Time forecasting satellite light curve patterns using Neural Networks

William Dupree
Aptima, Inc.
Louis Penafiel
Aptima, Inc.
Thomas Gemmer
Aptima, Inc.

CONFERENCE PAPER

Abstract: In this work we explore the application of Neural Networks to Space Domain Awareness (SDA) by modeling light curve predictions from historical data. This is done using a variety of models, including Feed Forward/Deep Neural (DNN) [1] and Long Short-Term Memory (LSTM) [2] Networks. Our current results serve as a strong indication that this type of approach will yield new advantages towards light curve anomaly and Pattern of Life (PoL) detection when used with ground-based sensor measurement data.

1. INTRODUCTION

With levels of technology increasing rapidly world-wide, space has become an increasingly contested domain in defense. It is imperative that the United States continues to grow and improve upon its current Space Domain Awareness (SDA) capabilities. Having greater knowledge of one's environment leads to safer and improved decision-making while operating inside of it. In this study we focus on improving SDA by characterizing satellites and their patterns of life (PoL) through analyzing historical data to forecast Visual Magnitude and light curve patterns. In previous work [3, 4], we have been successful in showing the application of Machine Learning (ML) to SDA data gives rise to improved detection of satellite maneuvers as a function of time. In this study we take a similar approach and apply ML techniques to satellite observational data to predict future trends in Visual Magnitude/light curves for a single satellite. We create models that characterize light curves based on a finite input window of observable measurements and output possible patterns the light curve may exhibit. We explore how these patterns can be used to alert analysts of anomalous activity when compared to the real-time data. Here we apply ML and Neural Networks to time series forecasting, focusing more on the application of these tools than delivering a fundamental background and derivation of Deep Learning. In Section 2 we introduce the data set we will be using, Section 3 we set up the feature space for future modeling, Section 4 we perform model experiments and benchmarking, and in Section 5 we give our conclusion.

2. LIGHT CURVE DATASET AS A NON-UNIFORM TIME SERIES

The data used in our study comes from EchoStar 7, a GEO satellite with Norad ID 27378. It was gathered over a period of roughly nine months, between August 2016 to April 2017. The data strongly resembles the challenges in using current Electro-Optical (EO) data and has a large number of samples (nearly 519 thousand time points) to add robustness to future modeling approaches.

Before we apply Machine Learning (ML) models to the data, we preprocess the data, performing general cleaning and feature engineering tasks. The first challenge we face in using light curve observational data typical to that in EO data is the observational data can be locally dense around small time scales but globally sparse in nature depending on sensor tasking and settings. Thus, the first step in our preprocess procedure is to gain general understanding of basic characteristics of the dataset, and how to handle the non-uniform sampling of the data.

Irregular time steps complicate what it means to predict/forecast the Visual Magnitude at the next time step. Much of the irregularity comes from the nature of satellite observation and how the majority of optical sensors observe them at night only. Other reasons may be due to specific sensor tasking or hardware challenges like individual differing sensor sampling rates. Due to these irregularities in the light curve data the model may be able to speak to the magnitude of a given measurement, but unless handled carefully will not be able to associate "when" the measurement occurred. If the difference in time between measurements becomes too large, the relevance between samples is drastically decreased.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

The data we use for EchoStar 7 is not uniform in sampling and has many gaps between nightly visibility. This trend can be seen in Fig. 1. Several large gaps can be seen in the 519K data points (Fig. 1, top), while over shorter time periods we can see the nightly cycle of Visual Magnitude values shown in "v" shapes (Fig. 2, bottom) as the satellite travels near a given sensor separated by time values of less than a day. A major challenge arises even amongst the nightly data, as every night does not always guarantee uniform sampling or the same number of measurements taken.

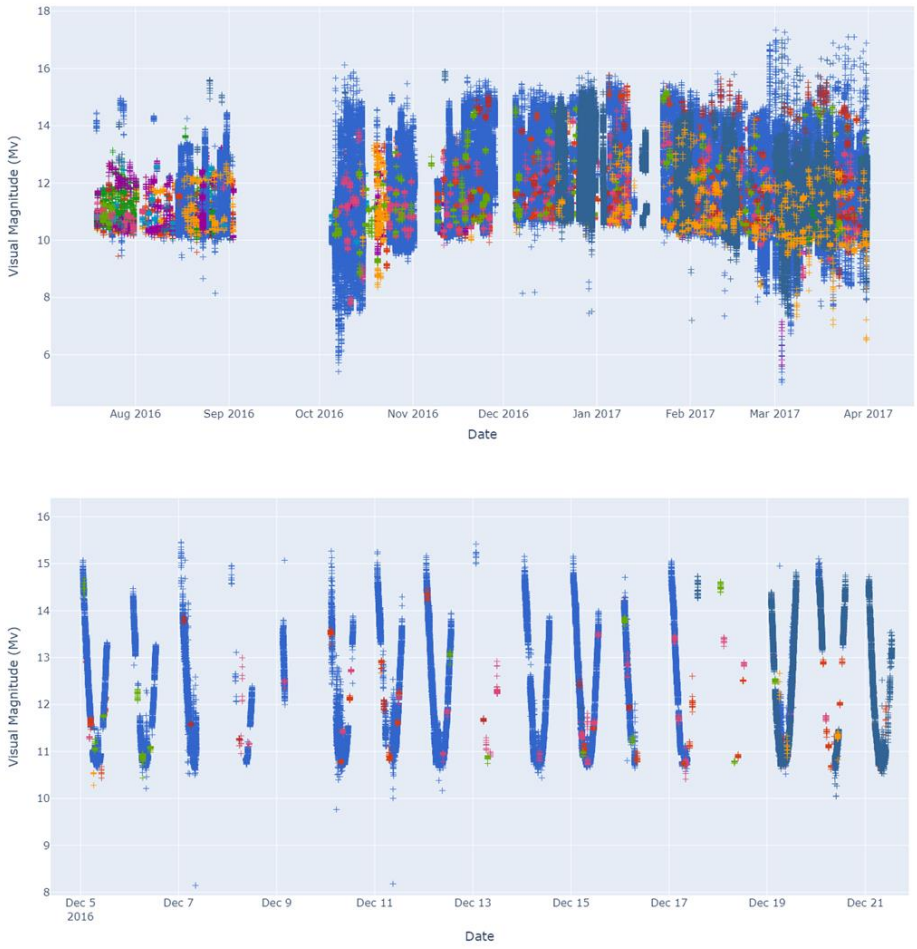

Fig. 1. EchoStar 7 light curve observations. **Top**: This figure shows the raw observations for EchoStar 7 between August 2016 and April 2017. The x-axis is date and the y-axis is Visual Magnitude measurements in *Mv*. The color of the scatter point corresponds which ground-based sensor a given measurement was taken from. **Bottom**: A zoomed-in view of the light curve data, showing the nightly periodicity of Visual Magnitude PoL.

It will help guide our future analysis if we can systematically characterize both these gaps in measurements as well as the density of samples. In this scenario density refers to the number of samples/measurements taken under a set time frame (points per time). To do so we focus on the timestamp of when an observation was made on an individual basis, and group/cluster the data based on a selectable threshold (i.e., 15 mins). We do this instead of aggregating samples, since the gap in time between nightly sequences would still introduce many missing values/unobserved sections for short duration aggregation and a coarse aggregation would drastically limit the sensitivity to light curve trends over a meaningful time period.

To cluster our measurements, we group samples together into sections that have a gap less than 15 minutes apart. As an example, if we began taking measurements over an hour period, all measurements would be assigned group 1. If at 35 minutes into collecting data we did not receive another observation for 15 minutes, we would begin labelling the data in a new group. The samples collected from timestamp 50 minutes and on would be put into group

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

2. We would then repeat this process for all collected data, continuing for as many observations as we had and updating groups for gaps greater than 15 minutes.

This is the process we follow for our EchoStar 7 data, grouping and clustering by gaps in time. We are then able to analyze how many observations belong to a given cluster and quantify the complex nature of the dataset's temporal density. Results of this procedure for the training/cross-validation data (described in Section 4) are shown in the histograms displayed in Fig. 2.

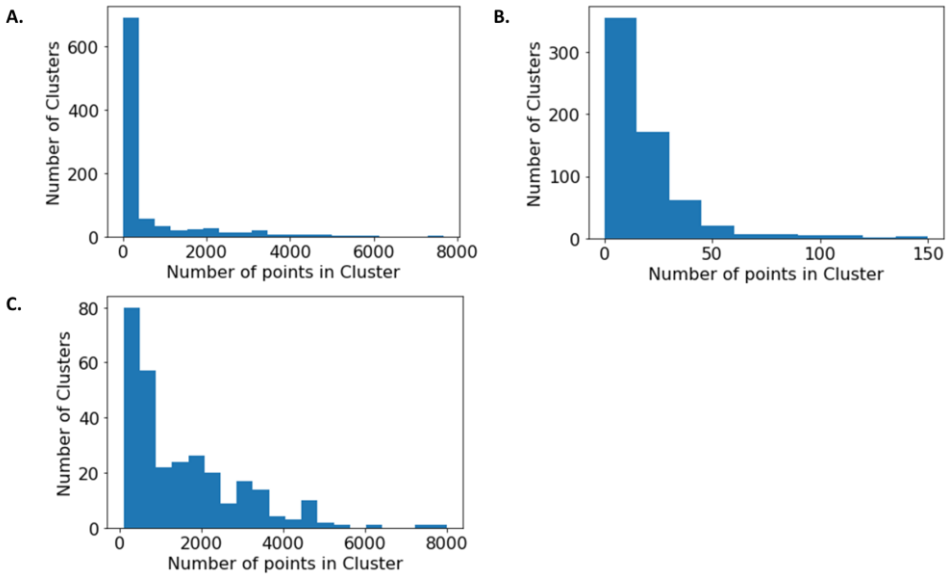

Fig. 2. In these figures we display the density of the data, described through clustering the observations around gaps in measurements greater than 15 minutes. The x-axis is the number of observations within a given cluster, and the y-axis is the number of clusters. **A)** This shows the full range of the density. We can see that most clusters have less than 1000 samples. **B)** We show the small range, casting the histogram with clusters between 0 and 150 observations within. **C)** This shows clusters between 100 and 8,000 observations.

This type of clustering is beneficial in characterizing large gaps in time existing in the data, as well as the temporal density around the non-uniform sampling. The histograms in Fig. 2 express the extreme challenges light curve analysis faces when the data temporal density is quite complex. There is a large number of clusters that have less than 200 observations/measurements, meaning this dataset contains many very small clusters of observations that are separated by more than 15 minutes from the rest of the data on either side. Other sections of the dataset have sections of time where there is no gap greater than 15 minutes for 1,000 to 5,000 observations in a row. These sections of the time series are much denser. Since we have nightly optical measurements this would make sense, as most nights we obtain relatively steady data. At the tail we observe several sections of data where 5,000+ samples are taken without a 15-minute break. We use this understanding of the EchoStar 7 temporal density to motivate future historical window parameter choices when searching for the optimal window size, as seen in Section 4.3.

We stress this globally sparse/locally dense nature to preface our approach for Visual Magnitude forecasting. In most forecasting approaches uniform sampling is required so that common frequency/regression/convolution approaches can be taken. Without uniform sampling there is no way for a method to tell *when* the forecasted measurements take place. Many basic methods do not allow non-uniform sampling in the input data for the algorithm to perform. We approach this problem from an abstract sense. Time is typically an independent variable in a forecasting problem that drives the rest of the solution. However, due to the stochastic nature of when optical measurements are made from the ground-based sensors, we cannot in this instance treat it as such to say how many or when a measurement(s) will come. In these first iterations of this work, we only tackle forecasting the magnitude of the light curve, and not *when* the values actually arrive. We will describe the models we use in greater detail in the beginning of Section 4, but for now we focus purely on the series generation aspect of the problem.

In doing so, we formulate a solution that forecasts the very next light curve measurement as to avoid multiple predictions into the future at once. Working under these assumptions, we could think of the samples being of uniform sampling in some abstract space, but not seek conversion between that reference time and the original time frame. Framing the problem in this fashion then naturally lends itself to Neural Networks, where the model and its

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

hidden layers can learn from the sequence of values unburdened by the human aspect of *when* a measurement is from. This idea of teaching Neural Networks about sequences, especially cyclical sequences, leads us to applying a Long Short-Term Memory (LSTM) Network [2] as our solution of choice.

3. TIME SERIES ROLLING WINDOW PREPARATION FOR NEURAL NETWORKS

In this section we prepare the data for machine learning. We do so by cleaning the data, performing feature creation, and formatting the data in a way that is most beneficial for a model to learn the intricacies of the sequential nature of the light curve data. As a result, we cover what numerical features we have/can calculate, and what we encode into relevant numerical information from categorical features included in the data set. The feature engineering is done to mitigate characteristics in the raw data that may weaken the prediction power of our final model. We also describe and show examples of the rolling window approach we took to give temporal understanding to our data and to prepare it as inputs in an LSTM Neural Network [2]. Doing so allows us to connect the past nature of the data to current measurements.

Before explaining the rolling window process, it is best to step back and briefly look at the problem from a very basic level. We apply Deep Learning to model and predict Visual Magnitude values, but at its core this is an ML supervised learning regression style problem. Supervised learning is the idea of teaching a computer through algorithms and data pairs of samples and targets. In regression problems we have a set of independent data we wish to use to model the values of a dependent variable. In the case of a single independent variable, this argument simplifies to the idea of a "best-fit line", where the equation of a line can be modeled and used on unseen data to give a better understanding of what its dependent value might be. We have been referring to the independent variables as our features, and the dependent variable we wish to predict as the target. If one were to think of the data as a table, such as Fig. 3, we could separate the target and features as seen in Fig. 4. In this way, each row is a sample consisting of features and a single target value. If we give the model enough samples, we hope that we can learn the behavior between feature and target to predict on previously unseen feature data with high accuracy.

	Observation Time	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness
0	2016-07-19 02:13:04.222997725	14.180812	94.844254	20.784191	0.082455
1	2016-07-19 02:13:43.682972789	13.987119	94.689554	20.784108	0.098558
2	2016-07-19 02:13:51.579007208	14.094036	94.659252	20.784091	0.089316
3	2016-07-19 02:14:07.353011370	14.190595	94.597813	20.784057	0.081715
4	2016-07-19 02:14:15.221003294	14.411509	94.567107	20.784041	0.066671
5	2016-07-19 02:14:23.107019663	14.452612	94.536735	20.784024	0.064194
6	2016-07-19 02:14:34.984967709	13.922983	94.490309	20.783999	0.104556
7	2016-07-19 02:14:42.860000432	14.069095	94.459706	20.783982	0.091391
8	2016-07-19 02:14:50.734992921	13.814418	94.429659	20.783965	0.115551
9	2016-07-19 02:14:58.639033735	13.971037	94.398640	20.783948	0.100029

Fig. 3. Example data from Satellite EchoStar 7. In the orange box we see numeric data we wish to use to model the behavior of the Visual Magnitude. Here "Observation Time" is the timestamp of the measurement coming from a specific ground sensor.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

	Observation Time	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness
0	2016-07-19 02:13:04.222997725	14.180812	94.844254	20.784191	0.082455
1	2016-07-19 02:13:43.682972789	13.987119	94.689554	20.784108	0.098558
2	2016-07-19 02:13:51.579007208	14.094036	94.659252	20.784091	0.089316
3	2016-07-19 02:14:07.353011370	14.190595	94.597813	20.784057	0.081715
4	2016-07-19 02:14:15.221003294	14.411509	94.567107	20.784041	0.066671
5	2016-07-19 02:14:23.107019663	14.452612	94.536735	20.784024	0.064194
6	2016-07-19 02:14:34.984967709	13.922983	94.490309	20.783999	0.104556
7	2016-07-19 02:14:42.860000432	14.069095	94.459706	20.783982	0.091391
8	2016-07-19 02:14:50.734992921	13.814418	94.429659	20.783965	0.115551
9	2016-07-19 02:14:58.639033735	13.971037	94.398640	20.783948	0.100029

Fig. 4. In supervised learning we use features and target variables to build models and algorithms to predict new target values for unseen features. Here the blue box shows an example of the features we use, and the red is the target. In this way a model is trained to predict "Visual Magnitude" values for unseen feature data.

To build a strong model capable of predicting unforeseen data (measurements from the future) it will be important to teach the model using only historical data. Our data includes previous measurement fields such as Visual Magnitude, Solar Phase Angle (SPA), Solar Declination, and Brightness. Both Solar Phase Angle and Solar Declination are given in *degrees*, while the units of the Visual Magnitude are *Mv*. Not included in our approach is a column for a range estimate of the satellite. We take this approach due to missing range measurements in future data and is supported by the small variation in distance between a GEO satellite and ground-based sensor pairing. We currently use astronomical calculations to populate the Solar Phase Angle and Solar Declination columns. The Brightness is a transformation to 'relative brightness' between the flux density of the measured Visual Magnitude and the flux density of the average Visual Magnitude seen over the data set. It takes the form

$$3.1\ brightness = 10\text{^}[0.4(\overline{mag} - mag)]$$

where the overbar denotes an average over all Visual Magnitude values in the current data set and the factor of 0.4 is scaled appropriately to return a unitless value.

Even with non-uniform sampling we keep the data in chronological order, so that the model learns that sequencing is important, and create window samples for our feature/target pairs. This rolling window sample technique is standard in ML time series regression today, but we found inspiration for this method using TensorFlow [5]. The window moves over the samples in chronological order, stepping forward one sample at a time. An example of two rolling window samples can be seen in Fig. 5, where after the windowing process the blue box shows a single window sample, and the red is a single Visual Magnitude target. The model then learns from a large number of these feature/target window samples in order to predict future behavior. Opposed to the flat arrays that made up the data in Fig. 4, both feature and target variables will be multi-dimensional and consist of this sliding window in time. The dimension size is given as the number of window samples by the feature/target window size by the number of features/targets (or number of columns used). As an example, if the data in Fig. 4 had 100 measurements and we performed the windowing process as in Fig. 5 (window length of 5), the resultant dimensions would be *(sample size, window size, number of columns) = (95,5,4)* for the features, and *(sample size, window size, number of columns) = (95,5,1)* for the targets. Specifically, the number of window samples given to the model for training will be equal to the difference between number of raw observations and the window size. The size of the rolling window for the features can be adjusted, and later we will use it as a parameter to be tuned in the model to improve accuracy of our predictions.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness
0	14.180812	94.844254	20.784191	0.082455
1	13.987119	94.689554	20.784108	0.098558
2	14.094036	94.659252	20.784091	0.089316
3	14.190595	94.597813	20.784057	0.081715
4	14.411509	94.567107	20.784041	0.066671
5	14.452612	94.536735	20.784024	0.064194
6	13.922983	94.490309	20.783999	0.104556
7	14.069095	94.459706	20.783982	0.091391
8	13.814418	94.429659	20.783965	0.115551
9	13.971037	94.398640	20.783948	0.100029

	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness
0	14.180812	94.844254	20.784191	0.082455
1	13.987119	94.689554	20.784108	0.098558
2	14.094036	94.659252	20.784091	0.089316
3	14.190595	94.597813	20.784057	0.081715
4	14.411509	94.567107	20.784041	0.066671
5	14.452612	94.536735	20.784024	0.064194
6	13.922983	94.490309	20.783999	0.104556
7	14.069095	94.459706	20.783982	0.091391
8	13.814418	94.429659	20.783965	0.115551
9	13.971037	94.398640	20.783948	0.100029

Fig. 5. An example of creating time series windowed data, where our feature window has a length of five (number of rows used, blue box) and the target window a length of one (red box). We can see from the first sample to the second sample both windows have shifted forward one measurement. This process would be done for the whole data set, resulting in a new windowed multi-dimensional feature and target data set.

In this analysis we focus on a target window of size one, which we call a single time step prediction. The single step prediction is limited for future prediction, given that using historical data only allows the model to predict one sample into the future. We are careful here to discuss predictions by sample size and not by time frame due to the irregular time steps we discussed earlier. If we had regular time steps, we could reframe the problem in a way to predict an exact period into the future, but due to our constraints the best we can say is to predict one future sample from the model for a given rolling window sample.

After the windowing process we also perform several other feature engineering tasks to aid the success of the model. The first step is to normalize the previous numerical data to zero mean and unit variance, as is typically done for Neural Network applications. We then build in knowledge of the categorical sensor ID information via one-hot-encoding/dummy variables. One-hot-encoding is a process of creating new columns that hold binary numerical values and signify what sensor a measurement came from. This process can be seen in Fig. 6. Our current data limits us on numerical sensor information, so we take the one-hot-encoding approach [6]. In the future we hope to have greater access to a variety of providers and use geographical data (longitude/latitude) of the sensor location over this categorical encoding. In Fig. 6 the data was limited to 3 sensors, but the data for EchoStar 7 spanning our 9-month dataset has 24 sensors. The same process can be followed for the EchoStar 7 dataset, the major difference being the column space will grow from four (the original numerical) to 28 (24 sensor columns). The one-hot-encoding process also assumes that future data will not include a new sensor, as data input shape must remain the same once a model is trained on a given set of data. There are ways around this, such as pre-network feature encoding similar to Natural Language Processing techniques (e.g.,Word2vec) [7, 8], but this approach is out of scope for our current analysis.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

	Observation Time	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness	Sensor 01	Sensor 02	Sensor 03
0	2016-07-19 02:13:04.222997725	14.180812	94.844254	20.784191	0.082455	0	1	0
1	2016-07-19 02:13:43.682972789	13.987119	94.689554	20.784108	0.098558	0	1	0
2	2016-07-19 02:13:51.579007208	14.094036	94.659252	20.784091	0.089316	0	1	0
3	2016-07-19 02:14:07.353011370	14.190595	94.597813	20.784057	0.081715	0	1	0
4	2016-07-19 02:14:15.221003294	14.411509	94.567107	20.784041	0.066671	0	1	0
5	2016-07-19 02:14:23.107019663	14.452612	94.536735	20.784024	0.064194	0	1	0
6	2016-07-19 02:14:34.984967709	13.922983	94.490309	20.783999	0.104556	0	1	0
7	2016-07-19 02:14:42.860000432	14.069095	94.459706	20.783982	0.091391	0	1	0
8	2016-07-19 02:14:50.734992921	13.814418	94.429659	20.783965	0.115551	0	1	0
9	2016-07-19 02:14:58.639033735	13.971037	94.398640	20.783948	0.100029	0	1	0

Fig. 6. We capture the categorical nature of which sensor a measurement was taken from through a process of one-hot-encoding. Here, for a list of Sensor IDs, we create new columns that serve as true/false if a measurement came from that specific sensor. The last three columns in the orange box would be these sensor columns, wherein this example all measurements are from Sensor 02.

Finally, to aid against our irregular time steps for every windowed sample, we create new columns that aid in the model learning about the time constraint through the difference between measurement values. This is done by taking the difference between the first value in a window sample for a given feature and all later values, creating columns that begin at zero and track the change in that value as the measurement evolves in time. Every time the window is moved forward the process is reset; an example is shown in Fig. 7. In our analysis we do not combine multiple satellites but handle a satellite separately with its own model.

In this section we described what features our model will use, and how the rolling window sampling will improve the model's ability to learn temporal dependencies in a non-uniform dataset. We also augmented the data by feature engineering numerical values to describe both categorical sensor information inside of the data and the rolling difference between measured values. Preparing the data this way will allow us to use Neural Networks to input a rolling window of data and output a single prediction (per sample) to forecast the Visual Magnitude.

	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness	Sensor 01	Sensor 02	Sensor 03	Differenced Visual Magnitude	Differenced Solar Phase Angle	Differenced Solar Declination	Differenced Brightness
0	2.187405	2.046367	3.638722	-0.573436	0.0	1.0	0.0	0.000000	0.000000	0.000000	0.000000
1	2.026482	2.039380	3.638714	-0.567053	0.0	1.0	0.0	-0.193693	-0.154700	-0.000084	0.016104
2	2.115311	2.038011	3.638712	-0.570717	0.0	1.0	0.0	-0.086776	-0.185002	-0.000101	0.006861
3	2.195533	2.035237	3.638708	-0.573729	0.0	1.0	0.0	0.009783	-0.246441	-0.000134	-0.000740
4	2.379072	2.033850	3.638707	-0.579693	0.0	1.0	0.0	0.230697	-0.277147	-0.000151	-0.015784
5	2.413221	2.032478	3.638705	-0.580675	0.0	1.0	0.0	0.271800	-0.307520	-0.000167	-0.018261
6	1.973197	2.030381	3.638702	-0.564676	0.0	1.0	0.0	-0.257829	-0.353945	-0.000193	0.022101
7	2.094589	2.028999	3.638700	-0.569894	0.0	1.0	0.0	-0.111717	-0.384548	-0.000209	0.008936
8	1.883000	2.027642	3.638699	-0.560317	0.0	1.0	0.0	-0.366394	-0.414595	-0.000226	0.033096
9	2.013121	2.026241	3.638697	-0.566470	0.0	1.0	0.0	-0.209775	-0.445614	-0.000243	0.017574

	Visual Magnitude	Solar Phase Angle	Solar Declination	Brightness	Sensor 01	Sensor 02	Sensor 03	Differenced Visual Magnitude	Differenced Solar Phase Angle	Differenced Solar Declination	Differenced Brightness
0	2.026482	2.039380	3.638714	-0.567053	0.0	1.0	0.0	0.000000	0.000000	0.000000	0.000000
1	2.115311	2.038011	3.638712	-0.570717	0.0	1.0	0.0	0.106917	-0.030302	-0.000017	-0.009243
2	2.195533	2.035237	3.638708	-0.573729	0.0	1.0	0.0	0.203476	-0.091741	-0.000050	-0.016843
3	2.379072	2.033850	3.638707	-0.579693	0.0	1.0	0.0	0.424390	-0.122447	-0.000067	-0.031887
4	2.413221	2.032478	3.638705	-0.580675	0.0	1.0	0.0	0.465493	-0.152820	-0.000084	-0.034364
5	1.973197	2.030381	3.638702	-0.564676	0.0	1.0	0.0	-0.064136	-0.199245	-0.000109	0.005997
6	2.094589	2.028999	3.638700	-0.569894	0.0	1.0	0.0	0.081976	-0.229848	-0.000126	-0.007167
7	1.883000	2.027642	3.638699	-0.560317	0.0	1.0	0.0	-0.172701	-0.259895	-0.000142	0.016993
8	2.013121	2.026241	3.638697	-0.566470	0.0	1.0	0.0	-0.016082	-0.290915	-0.000159	0.001471
9	2.020638	2.024844	3.638695	-0.566799	0.0	1.0	0.0	-0.007034	-0.321844	-0.000176	0.000641

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

Fig. 7. In this image we share the current feature data used in our models. The original 4 columns can be seen to be scaled, followed by the one-hot-encoding sensor columns. The final 4 columns are the rolling difference, reset every time a new window is considered. The two blue boxes make up two samples of the multi-dimensional data set (samples, window length, features) we will use to train a model. Note how the difference columns reset inside of the blue box that is an example window.

4. NEURAL NETWORK EXPERIMENTS USING SATELLITE DATA

The goal of this work is to validate that Neural Networks are useful in the application of time series forecasting for satellite light curve analysis. We do this to improve SDA by identifying anomalous behavior when a light curve time series shows uncharacteristic behavior when compared to models framed from historical data. To this end, this section describes the experiments we performed to choose an accurate model, explored the ideal rolling window size for the EchoStar 7 dataset, and tested different feature sets all in the effort to produce a single model that will be used against hold-out data to test accuracy.

4.1 MODEL CHOICES AND FIRST PASS ACCURACY

So far, we have focused heavily on the raw data and the processing we used to prepare for modeling. The formatting is useful not only to avoid the traditional "looking ahead" bias that comes in time series analysis, but also in the approach we will take in using Recurrent Neural Networks (RNN) in the form of an LSTM [2] to model the Visual Magnitude behavior. We will use both LSTM and standard Feed Forward/ Deep Neural Networks (DNN) to analyze our data, exploring the results of networks and their ability in making future predictions. LSTM models are popular for time series due to their ability to analyze complex sequential data while capturing characteristics that may not occur frequently. We test these two types of networks against our current data after preprocessing and cleaning. To ensure the networks are worth the effort of development and future work, we also test their accuracy compared to that of a basic baseline model based on moving averages.

We input a sample consisting of a previous window of data and output a single Visual Magnitude value that is likely to come next in the sequence. The baseline does this by taking the average of the Visual Magnitude values given in the historic window and outputting it as the next best measurement. For most time series cases this approach is standard and typical under the assumption that the output is slow changing. It is a stretch to say our dataset fits this assumption well given the gaps in measurements, irregular sampling, and stochastic nature of when observations are made for a given ground-based sensor. However, the majority of the data fits the assumption well enough that we still chose this as our baseline to compare accuracy for future complex models involving Neural Networks.

We focus on two Neural Network models here, one being a Feed Forward DNN network with two hidden layers of 32 dense neurons each (sometimes referred to as Dense in our figures), and an LSTM model with a single layer and 32 LSTM cells. It is not in the scope of this discussion to delve into the exact inner workings of the LSTM cells, only to know we are taking advantage of the networks properties at learning and "remembering" unique trends in the sequenced data. Both networks have a final output layer with a single neuron to output the Visual Magnitude, while the DNN includes a flatten and reshape layer in its architecture to appropriately use the rolling window samples [5]. The two major parameters we can tune in this type of analysis are the number of neurons/cells in the given hidden layers and the size of the moving window we use on our feature data. We do not currently optimize for number of neurons/memory cells, but instead begin with simple networks. By using simple networks in the beginning stages to outperform the baseline averaging, we can be fairly confident that future and more complex models should perform the same, if not better.

As is typical in ML research and experiments, we split our data into a set of training data and a set of test data. It is important to set the test data aside for later, never training or changing parameters to the model after testing its accuracy. We do this to test as best we can the scenario of having new and unseen data by the model, avoid biases to the data, and to create a network that generalizes well to new data. We will refer to our training data as the cross-validation set (explained shortly), and the test data as the blind set. The cross-validation set is the first 90% of the total EchoStar 7 data, and the blind data is the remaining 10%. The training samples are only rearranged by shuffling the rolling window samples, but not by shuffling the entire time series, as the windowed samples are not created with shuffled data. The order of the windowed data is important to leave intact for the models to learn about time series sequencing. The data sets can be seen in Fig. 8. We do not predict on the blind set until the very end of the analysis once we have a chosen set of parameters. For this study our parameters will focus on the number of samples included in the rolling window, and what features given in Section 3 will be used for training.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

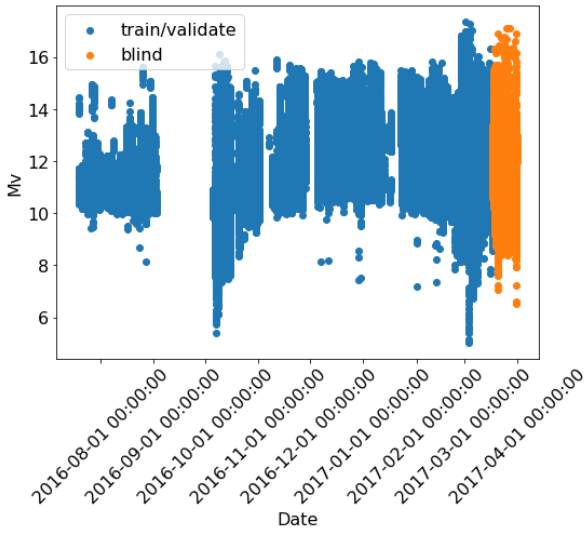

Fig. 8. Our data set for EchoStar 7 split into two sets: the cross-validation for training (blue) and the blind set (orange) for future accuracy testing. The x-axis is the timestamp, and the y-axis is the Visual Magnitude value in *Mv*. In this condensed view it is difficult to see the nightly cycles, but zooming in to a day view would reveal the trends shown in the bottom of Fig. 1. It is typical in ML research to set aside data for later use to remove bias while training a Neural Network/model by keeping unseen data for accuracy evaluation.

The first experiment we perform is taking the cross-validation set and training the network on the first 90% of the set and computing the accuracy on the remaining 10% for all three methods described above using basic parameters. This translates to 420,047 observations used for training and 46,672 observations used for accuracy validation before the windowing process is performed on the data. In this analysis we use Mean Squared Error (MSE) as our accuracy metric to compare actual measurements to predicted model values [9]. The MSE is defined by

$$4.1.1\ MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - \hat{p}_i)^2$$

where it is found by taking an average over n data points, where the true value p is compared to the model predicted value of $\hat{p}$. The lower the MSE score, the better the model has done at forecasting values. We choose the MSE to compare the continuous nature of Visual Magnitude values, as well as being a little more aggressive in penalizing incorrect predictions inherent in the quadratic nature of this metric.

Visual Magnitude observations for a portion of the 10% validation set can be seen in Fig. 9. This portion of the data looks uncharacteristic and anomalous, but we still move ahead with predicting the observations one sample at a time given our three methods.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

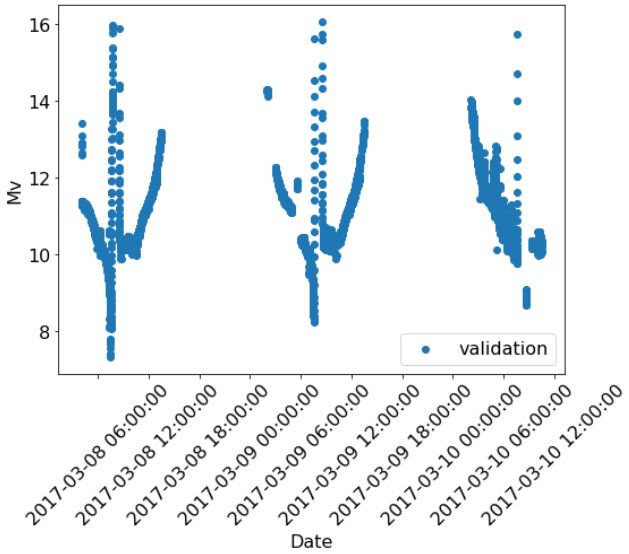

Fig. 9. A nightly view of the 10% validation portion of the cross-validation set. We again have the timestamp on the x-axis and the Visual Magnitude in *Mv* on the y-axis. This section of the data shows challenging data aspects, as the light curve is exhibiting a line directly in the middle of its typical "v" shape. This could be due to many things. Possibly in this case, the satellite was spinning causing the light curve to take on this unique pattern.

The rolling window size is initially set to a length of 10 (so that the past 10 samples are used to predict the next sample) and the features used are Visual Magnitude, Solar Phase Angle, Solar Declination, and Brightness. Our experiments were done in Python using the TensorFlow [5] and Keras [10] libraries, where the Neural Networks used the ADAM [11] optimizer and early stopping when accuracy gains slowed for training. We used early stopping in an effort to best reach model weight convergence, instead of training for a set number of Epochs/iterations for the DNN and LSTM models. Results for this first pass can be seen in Fig. 10, where we show predictions for a portion of the back 10% of the cross-validation set (the total set contains nearly 47,000 observations). Of the three networks the LSTM performs best with an MSE of 0.0223 *Mv squared*, followed by the DNN with a score of 0.0247 *Mv squared*, and last is the baseline model performing at 0.0359 *Mv squared*. In terms of MSE, lower scores are better, as we wish the predictions to lie closer to the actual measured values. We can see some challenged fits around the nightly gaps, but in the current state of the data this seems to be expected and normal behavior since drastic changes between samples cannot be smoothed without aggregation of some type. This is a good sign that both Neural Networks outperform the averaging baseline, and that the LSTM does best.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

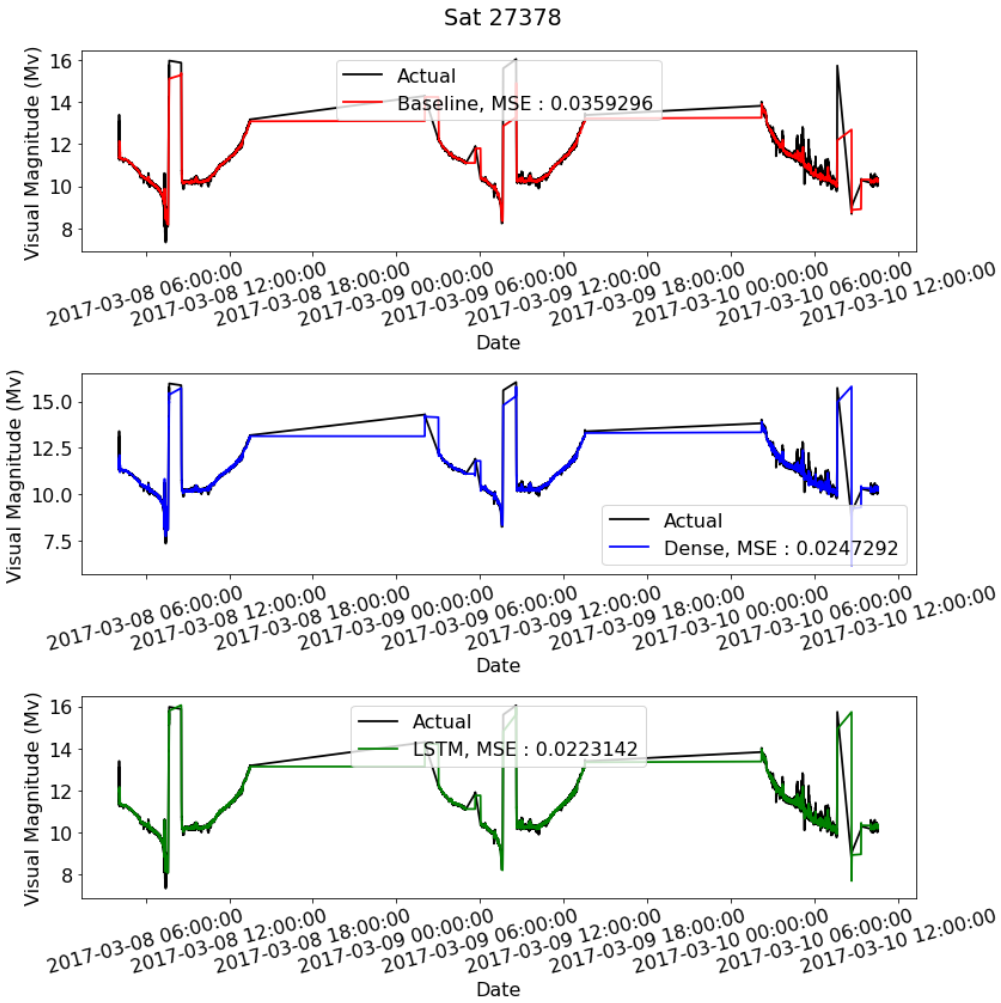

Fig. 10. Model prediction results for the first pass test using the three models of interest. For each figure, the x-axis shows the data while the y-axis shows the Visual Magnitude in *Mv*. We can see positive results through not only limiting the MSE score but can also visually tell model accuracy and success through the way the predicted line closely overlays the actual observations. **Top**: This is the baseline averaging model, taking the average of the 10 historical Visual Magnitude measurements that appear in the rolling window to predict the next sample (red lines are predictions, black lines are actuals). **Middle**: The DNN using multiple hidden layers and fully connected neurons to predict the Visual Magnitude (blue lines are predictions, black lines are actuals). **Bottom**: The LSTM network predicting the Visual Magnitude using a rolling window of size 10 (green lines are predictions, black lines are actuals). All three plots share the MSE on the 10% validation set, with the LSTM network performing best.

4.2 CROSS-VALIDATION APPROACH AND FINDING THE BEST PERFORMING MODEL

To ensure that the LSTM network truly is the best performer we use our cross-validation set to perform an iterative analysis over different data stretches. In ML this is called cross validating the data, hence our name for the training set. As we have stated previously, it is important in time series data and time series regression style problems to not shuffle the raw data to avoid the model learning about the future at the wrong time and becoming biased. Instead of shuffling for cross-validation, we take the cross-validation set and split it into 10 equal sections, each section having 10% of the data. We then iteratively perform training and prediction testing by increasing the amount of training data 10% at a time and predicting on the next 10% for validation until the data is exhausted. In other words, the amount of data used for training follows the sequence 10%, 20%, 30%, 40%....,90% of the cross-validation data. We can refer to the amount of training data as “Training Set Percentage Size”. This process can be seen visually in Fig. 11 for the first 5 iterations. This process is done until the final iteration where we use 90% of the cross-validation set to train and 10% to validate model MSE results.

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

Using this method of dividing our data for multiple tests we can now use each iteration to compare the 3 models. We again keep the rolling window parameter set at 10 samples and compare the models over the cross-validation set. For comparison we again use the MSE as our metric and show results of this test in Fig. 12.

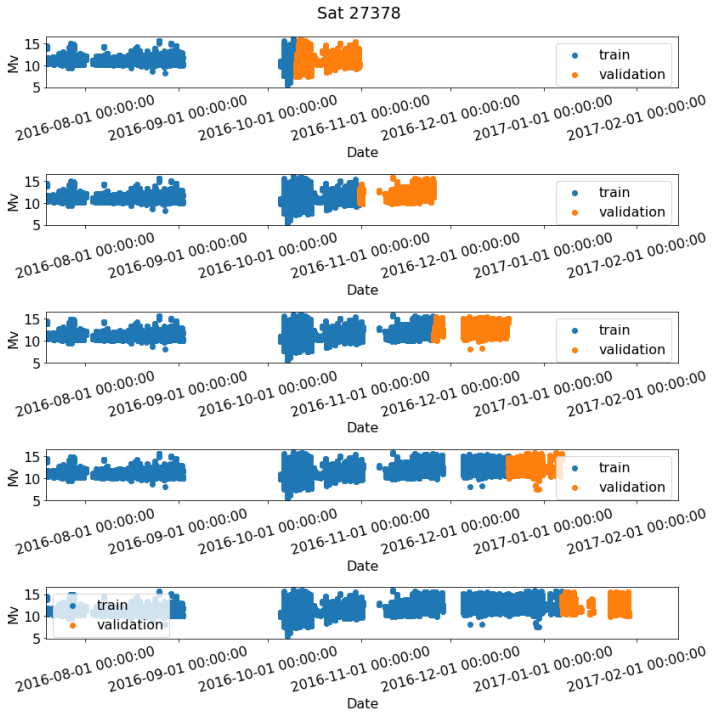

Fig. 11. First 5 cross-validation subsets used to test robustness of model outcome. For each figure, the x-axis shows the data while the y-axis shows the Visual Magnitude in *Mv*. Here we cannot truly see the shape or trend of the Visual Magnitude but use this to simply show the 10% iterative cross-validation plan. For each new subset we add 10% to the training data (blue) and predict the model outcome on the next 10% (orange). We do this recursively until we reach training with 90% and testing on the final 10%.

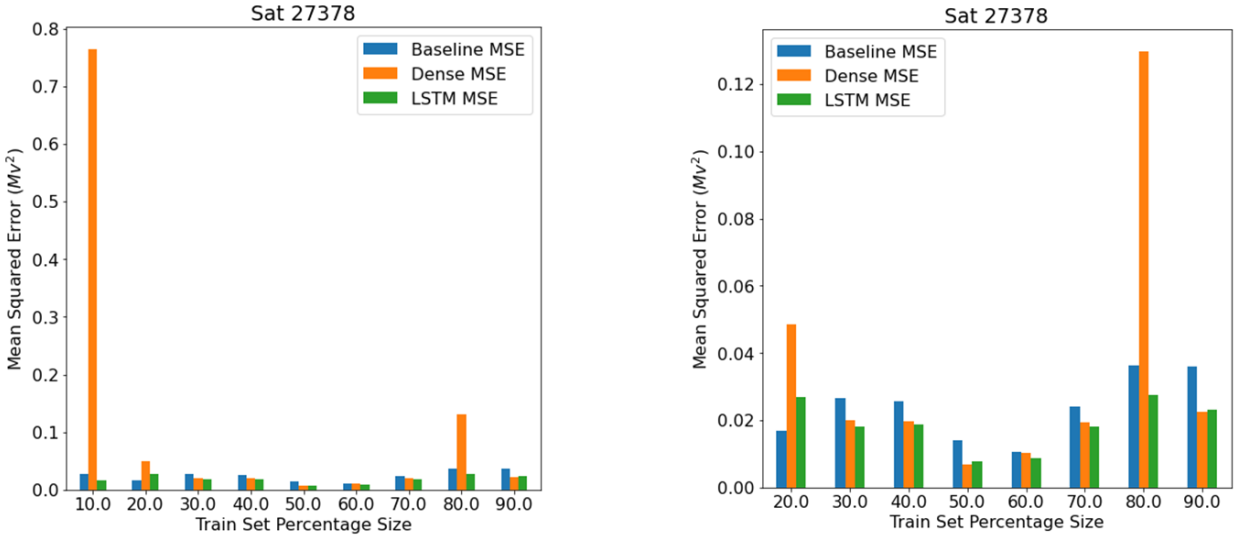

Fig. 12. Cross-validation results comparing the MSE score between the three models of interest. For both figures, the x-axis is the percentage of the cross-validation set used for training, while the y-axis is the MSE for the points found in the next 10% in units of *Mv squared*. The lower the MSE, the better the model has done at predicting future Visual Magnitude values. **Left**: Here we share the results over all 9 cross-validation splits, noting the

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

extremely high score for the DNN (orange). **Right**: A rescaled view of the results, capturing the set splits from 20%-80% to better show the MSE comparisons (baseline in blue, LSTM in green) that are closer in scale.

We can see in Fig. 12 that the most consistently performing model for the EchoStar 7 cross-validation data set is the LSTM network. At times the DNN performs slightly better but appears to struggle with consistency. This variation could also be due to the large gaps in the early parts of the data set, where the baseline and DNN will suffer from larger gaps in data. In the early stages the Neural Networks struggle with outperforming the baseline, but once they have access to a sizeable amount of training, data quickly perform better. This is typical of Deep Learning, reconfirming that these types of models need access to a large amount of training data to perform well. The best performing at predicting single step Visual Magnitude values, and most consistent model, is the LSTM network. Due to this, subsequent experiments will focus on using the LSTM only.

4.3 SEARCHING FOR THE OPTIMAL ROLLING WINDOW PARAMETER

Now that we have narrowed our model choice down to the LSTM, we wish to focus on other parameters that will affect the accuracy of the predictions. Namely, the rolling window size and the number of features used during training. For both cases we use the same 9 subset iteration process (Fig. 11) to test the cross-validation set and see what parameters perform best.

First, we examine the window size parameter. We begin by choosing two sizes by hand, 10 and 50 samples. For larger window sizes (and future work) it will be better to have an automated approach to choosing a window size. Not all satellites and light curve data will have the same density and gap characteristics as EchoStar 7 in this timeframe. Currently our solution is to return the cluster/grouped data from Fig. 2. By analyzing the density and focusing on the number of samples in the majority of clusters we can systematically choose a range of values from that information. In this analysis we found our other sizes to be: 97, 146, 194, 243, and 292. In Fig. 13 we share our cross-validation results for all windows and subsets, except for the final subset with the window size set at 292. The data shapes and sizes in these tests begin to get quite extreme, training on over 400,000 windowed samples in the case of the final subset. Due to this and its large window size the final window test ran into resource complications, so we currently leave its score empty as Not-A-Number (NaN). Despite this, we can see in Fig. 13 as we train with more and more of the cross validation set the MSE becomes less sensitive to the window size chosen over the explored parameter space. Due to the large nature of the dataset we would expect this, as the rolling window may not be as large, but the model has still been exposed to many Visual Magnitude samples.

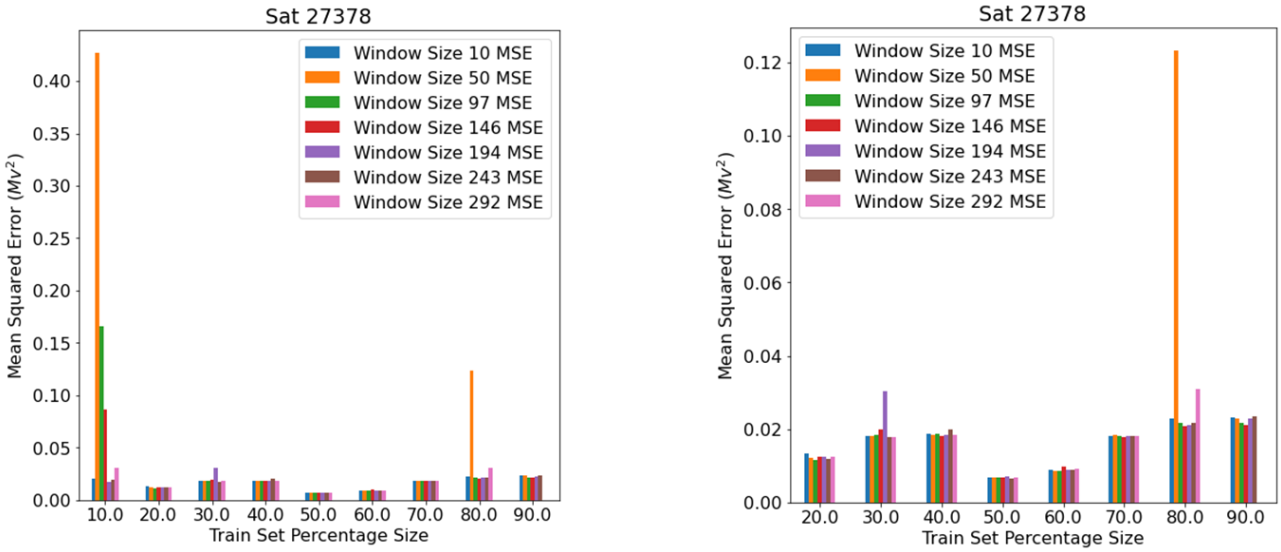

Fig. 13. Cross-validation results comparing the MSE score between different rolling window sizes. For both figures the x-axis is the percentage of the cross-validation set used for training, while the y-axis is the MSE for the points found in the next 10% in units of *Mv squared*. The lower the MSE, the better the model has done at predicting future Visual Magnitude values. **Left**: Here we share the results over all 9 cross-validation splits, where we see the most variation in the first subset of 10% training (first grouping of columns on far left). After this the scores tend to be comparable to each other. The outliers may be due to a window falling directly on a gap in the data. **Right**: A

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

rescaled view of the results, capturing the set splits from 20%-80% to better show the MSE comparisons that are closer in scale.

4.4 FINE TUNING FEATURE SELECTION

The final parameter space we explore is the choice in features used during training of the model. Given the choices from Section 3, we test the cross-validation data against three distinct choices. We will refer to them as *Base Features*, *Base Plus Sensor Features*, and *All Features*. For Base Features we include the four columns: Visual Magnitude, Solar Phase Angle, Solar Declination, and Brightness. In Base Plus Sensor Features we include the previous four but add the one-hot-encoded numerical sensor columns (see Fig. 6). For EchoStar 7's data this will result in a total of 28 features for base-plus-sensor features. The All Features set will include all previous features, with the addition of the cumulative difference columns added per window given towards the end of Section 3 and seen in Fig. 7. This gives a size of 32 features for the full feature set.

We again use the percentage split cross-validation subsets to test each of these three feature sets, training an LSTM network for each and calculating the MSE accuracy on that subset's validation data (the next 10% of data). For the rolling window size parameter, we chose a smaller value to save on computational resources and training time. In an ideal world we would perform a more rigorous cross-validation test comparing all combinations of networks, the differing window sizes, and the feature sets considered. Under current constraints, time, and computational resources, we have opted to test them separately, changing only one parameter at a time. Given this and the closeness in scores shown in Fig. 13, we set the rolling window size to 10. Shown in Fig. 14 we see the results of feature selection. From these results we can see that for the third validation split the Base Plus Sensor Feature set had quite a large error compared to the rest of the trials. This could be for several reasons, but we suspect the sparsity of the one-hot-encoded rows and the gaps in data create a situation for the accuracy to diminish here. After that subset the rest of the cross-validation splits settle to values close to each other.

We can see though that the Base Features do not always perform the best (e.g., Training Set Percentage Size set at 80% of total cross validation set) but performs quite well for the other subsets. Given these results, and the future complication of needing an exhaustive list of sensors used in both training and blind data, we opt to say the best choice is Base Features. There could be a situation where new unseen data has a different sensor configuration, and this would make the Base Plus Feature or All Feature trained model incompatible if those extra features were to be used as model inputs. For our final model used on the blind data we will choose Base Features.

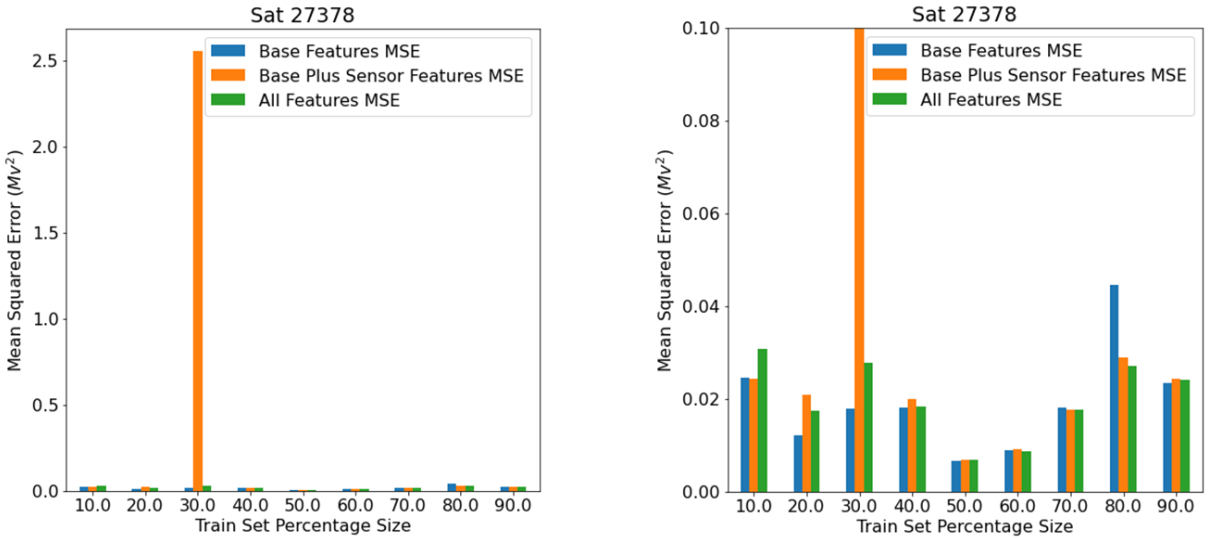

Fig. 14. Cross-validation results comparing the MSE score between different features selected for training. The three options range from Base Features all the way to Full Features which include categorical encoded sensor data and sample difference data. For both figures the x-axis is the percentage of the cross-validation set used for training, while the y-axis is the MSE for the points found in the next 10% in units of *Mv squared*. The lower the MSE the better the model has done at predicting future Visual Magnitude values. **Left**: Here we share the results over all 9 cross-validation splits, where we see the most variation in the third subset of training. After this the scores tend to be closer in accuracy. The outliers (orange spike) may be due to a window falling directly on a gap in the data. **Right**:

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

A rescaled view of the results, lowering the y-axis limits to better show the MSE comparisons that are closer in scale.

4.5 FINAL MODEL AND PERFORMANCE ON BLIND DATA

In the previous sections we have explored several model choices, as well as the LSTM model space defined by the data features and rolling window size. We have found the LSTM model to perform the best, and out of the tested parameters the Base Features give the best and most consistent result. There was no strong indication under current data and window size parameters that any one given window size performed better than the other, so for evaluation on the blind data we will use a size of 100 samples for the rolling window. Given we are only training a single model, and not multiple cross-validation experiments, we can use a slightly larger size as opposed to Section 4.4. We also use the full cross-validation set for training, and this time predict on the unseen blind data. The MSE metric will again be used to benchmark how accurate our predictions are. For the blind data we get an MSE of 0.0158 *Mv squared*, and visual results can be seen in Fig. 15. Remember, even though there are multiple predicted points in Fig. 15, the way the model works is to use the previous 100 features to predict the very next Visual Magnitude value. In a real-world application of this model this would require the data to be at least 100 samples before you could begin making future forecasts.

We can see in Fig. 15 that the LSTM (green line) does a good job following the trend of the light curve (black line). There are portions of the time series that have gaps between nights, as well as behavior in the middle of the "v" trends typical with a satellite getting occluded by the Earth. The behavior is exhibited through the spikes and dips similar to those shown in Fig. 9 (resulting in the triangle shape in Fig. 15 left and the diamond shape in Fig. 15 right). Other sources of noise and anomalies not shown directly are light curve noise due to ground-based sensor ability and out-of-pattern time series from spin instability. Given these challenges, we can see the LSTM is still able to pick up on the light curve trend from its training data. The model is able to follow the light curve PoL closely, taking advantage of previously seen historical behavior. Overall, we are quite pleased with the LSTM performance and the results shown in Fig. 15.

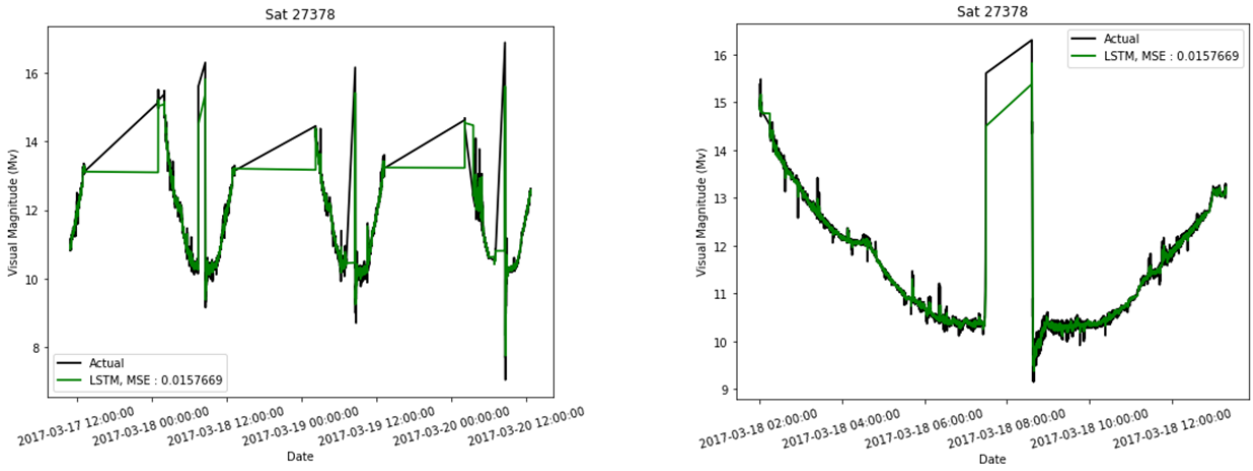

Fig. 15. Final LSTM model performance on our set-aside blind data from EchoStar 7. It achieves an MSE of 0.0158 *Mv squared*. The lower the MSE, the better the model has done at predicting future Visual Magnitude values. Both figures have the timestamp on the x-axis and the Visual Magnitude in *Mv* on the y-axis. **Left**: This is a zoomed-in view of the blind data, showing the first several nights of raw data (black) compared to the LSTM single step prediction (green). **Right**: A rescaled view of the results, zooming in on a single night.

5. CONCLUSIONS

In this analysis we have explored forecasting light curve data using Neural Networks. We were successful in showing that LSTM networks are capable of capturing the cyclic and repeated trends found in nightly light curve data. This was done by training an LSTM model on data from EchoStar 7 and predicting a single sample into the future based on a historical window of ground-based sensor data. Shown above we achieved an accuracy MSE of 0.0158 *Mv squared* while forecasting future Visual Magnitude values. The problem is not trivial, as the data can be

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

locally dense and globally sparse at times. Inclusion of daylight tracking for Visual Magnitude observations may help reduce the sparsity as seen in this current data set. The difficulty is compounded by the non-uniform sampling of observations that make up light curve data. However, we are confident that LSTM networks can improve forecasting these complex time series while outperforming basic models and working around sampling challenges.

While this work has been successful, it is our hope to continue to explore LSTM applications to light curve data. Given the right dataset, we could expand our method to forecast Visual Magnitude values at set time intervals. Following the success here, we also hope to begin benchmarking this method against a variety of satellites we can access on the Unified Data Library (UDL) [12] to produce light curve anomaly analysis. Future scenarios of interest include identifying measurements as mis-tagged satellites and irregular light curves caused by changes in rotational state. In both cases, the light curve can exhibit deviations from previously exhibited patterns and comparing to an accurate forecast may distinguish these anomalies.

6. FUNDING

The work shared here was supported by AFWERX, Air Force Research Laboratory (Space Vehicles Directorate), and Space and Missile Center.

7. REFERENCES

[1] Rumelhart, David E., Hinton, Geoffrey E., and Williams, R. J., "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation*. MIT Press, 1986.
[2] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. *Neural Computation* 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735
[3] C. Shabarekh, J. Kent-Bryant, G. Keselman, and A. Mitidis, “A Novel Method for Satellite Maneuver Prediction.”
[4] T. Gemmer and C. Shabarekh, “Leveraging Non-Traditional Sources (NTS) for Space Situational Awareness (SSA) Analytics.”
[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[6] Alice Zheng and Amanda Casari. 2018. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists (1st. ed.)*. O'Reilly Media, Inc.
[7] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781
[8] Mikolov, Tomas (2013). "Distributed representations of words and phrases and their compositionality". *Advances in Neural Information Processing Systems*. arXiv:1310.4546
[9] Scikit-learn: Machine Learning in Python, Pedregosa et al., *JMLR* 12, pp. 2825-2830, 2011.
[10] Chollet, F., & others. (2015). Keras. https://keras.io.
[11] Diederik P. Kingma and Jimmy Lei Ba. “Adam : A method for stochastic optimization”. 2014. arXiv:1412.6980v9
[12] Unified Data Library. https://unifieddatalibrary.com

Copyright © 2021 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com