Data Driven Prediction of Steady Flow Past a Circular Cylinder Using Deep Learning Techniques

B. Prasannavenkatesh[1], Sai Guruprasad Jakkala[2], and Dr. S. Vengadesan[2]
[1]*School of Mechanical Engineering, SASTRA Deemed University, Thanjavur 613 401*
and
[2]*Department of Applied Mechanics, Indian Institute of Technology Madras, Chennai 600 036*
Presenting Author : B. Prasannavenkatesh, Email : bprassi@gmail.com

Program : Oral Presentation

Abstract

In this work, we implement a deep learning based method for predicting two-dimensional steady flow past a circular cylinder. We apply a tailored deep learning model in the form of a convolutional neural network for this purpose. We make use of a modern U-Net architecture to design the neural network, which is easier to construct and modify and also effective in its performance. We focus on training the model for predicting the pressure and the vertical and horizontal components of velocity in the flow. A structured mesh for the cylinder geometry is created using the meshing tool Gmsh and then used to simulate laminar flow with a Reynolds number(Re=100) using the models in OpenFOAM. We adopt a supervised training approach to train the network. The datasets required for training and subsequently testing the network are generated separately, and the open-source deep learning library Keras is used extensively to construct the network as well as the functions for training and testing respectively. The images of the three flow parameters as mentioned above predicted by the deep learning model are compared with the OpenFOAM-generated ground truth images to obtain the error and the results are further analysed.

Keywords: Machine learning; deep learning; neural networks; laminar flow; steady flow prediction.

1 Introduction

Simulation of the Navier-Stokes equation in an effective, accurate manner has been a continuing challenge for researchers worldwide. This is due to the fact that the equation has extremely wide-ranging applications from fluid dynamics research to computer graphics and movie-making. The primary obstacle to this challenge has been the requirement of extensive computational resources in addition to long periods of time. Though machine learning is not completely nascent in its application to CFD, the development of deep learning models and libraries have greatly aided the rapid growth of data-driven CFD research.

In this work, we intend to bypass this debilitating obstacle by tackling the problem at hand with a data-driven approach. With such an approach, we are capable of leveraging the inherent power of neural networks in approximation. Even a data-driven approach of a large scale uses just a fraction of the computational resources and time demanded by disretization based solvers. The only pitfall to this approach is that it generally requires a huge amount of data to train a deep learning model, even to a satisfactory level. Nevertheless, this is a one-time activity and an effectively trained model can be tested with different datasets for a number of times with just a few modifications to the neural network architecture.

The work by the research group under Dr. Nils Thuerey at TU Munich on a deep learning framework for airfoil flows has been an underlying framework and contributing factor to our work [1].

This deep learning model is not too tailored and is generic enough to handle other similar problems in the domain given that it is trained afresh. We hope that in the future, one truly intelligent model with complete knowledge of the flow physics is created, one which can surpass traditional CFD solvers in terms of accuracy.

2 Methodology

This paper is structured as follows:

(i) Neural Network Architecture
(ii) Data Generation
(iii) Training/Testing the model
(iv) Results and analysis

2.1 Neural Network Architecture

Convolutional neural networks (CNN) are one of the most widely applied tools in deep learning. The U-Net is a special variant of encoder-decoder CNN architecture [2]. See Fig 1. We use an input size of 128^2x3(one layer each for pressure and the two components of velocity) to the network to maintain uniformity with the solutions from OpenFOAM simulations. Then seven-layer encoder part of our network samples down the input image using *4x4* filters

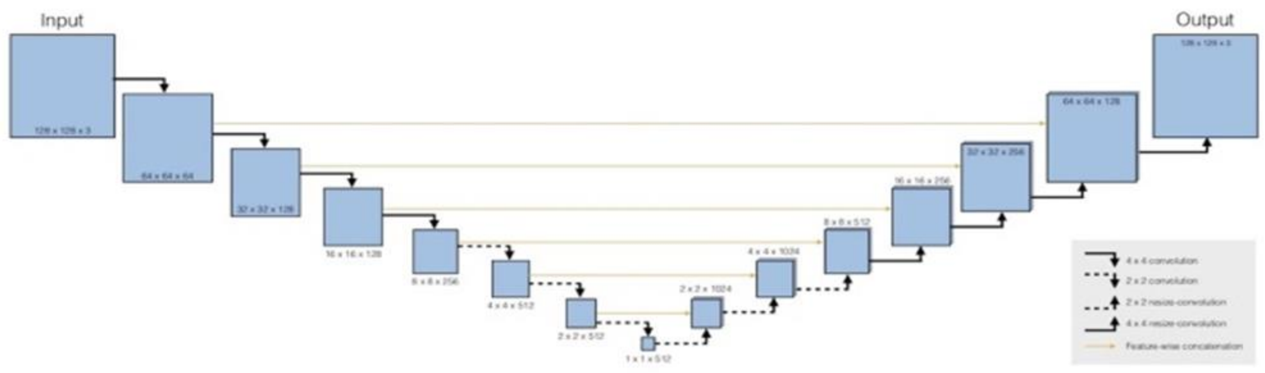

Figure 1.The U-Net architecture

in the first four steps and *2x2* filters in the subsequent three steps respectively, through strided convolutions. This results in 512 features being extracted from a single input image at the end of the encoder part. The decoder part of our architecture essentially mirrors the encoder part. Through another seven symmetric layers, it reconstructs the target image, thus rendering a symmetrical *bowtie* structure to the network and maintaining equal-sized input and output. See Fig 1.

2.2 Data Generation

In this work, we randomize the cylinder positions in the domain using three variables namely, the Cartesian coordinates of the cylinder center. The randomly generated cylinder positions are written into a single file to be read by the OpenFOAM solver with a constant Reynolds number of 100. This randomizing is done in such a way that the complete cylinder geometry falls within the sampled *128x128* grid which is then fed to the CNN as input. See Fig 2.We automate the aforementioned process by writing a dedicated Python script.

The mesh is created using the meshing tool Gmsh. For every cylinder position the mesh is automatically modified as all the mesh regions are just a function of the three independent variables mentioned above. We generate 4500 random cylinder positions and divide it in an 8:1 ratio for training and testing the model respectively. The training and testing datasets, once generated are stored in separately labeled directories.

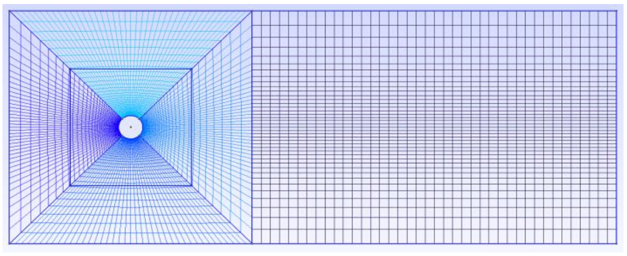

Figure 2.The 128x128 grid highlighted in the actual mesh

2.3 Training/Testing the model

Keras is an open-source neural network library for standard, convolutional and recurrent neural networks. It is beneficial to use such a high-level library for our work given its user friendly and modular nature. It allows for Graphics Processing Unit (GPU) based distributed training of our model which accelerates the process. It also supports additional utility layers like pooling and normalization which are effective in gradient propagation and scaling down a large task to a manageable level.

2.3.1 Training

We adopt a supervised training methodology to train our model i.e. we feed the complete dataset consisting of 4000 samples for training the model in a systematic manner. For this purpose, we use the Adam optimizer [3] to control the learning rate and thereby the gradient descent. This is because as standard stochastic gradient descent maintains a constant learning rate throughout, it requires a lot more samples to reach convergence and thus slower. A learning rate of 0.0004 is chosen through a trial and error basis as an optimal learning rate ensures faster convergence and convergence at the local minima. Also, a learning rate decay of 10% in the second half of the training process i.e. the last 2000 samples is maintained, to stabilize results. A range of [0.0004, 0.004] is chosen as ideal for convergence. A non-linear activation function called the Leaky ReLU (See Eqn 3.) is used to process the activation of each neuron in the encoder part of the CNN. The decoder part uses regular ReLU. They simply convert the activation into an output signal. These functions are useful in learning the complex non-linear mapping between neurons. Finally, we choose a simple cost function to compute the magnitude of error between the predicted and ground truth results.

2.3.2 Testing

The testing is simply a repetition of training in principle with one critical change being the CNN does not have previous knowledge of the test dataset. It works on learning physical features from the data samples through the use of kernels being convolved over the input image’s matrix. For every new sample in the test dataset, as soon as the CNN is fed with the *128x128* grid, it employs its memory of features learned during training and attempts to approximate a specific target image for the features learned from the sample.

2.3.3 Equations

1. $a_i^{(j)} = \sigma(\sum_{i=0}^{i=n} \theta_{j-1,i}^{(j-1)} x_i)$ - standard forward propagation equation where $a_i^{(j)}$ is the activation of neuron i in layer j; θ is the matrix of weights/parameters mapped from layer j-1 to j; x_i is the input i in the first layer; σ is the non-linear activation function used.
2. $L_1 = |\hat{y}(\theta)\text{-}y(\theta)|$ - Cost/Error function used to compute the magnitude of difference between the ground truth and predicted results.
3. $Leaky\ ReLU = f(x) = \begin{cases} x, & x > 0 \\ 0.01x, & otherwise \end{cases}$ - the non-linear activation function used in the model.

2.4 Results and Analysis

The results are displayed in RGB and grey-scale formats with the first row displaying the ground truth – the CFD based pressure, vertical and horizontal velocity and the second row displaying the results predicted by our deep learning model. Through the grey-scale version of results, the error in prediction is clearly visible to the naked eye. See Fig 3 and Fig 4.

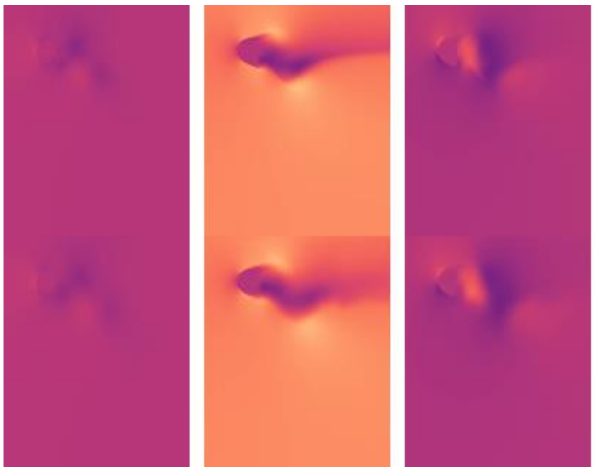

Figure 3.RGB comparison of ground-truth and predicted results. This corresponds to the 106th sample in the test dataset

In addition to this visual error analysis, certain numerical results are computed. For each data sample the following entities are computed and displayed on the terminal once the dataset is tested completely.

- The absolute difference between the actual and predicted results and the corresponding ratio, for all the three parameters.
- The loss percentage for pressure, velocity and the combined loss percentage.
- The cost or the L_1 error value as mentioned previously.
- The denormalized error. Normalization is generally performed for the neural network's input vectors to control the range of all inputs to a value between 0 and 1. This is done to ensure unnecessary computational power is not wasted in learning the ranges of input variables. Once the prediction is over, the vectors are correspondingly denormalized to their original values subsequently denormalizing the error too.
- Finally, the relative error averaged over all the test data samples and the standard deviation.

3 Conclusions

We have performed this work in an attempt to validate the fact that by tapping into the advantageous nature of deep learning in approximating, that too to an impressively accurate level, a new perspective to CFD is obtained. The future scope to this work is that when such powerful models are trained to solve the Navier-Stokes equation in a physics-coupled manner, the need for a large amount of training data can be bypassed too. This would result in one truly intelligent and independent model which can predict based on physics without prior training and one which is generic enough to be applied to all problems in the Navier-Stokes domain.

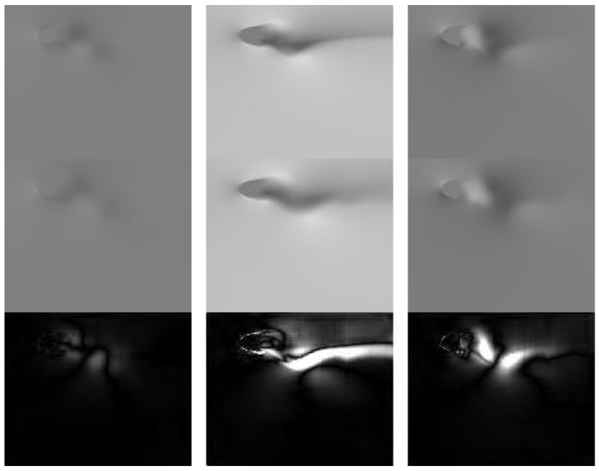

Figure 4.Results of the 106th test sample converted to a grey-scale format clearly visualizing the error in prediction.

4 Acknowledgements

We sincerely thank the Department of Applied Mechanics, IIT Madras for providing the computational resources required, and for the opportunity.

5 References

[1] Thuerey N, Xie Y, Chu M, Wiewel S, Prantl L. Physics-Based Deep Learning for Fluid Flow.

[2] Ronneberger O, Fischer P, Brox T, U-Net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention 2015 Oct 5 (pp. 234-241). Springer, Cham.

[3] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.