ParCFD2022
33^{rd} International Conference on Parallel Computational Fluid Dynamics
May 25-27 2022, Alba Italy

PARALLEL AND SCALABLE DEEP LEARNING TO RECONSTRUCT ACTUATED TURBULENT BOUNDARY LAYER FLOWS. PART I: INVESTIGATION OF AUTOENCODER-BASED TRAININGS

R. Sarma*, M. Albers†, E. Inanc*, M. Aach*, W. Schröder†, AND A. Lintermann*

* Forschungszentrum Jülich GmbH
Jülich Supercomputing Centre
Wilhelm-Johnen-Straße, 52425 Jülich, Germany
e-mail: [R.Sarma,E.Inanc,M.Aach,A.Lintermann]@fz-juelich.de
web page: https://www.fz-juelich.de/ias/jsc/

† RWTH Aachen University
Institute of Aerodynamics and Chair of Fluid Mechanics
Wüllnerstraße 5a, 52062 Aachen, Germany
e-mail: [M.Albers,Office]@aia.rwth-aachen.de,
web page: https://www.aia.rwth-aachen.de

Key words: Autoencoders, Distributed Learning, Physics-Informed Neural Networks

Summary. With the availability of large datasets and increasing high-performance computing resources, machine learning tools offer many opportunities to improve and/or augment numerical methods used in the field of computational fluid dynamics. A low-dimensional representation of a turbulent boundary layer flow field is generated by a plain and a physics-contrained autoencoder. The training makes use of a distributed learning environment. The average test error of the plain autoencoder is ≈4.4 times smaller than the error of the physics-constrained autoencoder although the latter integrates physical laws in the training process. Furthermore, after 1,000 epochs, the training loss of the physics-constrained autoencoder is ≈9.1 times higher than the plain autoencoder after 300 epochs. The neural network corresponding to the plain autoencoder is able to provide accurate reconstructions of a turbulent boundary layer flow.

1 Introduction

Machine learning (ML) provides multiple alternative approaches to develop solutions to fluid flow problems tackled by various computational fluid dynamics (CFD) methods. Convolutional autoencoders (CAEs) are unsupervised learning methods that map a high-dimensional input to a low-dimensional latent space and then back to the original high-dimensional space. In contrast to traditional methods such as principal component analysis [1], CAEs provide an automated way to perform non-linear dimension reduction. There exist many examples of applications of autoencoders in CFD [2]. Despite such

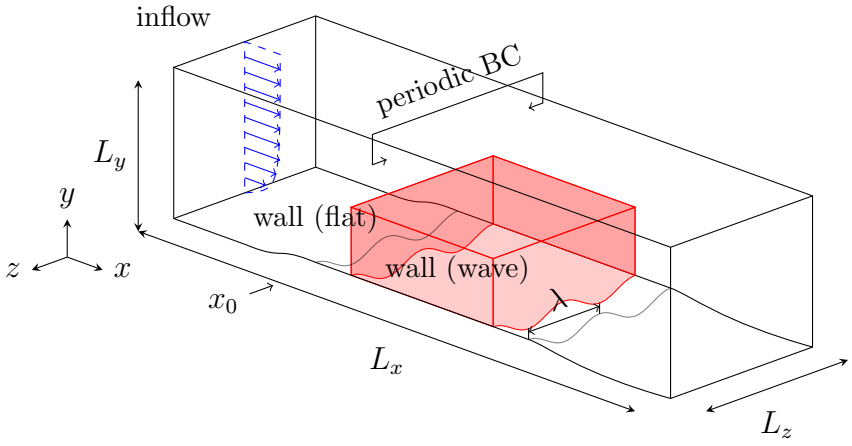


Figure 1: Computational domain of the actuated flat plate.

neural networks being widely used, their physical interpretation remains an issue. In this context, physics-informed neural networks (PINNs) provide a good alternative, which is explored in this study.

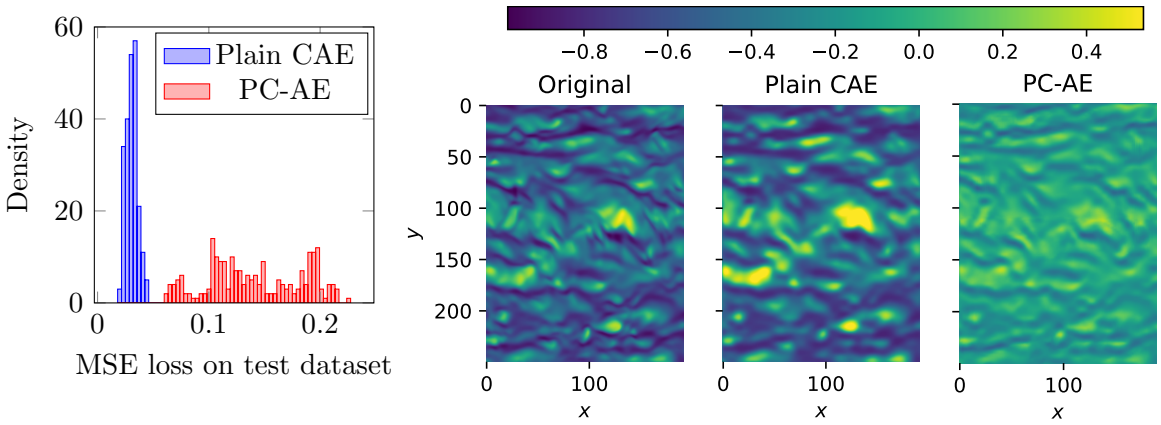
High-fidelity CFD simulations generate a huge amount of data, where ML methods can be used to learn from these data. However, processing these big data volumes efficiently with large state-of-the-art neural networks requires high-performance computing (HPC) systems. In this study, the large data availability is exploited in a modular supercomputing setup. Highly scalable, distributed learning frameworks are developed to train CAEs applied to a turbulent boundary layer (TBL) flow problem.

The objective of this study is to reconstruct actuated TBL flow fields using CAEs. The physical information of the TBL flow, i.e., the velocity fields, are initially compressed into a reduced (latent) space, which is then decompressed into the original dimension of the dataset. This latent space is then used to reconstruct TBL flow fields without running a CFD simulation.

The whole work is divided into two parts: The possible application of different CAEs to TBL problems is discussed in PART I, while in PART II, the training of CAEs using an HPC system by exploiting distributed data-parallelism is presented.

2 Turbulent Boundary Layer Specification

CAEs are used to investigate active drag reduction (ADR) techniques. Such techniques are designed to lower energy consumption, e.g., spanwise travelling transversal surface waves can improve the aerodynamic efficiency [3]. Figure 1 shows the computational setup of a large-eddy simulation (LES) of an actuated flat plate TBL flow providing the training data. For the simulation, the m-AIA solver [4], previously known as ZFS, is used. The freestream MACH number of this problem is $M_\infty = 0.1$, i.e., incompressibility can be assumed, except for local regions with slightly higher values. The training data consists of the velocity components u, v, w, pressure p, and density ρ, available in the red box depicted in Fig. 1. These quantities are scaled to the range $[-1, 1]$ during training.



(a) Error histogram. (b) Cross sections of the scaled normal velocity component.

Figure 2: Comparison of the performance of the plain CAE and PC-AE.

3 Framework for Autoencoders

A plain CAE and a physics-constrained AE (PC-AE) are developed in the ML-based framework PyTorch [5]. The former consists of two en- and de-coding two-dimensional convolutional layers with *LeakyReLU* activation function to introduce nonlinearity. A batch normalization is applied at the end of each activation layer and down-sampling is performed with max pooling operation. The Adam optimizer [6] with a weight decay parameter of 0.003 and a learning rate scheduler based on an exponential decay is employed, which has proven to work best for this application. The mean-squared-error (MSE) [7] between the original and reconstructed fields defines the loss function. The training set is comprised of 85% of the dataset, while the rest is used for testing.

The PC-AE is based on the work in [8]. Incompressibility is prescribed as a hard constraint by adding non-trainable layers after the encoder/decoder part of the plain CAE. The trainable part of the CAE yields the velocity potential. The curl of this potential is computed by kernels with fixed, non-trainable weights [9], which is then used to reconstruct the velocity field.

The CAEs are trained in a distributed, data-parallel environment, i.e., data is distributed across multiple workers, e.g., graphics processing units (GPUs) and central processing units (CPUs), while exchanging the training parameters in certain intervals to synchronize the gradients among the workers. This approach massively reduces the training time, which has been investigated with multiple open-source frameworks. However, it suffers from a loss in accuracy when scaled to a large number of GPUs due to a corresponding increase in the batch-size. This is further explored in PART II of this study.

4 Results, Conclusions, and Future Work

Figure 2 compares the performance of the AEs. The training of the plain CAE converges faster compared to the PC-AE. The training loss of the PC-AE (after 1,000 epochs) is ≈9.1 times higher than the loss of the plain CAE (after 300 epochs). The average test

errors in the plain CAE and PC-AE cases are 0.0314 and 0.1373, i.e., the former has ≈4.4 times smaller test error. The superior performance of the plain CAE can be clearly seen in Fig. 2a. Figure 2b qualitatively compares the reconstructed and original scaled velocity component v at a certain time instance. It is observed that the plain CAE is able to provide good reconstructions, while the PC-AE is able to only provide a qualitative agreement with good performance for values close to 1, while performing worse for smaller values close to -1. Quantitatively, the plain CAE outperforms the PC-AE in terms of accuracy and speed of convergence. The worse performance of the PC-AE could be related to the insufficient definition of the constraint, which needs to be reformulated. Further analysis is required to arrive at a conclusive argument.

The application of CAEs to an actuated TBL problem using a distributed learning framework is demonstrated. The plain CAE is able to provide good reconstructions, while a low accuracy in the PC-AC is observed, although the qualitative agreement for positive velocities seems promising. In the future, the training will employ larger datasets and alternative physical constraints for the TBL problem will be implemented.

REFERENCES

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] S. Wiewel, M. Becher, and N. Thuerey, "Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow," 2019. [Online]. Available: arXiv:1802.10123

[3] M. Albers, P. S. Meysonnat, and W. Schröder, "Actively Reduced Airfoil Drag by Transversal Surface Waves," *Flow, Turbulence and Combustion*, vol. 102, no. 4, pp. 865–886, 2019.

[4] A. Lintermann, M. Meinke, and W. Schröder, "Zonal Flow Solver (ZFS): a highly efficient multi-physics simulation framework," *International Journal of Computational Fluid Dynamics*, vol. 34, no. 7-8, pp. 458–485, 2020.

[5] A. Paszke, S. Gross, F. Massa, and A. Lerer et.al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017. [Online]. Available: arXiv:1412.6980

[7] X. Jin, P. Cheng, W.-L. Chen, and H. Li, "Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," *Physics of Fluids*, vol. 30, no. 4, p. 047105, 2018.

[8] A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence," 2020. [Online]. Available: arXiv:2002.00021

[9] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from Data," 2018. [Online]. Available: arXiv:1710.09668

The research leading to these results has been conducted in the CoE RAISE project, which receives funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.