Applications of Machine Learning in Tracking STLE of Hypersonic Vehicles

Andrew Lee
Department of Mathematics
William & Mary
Advised by Dr. Greg Hunt

Abstract

Hypersonic vehicles are objects that travel at least 5 times faster than sound. These engines operate by taking in air at an inlet, passing it through a duct called an isolator, and then mixing it with fuel for combustion. At hypersonic speeds, a series of shock waves, called a shock train, is formed in the engine's isolator. The location of this shock train changes in relation to changes in flow conditions of the vehicle (e.g. trajectory or fueling rate). Tracking and controlling the shock train is paramount for a working engine. Indeed, if not properly controlled, a run-away shock train can lead to catastrophic engine failure. Real-time prediction of shock train location is difficult. Previous studies have showcased methods to predict location through photo imagery, large-scale data analysis, and weak estimation methods which all have limitations in real-time settings. This project explores a traditional statistical learning approach to tracking the shock train leading edge. This is a problem of direct interest to researchers at NASA and the Air Force and several active flight tests.

1 Introduction

Accurately predicting the shock train leading edge is a complicated task, and many previous methods have shown their limitations due to various restrictions in their experimental facilities and computational demands. This project explores various machine learning algorithms in predicting the shock train leading edge, given no restrictions in computational cost. Two datasets have been used in this project, namely the University of Michigan Direct-Connect Isolator (UMDCI) and Air Force Institute of Technology (AFIT), to train and test a model. Each dataset contains series of labeled test runs, and three major tasks are discussed here: (1) Given a single run, can we use the first half to train, and predict on the other? (2) Given N runs, can we train a model on n_0 runs and predict on the rest n_1 runs such that $n_0 + n_1 = N$. (3) Can we transfer knowledge from one dataset to another? For this project, (1) and (2) are answered with promising results. (3) will be discussed; however, it is much more complicated task in nature. Regularized linear regression and neural networks were mainly used for building the model, but other methods such as random forest, support vector machines, and k-nearest-neighbor were experimented as well.

2 Dataset Description

Before diving into methodologies, we present important dataset descriptions and terminologies. The only observations made from the runs is pressure which is mesaured by psi (p1-p5 in Figure 1). Pressure transducers are tap-devices attached to the isolator to measure pressure during experimental runs. px1-px5 represents the stream-wise location of the pressure transducers measured by mm. x_1 is the target variable, STLE Location, we are interested in predicting. H and W are the height and width of the isolator measured in mm. In short, we will be using pressure measurements to predict the stream-wise location of the STLE.

	px1	px2	px3	px4	px5	p1	p2	p3	p4	p5	x1	Run	H	W	time
0	521.81	565.5	609.19	652.88	696.56	12.778115	14.397894	14.224309	14.228612	22.287420	714.009158	R0	57.15	69.85	0.0000
1	521.81	565.5	609.19	652.88	696.56	12.282638	13.825813	13.646973	13.673873	21.732822	714.293896	R0	57.15	69.85	0.0001
2	521.81	565.5	609.19	652.88	696.56	12.375714	13.774987	13.601214	13.649611	21.445008	714.578634	R0	57.15	69.85	0.0002
3	521.81	565.5	609.19	652.88	696.56	12.315807	13.875195	13.664667	13.705164	21.687339	714.436265	R0	57.15	69.85	0.0003
4	521.81	565.5	609.19	652.88	696.56	12.432873	13.823566	13.670350	13.751020	21.101177	714.009158	R0	57.15	69.85	0.0004

Figure 1: Table Head of UMDCI Dataset

3 Multiple Linear Regression: Predicting a Single Run

Multiple linear regression is a widely used machine learning method that fits a least-squares regression line as shown in Equation (1).

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{1}$$

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{y} - y^{(i)})^2 \tag{2}$$

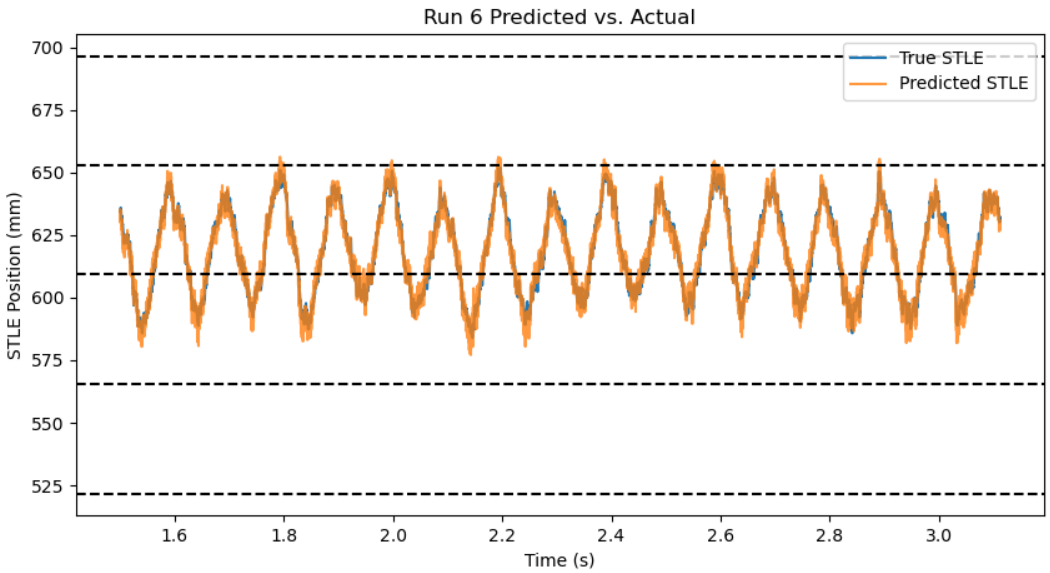


Figure 2: UMDCI R6 Predictions

The computations of the β coefficients are done by minimizing the cost function MSE (Mean Sqared Error) as shown in Equation (2).

Let O be the observations and t be its corresponding time. We define the training and testing set by letting $R = \{x \in O : t \leq 1.5\}$, and $T = \{x \in O : t > 1.5\}$. Individual models were built on each of the 37 test runs. The average R^2 of these models in the test sets were 0.932751 with average MSE of 63.875930.

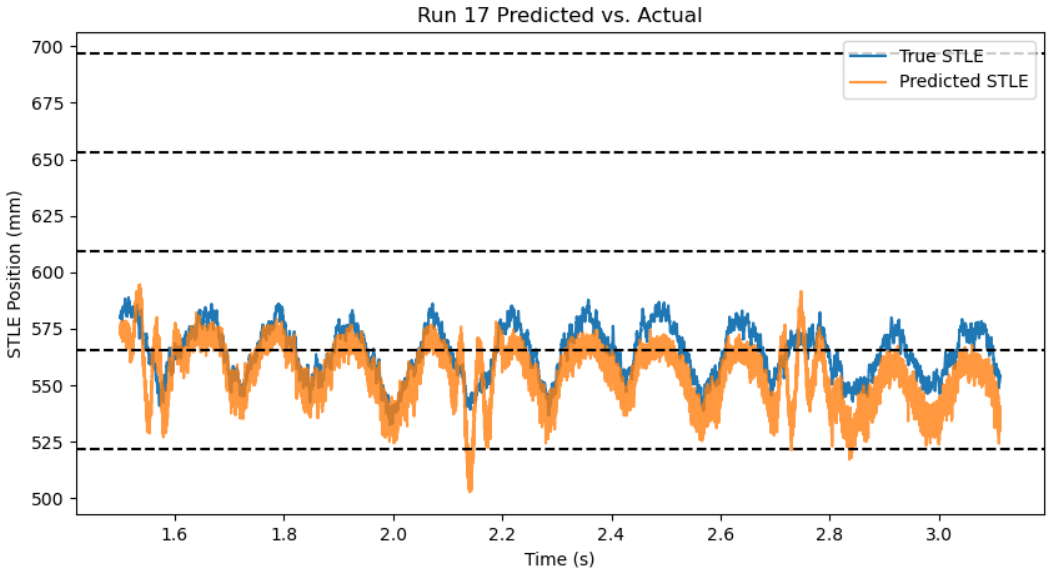


Figure 3: UMDCI R17 Predictions

Interestingly some of the runs performed far worse than others, which are illustrated in the Figure 2 and Figure 3. Notice that the model for Run 6 predicts with high accuracy over the period of time, whereas Run 17 yields poor results particular at t=1.6, t=2.2, and t=2.7.

Multiple linear regression captures the linear relationship between the dependent variable and two or more independent variables. Hence, the heatmap presented in Figure 4 may explain some of the reasons why Run 6 and Run 17 yielded vastly different results. Notice that R6 contains three variables (p3, p4, p5) that have absolute correlation coefficients greater than 0.9. On the other hand, R17's variable with the maximum correlation coefficient is p1 and p2 with -0.78. Hence, one could conclude that R17 lacks variables that are linearly related with the target variable.

Furthermore, Figure 5 helps understand the odd behavior of the R17 model. Notice that the pressure measurements have five occurrence of high noise throughout the experiment. Not surprisingly, these noises correspond exactly to the time points where our R17 model fails to perform well. The linear model was not able to capture the noise patterns. In this case, there are two approaches we could take: complex model selection and feature engineering. With the hope that non-linear model would enhance the model, we proceeded to train a random forest regressor and a SVM (Support Vector Machine) regressor. However, adding the complexity of the model did not yield satisfactory results. Now, the task was to focus on the dataset itself.

3.1 Feature Engineering

In data science, feature engineering is a process manipulating and transforming variables using domain knowledge. Since R17 model performed fairly well except for the occurrence of odd noises, we hypothesize that allowing the model to have some sort of memory of the past observations would help reduce the noise. Hence, we define new sets of variables that have the above property mentioned.

$$\bar{p}_{SMA} = \frac{p_M + p_{M-1} + ... + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i} \tag{3}$$

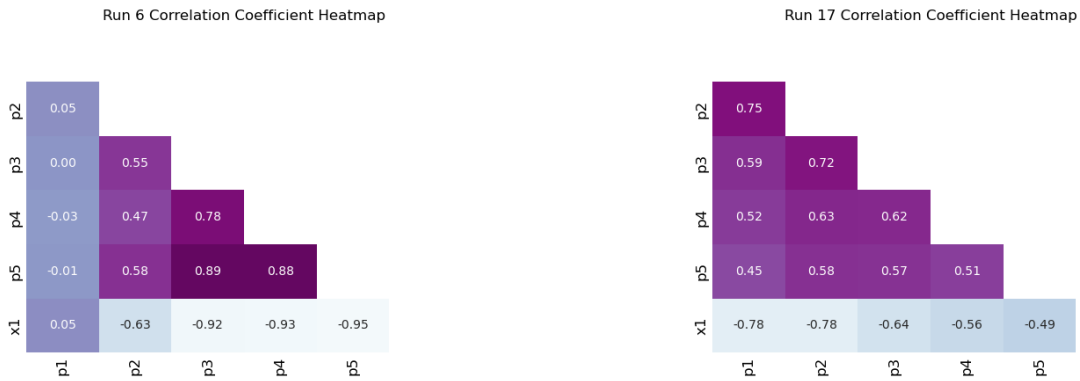


Figure 4: Correlation Coefficient Heatmap of R6 and R17

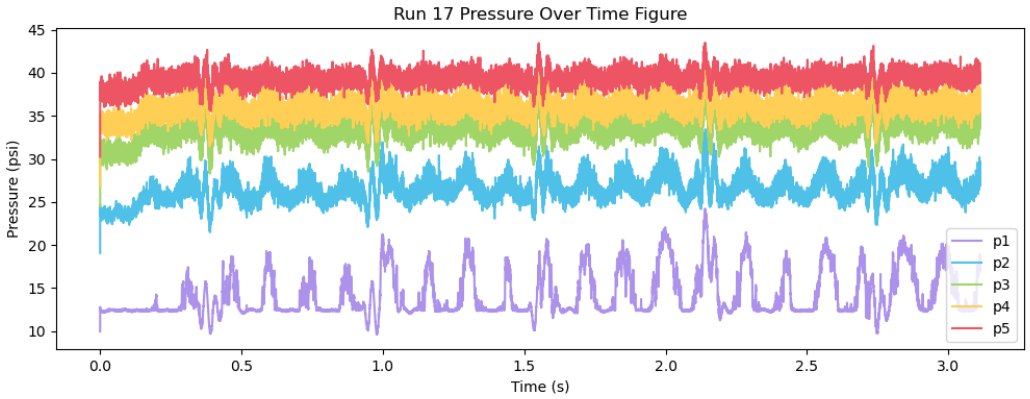


Figure 5: Pressure Profile of R17

For this project, a method of simple moving average was used. This method widely used in time series analysis, which allows to ignore the noises between close by observations. Equation 3 mathematically expresses how the simple moving average is calculated. Given a window size n, each observation is computed by taking the average of $n-1$ previous observation. Hence, the method loses information about the n initial observations. After series of experimentation, the window size was chosen to be $n = 300$ for this particular dataset. Also, losing 300 initial data points did not hurt our model nor interpretability as each runs had over 30000 observations.

Fortunately, feature engineering improved the model greatly in terms of its predictive power. The R17 model yielded a R^2 of 0.892446 and a MSE of 14.272526. Figure 6 visually shows the improvement of the R17 model with reduced noise and higher accuracy.

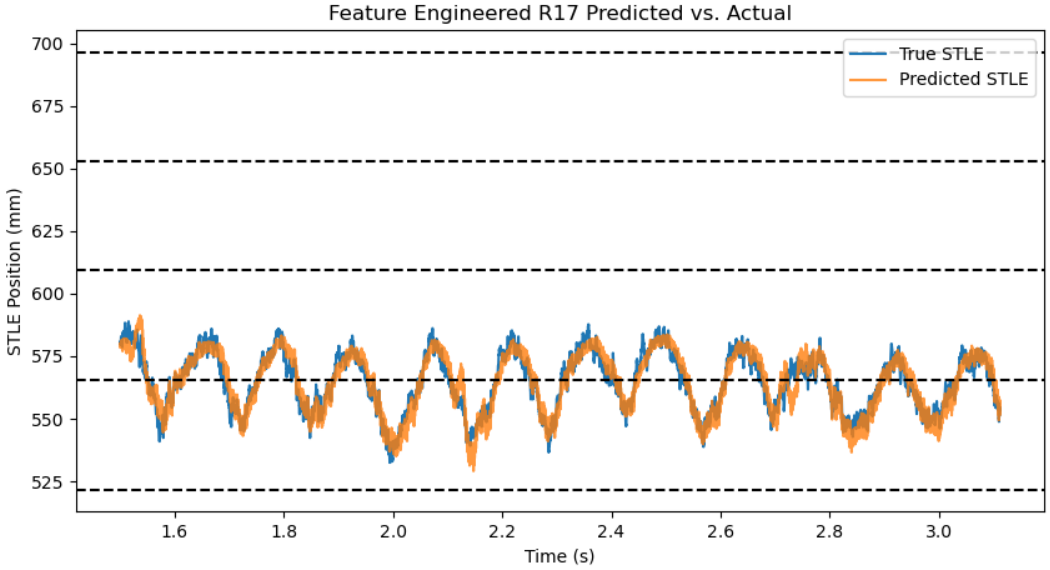


Figure 6: UMDCI R17 Predictions with Feature Engineering

With a simple and inexpensive model, we were able to accurately predict the STLE location for each individual runs. However, fitting individual models for each runs would not ideal especially if it can't generalize into different experimental runs. Hence, our next goal is to create a comprehensive model that could generalize into different runs and, potentially, predict accurately on unseen test runs.

4 Neural Network: Predicting Multiple Runs Simultaneously

Now that we have shown the effectiveness of statistical learning methods in predicting the STLE location, our interest lies on making the model robust. Specifically, can a machine learn from multiple experimental runs to predict on unseen test runs? Hence, the difference between the method used above is that we are creating a single model for a particular dataset. Recall that UMDCI dataset contained 37 experimental runs, each with $>$ 30000 observations, yielding a total of 1151551 observations. Thus, an appropriate model must be selected to take full advantage of the abundance of data.

Figure 7 illustrates an overview of model performance relative to the volume of data. According to the figure, traditional machine learning models tend to have a performance limit regardless of fitting more data. Deep neural networks tend to perform poorly on limited amount of data; however, given abundance of data to train on, they tend to have drastic increase of predictive power. Given the nature of our task, this justifies the choice of fitting a neural network model.

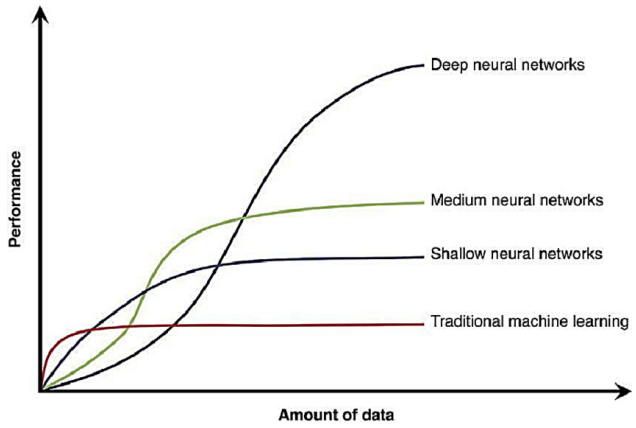


Figure 7: Model Performance Relative to Data Volume

As in our previous method, we will start by defining the training and testing sets for the neural network. We will be training on the following runs: R0, R2, R8, R12, R24, R32, R36, and testing on the following runs: R1, R3, R4, R5, R6, R7, R9, R10, R11, R13, R14, R15, R17, R18, R19, R20, R21, R22,R23, R25, R26, R27, R28, R29, R30, R31, R33, R35, R37 leading to 7 train runs (217861 observations) and 29 test runs (902567 observations). The choice of which runs to be used for training/testing was made to prevent overfitting by minimizing the distributional differences between the two sets.

The neural network model we built was a simple feed-forward network with not too complicated architecture. The cost function used to train the model was MSE, and the model yielded an MSE of 36.9356.

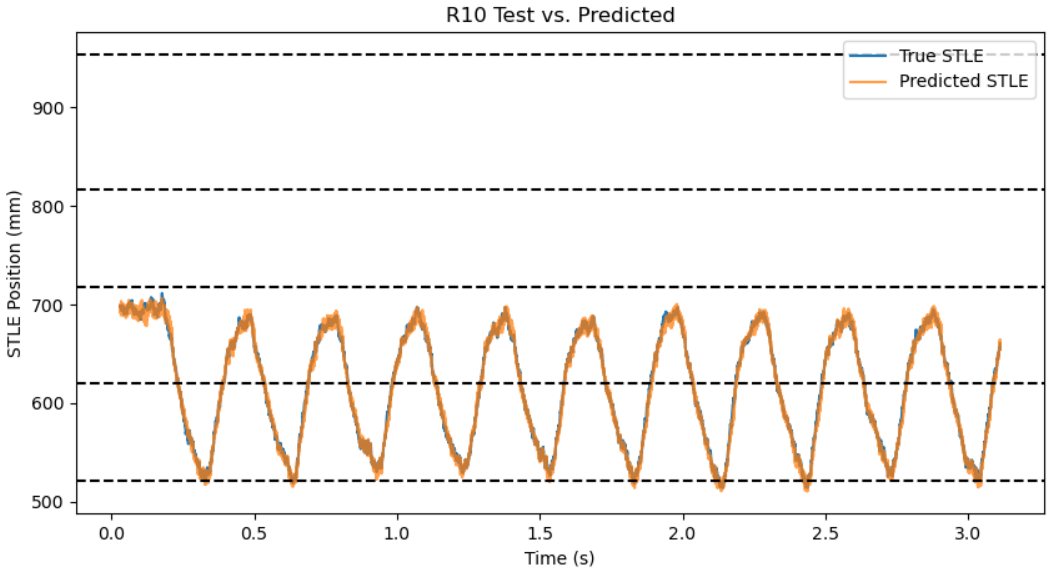
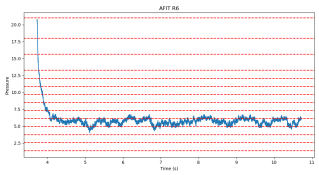
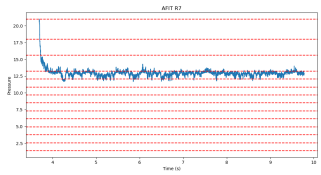


Figure 8: UMDCI R10 Neural Network Predictions

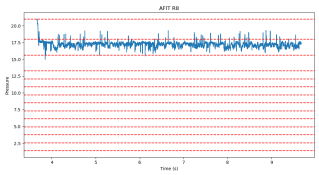
Figure 8 illustrates one of the predictions of our neural network model. Notice that the x-axis ranges from 0 to 3 seconds. The major improvement from the previous method is that the neural network can generalize into experimental runs it has never seen before. Although omitted, the neural network model shows high consistent predictive power with no erroneous runs seen previously in the multiple linear regression model.



(a) R6 Time vs. Pressure



(b) R7 Time vs. Pressure



(c) R8 Time vs. Pressure

Figure 9: AFIT Time vs. Pressure

Thus, we were able to train a model on only 7 out of 37, and predict the rest without overfitting and returning quality results.

However, there is still a problem with the model. Recall that defining training/testing set was not made arbitrary. That is, the assignment of experimental runs were handpicked to prevent overfitting of the model. This was not a concern for the UMDCI dataset because of its large volume of data; however, we face an immediate consequence in the AFIT dataset.

4.1 AFIT

A similar dataset from the AFIT (Air Force Institutde of Techonology) was used to train a neural network model. The major difference between the AFIT and UMDCI dataset was that AFIT contained only three experimental test runs resulting in relatively small volume of data. Figure 9 illustrates the three different experimental runs. Notice that, for each runs, the pressure measurements hovers around a different range over time. Due to the dissimilarity between the runs, we expect the model to perform poorly.

For this task, we considered three cases: (1) Train on R6, Test on R7, R8 (2), Train on R7, Test on R6, R8, (3) Train on R8, Test on R6, R7. Hence, we are considering all possible combinations of train-test split to see if model's performance is dependent on it. The neural network model shares same architecture and feature usage as in the UMDCI dataset.

Figure 10: AFIT Predictions on R6 and R7

Not surprisingly, the model performed poorly on the third case where we trained the model on R8, and tested on R6 and R7. Both predictions of R6 and R7 are shown in Figure 10, and the model completely fails to predict. However, for the first and second case, the model predicted each runs with high accuracy. Hence, the combination of train-test split directly affects the model performance. This result shows that we must have abundant training data that are well-distributed to accurately predict unseen experiments. UMDCI dataset would be an ideal type to train a model on; however, running numerous experiments, if possible, can oftentimes be costly and inefficient. This now shifts our attention to building a robust model that could transfer knowledge from one dataset to another. In context, we would build a pre-trained model of the UMDCI dataset, and use that knowledge to boost performance when predicting the AFIT dataset. This would fix the issue, in general, of lacking previous experiments by utilizing the knowledge learnt before.

5 Transfer Learning: Robust Modeling For Varying Conditions

Transfer learning (TL) is technique in machine learning that allows the use of knowledge learnt from a previous task to boost the performance of a similar current task. TL can be, in general, broken down to two big ares: homogeneous transfer learning and heterogeneous transfer learning. Heterogeneous transfer learning tasks refer to when having a different domain and target space. That is, the feature dimensions of the domain dataset does not match the target dataset. Because of this, heterogeneous transfer learning tasks are generally much more difficult than that of homogeneous transfer learning task. There are numerous methods to alleviate this problem such as minimizing the distribution with the Maximmum Mean Discrepancy (MMD) matric, feature augmentation, and feature mapping. However, many of the TL implementations in the field are focused on image classification through Convolutional Neural Networks (CNNs) and Natural Language Processing (NLP). Nonetheless, few literature in the field has shown practice in regression setting.

In the context of predicting the STLE location, we hope that TL can aid to bridge a gap between the distribution difference of each facility data. In this project, another method of TL called fine-tuning was implemented. Fine-tuning loads a pre-trained model and uses as a base model to train a new dataset. Hence, it is a homogeneous TL task requiring equal dimensions of the domain and target space. Recall that UMDCI dataset was observed an on isolator with five pressure transducers whereas, AFIT dataset contained 14 pressure transducers. Using all of the pressure measurements from AFIT would result in different feature space. Hence, there needs a way to reduce the feature space to match that of UDMCI dataset. There were two approaches to fix this issue: (1) Only select five pressure transducers from the AFIT to match UMDCI, or (2) Manipulate the pre-existing features to create a new set of features independent of transducer numbers. The first approach was not ideal since there would be 2002 total combinations of choosing 5 out of 14 pressure measurements. Conducting such experiment would be inefficient and non-robust when new datasets examined. The second approach is more robust in a sense that the newly created features would be independent of the number of pressure transducers.

To adopt the second approach, we computed the row-wise mean, median, standard deviation, variance, minimum and maximum for each observation. The newly created features were as good as the original features for predicting the UMDCI dataset with the method of section 4. Then this pre-trained UMDCI model was used as a base model to train the AFIT dataset, which also has the new features computed. Unfortunately, the fine-tuned model faced the same problem the AFIT dataset had originally. Thus, to model the distributional differences between the two datasets, more complicated and sophisticated methods of TL might be required, which is not discussed in this paper.

6 Conclusion/Discussion

Locating the STLE via pressure measurements is not simple which is shown throughout the results of the project. Nonetheless, few successes were observed when considering a single dataset at a time. We showed that statistical learning methods are suitable approaches to this particular task, and suggests statistical modeling could solve complex problems aiding the advancement of hypersonic vehicles. Starting off from

a multiple regression model, we made an effort to making the model as robust as possible, attempting to integrate different facilities with varying experimental conditions. Although the TL method shows insufficiency, we hope to see in the near future a more sophisticated approach achieve satisfactory results. There are still numerous computational experiments that could be conducted such as changing the architecture of the neural network model, hyperparameter tuning, pressure variable selection, and feature engineering.