CS 595 Final Project Report

Stephen Brutch and Rocio Jado Puente
Group 4

April 2024

1 Introduction

One of the greatest causes of fatal aviation accidents remains the Loss-of-Control In Flight (LOC-I) which is defined as an unintended departure from aircraft controlled flight. Many of these incidents tend to have root causes in errors performed by the pilot before/after failure [1] and entering LOC regimes in flight envelope [2]. Therefore an analysis to predict pilot input to the control system is studied to mitigate the introduction of dangerous flight conditions. The widely used mathematical model to describe pilot control is [1],

$$\dot{x}(t) = -\frac{1}{T_{Lag}}x(t) + e(t-\tau) \tag{1}$$

$$p(t) = -\frac{K_p}{T_{Lag}^2}(T_{Lag} - T_{Lead})x(t) - \frac{K_p T_{Lead}}{T_{Lag}}e(t-\tau) \tag{2}$$

Where K_p, T_{Lead}, T_{Lag} are pilot model parameters that correspond to pilot gain, pilot lead compensator and pilot lag compensator respectively that are to be estimated. The term $e(t-\tau)$ represents the pitch error that is delayed by τ seconds. Lastly $\dot{x}(t)$ is a 1x1 human pilot control state.

The estimation will look at a certain time history of flight data and will look to predict the time history of the pilot state $x(t)$ and the pilot parameters T_{Lead}, T_{Lag}, K_p. That is the features for the model to learn will be $[\theta_{cmd}, \theta, \theta_{error}, Time]$ and the targets will be $[x(t), T_{Lead}, T_{Lag}, K_p]$,where $\theta_{cmd}, \theta, \theta_{error}$ correspond to the commanded pitch for the pilot, the outputted pitch angle and the pitch error for the flight. The overarching goal will be to combine all of the outputs along with equation 2 to create an estimated pilot control $\hat{p}(t)$ to compare with the true pilot control $p(t)$ from simulation.

Five machine learning and deep learning models will be used to transform this problem into a multi output regression problem. The first three will utilize Sci-Kit Learn [3] to create Linear Regession model with Ridge Regularization, Random Forest, Gradient Boosting Regession models. The last two models will

utilize PyTorch [4] to create Multi Layer Neural Network (NN) and a Physics Informed Neural Network (PINN) which will be shown to only need to utilize 50% less truth data.

2 Related Work - Background

In this study the human pilot model control parameters will be estimated using various machine learning and deep learning models. In this section different approaches to the problem will be outlined.

In [5] they published a dissertation where its main focus is human control model parameter estimation in real-time with an Unscented Kalman Filter (UKF). The results obtained are from simulation experiments as well as human in the loop experiments with an actual test subject operating a joystick responding to a visual stimulus. In this work the goal was to estimate the time varying parameters of the human pilot model in real time using the UKF as estimating those parameters has direct impact on the metrics as well as gives information to the overall stability of the pilot-vehicle system (PVS). Information of the stability of the PVS is crucial as destabilization can lead to pilot induced oscillations and ultimately loss of control of the aircraft.

In [1] they published a thesis where its main focus is to also estimate human control model parameters in real-time using UKF, an online learning Neural Network and Fourier Transform Regression (FTR) methods. The work aims to estimate the parameters using each of the three methods and for both time invariant and variant parameters. The overarching goal is to use those parameters to accurately predict the pilot control and they show accompanying graphs comparing the estimated to the true. This paper shows even with parameter errors being quite significant the estimated pilot control was predicted well meaning these algorithms show promise in the prediction of future pilot actions in the attempt to avoid the onset of dangerous flight conditions.

In [2] is a paper published with AIAA that is a continuation from [1] that shows the results of parameter estimation using UKF and FTR as well as new Physics Informed Neural Network. Each of these three methods look at estimating the parameters for a specific case with the goal of accurately matching the pilot control. These three methods again show even with small error in the parameters the pilot's control is readily predicted.

In [6] is a paper published by NASA with the goal of estimating time varying pilot model parameters using a two-step method using wavelets in the frequency domain and the second method is the use of Maximum Likelihood Estimation (MLE) to estimate the time varying parameters in the time domain.

In [7] they published an AIAA paper that utilized a sub-scale aircraft performing

steady level flight and landing phase flight tests. The data from these experiments is passed through a non-linear least square estimation tools in Simulink to provide parameter estimates and those parameter estimates compute an estimated pilot control. The estimated pilot control is compared to the true pilot control from operating the sub-scale aircraft. Different simplified versions of the pilot mathematical model are used to compute the estimated pilot control and their different configurations are compared to the true.

3 Approach

3.1 Problem Analysis

In this form of the equations the term, $e(t-\tau)$ is denoted in the literature as the time delay by τ seconds. However this variable in this model is an implicit variable which means its estimation has no effect on the equations. For this the estimation of τ will be omitted.

Another limitation encountered is that because there is a large sample of data (100 second flight simulations sampled at 100 Hz) to be trained/fitted on each call of the .fit for SkLearn or training a Neural Network will take several minutes at the least, some methods of mitigation had .fit times of atleast 10 minutes even for a Linear Regression model, so training time is a factor. To mitigate this downsampling of the data to a lower frequency will have to be implemented. For this problem manually downsampling to 1 Hz for the flight data was selected to speed up computation times.

Lastly the Physics Informed Neural Network (PINN) will integrate the pilot model equations into the learning process to add additional components to the loss function. To avoid the expensiveness of performing automatic differentiation for 100's of sets of data to obtain $\dot{x}$, the analytical solution of $x(t)$ will be calculated for the condition $x(t=0)=0$. Equation 1 will be integrated to give,

$$x(t) = -\frac{e(t-\tau)}{T_{Lag}} e^{-t/T_{Lag}} - \frac{e(t-\tau)}{T_{Lag}} \tag{3}$$

Equations 2,3 will form the basis of the PINN's training.

3.2 Data Analysis

3.2.1 Data Visualization

The dataset used consists of many features that would be too hard to visualize all of them. However the pilot parameters to be estimated can be visualized to ensure that the machine learning models see close to a uniform distribution of each the parameters across the Train/Test sets. See figure 1 for the histograms of the pilot parameter values across the Train/Test sets,

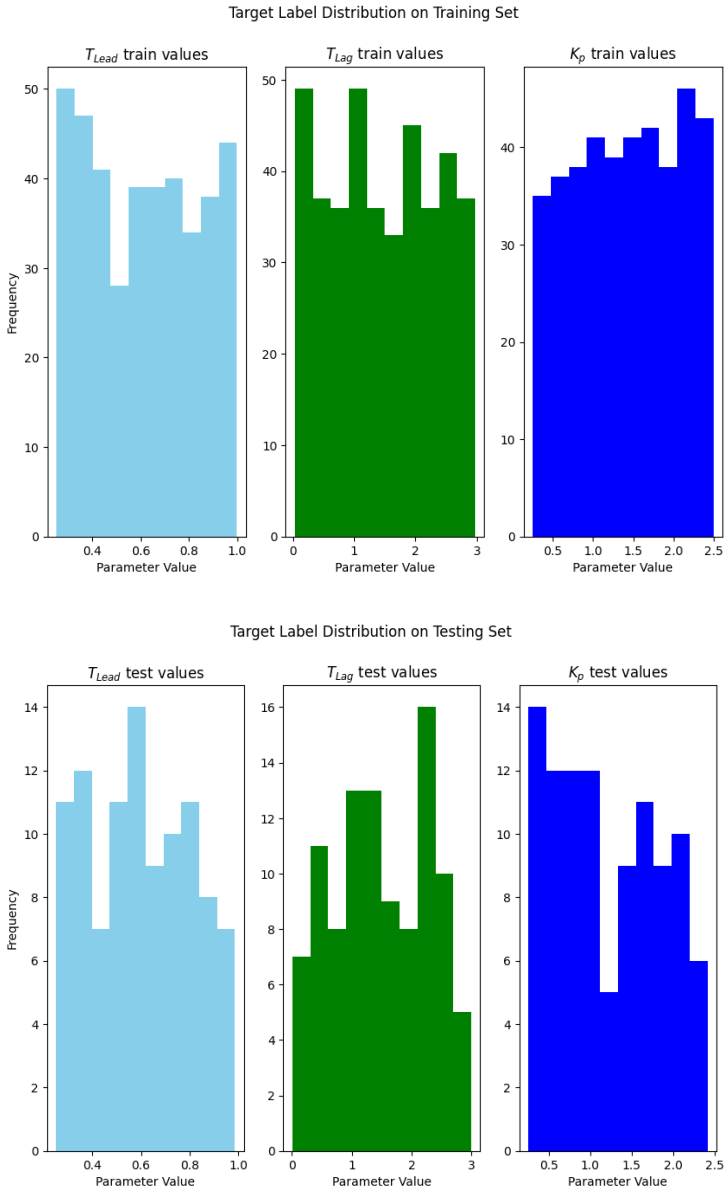

Figure 1: Pilot Parameter Distribution values across Training & Testing Set

3.2.2 Data Normalization

For machine learning models it is generally common to preprocess the input data by either normalization or standardization to obtain more accurate results

[8] than if neither method were utilized. For this study each of the input features were normalized to a scale of (0,1). This can be done for an arbitrary variable, x, such that

$$x_{Normalized} = \frac{x^i - x_{min}}{x_{max} - x_{min}} \tag{4}$$

This technique to normalize the input data will be utilized by each of the five machine learning models.

3.3 Resources Used

The dataset is obtained from Simulink simulations. There is a commanded pitch angle that the simulated pilot is set to follow and the error from the commanded and output pitch signal serves as the pitch error. The $p(t)$ is acting as the input to a standard 3rd order pitch system model from [1]. This simulation takes place at every time step and used to obtain simulation flight data. A visualization of the simulation environment for data collection can be seen in figure 2, where various features of the data simulation runs are collected for training.

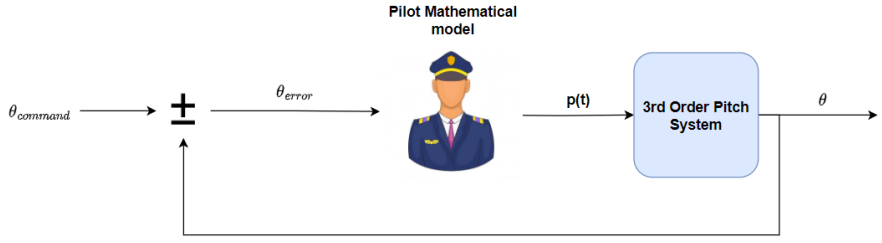

Figure 2: Visualization of simulation environment

The simulation will be used to create 100's of simulation flights with varying parameters to simulate different pilot control.

3.4 Software Design

The proposed solution to address the multi-output regression problem consists of the definition of five models that will be described in this section. As mentioned in the introduction, the first three machine learning models were implemented using scikit-learn libraries in python whereas the two last ones utilize PyTorch machine learning library.

3.4.1 Formation of X & y

For all 5 models the data for the preprocessing steps to formulate the features (X) and the targets (y) was done the same. The downsampled time history data of each input variable (4 total) for each run (500) has a size of 101. In totality there are 500 simulation runs consisting of 4 features each of length 101 which was used to create an X variable of size (500, 101, 4). This structure is not accepted by SkLearn or by Linear Layers of a Neural Network thus this 3-dim input was flattened to create a new X variable of size (500,404). A similar process is considered where each target label will consist of the $x(t)$ of size 101 and the pilot parameters of size 3. This creates a y variable of size (500,104).

3.4.2 Linear Regression with Ridge Regularization

The first model defined will be Linear Regression with Ridge Regularization. Linear Regression is a machine learning algorithm that learns the relationship between features and targets by calculating a weighted sum of the input features, plus a constant known as bias or intercept term. The performance of the model is measured using the Mean Square Error or MSE. One of the problems faced when training a machine learning model is overfitting, where the model cannot learn the underlying structure of data, resulting in a high accuracy prediction of the training data but a poor performance when new data is fed. A common way to prevent overfitting is to increase the bias of the model by constraining the weights of the model. In this work, Ridge Regularization is used, adding a regularization term $\alpha \sum_{i=1}^{n} \theta_i^2$ to the training MSE, being the total cost function $J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} \theta_i^2$ where θ_n are the feature weights, intended to be kept as small as possible. The hyperparameter α that controls the regularization is set to 1 [9].

3.4.3 Random Forest

The second method proposed is a Random Forest Regressor. This bootstrapping algorithm uses an ensemble learning method within a decision tree framework. The decision trees run in parallel, without interaction between them. The results are averaged to output a new result, leading to stronger predictions [9]. In this

work, the number of decision trees is 100, and MSE is utilized as the function to measure the performance of the model.

3.4.4 Gradient Tree Boosting

This sequential ensemble method combines several learning methods (predictors), sequentially adding them to the ensemble, each one using the error from the predecessor as its target for training [9]. The third implemented model is the Gradient Boosted Regression Trees (GBRT), using 100 decision trees as predictors and MSE as loss function, being cons.

3.4.5 Neural Network

The main focus of these next subsections will be how the Neural Network (model 4) and Physics Informed Neural Networks are implemented (model 5). A schematic of the multi layer Neural Network can be seen in Figure 3. The

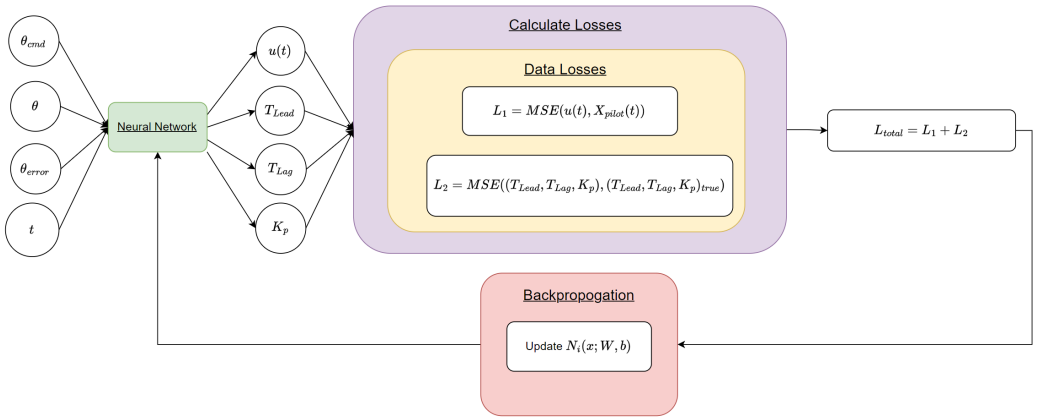

Figure 3: Visualization of Neural Network model

inputs and outputs are the same as the SkLearn models with the only difference being able to decompose the outputs into the predictions of $x(t)$ and the pilot parameters. The true data from the simulation is used to compare the predictions through a mean squared error loss (MSE). The MSE of the $x(t)$ and pilot parameters are summed up to create a total loss term that is then backpropogated to update our Neural Network.

A common way to train Neural Networks is to decompose the entire training set into mini batches which is computationally efficient [10] and allows our network

to update and learn multiple times through each training step. A common approach in the field is to set the batch size to a multiple of 8 and the most popular of those is batch size 32 because it will quicken the computation of the network but will decrease the updates required for the network to reach convergence[11]. Batch size 32 was used for training.

The Neural Network block is used to visually represent a multi layer multiple hidden units Neural Network. The structure of the chosen Neural Network can be seen in Table 1,

Table 1: Structure of Neural Network

Layer	**Input Size**	**Output Size**	**Activation Function**
Input Layer	404	256	ReLU
Hidden Layer	256	512	ReLU
Output Layer	512	104	Linear

The input and output sizes are set numbers while the output size of 256 and 512 from the table are numbers to be experimentally tuned and used. This same method was used to obtain the activation function where the Rectified Linear Unit (ReLU) was used due to its general capabilities to reach convergence, and the output layer has a linear activation which is to say there is no activation applied to the output such that the raw outputs will be used. Many factors such as output size, the amount of hidden layers, and activation function can all be tuned to optimize results however to not incur large training times this structure was found heuristically.

3.4.6 Physics Informed Neural Network

A schematic of the Physics Informed Neural Network (PINN) can be seen in Figure 4. The main difference between this model and the previous Neural Network model is that the PINN integrates the dynamics equations into the training process such that the Neural Network will not only learn to obey the distribution of the data but also the dynamics. Due to the formation of equation 3, the Neural Network outputs all of the variable necessary to perform the calculations all that is to do is perform calculations to ensure the outputs obey the physics equations. The new L_1, L_2 terms are used to check that the output data from the Neural Network obeys equations 3 and 2 respectively. Lastly the L_3 term is adjusted to display the power of the PINN such that 50% less data is necessary to obtain still accurate results. The term is $L_3 = MSE(u(t), x(t))$ where $u(t), x(t)$ only correspond to the first 50% of the predicted and true time history being regulated by the loss function and it will be shown the PINN is able to learn the dynamics and able to fill in the blanks and form accurate predictions for the last 50% of the time history predictions even though it does not see what the true data for the last 50% of the time history is.

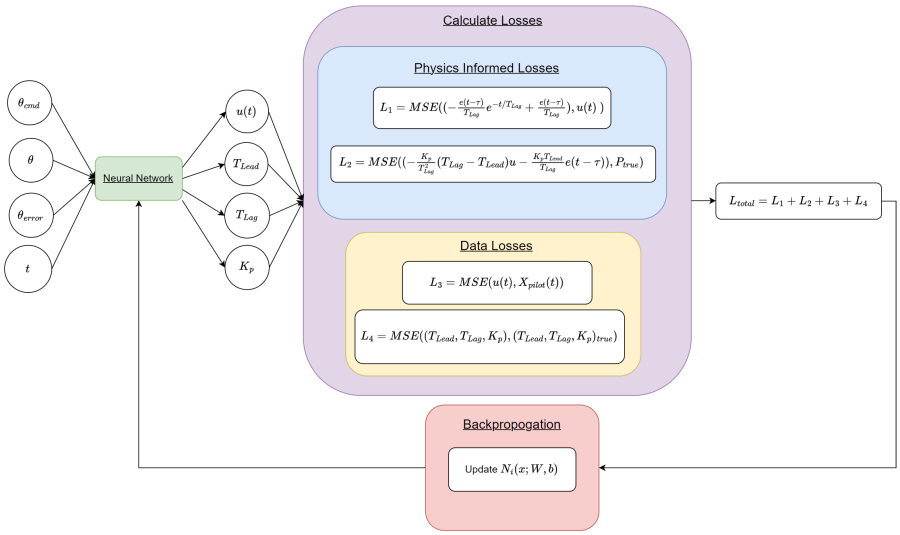

Figure 4: Visualization of Physics Informed Neural Network model

The Neural Network structure is the same as in section 3.4.5.

This formulation uses inspiration from [1] where an adaptive Neural Network learns online by utilizing a similar structure as the one in this study along with equation 2 to generate an estimated pilot control. The estimated pilot control along with the true pilot control is used as the only loss function term. This work is also a continuation from [2] where a Physics Informed Neural Network is used to estimate the pilot parameters as well as plot a comparison of the estimated and true pilot control at the end of training. The main differences is that that study utilized equations 1 and 2 as the basis of the loss function where automatic differentiation was used to obtain $\dot{x}(t)$ from $u(t)$. This study also only looked at obtaining results for 1 specific case, whereas this one looks to learn from hundreds of flight data. Equation 3 was used over equation 1 in this study as performing automatic differentiation for hundreds of simulation runs consisting of hundreds of data points proved to be too expensive computationally and would cause crashes in the Google Colab. To mitigate this it was found in the simulation that the initial condition of $x(t)$ was zero, thus equation 1 was integrated with inital condition equal to zero to obtain equation 3.

3.5 Source Code Description

The description and overview of the files in the source code folder can be seen in Table 2.

Table 2: Structure of Source Code Folder

File Name	Type	Description
paramsdata3	mat	Array of the true pilot parameters
pilotouput3	mat	Array of the true pilot control
pitch_error_data3	mat	Array of the pitch error
pitch_output_data3	mat	Array of the pitch output
pitchcommand3	mat	Array of the pitch commanded
xpilot3	mat	Array of the true $x(t)$ data
CS_595a_Source_Code (2)	ipynb	Source Code

4 Evaluation and Results

4.0.1 Results

The five machine learning models will be trained and tested on the same dataset for comparison. The metrics used to compare performances will be the mean squared error of the true and estimated of the $x(t)$, pilot parameters, and pilot control.

The MSE of the human pilot control state $x(t)$ is seen in Figure 5, where Ridge could be considered as the best-performing model, but it will be discussed in the following subsections. Figure 6 displays the MSE of the pilot model parameters, with GBRT having the smallest error. Finally, the pilot control ($Sp(t)$) is seen in Figure 7, where PINN would be the best performer.

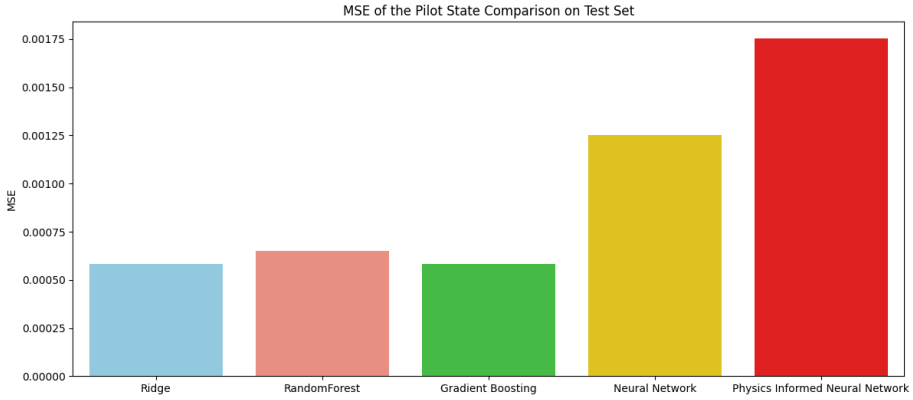

Figure 5: MSE bar plot comparison of $x(t)$

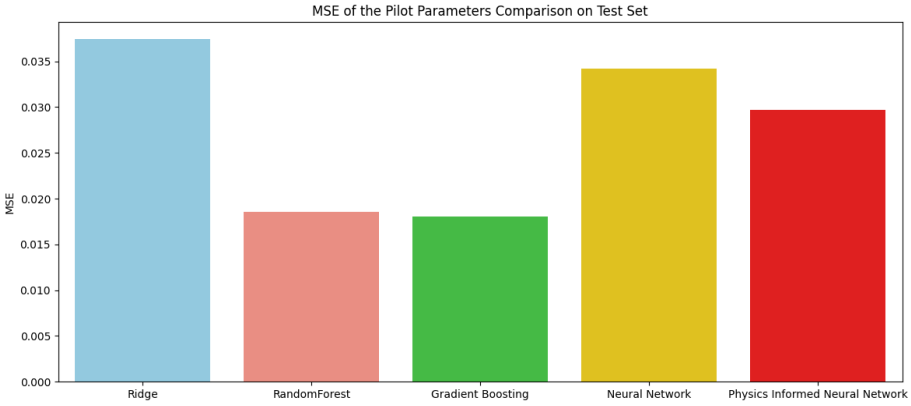

Figure 6: MSE bar plot comparison of parameters

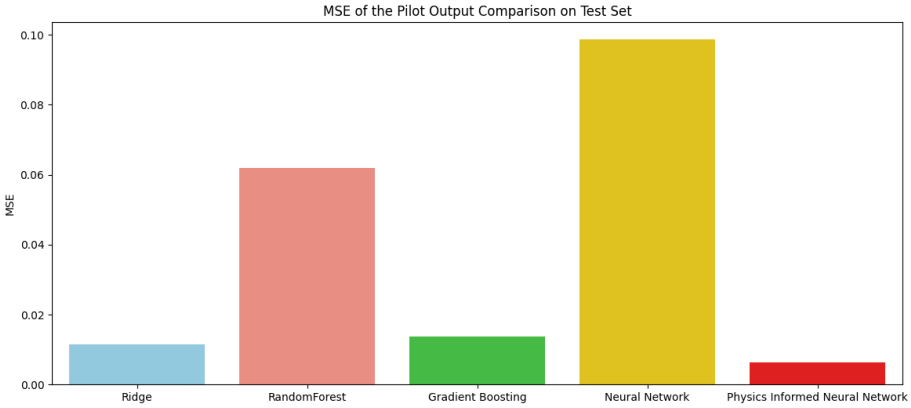

Figure 7: MSE bar plot comparison of pilot control

For comparison, one of the test cases at random will be used to visualize the differences in the predictions of the pilot state $x(t)$ (Figure 8) and pilot control $p(t)$ (Figure 9). In these Figures, it can be observed that the five models' prediction is more accurate for the pilot state.

4.0.2 Results Summary

A summary of the results is displayed in Table 3, where the MSE of the three predictions done (Pilot State $x(t)$, Pilot Parameters, Pilot Control $p(t)$) by the five tested models is shown.

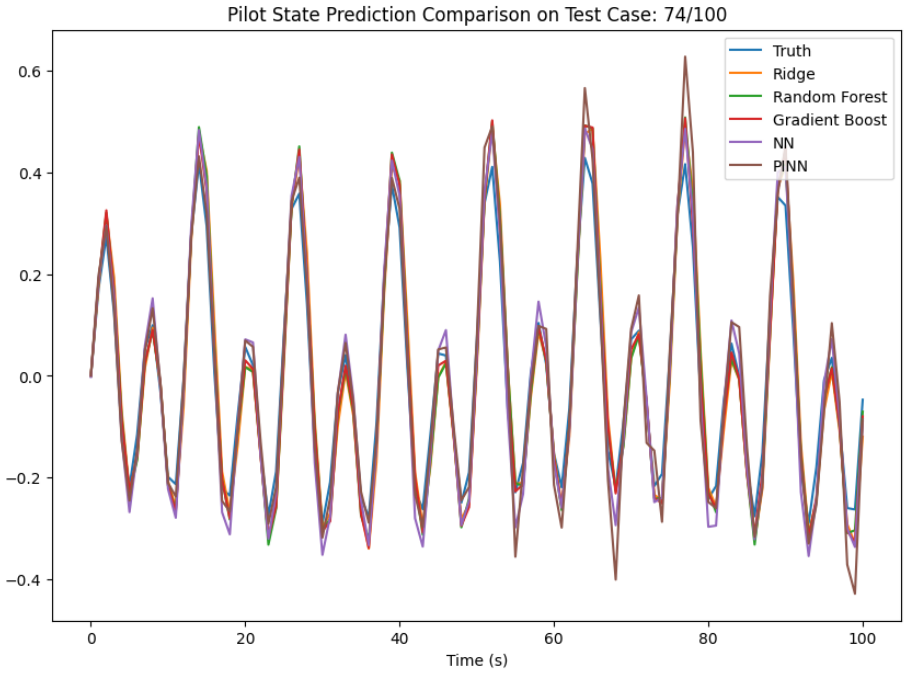

Figure 8: Plot comparison of one of the test cases of $x(t)$

Table 3: Summary of Results

Model	**MSE**		
	x(t) (10^{-4})	**Pilot Parameters**	**Pilot Control**
Ridge	5.8329	0.0374	0.011408
Random Forest	6.5683	0.0190	0.081994
Gradient Boosting	5.8558	0.0187	0.014814
NN	11.5790	0.0345	0.062220
PINN	18.7890	0.0501	0.006312

4.0.3 Results Discussion

The main goal of this study is to estimate human pilot state $x(t)$ and pilot parameters for a specific simulation flight test and use those parameters along with the mathematical model of the pilot to estimate its input to the control system. The first 4 models used (Ridge, RandomForest, Gradient Tree Boosting, Neural Network) are trained to match those parameters directly and the calculation of pilot input is calculated post. While the last model, Phyiscs In-

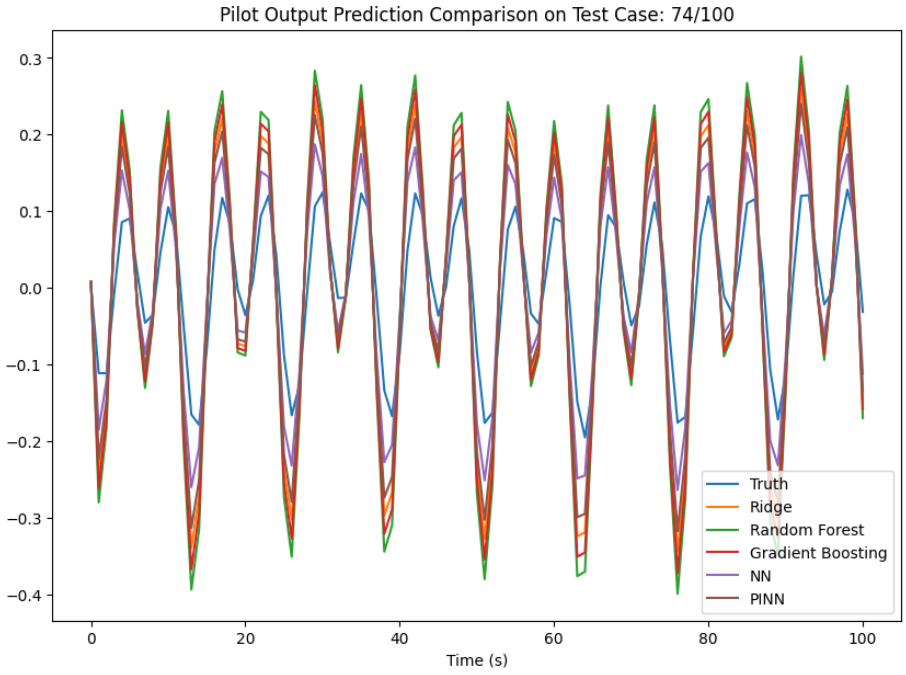

Figure 9: Plot comparison of one of the test cases of $p(t)$

formed Neural Network (PINN), embeds the mathematical model of the pilot into the learning process and because of this 50% less truth data is necessary for this model. Each of the models showed good capabilities of estimating each of the parameters with the PINN performing the best when it came to estimating pilot control, which is to be expected because the pilot control equation is a a loss term that is a part of its training process for it to minimize. Pilot control, equation 2, is in terms of $x(t)$ and the parameters thus if the other four models estimate them well, then pilot control will be estimated well, and that is also seen in Figure 9 and Table 3.

5 Conclusion

The lessons learned from this study were that the SkLearn models even with default parameters showed very promising results at estimating the pilot input even though that equation was not a part of its learning process. Also, the computational cost of data was experienced when using the raw 100 seconds at 100 Hz data was input to even a Linear Regression model saw .fit() times of nearly 10 minutes. The cost of downsampling to 1 Hz proved much quicker .fit() times for all models and saw promising results even with the shorter time sequence.

The Neural network models also showed promise as only 1 hidden layer, and only linear layers were used in the network. The cost of training a model with different structures like CNN, RNN, and LSTM can prove to be much more computationally intensive before and even after downsampling. The computational cost of using those different structures may not even be warranted as the linear layer Neural Network performs very well.

5.1 Future Work

Next steps for this research could be too train and test the trained models on different commanded pitch signals to give the model more diversity in its data. The trained models were given a commanded pitch signal that was sinusoidal to excite different frequencies of the pilot [1] and these models could be tested on for example, step signals of pitch to test for their robustness and if necessary be trained on step signals.

Also in the literature their are attempts to estimate these parameters as if they were time varying whereas in this study the parameters are assumed to be constant. The assumption of constant given a 100 second time history is consistent with the literature as these parameters are assumed to be very low frequency, however a framework for these models could be developed to test their robustness on time varying pilot parameters.

References

[1] F. Schill, "Online pilot model parameter estimation for loss-of-control prevention in aircraft systems," 2022.

[2] S. Brutch and H. Moncayo, "Machine learning approach to estimation of human-pilot model parameters," in *AIAA SCITECH 2024 Forum*, p. 1200, 2024.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "scikit-learn: Machine learning in Python," 2011. Accessed: 02,21, 2024.

[4] P. Contributors, "Pytorch documentation," 2022. Accessed: 02,21, 2024.

[5] T. K. Mandal, *Real-Time Monitoring and Prediction of the Pilot Vehicle System (PVS) Closed-Loop Stability*. West Virginia University, 2016.

[6] P. Zaal and B. Sweet, "Estimation of time-varying pilot model parameters," in *AIAA Modeling and Simulation Technologies Conference*, p. 6474, 2011.

[7] T. Kumar and Y. Gu, "Pilot vehicle system modeling using sub-scale flight experiments," in *AIAA Modeling and Simulation Technologies Conference*, p. 1004, 2014.

[8] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.

[9] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[10] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018.

[11] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.

6 Appendix

6.1 Key Contributions from Stephen Brutch

Stephen's contributions were the data preprocessing steps and the creation of the Neural Network and Physics Informed Neural Network models. The data preprocessing step is an important one, as most practitioners will say that preprocessing/cleaning the data takes up most of the time when in the lifecycle of building a ML/DL model. The formation of X,y were crucial as those are the direct input, output pairs the model will be trained and tested on. This provided experience in researching not only the problem the project is about but also researching how to solve certain coding errors that were encountered in the processing steps.

Stephen also worked on both Neural Network models. Many different architectures and hyperparameters could have been chosen however 1 hidden layer consisting of linear layers was chosen. ALso the hyperparameters of using the Adam optimizer, ReLU activation function, 100 epochs and 80/20 train/test split were chosen. The code for the Physics Informed Neural Network is an extension from Stephen's graduate research work so this solution is a continuation from that.

6.2 Key Contributions from Rocio Jado Puente

Rocio was focused on building the machine learning models using the data available (models 1, 2, 3). For this matter, she decided that using scikit-learn libraries was the best course of action since they are simple and well-known, studied in class. It is worth mentioning that the first approach for model1 was

linear regression without any regularization, but after the poor performance obtained and further research, it was decided that Ridge Regularization had to be implemented.

While developing these models, training, and testing them, she has gained some experience in the use of the different python libraries that will be very valuable in her future research.

She supported Stephen's previous background research with some more work done on the topic, where she could learn the different approaches taken to solve this problem.