Fast Estimation of Airfoil Pressure Fields using Deep Convolutional Neural Networks

Team Members:

Sam Pfrommer (PennKey: `spfrom`; Email: `spfrom@seas.upenn.edu`)
Arjun Guru (PennKey: `aguru`; Email: `aguru@seas.upenn.edu`)
Dhruv Karthik (PennKey: `dhruvkar`; Email: `dhruvkar@seas.upenn.edu`)
Jonathan Mak (PennKey: `jhwmak`; Email: `jhwmak@seas.upenn.edu`)

Assigned Project Mentor:

Barry Plunkett

Team Member Contributions:

Team Member	Contributions
Sam Pfrommer	OpenFOAM data generation Training architecture & codebase GuoCNN implementation
Dhruv Karthik	U-Net implementation Tensorboard setup & visualization
Arjun Guru	Custom loss functions Problem research
Jonathan Mak	Google cloud compute setup Figure and slides creation

Code Submission: Code is available on GitHub: https://github.com/spfrommer/520FinalProject

Abstract

Determining the aerodynamic properties of various mechanical structures is a common problem across many engineering disciplines. In this study we restrict our focus to predicting steady state pressure fields surrounding airfoils, which are the key determinants of important metrics such as lift/downforce and drag. Conventional Computational Fluid Dynamics (CFD) methods are slow and prohibit the development of interactive design tools with immediate feedback. To accelerate this process, we explore two different CNN architectures, U-Net and Guo-CNN, as well as two different input representations, binary mask and signed distance function (SDF). We find that these convolutional neural network approximation models estimate pressure fields two orders of magnitude faster than CFD analysis while maintaining qualitatively similar pressure field outputs. Our trained models allow for rapid analysis and are a step towards the eventual development of interactive design tools for airfoils.

1 Introduction

Airfoils are surfaces designed to aid in lift or downforce by making use of the air currents through which they move. A key problem in aerodynamics-related design is the estimation of pressure in the region surrounding an airfoil. This is necessary to determine the lift and drag performance of the airfoil. Our research specifically focuses on estimating pressure fields for steady-state laminar flow at low Reynolds numbers. In this scenario, the airfoil is subject to fluid flowing smoothly in parallel layers, with no disruption between these layers. The flow may exhibit differences from point to point, but these differences do not vary with time. This scenario often occurs at the front of streamlined bodies and is highly applicable to flight and especially automobile aerodynamics, which is our focus in this study [3].

Computational Fluid Dynamics (CFD) analysis, the current state-of-the-art, involves iteratively solving a set of partial differential equations simulating the interaction of fluids with the airfoil. CFD simulation is computationally extensive and time consuming, often taking hours for more complicated systems. This prevents interactive design and rapid design space exploration [5]. For this reason, fast approximation models are often used instead of CFD simulation [2], and in this paper, we explore CNN-based approximation models for predicting the pressure field surrounding an airfoil in the scenario of steady-state laminar flow.

2 Related Work

Researchers from the Georgia Institute of Technologys published a paper investigating the application of convolutional neural networks for another aerodynamic prediction task, airfoil lift coefficients. They developed a network that adapted the LeNet-5 architecture, a digit recognition network, that took a cross section input image and outputted a variety of aerodynamic indicators used to compute the lift coefficient [6].

In other related work, researchers from Stanford University conducted a study using fully convolutional neural network for depth estimation [4], whereby they outputted real-value labels for each pixel of the image. The techniques implemented are relevant for our purposes, as for our aerodynamics prediction task similarly requires taking in an image and outputting pixel-level labelling on top.

Our paper additionally builds on work done by Guo et. al. in fast estimation of the velocity field surrounding an airfoil [5], also focusing on the scenario of steady-state laminar flow. We use similar methods but instead attempt to estimate the pressure field surrounding an airfoil. As mentioned earlier, estimation of the pressure field is highly applicable for the calculation of the lift or downforce generated by an airfoil.

3 Data Set

Nearly 1500 different airfoil geometries were downloaded from the UIUC Airfoil Database [1]. For each geometry, 10 different test cases were randomly generated, with chord length varying between 0.3 and 1 meters and angle of attack varying between -25 and 25 degrees. These ranges are appropriate for the kinds of airfoils frequently used in race car aerodynamics packages. Thus, we have a total dataset size of 15000.

Output Labels (pressure field): To generated ground truth pressure data, we used the open source CFD simulation OpenFOAM. Since we were looking at race car airfoils, the freestream air velocity was set to 30mph, roughly consistent with car speed in cornering where aerodynamic performance is critical. This corresponds to a relatively low Reynolds number of around $500,000$. OpenFOAMs pressure fields are outputted in terms of kinematic pressure, which describes pressure changes due to particle motion and is measured in units of m^2/s^2. Unstructured meshes were generated using Gmsh and solved using the incompressible flow solver simpleFoam.

Inputs (Binary Mask and SDF): We train our model on two alternative types of input feature maps. One representation we applied is a binary mask. In this structure, all pixels inside the airfoil are zero, and all other pixels are 1. An example can be seen in Figure 2 of the appendix.

As another representation of the airfoil geometry as a 256x256 input feature map, we applied a signed distance function (see Figure 3 of the appendix for reference). The goal behind this representation was to introduce more information about the distance of each point in the feature map from the border of the airfoil, with the ultimate goal of giving a trained CNN greater intuition on distance from the airfoil and its effects. Specifically, the value of each cell in the input feature map was determined by its Euclidean distance from the border of the airfoil; points inside the airfoil had negative distance values, and points outside the airfoil had positive distance values. To formalize this, let us define a set Z which is the set of points (i,j) in 2D that give the geometric boundary of the image, ie $Z = \{(i,j) \in R^2 : f(i,j) = 0\}$ where f is a level set function such that, $f(i,j) = 0$ iff (i,j) is on the boundary, $f(i,j) < 0$ iff (i,j) is within the boundary, $f(i,j) > 0$ iff (i,j) is outside the boundary. The SDF $D(i,j)$ associated with a level set $f(i,j)$ is then defined by $D(i,j) = \min_{(i',j') \in Z} |(i,j) - (i',j')| sign(f(i,j))$, similarly as in [5].

4 Problem Formulation

Data Preprocessing: Raw pressure fields and inputs were downsampled to a 256x256 square with the leading edge of the airfoil shifted towards the flow origin to capture trailing airflow characteristics. Furthermore, both pressure fields and SDF image values were scaled by an empirically determined factor of 400 in order to improve backpropagation.

Performance Measures: To train our network, we implemented a modified Mean Squared Error (MSE) loss that only considers pixels outside of the airfoil boundary. S_i denotes the set of pixels (denoted by their coordinates) not on the airfoil within image i. In other words S_i denotes the surrounding pixels of airfoil i. P^i denotes the predicted pressure field for airfoil example i, and A^i denotes the actual pressure field for airfoil example i. m denotes the batch size.

$$\frac{1}{m}\sum_{i=1}^{m}\frac{1}{|S_i|}\sum_{(j,k)\in S_i}(P^i_{(j,k)} - A^i_{(j,k)})^2$$

We additionally chose to report a similarly modified Mean Absolute Error (MAE), since it gives better intuition for the actual spread of the data relative to the max/min values of the pressure field.

5 Algorithms

Convolutional Neural Networks are ideally suited to the task of image classification and feature extraction. Each layer of a CNN is composed of learnable kernel filters (matrices that are convolved with the input domain), allowing for relevant features to be extracted from images. They succeed traditional neural networks due to the reduction in the number of parameters required for learning, and the reusability of weights (we only learn weights for each filter, whereas traditional NN's would learn weights for each pixel).

The basic components of a CNN (convolutional, pooling, and activation functions) operate only on *relative* spatial coordinates. Thus, the activation of locations in higher layers can be traced back to lower layers by simply backtracking the series of spatially relative transformations applied to them, commonly known as their *receptive fields*.

This property allows us to use the general class of network architectures known as **Fully Convolutional Neural Networks** (FCN's). These differ from standard Convolutional Neural Networks by the replacement of fully connected layers with **deconvolutional layers**. These allow for the feature-map/heatmap representation of the input data at higher levels to be upsampled into an image, resulting in the sparse classification of every pixel in the image. We use variations of the classic FCN architecture, known as the **Guo-CNN**, and the **U-Net** (Figure 1).

1. The **Guo-CNN** has been used effectively in the past for estimating velocity fields around cross sections of cars. We modify its architecture to estimate pressure fields by removing

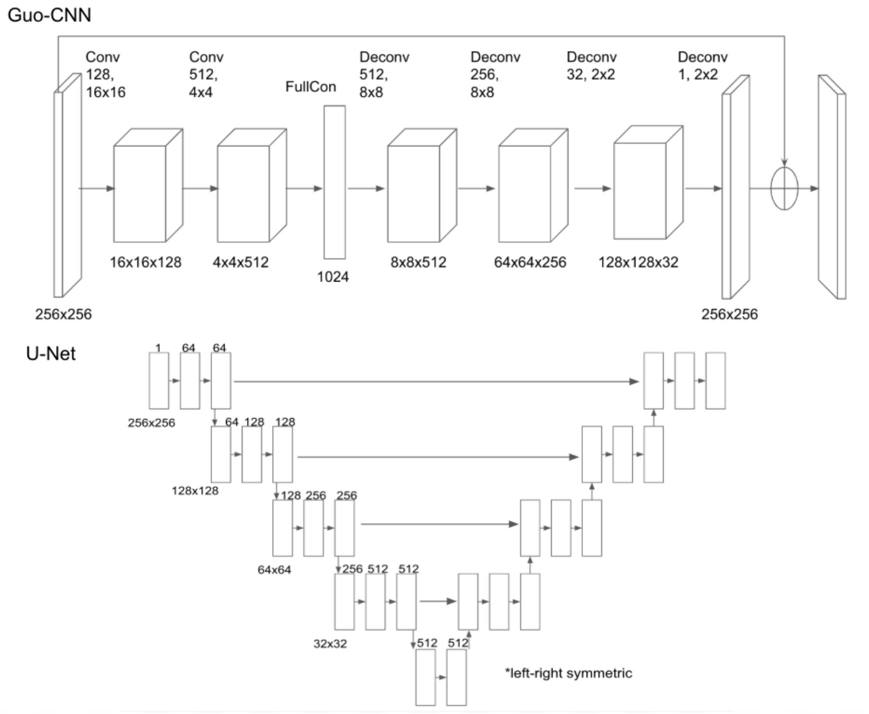

Figure 1: Guo-CNN & U-Net model representation

layers that estimate the x and y components of the velocity. This architecture is notable as it uses no max-pooling layers, motivated by the need for **all** activations to propagate through all layers, whereas max-pooling would select the only highest performing activations from sections of the image to create a more high-dimensional feature representation of the input data. We use the ReLu activation function.

2. The **U-Net** is popular in biomedical image segmentation. We were motivated to use our own variation of this architecture due to its published performance in extracting feature representations of the input data at multiple scales. This allows for sensitivity to minor changes in the rotations and scaling of our input data, which is desirable in precisely estimating pressure values at every pixel around the airfoil. It involves several stacked convolutional and max-pooling layers, combined with a concatenation of lower level input features to higher level features at every scale, so as to avoid the issue of a vanishing gradient that is prone to deep CNN's. On every convolutional stack (displayed by the three consecutive layers at every scale in Figure 3, we use Batch Normalization and a ReLu activation before each convolutional layer).

All models are implemented using PyTorch and trained on a Tesla K80 GPU using a Google Cloud Compute virtual machine instance. Training times varied from 6-12 hours per model.

6 Experimental Design

Tested models: We trained four different models in total: Guo-CNN on binary mask, Guo-CNN on SDF, U-Net on binary mask, U-Net on SDF.

Test, Train, Valid Splits: Across 15000 image dataset, we created random splits with 60% Training, 20% Validation, and 20% Testing.

Optimization Function: We use the **Adam** Optimizer. Adam is able to maintain a per-parameter learning rate, that eliminates the risk of sparse gradients that may arise from having a binary mask that composes a small fraction of the input image. Across several trial epochs, it also showed empirically faster optimization performance when compared to other methods such as RMSProp.

Batch Size: We use a batch size of 8. The U-Net is especially heavy on GPU memory because of the residual concatenations it makes, so we chose empirically chose this batch size to minimize the trade-off between speed and GPU memory.

Num Epochs: We trained the Guo-CNN for 100 epochs, and the U-Net for 10 epochs in each instance of the models. The reason for this is further explained below.

7 Results

A selection of the results can be seen in Figures in the appendix, and the table below provides a summary:

	Train MAE	Validation MAE	Test MAE
Guo-CNN: Binary (Figure 2)	18.484	18.948	18.244
U-Net : Binary	18.628	18.736	18.112
Guo-CNN: SDF	18.220	18.504	17.864
U-Net : SDF (Figure 3)	18.632	19.124	18.472

Test Performance: Overall, the Guo-CNN with SDF performed the best on the test set, with a margin of 0.248 from the second largest error, which was the U-Net with Binary Mask inputs. The main reason we used the SDF representation was to "echo" the structure of the binary mask at multiple scales across the entire input image. However, the U-Net structure is already designed to extract features at multiple scales, which is why it has a better MAE on all fronts with the standard Binary Mask Input than with the SDF Input - on which it likely finds correlations between artifacts generated by the SDF representation of the image. On the other hand, the Guo-CNN does not have this property (it only considers the input at a single scale to preserve activations), it performs better on the SDF input representation. In fact, it is likely the Guo-CNN's property of using no pooling layers that results in the lowest overall MAE - the per pixel activations are propagated throughout the network, so information loss throughout the network is minimized.

One interesting result is that while validation errors were appropriately higher than training errors, test errors were surprisingly slightly lower. One possible explanation is that in the random training/test/validation split (which was consistent across model trainings) the test set happened to receive an easier to predict set of examples. However, this could also have to do with noise in the input images; the discretization of the flow region into distinct cells is arbitrary and difficult for the CNNs to predict. Further steps could involve first preprocessing images to remove this noise, e.g. with blur.

Training Times and Memory Usage: The U-Net implements a residual block that copies activation's from each downsampling step, to the corresponding upsampling step in the same scale. Concatenating two large PyTorch tensors with several channels is an extremely memory intensive process, as it requires PyTorch to maintain the Autograd tracking feature for every element in the tensor. However, this results in a much faster convergence rate in terms of epoch. As seen in Figure 5, we see that the U-Net converges to an MAE of approximately 18.6 in only 5 epochs, whereas it takes the Guo-CNN as long as 20 epochs to reach the same point. In terms of training times, the U-Net takes roughly 5 times longer to train per epoch than the Guo-CNN. This is the reason we chose a larger batch size for Guo-CNN than for the U-Net

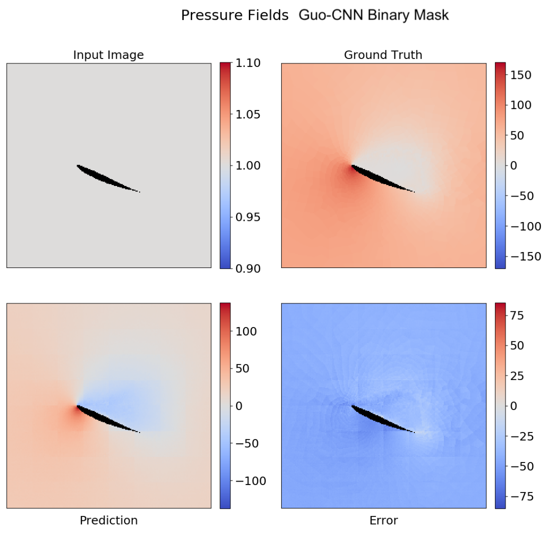

Figure 2: Guo-CNN with binary mask. Note the square "chunks" in the error plot.

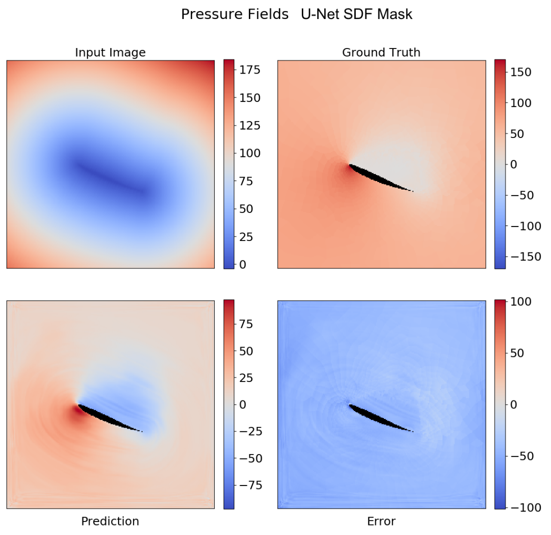

Figure 3: Guo-CNN with sdf mask. Note the eliptical ripples in the error plot.

8 Discussion and Conclusion

We ultimately had roughly equivalent performance with all four methods, which is most likely a result of a sub optimal dataset. As can be seen from the output images, the networks seemed to qualitatively predict the correct pressure "deltas" from ambient pressure quite well–potential future work could include subtracting the far field pressure before training. Preliminary tests we performed suggest this could reduce mean absolute error by over 50%. Another interesting avenue of research would be applying transfer learning to adapt this network for multi-element airfoils, which are computationally much more difficult for CFD solvers. Despite our trained CNNs deviating from the CFD predictions in some errors, we still believe that this research is a good step towards developing fast airflow approximation algorithms to assist mechanical design. Using the CNN prediction, a CFD solver could be warm started to converge far faster than with an arbitrarily initialized flow field, allowing designers to rapidly design, test, and refine their aerodynamic structures.

Acknowledgments

We would like to acknowledge the guidance provided by our TA mentor, Barry Plunkett, as well as Keshav Vendula for his assistance in configuring the CFD solver.

Appendix

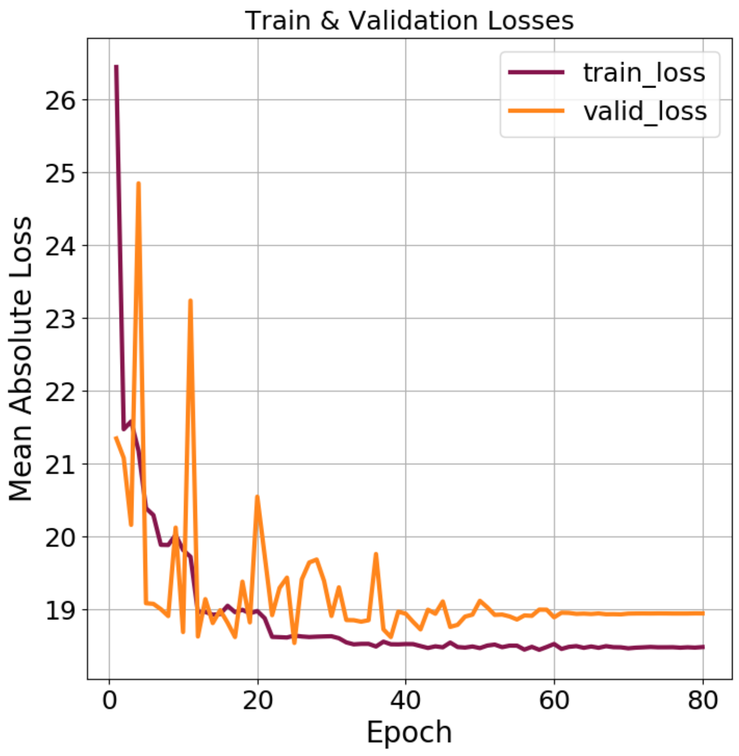

Figure 4: Mean Absolute Loss for Guo-CNN with Binary Mask

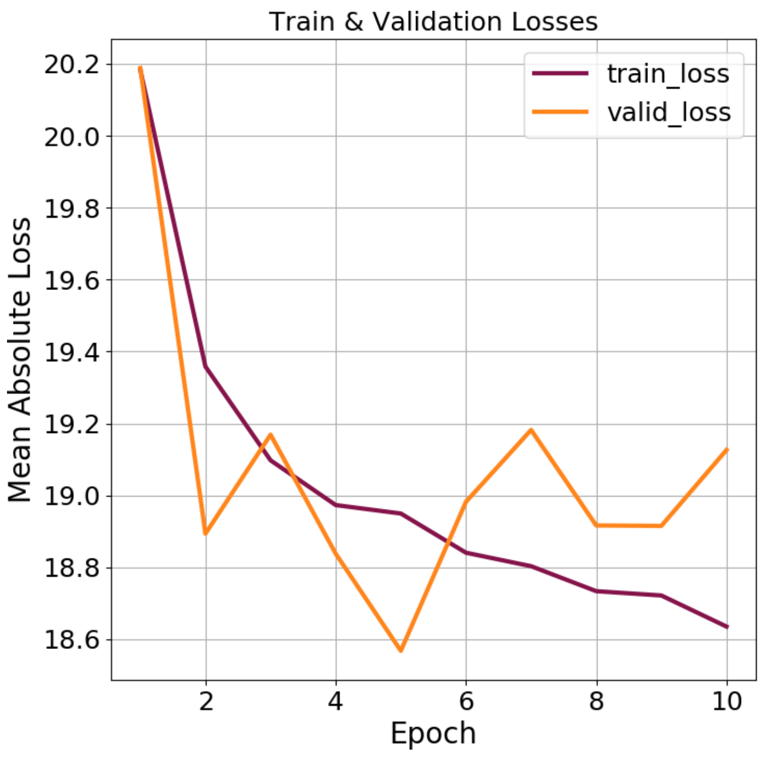

Figure 5: Mean Absolute Loss for U-Net with SDF Mask

References

[1] UIUC Airfoil Data Site.

[2] M. Ahmed and N. Qin. Surrogate-Based Aerodynamic Design Optimization: Use of Surrogates in Aerodynamic Design Optimization. *International Conference on Aerospace Sciences and Aviation Technology*, 13(AEROSPACE SCIENCES):1–26, May 2009.

[3] John D. Anderson. *Computational Fluid Dynamics*. 1995.

[4] Yokila Arora, Ishan Patil, and Thao Nguyen. Fully Convolutional Network for Depth Estimation and Semantic Segmentation. page 7.

[5] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional Neural Networks for Steady Flow Approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 481–490, San Francisco, California, USA, 2016. ACM Press.

[6] Yao Zhang, Woong-Je Sung, and Dimitri Mavris. Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient. *arXiv:1712.10082 [cs, stat]*, December 2017. arXiv: 1712.10082.