Data-Driven Modeling of Steady Navier-Stokes using Deep Convolutional Neural Networks

Julian Kizanis

Neural Networks (NNs) have been able to solve Partial Differential Equations (PDEs) using much less computational power than traditional methods such as Finite Volume Method (FVM). Particularly, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs) and Fourier Neural Operators (FNOs) have shown great promise. These methods allow the network to efficiently extract both local and global information about a system. This work will focus on using CNNs as an encoder-decoder pair. The trained model will be fast enough to allow real time simulation of flow fields about an airfoil with varying geometry, angle of attack and Reynolds number.

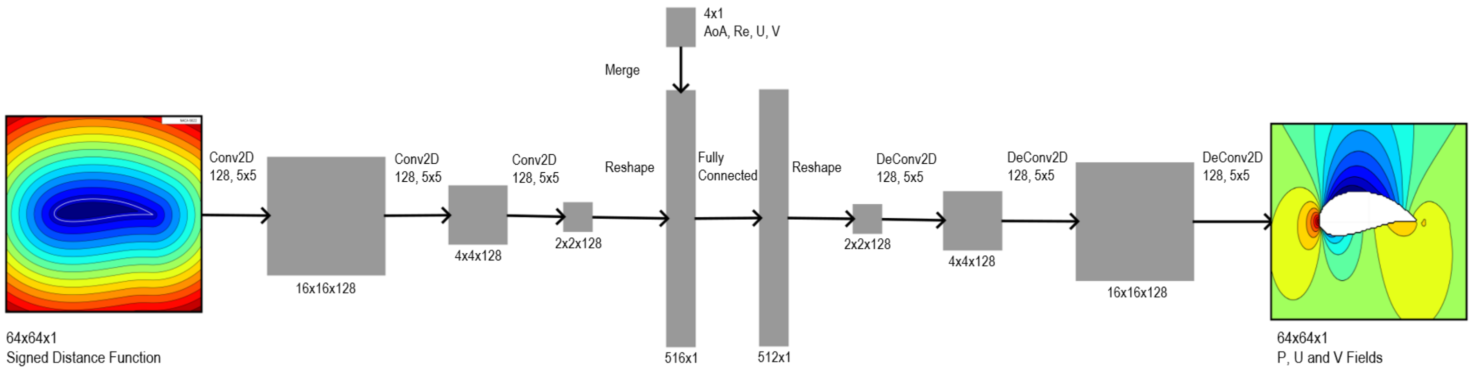


Figure 1: Autoencoder network. All strides are 1. Mean pooling is used to reduce the dimension of the network, while bilinear up sampling is used to increase the dimension of the network. Activation functions relu, leaky_relu, swish, tanh and sigmoid were tried. The best model (V5), used leaky_relu.

1 Introduction

Recently there has been much progress simulating fluid flow using neural networks. Gou et al. [3] used a convolutional neural network to predict non-uniform steady state flow about a body. Bhatnagar et al. [1] combined the decoding CNNs for pressure, x-velocity and y-velocity. This lead to an large improvement in training time and overall accuracy. Lee and You [6] used CNNs to predict unsteady flow fields over a circular cylinder. Tangsali et al. [10] demonstrated that CNNs can generalize into compressible regimes. Li et al. [7] demonstrated neural operators could approximate various PDEs in a mesh invariant manner. Bonnet et al. [2] provided a framework for comparing the efficacy of various neural network methods with regards to CFD. Ma et al. used U-nets to learn the steady state Navier-Stokes equations without the use of outside data. This was accomplished by using the conservation equations as the loss function. The primary contribution to this work are Bhatnagar et al. [1] and Tangsali et al. [10]. Source code can be found at https://github.com/Vizia128/EMEC436-Final-Project.git

2 Methodology

2.1 CFD simulation

The geometry for every naca airfoil from naca0001 to naca6624 was generated using MATLAB and Julia scripts. Then a template simulation was created in Star-CCM+. The simulation used a 2-dimensional, steady model with segregated flow, constant density and K-Epsilon turbulence. The inlet of the domain was set to a constant velocity. The output was set to a constant pressure. A no slip condition was imposed on the surface of the airfoil. A java macro that could import geometry and change the angle and magnitude of the input flow. A uniform sampling of airfoil geometries, angle of attacks (AoA) and Reynold numbers (Re) was taken. With -16° < AoA < 16° and 2 < log10(Re) < 7. The Star-Simulation was ran for 560 different sets of parameters. About 5% of the models did not converge and were discarded. Figure 2 shows a sample of the simulated parameters.

2.2 Feed Forward Neural Networks

A Feed Forward Neural Network consists of two or more layers. Each layer (L_a) can be represented with a vector ($\mathbf{v_a}$). Each element of the vector, represents a node of the neural network. To propagate from layer (L_a) to layer (L_{a+1}), we multiply $\mathbf{v_a}$ with a weight matrix (M). To

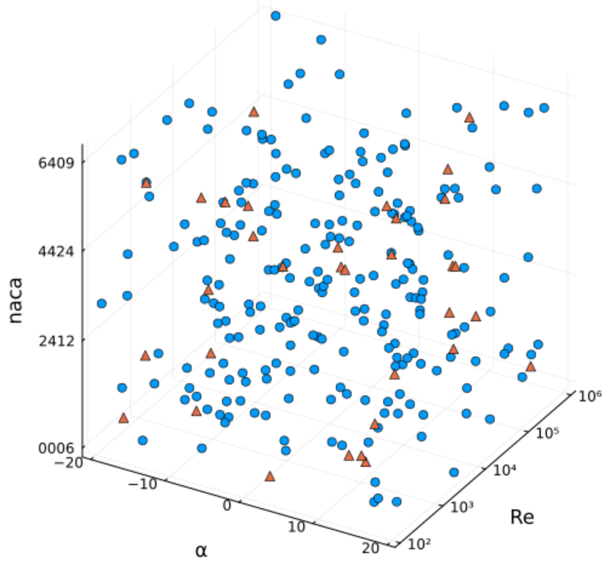


Figure 2: Scatter plot of the feature space for training and testing. The circles are from the training set, while the triangles are from the testing set.

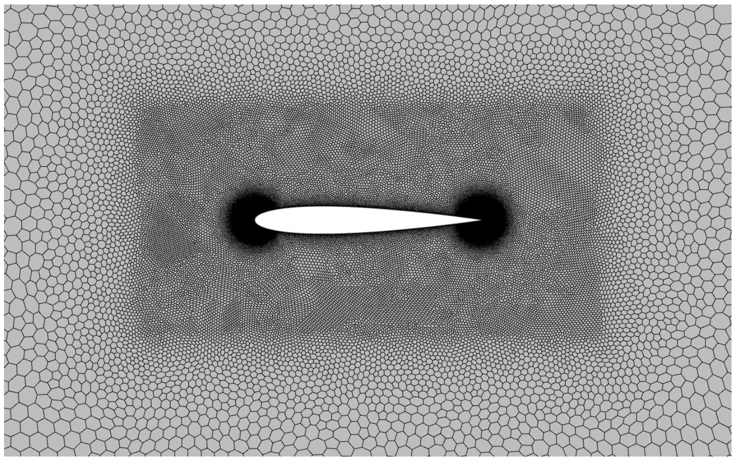
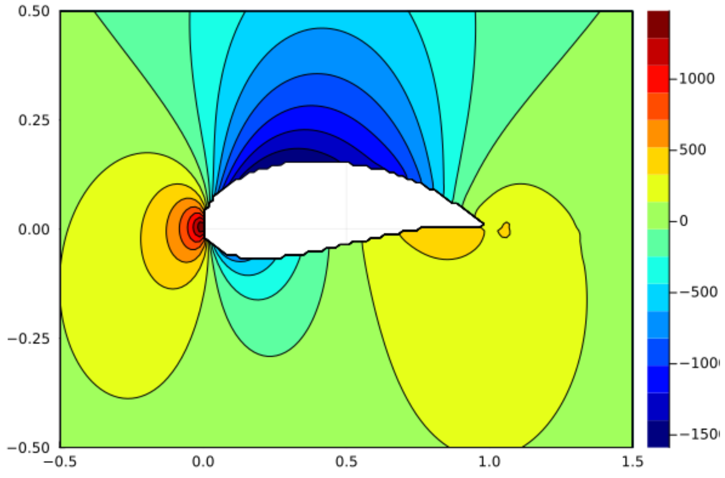
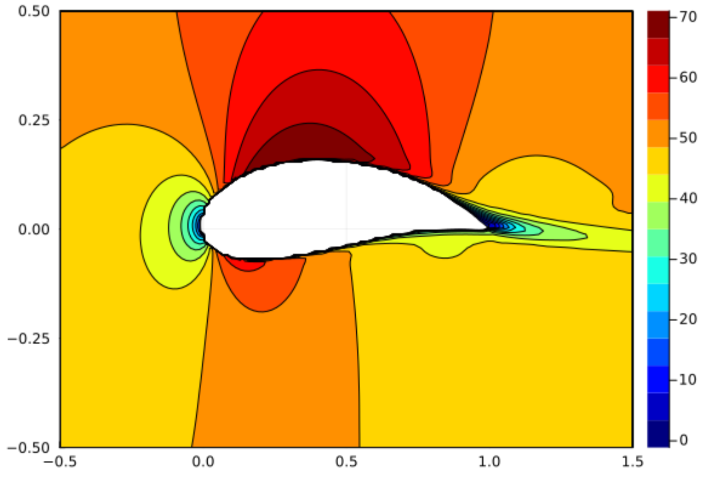


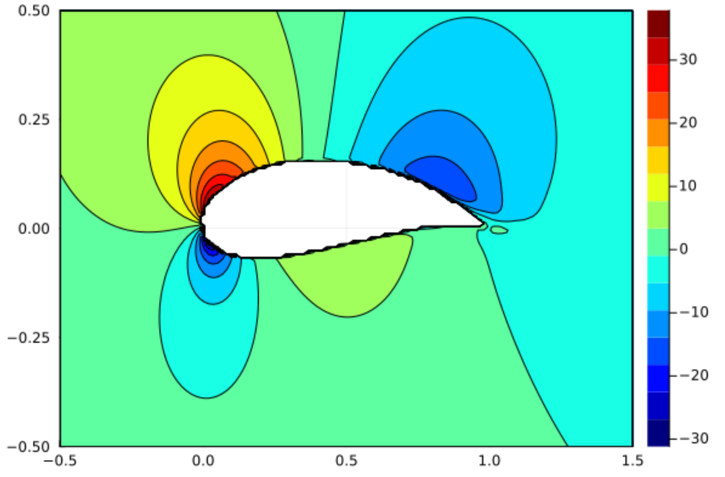
Figure 3: Mesh geometry for naca4308 with polygonal and prism cells



(a) Pressure



(b) X Velocity



(c) Y Velocity

Figure 4: CFD simulation results interpolated onto a 128x128 grid.

the output, a bais vector (**b**) is added before the elements of resulting vector are ran through an activation function (σ). Each element of the weight matrices and bias vectors represent a parameter of the neural network.

$$\mathbf{v_{a+1}} = \sigma.(M\mathbf{v_a} + \mathbf{b}) \tag{1}$$

The layers of the network can be separated into the input, output and hidden layers. The input layer inputs information from outside the model, while the output layer outputs data to outside the model. The inputs and outputs of the hidden layers are contained within the system and do not directly interact with the ouside world. With even one hidden layer, a feed forward network can approximate any Borel measurable function to an arbitrary level of accuracy [4].

Since the input of a Feed Forward Network is a vector, they loose all of the information contained within the location of each data point. Furthermore, the number of parameters needed scales with the square of the number of nodes. For a 100x100 pixel image, a fully connected feed forward network would need 100,010,000 parameters!

2.3 Convolutional Neural Networks

The convolution of two functions (f and g) results in a third function ($f \star g$) that is a blending of f and g.

$$(f \star g)(t) = \int f(\tau)g(t-\tau)d\tau$$

A convolution can also be represented taking the Fourier transform of f and g, multiplying the resultant functions together and then taking the inverse Fourier transform. This interaction with Fourier space allows a convolution to efficiently extract important information from very high dimensional spaces. A Convolutional Neural Network (CNN) uses convolutions to encode important features. A 2D convolutional layer can be represented by a 3D array. The extra dimension represent the depth of the layer. To propagate from one layer (C_a) to the next (C_{a+1}), C_a is convoluted with k kernels (K). Where k is the depth of (C_{a+1}) and a kernel is a nxnxk array of parameters. Each element of the result is then ran through an activation function (σ).

$$C_{a+1} = \sigma(\sum_{i=1}^{k} C_a \star K_i)$$

2.4 Signed Distance Functions

A signed distance function, or SDF, outputs the smallest distance from a given point, $\mathbf{x_a}$, to the boundary of a given shape $\partial\Omega$. Points that are within the shape are given a negative sign.

$$SDF(\mathbf{x}) = \begin{cases} d(\mathbf{x}, \partial\Omega), & \mathbf{x} \notin \Omega \\ 0, & \mathbf{x} \in \partial\Omega \\ -d(\mathbf{x}, \partial\Omega), & \mathbf{x} \in \Omega \end{cases} \tag{2}$$

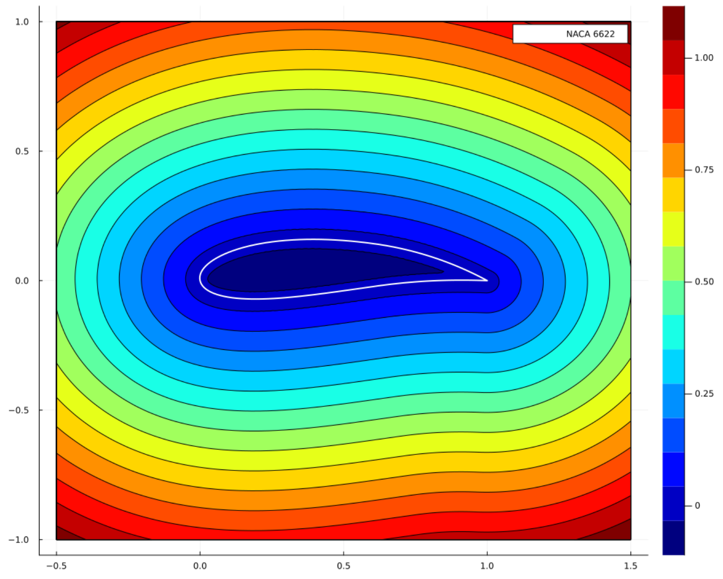
Figure 5a shows a 256x256 contour map of the SDF of a NACA 6622 airfoil. Even at the extremely low resolution of Figure 5b, most of the information of the original airfoil is recoverable. Comparing Figure 5b with Figure 5c, the sdf retains much more information than the bit-map for lower resolutions. For this work, the most successful model was trained with an input SDF size of 64x64.

2.5 Encoder Decoder Networks

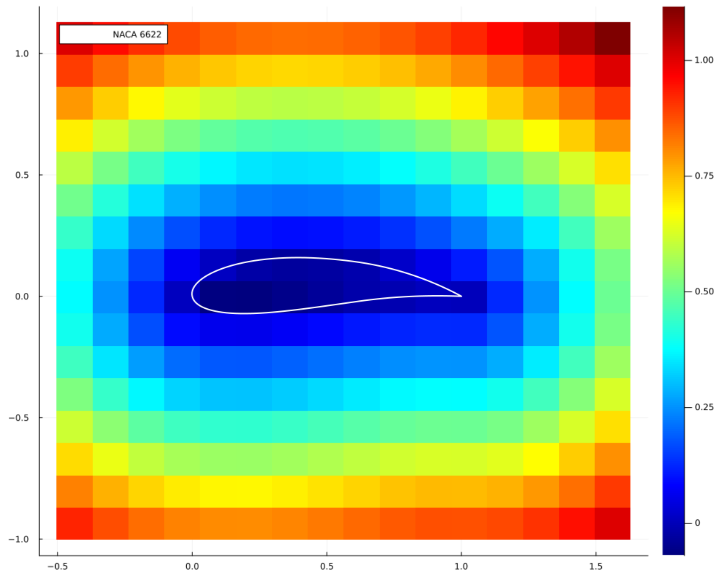
Encoder decoder networks, or autoencoders, combine two CNNs with a feed forward network. First a CNN extracts geometry information from the input SDF while decreasing its dimensionality. Then the layer is flattened to a vector. Reynolds number and angle of attack information is concatenated. This vector is fed forward to an output vector. This new vector is reshaped into an array. Lastly, this array is run through a CNN that creates the pressure and velocity fields.

3 Results and Discussion

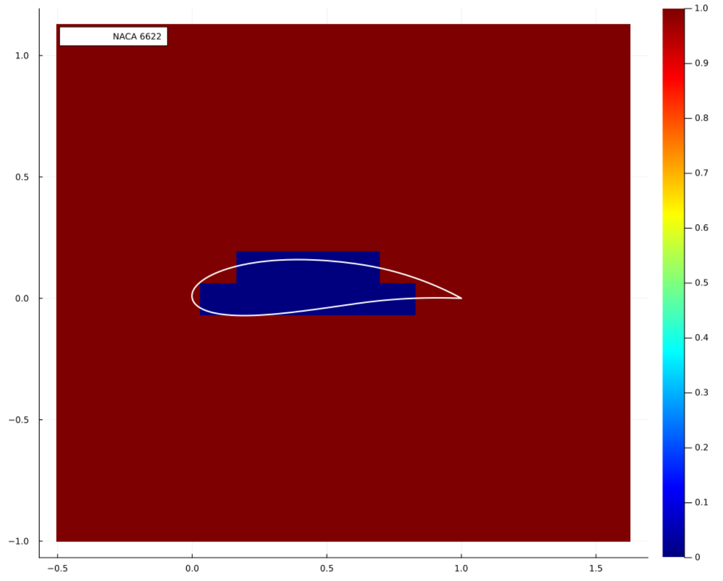
As demonstrated in Figure 8 the network was able to accurately model the general behavior of the system. The network has a difficult time reproducing the features near the leading edge and along the airfoils wake. Furthermore, there is very little variation in predicted output from different input parameters. In Figure 7, the network was completely unable to reproduce the effects of a high angle of attack. It appears the model is converging to a constant mapping of the mean. To help resolve this issue, I fixed all the bias parameters to zero [8]. I also tried adding batch normalization layers [5]. While these helped, the model still tends to ignore it's input parameters. Since the latent space of my model was half as large as other successful models [Tangsali, Krishnamurthy, and Hasnain [10]][9][1], it is possible there is not enough bandwidth to adequately model the system.



(a) Contour plot of SDF.



(b) SDF plotted at 16x16 pixels.



(c) Bit-map plotted at 16x16 pixels.

Figure 5: Plots of the naca6622 airfoil. The airfoil boundary is in white.

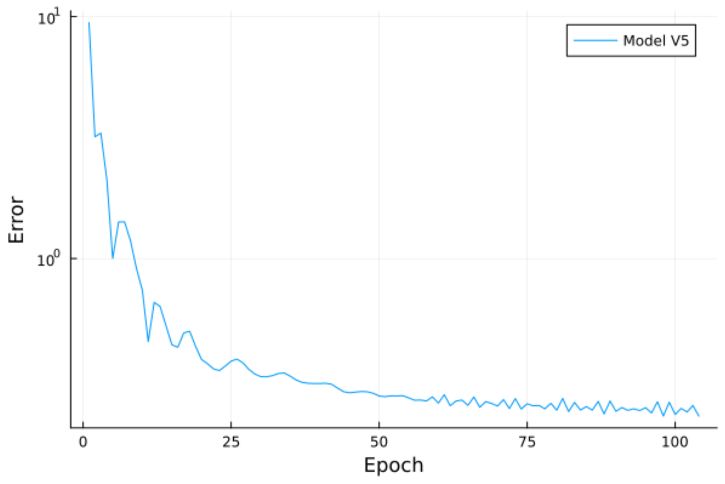
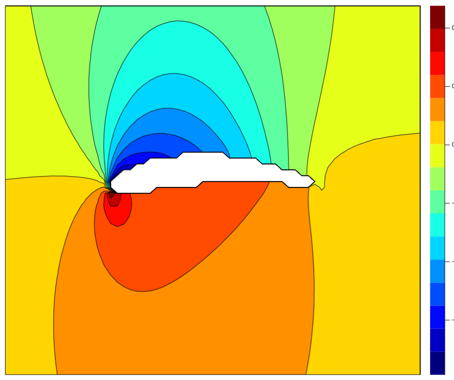


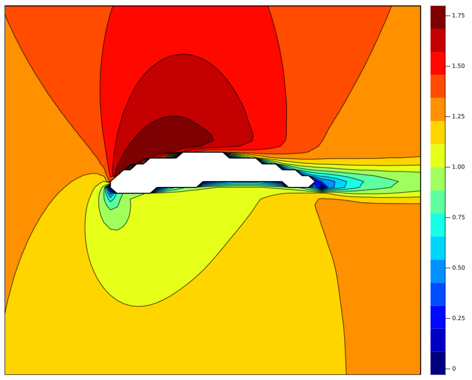
Figure 6: Convergence of the network

4 Conclusions

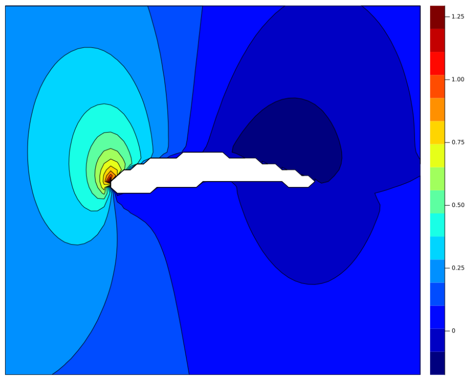
After they are trained, neural networks are an extremely fast method to model fluid dynamics. However, the models remain stuck in the second dimension with poor resolution. Furthermore, training these models can be very time and resource intensive. In further works, I would like to try varying the training parameters. I believe adding more energy to the system will dislodge the solution from any false peaks. Furthermore, using U-nets or graph networks would allow the model to be more mesh agnostic.



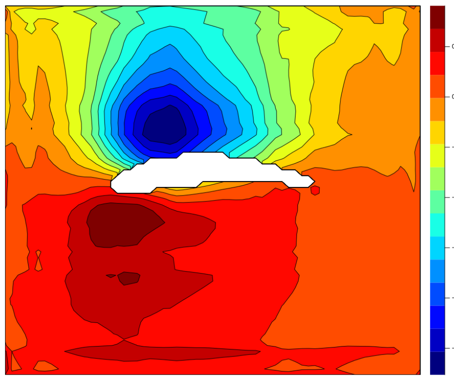
(a) True Pressure



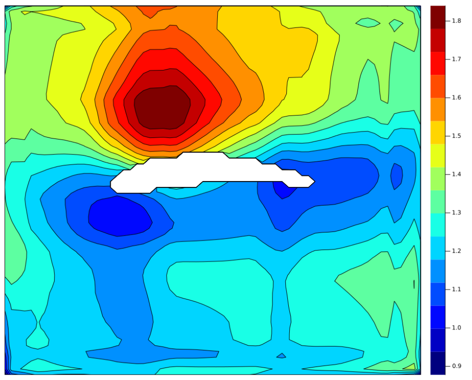
(b) True X-Velocity



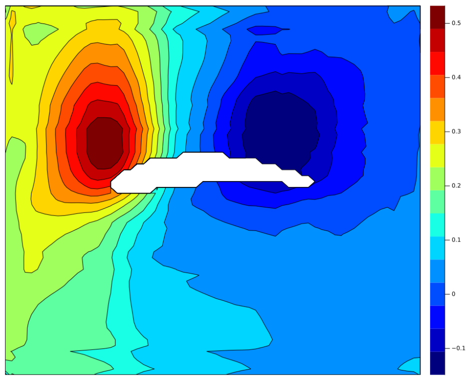
(c) True Y-Velocity



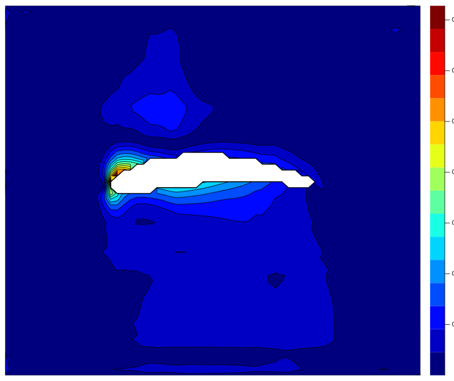
(d) Predicted Pressure



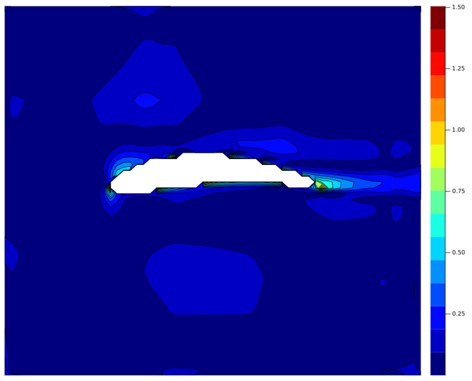
(e) Predicted X-Velocity



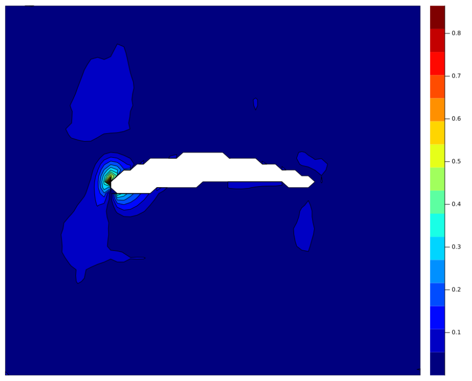
(f) Predicted Y-Velocity



(g) Absolute difference Pressure



(h) Absolute difference X-Velocity



(i) Absolute difference Y-Velocity

Figure 7: Contour plots with $Re = 24618$ and $AoA = 5.37$

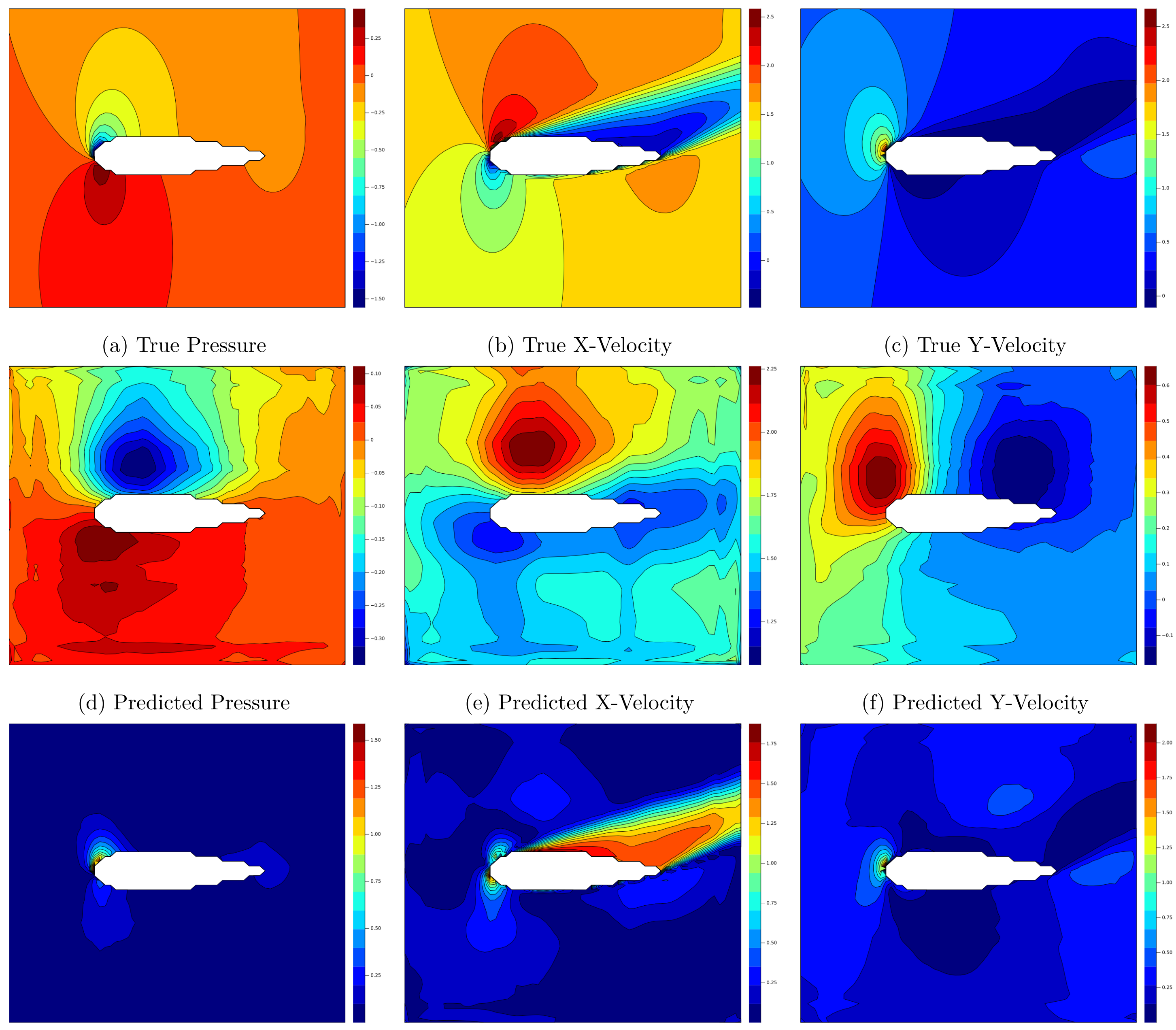

(a) True Pressure
(b) True X-Velocity
(c) True Y-Velocity
(d) Predicted Pressure
(e) Predicted X-Velocity
(f) Predicted Y-Velocity
(g) Absolute difference Pressure
(h) Absolute difference X-Velocity
(i) Absolute difference Y-Velocity

Figure 8: Contour plots with $Re = 30255$ and $AoA = 15.1$

References

[1] Saakaar Bhatnagar et al. “Prediction of aerodynamic flow fields using convolutional neural networks”. en. In: *Computational Mechanics* 64.2 (Aug. 2019), pp. 525–545. ISSN: 0178-7675, 1432-0924. DOI: 10.1007/s00466-019-01740-0. URL: http://link.springer.com/10.1007/s00466-019-01740-0 (visited on 10/21/2022).

[2] Florent Bonnet et al. “AN EXTENSIBLE BENCHMARKING GRAPH-MESH DATASET FOR STUDYING STEADY-STATE INCOM-PRESSIBLE NAVIER-STOKES EQUATIONS”. en. In: (2022), p. 25.

[3] Xiaoxiao Guo, Wei Li, and Francesco Iorio. “Convolutional Neural Networks for Steady Flow Approximation”. en. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 481–490. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939738. URL: https://dl.acm.org/doi/10.1145/2939672.2939738 (visited on 12/10/2022).

[4] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. en. In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0893608089900208 (visited on 12/10/2022).

[5] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv:1502.03167 [cs]. Mar. 2015. URL: http://arxiv.org/abs/1502.03167 (visited on 12/08/2022).

[6] Sangseung Lee and Donghyun You. “Data-driven prediction of unsteady flow fields over a circular cylinder using deep learning”. In: *Journal of Fluid Mechanics* 879 (Nov. 2019). arXiv:1804.06076 [physics], pp. 217–254. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2019.700. URL: http://arxiv.org/abs/1804.06076 (visited on 10/24/2022).

[7] Zongyi Li et al. *Neural Operator: Graph Kernel Network for Partial Differential Equations*. arXiv:2003.03485 [cs, math, stat]. Mar. 2020. URL: http://arxiv.org/abs/2003.03485 (visited on 09/26/2022).

[8] Philipp Liznerski et al. *Explainable Deep One-Class Classification*. arXiv:2007.01760 [cs, stat]. Mar. 2021. URL: http://arxiv.org/abs/2007.01760 (visited on 12/10/2022).

[9] Jiang-Zhou Peng et al. “Data-Driven Modeling of Geometry-Adaptive Steady Heat Convection Based on Convolutional Neural Networks”. en. In: *Fluids* 6.12 (Dec. 2021), p. 436. ISSN: 2311-5521. DOI: 10.3390/fluids6120436. URL: https://www.mdpi.com/2311-5521/6/12/436 (visited on 10/29/2022).

[10] Kaustubh Tangsali, Vinayak R. Krishnamurthy, and Zohaib Hasnain. “Generalizability of Convolutional Encoder–Decoder Networks for Aerodynamic Flow-Field Prediction Across Geometric and Physical-Fluidic Variations”. en. In: *Journal of Mechanical Design* 143.5 (May 2021), p. 051704. ISSN: 1050-0472, 1528-9001. DOI: 10.1115/1.4048221. URL: https://asmedigitalcollection.asme.org/mechanicaldesign/article/doi/10.1115/1.4048221/1086540/Generalizability-of-Convolutional-Encoder-Decoder (visited on 10/21/2022).