14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11-15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

INVESTIGATING MODEL-DATA INCONSISTENCY IN DATA-INFORMED TURBULENCE CLOSURE TERMS

Marius Kurz[1] **and Andrea D. Beck**[2]

[1] Institute of Aerodynamics and Gas Dynamics, University of Stuttgart
Pfaffenwaldring 21, 70569 Stuttgart, Germany
marius.kurz@iag.uni-stuttgart.de

[2] Lab. of Fluid Dynamics and Technical Flows, University of Magdeburg "Otto von Guericke"
Universitätsplatz 2, 39106 Magdeburg, Germany
mail@beck.aero

Key words: Turbulence Models, LES, Artificial Neural Networks

Abstract. *In the present work, we investigate the stability of turbulence closure predictions from neural network models and highlight the role of model-data-inconsistency during inference. We quantify this inconsistency by applying the Mahalanobis distance and demonstrate that the instability of the model predictions in practical large eddy simulations (LES) correlates with a deviation of the input data between the training dataset and actual simulation data. Moreover, the method of "stability training" is applied to increase the robustness of recurrent artificial neural networks (ANN) against small perturbations in the input, which are typically unavoidable in any practical scenario. We show that this method can increase the stability of simulations with ANN-based closure term predictions significantly. The models also achieve good accuracy on the blind testing set in comparison to the baseline model trained without stability training. The work presented here can thus be seen as a building block towards long-term stable data-driven models for dynamical systems and highlights methods to detect and counter model-data-inconsistencies.*

1 INTRODUCTION

Most flows in engineering and nature exhibit turbulent behavior. Accurate predictions of such flows are crucial in order to increase the reliability of weather forecasts or to reduce fuel consumption in aircraft transportation. Due to the multiscale character of turbulence, direct numerical simulation (DNS) is generally intractable for turbulent flows at medium to high Reynolds numbers. Instead, reduced order models like large eddy simulation (LES) or Reynolds-averaged Navier-Stokes (RANS) methods are used, which only resolve the largest turbulent scales or the temporal mean quantities, respectively. Due to the nonlinearity of the governing equations, these approaches introduce additional closure terms into the surrogate formulations, which describe the imprint of the fine-scale turbulent dynamics on the resolved scales. These closure terms are generally unknown for practical simulations and are thus approximated by adequate turbulence models. Despite decades of research, no *universally* accurate and overall *best* closure model has been identified to date.

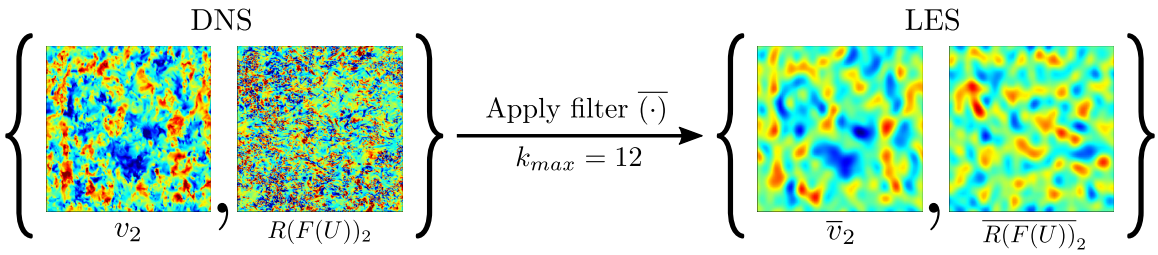

Figure 1: Slices of the DHIT test case. The coarse-grid quantities (right) are obtained by applying a Fourier cutoff filter to the DNS results (left). The filtered flow field thus contains only the first $k_{max} = 12$ Fourier modes of the underlying DNS solution.

Machine learning methods have gained significant popularity for turbulence modeling tasks in the last years since they allow either to recover the closure terms solely from data and without any prior assumptions or to fit appropriate models directly [2]. However, data-driven models have shown to be prone to instabilities in practical simulations [1, 12] due to the inherent data inconsistency between the offline training and the online inference in practical LES. Several approaches have been applied to increase the robustness of data-driven turbulence models in simulations. Rasp et al. [13] increased the stability of a machine learning closure model by performing additional training steps with the pre-trained ANN during a simulation to correct for this inconsistency. Other methods ensure the stability of the surrogate model by truncating its predictions [10] or projecting them onto a stable basis [1].

In this work, stability training [14] is employed to increase the stability of an artificial neural network (ANN) as data-driven model for the LES closure terms [7]. This approach can be incorporated easily into existing machine learning pipelines and allows to balance the accuracy in the offline training with the stability during online inference. We show that this approach increases the model robustness in inference for our application, while the loss in accuracy remains reasonable. Moreover, we quantify the deviation of the data during training and inference by the Mahalanobis distance measure. This demonstrates that the instability during inference is correlated with the shifted statistics between the inference data and the training dataset.

The outline of this paper is as follows: Section 2 shortly introduces the task of predicting the closure terms and the training dataset. Moreover, the stability training method and the employed ANN architecture are discussed here. Results of a priori and a posteriori analysis of the trained ANN are given in Section 3. Section 4 concludes the paper.

2 METHODS

2.1 The machine learning framework

In order to derive data-driven LES closure prediction models with supervised learning, the *exact* closure terms have to be defined first. To this end, we use the framework of *optimal LES* by Langford and Moser [8] to derive the exact LES closure terms in a consistent manner. The governing Navier-Stokes equations can be written in short notation as

$$U_t + R(F(U)) = 0\,, \tag{1}$$

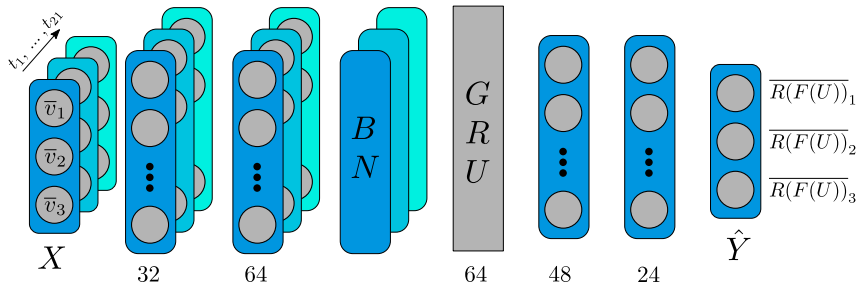

Figure 2: Architecture of the used ANN with batch normalization (BN) [4] before the gated recurrent unit (GRU) [3]. All other layers are fully connected layers with rectified linear units (ReLU) as the activation functions. The number of neurons in each layer is specified below the respective layer. The layers before the GRU are executed individually for each of the 21 time instants in the input sequence of a training sample X. The GRU works in a many-to-one prediction mode yielding only a single prediction in the last time step of the sequence.

where $R()$ denotes the divergence operator applied to the non-linear fluxes $F(U)$ and U denotes the solution vector. Instead of solving the full equation, the LES method resolves only the large scales in the solution. This can be interpreted as applying a low-pass filter $\overline{(\cdot)}$ to Eq. (1), which yields

$$\overline{U}_t + \overline{R(F(U))} = 0\,. \tag{2}$$

However, the second term $\overline{R(F(U))}$ is unknown since it depends on the full solution U. To this end, the coarse-scale solution $\overline{U}$ is usually advanced in time using some numerical discretization $\tilde{R}()$ applied to the coarse scale solution field. This yields the LES formulation

$$\overline{U}_t + \tilde{R}(\overline{U}) = \underbrace{\tilde{R}(\overline{U}) - \overline{R(F(U))}}_{\text{perfect LES closure}}\,. \tag{3}$$

The left-hand side of Eq. (3) depends only on known coarse-scale quantities, while the right-hand side exhibits the perfect LES closure term, which depends on the unknown full-scale solution U. While the perfect closure terms are generally unknown during LES, Eq. (3) provides the framework to compute them in a consistent manner if the DNS solution is known.

In this work, we use the dataset obtained in [1, 7] as training set for the ANN. This dataset is based on the decaying homogeneous isotropic turbulence (DHIT) test case and consists of the DNS solutions for several different flow realizations of the same turbulent statistics. A subset of 10 flow realizations is used in this work. Performing a Fourier transform of the DNS and applying a cutoff filter with $k_{max} = 12$ then yields the filtered LES velocity field $\overline{v}_{j=1,2,3}$ and the corresponding unknown part of the closure terms $\overline{R(U)}_{i=1,2,3}$, as shown in Figure 1. The dataset contains almost 30 million training samples and a single run is kept hidden for blind testing of the trained ANN. The reader is referred to [7] for a more detailed discussion of the training data. The ANN is then trained to recover the unknown closure terms from a time series of the filtered velocity field and thus to approximate the functional relationship

$$\overline{R(U)}_{i=1,2,3} \approx f\left(\overline{v}_{j=1,2,3}\right)\,. \tag{4}$$

In this work, we use a recurrent neural network architecture with gated recurrent units (GRU) [3] as shown in Figure 2. This ANN is comprised of 31,669 trainable parameters in total. Each training sample X consists of the pointwise coarse-scale velocity vectors $\overline{v}_{j=1,2,3}$ at 21 time instants with a constant

time step of $\Delta t = 1 \cdot 10^{-4}$ with respect to unit length and velocity. The GRU is used in a many-to-one prediction mode and gives only a single prediction in the last time step of the sequence. The dimensions of the input and output quantities are thus $X \in \mathbb{R}^{3\times 21}$ and $Y \in \mathbb{R}^3$, respectively.

2.2 Model-data-inconsistency

In order to define the issue of *model-data-inconsistency* in the context of turbulence modeling, we first establish three different types of uncertainties and errors which are present in machine learning tasks.

- **Data uncertainty:** This comprises all experimental and numerical errors in the training data, which are introduced either in the gathering or the processing of the training data. Moreover, the stochastic nature of the underlying functional relationship between the input and output quantities is included here. Turbulence modeling is an example for such a stochastic relationship since the mapping between the coarse-grid flow field and the corresponding closure terms is generally not unique [11]. This property results from the information loss induced by the filtering operation. By applying the filter function, multiple DNS solutions with identical large-scale but different small-scale characteristics are mapped to the same coarse-scale representation. However, the filtered non-linear fluxes generally differ for these flows fields, which results in an inherent many-to-one mapping. For a more thorough discussion of this aspect, see [11].
- **Interpolation errors:** This type of errors is summarized by the model's ability to generalize to unseen validation and testing data drawn from the same distribution as the training data. This type of error can thus be assessed by applying the ANN on unseen testing data. A common type of such errors are *overfitting* errors.
- **Extrapolation errors:** Under this point we summarize all errors which arise when trained ANN are applied to input data which does not match the distribution of the training data. Without the support of the training data, the prediction of the model depends heavily on extrapolation and the stochastic components of the training procedure and is generally not reliable. Such errors can usually not be prevented other than augmenting the training dataset with additional data points and retraining, or the application of some sort of feedback mechanism as for example in reinforcement learning.

The data uncertainty is inherent in the modeling task and the process of data acquisition. It thus poses an upper limit to the accuracy with which the ANN can learn the functional relationship in the data. The interpolation errors determine how well the ANN can generalize to unseen data drawn from the same distribution. For machine learning tasks, this is typically examined by evaluating the ANN on unseen validation and test data. Extrapolation errors however can generally not be prevented and typically stem from the systematic shift in data statistics between training and inference. This inconsistency is inherent in many supervised machine learning frameworks for turbulence modeling, especially if discretization effects are not considered during training, e.g. [1, 10]. This is what we refer to as *model-data-inconsistency* in this work.

In the following, we present a simple approach to check for model-data-consistency and discuss a general method of lessening its effects. Inspired by [9], the Mahalanobis distance measure is used to quantify the inconsistency between the training dataset and inference data. Moreover, the stability training approach by Zheng et al. [14] is used to stabilize the model against small perturbations in the input data. While

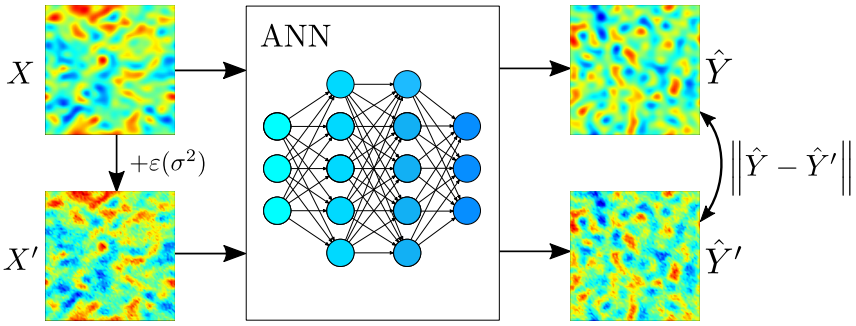

Figure 3: In the stability training method, the ANN is evaluated in each training step on the training sample X as well as the noisy sample $X' = X + \varepsilon(\sigma^2)$. The distance between the predictions $\left\|\hat{Y} - \hat{Y}'\right\|$ is then a measure of the ANN's sensibility to input perturbations and is used as an additional optimization constraint during training.

this does not solve the underlying data mismatch between training and inference, the proposed approach alleviates the effects of extrapolation and improves the models' stability in simulations.

2.3 Mahalanobis distance

The Mahalanobis distance $\mathcal{D}$ is a measure of distance between a data point x and a multivariate distribution $P(\mu,\Sigma)$ characterized by its mean μ and the covariance matrix Σ:

$$\mathcal{D} = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)} \,. \tag{5}$$

As proposed in [9], this measure can be used to assess whether a new data point lies inside the hull spanned by the training data. For large $\mathcal{D}$, the new data point is likely to be outside of the training data and thus requires the trained ML model to extrapolate. Strictly speaking, the Mahalanobis distance assumes the data to follow a Gaussian distribution which holds only approximately for our application. However, this comes at the advantage that in order to characterize an n-variate dataset, only n and n^2 elements have to be stored for μ and Σ, respectively. The Mahalanobis distance can thus be seen as a simple and efficient extrapolation indicator [9].

In the following, we compute the Mahalanobis distance during inference for the input features, based on the statistics μ and Σ collected during the training phase. For simplicity, we neglect the time dimension and compute the distribution quantities solely based on the last timestep of each input sequence X. The distribution parameters thus have the dimension $\mu \in \mathbb{R}^3$ and $\Sigma \in \mathbb{R}^{3\times 3}$.

2.4 Stability training

The stability training method was originally proposed by Zheng et al. [14] in the context of image recognition to increase the robustness of the trained ANN against noisy input data and adversarial attacks. This method asserts that small perturbations in the input training data induce only small perturbations in the ANN prediction. To this end, an additional penalty term (*stability loss*) is added to the standard loss

		$\sigma = 0.01$			$\sigma = 0.02$		
	Standard	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 1.0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 1.0$
MSE	$9.7 \cdot 10^{-3}$	$1.2 \cdot 10^{-1}$	$3.5 \cdot 10^{-1}$	$1.2 \cdot 10^{0}$	$1.8 \cdot 10^{-1}$	$5.1 \cdot 10^{-1}$	$1.6 \cdot 10^{0}$
CC	0.9993	0.9990	0.9983	0.9943	0.9990	0.9981	0.9926

Table 1: Accuracy of the standard and the stabilized models on the test set given as the mean-squared-error (MSE) and cross-correlation CC.

function, which yields

$$\mathcal{L} = \underbrace{\left\| \hat{Y} - Y \right\|}_{\text{standard loss}} + \alpha \underbrace{\left\| \hat{Y} - \hat{Y}' \right\|}_{\text{stability loss}} \quad \text{with} \quad \hat{Y}' = \text{ANN}\left(X + \varepsilon\left(\sigma^2\right) \right) . \tag{6}$$

The *standard loss* corresponds to the loss function in a standard training procedure. For this, the ANN is evaluated on the input sample X which gives a prediction $\hat{Y} = \text{ANN}(X)$. In the stability training, also a noisy sample X' is generated by adding noise $\varepsilon(\sigma^2)$ with variance σ^2 to the input sample X. Generally, the type of noise is arbitrary but we restrict ourselves to Gaussian noise in the present work. The ANN is then also applied to this noisy sample yielding the corresponding noisy prediction $\hat{Y}'$. The distance between the noisy prediction $\hat{Y}'$ and the standard prediction $\hat{Y}$ then yields the *stability loss* in Eq. (6). In order to reduce the overall loss in the training, the optimizer has to find a set of parameters which gives not only accurate results, i.e. reduces the *standard loss*, but is also robust to perturbations in the network input, i.e. reduces the *stability loss*. The weighting factor α balances these two objectives. Both parameters σ and α are additional hyperparameters which have to be tuned to the specific test case. The general outline of the stability training is shown in Figure 3.

All investigated ANN were trained for 20 epochs with a batch size of 256 and an initial learning rate of $5 \cdot 10^{-4}$ with the Adam optimizer [5]. The learning rate was halved every 5 epochs. The input features are normalized to zero mean and unit variance. The mean-squared-error (MSE) was used for the *standard loss* as well as the *stability loss*.

3 RESULTS

In the following section, the accuracy of the trained ANN is assessed in an a priori analyis on the testing data in Section 3.1. In Section 3.2, the a posteriori performance of the ANN is investigated in practical simulations. The ANN was trained with different hyperparameter configurations ($\alpha \in \{0.2, 0.4, 1.0\}$ and $\sigma \in \{0.01, 0.02\}$) to highlight the qualitative influences of the individual hyperparameters on the results.

3.1 A priori analysis

The performance of the trained ANN is first assessed on the unseen testing run (see Section 2.1). The mean-squared-error (MSE) and the cross-correlation (CC) are used as performance metrics and are shown for all trained ANN in Table 1. The predictions of all trained networks achieve very high cross-correlation $CC > 0.99$ with the exact closure terms. However, the MSE increases for all ANN with stability training by an order of magnitude in comparison to the baseline ANN. The results in Figure 4 indicate that the predictions become increasingly "washed out" with higher weighting factors α. The

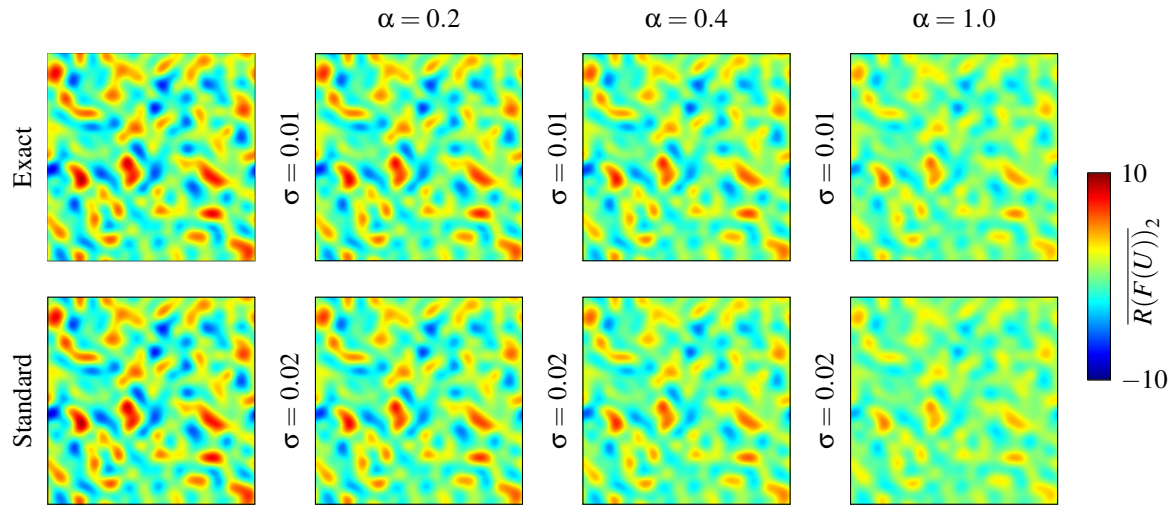

Figure 4: Predictions for $\overline{R(F(U)}_2$ of the trained ANN on the unseen testing data. The exact closure terms and the predictions of the ANN trained without stability training (Standard) are shown for comparison in the leftmost column. The predictions of the stability trained ANN are sorted by their respective hyperparameters α (columns) and σ (rows). Shown are slices at $z = 2.4$ of the three-dimensional field solution.

networks thus systematically underestimate the extrema in the closure terms. This directly follows out of the stability training procedure, which strives to flatten the functional response of the ANN around the training points.

However, the ANN with stability training still achieve good accuracy on the blind test data. Furthermore, it is sensible for many applications to accept slightly reduced accuracy for increased stability. As shown in Section 3.2, even small noise levels lead to significant increases in stability, while still retaining high accuracy on the testing dataset. It is thus crucial to include both tasks into the optimization process and to find the optimum balance between accuracy and stability.

3.2 A posteriori analysis

For the a posteriori assessment of the model performance, the trained ANN are used as predictions for the closure terms during actual LES. To this end, the filtered flow state of the blind test run at $t = 1.402$ is used as initial state for an LES with the discontinuous Galerkin flow solver FLEXI [6]. The solution is advanced in time by integrating Eq. (3) with a third-order Adams-Bashforth scheme and a time step of $\Delta t = 1 \cdot 10^{-4}$, which corresponds to the time step of the training data. It proved sufficient to evaluate the ANN and to thus update the prediction of the closure terms only in every tenth time step. Thereby, the solution at the last time steps is stored and every evaluation of the network is independent of the previous ones. The flow states $t \in [1.4, 1.402]$ are used to initialize the ANN for the first 20 time steps.

The key task of a turbulence model is to mimic the energy transfer from the resolved scales to the non-resolved ones. This dissipation mechanism is crucial to obtain a stable LES. We thus use the evolution of the total kinetic energy in the domain to assess the stability of the trained ANN as turbulence models. The results for the different models are shown in Figure 5 with the filtered DNS results as reference.

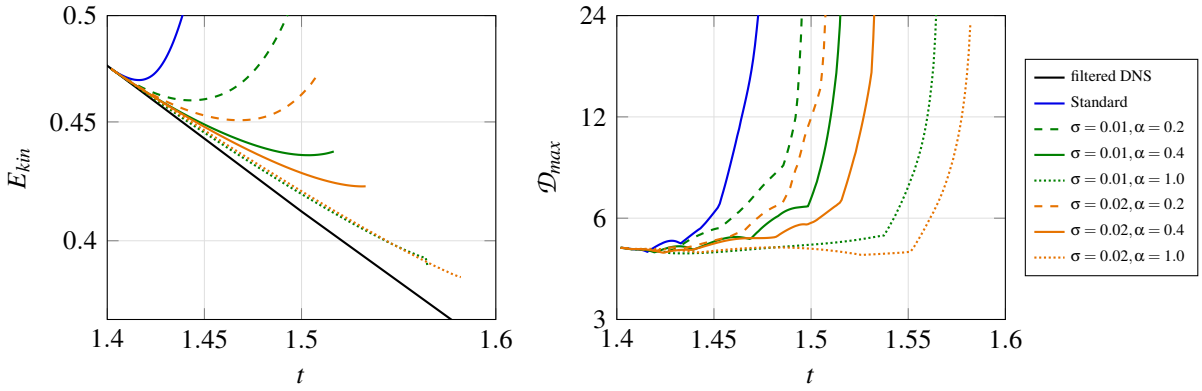

Figure 5: *Left:* Total kinetic energy in the domain over time for the inference tests with the closure terms predicted by the trained ANN. The filtered DNS solution is given for reference. *Right:* Maximum Mahalanobis distance $\mathcal{D}_{max}$ of the input data with respect to the training dataset. $\mathcal{D}_{max}$ is plotted on a logarithmic scale

The results show that the models trained with stability training remain stable for an increased length of time in comparison to the ANN without stability training. Moreover, a higher weighting factor α during training directly corresponds to increased stability in the simulation. This is to be expected since a higher α increases the importance of the stability constraint during training.

Comparing the stability of the simulations with the maximum Mahalanobis distance $\mathcal{D}_{max}$ in the flow field in Figure 5 shows a distinct correlation. While $\mathcal{D}_{max}$ seems to grow exponentially in time, the stability training can reduce the growth rate and thus favor the stability of the simulation. It was found that the variance of $\mathcal{D}$ increases only slowly over time. This suggests that merely a few points in the domain obtain such a high $\mathcal{D}$, while the majority of points still resemble the distribution of the training data, which was verified empirically for the investigated configurations. These isolated points with high Mahalanobis distance to the training data were also identified to cause the simulation to eventually crash.

4 CONCLUSION

In this work, we have demonstrated that the instability of machine learning models in simulations can correlate with the systematic mismatch of the training dataset and the input data during inference by means of the Mahalanobis distance. To this end, we have applied the method of *stability training* to improve the robustness of ANN as closure models for LES. We have shown that this approach can increase the stability of simulations significantly, while still providing accurate predictions of the closure terms.

In future work, a fallback mechanism can be implemented based on the Mahalanobis distance, which showed to be an appropriate extrapolation indicator. While traditional turbulence models can be employed in regions where the model is assumed to extrapolate, the accurate predictions of the ANN-based model can still be leveraged in the remaining domain, where the input data complies with the statistics of the training data. Such a classifier could obviously also be based on an ANN, which is trained to blend the model prediction with a traditional turbulence model based on the expected reliability of the former.

ACKNOWLEDGEMENTS

This research was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy - EXC 2075 – 390740016. The authors gratefully acknowledge the support by the Stuttgart Center for Simulation Science (SimTech) as well as the support and the computing time provided by the HLRS through the project "hpcdg".

References

[1] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, 398:108910, 2019.

[2] Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modelling. *GAMM Mitteilungen*, to appear in 2021.

[3] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103 – 111, October 2014.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Nico Krais, Andrea Beck, Thomas Bolemann, Hannes Frank, David Flad, Gregor Gassner, Florian Hindenlang, Malte Hoffmann, Thomas Kuhn, Matthias Sonntag, et al. FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws. *Computers & Mathematics with Applications*, 81:186–219, 2021.

[7] Marius Kurz and Andrea Beck. A machine learning framework for LES closure terms. *arXiv preprint arXiv:2010.03030*, 2020.

[8] Jacob A. Langford and Robert D. Moser. Optimal LES formulations for isotropic turbulence. *Journal of Fluid Mechanics*, 398:321–346, 1999.

[9] Julia Ling and Jeremy Templeton. Evaluation of machine learning algorithms for prediction of regions of high Reynolds-averaged Navier-Stokes uncertainty. *Physics of Fluids*, 27(8):085103, 2015.

[10] Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.

[11] Robert D. Moser, Sigfried W. Haering, and Gopal R. Yalla. Statistical properties of subgrid-scale turbulence models. *Annual Review of Fluid Mechanics*, 53, 2020.

[12] Jordan Ott, Mike Pritchard, Natalie Best, Erik Linstead, Milan Curcic, and Pierre Baldi. A Fortran-Keras deep learning bridge for scientific computing. *Scientific Programming*, 2020, 2020.

[13] Stephan Rasp. Online learning as a way to tackle instabilities and biases in neural network param-

eterizations. *arXiv preprint arXiv:1907.01351*, 2019.

[14] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.