re=300 to 268324 for Re=1200. However, The RMSE error increases from 300 to 900 and decreases by a small amount for Re=1200. As the reynolds number increases, the flow becomes more and more turbulent, its complexity increases and modelling the flow becomes difficult. This explains the error increase form 300 to 900 case. However, the error decreases a little for Re=1200 probably because of lesser noise in the dataset as compared to Re=900 case.

7.2.4 Experiment 4: Models trained on Re=300 and Re=600, tested on Re=900

In this experiment, we train the models on Re=300 and Re=600 ductflow cases and test the model's generalizability on Re=900 test case. Here we try three different normalization settings as explained below. We also compare these three models with the model trained solely on Re=900 and tested on the same. The quantitative and qualitative performance of the models are given in Table 4. and Figure 6. respectively.

Model	Train dataset	Normalization ?	Norm. condition	RMSE Loss
A	Re=900 (itself)	yes	self norm	0.057
B	Re=300 and Re=600	yes	individual norm	0.069
C	Re=300 and Re=600	yes	same norm	0.076
D	Re=300 and Re=600	no	no norm	0.091

Table 4: Error comparison between different models tested on Ductflow Re=900 case.

Table 4. gives the error comparison between the four models. The meaning of normalization conditions is as follows.

- self norm — dataset (Re=900) is normalized by its own mean and variance.
- individual norm — each dataset (Re=300, Re=600 and Re=900) is normalized individually with its own mean and variance.
- same norm — all datasets (Re=300, Re=600 and Re=900) are normalized with a same mean and variance.
- no norm — no normalization is used at all.

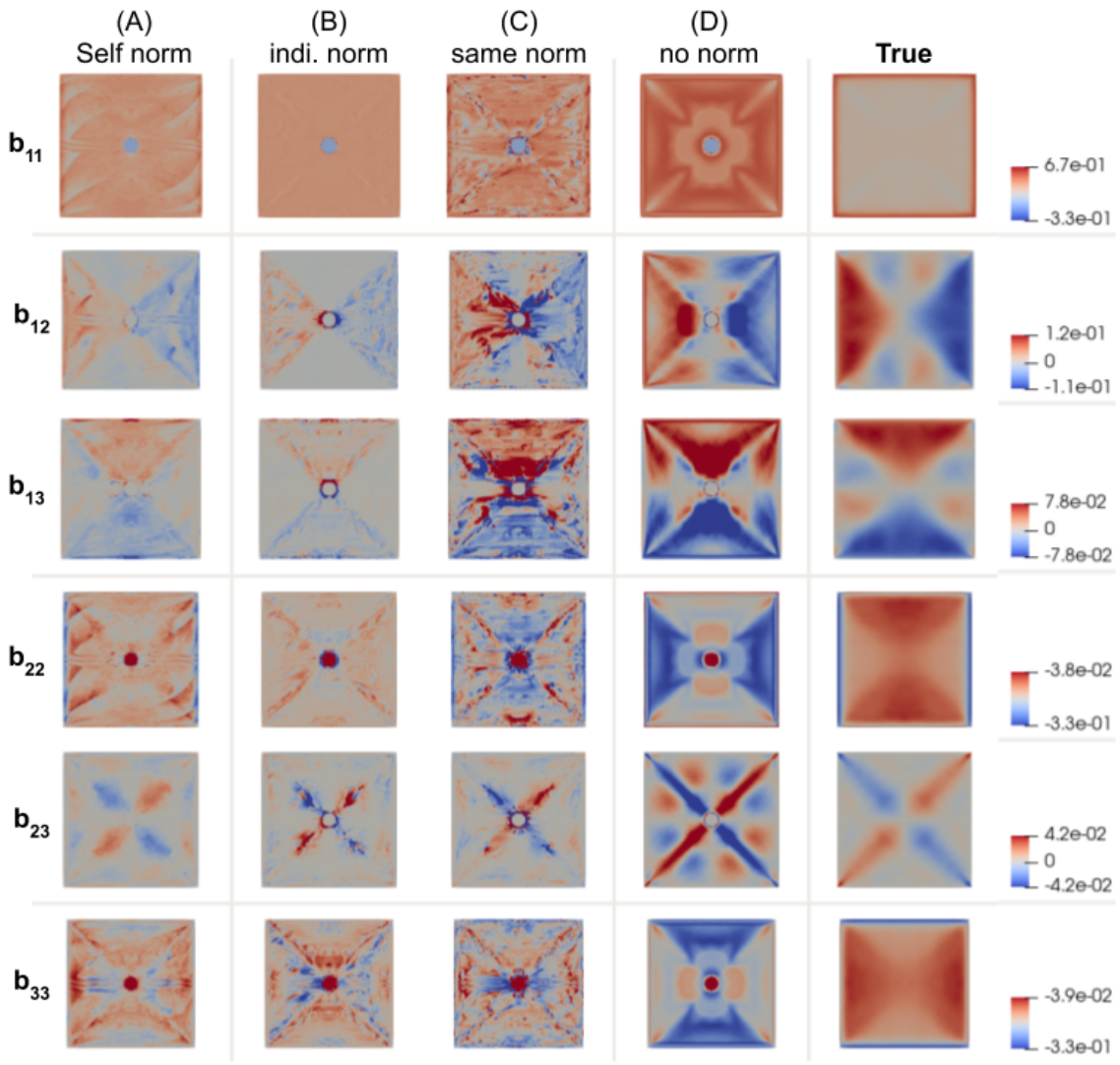

Figure 6: Qualitative Results of TBNN model on Ductflow Re=900 dataset. (A) self norm — dataset (Re=900) is normalized by its own mean and variance. (B) individual norm — each dataset (Re=300, Re=600 and Re=900) is normalized individually with its own mean and variance. (C) same norm — all datasets (Re=300, Re=600 and Re=900) are normalized with a same mean and variance. (D) no norm — no normalization is used at all.

The models A, B, C and D given in Figure 6. have the same meaning as in Table 4. Each of the first four columns are predictions of the four models and the last column is the ground truth value of b obtained from DNS. Each of the six rows give the results on each b_{ij} value. $b_{ij} = b_{ji}$ hence only six of nine values are shown. The scale is same across a row but differs for each b_{ij} value for better visualization.

Conclusion: Comparing Table 4. and Figure 6. one can conclude the following. According to Table 4. the model trained on Re=900 data (model A) performs better than the other three models. The error increases from model A to model D suggesting a decrease in performance. The qualitative results depict exactly the opposite trend. Model D is seen to represents the true value much better than the other models. The qualitative performances appears to decrease from model D to model A as against the quantitative performances. One reason for this trend in results could be that model D predicts the extreme values well

(extremum on the scale) but fails to predict the intermediate values correctly. Where as model A would be predicting the intermediate values well but would not be able to predict the extreme values. Hence, visually model A might appear to perform wrost than other models.

Discussion: The results of normalizing in different ways is as follows. Model does not predict correctly when normalized by a different mean and std. e.g. when only Re=300 data is loaded, model predict correctly on Re=300. But when Re=300 and Re=600 are loaded, model fails to give correct results on Re=300 case because now the mean and std are different from what it was trained on.

Normalizing by the same mean and std deviation for all datasets also did not improve the results much. One thing that was observed when the model was trained on Re=600, it tends to unlearn what it learned when trained on Re=300.

To illustrate this effect, a model was trained first on Re=300 and then on Re=600. A copy of the model was saved when training on Re=300 was done (say model A). Then another copy was saved when training on both Re=300 and Re=600 was done (say model B). When model A was tested on Re=300 again, it performed very well (because it is only trained on Re=300 itself). But when model B was tested on Re=300, it did not perform well (though it was first trained on Re=300 and then trained on Re=600). The model tends to unlearn its training on Re=300.

One of the following two things are suspected:

1. The model is generalizing to the other datasets and hence it loses its ability to predict correctly on Re=300 dataset.
2. There are some inherent differences in the two datasets. Say for the same R and S value, there are different true b values in Re=600 dataset than in Re=300 dataset.

7.3 Experiment on Cylinder-flow dataset

Figure 7. shows the performance of the TBNN model on flow around a square cylinder dataset. This dataset is a very turbulent flow with a high reynolds number Re=22000. It also has a large number of data points (1498224 points).

Conclusion:

As seen in Figure 7. the model perform poorly on this dataset. The dataset has a lot of noise and has lot of unnecessary readings far away from the cylinder. The model is trained with a lot of data far away from the cylinder. Hence, probably the model is not able to capture the intrinsic details of the true flow close by the cylinder. As also seen in 7.2.3, the model performance decreases for high reynolds number. Reynolds number for this dataset is extremely high which might result in the poor performance of the model.

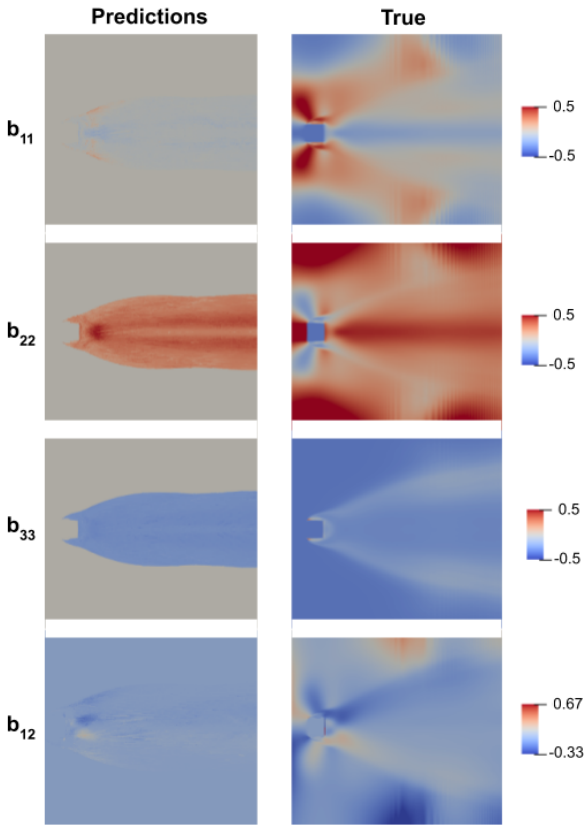

Figure 7: TBNN predictions on flow around a square cylinder. Left: Predictions, Right: True values.

8 Conclusion

The following overall conclusions can be drawn from the experiments performed in section 7.

- The performance of the model degrades with increase in reynolds number of the flow. That is, as the reynolds number is directly proportional to the flow velocity, as the flow velocity increases the model performance decreases.
- This is evident from the experiments performed in section 7. Section 7.1 shows channel-flow dataset with very low reynolds numbers (below Re=600). The model performs accurately on this dataset. Section 7.2 shows ductflow dataset with increasing reynolds numbers. The model performance decreases gradually from Re=300 to Re=1200. Finally, section 7.3 contains flow around a cylinder with Re=22000 which is a very high reynolds number. The model performs poorly on this dataset.
- The predictions of the model are very fluctuating especially in sections 7.1 and 7.2. This is because the model is designed to be trained on large datasets. However, in these experiments, the model is trained on small or intermediate sized datasets. The model complexity is too high which results in the fluctuations. One work around for this issue is to smoothen the curve using Gaussian kernel or any other technique.
- Overall, the TBNN (deep learning) model performs significantly better than the linear or quadratic models for RANS turbulence modelling.
- This method can be adopted in practice where quick (real-time) solutions are required but high accuracy is not needed.

9 Future Work

Implementing the Tensor Based Neural Network and analysing its results on various datasets has opened door for a large number of future works.

- As seen in Figure 5., a large number of points violate the eigenvalue constraints given in equation (9). However, how to limit the points so that they obey these constraints is still to be explored.
- The current results are very fluctuating. Error must be compared after smoothening the predicted results. This post processing is very likely to improve the results.
- Some more datasets are publicly available on the internet. Experimenting on these datasets will give more insights about the TBNN model. The datasets are as follows.
- Flow through a convergent-divergent nozzle [13]:

  Flow Reynolds Number = 12600

  Number of data points = 887040

  The dataset contains DNS information of turbulent flow at Re=12600 through a convergent-divergent nozzle. The flow is in X direction.
- Backward Facing Step Flow [14]:

  Flow Reynolds Number = 395

  Number of data points = 459648

  Contains DNS information of Re=395 flow around a backward step.

References

[1] CRAFT , T. J., LAUNDER , B. E. SUGA , K. 1996 Development and application of a cubic eddy-viscosity model of turbulence. Intl J. Heat Fluid Flow 17, 108–115.

[2] TRACEY , B., DURAISAMY , K. ALONSO , J. J. 2013 Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. AIAA Aerospace Sciences Meeting 2013- 0259.

[3] ZHANG , Z. J. DURAISAMY , K. 2015 Machine learning methods for data-driven turbulence modeling. AIAA Computational Fluid Dynamics Conf. 2015-2460.

[4] LING , J., J ONES , R. T EMPLETON , J. 2016 Machine learning strategies for systems with invariance properties. J. Comput. Phys. 318, 22–35.

[5] GOODFELLOW, I., BENGIO, Y. COURVILLE, A. 2016 Deep Learning. MIT Press.

[6] LING , J. TEMPLETON , J. A. 2015 Evaluation of machine learning algorithms for prediction of regions of high RANS uncertainty. Phys. Fluids 27, 085103.

[7] MOGHADDAM, A. A. SADAGHIYANI, A. 2018 A deep learning framework for turbulence modeling using data assimilation and feature extraction. arXiv preprint arXiv:1802.06106 .

[8] LING, J., KURZAWSKI, A. TEMPLETON, J. 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. 807, 155-166.

[9] POPE , S. B. 1975 A more general effective-viscosity hypothesis. J. Fluid Mech. 72, 331–340.

[10] MOSER, R. D., KIM, J. MANSOUR N. N. 1998 DNS of Turbulent Channel Flow up to Re_tau=590. Phys. Fluids. 11, pg 943-945.

[11] ZHANG, H., TRIAS, F. X., GOROBETS, A., TAN, Y., OLIVA, A. Direct numerical simulation of a fully developed turbulent square duct flow up to Ret = 1200. 2015 Int. J. Heat Fluid Flow 54, 258.

[12] TRIAS, F. X., GOROBETS, A. OLIVA, A. 2015 Turbulent flow around a square cylinder at Reynolds number 22,000: A DNS study. Computers and Fluids 123 (22), 87–98.

[13] LAVAL, J.-P. and MARQUILLIE, M., 2010. Direct Numerical Simulations of Converging-Diverging Channel Flow, Progress in Wall Turbulence: Understanding and Modelling, Villeneuve d'Ascq, France, April 21-23.

[14] PONT-VILCHEZ, A., TRIAS, F. X., GOROBETS, A. OLIVA, A. 2019 Direct numerical simulation of backward-facing step flow at Re = 395 and expansion ratio 2. Journal of Fluid Mechanics, 863:341–363.