Artificial Surrogate Model for Computational Fluid Dynamics

Abdallah Alfaham[1] Siegfried Mercelis [2]

1-2- University of Antwerp - imec IDLab - Faculty of Applied Engineering
Antwerp, Belgium

Abstract. Simulating fluid dynamics is challenging due to the computational complexity of processing high-dimensional data, which often requires significant time. Fluid behavior is typically governed by partial differential equations (PDEs), and the complexity escalates when obstacles disrupt the flow, reinforcing vorticity formation. Vorticity describes the local rotational motion of a fluid. In this paper, we present a data-driven approach to automate PDE simulations and develop surrogate models to generate fluid dynamics based on the Kármán vortex street. Our approach aims to generate accurate fluid simulations with faster computation through architectural adjustments.

1 Introduction

Computational fluid dynamics (CFD) uses mathematical models and computational frameworks to address fluid flow challenges, replicating fluid behavior through predefined boundary conditions [1]. Differentiable PDE operations can be a useful approach for simulating complex fluid phenomena, such as flows with varying vorticities. High-speed supercomputers are essential for achieving efficient solutions, especially for large-scale, intricate problems.

The PhiFlow simulation toolkit, a differentiable PDE-solving framework, is designed for optimization and machine learning applications [2]. Its integration with machine learning frameworks simplifies the development of differentiable functions that seamlessly combine learning models and physics simulations.

A key challenge is the slow execution when using differential equations in programming. To address this, we propose surrogate machine learning models based on CNN-LSTM layers [3][4], enabling faster fluid dynamics simulations while maintaining precision.

2 Related Work

Several attempts have been made in the literature to develop surrogate models for CFD. Some focus on using optimization strategies to guide and control the generated flow, such as Forrester's study [5], which employs a two-stage surrogate model: construction and evaluation. If accuracy is insufficient, the model returns to stage one. Schaefer's study [6] applied a surrogate model error estimation technique, using withheld test points to estimate discretization errors and uncertainties in the airfoil problem. Additionally, some studies explore active flow control in CFD simulations using deep reinforcement learning and model-free methods to handle continuous actions [7][8].

Incorporating physics-informed data can significantly enhance the performance of the surrogate model [9]. However, we observe that applying the incompressible Navier-Stokes (NS) equations introduces certain losses in the simulator's behavior. These losses are likely due to the discretization of the continuous flow governed by the PDEs equations and the inherent complexity of the governing matrix, particularly the interactions between pressure and velocity fields. Therefore, this study focuses on refining the surrogate model architecture to explore the potential for improving performance and generating accurate dynamic fluids with time efficiency through architectural adjustments.

3 Methodology

CFD simulator generates flow based on pressure, velocity, and specific obstacles.

The environment has two horizontally placed cylinders with a specific distance between them. Since the cylinder properties remain fixed and only the distance can vary from one experiment to another which leads to unique vorticity formations, we used the distance as input to the surrogate model.

Pressure and velocity inputs are complex due to their multi-dimensional nature. CNN layers are used to downsample the data, reducing dimensionality while retaining key information [3]. An LSTM layer [4] is added to create local memory and preserve historical data. The updated pressure and velocity outputs are fed back into the model for the next timestep, forming closed loops.

However, closed loops predictions may expose the system to mismatch data and compound error.

We address data complexity and system vulnerability by splitting the surrogate model into two sub-models, each focusing on either velocity or pressure. We reduced the mismatch data and compound error by using the simulator as a helper in which during the training stage, the sub-model of velocity will receive the actual pressure, and the sub-model of pressure will receive the actual velocity. In other words, we not only use the actual velocity and pressure data from simulator to calculate the losses of the predicted ones, but also as a learning guide as shown in the fig 1. For testing, the simulator inferences are replaced by the model's output

For convergence and optimization, mean squared error (MSE) between the predicted data and the actual data is used as loss function 2, in addition to including physics information derived from the incompressible Navier-Stokes (NS) equations 1:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla \cdot \nabla \mathbf{u} + (0,1)^T \xi d \quad \text{s.t.} \quad \nabla \cdot \mathbf{u} = 0, \\ &\frac{\partial d}{\partial t} + \mathbf{u} \cdot \nabla d = 0 \end{aligned} \tag{1}$$

For an incompressible flow, we used a simple buoyancy model (the Boussinesq approximation) via the term $(0,1)^T \xi d$. This approximate changes in density for incompressible solvers. We assume a gravity force acts via the vector $(0,1)^T$. We solved this

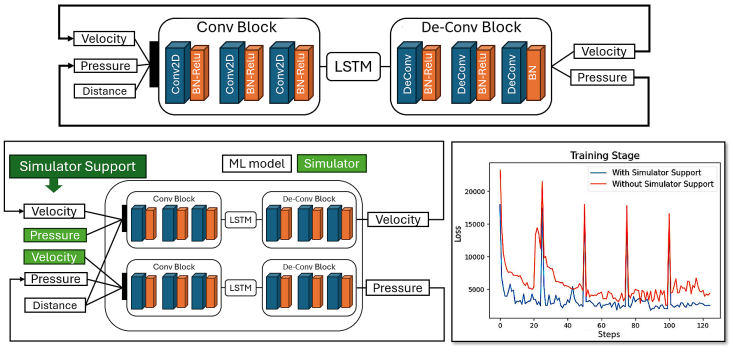

Fig. 1: Learning is boosted, and mismatched data and errors are reduced with simulator guidance. Complexity is minimized by splitting the surrogate model, while batch normalization (BN) layers reduce overfitting and enhance generalization.

PDE on a closed domain with Dirichlet boundary conditions $\mathbf{u} = 0$ for velocity, and $\frac{\partial p}{\partial x} = 0$ Neumann boundaries for pressure [9].

$$\text{Loss} = \alpha 1 \times Loss_{MSE}(Velocity) + \alpha \mathbf{2} \times Loss_{MSE}(Pressure) + \alpha \mathbf{3} \times Loss_{Physics} \quad (2)$$

where, $\alpha \mathbf{1}, \alpha \mathbf{2}$ are set to 1 to account for all potential mismatched data, and $\alpha \mathbf{3}$ is set to 0.2 to ensure that the $Loss_{Physics}$ corresponds to velocity and pressure losses.

4 Training

Flow trajectories at varying distances $(65, 195, 300)$ were used, with 5000 timesteps taken from each to analyze vorticity, as larger distances lead to lower vorticities. Loss peaks were observed at the start of each epoch due to the resetting of velocity and pressure 1. The training spans 150 epochs, with the loss plotted every 200 steps (25 steps per epoch). We plotted the loss for the first five epochs to emphasize the impact of simulation support 1.

This approach works well for fixed cylinder distances, but varying distances in multiple trajectories pose challenges due to unique vorticity formations. To create a surrogate model that handles these variations, we must minimize false dependencies, like those introduced by sequential passing of trajectories during training. Currently, our model operates as a centralized unit generating different flows, which can be a single point of failure (SPOF), limiting scalability and complicating maintenance.

We addressed these issues by shifting from centralization to decentralization, as complex systems can exhibit varying behavior based on the system state (e.g., cylinder position). Capturing diverse behavior across a large state space is challenging for a single model, so specialized local models help break down the problem. The decentralized model functions as a Mixture of Experts (MoE), which is a model that combines multiple specialized sub-models and selects the most relevant ones for a task [10]. The

distance determines which model is selected, as shown in Fig. 2, ensuring that only a relevant part of the system is engaged, unlike a centralized surrogate model.

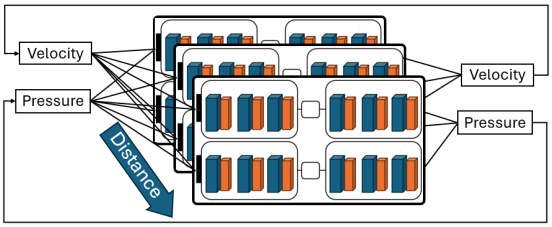

Fig. 2: In the Mixture of Experts (MoE) model, only the local model corresponding to the given distance will be activated. If there is no direct match, the closest models will be enabled, and the one with the highest performance will be selected as shown later in the Result section 5

5 Results

We used the trained ML models with in-context setup and generate 1000 frames for every fluid trajectory to check different vorticity formations without involving the simulator. Our objectives are time efficiency and ensuring high accuracy. As been shown in fig 3 and table 2, both centralized & MoE surrogate models generate flows in short time but they differ in accuracy. The performance of the centralized model is biased towards generating a specific fluid flow and its accuracy decreases as distances vary. This can be attributed to the effects of closed loop inputs and the false dependencies created between the fluids. While MoE model can avoid these shortcomings.

Table 1: Time & Performance for generating fluid dynamics

Type	Distance	Time consumption ↓	MSE Error ↓
CFD-Simulator	65	01:06:07	0
Centralized ML model	65	00:02:32	0.196
MoE Model	65	00:02:21	0.052
CFD-Simulator	195	01:10:17	0
Centralized ML model	195	00:02:52	0.136
MoE Model	195	00:02:21	0.061
CFD-Simulator	300	01:04:47	0
Centralized ML model	300	00:02:30	0.098
MoE Model	300	00:02:16	0.057

For MoE model, the ideal situation is to create a local model for every possible distance between the cylinders, then predict the fluid for infinite time. Nevertheless, it is interesting to evaluate the accuracy of fluid predictions from non-existent local models.

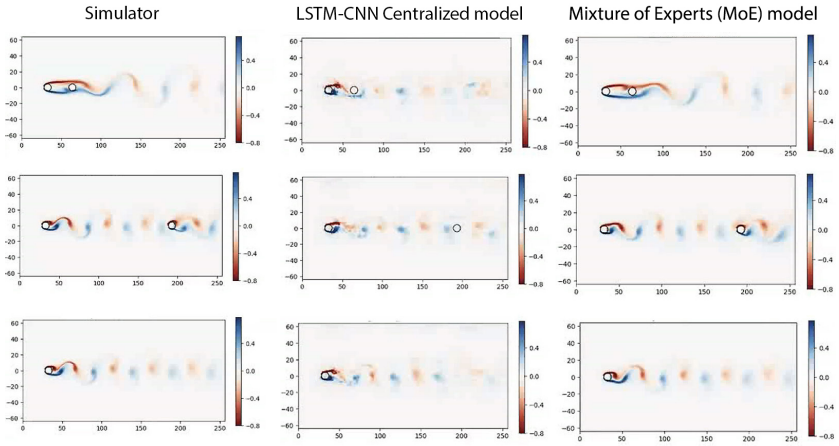

Fig. 3: Testing CFD-simulator and two surrogate models on three different flows. Where the colors refer to the mirrored response of generated fluid along the Y axis (negative & positive)

Therefore, we trained both a centralized model & a MoE model over three distances $(20, 25, 30)$, then we generated from them every possible fluid within the distance range [20..30].

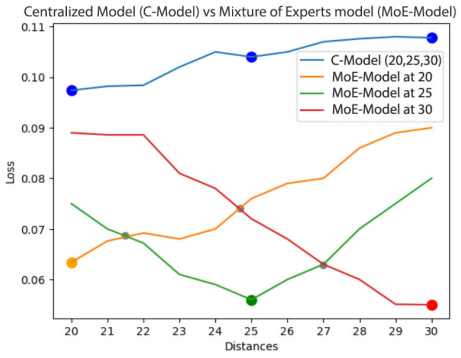

Fig. 4: Comparison of fluid predictions within the distance range [20..30].

As we can see from fig 4, the overall performance of the MoE model outperforms the centralized model. The table 2 provides more details about the performance of the MoE model. It shows the suitable local model to employ based on specific ranges.

We notice that the accuracy decreases as we move away from the trained local models and when we add more to the system, we maintain higher accuracy.

Table 2: Approximate MoE model performance across varying distances.

Use local model of distance 20	Range	Avg loss per frame ↓
MoE(20,25,30)	distance [20..21.5]	≤ 0.07
MoE(20,30)	distance [20..25]	≤ 0.075
Use local model of distance 25	**Range**	**Avg loss per frame ↓**
MoE(20,25,30)	distance [21.5..27]	≤ 0.07
Use local model of distance 30	**Range**	**Avg loss per frame ↓**
MoE(20,25,30)	distance [27..30]	≤ 0.065
MoE(20,30)	distance [25..30]	≤ 0.075

6 Conclusion

In this paper, we introduced surrogate models for fluid dynamics with different architectures. We demonstrated how utilizing simulation support and splitting the surrogate model into two sub-models can help reduce complexity. Both the centralized and decentralized (MoE) models generated flows and simulated fluids in a short time. The MoE model showed greater potential to maintain accuracy and handle data complexity. Future work may involve a dynamic function for MoE in environments with moving obstacles and improving scalability and real-time adaptation.

References

[1] Jiyuan Tu, Guan Heng Yeoh, Chaoqun Liu, and Yao Tao. *Computational fluid dynamics: a practical approach*. Elsevier, 2023.

[2] Philipp Holl, Vladlen Koltun, Kiwon Um, and Nils Thuerey. phiflow: A Differentiable PDE Solving Framework for Deep Learning via Physical Simulations, 2020. http://montrealrobotics.ca/diffcvgp/assets/papers/3.pdf.

[3] Shadman Sakib, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. An overview of convolutional neural network: Its architecture and applications. 2019.

[4] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[5] Alexander IJ Forrester, Neil W Bressloff, and Andy J Keane. Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 462(2071):2177–2204, 2006.

[6] John A Schaefer, Andrew W Cary, and Manas Khurana. Surrogate model and discretization error impacts on the fluid dynamics uncertainty quantification challenge problem. In *AIAA SCITECH 2024 Forum*, page 0706, 2024.

[7] Hongwei Tang, Jean Rabault, Alexander Kuhnle, Yan Wang, and Tongguang Wang. Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids*, 32(5), 2020.

[8] Jean Rabault and Alexander Kuhnle. Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach. *Physics of Fluids*, 31(9), 2019.

[9] Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

[10] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.