# Predicting Pipe Flow With Deep Learning

Joshua Neronha, Alexander Koh-Bell, Jason Gong

https://github.com/joshuaneronha/pipeflow-nn

## Introduction

When engineers are faced with computational problems that are too difficult to analytically solve, they often turn to finite element methods (FEM), which numerically solve the underlying constitutive equations describing the physics. While FEM methods are accurate and effective tools for simulating behavior in fields ranging from optics to fluid mechanics, they are very computationally expensive and often take hours, if not days, to converge at a solution.

Deep neural networks may be particularly well-suited to solve this problem because of the nonlinear nature of many engineering computations and their ability to detect patterns and features. We present a convolutional autoencoder neural network architecture that can predict the velocity and pressure around an obstruction in a given pipe geometry, a common elemenatry problem in fluid mechanics, as a starting point in our research.

## The Data

We use the DeepCFD dataset (1), which contains two sets of images. The first, shown below in Figure 1, represents a given randomly-generated obstruction geometry. It is a two-dimensional representation of pipe flow and can be considered as looking at a cylindrical pipe from the side. Then, for each geometry, there are three plots showing the horizontal and verticial velocities in addition to the pressure distribution. We train our network to generate these three plots for a given geometry.

**Figure 1:** Sample geometry of pipe obstruction. Black indicates empty space while white indicates the presence of some obstruction.

## Methodology

We choose a convolutional autoencoder architecture because we wish to generate a new image (the flow properties) based on features from the input (in our case, the geometry of the pipe). We follow the general architecture presented in Hajgató et. al (2) which uses three convoltuonal layers as an encoder followed by a dense layer and then three separate transpose convolution (often called a "deconvolution"), one for each fluid flow property, to generate the flow fields we hope to see. Figure 2 serves as a visualization of the architecture.

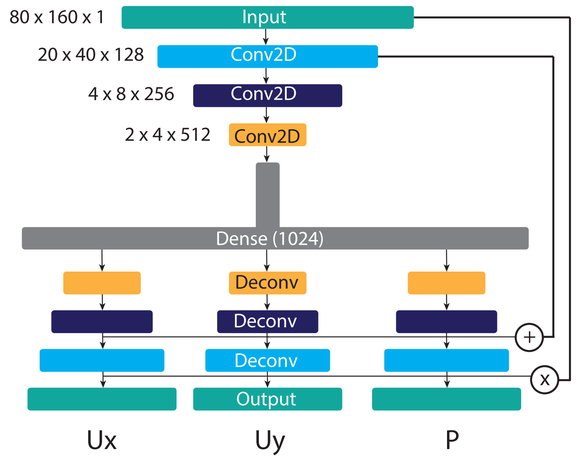


**Figure 2:** Network architecture used in our model. Each color refers to a different intermediate tensor size throughout the flow of the network.

In addition to the standard network architecture detailed above, we also implement a few specialized techniques disucssed in Hajgató et. al. Most importantly, we add the output of the first convolution operation to the output of the second deconvolution layer, which serves as a "residual connection" to provide information from early on in the encoder to the network later in the decoder. We also multiply the initial geometry by the final output before computing loss; this serves as a mask because any pressure or velocity data inside the obstruction is mathematically and physically useless. We use absolute error for pressure and square error for both Ux and Uy while optimizing our model.

## Results

We tuned the hyperparameters of our model over a wide variety of learning rates, filter sizes in each convolutional layer, epochs, batch sizes, and loss functions and quantified the loss in order to determine which parameters maximized the accuracy of our model. Below are Tables I and II, which show our best results and the hyperparameters used to achieve them.

| | Error |
|---|---|
| U_x | 2.80% |
| U_y | 18.14% |
| P | 11.31% |
| Overall | **11.05%** |

| | Value |
|---|---|
| Learning Rate | 0.004 |
| Batch Size | 32 |
| Epochs | 40 |
| Optimizer | Adam |

**Tables I and II:** Table I on the left shows the best-case accuracy for each of our three variables. Table II on the right shows our parameters.

Because we have built a convolutional autoencoder model that generates images, we can generate plots of Ux, Uy, and P for a test geometry and plot it against the ground truth results generated by CFD methods, examples of which are shown in Figure 3 below. The image shows our Ux prediction is best and P worst.

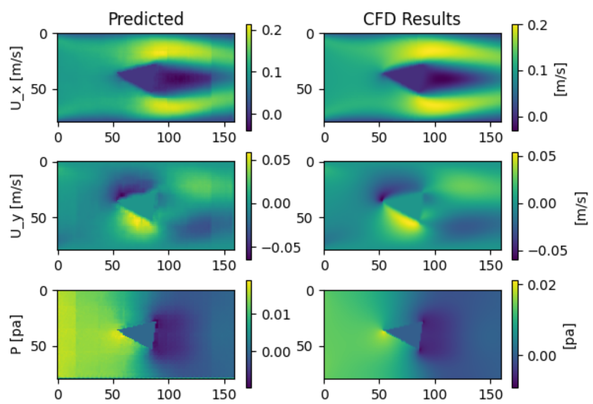


**Figure 3:** We plot the results of a test geometry; on the left are the model-generated results, compared to the ground truth results on the right.

## Discussion

Overall, we surpassed our baseline goals by achieving 85% accuracy compared to COMSOL simulations with massively improved runtime; after training the model, which takes no more than half an hour, results for a given geometry can be generated within 0.072 seconds compared to the 52.51 seconds on average that an FEM software such as COMSOL Multiphysics woud take. This means our network is 729x faster!

This is a fantastic improvement in efficiency, and models like these may prove useful to fluid mechanists who need a general sense of flow behavior without needing to know every detail. For example, in Figure 3, Ux, Uy, and P predictions generally agree very well between the model's predictions and CFD results. While the accuracy is not perfect, particularly in Uy and P, the general flow behavior is preserved: a pressure drop across the geometry, with high pressure areas at the front of the body. This is the important part of simulation for many fluids researchers, so our model may be quite useful!

Interestingly, even after extensive ablation testing, the hyperparameters provided by Hajgató et al. proved the most accurate (except for learning rate), even for a different dataset, indicating a very robust model that may be useful for a wide variety of pipe flow problems. There are many possible ways in which we might improve this model in the future, for example increasing the size of our relatively small training set and designing new situations such as pipe expansions and testing how the model performs in new flow situatitons like these.

## References

(1): "DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks." arXiv:2004.08826 [physics.comp-ph]
(2): Hajgató et al. Predicting the flow field in a U-bend with deep neural networks. arXiv:2010.00258 [cs.LG]

Thank you to Professor Chen Sun and all the TAs of CS1470 who enabled our success with this project!