SAND2016-7345J

Reynolds Averaged Turbulence Modeling using Deep Neural Networks with Embedded Invariance

Julia Ling* and Jeremy Templeton

Sandia National Laboratories

Thermal/Fluid Sciences and Engineering Department

Livermore, CA

Andrew Kurzawski

University of Texas at Austin

Mechanical Engineering Department

Austin, TX

(Dated: July 24, 2016)

Abstract

There exists significant demand for improved Reynolds Averaged Navier Stokes (RANS) turbulence models. This paper presents a method of using deep neural networks to learn a model for the Reynolds stress anisotropy tensor from high fidelity simulation data. A novel neural network architecture is proposed which uses a multiplicative layer with an invariant tensor basis to embed Galilean invariance into the predicted anisotropy tensor. It is demonstrated that this neural network architecture provides improved prediction accuracy compared to a generic neural network architecture which does not embed this invariance property. The Reynolds stress anisotropy predictions of this invariant neural network are propagated through to the velocity field for two test cases. For both test cases, significant improvement versus baseline RANS linear eddy viscosity and non-linear eddy viscosity models is demonstrated.

PACS numbers: 47.27.em, 47.27.eb, 02.20.-a

* jling@sandia.gov

I. INTRODUCTION

Reynolds Averaged Navier Stokes (RANS) models are widely used because of their computational tractability. Most two-equation RANS models rely on the Linear Eddy Viscosity Model (LEVM) for their Reynolds stress closure. The LEVM postulates a linear relationship between the Reynolds stresses and the mean strain rate. However, this model does not provide satisfactory predictive accuracy in many engineering-relevant flows, such as those with curvature, impingement, and separation [1, 2]. In particular, it has been demonstrated that the LEVM does not capture the correct Reynolds stress anisotropy in many flows, including simple shear flows [3]. More advanced non-linear eddy viscosity models have also been proposed [1, 4, 5] which rely on higher order products of the mean strain rate and mean rotation rate tensors. These non-linear models have not gained widespread usage because they do not give consistent performance improvement over the LEVM and often lead to worsened convergence and stability properties [6, 7].

More recently, there has been interest in using machine learning methods to develop Reynolds stress closures. Tracey et al. [3] used kernel regression to model the Reynolds stress anisotropy eigenvalues. This method was shown to have limited ability to generalize to new flows and to scale to large amounts of training data. Tracey et al. [8] later used neural networks with a single hidden layer to model the source terms from the Spalart Allmaras RANS model. These neural networks were shown capable of reconstructing these source terms, demonstrating the potential utility of neural networks for turbulence modeling. Zhang and Duraisamy [9] investigated applications of neural networks to predict a correction factor for the turbulent production term. Parish and Duraisamy [10] proposed the use of Gaussian processes to provide closure models for modeling terms inferred through Bayesian inversion. These Gaussian processes, however, are not easily scalable to larger data sets. Ling et al. [11] and Xiao et al. [12] proposed the use of Random Forests to predict the Reynolds stress anisotropy. However, Random Forests are limited in their ability to predict the full anisotropy tensor because they cannot easily enforce Galilean invariance for a tensor prediction.

Deep neural networks, also referred to as *deep learning*, are a branch of machine learning in which input features are transformed through multiple layers of non-linear interactions [13]. Deep neural networks have gained attention for their ability to represent complex interactions

and achieve superior results across a wide range of applications, including video classification [14], voice recognition [15], and playing the game of Go [16]. Despite the widespread success of neural networks at providing high quality predictions in complex problems, there have been only limited attempts to apply deep learning techniques to turbulence modeling. Zhang and Duraisamy [9] and Tracey et al. [3] used neural networks with only one or two hidden layers. Milano and Koumoutsakos [17] used neural networks with multiple hidden layers to replicate near wall channel flows, but did not build these neural networks into forward models for turbulence model prediction. This paper will demonstrate the utility of deep neural networks for providing improved Reynolds stress closures for RANS turbulence models.

A key strength of neural networks is in the flexibility of their architecture. While Random Forests cannot easily be restructured to preserve tensor invariance properties, neural networks can be. Ling et al. [18] used Random Forests and neural networks to predict the Reynolds stress anisotropy eigenvalues, and showed significant performance gains when a rotationally invariant input feature set was used. These results showed that embedding invariance properties into the machine learning model is critical for achieving high performance. In the current work, a specialized neural network architecture is proposed which embeds Galilean invariance into the neural network predictions. This neural network is able to predict not only the anisotropy eigenvalues, but the full anisotropy tensor while preserving Galilean invariance. The neural network model is trained and evaluated on a database of flows for which both RANS and high fidelity (Direct Numerical Simulation or well-resolved Large Eddy Simulation) data are available. The performance of this specialized neural network is compared to that of a more generic feed forward multi-layer perceptron.

Section II will introduce the concepts of deep neural networks and will present the architecture and training and validation procedures for the multi-layer perceptron neural network and the specialized invariant neural network. Section III will detail the data sets used for training, validation, and testing. Section IV will present the *a priori* results of the neural networks in predicting the Reynolds stress anisotropy as well as the *a posteriori* results of the velocity fields calculated when using these Reynolds stress closures. Finally, Section V will present conclusions and ideas for next steps.

II. DEEP NEURAL NETWORKS

Neural networks are composed of connected nodes. Each node takes in inputs $\vec{x}$, transforms the inputs through an activation function $f(\vec{x})$, and outputs the result, as shown in Fig. 1. Common activation functions include linear functions: $f(\vec{x}) = \vec{w}^T\vec{x}$, tanh functions: $f(\vec{x}) = tanh(\vec{w}^T\vec{x})$, and the rectified linear unit (ReLU): $f(\vec{x}) = \max(0, \vec{w}^T\vec{x})$. In these functions, $\vec{w}$ are node weights which are set during the training phase of the model. The leaky ReLU activation function is an adaptation of ReLU with a small negative slope for negative values of $\vec{w}^T\vec{x}$. The leaky ReLU was employed in the neural networks discussed in this paper because it enables efficient training of deep neural networks [19]. In deep neural networks, there are multiple layers of nodes, with the outputs of one layer becoming the inputs to the next layer. By using multiple layers of transformations, deep neural networks are able to capture complex, hierarchical interactions between features. The layers between the input layer and the output layer are called *hidden layers* because the physical interpretation of their activation is not always clear.

The inputs to the neural networks were based on the mean strain rate tensor $\mathbf{S}$ and the mean rotation rate tensor $\mathbf{R}$, non-dimensionalized using the turbulent kinetic energy k and the turbulent dissipation rate ϵ as suggested by Pope [5]. The output was the normalized Reynolds stress anisotropy tensor, $\mathbf{b}$, where $b_{ij} = \frac{\overline{u_i'u_j'}}{2k} - \frac{1}{3}\delta_{ij}$.

The neural networks were trained using back propagation with gradient descent. The process of training a neural network is analogous to calibrating a regression model. In training, the model is fit to a subset of the data, known as the *training data*, by iteratively adjusting the node weights to provide the lowest mean squared error between the predicted and true values of the anisotropy tensor. Back-propagation is a method of efficiently calculating the derivative of the mean squared error with respect to node weights in each layer of the neural network. Once these gradients are calculated, the node weights are updated using gradient descent. For the multi-layer perceptron (MLP), mini-batch gradient descent was employed, in which the gradient-based updates were averaged over a random subset of the training data to reduce noise. For the Tensor Basis Neural Network (TBNN), stochastic gradient descent was used, in which the weights were updated after each training point. This technique was used to facilitate the implementation of a multiplicative layer, as detailed in the following sections.

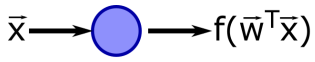


FIG. 1. Schematic of neural network node.

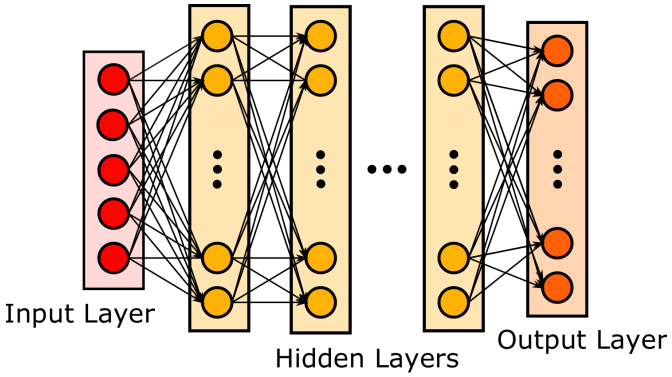


FIG. 2. Schematic of Multi-Layer Perceptron.

For both neural network architectures, there were three main hyper-parameters that had to be fit: the number of hidden layers, the number of nodes per hidden layer, and the learning rate in the gradient descent algorithm. All three of these parameters can have a significant effect on model performance. Larger networks (with more hidden layers and more nodes per layer) can fit more complex data, but are also more prone to over-fitting the data. These three hyper-parameters were determined through a Bayesian optimization process that sampled the parameter space efficiently to find the parameters that yielded the best performance. The neural network performance was evaluated on a validation data set that was different from both the training data and the test data. The Spearmint python package [20] was used to implement the Bayesian optimization process.

A. Multi-Layer Perceptron

An MLP is the most basic type of neural network: it is a densely connected feed-forward neural network. Feed-forward means that there are no loops in the connections between the nodes. Densely connected means that each node in a given layer is connected to all the nodes in the next layer. Fig. 2 shows a schematic of an MLP.

The inputs to the MLP were the 9 distinct components of the non-dimensionalized strain

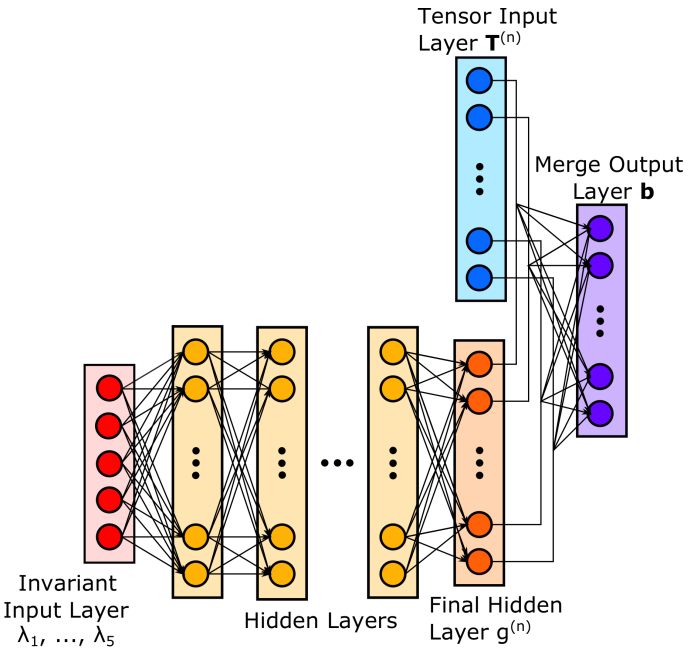


FIG. 3. Schematic of Tensor Basis Neural Network.

rate and rotation rate tensors, $\mathbf{S}$ and $\mathbf{R}$, at a given point in the flow as predicted by RANS. Through Bayesian optimization, the network was set to have 10 hidden layers, each with 10 nodes, and the learning rate was 2.5e-6.

B. Tensor Basis Neural Network

A special network architecture, which will be referred to as the Tensor Basis Neural Network (TBNN), is proposed. This network architecture, shown in Fig. 3, embeds rotational invariance by enforcing that the predicted anisotropy tensor lies on a basis of isotropic tensors. Rotational invariance signifies that the physics of the fluid flow do not depend on the orientation of the coordinate frame of the observer. This is a fundamental physical principle, and it is important that any turbulence closure obeys it. Otherwise, the machine learning model evaluated on identical flows with the axes defined in different directions could yield different predictions.

If the goal were just to predict the eigenvalues of the anisotropy tensor, then a basis of scalar invariants could be used as the input features to ensure the rotational invariance of the predicted eigenvalues, as was shown by Ling et al. [18]. However, to predict the

full anisotropy tensor, this approach is not sufficient. Unlike the anisotropy eigenvalues, the full anisotropy tensor will change when the coordinate axes are rotated. The key to ensuring Galilean invariance is therefore to ensure that when the coordinate frame is rotated, the anisotropy tensor is also rotated by the same angles. Therefore, if the input to the neural network is a rotated velocity gradient tensor, the output should be the corresponding rotated anisotropy tensor. This can be achieved by constructing an integrity basis of the input tensors. Further explanation of integrity bases for isotropic functions can be found in Ref. [21].

For the specific case of interest here, with input tensors $\mathbf{S}$ and $\mathbf{R}$, Pope [5] has previously derived the relevant integrity basis. Pope proved that in the most general case, an eddy viscosity model for incompressible flow that is a function of only $\mathbf{S}$ and $\mathbf{R}$ can be expressed as a linear combination of 10 isotropic basis tensors:

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

Any tensor $\mathbf{b}$ which satisfies this condition will automatically satisfy Galilean invariance. There are only a finite number of tensors because by the Caley-Hamilton theory, higher order products of these two tensors can be reduced to a linear combination of this tensor basis. The 5 tensor invariants $\lambda_1, ..., \lambda_5$ are known scalar functions of the elements of $\mathbf{S}$ and $\mathbf{R}$. Pope [5] gave a detailed derivation of these 10 tensors, $\mathbf{T}^{(1)}, ..., \mathbf{T}^{(10)}$ and 5 invariants $\lambda_1, ..., \lambda_5$, which are listed below:

$$\begin{aligned}
\mathbf{T^{(1)}} &= \mathbf{S} & \mathbf{T^{(6)}} &= \mathbf{R}^2\mathbf{S} + \mathbf{S}\mathbf{R}^2 - \frac{2}{3}\mathbf{I} \cdot Tr(\mathbf{S}\mathbf{R}^2) \\
\mathbf{T^{(2)}} &= \mathbf{S}\mathbf{R} - \mathbf{R}\mathbf{S} & \mathbf{T^{(7)}} &= \mathbf{R}\mathbf{S}\mathbf{R}^2 - \mathbf{R}^2\mathbf{S}\mathbf{R} \\
\mathbf{T^{(3)}} &= \mathbf{S}^2 - \frac{1}{3}\mathbf{I} \cdot Tr(\mathbf{S}^2) & \mathbf{T^{(8)}} &= \mathbf{S}\mathbf{R}\mathbf{S}^2 - \mathbf{S}^2\mathbf{R}\mathbf{S} \\
\mathbf{T^{(4)}} &= \mathbf{R}^2 - \frac{1}{3}\mathbf{I} \cdot Tr(\mathbf{R}^2) & \mathbf{T^{(9)}} &= \mathbf{R}^2\mathbf{S}^2 + \mathbf{S}^2\mathbf{R}^2 - \frac{2}{3}\mathbf{I} \cdot Tr(\mathbf{S}^2\mathbf{R}^2) \\
\mathbf{T^{(5)}} &= \mathbf{R}\mathbf{S}^2 - \mathbf{S}^2\mathbf{R} & \mathbf{T^{(10)}} &= \mathbf{R}\mathbf{S}^2\mathbf{R}^2 - \mathbf{R}^2\mathbf{S}^2\mathbf{R}
\end{aligned} \tag{2}$$

$$\lambda_1 = Tr(\mathbf{S}^2), \quad \lambda_2 = Tr(\mathbf{R}^2), \quad \lambda_3 = Tr(\mathbf{S}^3)$$
$$\lambda_4 = Tr(\mathbf{R}^2\mathbf{S}), \quad \lambda_5 = Tr(\mathbf{R}^2\mathbf{S}^2)$$

The goal of the TBNN, then, is to determine the scalar coefficients $g^{(n)}(\lambda_1, ..., \lambda_5)$ in Eq. 1. Once these coefficients are determined, then Eq. 1 can be used to solve for the anisotropy tensor $\mathbf{b}$.

As Fig. 3 shows, the network architecture was designed to match Eq. 1. There are two input layers: an Invariant Input Layer and a Tensor Input Layer. The Invariant Input Layer takes in the 5 invariants $\lambda_1, ..., \lambda_5$, and is followed by a series of hidden layers. The Final Hidden Layer has 10 elements and represents $g^{(n)}$ for $n = 1, ..., 10$. The output from these 10 nodes is then multiplied by the Tensor Input Layer, which also has 10 nodes. These 10 nodes represent the 10 tensors $\mathbf{T}^{(n)}$ for $n = 1, ..., 10$. The Merge Output Layer performs element-wise multiplication between the outputs of the Final Hidden Layer and the Tensor Input Layer and then sums the result to give the final prediction for $\mathbf{b}$. This innovative architecture ensures that Eq. 1 is satisfied, thereby guaranteeing the Galilean invariance of the network predictions.

Because of this final multiplicative layer, stochastic gradient descent was used during network training. While noisier, this method was more convenient to implement given the added complexity of back-propagation through a multiplicative layer. Through Bayesian optimization, the number of hidden layers was set to 8, with 30 nodes per hidden layer. The learning rate was 2.5e-7.

III. DATA SETS

The neural networks were trained, validated, and tested on a database of 9 flows for which both high fidelity (DNS or well-resolved LES) as well as RANS results were available. The RANS data, obtained using the $k - \epsilon$ model with a Linear Eddy Viscosity Model for the Reynolds stresses, were used as the inputs to the neural networks. The high fidelity data were used to provide the truth labels for the Reynolds stress anisotropy during model training and evaluation.

The nine flows in the database were selected because of the availability of high fidelity data and because they represent canonical flow cases. All of the high fidelity simulation results have been previously reported in the literature. 6 cases were used for training: Duct flow at $Re_b = 3500$ [22], channel flow at $Re_\tau = 590$ [23], a perpendicular jet in crossflow [24], an inclined jet in crossflow [25], flow around a square cylinder [26, 27], and flow through

a converging-diverging channel [28]. One case was used for validation when selecting the neural network hyper-parameters through Bayesian optimization: a wall-mounted cube in crossflow at bulk Reynolds number of 5000 [29]. Two cases were used for testing: duct flow at $Re_b = 2000$ and flow over a wavy wall at $Re = 6850$. These cases were chosen for test sets because they are both flows in which Reynolds stress anisotropy plays a key role.

It is notable that this flow database represents a wide variety of different flow configurations. We are therefore testing the ability of the neural network to do more than just interpolate between similar flow configurations at different Reynolds numbers; we are evaluating the ability of the neural network to learn about the underlying flow regimes present in these configurations.

IV. RESULTS

A. *A Priori* Results

The MLP and TBNN were used to predict the normalized Reynolds stress anisotropy tensor $\mathbf{b}$ for two different test flow cases: turbulent duct flow and flow over a wavy wall. Figures 4 shows the anisotropy predictions for the duct flow case. For comparison, the anisotropy predictions from two different baseline RANS simulations are also shown. The first RANS was run using the default Linear Eddy Viscosity Model (LEVM): $\mathbf{b} = \frac{-\nu_t \mathbf{S}}{k}$. The second RANS used a Quadratic Eddy Viscosity Model (QEVM) based on Craft's non-linear eddy viscosity model [1]:

$$\begin{aligned}\mathbf{b}_{ij} = \; & -\frac{\nu_t \mathbf{S}_{ij}}{k} + C_1 \frac{\nu_t}{\tilde{\epsilon}}(2\mathbf{S}_{ik}\mathbf{S}_{kj} - \frac{2}{3}\mathbf{S}_{kl}\mathbf{S}_{kl}\delta_{ij}) \\ & + C_2 \frac{\nu_t}{\tilde{\epsilon}}(2\mathbf{R}_{ik}\mathbf{S}_{kj} + 2\mathbf{R}_{jk}\mathbf{S}_{ki}) + C_3 \frac{\nu_t}{\tilde{\epsilon}}(2\mathbf{R}_{ik}\mathbf{R}_{jk} - \frac{2}{3}\mathbf{R}_{kl}\mathbf{R}_{kl}\delta_{ij})\end{aligned} \tag{3}$$

In this model, the coefficients are set as $C_1 = -0.1$, $C_2 = 0.1$, and $C_3 = 0.26$.

As Fig. 4 shows, the baseline LEVM completely failed to predict the anisotropy values. It predicted zero anisotropy for $\mathbf{b}_{11}$, $\mathbf{b}_{22}$, $\mathbf{b}_{33}$, and $\mathbf{b}_{12}$. This is due to the fact that the LEVM model does not predict any secondary flows in the duct, so the mean velocity in the x_1 and x_2 directions is zero, causing $\mathbf{S}_{11}$, $\mathbf{S}_{22}$, and $\mathbf{S}_{12}$ to be zero. Since the flow is periodic in the x_3 direction, $\mathbf{S}_{33}$ is also zero. As a result, the corresponding $\mathbf{b}$ components are zero, as shown. The QEVM does slightly better, predicting the right sign, though still too small

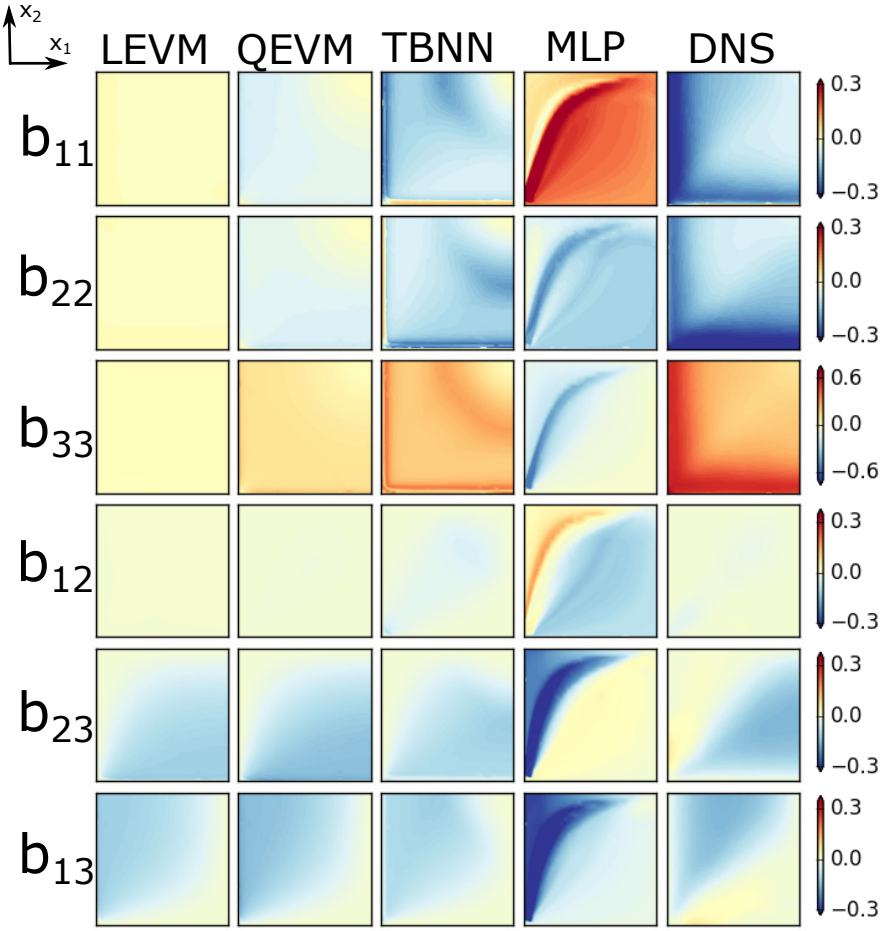


FIG. 4. Predictions of Reynolds stress anisotropy tensor $\mathbf{b}$ on the duct flow test case. Only the lower left quadrant of the duct is shown, and the streamwise flow direction is out of the page. The columns show the predictions of the LEVM, QEVM, TBNN, and MLP models. The true DNS anisotropy values are shown in the right-most column for comparison.

a magnitude, for these components of $\mathbf{b}$. The TBNN does even better, getting closer to the correct magnitude for the components of $\mathbf{b}$. The MLP, on the other hand, completely fails to predict the anisotropy in this configuration. Its predictions are both qualitatively and quantitatively inaccurate.

A more quantitative comparison of these model performances is given by Table I, which provides the root mean squared error (RMSE) for these four models with respect to the DNS

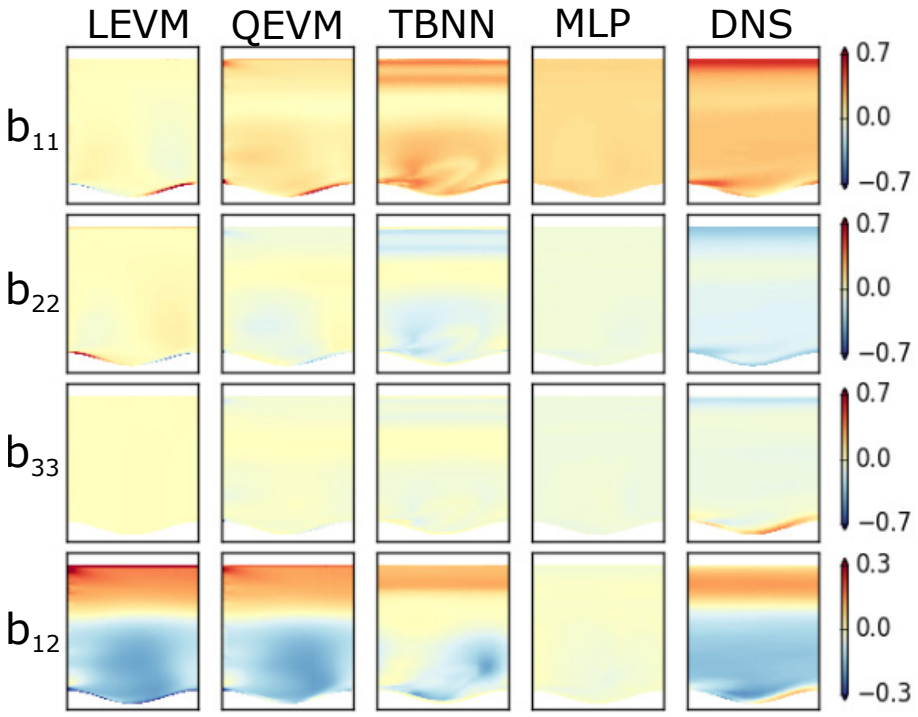


FIG. 5. Predictions of Reynolds stress anisotropy **b** tensor on the wavy wall test case. The columns show the predictions of the LEVM, QEVM, TBNN, and MLP models. The true DNS anisotropy values are shown in the right-most column for comparison.

data. This table confirms that the TBNN provides by far the most accurate predictions in this case: 43% more accurate than the LEVM and 28% more accurate than the QEVM.

Figure 5 shows the anisotropy predictions on the wavy wall test case. $\mathbf{b}_{13}$ and $\mathbf{b}_{23}$ are not shown because they are identically zero in this 2-D test case. Once again, the LEVM is unable to predict even the qualitative trends in the anisotropy components, and the QEVM is only slightly better, predicting the correct sign for the on-diagonal components of **b**. The

TABLE I. Root Mean Squared Error on Test Cases

Model	Duct Flow	Flow over Wavy Wall
LEVM	0.23	0.18
QEVM	0.18	0.11
TBNN	0.13	0.08
MLP	0.33	0.09

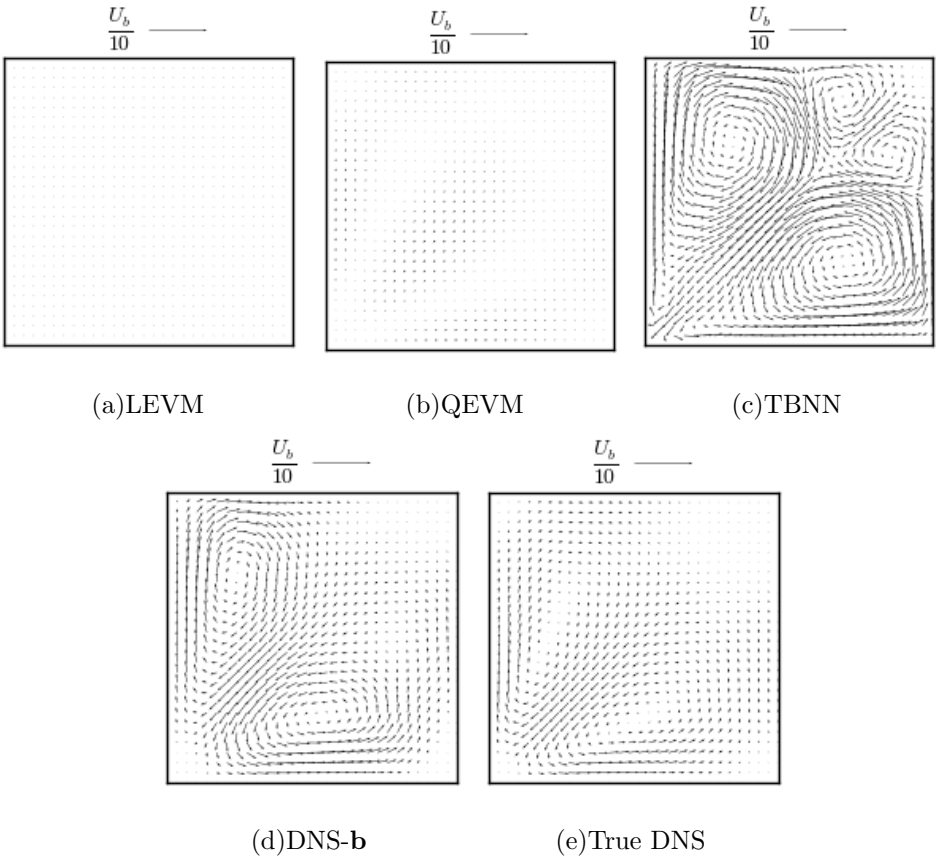
TBNN predicts closer to the correct magnitude for the on-diagonal components of $\mathbf{b}$. For the $\mathbf{b}_{12}$ component, it predicts too low of anisotropy near the center of the channel. Once again, the MLP is unable to make satisfactory predictions for the anisotropy in this flow.

Table I lists the RMSE values for each of the four models in the wavy wall test case. Once again, the TBNN provides the predictions with the lowest error, giving a 56% reduction in error with respect to LEVM and a 27% reduction in error with respect to QEVM. In this case, the MLP surpasses LEVM and QEVM, but still falls short of the TBNN in accuracy. Therefore, on both test cases, the TBNN has been shown to provide substantial reductions in error with respect to the baseline RANS models. The TBNN has also been shown to be consistently more accurate than the MLP, emphasizing the performance benefits of embedding invariance into the network architecture.

B. *A Posteriori* Results

Given the significantly improved Reynolds stress anisotropy predictions of the TBNN, it was of interest to determine whether these improved anisotropy values would translate to improved mean velocity predictions. Therefore, the Reynolds stress anisotropy tensor predicted by the TBNN was implemented in an in-house RANS solver, SIERRA Fuego [30], in the momentum equations and in the turbulent kinetic energy production term. Notably, the Reynolds stress anisotropy is not the only source of uncertainty in the RANS equations, which also relies on approximate transport equations for k and ϵ. Therefore, the true DNS anisotropy values were also implemented in the RANS solver as a point of comparison. This DNS anisotropy (DNS-$\mathbf{b}$) model shows the flow predictions that would be achieved by using the correct normalized anisotropy tensor $\mathbf{b}$, given the other RANS model assumptions. It therefore represents the upper performance limit of an improved Reynolds stress anisotropy model.

In the duct flow test case, one of the major failings of LEVM is that it cannot predict the corner vortices that are known to form in this flow. Figure 6 shows the secondary flow predictions made by the LEVM, QEVM, TBNN, DNS-$\mathbf{b}$, and the true DNS. As this figure shows, the LEVM does not predict any secondary flows at all. The QEVM predicts corner vortices, but significantly under-predicts their strength. The TBNN, on the other hand, over-predicts the strength of these vortices, with incorrect counter-rotating vortices forming



(a)LEVM (b)QEVM (c)TBNN

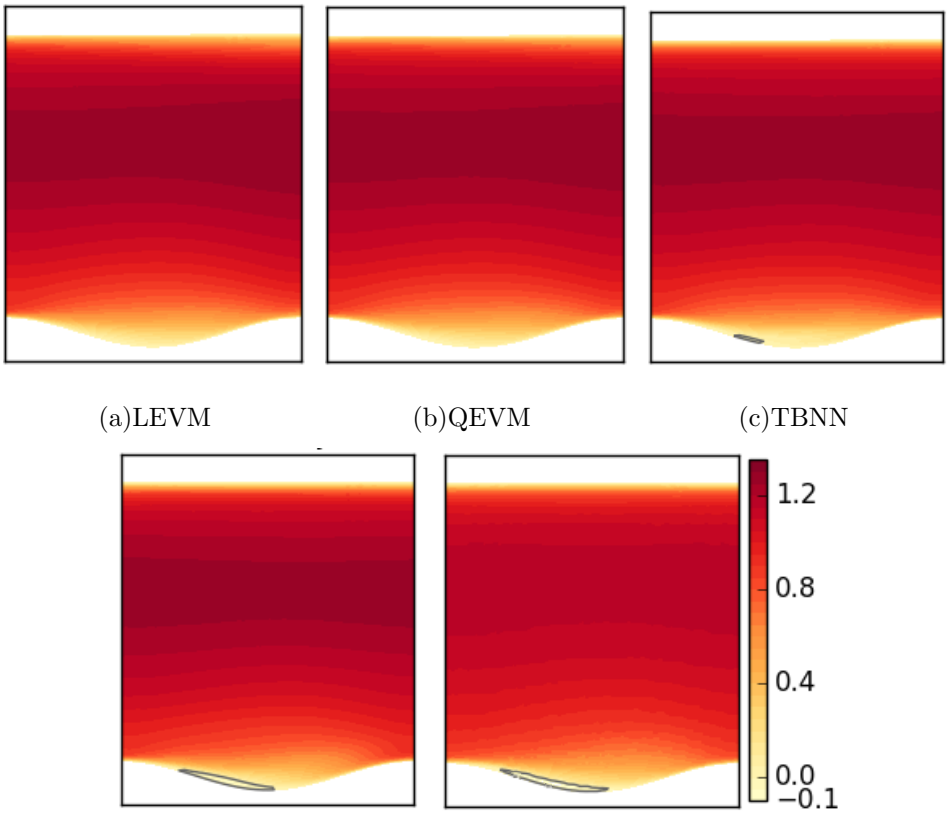
(d)DNS-**b** (e)True DNS

FIG. 6. Plot of secondary flows in duct flow case. Reference arrows of length $U_b/10$ shown at the top of each plot.

near the center of the channel. The DNS-**b** slightly over-predicts the strength of the corner vortices, but is overall the most accurate, unsurprisingly. In this test case, then, while the TBNN provides improved secondary flow predictions, it is still not able to correctly capture the strength and shape of the corner vortices.

In the case of periodic flow over a wavy wall, the DNS shows flow separation on the leeward side of the bump. RANS struggles to correctly capture this separation region. Figure 7 shows contours of streamwise velocity for this flow case. The RANS LEVM and QEVM both fail to predict flow separation downstream of the bump. The DNS-**b** correctly predicts the size and shape of the separated region. TBNN predicts flow separation, though in a smaller region than the DNS.

In both test cases, the TBNN improvements in predicting the anisotropy tensor yield more accurate velocity predictions in comparison to RANS LEVM and QEVM. On the



(a)LEVM (b)QEVM (c)TBNN

(d)DNS-**b** (e)True DNS

FIG. 7. Contours of streamwise velocity. Separated regions outlined in black.

other hand, there is still room for improved RANS predictions, as is shown by the DNS-**b** results.

V. CONCLUSIONS

A deep learning approach to RANS turbulence modeling was presented which embedded Galilean invariance into the network architecture using a higher order multiplicative layer. This architecture used concepts from representation theory to ensure that the predicted anisotropy tensor lay on a invariant tensor basis. This invariant Tensor Basis Neural Network was shown to have significantly more accurate predictions than a generic multi-layer perceptron that did not have any embedded invariance properties.

The accuracy of the TBNN was explored in both an *a priori* and an *a posteriori* sense. The Reynolds stress anisotropy predictions were compared to those of the default LEVM

RANS model, as well as a non-linear eddy viscosity model. The TBNN anisotropy predictions were shown to be significantly more accurate than either of these two conventional RANS models on two different test cases. Notably, both of the test cases represented challenges for the TBNN. While there was also a duct flow test case in the training set, the test case was at a significantly different Reynolds number, and therefore had distinctly different anisotropy and secondary flows. The wavy wall test case was a completely different geometry than any of the training cases. The fact that the TBNN was able to provide improved predictions in this test case demonstrates that the TBNN learned about the underlying flow regimes and has the capability to extrapolate to new flow cases.

The *a posteriori* evaluation showed that the TBNN was able to predict corner vortices in the duct flow case and flow separation in the wavy wall case, two key flow features that the baseline LEVM model completely failed to predict. While the TBNN was not able to perfectly reproduce the DNS results, it represents a significant step towards improved RANS predictive accuracy. In order for this innovative approach to reach its full potential, the TBNN needs to be trained and tested across a much broader set of a flows. Training across more flows will improve the neural network accuracy, and testing across more flows will enable uncertainty analysis for the TBNN predictions.

ACKNOWLEDGMENTS

The authors wish to thank V. Brunini for his valuable support in the code implementation stage of this research. Funding for this work was provided by the Sandia LDRD program, and its support is gratefully acknowledged. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

---

[1] T. Craft, B. Launder, and K. Suga, International Journal of Heat and Fluid Flow **17**, 108 (1996).

[2] S. Pope, "Turbulent flows," (Cambridge University Press, 2000).

[3] B. Tracey, K. Duraisamy, and J. Alonso, AIAA Aerospace Sciences Meeting , 2013 (2013).

[4] S. Wallin and A. Johansson, Journal of Fluid Mechanics **403**, 89 (2000).

[5] S. Pope, Journal of Fluid Mechanics **72**, 331 (1975).

[6] L. Belhoucine, M. Deville, A. Elazehari, and M. Bensalah, Computers and Fluids **33**, 179 (2004).

[7] T. Gatski and C. Speziale, Journal of Fluid Mechanics **254**, 59 (1993).

[8] B. Tracey, K. Duraisamy, and J. Alonso, AIAA SciTech , 2015 (2015).

[9] Z. Zhang and K. Duraisamy, AIAA Aviation , 2015 (2015).

[10] E. Parish and K. Duraisamy, Journal of computational physics **305**, 758 (2016).

[11] J. Ling, A. Ruiz, G. Lacaze, and J. Oefelein, ASME Turbo Expo 2016 (2016).

[12] J. Wang, J. Wu, and H. Xiao, arXiv preprint arXiv:1606.07987 (2016).

[13] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition , 1725 (2014).

[15] G. Hinton *et al.*, IEEE Signal Processing Magazine **29**, 82 (2012).

[16] D. Silver *et al.*, Nature **529**, 484 (2016).

[17] M. Milano and P. Koumoutsakos, Journal of Computational Physics **182**, 1 (2002).

[18] J. Ling, R. Jones, and J. Templeton, Journal of Computational Physics **318**, 22 (2016).

[19] A. Maas, A. Hannun, and A. Ng, Proceedings of ICML **30**, 1 (2013).

[20] J. Snoek, H. Larochelle, and R. Adams, Advances in Neural Information Processing Systems , 2951 (2012).

[21] G. Smith, Archive for rational mechanics and analysis **18**, 282 (1965).

[22] A. Pinelli, M. Uhlmann, A. Sekimoto, and G. Kawahara, Journal of Fluid Mechanics **644**, 107 (2010).

[23] R. Moser, J. Kim, and N. Mansour, Physics of Fluids **11**, 943 (1999).

[24] A. Ruiz, J. Oefelein, and G. Lacaze, Physics of Fluids **27**, 045101 (2015).

[25] J. Ling, K. Ryan, J. Bodart, and J. Eaton, in *ASME Turbo Expo 2015* (ASME Turbo Expo, Montreal, Canada, 2015).

[26] J. Ray, S. Lefantzi, S. Arunajatesan, and L. Dechant, AIAA Paper , 2085 (2014).

[27] S. Lefantzi, J. Ray, S. Arunajatesan, and L. Dechant, Sandia Technical Report , SAND2015 (2015).

[28] M. Marquillie, U. Ehrenstein, and J. Laval, Journal of Fluid Mechanics **681**, 205 (2011).

[29] R. Rossi, D. Philips, and G. Iaccarino, International Journal of Heat and Fluid Flow **31**, 805 (2010).

[30] S. Domino, C. Moen, S. Burns, and G. Evans, AIAA Aerospace Sciences Meeting (2003), AIAA-2003-149.