Evolving Neural Network Weights for Time-Series Prediction of General Aviation Flight Data

Travis Desell[1], Sophine Clachar, James Higgins[2], and Brandon Wild[2]

[1] Department of Computer Science, University of North Dakota
tdesell@cs.und.edu, sophine.clachar@my.und.edu
[2] Department of Aviation, University of North Dakota
jiggins@aero.und.edu, bwild@aero.und.edu

Abstract. This work provides an extensive analysis of flight parameter estimation using various neural networks trained by differential evolution, consisting of 12,000 parallel optimization runs. The neural networks were trained on data recorded during student flights stored in the National General Aviation Flight Database (NGAFID), and as such consist of noisy, realistic general aviation flight data. Our results show that while backpropagation via gradient and conjugate gradient descent is insufficient to train the neural networks, differential evolution can provide strong predictors of certain flight parameters (10% over a baseline prediction for airspeed and 70% for altitude), given the four input parameters of airspeed, altitude, pitch and roll. Mean average error ranged between 0.08% for altitude to 2% for roll. Cross validation of the best neural networks indicate that the trained neural networks have predictive power. Further, they have potential to act as overall descriptors of the flights and can potentially be used to detect anomalous flights, even determining which flight parameters are causing the anomaly. These initial results provide a step towards providing effective predictors of general aviation flight behavior, which can be used to develop warning and predictive maintenance systems, reducing accident rates and saving lives.

Keywords: Time-Series Prediction, Asynchronous Differential Evolution, Neural networks, Flight Prediction, Aviation Informatics

1 Motivation

General aviation comprises 63% of all civil aviation activity in the United States; covering operation of all non-scheduled and non-military aircraft [12, 24]. While general aviation is a valuable and lucrative industry, it has the highest accident rates within civil aviation [21]. For many years, the general aviation accident and fatality rates have hovered around 7 and 1.3 per 100,000 flight hours, respectively [1]. The general aviation community and its aircraft are very diverse, limiting the utility of the traditional flight data monitoring (FDM) approach used by commercial airlines.

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 January 2014, the database stores FDR readings from over 208,000 flights, with more being added daily. It currently stores almost 400 million per-second records of flight data. 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.

This research presents initial work done using data from the NGAFID for the prediction of flight data parameters. Five flights were selected from the NGAFID, and a rigorous examination of training the weights to various neural networks using backpropagation and differential evolution [23] was performed. Backpropagation is shown to be insufficient to train the neural networks. 12,000 runs of parallel differential evolution with various search parameters were executed on a high performance computing cluster, testing 15 different neural network designs. The results show that it is possible to have significant improvement over a baseline random noise estimator for airspeed and altitude. The best neural networks were cross-validated on flights they were not trained on, showing predictive ability and the potential to detect anomalous flights. These results provide a first step towards accurate prediction of FDR parameters, which could not only warn pilots of problematic flight behavior, but also be used to predict impending failures of engines and other hardware. This has the potential to reduce costs for maintaining general aviation fleets, and more importantly save lives.

2 Time-Series Prediction

Neural networks have been widely used for time series data prediction [7, 29], however to the authors' knowledge, this is the first attempt to utilize them in predicting general aviation flight parameters. Our approach is similar to Khashei *et al.* [16] and Omer *et al.* [22], who utilize *residuals*, or *lags*, from linear data as additional inputs to the neural network. A significant difference is that this work uses multiple streams of input time-series data (airspeed, altitude, pitch and roll) to predict future values of each of those parameters; instead of prediction on single parameter time series data. This allows us to exploit dependencies between the input parameters.

2.1 Neural Network Design

Feed forward, Jordan and Elman networks were examined, each with 0, 1 and 2 layers of lag variables (as used in ARIMA time series models [28]); and 0 and 1 hidden layers, except for the Elman networks which require at least 1 hidden layer. The neural networks were used to predict one of the four input parameters, with examples shown in Figure 1. All the networks were fully connected between layers. The lag layers were added as additional input nodes to the neural network (one lag layer would add four additional input nodes, and two lag layers would add eight). The first order lag variables (Δ) are the difference between the current and previous timestep, *e.g.*:

$$\Delta_t(Airspeed) = Airspeed_t - Airspeed_{t-1} \tag{1}$$

where t is the current timestep and $t-1$ is the previous timestep. The second order lag variables (Δ^2) are the difference between the current and previous first order lag variables, *e.g.*:

$$\Delta_t^2(Airspeed) = \Delta_t(Airspeed) - \Delta_{t-1}(Airspeed) \tag{2}$$

The following naming scheme was used to describe the various neural networks: *network_type/l(lag_layers)/h(hidden_layers)*. In the following scheme, *ff/l1/h1* would describe a feed forward network, with one lag layer and one hidden layer; while *jordan/l2/h0* would describe a Jordan network with two lag layers (the first order lag variables and the second order lag variables) and no hidden nodes.

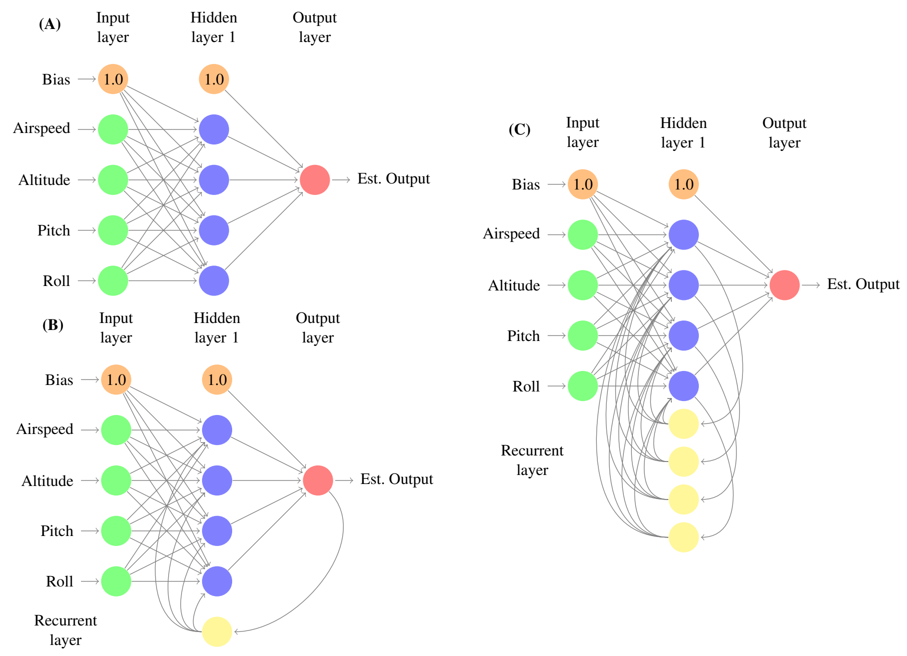

Fig. 1. Feed Forward (A), Jordan (B) and Elman (C) networks with a hidden layer and single output node. These networks were trained separately for each of the four possible outputs: airspeed, altitude, pitch and roll.

2.2 Objective Function

The neural networks were designed to predict the next second of flight data given the previous second of flight data and the first and second order lags variables, if used. To determine the optimal weights for the neural networks, the following objective function was used for both backpropagation and differential evolution:

$$f(w) = \frac{\sum_{t=0}^{n-1} |nn(I_t, \Delta_t, \Delta_t^2, w)_{output} - I_{t+1,output}|}{n-1} \tag{3}$$

Where $f(w)$ is the objective function evaluated over the weights w. With I_t being the input at timestep t, the absolute difference between the output predicted by the neural network, $nn(\ldots)_{output}$, and that value at the next time step, $I_{t+1,output}$, is calculated over every per second data reading $(0 \ldots n-1)$, given the input parameters, and input lags if used (I_t, Δ_t, Δ_t^2). This was then divided by the number of comparisons, $n-1$. This produces the mean absolute error (MAE) for the target output for the entire flight.

2.3 Neural Network Bounds and Data Cleansing

The flight data required some cleaning for use, as it is stored as raw data from the flight data recorders uploaded to the NGAFID server and entered in the database as per second data. When a FDR turns on, some of the sensors are still calibrating or not immediately online, so the first minute of flight data can have missing and erroneous values. These initial recordings were removed from the data the neural networks were trained on. Further, the parameters had wide ranges and different units, *e.g.*, pitch and roll were in degrees, altitude was in meters and airspeed was in knots. These were all normalized to values between 0 and 1, where 0 was the minimum value recorded and 1 was the maximum value recorded for each parameter to remove bias.

Additionally, the recurrent neural networks needed to be bounded. The flights consisted of between 5412 and 5941 per second recordings (over an hour and a half of per second data). This led to a problem where poor weights to the recurrent layer resulted in the fitness growing beyond the bounds of double precision. To alleviate this problem, the values for the recurrent nodes were bounded between -2 and 3 which eliminated overflow errors. Lastly, the weights for the neural networks were all bounded between -1.5 and 1.5 to limit the search space of the evolutionary algorithms and initial values for backpropagation.

3 Parallel Evolutionary Algorithms

A wide range of evolutionary algorithms have been examined for different distributed computing environments. Generally, these fall into three categories: single population (panmictic, coarse-grained); multi-population (island, medium-grained); or cellular (fine-grained); as classified by Cantu-Paz [6]. These various approachs have different effects on the explorative and exploitative properties of the evolutionary algorithms [26], with smaller subpopulations allowing faster exploitation of their search subspaces.

Given the scale of the data in the NGAFID, and the potential complexity of examining complex neural networks over per-second flight data, a package requiring easy use of high performance computing resources was required. While there exist some standardized evolutionary algorithms packages [2, 5, 27, 17], as well as those found in the R programming language [20, 3] and MATLAB [18], they do not easily lend themselves towards use in high performance computing environments.

This work utilizes the Toolkit for Asynchronous Optimization (TAO), which is used by the MilkyWay@Home volunteer computing to perform massively distributed evolutionary algorithms on tens of thousands of volunteered hosts [10, 11, 8]. It is implemented in C and MPI, allowing easy use on clusters and supercomputers, and also provides support for systems with multiple graphical processing units. Further, TAO has shown that performing evolutionary algorithms asynchronously can provide significant improvements to performance and scalability over iterative approaches [25, 9]. Its code is also open source and freely available on GitHub, allowing easy use and extensibility[3].

[3] https://github.com/travisdesell/tao

4 Results

4.1 Runtime Environment

All results were gathered using a Beowulf HPC cluster with 32 dual quad-core compute nodes (for a total of 256 processing cores). Each compute node has 64GBs of 1600MHz RAM, two mirrored RAID 146GB 15K RPM SAS drives, two quad-core E5-2643 Intel processors which operate at 3.3Ghz, and run the Red Hat Enterprise Linux (RHEL) 6.2 operating system. All 32 nodes within the cluster are linked by a private 56 gigabit (Gb) InfiniBand (IB) FDR 1-to-1 network. The code was compiled and run using MVAPICH2-x [13], to allow highly optimized use of this network infrastructure.

Each run of a differential evolution/neural network combination was submitted as a single job, allocating 32 processes across 4 nodes, with 1 master process to handle generating new individuals and updating the population, and 31 to handle the objective function calculations. The asynchronous differential evolution implementation from TAO was used to perform this optimization.

The results for optimizing all combinations of the neural networks and input parameters to differential evolution required 12000 jobs, approximately 4,800 hours of single CPU compute time. These jobs had a minimum runtime of 2.3 seconds, a maximum runtime of 278 seconds and an average runtime of 45.1 seconds. As such, utilizing parallel differential evolution on a HPC system enabled running these jobs in a reasonable amount of time, taking approximately a week given shared resources with other users of the cluster.

4.2 Evaluation Metrics

A random noise estimator (RNE), which uses the previous value as the prediction for the next value, $prediction(t_{i+1}) = t_i$, was chosen as a baseline comparison, as it represents the best predictive power that can be achieved for random time series data. If the neural networks did not improve on this, then the results would have been meaningless and potentially indicate that the data is too noisy (given weather and other conditions) for prediction. Additionally, it provides a good baseline in that it is easy for neural networks to represent the RNE: all weights can be set to 0, except for a single path from the path from the corresponding input node to the output node having weights of 1. Because of this, it also provides a good test of the correctness of the global optimization techniques, at the very least they should be able to train a network as effective as a RNE; however local optimization techniques (such as backpropagation) may not reach this if the search area is non-convex and the initial starting point does not lead to a good minimum.

4.3 Infeasibility of Backpropagation

Backpropagation was evaluated using both stochastic gradient descent (GD) and conjugate gradient descent (CGD) on flight 13588. Stochastic GD and CGD were run 20 different times for the networks described in Section 2. Figure 2 presents the range of fitnesses found for each backpropagation and network combination. In all cases, stochastic GD and CGD performed worse than RNE, demonstrating the challenging and non-convex nature of this search area. Accuracy further decreased and became more variable as neural networks became more complicated. Stochastic GD and CGD performed similarly for altitude, pitch, and roll.

Backpropagation was also evaluated using GD and CGD starting from a neural network which simulates a RNE, to address the case of all the initial starting positions of stochastic GD and CGD being poor. However, even the neural networks representing RNE for each network type were close to a local minima. Using backpropagation starting from a RNE, Airspeed was only improved upon at best by 0.018%, altitude improved on by 0.5%, pitch improved on by 0.025%, and roll improved upon by 0.022%.

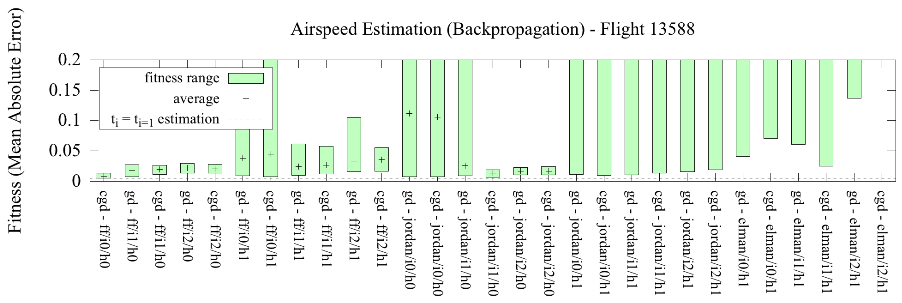

Fig. 2. Histograms for the estimation of airspeed for flight 13588 using backpropagation via gradient descent (GD) and conjugate gradient descent (CGD). The dashed line shows the baseline estimation for random noise, using the value at timestep t_i as the prediction for the value at timestep t_{i+1}.

4.4 Neural Network Optimization

Given previous experience training these neural networks, and in order to perform the analysis of more types of neural networks, we limited the differential evolution options to *de/best/3/bin* and *de/rand/3/bin* (differential evolution with best or random parent selection, 3 pairs, and binary recombiation – for a detailed descripton of differential evolution variants see [19]), with a population size of 500; as these settings had been shown to provide good convergence rates and effective solutions in the past.

Five flights (13588, 15438, 17269, 175755 and 24335) were used for analysis, and neural networks were trained to predict each of the four flight data parameters (airspeed, altitude, pitch and roll) used as input. Each differential evolution strategy and neural network combination was run 20 times, for a total of 12000 runs (2 DE strategies x 15 neural networks x 4 output parameters x 5 flights x 20 runs each). The parallel differential evolution was terminated after 15,000,000 total objective functions had been evaluated, or the best fitness in the population had not improved after 1,000 generations.

Table 1 show rankings for the different neural networks in terms of how well they predicted the given output. The ranks are what order the network came in, in terms of prediction; *i.e.*, a 1 means the neural network gave the best fitness for one of the flights, a 2 means it gave the second best fitness, and so on. Ties (in some cases multiple networks had the same best fitness) were given the same rank, so in some cases there are more than 5 of each rank. The rank column provides the average rank the network came in across all flights, and the avg. evals column gives the average number of evaluations it took for the differential evolution to terminate.

Airspeed

Network	Rank	Ranks	Avg. Evals
elman/i0/h1	19.9	10, 16, 17, 17, 19, 20, 23, 25, 26, 26	2667750
elman/i1/h1	14.0	4, 10, 11, 12, 13, 16, 18, 18, 19, 19	3334000
elman/i2/h1	6.7	1, 3, 3, 4, 5, 6, 7, 9, 14, 15	4225500
ff/i0/h0	22.6	21, 21, 22, 22, 23, 23, 23, 23, 24, 24	830000
ff/i0/h1	24.1	22, 23, 23, 24, 24, 24, 25, 25, 25, 26	1050000
ff/i1/h0	13.0	11, 11, 12, 12, 13, 13, 14, 14, 15, 15	1515000
ff/i1/h1	14.5	12, 13, 13, 14, 14, 15, 15, 16, 16, 17	1720000
ff/i2/h0	6.6	5, 5, 5, 6, 6, 6, 8, 8, 8, 9	2615000
ff/i2/h1	8.3	6, 7, 7, 7, 8, 8, 9, 10, 10, 11	2225000
jordan/i0/h0	19.8	19, 19, 19, 19, 20, 20, 20, 20, 21, 21	1573750
jordan/i0/h1	21.9	20, 20, 21, 21, 21, 22, 22, 22, 24, 26	2551750
jordan/i1/h0	5.0	2, 2, 3, 3, 3, 3, 5, 6, 11, 12	6897750
jordan/i1/h1	15.5	9, 13, 14, 15, 16, 17, 17, 18, 18, 18	3712000
jordan/i2/h0	1.6	1, 1, 1, 1, 1, 2, 2, 2, 2, 3	11695500
jordan/i2/h1	7.4	4, 4, 4, 5, 7, 8, 9, 10, 11, 12	5078750

Pitch

Network	Rank	Ranks	Avg. Evals
elman/i0/h1	13.0	6, 10, 10, 12, 12, 12, 12, 16, 20, 20	3543250
elman/i1/h1	11.6	1, 10, 10, 11, 11, 11, 13, 14, 16, 19	4104750
elman/i2/h1	6.2	1, 1, 2, 2, 3, 7, 8, 9, 11, 18	4793500
ff/i0/h0	23.8	23, 23, 23, 23, 24, 24, 24, 24, 25, 25	905000
ff/i0/h1	25.3	24, 24, 25, 25, 25, 25, 26, 26, 26, 27	1105000
ff/i1/h0	16.4	15, 15, 15, 15, 17, 17, 17, 17, 18, 18	1200000
ff/i1/h1	17.9	16, 16, 17, 17, 18, 18, 19, 19, 19, 20	1675000
ff/i2/h0	5.2	3, 3, 5, 5, 6, 6, 6, 6, 6, 6	2595000
ff/i2/h1	5.4	1, 2, 4, 5, 5, 7, 7, 7, 8, 8	2045000
jordan/i0/h0	20.7	20, 20, 20, 20, 21, 21, 21, 21, 21, 22	1372500
jordan/i0/h1	22.3	21, 21, 22, 22, 22, 22, 23, 23, 23, 24	2449750
jordan/i1/h0	12.5	9, 10, 12, 12, 13, 13, 13, 14, 14, 15	4874750
jordan/i1/h1	15.4	13, 13, 14, 14, 15, 15, 16, 17, 18, 19	3640000
jordan/i2/h0	3.1	1, 2, 2, 3, 3, 3, 4, 4, 4, 5	9692500
jordan/i2/h1	7.6	4, 5, 6, 7, 8, 8, 9, 9, 9, 11	5269750

Altitude

Network	Rank	Ranks	Avg. Evals
elman/i0/h1	22.4	20, 21, 21, 22, 22, 22, 23, 23, 24, 26	2629000
elman/i1/h1	16.6	11, 13, 13, 13, 16, 17, 17, 17, 24, 25	3704000
elman/i2/h1	16.2	12, 12, 13, 13, 15, 16, 18, 19, 19, 25	4583750
ff/i0/h0	20.4	18, 18, 19, 19, 20, 20, 21, 21, 24, 24	780000
ff/i0/h1	21.7	19, 19, 20, 20, 21, 22, 22, 23, 25, 26	1205000
ff/i1/h0	8.6	7, 7, 8, 8, 8, 8, 10, 10, 10, 10	1635000
ff/i1/h1	10.0	8, 9, 9, 9, 9, 10, 11, 11, 12, 12	1705000
ff/i2/h0	3.2	2, 2, 3, 3, 3, 3, 4, 4, 4, 4	2060000
ff/i2/h1	4.2	3, 3, 4, 4, 4, 4, 4, 5, 5, 6	2335000
jordan/i0/h0	17.1	15, 16, 16, 16, 16, 16, 17, 18, 20, 21	1153500
jordan/i0/h1	15.9	14, 14, 14, 15, 15, 15, 17, 18, 18, 19	2425500
jordan/i1/h0	6.6	6, 6, 6, 6, 6, 6, 7, 7, 8, 8	4860500
jordan/i1/h1	10.7	7, 9, 9, 10, 10, 11, 11, 12, 14, 14	4421750
jordan/i2/h0	1.3	1, 1, 1, 1, 1, 1, 1, 2, 2, 2	11706250
jordan/i2/h1	5.1	2, 3, 3, 5, 5, 5, 6, 7, 7, 8	6343750

Roll

Network	Rank	Ranks	Avg. Evals
elman/i0/h1	18.3	4, 16, 17, 17, 18, 19, 21, 23, 24, 24	3096000
elman/i1/h1	13.7	1, 2, 14, 15, 16, 16, 17, 18, 18, 20	3882500
elman/i2/h1	5.0	1, 1, 1, 1, 2, 2, 3, 9, 14, 16	4707750
ff/i0/h0	23.8	22, 22, 23, 23, 24, 24, 25, 25, 25, 25	710000
ff/i0/h1	25.3	23, 24, 24, 25, 25, 26, 26, 26, 27, 27	1090000
ff/i1/h0	11.6	8, 8, 11, 11, 11, 11, 13, 13, 15, 15	1435000
ff/i1/h1	13.1	9, 10, 12, 12, 13, 13, 14, 15, 16, 17	1655000
ff/i2/h0	5.4	4, 4, 4, 4, 5, 6, 6, 6, 7, 8	2355000
ff/i2/h1	7.2	5, 5, 6, 6, 7, 7, 8, 9, 9, 10	2375000
jordan/i0/h0	19.7	18, 18, 19, 19, 20, 20, 20, 21, 21, 21	1522500
jordan/i0/h1	21.5	19, 20, 21, 21, 22, 22, 22, 22, 23, 23	2387500
jordan/i1/h0	10.0	7, 7, 9, 10, 10, 10, 10, 10, 13, 14	3954750
jordan/i1/h1	14.5	11, 11, 12, 13, 14, 15, 15, 17, 18, 19	3740750
jordan/i2/h0	3.3	2, 2, 2, 3, 3, 3, 3, 4, 5, 6	6838000
jordan/i2/h1	7.7	3, 4, 5, 7, 7, 8, 8, 11, 12, 12	5292750

Table 1. Neural Network Rankings

These results show that there is no clear cut best neural network for prediction of all these flight parameters. However, the Jordan and Elman networks tend to outperform the feed forward neural networks; and in most cases adding input lags improves the predictive ability of the network. Except in the case of the Elman networks, adding a hidden layer does not seem to provide much benefit (perhaps due to the increased difficulty of optimization). Generally speaking, it seems that Jordan networks (with 2 input lags and no hidden layer) perform the best for predicting altitude and airspeed, while Elman networks (with 2 input lags and 1 hidden layer) perform the best for pitch and roll. Also worth noting is that when the Jordan networks perform the best, they take significantly longer to converge to a solution. Given this, it may be the case that the Elman networks are either converging prematurely or getting stuck. This question does beg further examination, as the more complicated neural networks should theoretically be able to provide more predictive power.

4.5 Cross Validation

In order to gauge the predictive power of the trained neural networks, the best found neural networks for each flight and output parameter were cross validated against the other flights. In Table 2, each row shows the best found neural network for each flight. The first row shows the random noise estimation (RNE), for baseline comparison. Each column in that row shows the mean absolute error (MAE) for the neural network trained for the flight specified flight against all the other flights. The bold values show the MAE where the input flight was the flight compared against; while italicized values show where the neural network performed worse than the RNE.

These results show that the trained neural networks have predictive parameters of other flights. They also show a dramatic difference in predictive ability for the different output parameters. Excluding the neural networks trained on flight 17269, predicted airspeed showed a 10-12% improvement over RNE, altitude showed near 70% improvement, while pitch and roll were much lower at 5-7% and 0.5-3%, respectively. Most of the trained neural networks were able to improve over RNE for all the other flights that they were not trained on. Further, the predictions are fairly accurate. As the input and output parameters were normalized between 0 and 1, the mean average error is also the average percentage error for the prediction. Airspeed predictions were around 0.6% error, altitude predictions were around 0.08% error, pitch was around 1.5% error, and roll was around 2% error.

These results lead to some interesting findings: first, the four parameters used (altitude, airspeed, pitch and roll) are probably not sufficient for prediction of pitch and roll, however they do provide good inputs for predicting airspeed and especially altitude. Using additional input parameters should allow better prediction for these values. Second, using this cross validation it appears that flight 17269 is an outlier, especially in pitch, as it was 50% worse than RNE in predicting pitch from the other flights. These findings present the possibility that it may be possible to determine atypical flights utilizing trained neural networks and potentially identify problematic parameters.

Airspeed

Method		13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$		0.00512158	0.00316859	0.00675531	0.00508229	0.00575537	0.0
13588	elman/i2/h1	**0.00472131**	0.00250284	0.00656991	0.00465581	0.00495454	10.78%
15438	jordan/i2/h0	0.00500836	**0.00218919**	0.0067222	0.00480868	0.00498588	10.47%
17269	jordan/i2/h0	*0.00513133*	0.0027844	**0.00620534**	0.00505878	0.00552826	4.90%
175755	jordan/i2/h0	0.0047884	0.00240848	0.00643301	**0.00459774**	0.00498664	11.63%
24335	jordan/i2/h0	0.00487011	0.00226412	0.00666179	0.00471104	**0.00485888**	11.54%

Altitude

Method		13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$		0.00138854	0.00107117	0.00200011	0.00137109	0.00192345	0.0
13588	jordan/i2/h0	**0.000367535**	0.000305193	0.000895711	0.000399587	0.000485329	69.18%
15438	jordan/i2/h0	0.000394097	**0.000263834**	0.000837357	0.0004203	0.00048358	69.87%
17269	jordan/i2/h0	0.000702832	0.000765161	**0.000801323**	0.000694245	0.000846411	48.65%
175755	jordan/i2/h0	0.00037486	0.0003003	0.000883877	**0.000390743**	0.00048446	69.42%
24335	jordan/i2/h0	0.000380966	0.000281196	0.000906039	0.000404582	**0.000468267**	69.43%

Pitch

Method		13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$		0.0153181	0.010955	0.0148046	0.0161251	0.0173269	0.0
13588	elman/i1/h1	**0.014918**	0.0100763	0.0147712	0.01514	0.0160249	4.90%
15438	jordan/i2/h0	*0.0163609*	**0.00881572**	*0.0159061*	0.0150275	0.015552	4.47%
17269	elman/i2/h1	*0.0199653*	*0.0249148*	**0.0142671**	*0.0199625*	*0.0291537*	-49.24%
175755	ff/i2/h1	*0.0153644*	0.00917981	*0.0148751*	**0.0145228**	0.0153566	7.35%
24335	elman/i2/h1	*0.0157302*	0.00911826	*0.0160291*	0.014868	**0.0149484**	5.47%

Roll

Method		13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$		0.0158853	0.00604479	0.0204441	0.012877	0.0192648	0.0
13588	elman/i2/h1	**0.0154541**	0.00587058	*0.0206536*	0.0127999	0.0182611	2.08%
15438	elman/i2/h1	*0.0164341*	**0.00544584**	*0.0217141*	*0.0129252*	0.0176981	1.60%
17269	elman/i1/h1	0.0157483	*0.00613587*	**0.0201234**	*0.0129124*	0.0184769	0.95%
175755	elman/i2/h1	0.0156503	0.00573676	*0.0205158*	**0.0125207**	0.017983	3.13%
24335	elman/i2/h1	*0.0163245*	0.00578885	*0.0215668*	*0.0131439*	**0.0174324**	0.68%

Table 2. Cross Validation for All Flight Parameters and Flights

5 Conclusions and Future Work

This work provides an extensive analysis of flight parameter estimation using various neural networks and input parameters to differential evolution. The neural networks were trained on data recorded during actual student flights, and consist of noisy, realistic general aviation flight data. Results show that while backpropagation is unable to provide much improvement over a random noise estimator (RNE), parallel differential evolution can provide strong predictions of airspeed (10% better than RNE) and altitude (70% better than RNE). These results were also fairly accurate, ranging between 0.08% accuracy for altitude to 2% accuracy for roll. Cross validation indicate that the trained neural networks have predictive ability, as well as the potential to act as overall descriptors of the flights. The trained neural networks could be used to detect anomalous flights, and even determine which flight parameters are causing the anomaly (*e.g.*, pitch in flight 17269).

For future work, how well the neural networks can be trained using particle swarm optimization [15] is of particular interest; as well as using autoencoders and other deep learning strategies [4], or hybrid strategies with genetic algorithms or ant colony optimization to evolve the structure of more complicated neural networks. Performing a grid search over potential evolutionary algorithm parameters is also suboptimal, which can be improved on by using hyperparameter optimization [14] or other metaheuristics. Further, training the neural networks over groups of flights could potentially improve their overall predictive ability as well as minimize overtraining.

The National General Aviation Flight Database (NGAFID) provides an excellent data source for researching evolutionary algorithms, machine learning and data mining. Further analysis of these flights along with more advanced prediction methods will enable more advanced flight sensors, which could prevent accidents and save lives; which is especially important in the field of general aviation as it is has the highest accident rates within civil aviation [21]. As many of these flights also contain per-second data of various engine parameters, using similar predictive methods it may become possible to detect engine and other hardware failures, aiding in the maintenance process. This work presents an initial step towards making general aviation safer through machine learning and evolutionary algorithms.

References

1. Aircraft Owners and Pilots Association (AOPA), January 2014.
2. M. Arenas, P. Collet, A. Eiben, M. Jelasity, J. Merelo, B. Paechter, M. Preuß, and M. Schoenauer. A framework for distributed evolutionary algorithms. *Parallel Problem Solving from Nature-PPSN VII*, pages 665–675, 2002.
3. T. Bartz-Beielstein. SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. *arXiv preprint arXiv:1006.4645*, 2010.
4. Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
5. S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.
6. E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.

7. S. F. Crone, M. Hibon, and K. Nikolopoulos. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635–660, 2011.
8. T. Desell. *Asynchronous Global Optimization for Massive Scale Computing*. PhD thesis, Rensselaer Polytechnic Institute, 2009.
9. T. Desell, D. Anderson, M. Magdon-Ismail, B. S. Heidi Newberg, and C. Varela. An analysis of massively distributed evolutionary algorithms. In *The 2010 IEEE congress on evolutionary computation (IEEE CEC 2010)*, Barcelona, Spain, July 2010.
10. T. Desell, B. Szymanski, and C. Varela. Asynchronous genetic search for scientific modeling on large-scale heterogeneous environments. In *17th International Heterogeneity in Computing Workshop*, Miami, Florida, April 2008.
11. T. Desell, C. Varela, and B. Szymanski. An asynchronous hybrid genetic-simplex search for modeling the Milky Way galaxy using volunteer computing. In *Genetic and Evolutionary Computation Conference (GECCO)*, Atlanta, Georgia, July 2008.
12. B. Elias. *Securing general aviation*. DIANE Publishing, 2009.
13. W. Huang, G. Santhanaraman, H.-W. Jin, Q. Gao, and D. K. Panda. Design of high performance MVAPICH2: MPI2 over InfiniBand. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 43–48. IEEE, 2006.
14. F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proc. of ICML-14*, 2014. To appear.
15. J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.
16. M. Khashei and M. Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11(2):2664–2675, 2011.
17. M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4j: a modular framework for metaheuristic optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1723–1730, New York, NY, USA, 2011. ACM.
18. MathWorks. Global optimization toolbox. Accessed Online: March 2013.
19. E. Mezura-Montes, J. Velazquez-Reyes, and C. C. A. Coello Coello. Modified differential evolution for constrained optimization. In *IEEE Congress on Evolutionary Computation 2006 (CEC2006)*, pages 25–32, Vancouver, BC, July 2006.
20. K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline. Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.
21. National Transportation Safety Board (NTSB), 2012.
22. D. Ömer Faruk. A hybrid neural network and arima model for water quality time series prediction. *Engineering Applications of Artificial Intelligence*, 23(4):586–594, 2010.
23. H.-P. Schwefel. *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.
24. K. I. Shetty. *Current and historical trends in general aviation in the United States*. PhD thesis, Massachusetts Institute of Technology Cambridge, MA 02139 USA, 2012.
25. B. Szymanski, T. Desell, and C. Varela. The effect of heterogeneity on asynchronous panmictic genetic search. In *Proc. of the Seventh International Conference on Parallel Processing and Applied Mathematics (PPAM'2007)*, LNCS, Gdansk, Poland, September 2007.
26. M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, July 2013.
27. S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás. Jclec: a java framework for evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 12(4):381–392, 2008.
28. W. W.-S. Wei. *Time series analysis*. Addison-Wesley Redwood City, California, 1994.
29. G. P. Zhang. Neural networks for time-series forecasting. In *Handbook of Natural Computing*, pages 461–477. Springer, 2012.