Machine Learning for Satellite Characterisation

Alexander Agathanggelou, Ryan Houghton, Joshua Collyer, Joshua Davis, Nicholas Pallecaros
Defence Science and Technology Laboratory (Dstl), UK

ABSTRACT

The space domain is increasingly congested. The number of satellite launches is growing rapidly; commercialisation of Space Surveillance and Tracking (SST) services and the number of space-capable nations continues to rise; and on-orbit collisions between active satellites and other objects is an ever present and growing risk. In tandem, the amount of observational and technical data available on Earth satellites is growing, delivering a more complex but potentially more veracious view of the situation. These two factors – increasing domain congestion and an overload of data, presents a significant challenge to contemporary orbital analysts attempting to maintain the standards of space domain awareness and services that were more achievable in past decades. However, it is shown that Machine Learning (ML) techniques can potentially offer assistance to analysts in managing these burdens: ML can leverage high volumes of data to provide timesaving, wider-reaching, or novel capabilities, and release human effort to focus on complex or higher-priority issues.

This paper demonstrates the feasibility and effectiveness of machine learning models for satellite characterisation. Dstl have developed models that successfully predict the operational status of satellites by measuring fluctuations in reflected sunlight while they orbit overhead at night, or similarly fluctuations in reflected radar pulses. We achieved a best achieved accuracy of around 93% for LEO, and 92% for GEO, approximately matching the estimated accuracy of the "truth" labels used to train and test the models. Notably, the accuracy of the ML assessment only dropped by a few percent when presented with a single track (observed overhead pass) rather than the entire dataset, demonstrating the utility of even a single light curve collection when using these characterisation techniques. We also developed machine learning models to identify the bus type of a satellite from the same data (with an accuracy of >98%).

The best results in both cases were achieved when the model was provided with additional contextual data, such as the launch date of the satellite, particularly improving the estimation of operational status in LEO from 86% to 93%. We found classical ML models (e.g. Random Forests) performed just as well as Deep Learning approaches: our LSTM (long short-term memory) models struggled to retain high performance when generalising to non-iid (independent identically distributed), out-of-distribution samples. The full range of characterisation tasks tackled by analysts is significantly broader than those investigated here, thus considerable scope remains for further research.

1. INTRODUCTION

Space is a domain that is rapidly evolving, becoming increasingly contested and congested whilst the UK amongst other developed nations become ever more reliant upon space based services and capabilities. One of the most pressing requirements in order to understand the space environment and threats to nations' highly valuable assets (HVAs) on orbit is the identification and characterisation of objects across a range of orbital regimes in order to assess the threat that they pose. These characteristics can include, but are not limited to:

- Size;
- Shape;
- Bus type;
- Attitude; and
- Operational Status.

An understanding of these characteristics (amongst others) allow the classification of objects observed in orbit, provide explanations for observed behaviour, and could even implicate the intent of observed actions.

Characterisation of objects in space is becoming an ever greater challenge due to the increasing congestion in Earth orbit (see Fig. 1); this growth is expected to accelerate rapidly over the next 10 years due to the advent of "megaconstellations" containing hundreds to thousands of satellites, such as OneWeb or Starlink. Compounding with this growth in satellite numbers, the variety, complexity and availability of datasets used for Space Domain Awareness (SDA) efforts continues to grow. As is, an orbital analyst will manually review and assess available datasets relevant to a particular object before making judgement and recommendations. Unsurprisingly, the burden on analysts attempting to carry out characterisation for SDA purposes on satellites is moving beyond what is feasible. Currently, this demand is mitigated by prioritising those satellites that are considered of highest interest. However, this approach will likely struggle with the expected growth of the orbital population. Automation of characterisation processes using machine learning based models is an attractive alternative mitigation strategy, which could assist and augment the capabilities of orbital analysts.

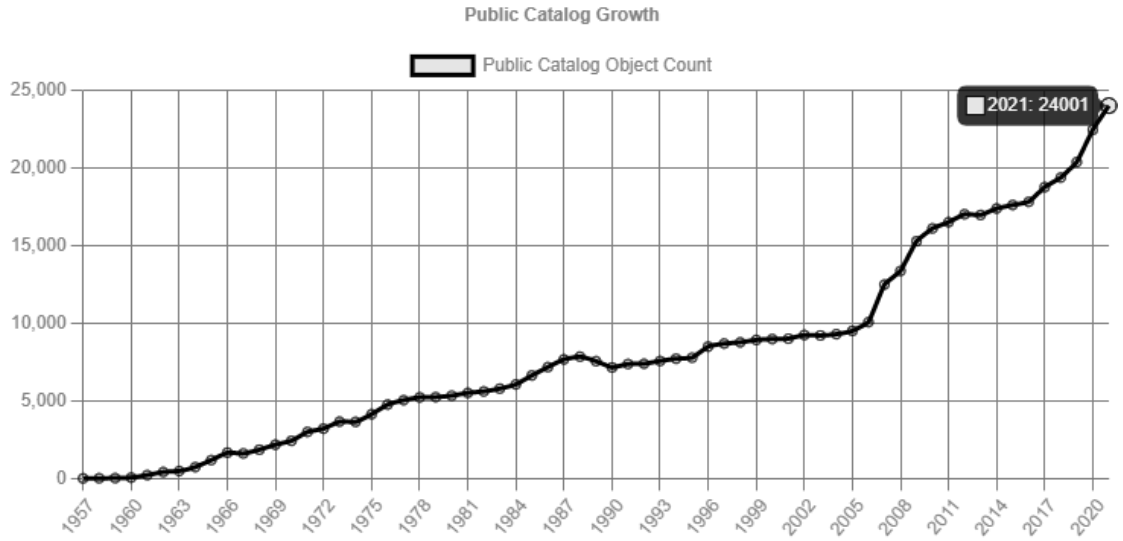

Fig. 1. Graph showing the growth of Resident Space Objects (RSOs) listed in the U S Space-Track public TLE catalogue from the start of the space age in 1957 to present (source: Space-Track.org accessed Oct 2021)

Artificial Intelligence (AI), particularly Machine Learning (ML), may be able to assist orbital analysts in tackling characterisation challenges. Certainly, the increasing volume and veracity of available space domain data gives ML methods a distinct advantage, which only improve in capability with additional (good quality – i.e. diverse, unbiased and broad) datasets. The feasibility, strengths and weaknesses of this approach needs to be determined, and the way that analysts can make best use of these tools needs to be understood; which sets the context for this work.

2. AIMS AND ASSUMPTIONS

The aim of this work is to assess the current feasibility of ML for satellite characterisation. That includes understanding the data and models that can be used, and their strengths, weaknesses and limitations; but it also includes understanding the sensors and infrastructure requirements. Our focus is entirely on ML algorithms; other novel methods that could be applied to satellite characterisation (e.g. advanced Bayesian statistical methods, or genetic algorithms) however viable, are not the topic of this work. However, techniques that can help enhance the inputs for ML models were considered (falling under *feature engineering*, see section 6).

The use of ML for satellite characterisation requires certain assumptions to be met. These include the following:

- *Sufficient data exists and is available for ML methods*;
- *Labels exist to support supervised ML techniques*; and
- *Resources exist to facilitate the application of ML.*

While these assumptions were found to be mostly valid over the course of this work, they were also all broken to a certain degree. Curation of training and testing data was able to demonstrate that some ML models were overfitting

and not generalising, due to a lack of diversity in the input data. Labels for operational status from different sources did not always agree, and were only accurate to 93% (LEO) and 98% (GEO).

Throughout this paper we deliberately use the term LEO loosely, to refer to any satellite *data* observed at a distance of less than 6000km. This maximises the amount of data available for 'low' orbits. We use the term GEO to refer to satellites with distances of around 36000km, including those not actively station-keeping and those in graveyard orbits. We use the term *track* to refer to all the time-series measurements for a single observed pass overhead of a given satellite.

3. DATA SOURCES

ML can be used with a wide range of different data formats. For this work, we focussed on using electro-optical (EO) light curves and radar cross-section (RCS) curves as inputs to the ML models. Each comes with different advantages and disadvantages, summarised in Table 1 below:

Table 1. Advantages and disadvantages of EO and Radar sensors respectively.

EO Sensors
Advantages: • Can observe satellites in most orbital regimes from LEO to GEO • Can perform multi-band or spectral observations for surface material estimation • Multiple commercial suppliers available
Disadvantages: • Requires clear skies and night time to make observations • Different sensors types better suited to different LEO and GEO regimes
Radar Sensors
Advantages: • Can operate at day or night and almost any weather • Can provide range and Doppler measurements in addition to RCS
Disadvantages: • Limited primarily to LEO regimes • Limited commercial suppliers, who do not currently provide data of sufficient quality for ML characterisation

We found that not all sources of EO data were appropriate for all orbital regimes (see Fig. 2), and similarly not all sources of RCS curves were useful for characterisation with ML.

Fig. 2. An illustration of the different light curves available from WFLAs and traditional telescope networks. Top row shows light curves for a LEO satellite, bottom row shows light curves for a GEO satellite. Left column shows light curves observed by MMT (a WFLA), right column shows light curves observed by Numerica (a traditional telescope network). For satellites in LEO, WFLAs provide long well sampled light curves ideal for characterisation; conversely for GEO, world-wide networks of traditional telescopes are sensitive enough to capture the long faint light curves needed for characterisation.

Historically, EO sensors for SDA have focussed on observations of satellites using a relatively narrow Field of View (FOV) telescope (in the order of a few degrees width) distributed in worldwide networks. These systems require a large amount of *a priori* information on the satellite orbits for tasking, rather than offering a true survey capability. However, recent progress in the astrophysics of transient objects (exoplanets and other variable objects) have led to development of low-cost Wide Field Lens Arrays (WFLAs) which can observe a much larger FOV using multiple lenses.

WFLAs use rapid exposure times to capture sudden changes and, with wide FOVs, remove the need for significant advanced planning for scheduling, tasking and collection. While many WFLA telescopes treat satellites as a nuisance, some such as the MMT (MiniMegaTORTORA) actively observe, record and publish[1] satellite light curves (in this case, excluding all Russian satellites) – providing an open source dataset useful for validating the technology for satellite characterisation.

Radar has several advantages over optical sensors. Firstly, it is an active system that can not only detect and track targets but can also determine target range and line-of-sight velocity. Secondly, it can operate with uncooperative targets. Thirdly, it can operate day and night, through most weather conditions.

The most significant challenge in using radar for SDA is the loss in sensitivity associated with the long distances involved. To increase the sensitivity one must employ a larger aperture or reduce the system temperature. All of these come with increased costs and engineering challenges. Consequently, most SDA radars are limited to LEO although there are notable exceptions e.g. MIT's Millstone Hill Radar.

A dominant branch of ML is supervised learning which requires both input data such as light-curves, and output labels to learn from. In the case of satellite characterisation, the labels refer to a *ground-truth*, which we wish to infer from the input data using the trained ML model. For example, in the case of satellite characterisation, the labels could provide the operational status of a satellite for the model to learn from. Standard supervised ML requires the

[1] http://mmt.favor2.info/satellites (accessed August 2022)

accuracy of labels to exceed the desired accuracy of the model, and labels that are not 100% accurate can limit performance.

We found three suitable sources of satellite information for use as labels:

- Seradata[2]: a commercial entity which deals mostly with sales of satellite information to various entities (e.g. insurance companies). Access to their SpaceTrak database is available through a paid subscription, and the data available is considerable;
- Celestrak[3]: an open-source website provides a limited amount of information on all satellites via its SATCAT, which includes *current* operational status as well as orbital details and owner/operator information; and
- MMT[4]: the website for the MiniMegaTORTORA observatory provides a *Type* keyword for each satellite which relates to whether the satellite is debris, a rocket body, an active satellite, or an inactive satellite.

With two apparently independent sources of labels (Seradata and Celestrak) we can compare them to assess their overall accuracy, where they overlap. This is the case for operational status, but not for bus type (which is only present in Seradata). There are considerable differences between the two sources on the operational status of satellites. Fig. 3 compares the status labels directly. Overall there is an 80% agreement, but this considers satellites that exist in one catalogue but not in the other as an inaccurate label; as such it is likely a lower limit on the accuracy of public operational status labels. We can also compare the labels as though predicting one from the other: this approach suggests the operational status label for LEO satellites is only 93% accurate, while that of GEO satellites is 98% accurate. These are likely upper limits on the overall accuracy of the operational status label and the relatively low accuracy for LEO satellites is concerning as it will likely constrain the performance of standard supervised ML models.

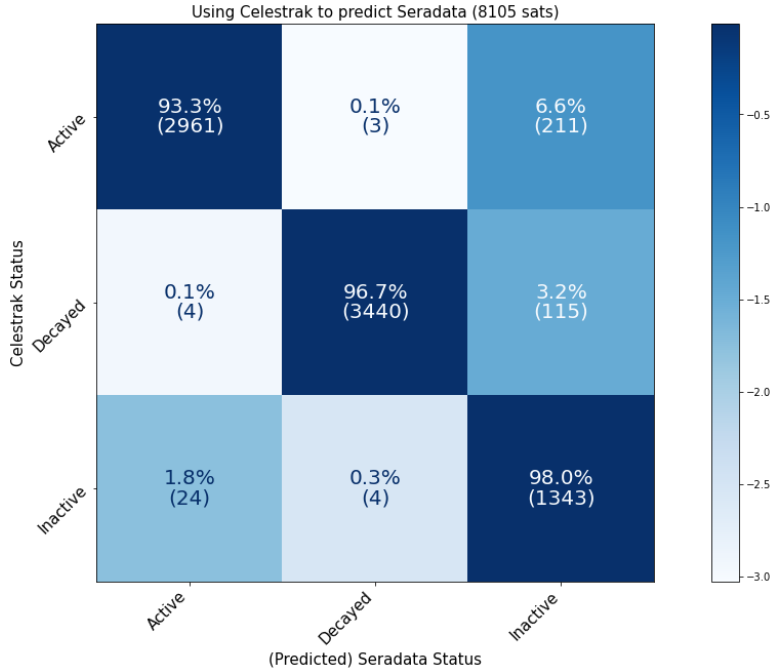

Fig. 3. A 2D histogram comparing operational status labels from CelesTrak and Seradata, excluding those labelled as unknown by CelesTrak (the unknown label is not present in Seradata), and excluding satellites only listed in one of the two data sources. The overall agreement between these label sources is 95% on average, specifically matching for 93% of LEO satellites and 98% of GEO satellites.

[2] https://www.seradata.com/
[3] https://celestrak.com/
[4] http://mmt.favor2.info/satellites

4. MACHINE LEARNING ENVIRONMENT

Machine Learning is more than just supervised and unsupervised learning. We define four stages in the process of developing ML solutions: *Exploratory Data Analysis (EDA)*; *Data Wrangling*; *ML Model Choice;* and finally *Verification and Validation*. We emphasise that this process is iterative rather than linear: exploring new ML models may require revisiting the EDA and Data Wrangling stages.

EDA, also known as "data exploration", is an important stage in development of an ML model, in order to ascertain how best to use the available data. Data from different sensors may be appropriate for different tasks, as discussed in Section 3. Furthermore, fusing data from different sources can lead to significant improvement in the performance of a model. Identifying which sensor or data combinations are most useful for a task is as important as finding which ML model performs the best: often the choice of data has far more impact on the final result than the model. But data exploration should only have to be done once, at the research stage, so long as a suitable automated pipeline is constructed and the data sources do not change.

Data Wrangling is the process of pre-processing the chosen datasets ahead of use in a ML model. This may include cleaning the data (of bad values), fusing and homogenising the data (from different sensors, catalogues or just modes of operation), restructuring the data (perhaps interpolating onto a regular interval) or reducing the data (by removing uninformative columns or extracting higher order information). Data wrangling commonly involves *feature engineering* which helps the ML model quickly identify the most relevant information in the data, but may also include methods to *augment* or expand the dataset (particularly for Deep Learning methods). Again, data wrangling should only need to be done once to develop an automated pre-processing pipeline for future data (so long as the data format or properties do not change).

ML Model Choice is the next key step in developing a ML solution. Exploring different ML methods such as supervised, unsupervised or reinforcement learning is all too often seen as the main decision relating to a choice of ML model; it also includes choices between using classical ML and deep learning approaches, and the associated requirements each brings (different data wrangling procedures, different computer hardware). Table 2 compares aspects of classical and deep learning ML models. In this work we foresaw classical ML as a means to develop quick simple *baseline* models with which to compare more complex deep learning models.

Table 2. Comparison of classical and deep learning ML models

Classical	Deep Learning

• Useful when data is limited
• Humans engineer features
• More easily explainable
• Computationally cheap
• Knowledge fixed
• Performance limited
• Includes:
  ○ K Nearest Neighbour
  ○ Decision Tree
  ○ Naive Bayes
  ○ Support Vector Machines
  ○ Random Forest
	• Useful for big data
	• System can learn features given enough data
	• Difficult to explain model outcomes
	• Computationally expensive (during training)
	• Knowledge transferrable
	• Performance scales with data volumes and model complexity
	• Includes: (all neural networks)
	  ○ Convolutional Neural Networks (CNN)
	  ○ Recurrent Neural Network (RNN)
	  ○ Long-Short Term Memory (LSTM)
	  ○ Generative Adversarial Network (GAN)
	  ○ Auto-encoders
	  ○ Attention and Transformer architectures

Verification and Validation is important to investigate the robustness and generalisation of ML models to new circumstances and data. Most traditional modelling approaches (e.g. representing the system using mathematical equations that describe the underlying physical mechanisms) fail to return good results unless the model has been

fitted correctly. However, ML models can return convincing results when an error has been made in their development that has led to data leakage or overfitting.

Validating ML models for robustness and generalisation is not trivial, but one approach is to maintain a *hold-out* set of data that the model has not used before (during training, hyper-parameter tuning[5] or testing). This *hold-out* set can contain all the examples of a particular group, to test how well the model *generalises* to new circumstances it has not seen before. Here, we can hold back all the light curves of certain satellites, or perhaps of a specific bus type. Comparing the formal test scores with the scores achieved on the *hold-out* set provides evidence for how the model will behave on new, unseen cases.

5. MACHINE LEARNING MODELS

We investigated a number of different ML models in this work. Two consistently performed well, *Random Forests* and *LSTM Neural Networks*.

The *Random Forest (RF)* is a classical ML model based around the Decision Tree (DT) algorithm. A DT is a simple algorithm that learns which decision rules can be applied to the inputs to best achieve the desired output (label). For example, a DT may quickly learn that if the maximum absolute elevation angle of a GEO satellite is much larger than a certain limit, the satellite is likely to be inactive. DTs are also one of the most easily explainable ML algorithms available as they can be represented by flow charts.

RFs combine many different DTs together by random sampling (with replacement) of both the training data and the input features of the data, for each new DT. This helps reduce the tendency of DTs to over-fit the training data. While we did investigate using DTs, their performance was always inferior to the more complex RFs. While RFs are in principle explainable (as they are made from a finite number of DTs), due to the large number of DTs that can be used (1000 is not uncommon), this is not always practical. As testament to the robustness of RFs and DTs, when validating the trained models using *hold-out* sets containing all the tracks of a given set of satellites, the scores on the test and *hold-out* sets were typically very similar. RFs are likely to be more robust when the training data is limited and contains poor diversity.

Long Short-Term Memory (LSTM) neural networks are a type of artificial neural network (ANN) in the form of a Recurrent Neural Network (RNN). ANNs have dominated the current revolution in ML. Inspired from biological neurons, they have inputs and outputs and are stacked in multiple layers. Each neuron is represented by a simple function operating on its input, which can be tuned (i.e. trained) to provide a desired output. While the response of individual neurons is very simple, even linear, extremely complex non-linear responses can be generated by stacking and training many layers together.

RNNs were developed specifically to model sequences of data, such as a time-series or words in sentences. LSTMs were developed to help solve a specific problem in RNNs (instability in the gradients of the neuron functions during training, causing them to vanish to zero or explode to infinity). Arguably, they have been superseded by Attention and Transformer networks, but only when sufficient volumes of rich and varied data exists to train such complex and data-hungry model architectures. Adopting overly complex ANN models with insufficient data can easily lead to over-fitting and lack of generalisation of the model to new circumstances[6].

[5] Many ML models need their numerous free parameters optimising in a procedure called *hyper-parameter tuning*. This is done using a *validation* set of data that has neither been trained on, not tested on. However, this stage is not to be confused with model validation, or model verification: hyper-parameter tuning is just another stage in the model fitting/training procedure.

[6] *Transfer learning* (training a model on a different larger dataset before partially re-training it (top layers only) using the limited data available) can help alleviate this problem.

6. FEATURE ENGINEERING

The process by which data is passed to an ML model can directly influence how the model performs. The type of model (classical ML or Deep Learning) also influences our choices.

Classical ML benefits from human expertise in curating or *engineering* key features from the data: this makes the data manageable, and helps accelerate the learning process by using human expertise to extract key understanding.

Deep Learning can also benefit from human expertise: data may be augmented in ways that the human expects operational data to vary (e.g. more noise added, or generating plausible variations of existing data). Feature engineering for Deep Learning models invariably expands the data by making different views and combinations of it to help generalise the model to different circumstances and avoid overfitting.

Input choices can be a simple way to engineer features in the datasets. In the case of satellite light-curves, we can pass the model information from a single track (light-curve from a single overhead pass), an ordered sequence of tracks, or even all the tracks available for a given satellite. We might expect the model to perform the best when given the most data (i.e. all tracks), but this is not always true. It may be necessary to summarise or reduce the data to make it manageable for the model; even a Deep Learning model requires a pre-determined input data size, which can lead to resampling or cropping. Summarising the data from all available tracks might lead to key information being glossed over and omitted, or it might provide more context and representation. In terms of operational use, it may be more practical for the model to ingest fewer tracks for a more rapid and up-to-date outcome. In this work we compared models that ingested single, multiple sequenced tracks and (summaries of) all tracks.

A common technique to reduce data size is to extract *simple statistical features* from each input variable, such as the mean, standard deviation, minimum and maximum. While this may appear overly simple, it can often reveal key insights. For example, the maximum and minimum elevation coordinate of a GEO satellite can establish if the satellite is station-keeping or not, which is correlated with the operational status of the satellite.

Light-curves often exhibit periodicity (see Fig. 2). This can be due to a number of causes (rotating payloads, spin stabilising methods) but can also indicate if the satellite is tumbling (after collision or simply loss of attitude control). As such, detecting periodicity may be a key feature for ML models with regard to predicting activity status.

Most light curve and RCS data is not uniformly sampled with a regular time step. This hinders the use of traditional Fast Fourier Transform (FFT) algorithms to identify signal power at isolated frequencies. The astronomical community has already tackled this problem however, in light curves from distant stars during planet hunting studies. Traditionally, the Lomb-Scargle Periodogram has been used to detect signals at fixed frequencies when the data is irregularly sampled, but this has numerous drawbacks. On reviewing scientific progress in this area and setting up an evaluation experiment of a range of frequency detection algorithms, we found that the more recent work on Quadratic Mutual Information/Entropy[7] provided far cleaner Periodograms leading to the more robust identification of fixed frequency signals.

While FFTs and Periodograms reduce and summarise the data, we also investigated methods to characterise the periodicity of a light curve using a rolling window approach. While potentially viable, we did not amend the Periodogram techniques above to operate on overlapping windows (as done in the Short-Time Fourier Transform (STFT) spectrogram). Instead, we investigated the use of Peak Prominence algorithms[8] to identify all the main peak maxima and minima in the light curve and then calculate simple statistics from these points, such as the average time between the significant peak values and associated standard deviation. This feature engineering approach can operate in a rolling manner to allow (Deep Learning) models to trigger if just part of a light curve was periodic.

Many software libraries exist to automatically generate input features for machine learning; this is especially true for time-series modelling[9]. For LSTM models we not only split light-curves into regular sized blocks and sampled

[7] Made available in the P4J Python package; see academic references therein (https://github.com/phuijse/P4J).
[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.peak_prominences.html
[9] See FATS (http://isadoranun.github.io/tsfeat/), TSFresh (https://tsfresh.readthedocs.io/en/latest/) or TSFEL (https://tsfel.readthedocs.io/en/latest/).

multiple blocks (with repetition/overlap) from each light curve, but we also ran a suit of standard time-series features over those blocks, providing over 70 features for every input.

Windowed Features (also known as lag features) are used heavily within the finance sector to provide indications of change over a given time period. They can come in various forms such as expanding windows (starting at a fixed point in time and growing thereafter) or a rolling/sliding windows (which is a fixed size and moves across the data through time, calculating a value for each new position). They are primarily useful for deep learning models, particularly LSTMs, which ingest the windowed features in parallel to the raw sequences of measurements. One example of a windowed feature is the *seven day average* used to report the number of cases and hospitalisations during the COVID-19 pandemic: the average is more robust to day-to-day variations and provides a more reliable view of the current situation than the latest available daily measurement does.

7. SUMMARY MACHINE LEARNING RESULTS

We now present the overall results on how different ML models (and inputs) performed on two satellite characterisation tasks: **predicting operational status** (summarised in Table 3) and **predicting bus type** (summarised in Table 4); primarily using light-curve data.

Table 3. Summary of results for predicting operational status of satellites (excluding any context). When considering these accuracy scores it is important to note that, given this is a 2-class problem (two possible answers: active/inactive) a 50% accuracy score would be achieved through purely random guesses. Additionally, recall that training labels are likely to be at most 93% accurate in the LEO case and 98% in the GEO case (see Section 3).

ID	Model	Data	Features	Tracks	Regime	Accuracy
1	RF	MMT	Simple Statistical	All	LEO	86 ± 5%
2	RF	MMT	Simple Statistical	Single	LEO	83 ± 1%
3	RF	MMT	Simple Statistical	Five-track Sequence	LEO	84 ± 4%
4	LSTM	MMT	Simple Statistical	Five-track Sequence	LEO	84 ± 1 %
5	RF	Numerica	Simple Statistical	All	GEO	92 ± 5%
6	RF	Numerica	Simple Statistical	Single	GEO	90 ± 1%
7	LSTM	MMT	Raw and Windowed	Multiple	LEO	70%

Table 4. Summary of results when predicting the bus types of satellites (excluding any context). When considering these accuracy scores it is important to note that, given this is an 8-class problem (the eight most common satellite buses in our datasets) purely random guesses would only achieve an accuracy score of 12.5%.

ID	Model	Data	Features	Tracks	Regime	Accuracy
8	RF	MMT	Simple Statistical	Single	LEO	78 ± 3 %
9	RF	MMT	Simple Statistical	Five-track Sequence	LEO	80 ± 3%
10	LSTM	MMT	Simple Statistical	Five-track Sequence	LEO	82 ± 2 %

For all results presented in Tables Table 3 and Table 4 we use a technique called cross-validation to generate the final accuracy scores. This technique groups the data into folds (typically 5) and we use a subset of these folds for training (typically 80%) and the rest for testing (typically 20%); we iterate these choices such that we have a test score for every fold, using a model that was not trained on that fold. We then average those scores and provide the standard deviation as the formal error quoted alongside the mean. As an additional constraint, we isolate all tracks from each satellite into single folds. Thus, as discussed in Section 4, we test not only on tracks that have not been used in testing, but using sets of tracks from entirely unseen satellites that have never been used in training. This is more stringent than the typical constraint of not testing on any tracks that have been seen in the training process (i.e. different light-curves from the same satellite would be in both the training and test datasets). As such, our test scores are what we would expect from observations of a new satellite that has not been seen before by the model: this gives us confidence that the scores would be replicated in operational environments, so long as the data sources

are the same or better. Notably, this process highlighted an example of one of our models cheating through "shortcut learning", where an initial test score of 96% dropped to 70% when tested using a *hold-out* set (ID 7 in Table 3).

8. PREDICTING SATELLITE OPERATIONAL STATUS

An important characterisation task in SDA is the assessment of the operational status of a satellite. This provides an understanding of whether an object is an active spacecraft or a piece of debris, whether a satellite has been affected or damaged by an on-orbit event such as a conjunction, or whether a satellite is capable of carrying out orbital changes or manoeuvres as part of standard operations, station-keeping, or to avoid collisions.

We present a summary of the results in Table 3. We used MMT data for LEO models and Numerica data for GEO models. For most models, we considered combining all tracks before extracting and passing features to the model, just extracting features from single tracks, and extracting features from single tracks within a consecutive sequence. Section 3 discussed the various label sources for this work, but recall that labels are likely to be at most 93% accurate in the LEO case and 98% in the GEO case.

As input features, we mostly adopted simple statistics on the various data columns (each varying with time) including magnitude, coordinates, phase angles, distance (derived from matched TLEs by MMT), and any uncertainties. In addition, for the MMT data we included a periodicity which was either zero (no periodicity detected), or the frequency detected from the magnitude data.

When combining all tracks for each satellite in the MMT data, we only had 803 satellites to train and test on. Fortunately, there are roughly equal numbers of active and inactive types (355 and 448, respectively), so we were able to use them all. When using single tracks, we had over 40000 to choose from and ended up sampling down to 10000 (5000 active and 5000 inactive). When using just a single track for prediction, we see a small drop in overall performance, but only by a few percent in each case (LEO and GEO). This is encouraging and suggests useful information can be extracted by a ML model from just a single light-curve. Fig. 4 shows the confusion matrices for some of these models.

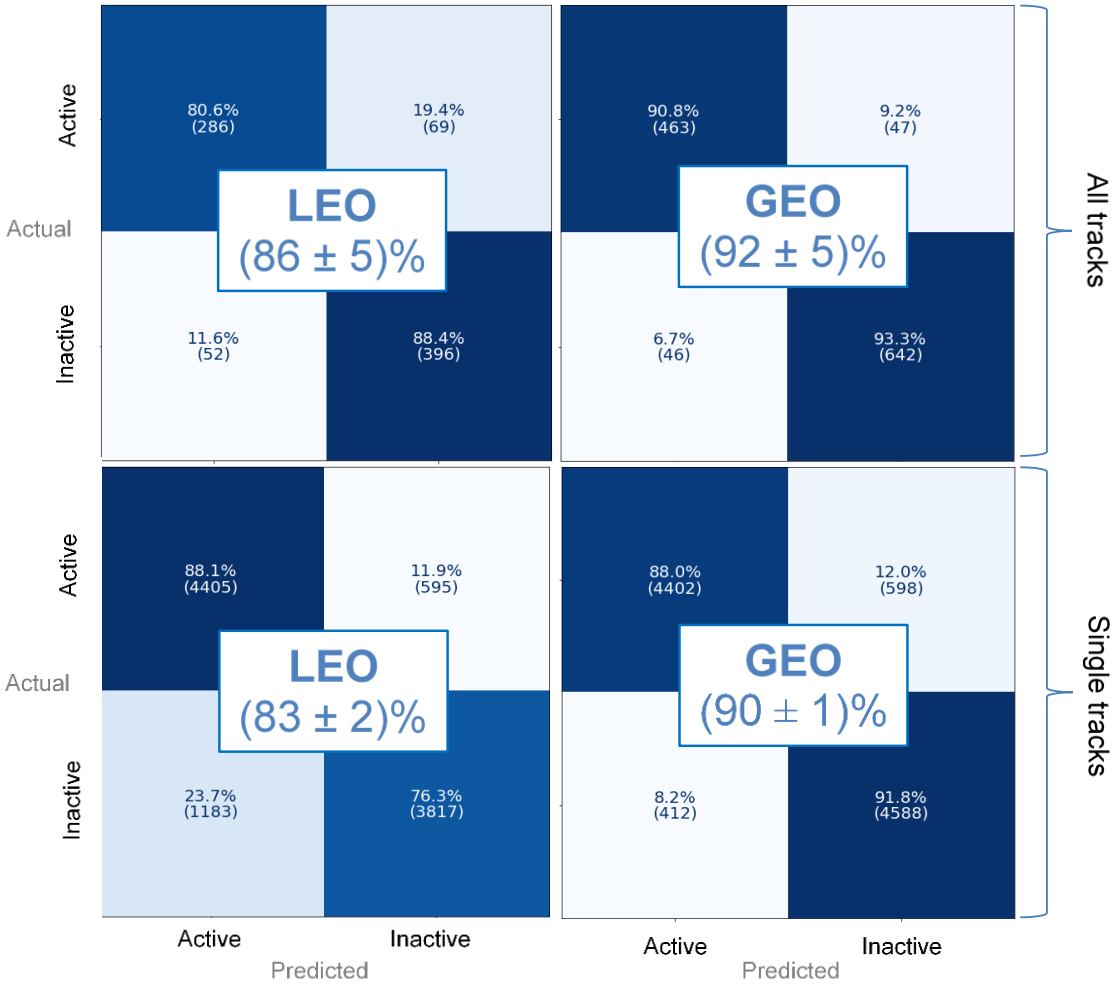

Fig. 4. Confusion matrices for prediction of the operational status of LEO (left) and GEO (right) satellites. Top row: models using simple statistics from all tracks for a given satellite (ID 1 and 5 in Table 3), Bottom row: models using simple statistics on single tracks (ID 2 and 6 in Table 3).

For most cases, we used variations of simple statistics with RF and LSTM models. However, we also developed a much more complex LSTM model to ingest raw time-sequence data such as magnitude as a function of time. Using MMT data, we chose a standard input length of 100 values (10 seconds), randomly sampling 5 of these from across the track and repeating this for a further 5 randomly sampled tracks for each satellite. We then augmented each column with a range of time-series (windowed) features (see Section 6). Consequently, the input for this LSTM was a heavily feature-augmented random sample of track sections.

We hoped that the step-change in complexity of the inputs, together with a complex deep learning model, would provide a step change in the performance. Indeed, when not initially using cross-validation, we found very high test accuracies (above 95%), with some dependency on the complexity of the model. Tracks being used for testing were not being used for training, so this appeared very exciting at first. But when we introduced cross-validation, such that the tracks in the test set were from satellites that the model had never seen before (a *hold-out* set), the scores dropped to 70% (ID 7 in Table 3). The previous high scores were only achieved by the model *cheating* and exploiting a form of data leakage (see Section 4).

Identifying exactly how an LSTM is cheating is difficult, as deep learning models are not easily explainable. The most plausible explanation is an imbalance in the dataset: if all examples of some bus types were mainly active or inactive, the model could learn to falsely link features associated with the bus type to the operational status label; alternatively perhaps the model was learning to memorise the distances of each satellite and associate a label based

on it. We investigated several different LSTM model architectures from single layer LSTMs to multiple bi-directional LSTMs with batch normalisation and dropout. They all performed at a similar level, and were all marred by “shortcut learning”, an example of data leakage.

9. PREDICTING SATELLITE BUS TYPE

An important aspect of characterisation is detecting the bus types of satellites. This provides understanding on what range of functions and payloads the satellite is capable of deploying. Unlike the operational status label, we believe the bus type labels to be highly accurate (>98%) as our models have achieved this level of performance. But we have no way of estimating the actual label accuracy as we only have one source available (see Section 3).

We use the MMT dataset and Seradata labels for this task. To ensure we have enough examples for each bus type, we limit the dataset to only those bus types that have more than 1000 tracks, which gives a total of **eight bus types**. Given the imbalance in the number of tracks for each bus type (we have 15419 tracks for the SSL-400 Globalstar but only 1047 tracks for the Elitebus Iridium Next) we further down-sample each bus type to a maximum of 1000 tracks. As before we use cross-validation that maintains groups of unique satellites in each fold, meaning the test scores presented here were derived from satellites the model had not seen in training (see Section 4).

Note that the previous operational status task was a 2-class problem meaning random guessing would achieve a 50% accuracy; here we have eight classes, meaning random guessing achieves a 12.5% accuracy. As such, even a 50% accuracy in this 8-class problem is providing useful information.

We use both simple statistics from single tracks and on each track in a five-track sequence. Results are broadly similar for all models and features; several LSTM architectures were explored, but a two-layer bidirectional LSTM with batch normalisation performed marginally better than the rest. We present a summary of results in Table 4, and a confusion matrix for model ID 9 in Fig. 5.

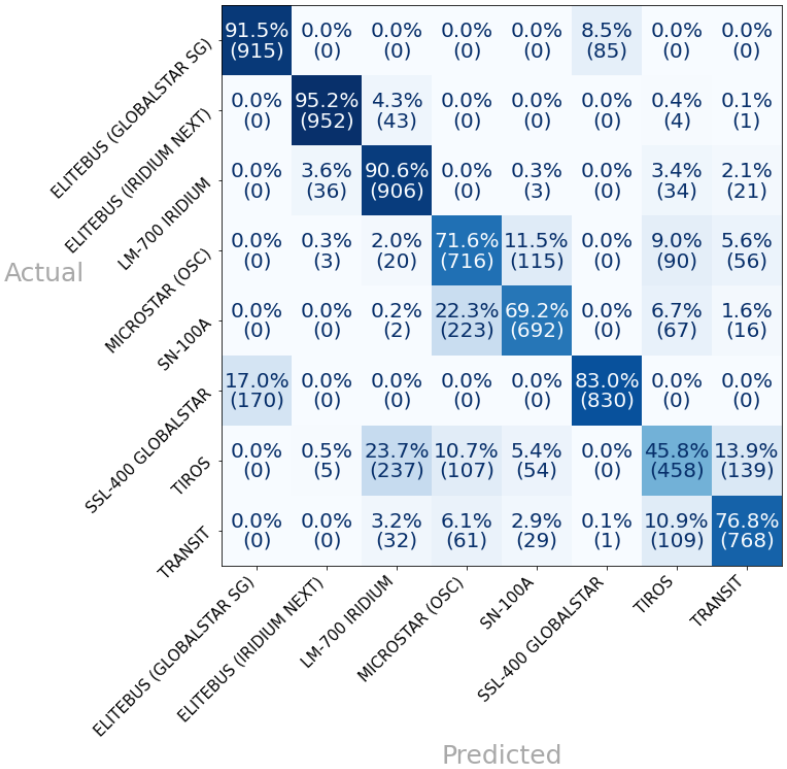

Fig. 5. Confusion matrix for a random forest (model ID 9) predicting the bus type of a satellite.

10. CONTEXTUAL DATA AND ADDITIONAL MODELLING

By chance, when constructing a model to predict the bus type we accidentally left the NORAD number in as an input feature; to our surprise it became the most important feature and improved the model accuracy by more than 10%. Clearly, if we were allowing models to train and test on tracks from the same satellites, then a model could (in principle) learn all the NORAD numbers for each bus type and use this to *cheat* and successfully predict the outputs (this would be an obvious example of data leakage). But we are deliberately *not* training and testing models on tracks from the same satellites (see Section 4), preventing exactly this type of data leakage from occurring.

NORAD numbers are assigned to satellites sequentially over time and so providing it to the model equates to providing information on the relative launch dates of each satellite. Thus similar bus types may have similar NORAD numbers if they were launched at similar times and this information could easily provide an advantage to a model. To test this proposition, we replaced the NORAD number with the launch date of the satellite (available from Seradata) and found the exact same effect: test scores improved by 10% or more. We then found this was true for *both* tasks investigated here: bus type prediction *and* operational status prediction. In retrospect, perhaps it is obvious that launch date would provide useful information for assessing the operational status of a satellite.

However, while clearly beneficial, adding the launch date to model inputs conflicts with the problem investigated here: satellite characterisation from *light curves*. Furthermore, satellite tracking can often become confused due to poor custody maintenance or proximity manoeuvres. If the tracking becomes confused, the launch date associated to observations as contextual data may be incorrect; this could catastrophically influence the model prediction because the most important feature for determining outputs is the launch date when it is available.

We present a summary of these results in Table 5 and confusion matrices in Fig. 6. Recall in section 3 we estimated that the status labels were only accurate to 93% in LEO, so model ID 11 appears to be limited only by the accuracy of its input labels.

Table 5. Summary of results including launch date contextual data.

ID	Model	Data	Features	Tracks	Regime	Task	Accuracy
11	RF	MMT	Simple Stats plus launch date	Five-track Sequence	LEO	Operational status	**93 ± 3%**
12	RF	MMT	Simple Stats plus launch date	Five-track Sequence	LEO	Bus type	**97 ± 1%**

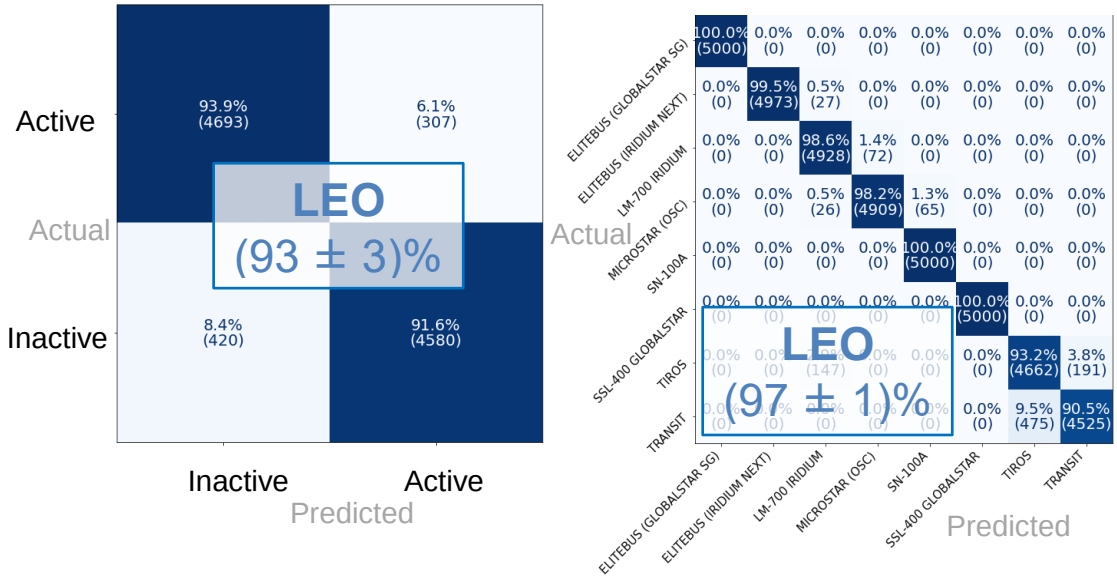

Fig. 6. Confusion matrices for models enhanced with launch date contextual data (model ID 11 on left, model ID 12 on right).

We also experimented with a number of alternative approaches which we briefly mention here.

Unsupervised learning is a technique that does not require labels to learn from the data. In general, this consists of either clustering (grouping data with similar input features) or dimensional reduction (finding a small number of input feature combinations that provide almost as much information as the raw feature set). We investigated clustering methods for a number of use-cases, including predicting operational status and change-point / anomaly detection. While some insights could be drawn from the results, they were not conclusive on their own.

Aside from LSTMs we also investigated (1D) CNNs, (time) vector embedding, attention layers (with LSTMs) and a number of other ANN architectures. None of these approaches performed better than those reported here.

Most ML models struggle to improve on the overall accuracy of their input labels. But a number of approaches have been developed to do exactly this. We investigated these and found mixed results: some had not matured enough from their academic background to apply to real-world problems; others made use of copy-left open-source licenses which prevent their use in defence.

11. CONCLUSIONS

This work has demonstrated that machine learning techniques can characterise satellites to improve space situational awareness using time-series satellite observation data such as light-curves. Specifically, we have demonstrated a capability to assist analysts in predicting the operational status of satellites (both LEO and GEO), and a capability to assist in distinguishing between the most common satellite bus types.

We have found that there is no single type of sensor (telescope or network of telescopes) that provides the best light-curve data for both the LEO and GEO regimes. Wide field lens arrays offer the best data for characterisation in LEO, and could also provide long term broad survey data for characterisation in GEO. However, (unmoderated) WFLA data is not currently readily available in the large quantities required for ML model training (commercially or otherwise). Conversely, traditional telescope networks operated by current commercial suppliers provide the best quality of data for characterisation in GEO due to their higher single target sensitivity, but can generally only gather data for one or a few satellites at a time (leading to compromises in sensor tasking), and can have difficulty tracking faster moving objects in LEO due to limitations in slew rate.

Most standard supervised machine learning models require accurate labels to train them. Although we were able to use ancillary data on satellites (such as knowledge of the operational status or bus type) to train such models we found that the labels were not always accurate, and overall this limited the performance of our models, particularly when predicting operational status. Early investigations into training specialist supervised learning models that can account for mislabelled inputs were promising.

Throughout our investigations, we maintained use of simpler classical machine learning models as baselines with which to compare the results of more complex deep learning models. We never found a situation where the deep learning models excelled and left the classical models behind. This is not a criticism of deep learning models; rather it is likely that the lack of size and diversity in the available datasets constrained their capability.

Our final point on the limitations of using deep learning is that of over-training and data leakage, especially when working with limited datasets. All machine learning models have a tendency to ‘cheat’ and obtain the answer through inappropriate means; it is down to the skill of the data scientist to curate the data to ensure this does not happen. But deep learning models are much more powerful and so much better at cheating! Current state-of-the-art deep learning models with billions of tuned parameters should be trained on terabytes – not gigabytes – of data. The power of these models means they trivially exploit weaknesses and inconsistencies in smaller, poorly curated datasets. As we found ourselves, deep learning models can learn to cheat in ways which would be hard to foresee beforehand, and are hard to detect (because of their black-box nature), until the model fails to perform at the expected level when deployed. *With great power comes great responsibility*: vast complex models need huge diverse datasets, and need considerable validation to ensure they will maintain performance when deployed on live data.

ACKNOWLEDGEMENTS