Using LSTM Recurrent Neural Networks to Predict Excess Vibration Events in Aircraft Engines

AbdElRahman ElSaid*, Brandon Wild†, James Higgins†, Travis Desell*

Department of Computer Science*, Department of Aviation†
University of North Dakota

University of North Dakota
Motivation: General Aviation Safety

General aviation comprises 63% of all civil aviation activity in the United States; covering operation of all non-scheduled and non-military aircraft [2, 4].

While general aviation is a valuable and lucrative industry, it has the highest accident rates within civil aviation [3].

For many years, the general aviation accident and fatality rates have hovered around 7 and 1.3 per 100,000 flight hours, respectively [1].

Motivation: The National General Aviation Flight Database

The National General Aviation Flight Information Database (NGAFID) has been developed at the University of North Dakota as a central repository for general aviation flight data. It consists of per-second flight data recorder (FDR) data from three fleets of aircraft.

As of June 2016, the database stores FDR readings from over 300,000 flights, consisting of over 550,000 flight hours with more being added daily. It currently stores over 1.2 billion per-second records of flight data (~2TB). The NGAFID provides an invaluable source of information about general aviation flights, as most of these flights are from aviation students, where there is a wider variance in flight parameters than what may normally be expected within data from professionally piloted flights.
Motivation: The National General Aviation Flight Database

Time series flight data for this work was gathered from the NGAFID, and this has been made available publicly for other interested researchers:

http://people.cs.und.edu/~tdesell/ngafid_releases.php
Motivation: Flight Data Prediction

Having the ability to predict flight parameters based on multiple other parameters as input is a first step towards developing sensors which can intelligently detect anomalous behavior or predict accident precursor behavior. Bringing machine learning strategies into flight data analysis and accident prediction has great potential for preventing future accidents in a proactive manner.

Further, these same strategies can be used to predict and prevent hardware failures or suggest pre-emptive maintenance, reducing costs for airlines.
Motivation: Flight Data Prediction

Various parameters contribute to engine vibration:

- engine design
- size
- service life span
- aircraft type
- placement
- weather
- pilot action
- etc.
Motivation: Flight Data Prediction

Much work has been done to generate physical models to predict vibration, however these are tied to all these parameters which may not be readily available.

The goal is to create a system which can generically predict vibration using FDR data.
Long-Short-Term-Memory Recurrent Neural Networks

"Learning to store information over extended period of time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error backflow."


Long-Short-Term-Memory Recurrent Neural Networks

Typical flight data involves between 10-100s of flight parameters gathered potentially multiple times per second and potentially asynchronously. An average flight in the NGAFID with 1 hz sampling has ~5800 per second records.

LSTM RNNs provide a solution to training what would otherwise be extremely deep RNNs.
LSTM Cell Design

The following gates control flow through a LSTM neuron:

1. the input gate, which controls how much information will flow from the inputs of the cell
2. the forget gate, which controls how much information will flow from the cell-memory
3. the output gate, which controls how much information will flow out of the cell.

This design allows the network to learn not only about the target values, but also about how to tune its controls to reach the target values.
LSTM RNN Architectures

LSTM neurons were arranged into three different architectures and trained to predict the vibration parameter 5, 10 and 20 seconds into the future.
First Layer(s) LSTM Cells (M1)
Second Layer LSTM Cells (M2)
Architecture I - 21,170 weights
Architecture II - 21,160 weights
Architecture III - 83,290 weights
Implementation

Python's Theano Library was used to implement the neural networks.

Main benefits included Theano's ability to compute error gradients (as opposed to manually deriving these) and an efficient implementation.
Experimental Data

The following parameters were used as inputs to the RNNs (normalized between 0 and 1):

1. Altitude  
2. Angle of Attack  
3. Bleed Pressure  
4. Turbine Inlet Temperature  
5. Mach Number  
6. Primary Rotor/Shaft Rotation Speed  
7. Secondary Rotor/Shaft Rotation Speed  
8. Engine Oil Pressure  
9. Engine Oil Quantity  
10. Engine Oil Temperature  
11. Aircraft Roll  
12. Total Air Temperature  
13. Wind Direction  
14. Wind Speed  
15. Engine Vibration
Training and Testing Data

Training set:
  28 flights
  41,431 seconds of data

Testing set:
  57 flights
  38,126 seconds of data
Activation Function

Sigmoid function performed significantly better than ArcTan, which resulted in distorted results.
Training Metrics

\[
\text{Error} = \frac{0.5 \times \sum (\text{Actual Vib} - \text{Predicted Vib})^2}{\text{Testing Seconds}}
\]

\[
\text{Error} = \sum [\text{ABS}(\text{Actual Vib} - \text{Predicted Vib})] \frac{1}{\text{Testing Seconds}}
\]

Both mean squared error (MSE, top) and mean absolute error (MAE, bottom) were used to evaluate the RNNs.

MSE was used for training as it provided a smoother search space than MAE.
RNN Training

The RNNs were trained for 575 epochs on a 3.5 GHz 12 core Mac Pro.

### TABLE III

<table>
<thead>
<tr>
<th>Architecture</th>
<th>5 seconds</th>
<th>10 seconds</th>
<th>20 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.000398</td>
<td>0.000972</td>
<td>0.001843</td>
</tr>
<tr>
<td>II</td>
<td>0.001516</td>
<td>0.001962</td>
<td>0.002870</td>
</tr>
<tr>
<td>III</td>
<td>0.000409</td>
<td>0.000979</td>
<td>0.001717</td>
</tr>
</tbody>
</table>

### VI. IMPLEMENTATION

#### A. Programming Language

Python’s Theano Library [9] was used to implement the neural networks. It has four major advantages:

i) it will compile the most, if not all, of functions coded using it to C and CUDA giving fast performance,

ii) it will perform the weights updates for back propagation with minimal overhead,

iii) Theano can compute the gradients of the error (cost function output) with respect to the weights saving significant effort and time needed to manually derive the gradients, coding and debugging them, and finally,

iv) it can utilize GPU’s for further increased performance.

#### B. Data Processing

The flight data parameters used were normalized between 0 and 1. The sigmoid function is used as an activation function over all the gates and inputs/outputs. The ArcTan activation function was tested on the data, however it gave distorted results and sigmoid function provided significantly better performance.

#### C. Machine Specifications

Each of the examined architectures runs on a hyperthreaded 3.5 GHz core and is considered capable of real-time processing. Results were collected using a Mac Pro with 12 logical cores, with each different architecture being trained for 575 epochs. Run times for training are shown in Table IV. Some unexpected variance might be realized in these run-times, due to CPU interruptions which may have occurred over the course of the experiments. In general, the first two architectures took similar amounts of time (approximately 8.5-9 hours) for each time prediction (5, 10 and 20 seconds), and the third took a bit more than twice as long, at approximately 20 hours for each time prediction.

### VII. RESULTS

The neural networks were run against flights that suffered from the excessive vibration in a training phase. They were then run against different set of flights, which also suffered from the same problem, in a testing phase. There were 28 flights in the training set, with a total of 41,431 seconds of data. There were 57 flights in the testing set, with a total of 38,126 seconds of data. The networks were allowed to train for 575 epochs to learn and for the cost function output curve to flatten.

### TABLE IV

<table>
<thead>
<tr>
<th></th>
<th>05</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture I</td>
<td>9</td>
<td>8.98</td>
<td>8.85</td>
</tr>
<tr>
<td>Architecture II</td>
<td>8.44</td>
<td>8.41</td>
<td>8.4</td>
</tr>
<tr>
<td>Architecture III</td>
<td>21.6</td>
<td>19.7</td>
<td>18.5</td>
</tr>
</tbody>
</table>

Fig. 7. Cost function plot for ART III predicting vibration in 10 future seconds

#### A. Cost Function

Mean squared error was used to train the neural networks as it provides a smoother optimization surface for backpropagation. A sample of the cost function output can be seen in Figure 7. The Figure is a logarithmic plot for architecture III, for predicting vibrations 10 seconds in the future.

#### B. Architecture Results

Mean Squared Error (MSE) (shown in Equation 7) was used as an error measure to train the three architectures, which resulted in values shown in Table V. Mean Absolute Error (MAE) (shown in Equation 8) is used as a final measure of accuracy for the three architectures, with results shown in Table VI. As the parameters were normalized between 0 and 1, the MAE is also the percentage error.

\[
Error = 0.5 \times (Actual\ Vibration - Predicted\ Vibration)^2
\]

\[
Error = \left| Actual\ Vibration - Predicted\ Vibration \right|
\]
Training Results (MSE)

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Error at 5 seconds</th>
<th>Error at 10 seconds</th>
<th>Error at 20 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture I</td>
<td>0.000398</td>
<td>0.000972</td>
<td>0.001843</td>
</tr>
<tr>
<td>Architecture II</td>
<td>0.001516</td>
<td>0.001962</td>
<td>0.002870</td>
</tr>
<tr>
<td>Architecture III</td>
<td>0.000409</td>
<td>0.000979</td>
<td>0.001717</td>
</tr>
</tbody>
</table>
## Testing Results (MSE and MAE)

### Mean Squared Error

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Error at 5 seconds</th>
<th>Error at 10 seconds</th>
<th>Error at 20 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture I</td>
<td>0.001165</td>
<td>0.002926</td>
<td>0.010427</td>
</tr>
<tr>
<td>Architecture II</td>
<td>0.009708</td>
<td>0.009056</td>
<td>0.012560</td>
</tr>
<tr>
<td>Architecture III</td>
<td>0.002386</td>
<td>0.004780</td>
<td>0.041417</td>
</tr>
</tbody>
</table>

### Mean Absolute Error

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Error at 5 seconds</th>
<th>Error at 10 seconds</th>
<th>Error at 20 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture I</td>
<td>0.033048</td>
<td>0.055124</td>
<td>0.101991</td>
</tr>
<tr>
<td>Architecture II</td>
<td>0.097588</td>
<td>0.096054</td>
<td>0.112320</td>
</tr>
<tr>
<td>Architecture III</td>
<td>0.048056</td>
<td>0.070360</td>
<td>0.202609</td>
</tr>
</tbody>
</table>
Architecture I Predictions

5s

10s

20s
Testing Results: Architecture I predicting 5, 10, 20 sec
Testing Results: Architecture II predicting 5, 10, 20 sec
Testing Results: Architecture III predicting 5, 10, 20 sec
Conclusions

Architecture I provided the best predictions:
  3.3% MAE for 5 seconds
  5.51% MAE for 10 seconds
  10.19% error for 20 seconds

Architecture III could potentially be trained longer.

RNNs did not train well on GPUs - needs future examination.
QUESTIONS?