Applications of Artificial Intelligence Methods for Satellite Maneuver Detection and Maneuver Time Estimation

Nicholas Perovich
MIT Lincoln Laboratory
Zachary Folcik, Rafael Jaimes
MIT Lincoln Laboratory

ABSTRACT

An important component of Space Domain Awareness (SDA) is detecting and timestamping satellite maneuvers, which allow space operators to provide accurate and timely predictions about the location of objects in Earth-orbit. SDA is a major concern not only for the military, but also for the commercial sector, which has drastically increased the number of satellites and debris in space in recent years. Automated SDA software is somewhat accurate for detecting and timestamping satellite maneuvers. However, as the number of satellites in Earth-orbit increases each year, reported false maneuvers and inaccurate timestamping will increase alongside it, unless these methods are improved. The current methods require operators to sift through thousands of maneuver detections in order to identify false alarms, visually estimate maneuver times, and manually update orbital states. This barrage of potentially inaccurate data requires significant time and resources to navigate. Space operators must often choose smaller numbers of satellites to track, resulting in less effecting SDA. Earth-orbiting satellites play an increasingly large role in our everyday lives, from leisure such as TV and internet, to crucial national security assets such as GPS and intelligence. If SDA activities don't include satellite maneuver awareness, services such as collision avoidance could be compromised. This has the potential to place essential resources in jeopardy.

In this study we test a variety of artificial intelligence (AI) tools on both simulated and real satellite residual data with the goal of improving detection rates, decreasing false alarm rates, and more accurately timestamping maneuvers. This will reduce operator workload and subsequent maneuver estimation costs in turn. This study takes advantage of the robust library of publicly-available AI tools, as well as the collegial relationship between MIT Lincoln Laboratory and satellite tracking data providers, such as the 18th Space Defense Squadron (18 SDS). To develop our methods, we use a confluence of real and simulated residual data for satellites in both Low Earth Orbit (LEO) and Geosynchronous Earth Orbit (GEO), gathered from both optical and radar sensors.

In our previous work, we developed two maneuver detection algorithms using a Random Forest (RF) classifier algorithm and a deep neural network (DNN) classifier [1]. We tested these models with simulated GEO data generated by optical sensors, achieving detection rates and false alarm rates far better than current statistical methods. In this work, we rigorously optimize the DNN model to further improve its performance. We also introduce the DNN to real GEO optical tracking data provided by 18 SDS to determine performance in an operational environment. We then simulate radar observation data for objects in LEO, evaluating RF and DNN on these novel measurements. Finally, we develop two maneuver timestamping algorithms using a Long Short-Term Memory (LSTM) neural network and a Transformer model.

This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

1. INTRODUCTION

The 18th Space Defense Squadron (18 SDS) maintains a catalog of all active satellites in Earth-orbit with a unique Satellite Catalog (SATCAT) number for each object to maintain Space Domain Awareness (SDA) continuity for the United States Space Command (USSPACECOM). To accomplish this mission, 18 SDS assumes command and control of the U.S. Space Surveillance Network (SSN), a global array of ground- and space-based optical and radar sensors that track objects in Earth-orbit [1]. 18 SDS employs around 115 military and civilian personnel, which we refer to as space operators [2].

One of the most challenging aspects of predicting future locations is the fact that satellites maneuver. Because satellite operators don't publicize their maneuver schedule, space operators must manually update a satellite's orbital state every time it maneuvers. This involves examining observational data to determine if a satellite maneuvered, applying algorithms to determine the magnitude of the maneuver, then manually updating the orbital state machine so it continues to make accurate location predictions. With only 115 operators to process reams of observational data for 16,000 active maneuvering satellites (not to mention the other 43,800 known debris objects), 18 SDS is susceptible to a phenomenon we refer to as operator Overload [1, 2]. We define operator Overload as an excess of time-sensitive and/or critical information that must be acted upon in a timely manner. Each operator, tasked with a stream of data for dozens of satellites, must logically cull down the information to a practical subset. The effect of operator Overload is to ignore most maneuvers except for known objects in specific areas of concern [3]. Operator Overload thus results in missed maneuvers and reduced SDA, decreasing national security and increasing the risk for collisions of valuable defense assets [4].

One of the most significant difficulties for detecting maneuvers is noise from optical and radar observations. Maneuver detection algorithms can interpret noise as maneuvers, triggering a false alarm, or algorithms too sensitive to outliers can miss maneuvers altogether, resulting in a lower detection rate. The ability of an algorithm to differentiate between observation noise and maneuvers is critical in evaluating its performance. Current algorithms are based on statistical methods, applying methods such as non-Gaussian pattern identification, thresholds, and filters to residual data. One such algorithm, developed in 2007 by Folcik et al., is able to correctly detect a maneuver 94% of the time (detection rate), but also detects a maneuver when no maneuver occurred 8% of the time (false alarm rate), when applied to real optical data for Geosynchronous Earth Orbit (GEO) objects [5]. Unfortunately, there is no such comparison metric for radar observation residuals gathered from Low Earth Orbit (LEO) objects. After maneuver detection, a subsequent task is to determine when the maneuver was initiated so that orbital state estimation systems can update new orbital state vectors. The same method mentioned above uses a threshold test to delineate the time of a maneuver. This method can produce a result accurate to within 24-48 hours of the beginning of a maneuver detected from a GEO object with optical observations [5].

1.1 Current Statistical Methods

Software tools allow operators to maintain SDA by estimating and predicting satellite orbital trajectories. The software compares predicted locations to observed values gathered from optical and radar sensors. Differences, or residuals, between predictions and observations are useful to determine if satellites deviate from their expected trajectories. Residuals are often the basis for maneuver detection algorithms. However, residuals combine the error sources of maneuvering motion and sensor observational noise. Fig. 1 shows a simple plot of residuals over time, with observational noise present that could deceive a threshold detection algorithm.

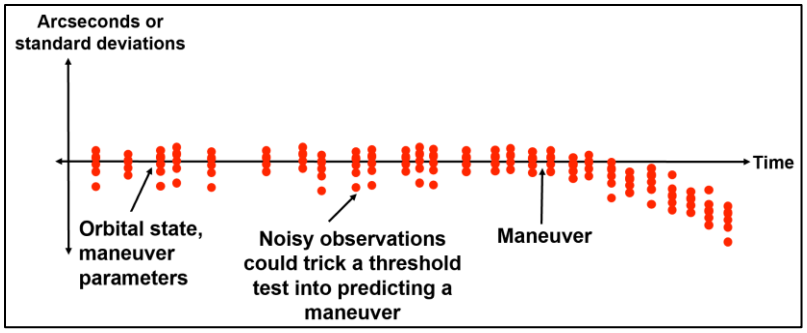

Fig. 1. Example Plot of Residuals over Time

Optical sensors provide observations in the form of Earth-centered inertial right ascension and declination. Radar sensors provide slant range, topocentric azimuth and elevation, and Doppler or range-rate. SDA methods for maneuver detection feed optical and radar observations to orbital estimate state machines, which use Kalman Filters sometimes combined with Weighted Least Squares. The state machine produces an accurate set of orbital state estimates with covariance, after which Kalman filters process incoming observations. While the maneuver detection method processes these observations, the Kalman Filter update step is bypassed to favor producing residuals. Residuals are normalized by the errors in observations and predicted state estimate. Normalized residuals are compared to a threshold in standard deviations (typically three or four). When a certain number of observations surpasses this threshold, a maneuver is declared by the algorithm [3, 5, 6].

These maneuver detections enable additional SDA information to be gathered – for example, an estimate for when the maneuver started. This additional product is derived using an algorithm that is also based on Kalman Filter or Weighted Least Squares methods. When the Kalman Filter calculates normalized residuals based on the current orbit estimate and the incoming observations, a threshold test is used to determine whether a maneuver is likely given the calculated observation residuals. This threshold test maintains a buffer containing the most recent residuals with observation times. The size of the buffer is configurable. As the threshold test is executed, the median normalized residual is calculated using the residuals in the buffer. If the median normalized residual value exceeds a configured threshold, typically four standard deviations, a maneuver is determined to have occurred. The date and time of the observation associated with the median residual is then assigned to the detected maneuver.

1.2 Detection Rate and False Alarm Rate

Maneuver detection algorithm performance is evaluated using two metrics called the *detection rate* and the *false alarm rate*. These are computed from true positives (TP), false positives (FP), and false negatives (FN). TP counts correctly identified maneuvers, while FP counts incorrectly identified maneuvers. FN is incorrectly identified non-maneuvers.

$$detection\ rate = \frac{TP}{TP+FN} \tag{1}$$

$$false\ alarm\ rate = \frac{FP}{FP+TP} \tag{2}$$

1.3 Objectives of this Study

It is the primary objective of this study to determine whether or not artificial intelligence (AI) methods can improve the false alarm rate and detection rate of maneuver detection software. A secondary goal is to determine if AI methods can be used to improve maneuver timestamping accuracy. In our previous work, we developed a Random Forest (RF) machine learning classification algorithm and a deep neural network (DNN) classifier to detect maneuvers based on simulated optical residual data from GEO objects. Our results are summarized in Table 1 [3].

Table 1: Summary of Results from the Initial Study

Method	Detection Rate	False Alarm Rate
Statistical	94%	8%
Machine Learning	91%	1%
Neural Network	94%	0.5%

In this follow-on study, we start with our previous model as a framework, while introducing new data, evaluation tools, and more robust algorithms. In this new study, we accomplish the following:

a) Optimize our DNN model to achieve better results than the ones presented in Table 1.
b) Introduce our models to real GEO optical observational data to evaluate performance realistically.
c) Evaluate DNN performance against simulated radar observations of LEO satellites.
d) Create and evaluate two maneuver timestamping algorithms: a Long Short-Term Memory (LSTM) neural network and a Transformer.

2. METHODS

2.1 Random Forest (RF)

The machine learning algorithm called Random Forest is an ensemble expansion of standard decision trees, which can be trained to predict a target value by learning a set of decision rules from features and corresponding targets in a dataset. This process is also known as supervised learning. RF randomizes a range of decision trees through bootstrap aggregation (i.e. bagging) to improve the overall stability and accuracy of the prediction. RF is a popular model for both regression and classification problems. Since it averages the result over multiple decision trees, RF reduces overfitting issues common in single decision trees. RFs have risen in prominence among other machine learning techniques for their relatively high accuracy with small datasets and their use of computational parallelization (multi-core CUPs). Before extensive work was conducted on developing the RF model for this study, its performance was compared against other machine learning classification algorithms: Stochastic Gradient, Logistic Regression, K-Nearest Neighbor, Gaussian Naïve Bayes, Perceptron, and Linear Support Vector Classification [3, 7]. RF significantly outperformed all these algorithms. The Python machine learning library Scikit-Learn is used to implement RF algorithms [7].

2.2 Deep Neural Network (DNN)

Neural networks are a newer, more advanced AI tool than machine learning algorithms. They generally provide higher accuracy but also require greater computational costs. A deep neural network analyzes data in a manner analogous to the human brain, and can be used for binary classification, multi-label classification, and regression problems. The fundamental DNN architecture contains multiple layers of neurons to process data before an output layer provides target predictions (Fig. 2).

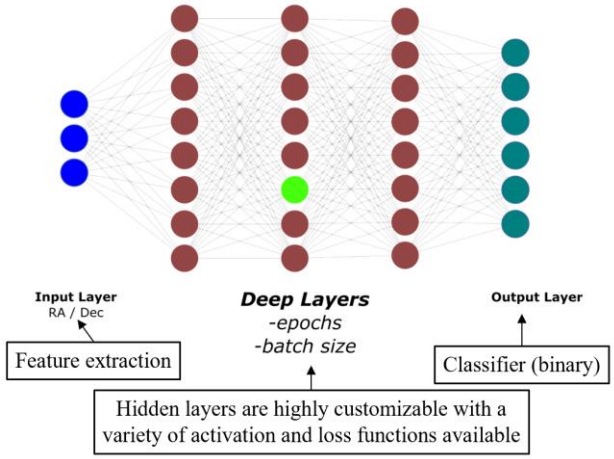

Fig. 2a. DNN Basic Architecture [8]

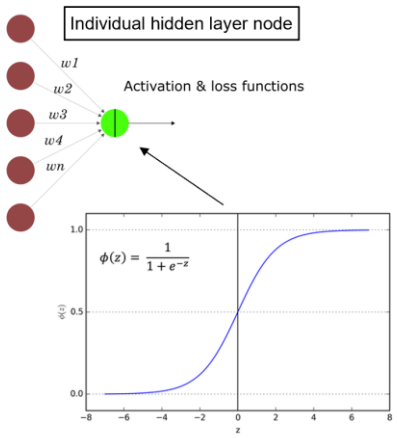

Fig. 2b. Individual Node Architecture [8]

The network's basic structure contains an input layer, one or more hidden layers, and an output layer. The number and size of hidden layers is specified by the user and can significantly affect the performance of the model. Neurons in the hidden and output layers are programmed with an activation function, which indicates whether the node is enabled or disabled, and loss functions, which calculate the accumulated error. We develop the DNN using Keras, a Python open-source library that allows users to interface with artificial neural networks in the TensorFlow library [1, 9].

2.2.1 Optimization

In the previous work [3], we only optimized the training time (i.e. number of training epochs) for the network. In this new study, we use a grid search method to optimize more of the hyperparameters. We employed a Scikit-Learn tool called Grid Search Cross-Validation (GSCV), and ran the optimizer on the MIT Lincoln Laboratory TX-Green supercomputer [7]. GSCV creates a 2D grid containing all combinations of specified hyperparameters. GSCV is computationally expensive even running on a supercomputer, so before running GSCV we narrowed down our hyperparameter options to those in Table 2.

Table 2. Hyperparameter Grid Space for DNN Optimization

Hyperparameter	**Dimensions / Sizes / Algorithms**					
# Hidden Layers	1	2	3	4	5	6
Size of Hidden Layers	1000, 800, 600	500, 300, 0	250, 150, 0	150, 75, 0	75, 37, 0	18, 0
Batch Size	100	200	300	–	–	–
Dropout Rate	0.0	0.25	0.5	–	–	–
Learning Rate	0.001	0.01	0.1	–	–	–
Epochs	50	100	150	200	–	–
Optimizer	Root Mean Square (RMS) Propagation	Adam	Adaptive Delta (Adadelta)	Nesterov Momentum Adam (NAdam)	–	–

The activation function and loss function were optimized trough trial-and-error. We chose functions that are generally regarded as the best for binary classification problems: *sigmoid* activation function and *binary cross entropy* loss function [3].

2.3 Maneuver Timestamping Models

To timestamp the maneuvers, we use different types of neural networks that look for patterns in the raw observation data. We use a long short-term memory (LSTM) network to develop a regression model and a multi-label classification model. We use a Transformer to build a second regression model. We compare the performance of these three networks against a baseline prediction.

2.3.1 Long Short-Term Memory (LSTM)

The LSTM network is an extension of the recurrent neural network (RNN) that addresses some of the inherent issues in RNNs, namely their difficulty handling problems with long time dependencies. Unlike DNNs, RNNs implement memory caches during computation. The product of one neuron impacts the subsequent neuron's input computation, so while DNNs treat inputs independently, RNNs link neighboring inputs. However, data with long timespans tends to cause the backpropagated error in RNNs to either increase or decrease exponentially. This is known as gradient explosion/vanishing. LSTMs remedy this by applying back-propagation between batches, thus minimizing the total loss in the network [3, 10].

Recurrently connected blocks in LSTM hidden layers contain multiple recurrent memory cells and three multiplicative units: input, output, and forget gates. Cell inputs are scaled by the activation function in the input gate, and the output to the rest of the network is scaled by the output gate. The forget gate is activated in the subsequent memory cell to scale the incoming products. Feature weights in each layer are also adjusted accordingly based on the propagated error [11, 12].

2.3.2 Transformer

The Transformer is a competing model to the LSTM, and has been shown to provide better accuracy and speed for natural language processing [13]. Transformer models address a fundamental limitation of RNN/LSTM models, namely that they do not process data sequentially. Transformer introduces the concept of "attention" in which data is encoded, attention is determined in a latent space, and then decoded. This approach allows data to be processed in parallel, decreasing computation time compared to RNN/LSTM. Attention also allows the model to learn which aspects of the data are most important to determine its meaning [14].

2.4 Real and Simulated Data

To train and evaluate our AI models, we require satellite residual data for maneuvering and non-maneuvering cases. Most of our initial data was generated with Monte-Carlo simulations designed to produce residuals that closely resemble those for real observations for optical and radar sensors. In this new study, however, we acquired real optical data from 18 SDS to test our models in an operational environment. We curated the 18 SDS observations so that we identified and solved for maneuvers in the maneuver cases. The dataset sizes and types are summarized in Table 3.

Table 3. Types and Sizes of Residual Datasets

Type of Data	Sim / Real	Source	# Maneuver Cases	# Non-Maneuver Cases
Optical	Simulated	Lincoln Laboratory Simulators	1000	1000
Radar	Simulated	Lincoln Laboratory Simulators	1000	1000
Optical	Real	18 SDS	100	100

Each "case" contains orbital element residual data for a 72-hour timespan. The maneuver cases contain observation residuals before, during, and sometimes after a maneuver. The non-maneuver cases contain observation residuals from steady-state orbits with sensor noise. For both simulated and real cases, the maneuver start time, magnitude, and duration are variable along the timespan. Solar exclusion timing introduces gaps in the real optical data; this is also

modelled and randomized in the simulated optical data. Since radar does not suffer from solar exclusion timing gaps, this was not simulated in these cases. Each case contains several orbital measurements, including Modified Julian Date, days from start of simulation, normalized azimuth residual, normalized elevation residual, normalized range residual, normalized range-rate residual, normalized right ascension residual, normalized declination residual, raw azimuth residual, raw elevation residual, raw range residual, raw range-rate residual, raw right ascension residual, raw declination residual, time of the residual, beta angle residual, height residual, time variance, beta angle variance, and height variance. The expected observation error is computed from sensor performance compared against satellites for which precise orbits are known. State error is computed from the orbital state covariance matrix. For more detailed information on how the simulated data was generated, please refer to our previous paper (Perovich et al.) [3].

The relevant residuals from optical sensor measurements that can be used to indicate maneuvers are raw and normalized measurements for right ascension (RA) and declination (DEC). Relevant radar sensor measurements are raw and normalized azimuth (AZ), elevation (EL), range (Ra), and range-rate (RR). Below we show plots of these 12 residuals for maneuvering and non-maneuvering cases. Normalized measurements are always in standard deviations (σ), while raw measurements vary:

Table 4. Raw Measurement Units

RA	DEC	AZ	EL	Ra	RR
arcseconds (asec)	arcseconds	arcseconds	arcseconds	meters	cm/sec

2.4.1 Plots of Simulated Residuals

2.4.1.1 Simulated Optical Residuals

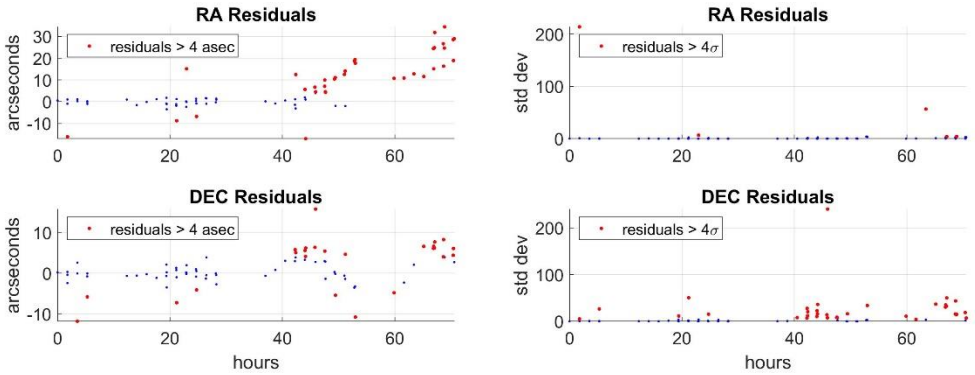

Fig. 3. Maneuver Case RA and DEC (Raw/Normalized)

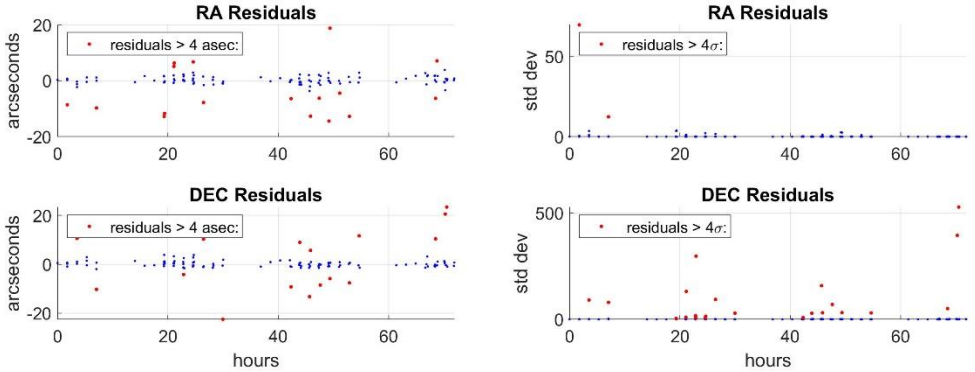

Fig. 4. Non-Maneuver Case RA and DEC (Raw/Normalized)

2.4.1.2 Simulated Radar Residuals

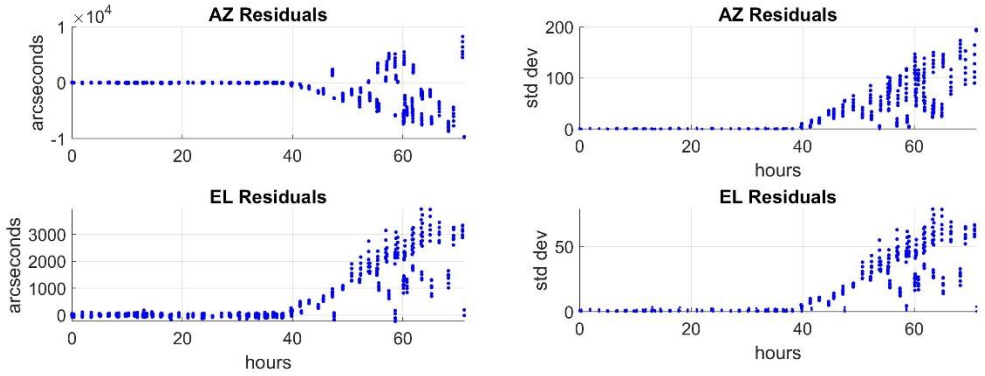

Fig. 5. Maneuver Case AZ and EL (Raw/Normalized)

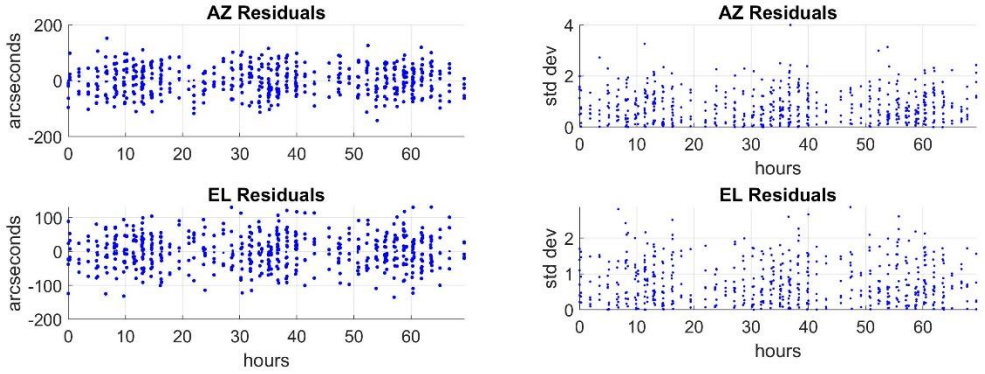

Fig. 6. Non-Maneuver Case AZ and EL (Raw/Normalized)

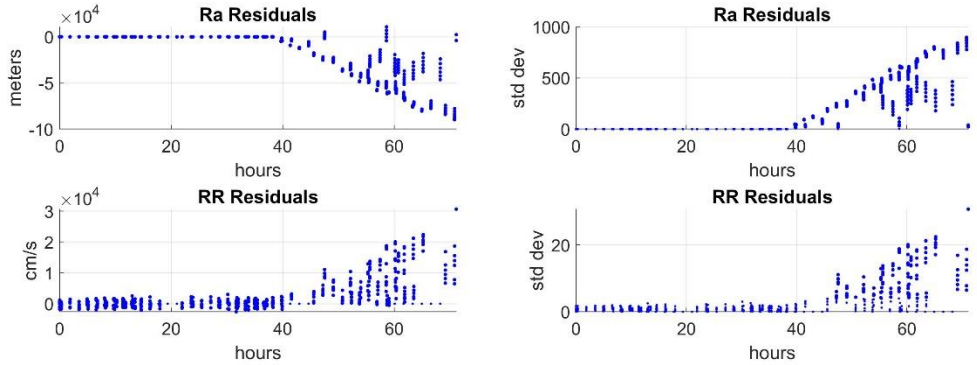

Fig. 7. Maneuver Case Ra and RR (Raw/Normalized)

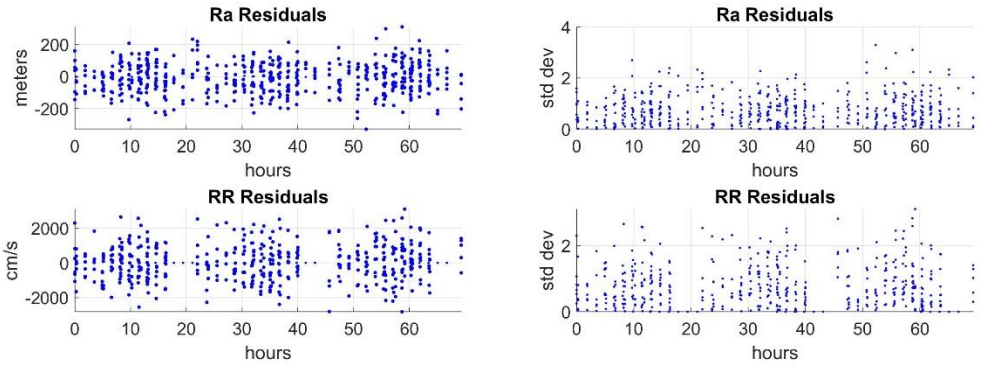

Fig. 8. Non-Maneuver Case Ra and RR (Raw/Normalized)

2.4.2 Plots of Real Residuals

Real optical data was provided by 18 SDS and curated by trained analysts at Lincoln Laboratory.

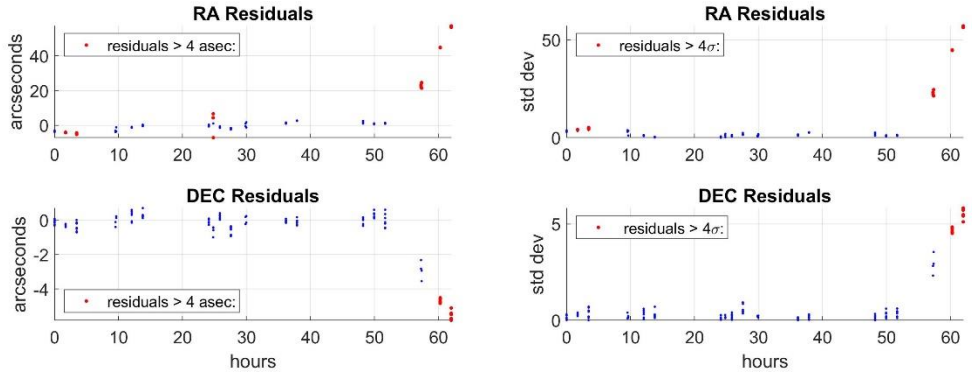

Fig. 9. Maneuver Real Case RA and DEC (Raw/Normalized)

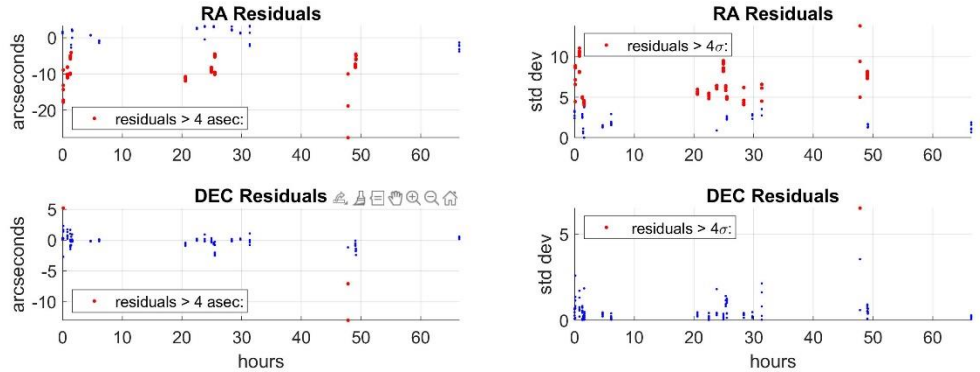

Fig. 10. Non-Maneuver Real Case RA and DEC (Raw/Normalized)

2.4.3 Plots of *Challenging* Real Residuals

Gaps in observations combined with variations in maneuver start times and magnitudes produce certain cases that are challenging for classifiers to identify as maneuvers. This is particularly noticeable for maneuvers of small magnitudes that start close to the end of the timespan. In these cases, there are only a small number of observations from which to detect the maneuver. Adding to this challenge are non-maneuver cases with large outliers close to the end of the

timespan. These two types of cases are virtually indistinguishable, even for an operator observing them. Examples of such cases from real optical observations from 18 SDS are below.

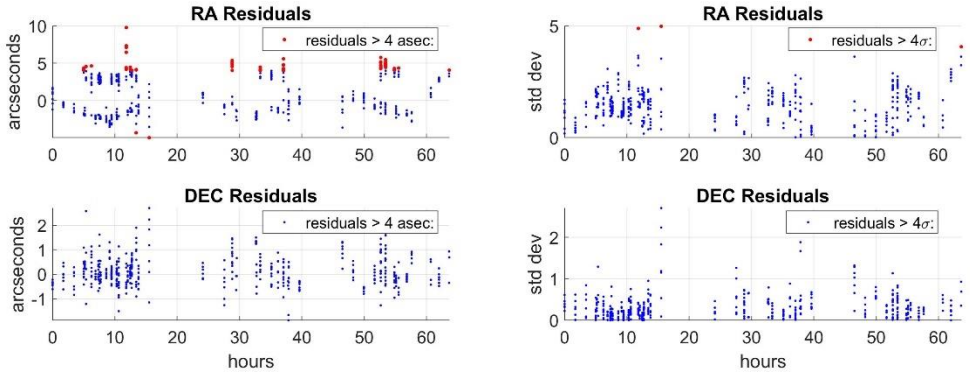

Fig. 11. *Challenging* Maneuver Real Case RA and DEC (Raw/Normalized)

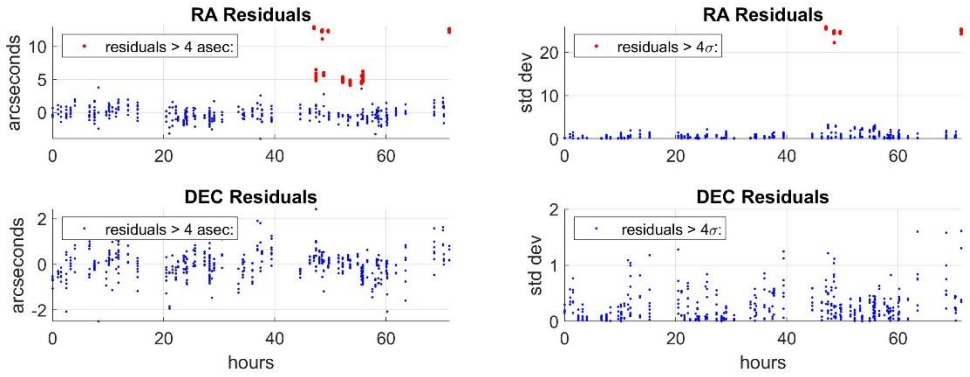

Fig. 12. *Challenging* Non-Maneuver Real Case RA and DEC (Raw/Normalized)

The only indications of a maneuver in Fig. 11 are the two or three larger residuals at the end of the timespan. Even so, their magnitude is half as large as some of the outliers earlier in the timespan. In Fig. 12 (non-maneuver), the large outliers at the end of the timespan look very similar to a late maneuver case. Even the best detection methods would have difficulty distinguishing these two apart, as did our DNN model.

2.5 Feature selection

RF and DNN require features from the data to be selected and computed before training the model. Features are quantifiable and relevant properties of the data that help the model determine patterns in the data. A complete feature collection should thoroughly explain every detail of the data applicable to the problem. Feature selection is subjective; it is based on the creator's estimate about what the classifier will find advantageous. For this study, 10 unique features were selected for each orbital measurement across the entire observation timespan. Thus, for optical data (RA + DEC) we computed 20 unique features for each case. For radar data (AZ + EL + Ra + RR) we computed 40 unique features for each case [3].

The first four features were selected based on averages and sums across the entire timespan:

1. Average magnitude of all residuals
2. Average time between residuals of a magnitude greater than four arcseconds
3. Total number of residuals with a magnitude greater than four arcseconds
4. Sum of all residuals with magnitudes greater than four arcseconds

The next six features were selected based on a sliding window average, which averages a specified number of consecutive residuals, shifting to the right with each step until all residuals are captured in at least one window. This tool helps to focus more computational power towards clustered averages, which tend to help identify difficult-to-detect maneuvers at the end of the timespan. They also help to smooth out the data by reducing the impact of observational noise. We chose to compute these in both raw and normalized measurements to further orient the algorithm to the difficult cases [3].

5. Maximum sliding window average (asec)
6. Maximum sliding window average (σ)
7. Difference between the window average at the initial timestep and maximum timestep (asec)
8. Difference between the window average at the initial timestep and maximum timestep (σ)
9. Difference between window average at the maximum timestep and average of the rest of the residuals (asec)
10. Difference between window average at the maximum timestep and average of the rest of the residuals (σ)

The sliding window tool is illustrated in Fig. 13. Each window captures a span of the data rather than simply looking at individual residuals. If outliers are captured in the windows, they get averaged and smoothed relative to the rest of the window. This has the effect of reducing the impact of noise on the algorithm [3].

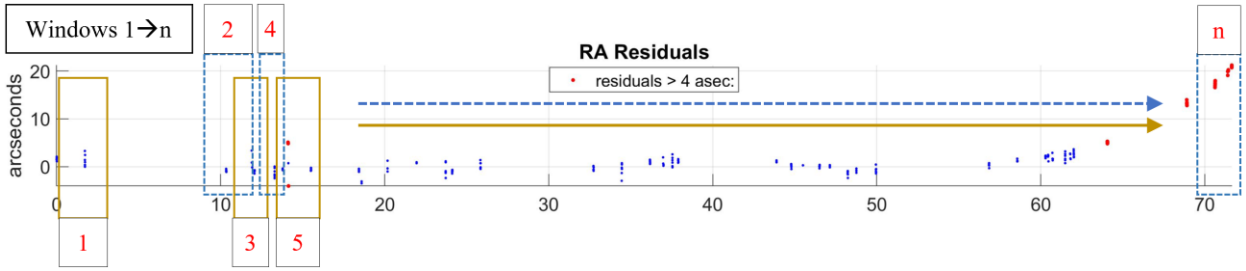

Fig. 13. Sliding Window Illustration

3. RESULTS AND DISCUSSION

For the purposes of training and validating the algorithm, residual data is collected in two sets: cases in which the satellite did and did not maneuver. Optical data contains raw and normalized RA and DEC residuals, and radar data contains raw and normalized AZ, EL, Ra, and RR residuals. Units for raw residuals are listed in Table 4. The data is split into training, validation, and testing batches. For each case, 20 features are computed for optical data, and 40 features for radar data.

DNN optimization was completed using simulated optical data. Table 1 shows the previous study's results with a non-optimized DNN model (except for epochs). Table 5 shows the previous hyperparameters used and the new GSCV-optimized hyperparameters.

Table 5. GSCV Optimization Results

Hyperparameter	**Non-Optimized** [3]	**Optimized**
Number of Hidden Layers	7 (1 input, 5 hidden, 1 output)	7 (1 input, 5 hidden, 1 output)
Size of Hidden Layers (# neurons)	1000, 500, 250, 125, 60, 30, 1	800, 300, 150, 75, 35, 18, 1
Batch Size	100	300
Dropout Rate	0.0	0.25
Learning Rate	Default (0.001)	0.001
Epochs	50	100
Optimizer	Adam	Adam

The main changes were the implementation of a dropout, smaller hidden layers, a larger number of epochs, and a larger batch size. Hyperparameters that remained the same and were not included in the GSCV sweep were the *sigmoid* activation function and the *binary crossentropy* loss function. After running the optimized model, it became clear that optimization provided a noticeable increase in performance on simulated optical data. Table 6 shows a comparison of the results from all three AI models on simulated optical data.

Table 6. AI Model Performance Comparison with Simulated *Optical* Residuals

Method	Detection Rate	False Alarm Rate
Random Forest [3]	90.95%	1.09%
Non-Optimized DNN [3]	93.62%	0.472%
Optimized DNN	96.67%	0.05%

An analysis tool Scikit-learn provides is the Area Under the Receiver Operating Characteristic Curve (AUROC) [7]. AUROC shows the performance of the algorithm in terms of true positives and false positives. An AUROC of 1.0 represents perfect separability between true and false positives, while an AUROC of 0.5 is the performance of a random classifier (plotted with a red line). Performance values near 1.0 indicate the algorithm is performing well. The ROC curves in Fig. 14 show that the optimized DNN has better separability than RF.

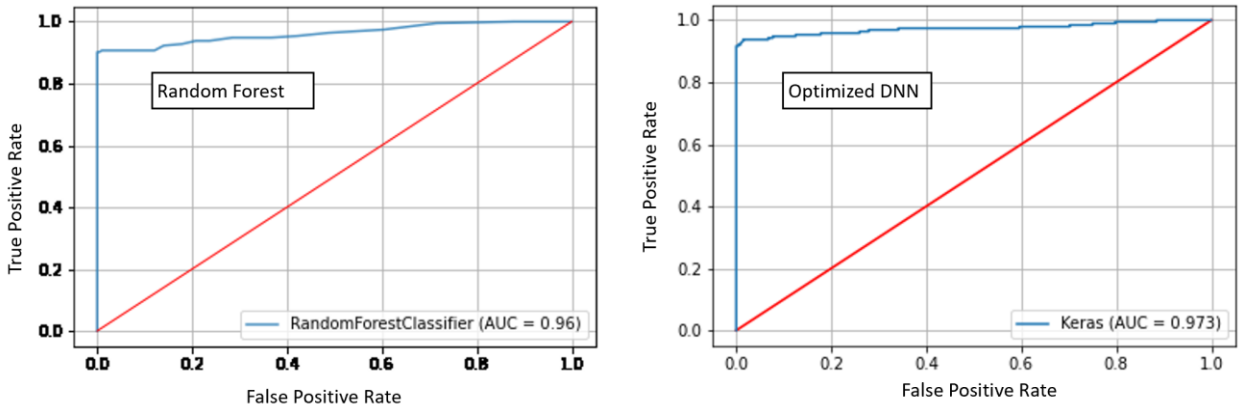

Fig. 14. ROC Curves for RF and Optimized DNN

3.1 Results with Simulated Radar Residuals

When we introduced RF and DNN to the simulated radar data, similar performance was achieved. No metrics were found in the open literature with which to compare these results, however these results are consistent with the results against optical data.

Table 7. AI Model Performance Comparison with Simulated *Radar* Residuals

Method	Detection Rate	False Alarm Rate
Random Forest	98.10%	0.10%
Optimized DNN	96.76%	0.05%

DNN achieves a lower false alarm rate but also a lower detection rate than RF, indicating that perhaps a combination of the two algorithms would produce an optimal maneuver detection algorithm for LEO objects. This hypothesis remains untested and is an item for future work.

3.1.1 RF Analysis

The Python Scikit-learn Random Forest classifier [7] provides tools to examine the importance of various features in the RF model by revealing the relative weights the model weights places on each feature. Abbreviations are given to each feature to identify them later.

Table 8. Features, and their Abbreviations, Used to Train RF and DNN

Feature Name	Feature Reference Abbreviation
Average magnitude of residuals	MAG
Average time between residuals > 4 asec	TIME
Number of residuals > 4 asec	NUM
Sum of magnitudes of all residual > 4 asec	SUM
Max of sliding window average for residual (asec)	WINDmax-AS
Max of sliding window average for residual (σ)	WINDmax-SD
Difference between window average at t_{min} and t_{max} for residuals (asec)	WINDdiff-AS
Difference between window average at t_{min} and t_{max} for residuals (σ)	WINDdiff-SD
Difference between avg residual magnitude at t_{max} and rest of residuals (asec)	TMAXdiff-AS
Difference between avg residual magnitude at t_{max} and rest of residuals (σ)	TMAXdiff-SD

All features were calculated for all six measurements: right ascension, declination, azimuth, elevation, range, and range rate. The abbreviation (RA, DEC, AZ, EL, Ra, RR) was added at the end of each name to differentiate each measurement. The importance of each feature is presented in the table below, ranked from 1 to 40.

Table 9. Summary of Feature Importance for RF

Importance	Feature Reference Abbreviation	Importance	Feature Reference Abbreviation
1	TMAXdiff-SD-Ra	21	WINDmax-AS-RR
2	WINDdiff-SD-Ra	22	SUM-RR
3	WINDmax-SD-Ra	23	WINDmax-SD-RR
4	TMAXdiff-AS-Ra	24	SUM-RR
5	WINDdiff-AS-Ra	25	TMAXdiff-SD-RR
6	MAG-Ra	26	MAG-EL
7	WINDmax-AS-Ra	27	WINDdiff-AS-RR
8	TMAXdiff-SD-AZ	28	WINDdiff-SD-RR
9	SUM-Ra	29	MAG-AZ
10	WINDdiff-AS-AZ	30	TMAXdiff-AS-RR
11	WINDmax-SD-AZ	31	TIME-RR
12	WINDdiff-SD-EL	32	NUM-AZ
13	TMAXdiff-SD-EL	33	MAG-RR
14	WINDdiff-SD-AZ	34	TIME-EL
15	TMAXdiff-AS-EL	35	NUM-EL
16	TMAXdiff-AS-AZ	36	NUM-Ra
17	WINDmax-SD-EL	37	TIME-AZ
18	WINDmax-AS-EL	38	TIME-Ra
19	WINDmax-AS-AZ	39	NUM-RR
20	WINDdiff-AS-EL	40	SUM-AZ

It appears that the range residual is the most important when detecting LEO maneuvers with radar sensors, as it appears eight times in the top ten most important features. The sliding window tool is also quite effective on radar data. It composes the overwhelming majority (90%) of the top half of the features. It also seems that standard deviation (normalized) data is slightly more important than raw data. The ROC curve (Fig. 15) shows that the RF algorithm is performing exceptionally well classifying maneuvers of LEO satellites.

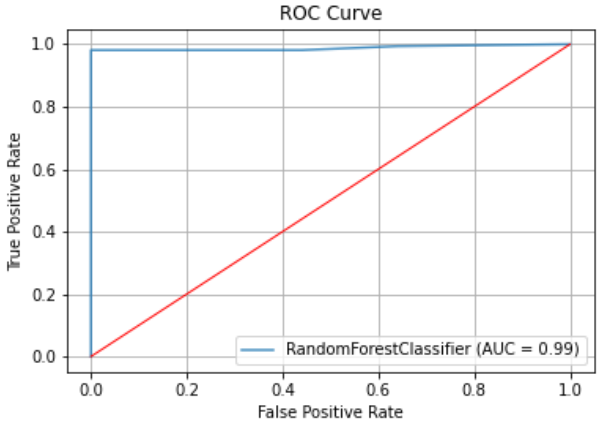

Fig. 15. ROC Curve for RF Model with Radar Data

To optimize the RF algorithm, we consider the number of "epochs" (number of trees used to solve before the model converges). Optimal RF performance can still be achieved while leaving the remainder of the hyperparameters as default [1].

Four metrics native to Scikit-learn's RF classifier are used to optimize epochs: validation accuracy, precision (detection rate), recall (false alarm rate), and AUROC [7]. These metrics are evaluated from 0 to 300 epochs and weighted as follows: precision: 30%, recall: 30%, accuracy: 20%, AUROC: 20%. While training accuracy was not used for optimization, it is useful to ensure the model is not overfitting. An average metric is computed and an optimum number of epochs is chosen. The red dot shows the optimum performance for the shortest computation time (250 epochs). The results from this analysis are shown in Fig. 16. RF achieves maximum performance at 250 epochs.

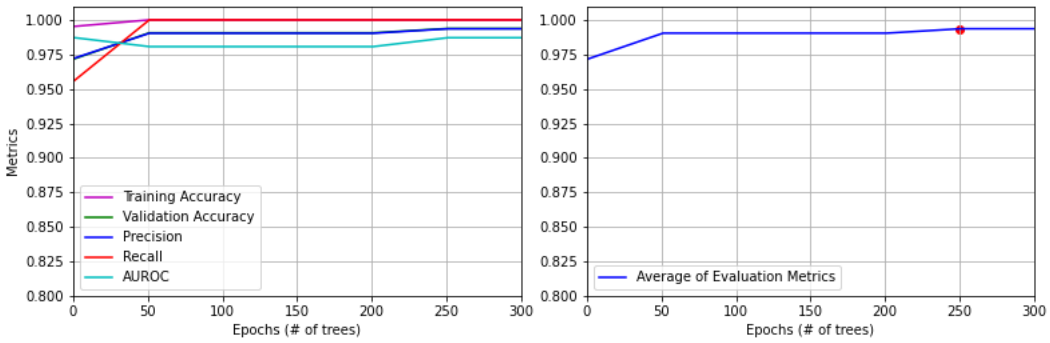

Fig. 16. Optimization of RF Algorithm with Radar Data

3.1.2 DNN Analysis

The ROC curve (Fig. 17) shows that DNN is likewise performing exceptionally well classifying maneuvers of LEO satellites.

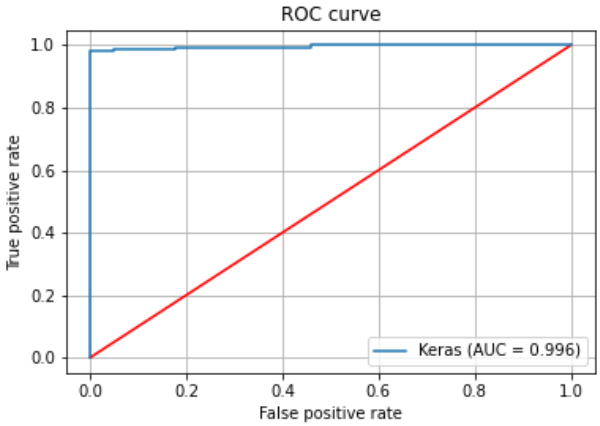

Fig. 17. ROC Curve for DNN with Radar Data

We perform a similar optimization for DNN as we do for RF, except this time we expand the test length to 500 epochs. This method shows that 350 epochs is ideal for this model.

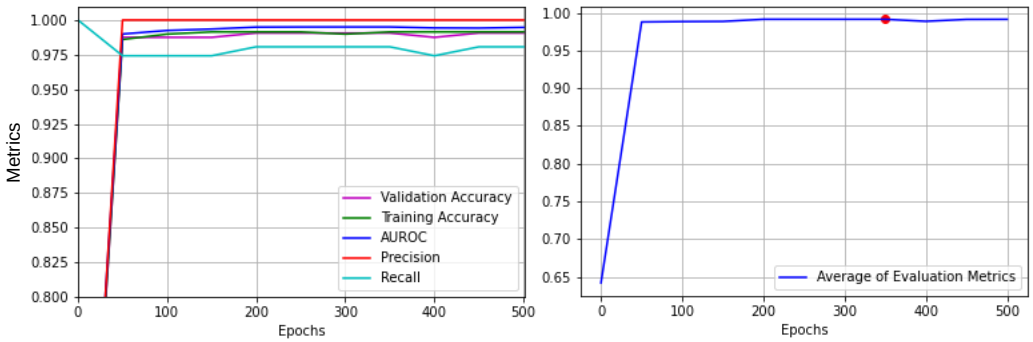

Fig. 18. Optimization of DNN with Radar Data

3.2 Results with Real Optical Residuals with Optimized NN

After acquiring real optical data from 18 SDS, we ran the optimized DNN model on different combinations of real and simulated data to determine (a) how well the model performs on real data, and (b) how well the simulated data mimics real data. We only had a small number of real cases: 100 maneuvering (M) and 100 non-maneuvering (NM). The first test we ran used real maneuver cases and simulated non-maneuver cases: 100 each for training and validation. The second test used 2000 simulated cases to train the model and 200 real cases to validate the model. The third test trained and validated solely on real data. The reported detection rate and false alarm rate are the averages of three separate runs.

Table 10. Optimized DNN Performance on Real Optical Data

Test	# Simulated training	# Simulated validation	# Real training	# Real validation	Detection Rate	False Alarm Rate
#1	100 NM	100 NM	100 M	100 M	100%	0%
#2	1000 NM, 1000 M	none	none	100 NM, 100 M	93.0%	21.8%
#3	none	none	100 NM, 100 M	100 NM, 100 M	98.5%	1.8%

Test #1 randomized and matched the numbers of simulated cases with the number of real cases to avoid skewing our model towards one type of case. The results show that simulated maneuver cases mimic real cases very closely and that the model performs exceptionally well on real maneuver cases. Fig. 19a shows testing and training accuracy over epochs. The model seems to reach maximum performance at 100 epochs and does not appear to be over-training. The ROC curve in Fig. 19b shows perfect separability.

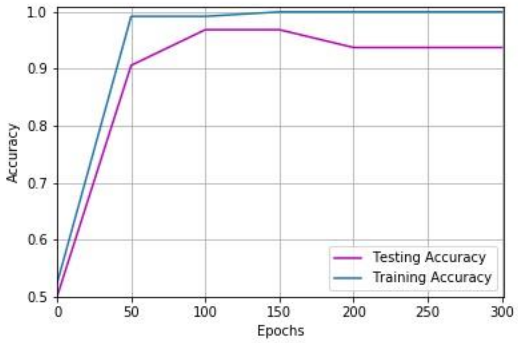

Fig. 19a. Accuracy vs Epochs (Test #1)

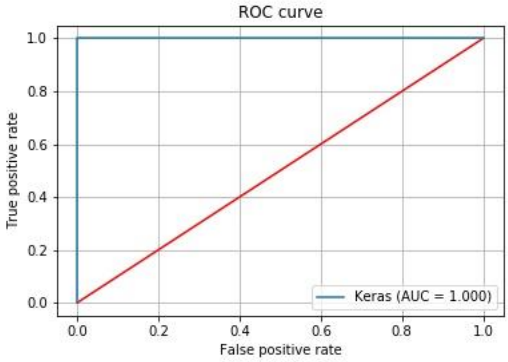

Fig. 19b. ROC Curve (Test #1)

Test #2 reveals that training solely on simulated data produces a relatively high detection rate but also a high false alarm rate. Since the high false alarm rate is due to a hypersensitivity to non-maneuvers, this indicates that simulated non-maneuver cases might not contain the appropriate magnitude or number of outlier residuals as real non-maneuver cases. The accuracy vs epochs plot in Fig. 20a shows that the model reaches near-peak performance at only 50 epochs, with very slight increases up to 300 epochs. A ROC curve (Fig. 20b) shows good separability.

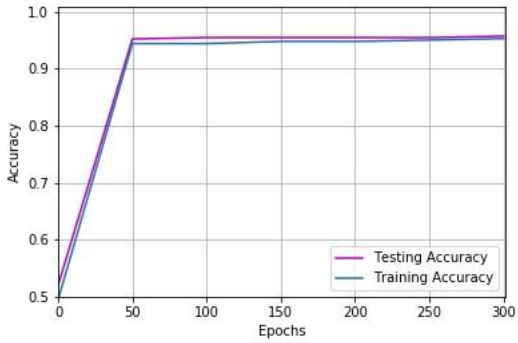

Fig. 20a. Accuracy vs Epochs (Test #2)

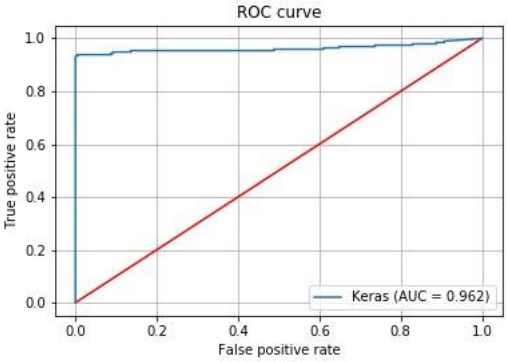

Fig. 20b. ROC Curve (Test #2)

Test #3 shows good performance with solely real data for training and validation. It has a higher detection rate than the optimized DNN with only simulated data (98.5% vs 96.7%), but also a higher false alarm rate (1.8% vs 0.05%), meaning that there are several non-maneuvers that appear as maneuvers in the real data. By examining the data, we find that there are several cases that resemble the ones plotted in Fig. 12 and 13 (non-maneuver cases that look almost identical to late-maneuvering cases). This again indicates that the simulated non-maneuvers could benefit from an increase both in outlier magnitude and quantity. It is interesting to note that the testing accuracy is consistently higher than training accuracy when plotted over epochs (Fig. 21a), indicating that there is potentially not enough training data. Future work in this area would be to acquire more real observations from 18 SDS and curate more real cases. Overall the results from test #3 show that the DNN performs far better than current statistical methods on detecting maneuvers.

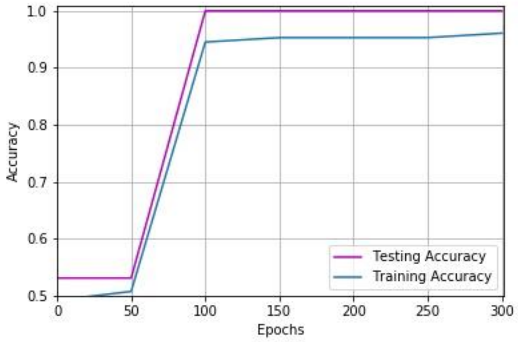

Fig. 21a. Accuracy vs Epochs (Test #3)

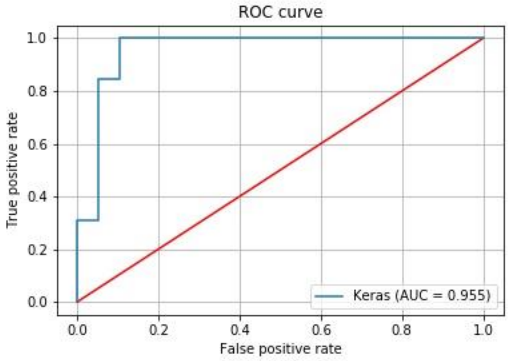

Fig. 21b. ROC Curve (Test #3)

3.3 Results from Maneuver Timestamping Models

The target value of the timestamping models is the index of observational data after the maneuver starts. For example, if a case contains 280 observations, and the maneuver starts between observations 99 and 100, the target value for this case would be 100. The evaluation metrics we use are based on how close the model is to the correct index. The training data for the model consists of simulated RA and DEC optical observation residuals. The metrics we use to evaluate the accuracy of the timestamping methods are:

1. Compare average index error to a baseline index error. The index error computes the distance from the predicted maneuver index to the actual maneuver index. This is then compared to a "baseline" (base) index error for an algorithm that always selects the middle observation index as its maneuver prediction. If the average index error is less than the baseline, the model is performing better than a predetermined algorithm.
2. Fit slope. When the magnitudes of the predicted indices are plotted from smallest to largest against timestep count, the slope of a linear fit should be positive if the model is detecting trends in the data. The closer the slope of the predictions is to the slope of the truth target values, the better the fit.
3. R^2 value. The closer the predicted indices are to a linear fit from observation 0 to max, the better the predictions are.

We developed two different LSTM models: one using a multi-label classification method, and one using a regression method. Multi-label classification requires the target value to be inputted as a vector the same length as the number of observations, with zeros for all indices where the maneuver did not start, and a one for the maneuver start index. This method of data encoding is known as "one-hot encoding". Regression requires the target value to be inputted as the integer of the maneuver start. Multi-label classification LSTM outputs return a probability distribution for all possible indices, with the highest probability associated with the predicted index. Regression LSTM outputs return a singular number prediction. The Transformer model also returns a probability distribution. For the LSTM multi-label classifier, we used a *categorical crossentropy* loss function and *Adam* optimizer. For the regression model we used *mean squared error* loss. We used *linear* activation functions for all layers with integer and one-hot encoded inputs, except for the one-hot encoded output layer, which used a *softmax* activation function to return a multi-label probability vector. The Transformer model used a sparse categorical crossentropy loss function, Adam optimizer, *linear* activation functions for hidden layers, and a *softmax* activation function for the output layer.

Table 11. Performance Metrics for Maneuver Timestamping Algorithms

Method	Label Format	LSTM Layers	Dense Layers	Avg Index Error	Avg Index Error (base)	R^2	R^2 (base)	Slope	Slope (base)
LSTM	**One-Hot**	1	0	32.50	24.51	0.0086	0.9611	0.0585	0.4692
		2	0	39.05	24.51	0.0098	0.9611	0.0643	0.4692
		2	1	22.30	24.51	0.0051	0.9611	0.0110	0.4692
	Integer	1	0	29.45	24.51	0.1272	0.9611	0.1201	0.4692
		2	0	22.12	24.51	0.0864	0.9611	0.1057	0.4692
		2	1	48.65	24.51	0.0226	0.9611	0.0680	0.4692

Method	Label Format	Transformer Blocks (1 attention and 1 dense layer each)	Avg Index Error	Avg Index Error (base)	R^2	R^2 (base)	Slope	Slope (base)
Transformer	**Integer**	1	37.06	26.68	0.0498	0.9408	0.1432	0.5815
		2	25.97	26.68	0.0260	0.9408	0.0757	0.5815
		4	40.44	26.68	0.0593	0.9408	0.1567	0.5815

The method that consistently returns the highest R^2 values and fit slopes was an LSTM with integer labels. Adding a dense layer to the LSTM networks decreased performance, and we found that adding more than one dense layer reduced the accuracy considerably. The method with the highest R^2 value is an integer-label LSTM with only one LSTM layer. The method with the closest slope to the baseline is a Transformer with one block, even though this method had a lower R^2 value than the LSTM (0.0498 vs 0.1272). The method that returned the lowest index error relative to the baseline is an integer-label LSTM with two LSTM layers (9.75% decrease in error over baseline). While the models did not perform as well as we expected, we can conclude that an AI-based approach to maneuver timestamping can help improve this task. We hypothesize that the model didn't perform as well as expected for one or more of three reasons:

1. There isn't enough training data to sufficiently allow the model to recognize maneuver patterns
2. The problem is too complicated for the methods we chose
3. A confluence of factors from non-optimized hyperparameters, data normalization, and model complexity

Future work in this area is to optimize the hyperparameters of the methods presented in this study, feed the model at least an order of magnitude more training data, and explore other AI methods.

4. CONCLUSIONS AND FUTURE WORK

Simulated data for optical sensors tracking GEO objects and radar sensors tracking LEO objects was presented. Simulated residuals for maneuvering and non-maneuvering cases were generated using Monte-Carlo simulations programmed so that they closely resemble real data. Real optical residuals were curated, including difficult maneuvering and non-maneuvering cases that closely resemble each other, even to the trained observer. RF and DNN algorithms were applied to simulated optical and radar data, and several of their hyperparameters were optimized. RF and DNN were also trained using 200 cases of real optical data and showed exceptional performance when compared with current statistical methods. LSTM and Transformer methods were tested as maneuvering timestamping algorithms. LSTM and Transformer were used to build two different multi-label classifiers, and LSTM was used to develop a regressor.

We showed that our optimized DNN model improved on the previous non-optimized DNN when trained with simulated optical data. It increased the detection rate from 94% to 97% and decreased the false alarm rate by an order of magnitude from 0.5% to 0.05%. Because DNN performance is far better than RF in both metrics, we suggest an SDA tool comprised of a DNN for optical (GEO) data. When trained with simulated radar data, RF outperformed DNN in detection rate (98% vs 97%), but the DNN outperformed in false alarm rate (0.05% vs 0.10%). To capitalize on the strengths of both methods, we hypothesize that an effective detection algorithm could first pass the data to an RF algorithm, then filter the predicted maneuvers through a DNN to filter out false alarms. This method has not yet been tested.

This study demonstrated that our DNN model performs exceptionally well on real optical data provided by 18 SDS and curated by trained analysts. The detection rate is better than when trained on simulated data (99% vs 97%), however the false alarm rate does increase when real data is introduced (2% vs 0.05%). This is likely due to the fact that simulated non-maneuvers might not sufficiently represent the quantity or magnitude of outliers in real cases. As a comparison, current statistical methods generally attain a detection rate of 94% and false alarm rate of 8% on real optical data [5]. It is important to note that there are no publicly-available metrics with which to compare the performance of our radar maneuver detection algorithm. Generating these statistics from existing methods would provide insight into the potential improvements with AI. Future work would involve testing our model on real radar data to determine if its performance compares to that with simulated data.

Maneuver timestamping models didn't perform as well as expected. The best performance was achieved with a network using only one or two LSTM layers. We found, generally, that simpler models performed better than complicated ones. As soon as the models became complex (adding dense layers, epochs, and batches), they began to over-fit. However, we were able to achieve a 9.75% decrease in error when compared to a baseline model that predicts the center of the observation indices every time. There is potential for AI to be a powerful tool in maneuver timestamping, but more work is required to reach this goal. We propose the following next steps: (a) Optimize the hyperparameters of LSTM and Transformer models, (b) Feed the model significantly more training data (we suggest at least 10,000 maneuver cases), and (c) Explore other AI methods for running regression and multi-label classification problems, such as BERT [15] and Raindrop [16].

5. REFERENCES

[1] N. Perovich, R. Jaimes and Z. Folcik, "Satellite Maneuver Detection Using Machine Learning Methods," in *2022 IEEE Aerospace Conference*, 2022.

[2] Space-Track.Org, "United States Space Force 18th Space Defense Squadron," United States Space Force, [Online]. Available: https://www.space-track.org/.

[3] Peterson Space Force Base, "18th Space Defense Squadron," United States Air Force Website, [Online]. Available: https://www.peterson.spaceforce.mil/About-Us/Fact-Sheets/Display/Article/2356622/18th-space-defense-squadron/.

[4] N. Perovich and N. Clark, "Policy Suggestions for Handling Space Situational Awareness in the Modern Space Environment," *16.891 Space Policy, Massachusetts Institute of Technology, Final Project*, 2022.

[5] Z. Folcik, P. Cefola and R. Abbott, "GEO Maneuver Detection for Space Situational Awareness," *Advances in the Astronautical Sciences*, 2008.

[6] B. S. Aaron, "Geosynchronous Satellite Maneuver Detection and Orbit Recovery Using Ground Based Optical Tracking," *Master's Thesis, Massachusets Institute of Technology*, 2006.

[7] 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 and M. Perrot, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[8] N. Perovich, Z. Folcik and R. Jaimes, "Satellite Maneuver Detection Using Artificial Intelligence Methods," in *IEEE Aerospace Conference Presentations*, Big Sky, MT, 2022.

[9] F. Chollet et. al., "Keras," 2015. [Online]. Available: https://keras.io.

[10] A. Mittal, "Understanding RNN and LSTM," Medium, 12 10 2019. [Online]. Available: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9(8), pp. 1735-1780, 1997.

[12] A. Graves, S. Fernandez and J. Schmidhuber, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," in *Artificial Neural Networks: Formal Models and Their Applications - ICANN*, Warsaw, Poland, 2005.

[13] A. Zeyer, P. Bahar, K. Irie, R. Schlüter and H. Ney, "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop*, Singapore, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," *Computing Research Repository*, vol. 1706.03762, 2017.

[15] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "Pre-training of Deep Bidirectional Transformers for Language," *Computing Research Repository*, vol. 1810.04805, 2018.

[16] X. Zhang, M. Zeman, T. Tsiligkaridis and M. Zitnik, "Graph-Guided Network for Irregularly Sampled Multivariate Time Series," in *International Conference on Learning Representations*, 2022.