Name： Yang Chen

School： Cranbrook Schools

State： Michigan

Country： United States of America

Instructor： Shengjia Zhao

Title： Deep Neural Network with Active Learning: Automated Engineering Design Optimization for Fluid-Dynammics Based on Self-Simulated Dataset

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：  　　指导老师： 

2018 年 09 月 12 日

Deep Neural Network with Active Learning: Automated Engineering Design Optimization for Fluid Dynamics Based on Self-Simulated Dataset

Yang Chen

Abstract

Designing shapes with high fluid-dynamic performance is an important engineering problem. Traditionally experts design shapes based on educated estimations, and use expensive simulations to verify their performance. This expensive process, both in regards to time and space, can only explore a limited number of shapes, which is a situation that could lead to suboptimal design. In this research, we apply a self-designed deep learning architecture to predict the performance of various shapes and search for better shapes by optimizing the learned prediction function under certain restrictions. The major challenge is that deep learning requires a large number of training data, which is costly to simulate. To remedy this problem, we use active learning to explore shapes that the deep network finds promising. This significantly reduces the number of data samples required. Based on these methods, we find shapes with extremely low drag with no human domain knowledge and modest computation overhead.

Keywords: Engineering Optimization, Fluid Dynamics, MATLAB Simulation, Deep Learning, Linear Regression, Cross Validation, Active Learning and Efficient Automated System.

Contents

Abstract……3

1. Research Background and Its Significance……5
    a. Fluid-Dynamics……5
    b. Wind Tunnel Test……7
2. Research Objectives and Points of Innovation……9
    a. Simulating mesh structure……9
    b. Our Targets……9
    c. What is new in our research……10
3. Methodology……11
    a. Physical Simulation with MATLAB……12
    b. Training the Model……14
    c. Automated Search for Drag-Reduced Shapes……23
4. Results……28
    a. Deep Network Regression Performance……28
    b. Automated Engineering Optimization Performance……31
5. Discussion……33
    a. Summary and Expectations……33
    b. Proposed Future Extensions……35
6. Conclusion……36
7. Acknowledgements……37
8. Reference……39

1 Research Background and Its Significance

1.1 Fluid-Dynamics

Fluid-dynamics is a subject that focuses on the study of moving air or fluid[1]. It especially investigates the properties of interaction between fluid environment and solid object that moves through the environment[2]. It is important because of its vital applications in fields such as aerospace engineering, vehicle production, material science and even architecture design[3]. By studying the effects of fluid moving past a solid object, engineers can optimize their designs of aerodynamic machines. One core objective in this process is to minimize the drag of solid objects under a number of realistic restraints[4].

Traditionally researchers optimize aerodynamic design by estimation. Experts design shapes based their experience and verifies them in a simulator or wind tunnel. Based on simulation results, they pick the optimal shape that has minimum drag. However, such method requires personal experience, which means only experts are capable of effectively finding the design close to optimized, and have it go through certain tests to confirm the results. Also, such results may not even be accurate after several rounds of searching. In general, the overall relationships between the shapes and drags are unknown under realistically given restrictions.

To tackle this problem, we would like to design automatic algorithms that explore and design shapes with desired aerodynamic performance. (Eismann et al) uses Bayesian optimization to accomplish this goal. In this work, researchers replace the costly design optimization methods which exhaust every required sample with a model-based approach. Compared to the usual random search[5], the Bayesian model usedf in this research requires much

less sample[6]. Simply put, although this late research in engineering optimization utilizes a data-based model to ease the complex enumeration process of samples, while applying Bayesian optimization to find desirable aerodynamic design. However, this approach requires a huge number of samples to accurately estimate drag, and fails when sample size is insufficient. For systems with a high degrees of freedom, it often takes an exponential number of samples to evaluate the consequence of each possible design choice. This can be prohibitively expensive.

This work solves this problem by active learning: we use a deep network to model the drag, and draw samples from regions where the deep network believes to be optimal. We only explore a very small number of possible **design choices**, and find shapes with rather **low drag under given design constraints**.

By design choices, we mean a way to shadow the three dimensional shape into a two dimensional graph, and picture a surface based on the following way. We first set an original point of the two-d graph, and start from delta = 0 which is a perpendicular dimension. Then, we divide the circle around the original point into eight evenly divided dimensions. Each dimension would get its own length value. Then, we fit a smooth curve to connect each end of the dimensions (the end that is opposite from the original point). Detailed explanation of the method will be made further explicit below in methodology section.

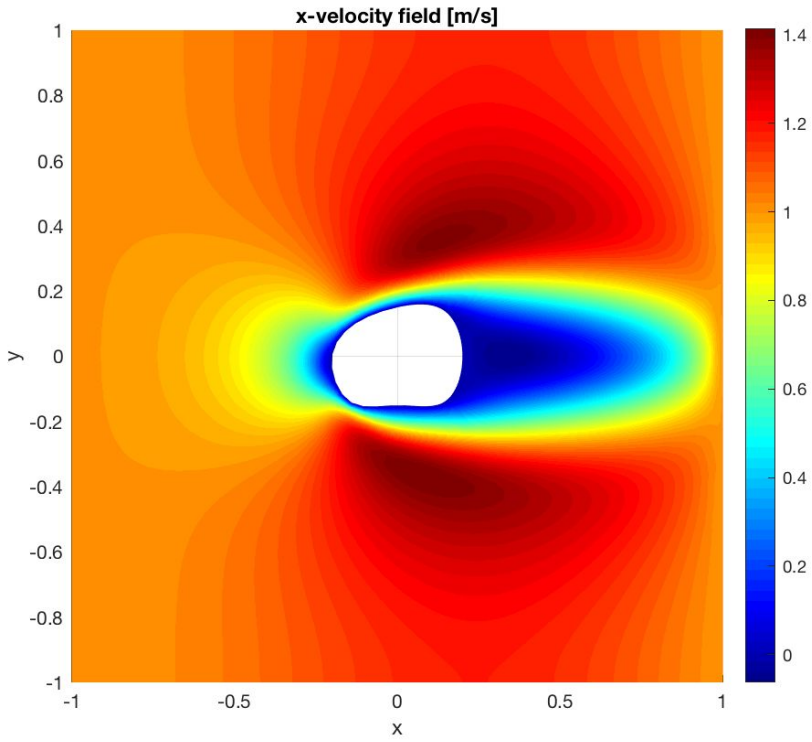

Figure 1: Graph that is Drawn from the Dimensional Data Produced

As Figure 1 shows above, our MATLAB system would produce a simulation as such for further training.

1.2 Wind Tunnel Test

A common method used to get drag information under different circumstances in real world is wind tunnel test (WTT). During WTT, the subject of the test is placed in the tunnel, and air propelled by a specialized, huge fan would flow through the subject. Varying instruments are at this time used to measure a list of required datas. These datas pull together a whole picture of the aerodynamic forces and other physical conditions on the model[7].

The problem of WTT is that it is both time and resource consuming, while the wind tunnels are not all that flexible. Firstly, wind tunnels, because of its usual desirable size by its nature, have to be built in an almost remote area. Although some subsonic wind tunnels are even constructable for private use[8], other wind tunnels usually take up a whole space roughly of the size shown in Figure 2 below in order to fit in a large object as the one shown in the figure as well as installing a fan huge enough to provide the wind environment[9]. A even more troublesome fact of physically using a wind tunnel is the inflexible nature of it. Because of the various sizes of the tested objects, and the different wind speeds tests require for distinctive projects to create a desired environment, specific projects usually need to face many restrictions when looking for a suitable testing site[10]. So the financial cost and the waste of human resources are not desirable when industries are using wind tunnels to test object properties.

Figure 2: Wind tunnel concept. A picture that illustrates a wind tunnel that tests the stream line design and many physical datas of a ship.[11]

2. Research Objectives and Points of Innovation

2.1 Simulating mesh structure:

In order to further reduce the computational complexity and sample expensiveness, our study is on the application of deep learning and other algorithms in design optimization.

2.2 Our targets are as following:

2.2.1 Our research tries to simulate the physical conditions of objects in any given condition. Our purpose is to utilize the platform, MATLAB, to create a template that can generate physical models that can reduce the costs of actually making desired objects and evaluating the attributes of them in reality. We simulate properties of the object with MATLAB. We can thus avoid the cost of making the object for real world experiments.

2.2.2 Our research tries to find a fit for the correlation between θ values, which determine the shape of the objects in fluid, and drag values, which stand for the resistance the objects encounter in the fluid environment. We take our efforts in using deep networks to maximize our accuracy in predicting such relationships, and apply a series of foolproof steps to make sure the result is generalizable.

2.2.3 Our research tries to automatically search for the best shape of the objects under provided conditions. Such search automation refers to two parts. Firstly, the machine determines when and how to further improve its model for fitting the relations between θ's and drag's. We set a checking process so that once the algorithm determines that the trend our program has found is not actually the optimized case, it will automatically start another round of training with certain modifications in some parameters. Secondly, we pre-set a process so that the machine

would check itself if it has found the optimized shape. This process basically prevents the algorithm of getting into local minimums of the found fit of the correlations between θ's and drags. In total, the two parts act as an active search (AS) which serves for more automated process of our system.

2.3 What is new in our research:

2.3.1 We utilize MATLAB to create a brand new dataset that includes various settings and object shapes. This research starts from the scratch to finally form a program that works to make simulations of a diverse group of aerodynamic shapes. This approach is both less time-consuming and costly. The machine learning processing later is trained based upon this new dataset. Our MATLAB simulation process is made to be accountable and innovative.

2.3.2 This research raises a new system of algorithm that makes the process of engineering optimization as automatic, and computational and sample economic as possible. This new system is built upon a fully connected neural network and has other self-determinant conditions to decide where the program would go next. In such way, we make sure that the best fitting of the correlation between θ's and drags can be found, while other problems, such as local minimum, out-of-the-reasonable-range dilemma and so on, shall be avoided. Besides, since fully connected neural network is not the most computationally cheap method, we also set rigid conditions before when our program actually decides to retrain the entire model, encountering namel problems. These conditions can change parameters in the architecture of the neural network to minimize the computational complexity at its best. In summary, our new integrated system is designed to make computation-economic, accurate (represented by its minimized loss), accountable, non-misleading and foolproven results.

2.3.3 The way we apply machine learning in aerodynamic engineering optimization is new. In the latest such work that is available to us at the time of our research shows a pretty different way of approaching the result (Eismann et al., 2018)[12]. Firstly, the simulation process is different. We form our own way of simulating the physical properties of certain fluid environments. And our purpose is not to enumerate most possible shapes as the previous work. Instead, we employ deep network that describes below to find the correlation between θ and drag with a much smaller datasets to predict on an evenly accurate basis. Secondly, the follow-up methods to find the optimized shapes are different. The previous research sought for the optimization through Bayesian algorithm, while we utilize a integrated searching process to look for the best θ values possible in a fit curve that is responded by the least drag values. Therefore, the way we search for the optimized engineering design is novel, sample-economic, liable in its application.

3 Methodology

The core of our research is a fully connected deep network used with an intelligent and self-determinant loop that checks pre-set conditions and produces the optimized shape of the object under any given circumstance, which is set in MATLAB through process described below in the MATLAB simulation section.

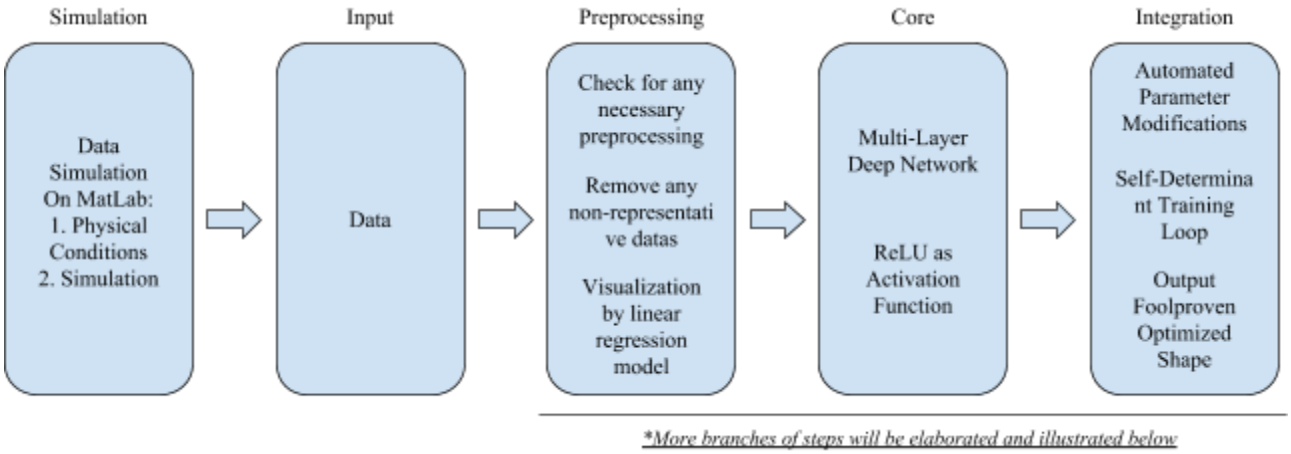

Figure 3: Flow of the system, procedure outline

Figure 3 shows the flow of our entire process. Our research focuses on making a comprehensive system that automatically finds an optimized shape under any given fluid environment. To be called a comprehensive aerodynamic optimization, our research covers everything needed from simulating objects to actually optimize their shapes. In such way, the process of aerodynamic optimization can be simplified and more digital based.

Detailed explanations of each step is illustrated below.

3.1 Physical Simulation with MATLAB

Incompressible flow in a volume V satisfies the Navier-Stokes equation[13].

$$\rho\left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u\right) = -\nabla p + \nu \nabla^2 u$$

In this equation

1. u represents the velocity of the flow, and is a vector field $V \to R^3$ for 3-dimensional flow problems, or $V \to R^2$ for 2-dimensional flow problems.

2. p represents the pressure in the volume, and is a function $V \to R$
3. ν is the viscosity of the fluid[14].

The Navier-Stokes equation is the equivalent of Newton's second law for fluids. To interpret the equation, we remark that $\frac{\partial u}{\partial t}$ represents the change in the flow with respect to time, and $(u \cdot \nabla)u$ represents the convective acceleration of the fluid. The right hand side represents the forces acting on the fluid: $\nabla^2 u$ is the difference between velocity of a point and the mean velocity of its neighborhood. This term encourages the vector field to become uniform in the absence of other influence factors. ∇p is the gradient of the pressure, and drives fluid motion.

A deterministic equation govern the dynamics of the system. However, understanding the system is still extremely challenging because no closed form solution exists due to its chaotic nature. In fact, even proving the existence/non-existence of a solution remains an important open problem. Therefore, expensive numerical simulation is required to assess the flow, and compute the relevant physical quantities in the system, including velocity, drag, etc. What is particularly important in our application is drag: the force an object experiences when it is traveling in the fluid.

We can use MATLAB to simulate the dynamics of an object traveling through a fluid environment. This avoid the high cost of manufacturing the object. To simulate the object we first create a geometry that represents the object and relevant boundary conditions. Then we convert the geometry into a mesh that represents the state of each point in the system. We use QuickerSim to simulate fluid dynamics according to Navier-Stokes equation, and read off end results including drag through the provided API. Based on this setup, we have a program that

simulates the drag for any shape, and we will use it to generate a dataset of shapes and their corresponding drag.

Another important factor to note is the number of datas required in our learning process. We require minimum such dataset size which is decided by the formula below:

$$n(x) = 2^x$$

In this equation, x stands for the number of θ lengths we use to simulate the two-dimension dataset that shadow to three-dimensional object, and n calculates the minimum data points needed. With this minimum data requirement, we will have sufficient data points to start our training.

3.2 Training the Model

We use two types of models to fit the relationship between shape and drag: linear regression and deep network.

3.2.1 Concepts of Linear Regression

Linear regression is a significant method to start with, both because of its essential presence in the field of artificial intelligence and its fundamental role in my research. Linear regression builds foundation for currently existing machine learning regression methods, as it provides a elementary assumption to the outcome of the regression model that it is a linear combination of the features[15]. Linear regression expresses itself in the form as below:

$$f(x) = \beta_0 + \sum_{j=1}^{N} x_j \beta_j$$

It is also seen in the form as a matrix:

$$f(x) = \alpha^T X$$

The loss function of such regression method is calculated in terms of the residual sum of squares, which is written in the expression as following:

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

This model is used first to roughly estimate the correlation between the actual drag values found through MATLAB simulations and drag values predicted by linear regression model from the full set of four-dimensional test θ values. We choose to use such model because of its relatively low computational complexity as well as its ability to straightforwardly graph the relationship between actual and predicted drag values, if there is any. Also, linear regression serves as a traditional method that our new method can potentially compete with. Later in the research, by fitting our four-dimensional θ data again with our new model, we can directly visualize if there is any veritable improvement in our new model compared to the traditional ones. In short, the linear regression model introduced here has its important role to start off the research both by visualizing trends with low computational cost and serving as a comparison to our new method later.

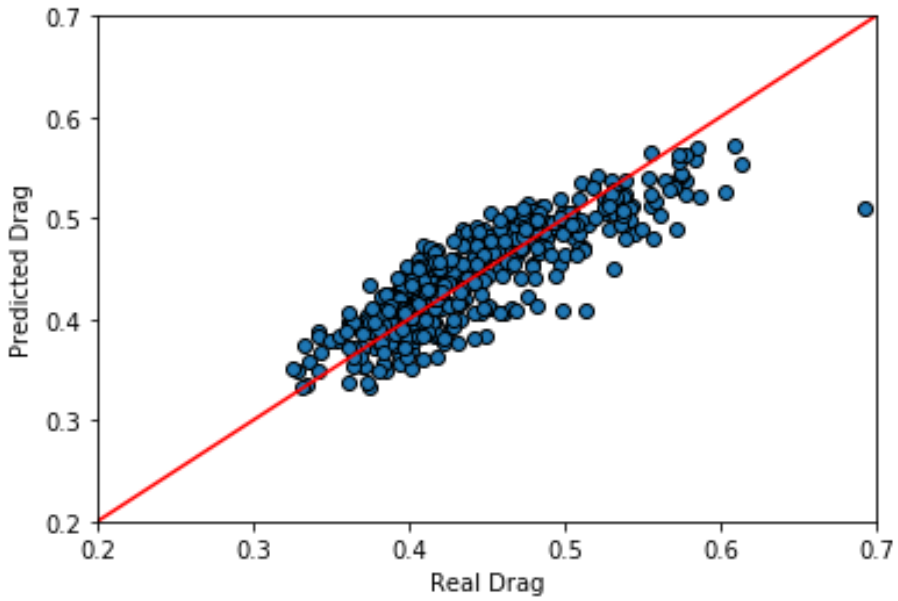

Figure 4: Linear regression result compared to real drag values, while object width is set to 0.18. Graph produced through programming.

Figure 4 represents how the linear regression result fits the real drag values. The red line crossing the diagonal of the chart represents the function:

$$f(x) = x$$

The closer all the blue dots are to the blue curve, the more accurate the linear regression prediction is. As is obvious on the graph, most of the data points attach to the blue line out of the 625 samples, while a few still alienates from the correct prediction. Two things are self explanatory in this case. Firstly, the linear regression model does find certain patterns so that it is able to make predictions rather accurate, especially in regard of the fact that the graph is zoomed in to a 0.40*0.40 square. Secondly, there are still rooms for improvement. In fact, the reduced

mean square of the loss function is calculated to be 0.00081 (training) and 0.00076 (testing) in the case of the preset width to be 0.15.

3.2.2 Cross Validation

My experiment is mostly based on finding the relationship between four-dimensional θ as the input dataset and the drags in the aerodynamic environment as the output dataset. Many generalization methods, such as L1 and L2 generalizations, are not applicable in this case, since there is only one feature, θ, to be taken into account for predicting drag values. However, cross-validation still finds its use in our training of the model, especially in the linear regression model. Therefore, it is also helpful to introduce the gists of cross-validation alongside other methods.

Cross-validation is a statistic model, also known as rotation estimation, or out of sample testing. The method tests the generalizability of certain predictions, which serves our purpose of the research well.

In our research, the entire dataset of 625 θ-drag pairs all directly serve as training datas. Every four-dimensional θ has a corresponding drag label. In our case of ten-fold validation, a random 10% of our dataset is made complementary subset. The remaining 90% of the datas are trained first before we perform testing or validation on the 10% partition. We repeat this process for 10 times to cover all data points randomly, so that a better generalization of our linear regression training can be achieved.

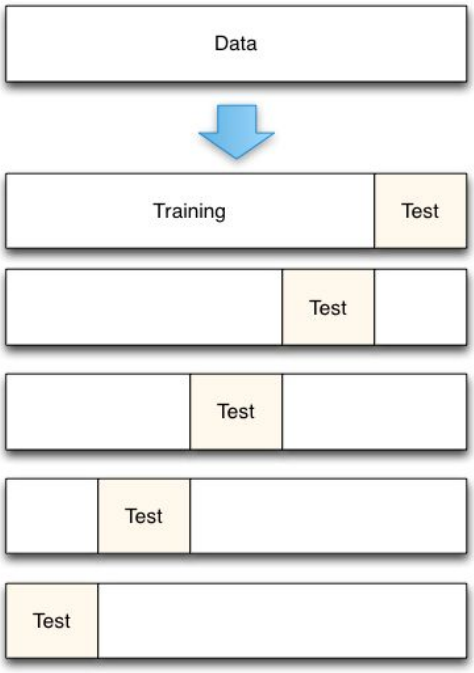

Figure 5: cross-validation and resampling. Figure from Accurately Measuring Model Prediction Error. (Scott Fortmann-Roe et al., 2012)[16]

We add this to our linear regression model because a more generalizable result is desired to make sure our visualization of the trend is relatable in the future study. Such generalization also ensures the result of the linear regression model accountable as a basis for our judgement of how drag values respond to different θ's.

3.2.3 Fully Connected Neural Network

We made the prediction mainly through a six-layer fully connected neural network with four hidden layers. Different from a convolutional neural network (CNN) that is used to analyse two-dimensional pictures either for classifications or automatic generation purpose, fully connected neural network (FCNN) is a deep learning method that has each of its layer connected to every neuron in the previous layer, with each connection taking its own weight[17]. Although

FCNN is more expensive than CNN memory and computation wise, it suits more in the context of our research. This is because we want to achieve a more accurate prediction and build a self-determinant loop to repeat the process of the neural network until a relatively optimized shape is found. In order to enable the automatic searching process and repetition, we need to leave weight between every pair of neurons available for retraining in each round so that no influential feature would be ignored.

Furthermore, such more expensive model as FCNN compared with CNN is suitable in our case, because our required computation is not as complicated as usual image processing tasks for two reasons. First, as stated above in the MATLAB simulation explanation, we only require 625 samples to reach a fair prediction that is generalizable. This number is significantly smaller than that of digital image processing which usually desire around 10000 two-dimensional samples. Moreover, our input datas consist of a list of arrays, which has a lower computational cost than two-dimensional images. Therefore, because of the remarkably lower computational cost as a nature of our dataset, FCNN is feasible. Thinking of the advantages and necessities of using FCNN, we chose to use it to fit the model.

FCNN is composed of fully connected layer (FC) which can be expressed in the following form:

$$drag_j^k = f\left(\sum w_{ij}^k \cdot x_i^{k-1} + b_j^k\right)$$

K represents a specific layer of the FCNN, while j represents the specific neuron the denoted variables are referring to. W serves as a parameter that tells the connection between the jth neuron on kth layer and the ith neuron on (k-1)th layer, alongside b as the bias adjuster. As the shape indicates, FCNN can somewhat be regarded as a CNN in computation or thinking.

The formula can be extended to look something as the series of equations below:

$$Z^{(1)} = f^{(1)}\left(W^{(0)}X + b^{(0)}\right),$$

$$Z^{(2)} = f^{(2)}\left(W^{(1)}Z^{(1)} + b^{(1)}\right),$$

$$\cdots$$

$$Z^{(L)} = f^{(L)}\left(W^{(L-1)}Z^{(L-1)} + b^{(L-1)}\right),$$

$$\hat{Y}^{(X)} = W\left(^{(L)}Z^{(L)} + b^{(L)}\right).$$

The six layers of fully connected neural network are as pictured in Figure 6. This architecture is chosen because of its superior accuracy of performance compared with the other kinds. For example, in the case when we set the preset width of subjects in MATLAB simulations to be 0.15, the reduced mean squares of prediction losses are lowered to 0.00029 (training) and 0.00019 (testing), while those in linear regression case accordingly are 0.00081 (training) and 0.00076 (testing).

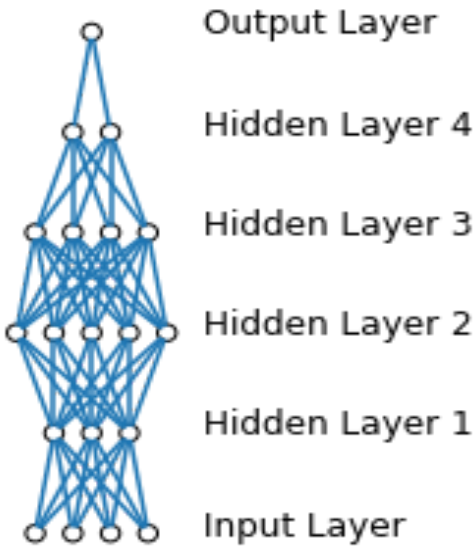

Figure 6: Neural network architecture in my research. Drawn on python by programming.

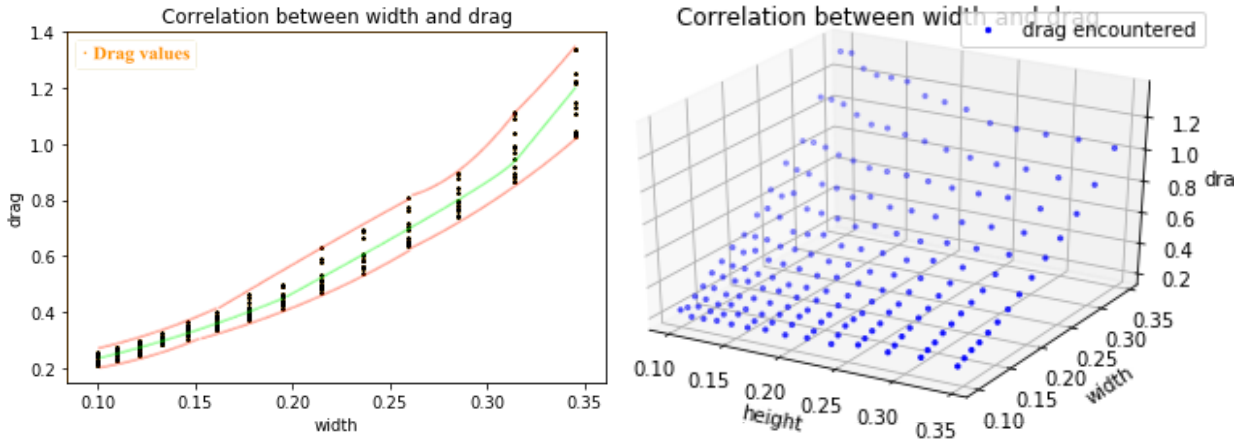

Figure 7: Visualization of how two-dimensional θ datas relate to drags

Also, we attempt to approximately visualize how θ would project to drags which has its representation graphed above in Figure 7. In common sense, we would easily come to the assumption that we should approach θ's in all dimensions of the objects to 0 to minimize the drag. This is not the whole picture. In reality, with many restrictions, we usually need to preset a width value for the object and then look for a best shape with the preset condition. In Figure 7, we see that there usually exists optimized values of θ's beyond just approaching zero all the time. For instance, in the two-dimensional visualization, we reach the minimum drag when the height is between 0.10 and 0.15 if the width is set to be 0.15. So later, when we use the network to predict how four-dimensional θ values correlate to drags, our purpose is to find such feasible minimum with the network.

3.2.4 Rectified Linear Unit

As the fully connected network is being trained, weights and biases between each pair of neurons are defined as variables. With real drag values as task targets to achieve through model prediction, the network is able to train itself to find weights and biases for minimized loss.

In the process of training, we need suitable activation functions. In our case, one activation function is rectified linear units (ReLU), calculated through the expression below:

$$f(x) = \max(0, x)$$ [18]

ReLU is a fairly popular kind of activation function in neural network.

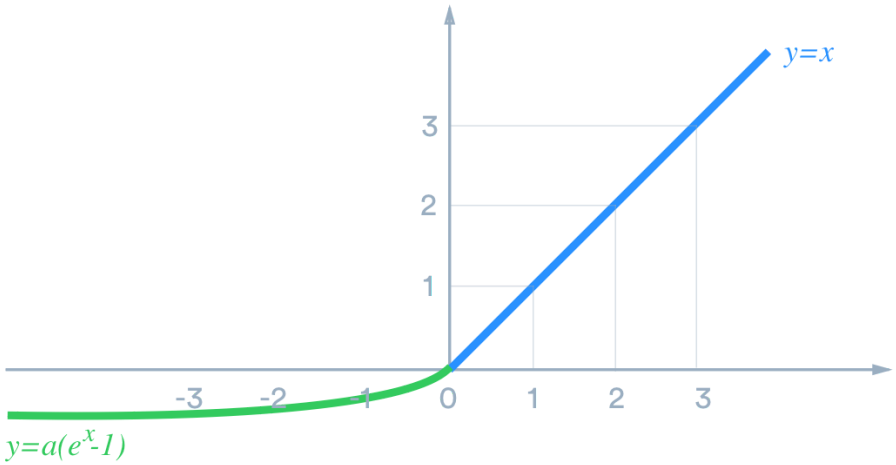

Figure 8: ReLU activation function, Dan-Ching Liu et al. [19]

Figure 8 shows a general shape of ReLU function. Compared with other activation functions, such as sigmoid, tanh, and linear, ReLU is a single-sided function. It satisfies the mechanisms of neural network better. ReLU has a constant-valued slope when $x \geq 0$ so that it does not have sigmoid's predicament of vanishing gradient. In the case of ReLU, only multiplications and comparisons are processed so that we may achieve a faster and more accountable convergence of

results. Furthermore, in practice of our research, ReLU is able to reduce the loss of our predictions on a perceptible scale.

ReLU is used in a convolutional way through the fully connected neural network that has been introduced in the previous section.

3.3 Automated Search for Drag-Reduced Shapes

When the search of a drag-minimized shape runs into a local minimum or when the initialization of the training leads to a zero-derivative, the prediction result is not in its most accurate case. So we need another process to 1) have the machine automatically adjust its parameters so that such local minimums and zero-derivatives can be avoided; and 2) make sure the found optimized shape actually encounters the minimum drag.

3.3.1 Restrictions for Self-Adjusting Parameters

Generally speaking, two parameters determine the way our model trains itself: train step and variable initializations. In the case of following conditions, our model would automatically determine to retrain itself: 1) when the taken derivative is found to be zero; or 2) the model is found to be stuck in a local minimum. Specific restraints we set are explained in the following sections.

3.3.2 Check for Decreasing Loss

During the first round of train, the model is trained three times, with the train steps being 1e-3, 1e-4 and 1e-5, which are tested to be the number of train steps by which our model would most likely find the trend this research is looking for. Each training process is trained for 20000 train

steps, because most training cases will reach its peak by this number. For instance, when the width of the objects is set to be 0.18, prediction loss has the trend as is illustrated in Figure 9.

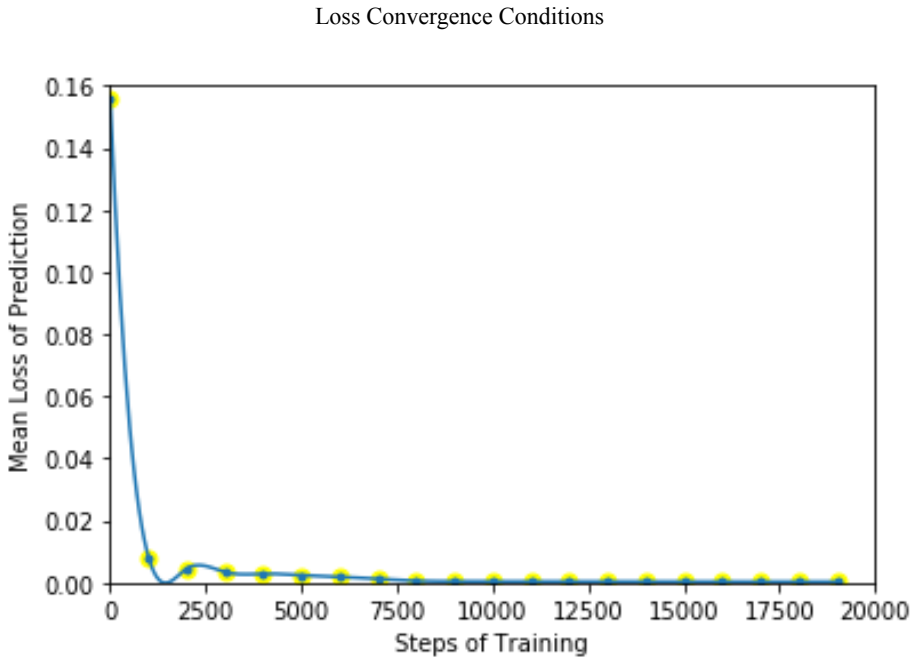

Figure 9: Loss convergence conditions, this graph shows that the convergence of loss through the training of 20000 effective train steps. Generated by programming on python.

As is indicated by Figure 9, the loss of 0.18-width objects converges to have a 0.000293707 predicted loss. Moreover, the losses are consistent around the value 0.0003 after 12000 train steps, so the remaining steps serve as foolproof for unexpectedly large desire of train steps to reach minimum.

After the first training round, there are two other possibilities, however, if not a single final optimized case is found, which are elaborated below:

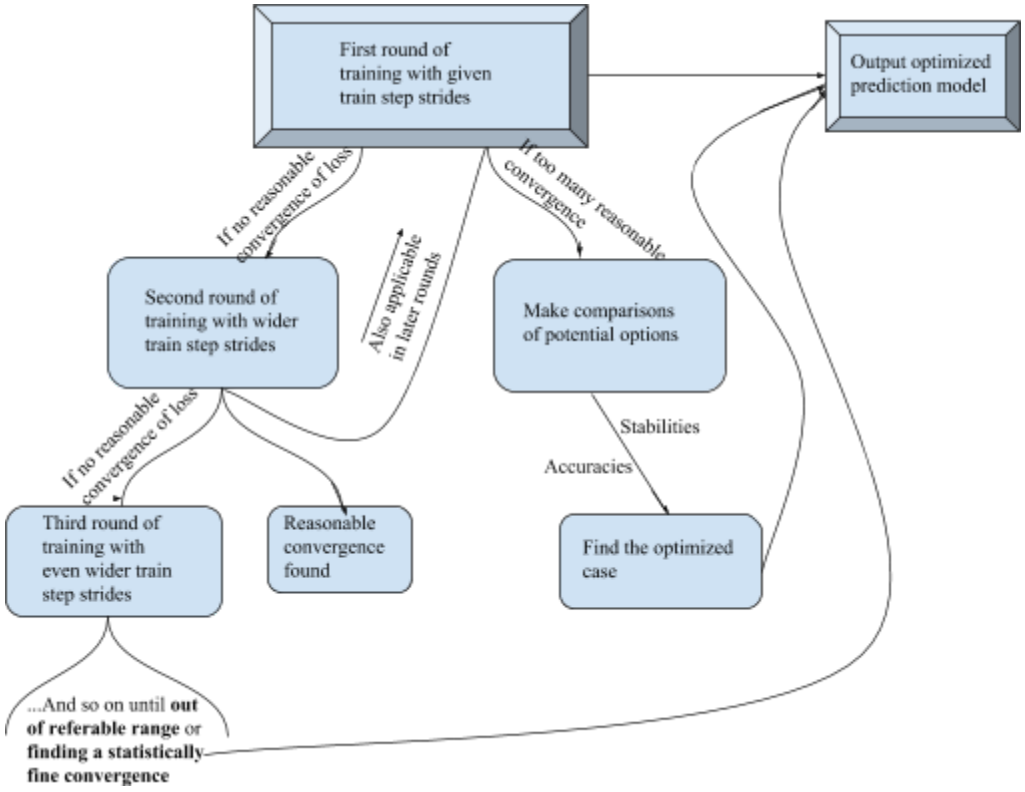

Figure 10: Flow graph of automatic training system. Detailed explanations are given below.

Firstly, there may exist no reasonable loss convergence in our tested cases. We create a mechanism to compare the loss condition each step. If the loss does not decrease in a reasonable distribution or is not reduced at all, the mechanism will send its judgement for our system so that it will start another round of training. In this round, the two train-step cases are le-2 and le-6, and so on until there is a reasonable convergence of loss. However, in some cases, it is also possible that there is no liable convergence until it is out of the range for logical train step sizes. At such time, the system would automatically refresh its initializer so that there is a new initialization to avoid zero derivatives.

Secondly, too many reasonable convergences will be produced in the first round (a.k.a

two or three reasonable convergences). In this case, our system is programmed to compare both accuracies and stabilities of the convergences. To collate the performance accuracies, we juxtapose each of the test and train accuracies in each plausible training, and find out which convergence has the best general performance among all. To compare the stabilities, we take the difference between the level of test and train accuracies of each reasonable convergence respectively. Then, we compare both train and test accuracies in each of the reasonable examples. The one that maintains relatively better train and test performances would be selected. While the accuracies and stabilities contrast with each other, accuracies would always be the prior factor of judgement.

3.3.3 Check for Drag-Minimized Shape

After the overrounded model training, the resulting shape predicted is still not necessarily the best outcome possible. This may be owing to undesirable initialization, incomplete fit of model and many other factors. To prevent these instances become our outputs, we design a last step to check if it is drag-minimized.

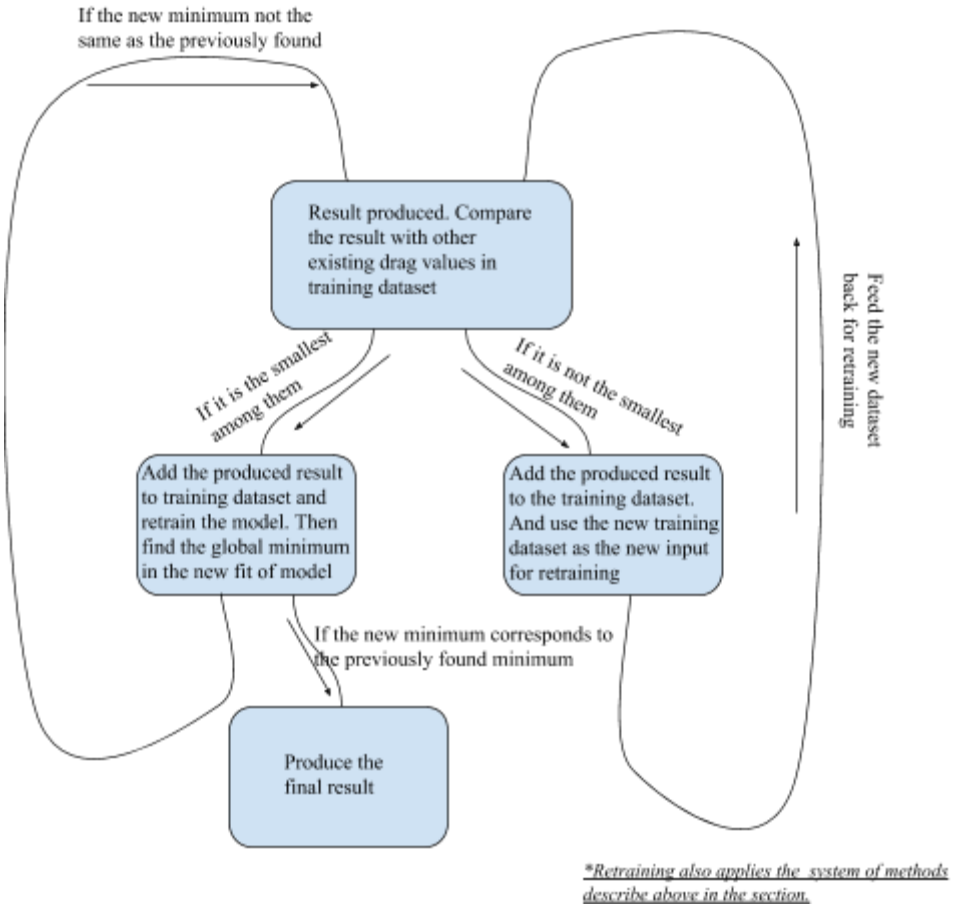

Figure 11: Flow graph of double checking result.

As Figure 11 above illustrates, our last step of the system is to automatically check if the found optimized shape actually have the minimized drag in the given environment. The mechanism basically works as following. Firstly, the found shape's drag is compared with the rest of the examples in our initial dataset of 625 drag values. If it is the smallest, it will be added to the original dataset and feedback to the training system for retraining. After the retraining, if the shape is actually also the global minimum in the new model, then it will be selected. If not, the new global minimum will be used for the new round of active search. Also, if the found optimized shape's drag is not in fact the smallest compared to the present drag values, then the

found value will also be added to the training dataset. Our system would determine to get another round of retrain until a veritable drag-reduced shape is ensured.

4 Results

4.1 Deep Network Regression Performance

Our own dataset is primarily the center of our investigation. Pretraining selection firstly take place to eliminate those simulations that is too close to singularity or off-scale to be for meaningful consideration. Selected 625 samples (the specific number is due to the reason stated above) with reasonable shapes are then fed into linear regression model for visualization purpose. This linear regression process is accompanied by the use of 10-fold cross-validation to make sure the visualization through linear regression is accountable and generalizable. The result of the traditional regression model clearly shows that there exists a correlation between the predicted drags and real drags, so that there should be an anticipatable trend between different θ values and drags. At this point, more accurate predictions of drag values from θ's are desirable. So, we develop a deep network built upon 6 fully connected layers which shows an optimized accuracy compared to the peers. Table 1 below shows some examples of comparisons of performance:

Table 1: Performance of different deep network architectures for regression loss

	Train Loss (0.15 width)	Test Loss (0.15 width)	Train Loss (0.18 width)	Test Loss (0.18 width)

Seven Layers	0.00071	0.00057	0.00085	0.00041
Six Layers	**0.00060**	**0.00046**	**0.00085**	**0.00041**
Five Layers	0.00389	0.00352	0.00135	0.00086
Four Layers	0.00135	0.00124	0.05154	0.05045

With two width cases * four types of deep network architectures, the total eight total different cases picture where the optimized deep learning architecture can be achieved. As there are more examples of different cases already tested, there is a common trend is represented by the two set of examples in Table 1. Six layer deep network has significant improvement of performance in terms of mean squared loss (both training and testing) compared to other architectures with fewer layers, while another additional layer shows no perceptible reduction in loss (both training and testing) in the case of 0.18 width but increases the mean loss in the 0.15 width case. Therefore, compared with fewer layers, six-layer deep network has superiority in its notable reduction of mean loss. As for more layers, six-layer architecture is no less accurate but less computationally expensive. Therefore, such architecture is selected.

Compared with traditional regression models, moreover, our deep learning model explicits its superiority in performance as is shown by the comparison in Table 2:

Table 2: Performances of different models

	Linear Regression	**Our Model**
0.15 Width Case Training	0.00081	**0.00029**
0.15 Width Case Testing	0.00076	**0.00019**
0.18 Width Case Training	0.00101	**0.00085**

0.18 Width Case Testing	0.00078	**0.00041**

In both width cases and more, our deep learning model with its project-specific architecture is proven to be superior in prediction accuracy in comparison with other traditional regression models.

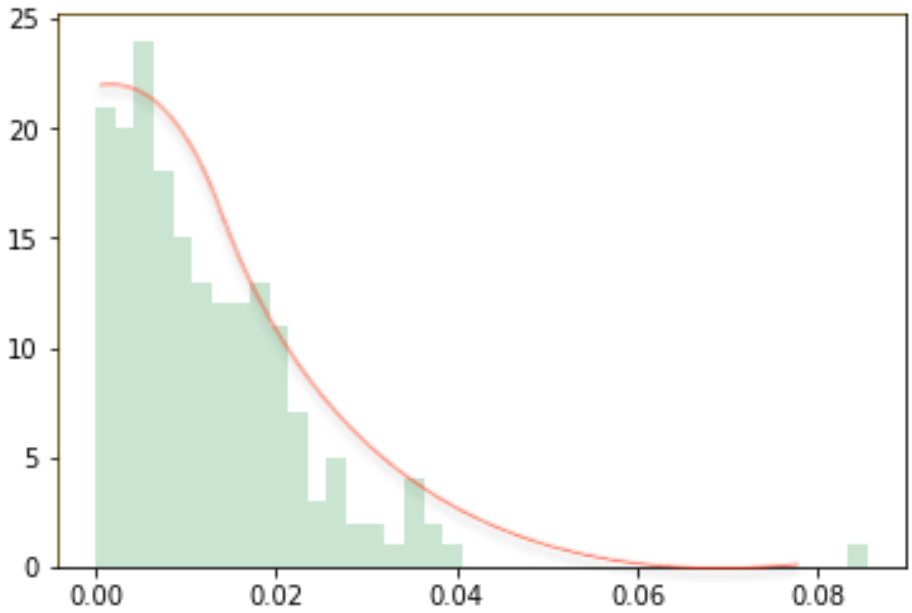

Figure 12: Loss distribution in the case of 0.15 preset width

Figure 12 visualizes the distribution condition of a well trained, applicable model. Under this condition, datas start to look statistically correct. As the red curve pictures out, the distribution of each individual prediction loss distribute in normal distribution, skew to the right with most loss in between **0.00 and 0.02**. This graph tells both the stability and accuracy of the

model at current status, which means, at this point, this case of width is already well trained for preceding to the next phase.

Our model does not take in other methods in the predicting process. One the one hand, the potential improvements are largely limited. As for stacking of multiple different machine learning methods, there can be only tiny to no boost to the accuracy. This is due to the mechanisms of how stacking and ensembling work: the efforts to remove unshared shortcomings of each method. Therefore, the possible improvements are usually ignorable. As for the chances of better accuracy because of the nature of other methods themselves, other methods generally show less accuracy compared to ours in terms of accuracy proven by direct comparisons of performance. One the other hand, the possible enhancements are not necessary. Sost drag values that our simulation system would get are above 0.30, while our system has already reduced the mean loss to the level mostly below 0.00050, so further advancements are not seen as critical.

So with many factors taken into account, our deep network with five fully connected layers is concluded as the most accountable model with comparably consistent performances and high level of accuracies in both train and test. This architecture improve the loss of our prediction to be generally under 0.0005.

4.2 Automated Engineering Optimization Performance

The performance of our generation of the optimized shape indicates the ultimate use and practicality of our research. Our design of the active search again boosts the automation of our system. When the search of a drag-minimized shape runs into a local minimum or when the initialization of the training leads to a zero-derivative, the prediction result is not in its best case. Our system would avoid such conditions.

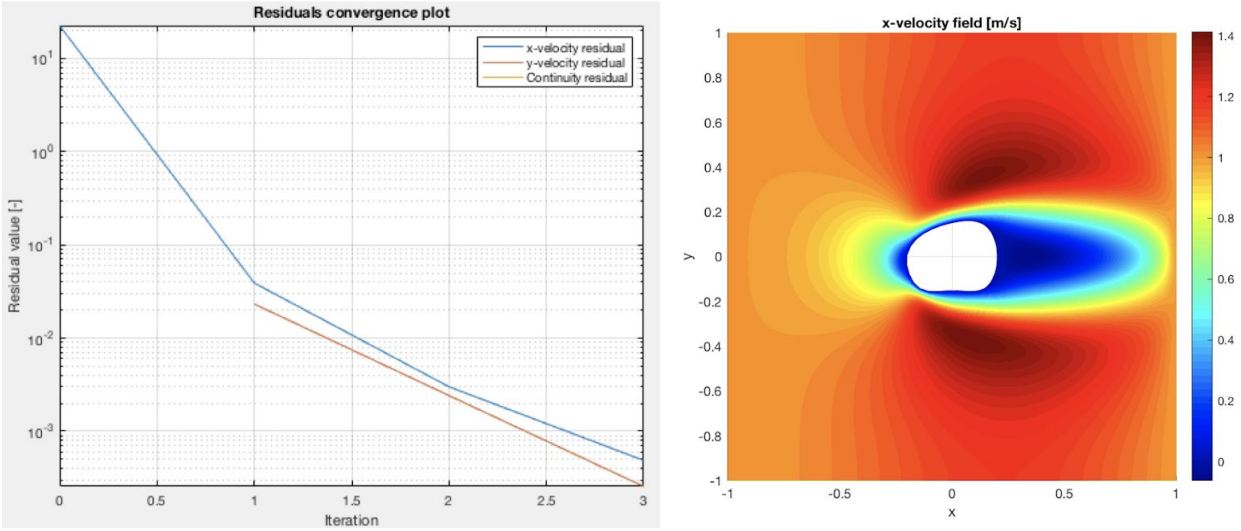

Figure 13: Simulation reflection of the predicted optimized shape after first round training.

For example, during our train of the case when width value is set to be 0.15, the optimized case is not achieved in the first round of judgement. In Figure 13, the graph on the left is the residuals convergence plot. The plot on the right shows the relative movement between the object and its surrounding environment. The color bar on the right of the plot assists to read the graph. Pretty self-explanatory, sections round the object have almost zero relative movements against the object, while further areas have higher speed of movements. These two in total serve as our base for predicting drag values. In this first effort to optimize, drag is shown to be **0.394**, which apparently is not indeed optimized and is captured by our mechanism. So it automatically goes into the second round of training, that reports the result below.

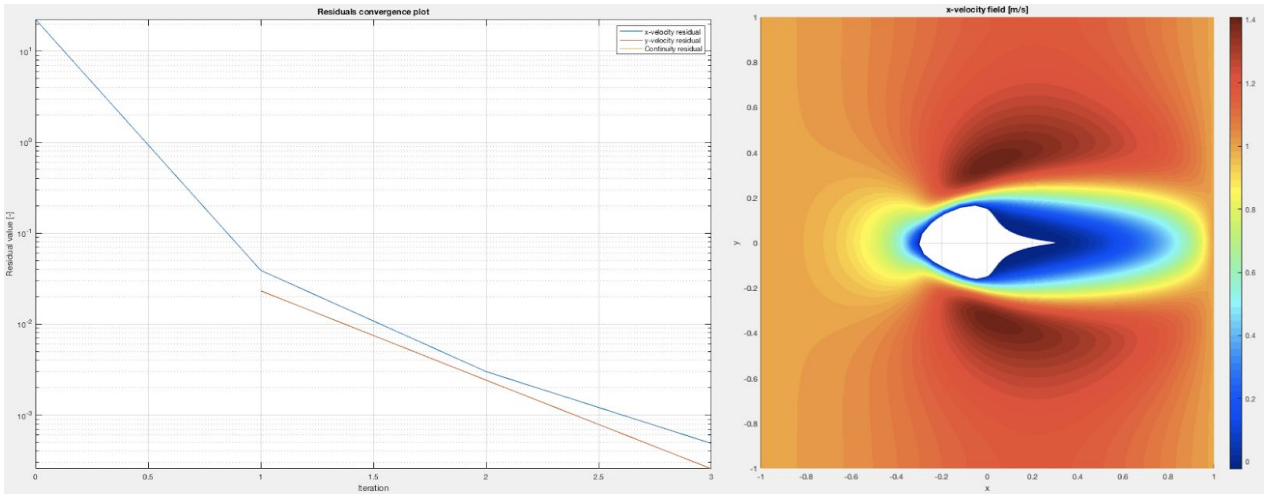

Figure 14: Final selected, optimized shape and its simulation.

The two plots above in Figure 14 work the same way as Figure 13. After few rounds of active searches, our system automatically comes to the shape in Figure 14, which has a drag of only **0.281**. This result is then proven to be the actual minimum, tested by our series of foolproof processes that are already explained in methodology section.

This instance is quite exemplary because it encounters most bugs that our training system can have and proves the reliability of our post-training selection process. This reliability is similarly observed in other cases with widths of **0.18, 0.20, and 0.30.**

5 Discussion

5.1 Summary and Expectations:

The deep learning architecture design in this research is superior in its accuracy and stability performance compared to linear models. It further reduces the loss of prediction on a perceptible scale compared to traditional regression models, such as linear regression. According to the data visualized above, it is clear that our model is able to predict the relationship between θ's and drags accurately and precisely. This result shows that our solution to find a fit for the correlation and then find the global minimum in this fit is feasible and our model has a higher performance than traditional ones in the experimental conditions that we set in methodology simulation section. The detailed comparisons between our model, and traditional models and other architectures are drawn in the result section above in Table 1, Table 2, and Figure 12.

5.1.1 The application to use a machine learning model to find a fit for the correlation between θ's and drags is new to this research. The way our data works also optimizes the use of computation power. We map a matrix of four-dimensional θ arrays to a column vector of one-dimensional drags, instead of finding the fit of the entire process of objects moving through given fluid environment. In our way, a relatively accurate result only requires a small data size as described above. To make the case of prediction more comprehensive, one way is to add more dimensions to θ arrays. Under that circumstance, an important additional step is to set restrictions to utilize projections to make sure the shape would be reasonable (a.k.a. No weird indentations in the shape). Also, it would require a bigger dataset with approximately 2^n data points (n stands for the dimension of θ), so a heavier computational cost.

5.1.2 Our optimization system has rather well-rounded loops to filter cases not yet fully trained. We make sure that our system actually find the optimized shape with the provided

information (e.g. simulated dataset size, model fit from the datas and so on). This also means that our system has the ability to avoid bugs in usual machine learning training.

5.1.3 Another rather significant, potential boost to our research is to make the fluid environment more complex. Now, we simulate under only viscosity and some primitive elements of fluids. Nonetheless, for designing aerodynamic devices, a more complex factor of influence is the wind, which is the key point why wind tunnels are still necessary. At this point, we are still short of samples that can provide information of actual wind tunnel experiment results. With sufficient samples from those wind tunnels, in fact, we can find a trend on how wind affects the engineering performance. As more such examples are used for more accurate prediction on such relationship, our idea of research is capable of further lessening our dependence on wind tunnel experiments.

5.1.4 Our research is highly applicable in real life. For the making of an optimized shapes for cars, spaceships or even tiny injection devices, aerodynamic engineering design would find its use. Our system applies machine learning models to find how the factors of influence would impact the performance of objects in the fluid, and search automatically for shapes that can accomplish their purposes the best. The greatest benefits of searching for the optimized shapes through our proposed method instead of the traditional counterpart are our system's low resource and time consumptions.

5.2 Proposed Future Extensions:

Digging deeper into the study by adding more complex critical elements and looking for complex trends from previous wind tunnel testing samples is one prospective extension as

referred to above in 5.1. However, following may also serve for the potential improvements of our system.

5.2.1 We may need to add other necessary restrictions to the objects themselves. For example, because aerodynamic machines usually have few specific purposes, we need to set minimum volumes of the objects, or sometimes minimum length on certain θ values so that their functions can be achieved. Therefore, to better make objects for specific tasks, we also need to specify restrictions for the optimized shapes we are looking for in future studies.

5.2.2 The integration of multiple aerodynamic objects shall also be studied. This happens when, because we sometimes need to set restrictions for shapes that we are optimizing to serve for specific tasks, the ultimate shape we get may not be close at all to the original global minimum of drag of our trained model. So, supplementary parts may be used to boost the engineering performance of the shape. Consequently, a method of finding such appropriate add-ons may be studied to include into our system.

5.2.3 Now we only have a general purpose of reducing the resistances of the objects travelling through a certain fluid environments. However, for specific tasks, we may also desire other features. For example, ability of an object maintaining its current height, stability or agility of an object. With varying purposes of the optimization, these conditions can be set into the MatLab environment to be run through and tested the same way as we do on the drags.

6 Conclusion

In the end, our research successfully demonstrates that a more systematic and automatic aerodynamic engineering optimization is feasible by getting a regression mean error at below 0.0005 for most preset width cases and achieving actually drag-reduced shapes with the fit model through our systematic model. Compared to the previous researches, we have the advantages of requiring less training samples, less computation costs and time while improving the automation of engineering design and avoiding training bugs. On top of what we have already done, detailed additions as described in the discussion section is able to further increase the comprehensiveness of our system. These future bonus shall be achieved in a pretty similar way as we do here with the drags, except with different simulation features.

Our research successfully innovates on two things: finding a trend that relates to drag values, and searching for a drag minimized shape. This provides a insight into how drag is influenced by objects' shape with merely our machine learning prediction. This process can be more straightforward and practical than other methods that attempt to find such correlation.

In the future, more features should be studied the same way as the drags so that other feature-specific tasks can also be sought out. Also, real datas from wind tunnel tests will also be helpful in the future for us to train out trend of the impacts of more complex features that MatLab can hardly simulate. With these extensions, our automation system shall be made more comprehensive for improving aerodynamic shape designs.

7 Acknowledgements

The author would like to offer his cordial gratitudes for the magnificent assistance of the research mentor, Shengjia Zhao, on this research. He bears constructive advices to the author when challenges and confusions are to be dealt with. The author is also thankful for the invaluable opportunities Dongrun-Yau Science Awards make available, including and especially for having this research paper reviewed by a committee of expert judges.

Reference:

[1] Dunbar, B. (2015, May 12). What Is Aerodynamics? Retrieved from https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what-is-aerodynamics-k4.html

[2] Timmermans, M. (2015, September 04). SHOL Wind tunnel testing. Retrieved from https://www.nlr.org/capabilities/wind-nuisance/shol-wind-tunnel-testing/

[3] Aerodynamics & Aeroacoustics. (n.d.). Retrieved from https://www.dantecdynamics.com/aerodynamics-aeroacoustics

[4] L. (2017, October 27). Bjorn's Corner: Aircraft drag reduction, Part 2. Retrieved from https://leehamnews.com/2017/10/27/bjorns-corner-aircraft-drag-reduction-part-2

[5] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012b.

[6] E., S., B., S., E., & S. (2017, December 10). Shape optimization in laminar flow with a label-guided variational autoencoder. Retrieved from https://arxiv.org/abs/1712.03599

[7] R., & J. (1964, October 31). Wind tunnel experiments on the flow over rectangular cavities at subsonic and transonic speeds. Retrieved from https://repository.tudelft.nl/view/aereports/uuid:a38f3704-18d9-4ac8-a204-14ae03d84d8c

[8] Tourn, S., Pallarès, J., Cuesta, I., & Paulsen, U. S. (1970, January 01). Characterization of a new open jet wind tunnel to optimize and test vertical axis wind turbines. Retrieved from https://aip.scitation.org/doi/abs/10.1063/1.4982750?ai=1gvoi&mi=3ricys&af=R&

[9] Petersen, G., Leitl, B., & Schatzmann, M. (1970, January 01). ABL-Flow over Hills: A Review of Theory and Wind Tunnel Studies. Retrieved from https://link.springer.com/chapter/10.1007/978-3-642-28968-2_36

[10] Selig, M. S., & McGranahan, B. D. (2004, November 01). Wind Tunnel Aerodynamic Tests of Six Airfoils for Use on Small Wind Turbines. Retrieved from http://solarenergyengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1457129

[11] (n.d.). Retrieved from http://www.clearcom.com/markets/case-story/MAG/clear-com-launches-communications-solutions-for-air-force-at-nasa-ames-research-center

[12] E., S., B., S., E., & S. (2017, December 10). Shape optimization in laminar flow with a label-guided variational autoencoder. Retrieved from https://arxiv.org/abs/1712.03599

[13] Constantin, Peter, and Ciprian Foias. *Navier-stokes equations*. University of Chicago Press, 1988.

[14] Wikipedia, https://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_equations

[15] David A. Freedman (2009). *Statistical Models: Theory and Practice*. Cambridge University Press. p. 26.

[16] Scikit-learn video #7: Optimizing your model with cross-validation. (2016, May 17). Retrieved from http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/

[17] S., & L. A. (n.d.). Fully Connected Layer. Retrieved from https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/fc_layer.html

[18] N., & H. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines* (PDF). Retrieved from https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

[19] Liu, D. L. (2017, November 30). A Practical Guide to ReLU – TinyMind – Medium. Retrieved from https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7