Optimising a Machine Learning Model for Reynolds Averaged Turbulence Modelling of Internal Flows

OPTIMISING A MACHINE LEARNING MODEL FOR REYNOLDS AVERAGED TURBULENCE MODELLING OF INTERNAL FLOWS

Man A.[1]*, Jadidi M.[1], Keshmiri A.[1], Yin H.[2] and Mahmoudi Y.[1]
*Author for correspondence
[1]Department of Mechanical, Aerospace and Civil Engineering,
[2]Department of Electrical and Electronic Engineering,
University of Manchester,
Manchester, M13 9PL
United Kingdom
E-mail: anthony.man@manchester.ac.uk

ABSTRACT

Various machine learning models in recent years have demonstrated capability in improving the accuracy of Reynolds-averaged Navier-Stokes (RANS) simulations. One such example is the tensor-basis neural network (TBNN), which has been shown to introduce improvements that would be challenging or impossible for RANS approaches to model, such as anisotropic normal Reynolds stresses. This paper reports on a hyper-parameter tuning exercise, to find optimal parameter settings for TBNNs predicting Reynolds stress anisotropy in channel flow, and these are also used in TBNNs deployed on internal flow over a forward-backward facing step. The accuracy of ensemble TBNN predictions is investigated, and it is shown that further improvements to Reynolds stress anisotropy predictions can be made when an ensemble size of 25 or more is used.

NOMENCLATURE

b_{ij}	[-]	Reynolds stress anisotropy tensor
g_n	[-]	General effective viscosity hypothesis coefficients
k	[m^2/s^2]	Turbulent kinetic energy
Re_τ	[-]	Channel flow Reynolds number
$\boldsymbol{S}$	[-]	Dimensionless mean strain rate tensor
$\boldsymbol{T}^{(n)}$	[-]	General effective viscosity hypothesis tensors
u_τ	[m/s]	Friction velocity

Special characters

δ	[m]	Channel half height
ε	[m^2/s^3]	Turbulent kinetic energy dissipation rate
ν_t	[m^2/s]	Eddy viscosity
$\boldsymbol{\Omega}$	[-]	Dimensionless mean rotation rate tensor

Subscripts

ij	Free indexes in Reynolds stress anisotropy tensor
k	Dummy index for number of location points
n	Dummy index for terms in general effective viscosity hypothesis

INTRODUCTION

Reynolds-averaged Navier-Stokes (RANS) approaches are widely used for simulating turbulent flows due to their computational efficiency and tractability. Most RANS models, including popular two-equation ones such as k-ω shear stress transport (SST), use Boussinesq Hypothesis (BH) for Reynolds stress closure. However, due to several assumptions that BH is based on, it is well-known that these RANS models do not give satisfactory accuracy in predicting certain flow features, including flow stagnation, recirculation, and separation [1]. Furthermore, BH cannot model anisotropic normal Reynolds stresses, which exist in boundary layers and secondary flows [2].

Nonlinear eddy viscosity models have been proposed as alternatives to BH [3]. These models use higher-order products of mean strain and rotation rate tensors to predict Reynolds stress, which have been shown to simulate stagnation and streamline curvature effects more accurately and can model anisotropic normal Reynolds stresses. However, nonlinear models have not seen widespread usage, as their coefficients must be tuned through a long and tedious process for every flow problem to give consistent accuracy improvements over BH [4].

To address the deficiencies in RANS approaches and improve their prediction accuracy, there has been gathering interest over recent years in using machine learning (ML) for modelling Reynolds stress. Neural networks (NNs) have commonly been the ML model of choice in these applications for their flexibility in approximating complex functions, such as Reynolds stress closure [5, 6]. In particular, the tensor-basis neural network (TBNN) developed by Ling et al. [7] has seen popularity, due to its high generalisation performance and prediction accuracy resulting from embedding physical laws in the model. Ling et al. showed that TBNN could reduce root mean squared errors in Reynolds stress anisotropy predictions from RANS by 44% for square duct flow and 56% for flow over periodic hills. Furthermore, the prediction of anisotropic normal Reynolds stresses and secondary flows were also reported [7].

Despite the proven capabilities of TBNN, numerous open questions remain regarding its parameters and best practices for model deployment. The best choices for TBNN hyperparameters that can consistently yield high prediction accuracy are not well-defined in the literature, including the learning rate and number of hidden layers and nodes. These are vital considerations for optimising the predictive performance of any NN. Although some TBNN studies have recorded attempts at tuning these parameters, their investigations have been limited in parameter space and detail. Due to randomness in weight initialization, NNs often converge to different local minima during training, instead of the global minimum. Therefore, an ensemble of

TBNN instances can be trained to provide multiple predictions that can be averaged to give a mean result when testing. The optimal number of these TBNN ensembles for prediction averaging has also not been studied properly in the literature. This work focuses on addressing these considerations for TBNNs trained, validated, and tested on channel flow shown in Figure 1(a). The optimal TBNN hyperparameters were used afterwards to predict anisotropy in flow over a confined forward-backward facing step (F-BFS) – a canonical internal flow problem, shown in Figure 1(b).

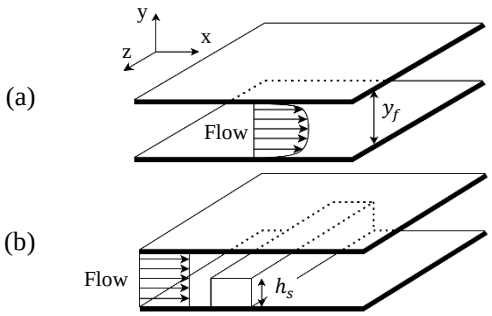


Figure 1 (a) Channel flow, (b) Flow over F-BFS

TENSOR BASIS NEURAL NETWORK

The mathematical foundation of TBNN is based on the general effective viscosity hypothesis (GEVH) – a generalised expansion of the Reynolds stress anisotropy tensor b_{ij} [8]:

$$b_{ij} = \sum_{n}^{10} g_n \boldsymbol{T}^{(n)} \qquad (1)$$

Coefficients g_1 to g_{10} are unknown scalars but are functions of the following five invariants of non-dimensional mean strain rate $\boldsymbol{S}$ and mean rotation rate $\boldsymbol{\Omega}$:

$$g_n = f\left(Tr(\boldsymbol{S}^2), Tr(\boldsymbol{\Omega}^2), Tr(\boldsymbol{S}^3), Tr(\boldsymbol{\Omega}^2\boldsymbol{S}), Tr(\boldsymbol{\Omega}^2\boldsymbol{S}^2)\right) \qquad (2)$$

where $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ have been non-dimensionalised by turbulent kinetic energy (TKE) k and turbulent dissipation rate ε. Tensors $\boldsymbol{T}^{(1)}$ to $\boldsymbol{T}^{(10)}$ are known functions of $\boldsymbol{S}$ and $\boldsymbol{\Omega}$, with terms consisting of their tensor products. Note that BH and nonlinear eddy viscosity models are truncated versions of the GEVH and hence Equation (1) gives their full expression. The goal of the TBNN is to approximate coefficients g_1 to g_{10}, which can then be used in Equation (1) to solve for anisotropy tensor.

Figure 2 shows that the TBNN architecture is specially designed to imitate the GEVH in Equation (1). The tensor input layer accepts $\boldsymbol{T}^{(1)}$ to $\boldsymbol{T}^{(10)}$ tensors as inputs and hence contains ten nodes. Similarly, the invariant input layer accepts the invariants in Equation (2) as inputs and therefore has five nodes. These invariants are propagated through a series of hidden layers, aiming to collectively approximate the complex function of Equation (2). The final hidden layer contains ten nodes, corresponding to approximations of coefficients g_1 to g_{10}. Finally at the merge output layer and to replicate Equation (1), element-wise products of the tensor input layer and final hidden layer are calculated and these terms are summed to give an approximation for anisotropy tensor. This architecture guarantees Galilean invariance in its anisotropy predictions.

The ML workflow for TBNN typically consists of model training, validating, and testing, with all three tasks requiring their own datasets. Fitting the TBNN to the training dataset allows coefficients g_1 to g_{10} to converge to an approximation. Stopping of the training process can be invoked after a certain number of epochs by testing the TBNN on the validation set. Model parameters can also be tuned by evaluating the validation performance after trying different parameter settings. Once fully trained, the TBNN may be tested on the test set to assess its predictive performance and readiness for real-world deployment. In these three tasks, RANS data is supplied to the two input layers and anisotropy values predicted by the TBNN are compared with those from a high-fidelity method, such as LES, which act as ground-truth values for evaluating the accuracy of TBNN predictions. Therefore, data from computational fluid dynamics (CFD) cases simulated with both RANS and a high-fidelity method must be gathered to create a dataset for each task. More specifically, these datasets must include $\boldsymbol{T}^{(1)}$ to $\boldsymbol{T}^{(10)}$ tensors and Equation (2) invariants calculated from RANS, as well as anisotropy results from the high-fidelity method.

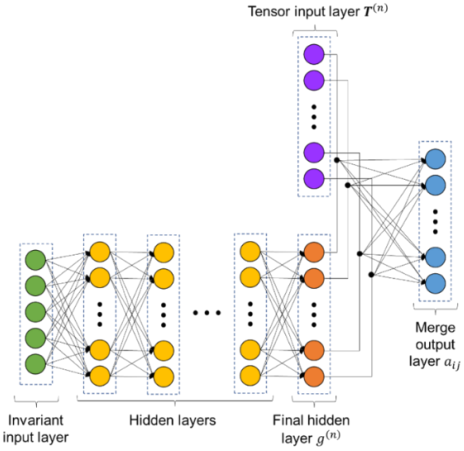


Figure 2 Typical TBNN architecture

CHANNEL FLOW DATASETS

Channel Flow Domain and Boundary Conditions

TBNNs were firstly trained, validated, and tested on turbulent channel flow problems. Seven channel flow cases with Reynolds numbers of 180, 290, 395, 490, 590, 760 and 945 based on friction velocity and channel half-height were simulated with both RANS and LES to create the datasets. Their corresponding bulk velocities were 2.2, 3.8, 5.4, 7.0, 8.6, 11.7 and 14.7 m/s. A truncated channel domain was used for all cases, with channel half-height δ = 0.04m, length and width 8δ, and kinematic viscosity of 1.57×10^{-5} m²/s. Periodic boundary conditions (BCs) were enforced in the streamwise and spanwise directions for all simulations, while no-slip and non-permeable BCs were assigned to the walls. At the walls of the RANS cases, a Neumann BC for pressure was used and specific TKE dissipation rate ω was set to a fixed value given by the equation proposed by Menter for SST in [9]. Eddy viscosity ν_t was set to calculated to allow transport equations to be integrated down to the wall. For the LES cases, Neumann BCs were applied for pressure and

eddy viscosity at the walls as wall-resolved LES was performed. Constant bulk flow velocity was maintained in all cases by an external force that was added into the momentum equation using the fvOptions utility in OpenFOAM, as the BCs used do not allow flow momentum to enter the domain. Grid independence studies were conducted for the RANS and LES simulations – the results for the LES one are shown in Figure 3(a). These studies led to 3.64×10^6 and 9.90×10^6 elements being used in the RANS and LES simulations respectively, with their grid resolutions shown in Table 1.

Table 1 Channel flow grid resolution

Cell dimensions	RANS	LES
Δx^+	76	48
Δy^+ (min)	2	0.95
Δy^+ (max)	12	12
Δz^+	38	24

Channel Flow Numerical Method

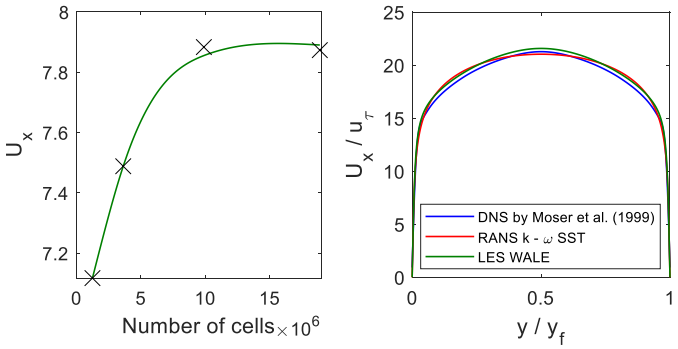
To demonstrate the applicability of TBNN to modern CFD practices, SST was the chosen turbulence model in the RANS simulations as it is well-validated and widely used in industry. The SimpleFoam solver in OpenFOAM was used to run the steady state RANS cases with the SIMPLEC setting to provide a more robust solution and the simulations were run until the initial residuals converged to less than 1×10^{-6}. Gauss linear was used as the gradient, interpolation and all divergence discretisation schemes for its second order accuracy.

The LES cases were run using the Wall-Adapting Local Eddy Viscosity (WALE) turbulence model. Gauss linear was used for the interpolation, gradient and divergence schemes to avoid introducing numerical dissipation, and PisoFoam was chosen as the solver. To carry out time integration, backward Euler was used for its second order accuracy and the maximum Courant number was kept below 0.7. The initial condition was provided by the converged RANS results of corresponding Reynolds number and the flow was perturbed using the perturbU utility in OpenFOAM before running the simulations.

Channel Flow Data Pre-processing

The noise of turbulent fluctuations in LES results can make accurate mapping of RANS quantities to anisotropy from LES challenging when training TBNNs. Therefore, time-averaged values of Reynolds stress were gathered by performing statistical time-averaging during the LES simulations for 50 δ/u_τ seconds after ten through-flow times. As channel flow is homogeneous in the streamwise and spanwise directions, the RANS and LES data were also spatially-averaged in these dimensions, giving their averaged results through the wall-normal direction. DNS results of channel flows at four Reynolds numbers (Re_τ = 180, 395, 590, 945) were used to validate these 1D results, with Figure 3(b) showing validation for the Re_τ = 590 case [10, 11]. Finally, the 1D LES results were interpolated at the RANS cell-centre wall-normal coordinate values in order for the RANS and LES results to have the same number of datapoints.

Data from five cases were used for training: Re_τ = 180, 290, 490, 760 and 945. Experiments have demonstrated that multi-layer perceptrons – which the TBNN is a type of – perform worse in extrapolation compared to interpolation, and especially when training data distribution is not sufficiently diverse [12, 13]. Therefore, Re_τ = 395 was selected for validation and Re_τ = 590 was chosen for testing on the basis that the Reynolds numbers for these cases are enveloped by those in the training dataset.



(a) (b)

Figure 3 (a) Probe results of mean streamwise velocity U_x at y^+ = 15 for Re_τ = 945 case simulated with LES, showing grid independence and (b) U_x profiles for Re_τ = 590 case simulated with DNS, LES and RANS

RESULTS

Hyperparameter Tuning

Five hyperparameters that can have significant influence on the training of any NN are the number of hidden nodes and the number of hidden layers, learning rate, batch size and choice of activation functions. A grid search was performed to find optimal choices for these parameters to use in TBNN deployed on channel flow. The parameter space was discretised as shown in Table 2, resulting in 3240 different combinations experimented. For each combination, 25 instances of TBNN were separately trained, validated and tested, resulting in 25 anisotropy predictions when validating or testing the TBNNs, which could be averaged afterwards to give a mean anisotropy result. An ensemble size of 25 was chosen to balance computational cost with accuracy of the mean anisotropy as ensemble size increases.

Table 2 Hyperparameter space discretisation

Hyperparameter	Discretisation
No. of hidden layers	2, 5, 10, 20, 30
No. of hidden nodes	5, 10, 25, 50, 75, 100
Activation function for hidden nodes	sigmoid, Exponential linear unit (ELU), softplus, rectified linear unit (ReLU), leaky ReLU (leakiness = 0.01, 0.33)
Learning rate	10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}
Batch size	5, 10, 20

Aggregate mean squared error (AMSE) was used as the loss function and can represent the accuracy of a TBNN instance:

$$AMSE = \frac{1}{9N_{data}} \sum_{k=1}^{N_{data}} \sum_{i=1}^{3} \sum_{j=1}^{3} \left(b_{ij,k,predicted} - b_{ij,k,LES}\right)^2 \qquad (3)$$

N_{data} is the number of location points where anisotropy is predicted by the TBNN. After every 10 training epochs, each TBNN instance was tested on the validation set and subsequently produced an AMSE. Early stopping in training each TBNN instance was invoked if the average of the three most recent AMSE values were greater than the average of the three before them. The following averaged root mean squared error (ARMSE) was used to evaluate the average error for TBNN instances of a specific parameter combination:

$$ARMSE = \frac{1}{N_{instances}} \sum_{m=1}^{N_{instances}} \sqrt{AMSE} \qquad (4)$$

where $N_{instances}$ is the number of TBNN instances in an ensemble. Performance of the parameter combinations was assessed using three metrics: total run time required to train, validate and test 25 TBNN instances, final ARMSE in predicting the validation set which invoked early stopping and ARMSE in predicting the test set.

The top five combinations which returned the lowest final validation ARMSE are shown in Table 3. Although ReLU and its variants have been preferred activation functions in recent years to avoid the vanishing gradient problem (VGP), sigmoid and tanh have been reported to perform better than the ReLU family for simple regression problems [14]. This may be why three combinations that used sigmoid are featured in Table 3. These models likely only experienced small effects of VGP due to their shallow architectures of two hidden layers. Their run time were 2.4 to 3 times faster than the Softplus combination and 3.4 to 4.5 times faster than the ELU one, which can be attributed to their low number of hidden nodes and higher learning rate of 0.01, compared to 0.001 for the other two combinations.

Table 3 Lowest final validation ARMSE combinations

No. hidden layers	No. hidden nodes	Activation functions for hidden nodes	Total run time (s)	Final validation ARMSE	Testing ARMSE
5	50	ELU	379	5.82E-02	5.09E-02
2	10	Sigmoid	113	5.84E-02	5.19E-02
2	10	Sigmoid	85	5.84E-02	5.22E-02
2	75	Softplus	272	5.86E-02	5.25E-02
2	5	Sigmoid	100	5.87E-02	5.30E-02

Table 4 shows the top five combinations that returned the lowest testing ARMSE. It is clear that some ELU combinations performed significantly better in testing, while sigmoid ones do not feature at all. This is likely because ELUs have been shown to possess higher training and testing accuracy compared to ReLU and sigmoid in deep NNs, and the test set may have been better represented in the training data compared to the validation set, given that testing ARMSE is consistently lower than the final validation ARMSE in both tables [15]. Moreover, these larger architectures in Table 4 with more nodes are known to have better memorisation, meaning they can predict more accurately on test samples similar to those in the training set. As ReLU and ELU do not cause VGP, deeper architectures with 5 or 10 layers as shown can be trained, which are known to have better accuracy and generalisation performance too. The learning rates of the top three ELU combinations and remaining two were 0.001 and 0.0001 respectively. Combinations in both tables used batch size of 5 or 10.

Table 4 Lowest testing ARMSE combinations

No. hidden layers	No. hidden nodes	Activation functions for hidden nodes	Total run time (s)	Final validation ARMSE	Testing ARMSE
5	100	ELU	819	5.92E-02	5.09E-02
5	50	ELU	379	5.82E-02	5.09E-02
5	75	ELU	358	5.92E-02	5.13E-02
10	100	ELU	1942	5.92E-02	5.14E-02
10	75	Leaky ReLU	1049	5.93E-02	5.15E-02

Differences between Tables 3 and 4 suggest that the choice of CFD cases for the validation and test sets influences optimal parameter values. Therefore, experimenting with different cases for validating and testing will be explored in a future study. The only model to feature in both tables is the top combination in Table 3. Therefore, this one was considered to possess optimal parameter settings in the grid search.

Ensemble Averaging

To assess how the ensemble size of TBNN instances influences the accuracy of mean anisotropy, 200 instances of the optimal combination from hyperparameter tuning were trained, validated, and tested. The average anisotropy result for x number of instances was defined as the average anisotropy result from the first x number of instances. The accuracy of individual component predictions in the mean anisotropy was assessed using component mean squared error ($CMSE_{ij}$):

$$CMSE_{ij} = \frac{1}{N_{data}} \sum_{k=1}^{N_{data}} \left(b_{ij,k,predicted} - b_{ij,k,LES}\right)^2 \qquad (5)$$

Table 5 shows CMSE results for the main shear component b_{12} and normal components of the mean anisotropy for different ensemble sizes of TBNN instances deployed on the test set. For comparison, CMSEs for the RANS data is also shown, whereby $b_{ij,k,predicted}$ in Equation (5) was replaced with $b_{ij,k}$ from RANS data. Note that CMSEs for off-diagonal components are equal to their symmetric counterparts i.e., $CMSE_{12} = CMSE_{21}$. Results for $CMSE_{13}$ and $CMSE_{23}$ have been omitted from Table 5 because very low values were calculated, due to negligible values of b_{13} and b_{23} predicted by RANS, LES and TBNN. However, it is worth noting that 3.61×10^{-7} and 3.78×10^{-8} for $CMSE_{13}$ and $CMSE_{23}$ respectively were calculated for all ensemble sizes and RANS. This shows that TBNN training is almost only sensitive to the other components.

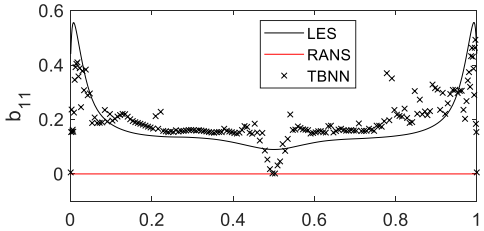
As the ensemble size increases from 5 to 25, CMSE reduces by 5% across the components in Table 5. Although all components of CMSE converge to non-zero values for an ensemble size of 25 or more, these results show that ensemble TBNNs can improve RANS results significantly, reducing $CMSE_{11}$, $CMSE_{22}$ and $CMSE_{33}$ by 80% and $CMSE_{12}$ by 57%.

Table 5 CMSE results for ensemble averages

No. TBNN instances	$CMSE_{11}$	$CMSE_{12}$	$CMSE_{22}$	$CMSE_{33}$
5	1.32E-02	7.99E-04	5.52E-03	1.83E-03
10	1.29E-02	7.66E-04	5.39E-03	1.77E-03
25	1.26E-02	7.56E-04	5.26E-03	1.74E-03
50	1.26E-02	7.53E-04	5.26E-03	1.73E-03
100	1.26E-02	7.60E-04	5.28E-03	1.74E-03
200	1.25E-02	7.59E-04	5.24E-03	1.74E-03
RANS	6.34E-02	1.76E-03	2.53E-02	9.02E-03

The remaining small discrepancies in these components may be explained with Figure 4, which shows their mean profiles predicted by the ensemble of 200 TBNN instances, along with those obtained from LES and RANS for comparison. y is the wall-normal distance from the bottom channel wall, which has been non-dimensionalised by the full channel height y_f. The left and right edges of these plots correspond to the bottom and top walls of the channel respectively. Values of zero were predicted for the normal components at the walls, in contrast to 0.44 for b_{11}, -0.33 for b_{22} and -0.11 for b_{33} by LES. This is because $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ used in the input layers were non-dimensionalised by TKE, and the no-slip condition dictates that TKE at the wall must be zero. Hence values of zero for $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ were given to the tensor input layer and Equation (1) returned zero for all b_{ij} components. This issue may be resolved by using a time-scale that does not include TKE for the non-dimensionalisation. A similar issue exists at the channel centre, whereby the TBNN profiles are 'pulled' to zero. The only non-zero velocity gradient component in channel flow is dU/dy, which becomes zero at the channel centre due to changing sign in the wall-normal direction. Therefore, values of zero were calculated here for all components of $\boldsymbol{S}$ and $\boldsymbol{\Omega}$, and also resulted in zero anisotropy predicted. This issue intrinsic to the GEVH has been reported in the literature and would require a modification to the GEVH and representative TBNN architecture to remedy it [16].

The authors believe the training of these TBNNs has been driven by the zero value constraints discussed above and loss reduction at the abundant number of datapoints between the buffer layer and channel centre (y/y_f = 0.1 to 0.4 and 0.6 to 0.9). TBNN weights and biases optimised for these regions have resulted in subpar accuracy in buffer layer predictions as a compromise. This may be resolved by training a separate TBNN model for the near-wall flow. Other issues pertaining to NNs are the outliers, such as at y/y_f = 0.8 and slight asymmetry in prediction between the top and bottom halves of the channel. These issues motivated the displaying of results for the full channel height in Figure 4. The asymmetry may be remedied by incorporating differences in anisotropy between location-symmetric data point pairs in the loss function.



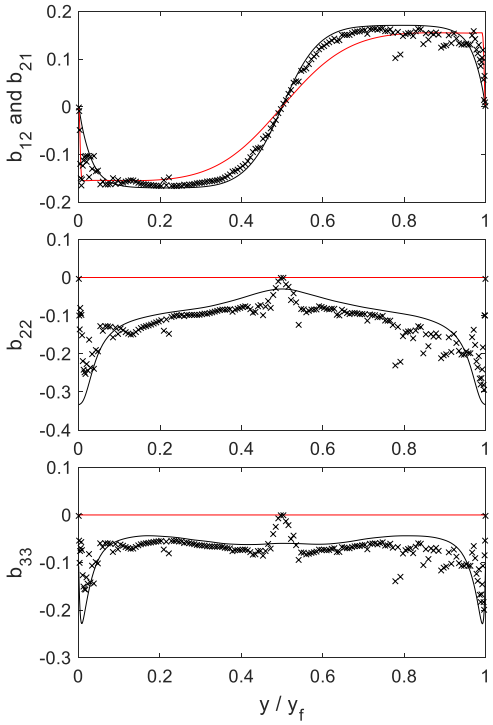


Figure 4 Channel flow anisotropy profile results

Flow over Forward-Backward Facing Step

To assess the performance on a more complex internal flow problem, an ensemble of 200 TBNNs with the top parameter combination from the channel flow study were separately trained, validated and tested to predict anisotropy in flow over a forward-backward facing step (F-BFS). Simulations of Reynolds number = 1800, 3600 and 7200 based on inlet velocities 1, 2 and 4 m/s and step height h_s = 18 mm were run with RANS and LES. As symmetric BCs were used in the spanwise direction in all simulations, results at the centreplane were extracted for TBNN workflow. The TBNNs were trained on centreplane data from the Re = 1800 case and validated on Re = 7200, before being tested on the Re = 3600 case. These were chosen so that the trained TBNNs would have some bias on two cases of Reynolds numbers that envelop the test case one.

b_{11} was found to have the highest magnitude out of all the components for the LES results and Figure 5 shows the contour results for b_{11} across the entire domain. It is clear that the TBNNs capture b_{11} more accurately in the regions of interest compared to RANS, e.g. on top and downstream of the block, and most crucially in the adverse pressure gradient region on the block leading edge. However, the TBNN prediction is inaccurate in the region upstream of the block as this flow is laminar and the physics in the downstream regions were represented more in the training data, with 74% of data for each case being downstream flow results. Therefore, improvements may be obtained in the upstream region predictions by providing more upstream flow data in the training dataset.

Lastly, Figure 6 shows the normal and main shear anisotropy profiles at x/h_s = 5. It is clear that TBNN predicts the normal components more accurately and follows the LES trend more closely than RANS. However, there is room for improvement in

the b_{12} component and the near-wall region for the normal components. The latter is due to the zero anisotropy constraint at the walls as discussed in the channel flow study, thereby showing this issue is present in more complex internal flows too.

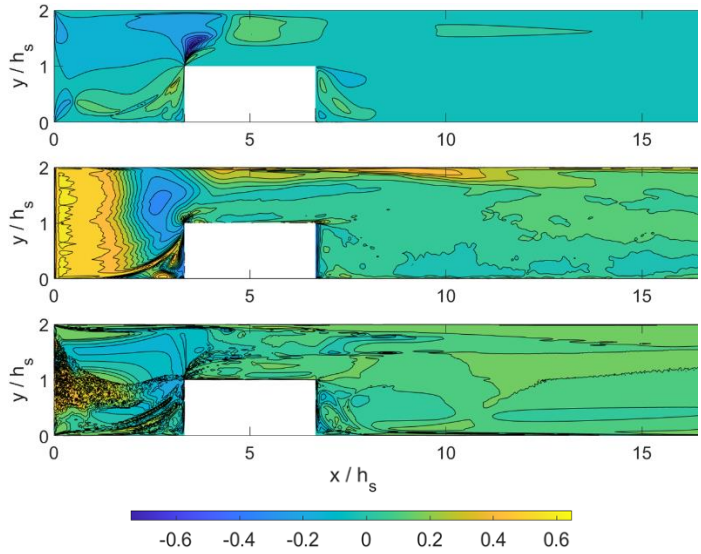


Figure 5 F-BFS b_{11} contour results, from top to bottom: RANS, LES and TBNN

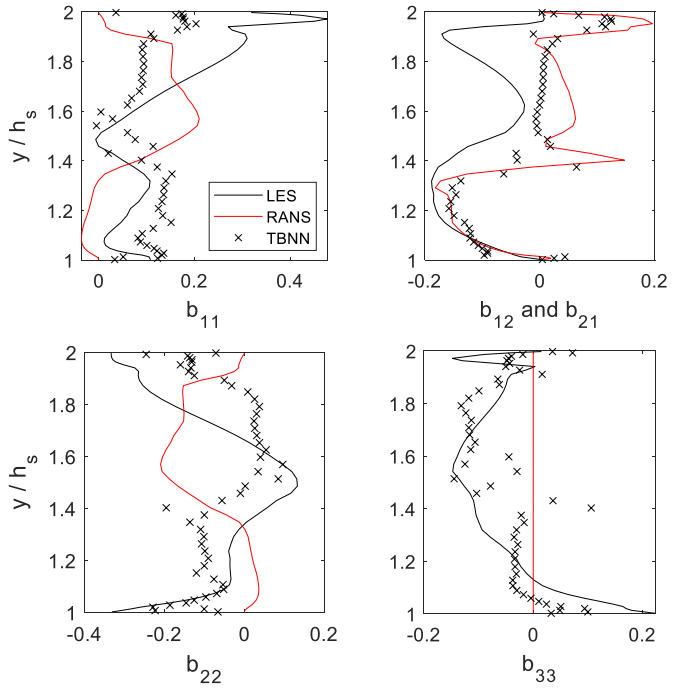


Figure 6 F-BFS anisotropy b_{ij} profile results at $x/h_s = 5$

CONCLUSION

An in-depth hyperparameter tuning study on TBNN predictions for anisotropy in channel flow has shown that the choice of CFD cases for validation and testing can have a significant influence on the optimal parameter set. Despite this, a TBNN architecture that used ELU activations predicted well in both validation and testing, outperforming ReLU. Running an ensemble of 25 TBNNs or more was shown to improve TBNN prediction accuracy of anisotropy components by 5%, thereby improving RANS results by 80% in the normal components and 57% in the b_{12} component. Some areas for improvement which are especially important in internal flows are predictions in regions of no velocity gradient and at the wall, where the typical TBNN predicts zero anisotropy, and the buffer layer where TBNN shows some discrepancy. The latter two issues were also reported in TBNN predictions for flow over a F-BFS.

ACKNOWLEDGEMENTS

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [grant numbers EP/T012242/1 and EP/T012242/2]. Data supporting this publication can be obtained on request. The authors would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility at The University of Manchester.

REFERENCES

[1] Wilcox D., Turbulence Modeling for CFD, DCW Industries, 2006.
[2] Versteeg H. and Malalasekera W., An Introduction to Computational Fluid Dynamics, 2nd ed., Pearson, 2007.
[3] Craft T., Launder B. and Suga, K., Development and application of a cubic eddy-viscosity model of turbulence, International Journal of Heat and Fluid Flow, 17(2), 1996, pp. 108-115.
[4] Leschziner M., Statistical Turbulence Modelling For Fluid Dynamics - Demystified: An Introductory Text For Graduate Engineering Students, 1st ed, Imperial College Press, 2016.
[5] Zhu L., Zhang W. and Kou J., Machine learning methods for turbulence modeling in subsonic flows around airfoils, Phys. of Fluids, Volume 31, 2019.
[6] Maulik R. et al., A turbulent eddy-viscosity surrogate modeling framework for Reynolds-averaged Navier-Stokes simulations, Computers and Fluids, vol. 227, 2021.
[7] Ling J., Kurzawski A. and Templeton J., Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. of Fluid Mech., vol. 807, pp. 155-166, 2016.
[8] Pope S., A more general effective-viscosity hypothesis, J. of Fluid Mech., vol. 72, pp. 331-340, 1975.
[9] Menter F., Zonal Two Equation k-ω Turbulence Models For Aerodynamic Flows. Orlando, FL, 24th Fluid Dynamics Conference, AIAA 93-2906, 1993.
[10] Moser R., Kim J. and Mansour N., Direct numerical simulation of turbulent channel flow up to Re=590, Phys. of fluids, 11(4), pp. 943-945, 1999.
[11] Hoyas, S. and Jiménez, J., Reynolds number effects on the Reynolds-stress budgets in turbulent channels, Phys. of Fluids, vol. 20, 2008.
[12] Barnard E. and Wessels L.F.A., Extrapolation and interpolation in neural network classifiers, IEEE Control Systems, vol. 12, Issue 5, 1992.
[13] Xu K. et al., How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks, arXiv:2009.11848, 2021.
[14] Szandała T., Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks, Bio-inspired Neurocomputing, vol. 903, 2021.
[15] Clevert D-A., Unterthiner T. and Hochreiter S., Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), arXiv:1511.07289, 2016.
[16] Fang et al., Deep learning for turbulent channel flow, arXiv:1812.02241, 2018.