Utilizing Convolutional Networks to Mimic Physics-Based Fluid Simulations

Teng A.[1], Duque S.[1], Chandhoke R.[1]

[1]College of Natural Sciences, The University of Texas at Austin, Austin, Texas

EXECUTIVE SUMMARY

The simulation of fluids is a long-standing problem with a strong foundation in physics research. However, performing simulations utilizing these physics equations is extremely time-consuming (Zuo 2010 and Wiewel 2019). We propose the alternative of using a machine learning approach to tackle the same problem but in a faster time and with acceptable accuracy. Through a convolutional neural network, we generated a machine learning model that can effectively replicate a subsequent frame of a fluid simulation but has difficulty representing long-term results.

INTRODUCTION

Simulating fluid flow is a common problem with many applications such as computer graphics or larger scientific simulations. Physics-based fluid simulations are currently incredibly accurate but limited due to their restriction of being computationally expensive (Zuo 2010 and Wiewel 2019). We propose an alternative method of simulating fluid flow using a data-driven approach of convolutional neural networks (Tompson 2016). A neural network can be trained to mimic the calculations performed by physics-based simulators, and once fully trained, the model ideally approximates the results of fluid simulations swiftly with sufficient accuracy.

The process of simulating fluid can contain multiple stages, notably, advection - computing the movement of fluids - and incompressibility - projecting the fluid based on pressure. The stages of advection and incompressibility can be computed through a variety of techniques for solving partial differential equations: of which, the MacCormack Method (MacCormack 1969) for advection and Euler’s equation for incompressibility (Euler 1757) are used throughout the simulation. This is an oversimplification of fluid dynamics that neglects more minor, yet potentially significant factors: such as viscosity and friction. These factors are ignored as they are typically negligible and take away from the focus of primarily comparing physics and data-based solutions.

Physics simulations can additionally be categorized as Lagrangian methods and Eulerian methods (Batchelor 1973). Lagrangian simulations represent fluids as discrete particles and simulates the movement of these particles. On the other hand, Lagrangian simulations represent a grid area fluid travels through; the simulation calculates the movement of fluid passing through each grid cell. This work utilizes solely Eulerian methods for fluid simulations.

Previous work has utilized convolutional neural networks to simulate individual stages of fluid flow - advection and incompressibility - and combined the separate stages to create a general fluid simulation model (Tompson 2016; Kochkov 2019). This work operates outside of the inner workings of fluid simulations and learns solely from the initial and resulting states of the simulated environment.

An alternative method for modeling physics-based nonlinear partial differential equations, such as fluid dynamics, exists by focusing on mimicking the relevant physics equations, instead of a simulation that solves these equations. A model's training can be guided by enforcing realistic physical limitations on the potential outputs of the model (Raissi 2019). For example, in an enclosed system, a model's output can be manually adjusted to ensure that the conservation of mass holds while a purely data-driven approach would hope the model naturally learns this through training.

Our work attempts to similarly mimic a Eulerian fluid simulation using machine learning models. The model used will be based off U-Net convolutional neural network architecture (Ronneberger 2015). Unlike the works of Tompson et all and Kochkov et all (Tompson 2016; Kochkov 2019), the model will treat fluid simulation as a black-box instead of separating the model into two parts: an individual model for advection and also incompresibility. The objective is produce a fluid flow simulation with acceptable accuracy and less computational time compared to a physics-based fluid simulation.

METHODS

Intuitively, we first need a source of data, then a model to process the generated data. The open-source physics-based fluid simulation PhiFlow is used due to its close integration with Python and PyTorch. The PhiFlow simulator acts as the source of the data and the basis of comparison for our machine learning models. From Gaussian random noise, PhiFlow randomly generates 64x64 2D grids of fluid density and velocity, as shown in Figure 1. Furthermore, PhiFlow uses fluid density and velocity fields to perform advection and incompressibility calculations.

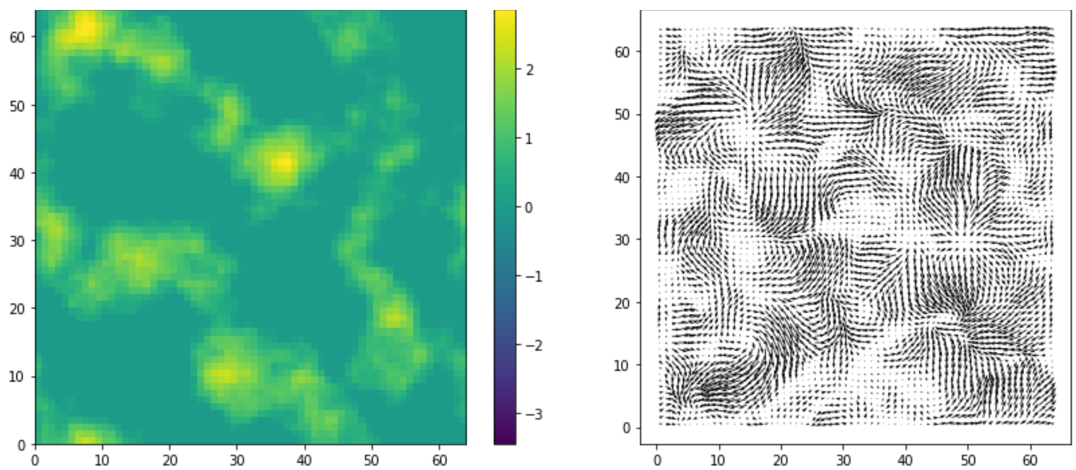


Fig. 1: 2D-Grid representations of fluid density and velocity.

Because of the spatial component of the data, convolutional neural networks were chosen to process the data.

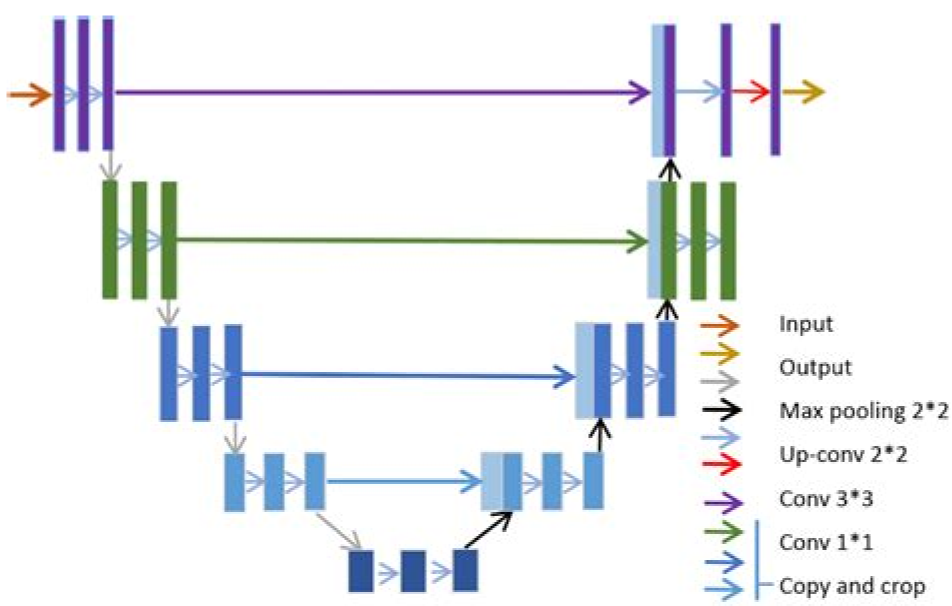


Fig. 2: U-Net CNN architecture structure.

We input multiple 64x64 images of the horizontal velocity, vertical velocity, and density for the convolutional network. The model outputs the next time-step of the simulation in the same format. As a result, U-Net is a promising architecture due to identical input and output sizes (Ronneberger 2015). A U-Net CNN consists of two stages, downsampling and upsampling. The downsampling stage utilizes filters to decrease to spatial size of the data while increasing its number of channels. The upsampling stage mirrors the downsampling; the data's spatial size is increased while its channels are decreased. Additionally, the data from each level of the downsampling stage is propagated forwarded to be concatenated with the same level equivalent in the upsampling stage. The model's general structure is depicted in figure 2.

RESULTS

The U-Net CNN model was implemented using four downsampling and four upsampling convolutional layers. The model was trained over a dataset of 460 samples of 32-frame fluid flow simulations for at most 100 epochs, ending by early stopping with a patience of two. Utilizing a validation dataset of 52 samples, MSE validation loss decreased over the course of training but stabilized above 0.6, as shown in Figure 3.

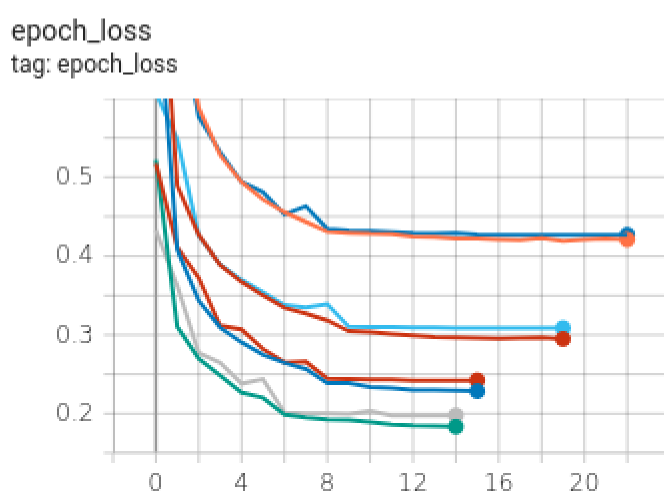


Fig 3: Training and validation loss with starting filter size of 4, 8, 16, and 32. (From top to bottom).

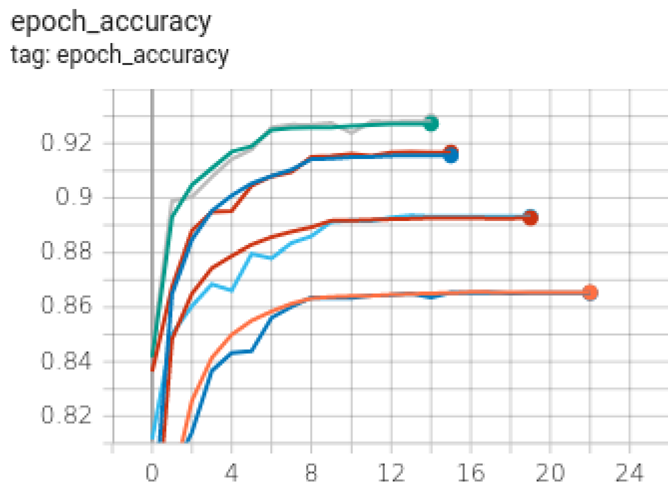
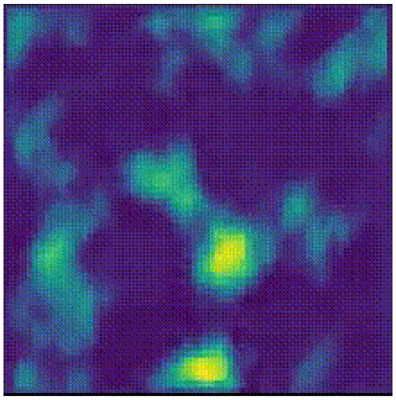
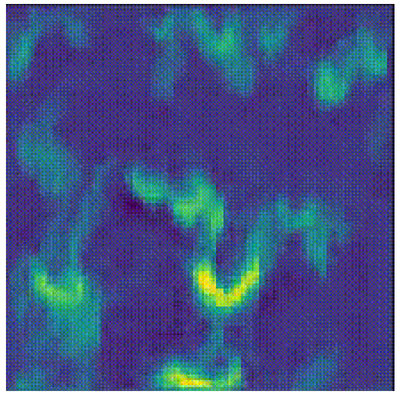
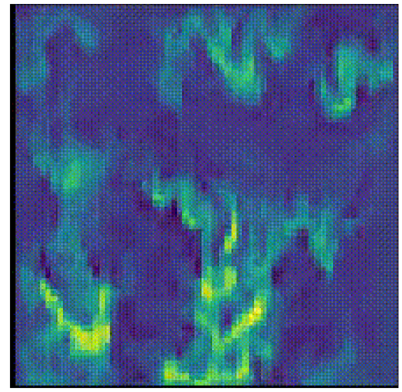
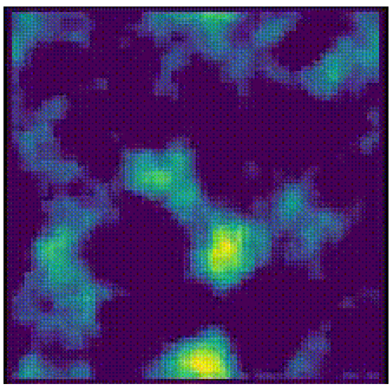
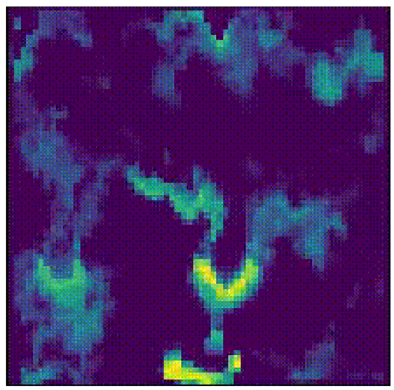
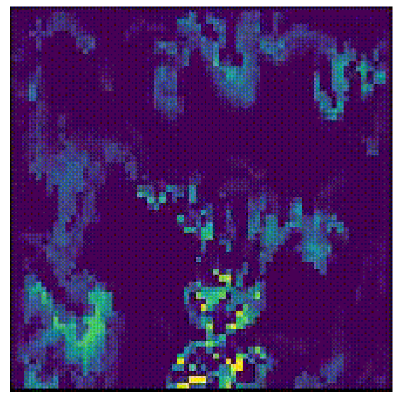


Fig 4: Training and validation accuracy with starting filter size of 32, 16, 8, and 4. (From top to bottom).

The model’s output is solely a single frame after the input frame but multiple subsequent frames can be predicted by self-feeding the model’s output into its input at the cost of rapidly shrinking confidence. This is clearly demonstrated in Figure 4, as the predicted frames diverge more from actual frames the further ahead a prediction is done.

Predicted data frame: 1	Predicted data frame: 2	Predicted data frame: 3
		
Actual data frame: 1	Actual data frame: 2	Actual data frame: 3
		

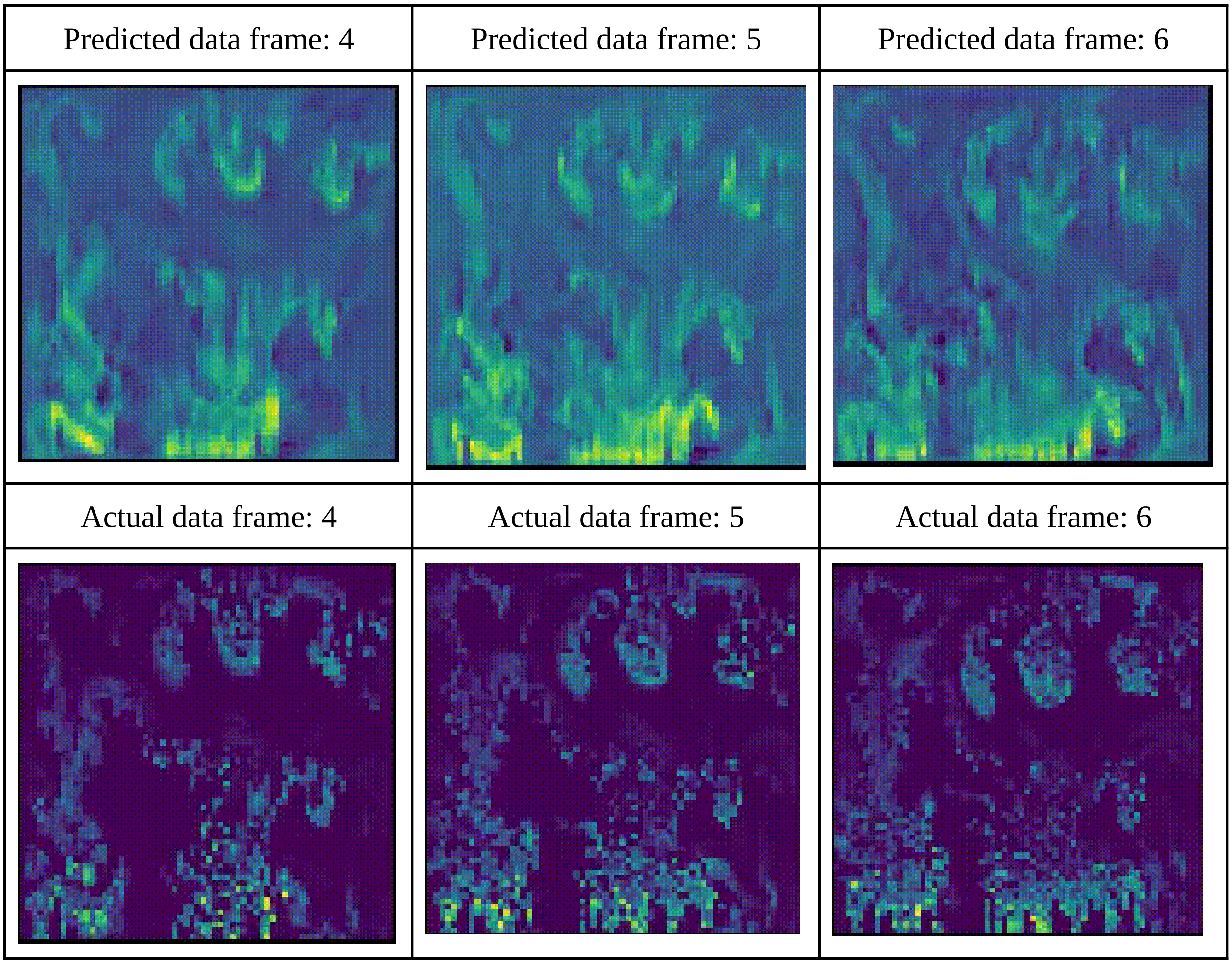
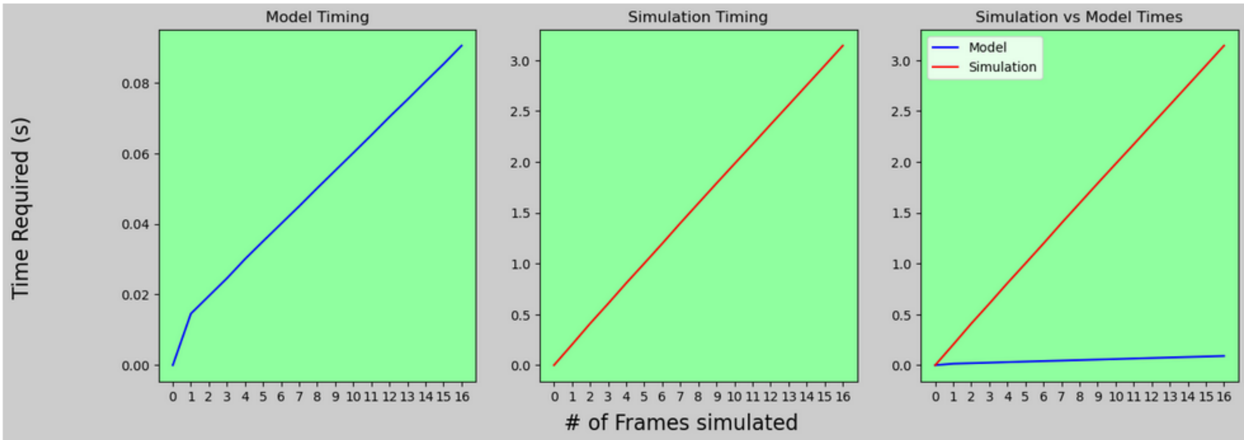


Figure 5: Predicted Frames vs. Actual Simulation Frames.

An advantage of using a neural network approach is a much faster simulation. After running a 16-frame simulation on a CPU, the U-Net CNN was able to finish in just 0.09 seconds, while PhiFlow finished in 3.14 seconds, as seen in Figure 6. This is exemplary of the time-accuracy trade-off and allows the CNN to scale to situations that require quick timing, such as real-time simulation.



Total time of model	Average time of model per frame	Total time of PhiFlow simulator	Average time of PhiFlow simulator per frame
86.7550 ms	5.4090 ms	3088.8727 ms	193.0411 ms

Figure 6: Comparing model and simulator speeds.

When comparing the model's ability to predict long-term fluid simulation with the simulator's actual results for the same time-frame, the model's root mean square error dramatically increases quickly.

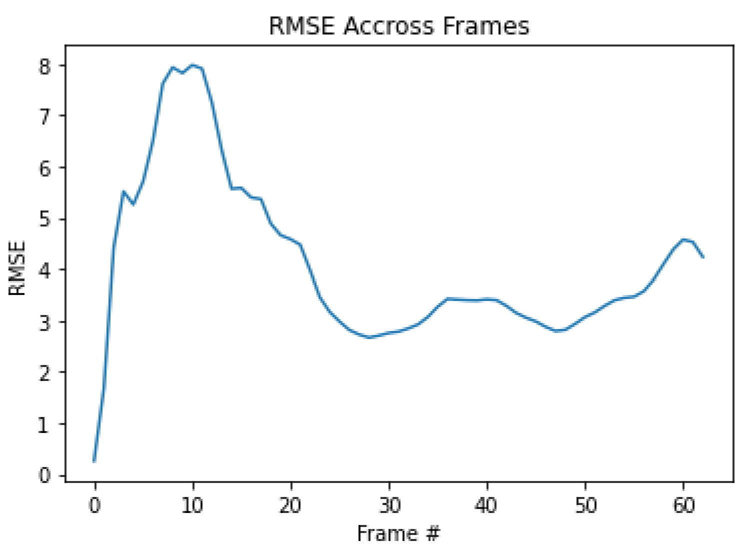


Figure 7: UNet RMSE for long-term simulation.

As seen in figure 7, the simulation accurately predicts the first and, to some extent, second frames but diverges heavily from the expected simulation afterwards. The first and second frames' RMSEs are only 0.260 and 1.677 respectively, while the third frame's RMSE jumps to 4.416 and increases then on. Interestingly, the model's output appears to converge back to the simulation around the 13th frame despite the input frame having a high RMSE itself. Additionally, the peak in RMSE appears in other examples as well but not necessarily on the same frame number, as shown in figure 8.

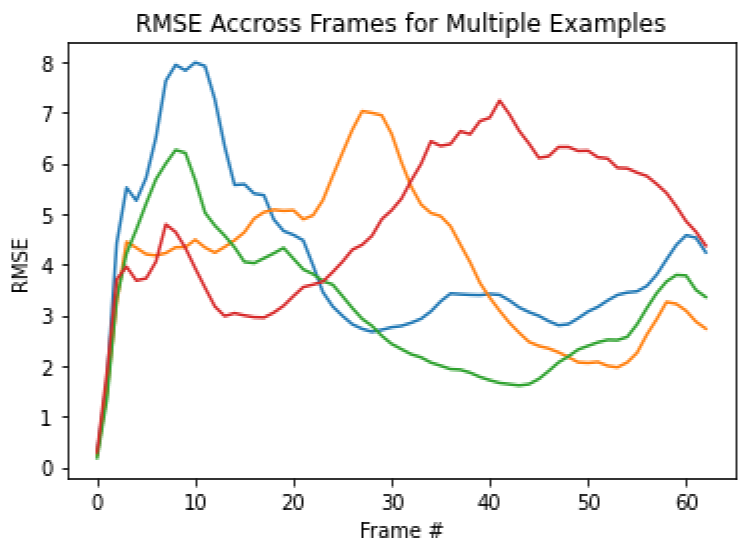


Figure 8: UNet RMSE across multiple testing samples.

DISCUSSION

The 32 filter size U-Net model appears to accurately predict a single time-step frame with roughly 93% accuracy. However, upon visual inspection, The U-Net CNN model is largely failing to correctly capture the fluid simulation over a longer period of time.

A possible cause can likely be due to the model's loss function. The model calculates loss solely on a single subsequent frame, while it may be better for the model to predict multiple frames ahead, by self-feeding its output to itself, then compare the predicted frames with the simulation frames. As a result, the model should then be punished for mispredicting the long-term effects of the simulation.

In general, the model alters the input frame for far less than the actual simulation. This could potentially be the result of the model overfitting the tail-end frames of every simulation, as after roughly 25 frames of the actual simulation the fluid settles and less movement between time steps exists. In fact, the model's effectiveness being dependent on the phase of the simulation is evident in the single frame simulation during training and appears consistent across samples. The model finds difficulty in mimicking the initial frames, improves during the middle frames, and is best in predicting the final frames of the simulation.

A potential cause of this could be the simulation being more dynamic in earlier frames, which becomes a problem if the model is overfitting the less dynamic ending frames. Alternatively, this may be the result of differing importance of the stages of fluid simulation. In the earlier frames, incompressibility plays a major role in the simulation while it becomes less impactful as the simulation progresses. Since we treat the simulation as a black box encompassing both incompressibility and advection, the model may be failing to capture the shifting importance of incompressibility and advection. As a result, two potential approaches to this problem arise. The earliest frames of the simulation can be removed in hopes of having the model focus on capturing the advection aspect of the simulation. Alternatively, two models could separately capture incompressibility and advection and be combined for simulation while potentially maintaining a faster computational time than the physics-based simulator, such as the approach conducted by Tompson et all (2016). Overall, the simulation may be too complex for the model to accurately capture.

The speed advantage of the data-driven simulation over the physics-based simulation is very promising and fulfills the primary motivation of the project. Overall, the potential of data-driven simulations is evident but further work needs to be done to improve the accuracy of the model for practical use.

CONCLUSION

While the data-driven simulation presented here is almost 35 times faster than physics-based models, it's hard to overlook the issues in accuracy that arise. The data-driven approach diverges from the expected result in as little as three frames, resulting in a completely different output.

REFERENCES

Batchelor, G.K. (1973), *An introduction to fluid dynamics*, Cambridge University Press, ISBN 978-0-521-09817-5

Euler, Leonhard (1757). "Principes généraux du mouvement des fluides" [The General Principles of the Movement of Fluids]. *Mémoires de l'académie des sciences de Berlin* (in French). *11*: 274–315.

Kochkov, D., Smith, J. A., Alieva, Ayya., & et all. (2021). Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*. (118)21. https://doi.org/10.1073/pnas.2101784118

MacCormack, R. W. (1969), The Effect of viscosity in hypervelocity impact cratering, AIAA Paper, 69-354.

Pradeepkumar Girija, Athul & Pednekar, Shourav. (2015). Incompressible Euler Solver using Artificial Compressibility Method. 10.13140/RG.2.1.2928.6569.

Raissi, M., Perdikaris, P., Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *(378)*686-707. https://doi.org/10.1016/j.jcp.2018.10.045.

Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: a machine learning approach for precipitation nowcasting. *In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (*NIPS'15*)802–810. https://dl.acm.org/doi/10.5555/2969239.2969329

Tompson, J., Schlachter, K., Sprechmann, P., & Perlin, K. (2016). Accelerating Eulerian Fluid Simulation with Convolutional Networks. In *arXiv [cs.CV]*. https://dl.acm.org/doi/pdf/10.5555/3305890.3306035

Wiewel, S., Becher, M., & Thuerey, N. (2019, May). Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum* (Vol. 38, No. 2, pp. 71-82).

Zuo, W., & Chen, Q. (2010). Fast and informative flow simulations in a building by using fast fluid dynamics model on graphics processing unit. *Building and environment*, *45*(3), 747-757.