Predicting Aircraft Performance Indicators Using Artificial Neural Networks

Chris Moore
The University of Oklahoma
Gallogly College of Engineering, School of Electrical & Computer Engineering
Chris.moore@ou.edu

Abstract

Purpose –The purpose of this study is to make predictions about some performance indicators of a turbofan aircraft by utilizing modern Artificial Neural Network (ANN) models.

The problem of determining optimum performance in today's modern turbofan aircraft is a very old problem with real test data to back up calculated predictions. Being a very old industry, aerospace has been slow to adopt the use of Artificial Neural Networks to predict the performance of new engine designs. However, with modern ANN models the real-world test data can be utilized to predict engine and aircraft performance with a high degree of precision. By using an ANN to predict performance, time spent calculating the impact of design changes can be reduced.

Objective

This study is intended to complement the presentation that analyzed other studies that attempted to use ANN models to predict performance indicators of an aircraft based on a few measured inputs. The goal will be to expand upon the work by attempting to perform the same predictions using a simple ANN model, the Levenberg-Marquardt (LM) and Bayesian regularization (BR) algorithms, as well as the ReLU activation function. This will be accomplished by creating the ANN in Python along with utilizing PyTorch.

The inputs that will be used to train and test the model will be:

- Airspeed
- Altitude
- Air Temperature
- Exhaust Temperature
- Cruise Mach Number

The expected outputs will be:

- Thrust
- Ground Speed
- Fuel Consumption

Approach

From concept to development, this whole project was meant as an experiment to see how different algorithms affect an ANN model that contained layers with the ReLU activation function. To do this, models were built in Python utilizing PyTorch to attempt to replicate models similar to those described in academic papers that were attempting to solve the same problem of aircraft performance with an ANN. Several iterations of the models were tried with different techniques and approaches.

Datasets

Locating sufficient sample data proved to be a significant unexpected challenge. By researching through many aerospace databases and repositories a source was finally identified. The sample flight data was acquired from NASA's Open Data Portal. This data contained sensor data from several aircraft over several flights, which is exactly what was needed to train and test the ANN. However, this came with a problem that was unaccounted for originally. That is that the datasets were provided in a manner that proved difficult to extract the necessary information.

Moreover, the datasets didn't include any of the data about the aircraft itself or its preflight conditions. Because this issue makes it impossible to predict some parameters, it became necessary to adjust the scope and expected outcomes of the project. The inputs, as well as the outputs, were scaled back to account for the lack of qualitative data.

To compile a sufficient amount of data, the sensor inputs were extracted from the flight data and tabulated in an excel spreadsheet. Because aircraft sensors are monitoring different sections of the aircraft, all of the data was presented with a "rate" value between 1-4 that indicated the frequency of the sensor sampling. This made it necessary to scrub the data after it was extracted to ensure that the variables aligned, and they contained the same number of samples.

The data was further manipulated to make it simpler, as well as more efficient, when training the ANN models. Manipulating the data was necessary because the flight data came from aircraft that had 4 engines, each with their own exhaust temperatures, thrust, and fuel consumption. A simple average was applied across the 4 engines for these 3 parameters, thus treating it as though the aircraft had only one engine.

Ultimately, flight data from two aircraft over several flights were extracted and manipulated. This resulted in nearly 23,000 sensor reading sets, acquired from the 8 sensors, to be split and used as a training set, a validation set, and a test set.

Models

To attempt to isolate the variables of interest, which were the Levenberg-Marquardt and Bayesian regularization algorithms and the ReLU activation function, several basic models were built that contained an input layer, a hidden layer, and an output layer. These models were implemented using Pytorch Lightning to make changes to the models easier. These basic models then had trainers created for the LM and BR algorithms. Lastly, the ReLU activation function was used to attempt to add some non-linearity to the data.

The models were run one after the other in the program so that the outputs could be compared after adjustments to the parameters. This also allowed for the model components to be isolated for review after each attempt to train and test the models.

Evaluation

The models were evaluated by comparing their mean squared error (MSE) after each training, validation, and test attempt. MSE was chosen as the measure of the model because of the size and complexity of the dataset being used. The MSE was the metric used to attempt identification of an architecture that worked best for the data being used.

However, finding an ideal model proved to be an issue that still remains unsolved. Although the MSE could be lowered by making some adjustments, the MSE was significantly higher than anticipated. Many adjustments and tests were performed to identify what was causing the issue.

Another hurdle in attempting to replicate these experiments was dealing with the negative temperatures that were observed during flight. That was a consideration that was overlooked when designing this experiment. Because there was a large portion of negative data in the dataset, it would suggest that the ReLU function was making that data 0, thus causing issues while training.

To illustrate, this first set of test results were one of the better outcomes and came from a model that had 3 fully connected layers, Adam optimizer, a dropout rate of 0.2, and LeakyReLU to try and mitigate the negative temperature data.

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name      | Type      | Params
-----------------------------------------
0 | criterion | MSELoss   | 0
1 | fc1       | Linear    | 768
2 | fc2       | Linear    | 16.5 K
3 | fc3       | Linear    | 387
4 | dropout   | Dropout   | 0
5 | relu      | LeakyReLU | 0
6 | flatten   | Flatten   | 0
-----------------------------------------
17.7 K    Trainable params
0         Non-trainable params
17.7 K    Total params
0.071     Total estimated model params size (MB)
`Trainer.fit` stopped: `max_epochs=100` reached.
Training MSE: 55276.80859375
Validation MSE: 24633.94140625
```

Test metric	DataLoader 0
test_loss	24633.94140625

As can be seen, the error is significant even with a basic model.

The next image shows the results from a similar model, except using the Bayesian regularization algorithm (BR) while training the model. The loss is similarly high, as with the previous example.

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name      | Type      | Params
-----------------------------------------
0 | criterion | MSELoss   | 0
1 | fc1       | Linear    | 768
2 | fc2       | Linear    | 16.5 K
3 | fc3       | Linear    | 387
4 | dropout   | Dropout   | 0
5 | relu      | LeakyReLU | 0
6 | flatten   | Flatten   | 0
-----------------------------------------
17.7 K    Trainable params
0         Non-trainable params
17.7 K    Total params
0.071     Total estimated model params size (MB)
`Trainer.fit` stopped: `max_epochs=100` reached.
Training MSE: 44578.57421875
Validation MSE: 29030.26953125
```

Test metric	DataLoader 0
test_loss	29030.26953125

While testing, it was observed that there didn't seem to be any single parameter, or mix of parameters, that made a significant difference to the loss. The models were built and rebuilt several times to try and correct this issue, but the loss stayed within the bounds of roughly 20,000-50,000 depending on how many layers were chosen, how many epochs to train for, the dropout rate, or any other parameter adjustment.

Conclusion

To conclude, this experiment ended up being a failure as well as something of a success. The failure is that there is an obvious issue somewhere in the experimental setup. Due to a lack of time, the issue was never identified.

However, the success came from the ideas and concepts learned from the failures. These experiments ended up being a learning experience in the importance of choosing appropriate data and choosing an appropriate approach to using that data. If either of these are chosen poorly the results will also be poor.

It is strongly suspected that the dataset used for the experiments was the cause of many of the problems encountered while performing this experiment. A cleaner set of non-negative data would likely yield better results than were achieved. A non-negative data set would avoid the issue with ReLU converting all negative numbers to 0s.

If this experiment were to be continued, the next logical step would be to analyze the data to determine if there are any issues with the structure or quality of the data used in the model. Following that, further research on innovative approaches to analyzing complex data, such as aircraft sensor data, using ANNs would provide some guidance in determining a better experimental setup.

Addendum

This addendum is intended to present new information surrounding the skewed MSE losses that were observed during experimentation. Although this research project has already been presented in the forum that it was written for, subsequent readers may benefit from the clarifications provided here.

After concluding the experiments and presenting this paper, it was discovered that there were some human errors evident in the programming which caused inflated loss numbers. The program was debugged and tested multiple times to ensure the errors were corrected. As an example, the below image shows the new MSE that was observed using the Adam Optimizer.

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name      | Type      | Params
-----------------------------------------
0 | criterion | MSELoss   | 0
1 | flatten   | Flatten   | 0
2 | fc1       | Linear    | 768
3 | fc2       | Linear    | 16.5 K
4 | fc3       | Linear    | 387
5 | dropout   | Dropout   | 0
6 | relu      | LeakyReLU | 0
-----------------------------------------
17.7 K    Trainable params
0         Non-trainable params
17.7 K    Total params
0.071     Total estimated model params size (MB)
`Trainer.fit` stopped: `max_epochs=100` reached.
Training MSE: 0.0017307059606537223
Validation MSE: 0.0010246665915474296
```

Test metric	DataLoader 0
test_loss	0.000944719125982374

As shown in the above image, the MSE was significantly lower than was previously observed during the model tests. All three of the models were retested and their MSE was captured using the same data set from previous tests.

Retesting the models started to show a pattern of which model was most efficient for this type of data. On repeated tests the lowest MSE was observed with the Bayesian regularization model, with the Levenberg-Marquardt model following second.

Bayesian regularization model

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name      | Type      | Params
-----------------------------------------
0 | criterion | MSELoss   | 0
1 | flatten   | Flatten   | 0
2 | fc1       | Linear    | 1.0 K
3 | fc2       | Linear    | 28.1 K
4 | fc3       | Linear    | 504
5 | dropout   | Dropout   | 0
6 | relu      | LeakyReLU | 0
-----------------------------------------
29.6 K    Trainable params
0         Non-trainable params
29.6 K    Total params
0.118     Total estimated model params size (MB)
`Trainer.fit` stopped: `max_epochs=100` reached.
```

```
Training MSE: 0.0014634444378316402
Validation MSE: 0.0007065354147925973
```

Test metric	DataLoader 0
test_loss	0.0006854256498627365

Levenberg-Marquardt model

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name      | Type      | Params
-----------------------------------------
0 | criterion | MSELoss   | 0
1 | flatten   | Flatten   | 0
2 | fc1       | Linear    | 768
3 | fc2       | Linear    | 16.5 K
4 | fc3       | Linear    | 387
5 | dropout   | Dropout   | 0
6 | relu      | LeakyReLU | 0
-----------------------------------------
17.7 K    Trainable params
0         Non-trainable params
17.7 K    Total params
0.071     Total estimated model params size (MB)
`Trainer.fit` stopped: `max_epochs=100` reached.
Training MSE: 0.0014377435436472297
Validation MSE: 0.0007794809644110501
```

Test metric	DataLoader 0
test_loss	0.0007170778117142618

In summary, the ANN models were tested successfully with the modifications made to the test program and the model with the least MSE loss was identified as the best model to use for analyzing flight data. Further research could also identify other models that could be used to monitor aircraft flight data to present a clearer picture of aircraft performance. Engineers can subsequently use that information to improve aircraft performance and efficiency; thus, saving time, money, and resources.