Early Classification of Space Objects based on Astrometric Time Series Data

Giovanni Lavezzi[*], **Peng Mun Siew**[†], **Di Wu**[‡]
Massachusetts Institute of Technology

Zachary Folcik[§]
MIT Lincoln Laboratory

Victor Rodriguez-Fernandez[¶]
Universidad Politécnica de Madrid

Jeffrey Price[‖]
United States Air Force

Richard Linares[**]
Massachusetts Institute of Technology

ABSTRACT

Space Situational Awareness (SSA) has gained prominence owing to its criticality in national defense and growing number of Resident Space Objects (RSOs) due to commercialization of space. The identification and classification of RSOs is a desirable objective for SSA. AI techniques, specifically Machine Learning (ML) and Deep Learning (DL) algorithms, now facilitate classification of space objects based on sensor observations. However, these AI algorithms are often constrained by the lack of a sufficiently large training dataset. This work aims to use supervised ML algorithms for early time series classification using Vector Covariance Message (VCM) data covering 22,303 RSOs over six months. ML techniques such as Random Forest, Logistic Regression, K-Nearest Neighbor, Support Vector Machine, Naive Bayes, along with a Multi-Layer Perceptron DL technique are initially tested. Subsequently, advanced time series classification algorithms like InceptionTime and TSiT, for regularly and irregularly sampled time series, respectively, are examined. Given the variability in RSO's VCM estimates and the problem of data imbalance between classes, time regularization and synthetic data are introduced. However, the use of synthetic data creates problems in the classification accuracy of the ML algorithms. The irregular time series classification technique can overcome this issue. Considering the novelty of using the VCMs as datasets for the ML algorithms, the study explores input combinations and variations of the datasets. Early classification of space objects is of key interest to space system operators, therefore the acceptable trade-off between accuracy and timeliness of the prediction is investigated.

1. INTRODUCTION

In response to the U.S. Air Force's goal to regulate, safeguard and ensure accessibility to space, the focus on Space Situational Awareness (SSA) has seen a significant surge recently. This area of study has become a critical field of interest due to its relevance and immediacy in relation to national defense, and to the recent exponential growth of Resident Space Objects (RSOs), a consequence of the commercialization of space. Precise RSO identification

[*]Corresponding author. Postdoctoral Associate, Department of Aeronautics and Astronautics, *glavezzi@mit.edu*
[†]Research Scientist, Department of Aeronautics and Astronautics, *siewpm@mit.edu*
[‡]Postdoctoral Associate, Department of Aeronautics and Astronautics, *d_wu@mit.edu*
[§]Technical Staff. *folcik@ll.mit.edu*
[¶]Associate Professor, Department of Computer Systems Engineering, *victor.rfernandez@upm.es*
[‖]Director of Technology Management, *pricej@mit.edu*
[**]Associate Professor, Department of Aeronautics and Astronautics, *linaresr@mit.edu*

Research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. DO NOT USE ANY MARKINGS THAT APPEAR BELOW THIS LINE.

and classification is desired for SSA activities and progress has been made possible by the latest advancements in sensor technology coupled with Artificial Intelligence (AI) techniques. Machine Learning (ML) and Deep Learning (DL) algorithms have been adopted to classify space objects based on various sensor observations from radar and telescope, such as light curves, and radar cross sections [1, 2, 3, 4]. These algorithms are also referred to as time series classification techniques, where the class or category of a time series sequence is predicted. Early time series classification, instead, involves the prediction of a class label based on partial observation of a given time series [5]. A critical aspect that affects the accuracy of AI-enable algorithms is often the lack of sufficiently large training datasets. This can be due to restricted data access, sparse observations for each RSO, or low precision measurements.

In this work, we aim to apply supervised ML algorithms to perform early time series classification on the Vector Covariance Message (VCM) data, comprising 22,303 RSOs over a period of six months. VCM data consist of RSO ephemerides produced using a high-precision special perturbations orbit propagator and estimator from tracking observations. The VCMs were used as measurements for the purpose of classification. The VCMs were provided by the United States Space Force Space Command (USSPACECOM), specifically the 18th Space Defense Squadron (18th SDS). The authors requested and obtained the VCMs using the USSPACECOM Orbital Data Request process [1]. Several conventional ML classification techniques [6] are initially tested, like Random Forest, Logistic Regression, K-Nearest Neighbor, Support Vector Machine, Naive Bayes, together with a Multi-Layer Perceptron (MLP) DL technique. Subsequently, more advanced time series classification algorithms are explored: InceptionTime [7] and TSiT [8], for regularly and irregularly sampled time series, respectively. For the regular time series classification technique, since each RSO VCMs are produced at non-uniform time intervals, a special perturbations propagator is used to interpolate the orbital state history with evenly spaced time steps. The time regularization, achieved by the use of the high-fidelity propagator, allows generation of synthetic data that can help with the data imbalance between the three classes. At the same time, the use of synthetic data in addition to the real data can create problems in the classification accuracy of the ML algorithms. This is due to long orbit propagation times and because the special perturbation propagator is not the same as the one originally used by the 18th SDS to generate the VCM. By avoiding the introduction of propagation errors, irregular time series classifiers may overcome the data imbalance problem.

Because using the VCMs as datasets for ML algorithms is novel, an analysis of several combinations of inputs to include in each dataset sample has been conducted. This involves one or more estimate or VCM epochs, the position and velocity vectors, the ballistic coefficient, the solar radiation pressure coefficient, and the VCM's orbital elements. In addition, differing datasets have been adopted when considering the input needs of the several ML algorithms. Of particular interest is the early classification of space objects, and consequently an investigation is conducted to demonstrate which is a satisfactory trade-off between accuracy and timeliness of the prediction, i.e. the minimum length of the time series measurement interval to achieve an acceptable classification accuracy. Results are presented in terms of both total accuracy and class specific F1 score, providing a balance between the precision and recall characteristics. This result scoring is especially useful in situations where the data is imbalanced among the multiple classes.

The paper is organized as follows. Section 2 introduces the VCM dataset and the RSOs class division. Section 3 characterizes the evaluation metrics for the ML classification algorithms. Section 4 describes some conventional ML classification algorithms, the dataset preparation, and initial results. Section 5 discusses the regular time series based ML classifier, the orbit propagator setup , the dataset structure, and its results. Section 6 presents the irregular time series classification technique, with focus on the dataset preparation and outcomes. Lastly, final remarks to conclude the paper are provided in Section 7.

2. VCM DATAFRAME

The Vector Covariance Message (VCM) data comprise 22,303 RSOs over a period of six months (9/1/2022-2/28/2023). VCM data consist of RSO ephemerides from a high-precision special perturbations orbit propagator and estimator using tracking observations. VCMs are issued by the US Space Force (USSF) Space Command (USSPACECOM) and were provided through an Orbital Data Request (ODR) the authors submitted to the 18th Space Defense Squadron (18th SDS). Table 1 reports an overview of the VCM dataset divided into three classes, Payload (P), Rocket Body (R), and Debris (D), for a total of 21,966 RSOs. The remaining RSOs instead are categorized as Unknown and have been excluded from the study.

[1] `https://www.space-track.org/documents/USSPACECOM_ODR.pdf`

Table 1: VCM dataset over a period of six months (9/1/2022-2/28/2023).

Class	# of RSOs [-]	# of Measurements [-]	Ratio [%]
Payload	9,169	8,080,946	62.2
Rocket Body	1,822	1,302,710	10.0
Debris	10,975	3,608,082	27.8

Each VCM orbital estimate is extracted and converted from its original format into a tabular structure which includes the satellite NORAD ID (North American Aerospace Defense Catalog Number), its international designator, and the following data items:

- *Epoch time* (UTC): orbital estimate epoch time, expressed in Coordinated Universal Time.
- *Rev*: epoch revolution number.
- *J2K/ECI/EFG Pos/Vel X-Y-Z* (km, km/s): the position and velocity vector components of the RSO expressed in Earth-Centered Inertial (J2K, ECI) and Earth-centered Earth-fixed (EFG) reference frames.
- *BC* (m^2/kg): ballistic coefficient.
- *Bdot* (m^2/kg-s): time rate of change of the ballistic coefficient.
- *C_SRP* (m^2/kg): solar radiation pressure coefficient.
- *Geo Pot*: geopotential model used, truncated to a certain degree zonals, and degree/order tesserals.
- *Drag*: atmospheric density model used.
- *EDR* (W/kg): energy dissipation rate.
- *F10, F10 Avg, ap Avg*: F10 (10.7 cm) solar flux or 81-day average F10, and average geomagnetic index.
- *Sigma U-V-W-Ud-Vd-Wd* (km, km/s): the standard deviation of the error in RSO's position and velocity.
- *COVARIANCE MATRIX (EQUINOCTIAL ELEMENTS)*: the covariance matrix estimates, which represent the lower triangular half of the covariance matrix in terms of equinoctial elements.
- *Polar Motion X-Y* (arcsec): components of polar motion.
- *IAU 1980 NUTAT*: number of terms used in the nutation model.
- Additional metadata related to the satellite center of mass offset and orbital perturbation model used by the orbit propagator, such as solar radiation pressure, solid earth tides, in-track thrust, lunar/solar and planets perturbations.

It must be noted that the VCMs are reported at non-uniform time intervals, and might not be available for certain RSOs.

3. EVALUATION METRICS

In ML classification, evaluating a model's performance extends beyond simple accuracy, especially when dealing with imbalanced datasets. This section describes the two evaluation metrics adopted in this work, accuracy and F1 score [9]. Accuracy is a metric that calculates the ratio of correct predictions to the total number of predictions:

$$Accuracy = \frac{TN + TP}{TN + FN + TP + FP} \tag{1}$$

with True Positives (TP) the correctly predicted positive observations, True Negatives (TN) the correctly predicted negative observations, False Positives (FP) the incorrectly predicted positive observations (or type I error), and False Negatives (FN) the incorrectly predicted negative observations (or type II error). Accuracy can be misleading, particularly with imbalanced datasets where one class significantly outweighs others.

The F1 Score provides a more comprehensive evaluation by considering both precision and recall, which is especially crucial when dealing with imbalanced datasets or when the cost of FP and FN differ significantly. Precision is the ratio of correctly predicted positive observations to the total predicted positives:

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

Instead, recall (also called sensitivity or true positive rate) is the ratio of correctly predicted positive observations to all actual positives:

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

In this way, the F1 score is defined as the harmonic mean of precision and recall:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Both accuracy and F1 score range between 0-1, where 0 indicates the worst possible performance (no true positives), and 1 represents perfect precision and recall (all positive predictions are correct, and all actual positives are identified).

4. CONVENTIONAL ML CLASSIFICATION ALGORITHMS

Initially, several common ML classification techniques [6, 10] are tested, such as Random Forest, Logistic Regression, K-Nearest Neighbor, Support Vector Machine, Naive Bayes, and a MLP DL technique. The Decision Tree classifier works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches" to "leaves" where the categories become more finitely similar. The Extra Trees Classifier is a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The Random Forest classifier is an expansion of the Decision Trees, an ensemble method that operates by training a number of decision trees and returning the class with the majority over all the trees in the ensemble. The Logistic Regression method is usually adopted for binary classification problems, but can be extended to support multi-class classification problems. It calculates the probability of a dependent variable, e.g., the RSO's type, given an independent variable, the RSO's position and velocity vectors. The Logistic Regression CV uses a cross-validation (CV) estimator, an estimator that has built-in cross-validation capabilities to automatically select the best hyper-parameters. The SGD classifier is an estimator that implements regularized linear models with stochastic gradient descent (SGD) learning; the gradient of the loss is estimated each sample at a time, and the model is updated along the way with a decreasing strength schedule, i.e. learning rate. The Ridge classifier is classifier using Ridge regression. Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares. For multi-class classification, the problem is treated as multi-output regression, and the predicted class corresponds to the output with the highest value. The Ridge classifier CV adopts a CV estimator, which has built-in cross-validation capabilities to automatically select the best hyper-parameters. The Linear Support Vector classification (SVC) is a Support Vector Machine algorithm, which is based on the concept of maximizing the minimum distance from hyperplane to the nearest sample point. It has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples with respect to other SVC algorithms, such as C-Support Vector Classification and Nu-Support Vector Classification, which are disregarded in the comparison. Indeed, their fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. The K-Neighbors classifier (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples. The Naive Bayes (NB) or Bayesian network are based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. The Bernoulli-NB works on Bernoulli distributions, and it is designed for binary/boolean features. The Gaussian-NB is another variant of NB, with the assumption that the continuous numerical attributes are normally distributed. The attribute is first segmented based on the output class, and then the mean and variance of the attribute are calculated for each class. In addition, Neural Networks (NN) are chosen as an alternative to conventional techniques which are often limited by strong assumptions of normality, linearity, and variable independence. In particular, we consider the MLP Classifier, which is a fully connected class of feed-forward artificial neural network.

4.1 Dataset Preparation

Three different datasets are considered. We refer to the first one as Single Epoch Approach, where the number of samples n_S comprises every single measurement or estimate in each VCM object, and the number of variables n_V in each sample is constituted by at least the position and velocity vectors *J2K Pos/Vel X-Y-Z*, with the addition of parameters such as *BC*, *Bdot*, and *C_ SRP*. We denote the second dataset as Sub-sampling Approach, where each n_S contains two consecutive measurements of each RSO for every VCM object, and each consecutive measurement is characterized by the same n_V used in the Single Epoch Approach. The third dataset is called Sub-sampling Approach with Overlapping, since each consecutive n_S share one measurement, i.e. the second measurement in one sample is the same as the first measurement in the next sample. Ideally we want to have measurements that are evenly spaced in time, but the VCMs are provided at non-uniform time intervals. So, we are augmenting the time interval between the two measurements as n_V to each sample line, in order to assist the algorithm in capturing the relationship between the two consecutive measurements. Regarding the Sub-sampling Approach, the overlapping procedure is used only for the Rocket Body class because of the much smaller data ratio.

Furthermore, feature normalization has shown to improve the performance of various ML algorithms. The goal of feature normalization is to transform features to be of a similar scale and to improve the performance and training stability of the model. Thus, a Z-score normalization is used, where all data-points are modified to have a mean of 0 and a standard deviation of 1. For each data point x in the dataset, the Z-score is calculated as:

$$Z = \frac{(x - \mu)}{\sigma} \tag{5}$$

where, Z is the Z-score, x is the original data point, μ is the mean of the dataset, and σ is the standard deviation of the dataset.

The dataset preparation is carried out on every class in the VCM dataset. A 80/20 train-test split is adopted, where the testing dataset is composed of RSOs with NORAD IDs that are not included in the training dataset. Table 2 reports the training/testing division in terms of number of measurements for the three datasets.

Table 2: Training/testing dataset division.

Split	# of measurements	Single Epoch Approach	Sub-sampling Approach	Sub-sampling Approach with Overlapping
Training	Total	10,393,391	5,712,303	10,375,813
Testing	Total	2,597,171	1,426,901	2,593,421
	P	1,616,048	807,419	1,614,097
	R	259,691	259,297*	260,045
	D	721,432	360,185	719,279

* Sub-sampling with overlapping is used.

4.2 Results and Discussion

The ML classification algorithms are provided by the Scikit-learn library [10], and implemented in a Python environment running on the MIT SuperCloud, a high performance computing system [11]. Table 3 reports the initial results for the conventional ML classification algorithms with default settings, *J2K Pos/Vel X-Y-Z* as variables, and no normalization. The Decision Tree, Extra Trees, Random Forest, and K-Neighbors are the top performing algorithms both when considering the accuracy between the three classes, and the three dataset approaches. The Sub-sampling Approach with Overlapping allows the top performing algorithms to reach the highest accuracy, probably due to the inclusion of the time interval between two consecutive measurements in each sample. The overlap also results in a larger and more representative dataset, capturing more temporal dynamics and patterns compared to the single epoch and sub-sampling without overlapping approaches.

Table 3: Results in terms of accuracy (0-1) between all the classes for the conventional ML classification algorithms.

Algorithm	Single Epoch Approach	Sub-sampling Approach	Sub-sampling Approach with Overlapping
Decision Tree	0.75	0.71	0.79
Extra Trees	0.78	0.75	0.82
Random Forest	0.78	0.76	0.82
Logistic Regression	0.62	0.59	0.66
Logistic Regression CV	0.62	0.59	0.66
SGD	0.61	0.55	0.66
Ridge	0.62	0.57	0.64
Ridge CV	0.62	0.57	0.64
Linear Support Vector	0.31	0.57	0.59
K-Neighbors	0.76	0.73	0.78
Bernoulli-NB	0.62	0.58	0.62
Gaussian-NB	0.29	0.28	0.46
MLP	0.64	0.59	0.69

The Sub-sampling Approach with Overlapping is further explored to understand the capabilities of every classification algorithm to predict the specific RSOs class. The n_V is extended to include *BC* and *C_SRP*, since they contain information about the mass and size of the RSOs, which we noted to be beneficial to differentiate and correctly categorize the RSOs. It must be noted that *Bdot* is not included in n_V since it is zero for all the RSOs in the VCM data. Table 4 shows the results in terms of accuracy and F1 score for each class. The *Linear Support Vector* and K-Neighbors are not included due to their long training times. It is evident that the F1 score related to the Rocket Body class is very low, if not zero for many ML classification algorithms. Moreover, the inclusion of the additional variables allows the methods to reach higher accuracy values overall. The Random Forest retrieves the highest accuracy and F1 score values in every class. Similarly, the MLP, which is the only NN considered in this analysis, is able to reach quite high scores in P, D, but not in R. Accordingly, we introduce two techniques which can lead to an increase in MLP performance: (1) the Z-score normalization, for faster convergence and improved generalization, and (2) the addition of hidden layers, to increase model depth and try to capture the non-linear relationships in the data. In addition, a preliminary data augmentation for the Rocket Body class is adopted, where the original R data are duplicated, and random noise is assigned to each variable. In particular, a zero mean Gaussian random variable with a σ of 0.05 and 0.1 is introduced. This corresponds to 5% and 10% error, respectively, when added to the normalized data. As shown in Table 5, both Z-score normalization and increased model depth improved the accuracy and F1 scores of the MLP, resulting in performance close to or exceeding that of the Random Forest. Furthermore, the results clearly demonstrate that preliminary data augmentation nearly doubles the F1 score for the R model.

Table 4: Results in terms of accuracy and F1 score (0-1) for the conventional ML classification algorithms.

Algorithm	Accuracy	F1 score - P	F1 score - R	F1 score - D
Decision Tree	0.84	0.89	0.48	0.83
Extra Trees	0.86	0.90	0.47	0.85
Random Forest	0.86	0.90	0.49	0.85
Logistic Regression	0.72	0.82	0	0.61
Logistic Regression CV	0.73	0.82	0	0.59
SGD	0.68	0.79	0	0.51
Ridge	0.67	0.79	0	0.32
Ridge CV	0.68	0.79	0	0.32
Bernoulli-NB	0.72	0.82	0.22	0.62
Gaussian-NB	0.71	0.82	0.33	0.40
MLP	0.80	0.86	0	0.77

Table 5: Comparison in terms of accuracy and F1 score (0-1) for *Random Forest* and *MLP* with normalization and preliminary data augmentation.

Algorithm	Accuracy	F1 score - P	F1 score - R	F1 score - D
Random Forest	0.86	0.90	0.49	0.85
MLP, layers = (100,)	0.83	0.88	0.38	0.80
MLP, layers = (40,40,40)	0.88	0.92	0.49	0.88
MLP, layers = (40,40,40), $\sigma = 0.05$	0.87	0.90	0.80	0.82
MLP, layers = (40,40,40), $\sigma = 0.1$	0.88	0.90	0.83	0.85

As an additional approach, the classical orbital elements (COEs) for each VCM are combined with the position and velocity vector, i.e. state vector, and added to the n_V for each sample. The COEs comprise of a the semi-major axis, e the orbital eccentricity, i the orbital inclination, ω the argument of perigee, Ω the right ascension of the ascending node, and v the true anomaly. The relationship between the state vector and COEs can be found in [12]. Table 6 compares the *Random Forest* and *MLP* when considering the Z-score normalization, the preliminary data augmentation, and the inclusion of the COEs. The two algorithms show an improvement, whereas similar results are obtained when removing v from the COEs. This is expected, since v is a set of fast-changing variables and should be less pivotal in RSO orbit characterization, and, consequently, categorization.

Table 6: Comparison in terms of accuracy and F1 score (0-1) for *Random Forest* and *MLP* with normalization, preliminary data augmentation, and COEs.

Algorithm	Accuracy	F1 score - P	F1 score - R	F1 score - D
Random Forest, $\sigma = 0.1$	0.89	0.91	0.87	0.86
MLP, layers = (40,40,40), $\sigma = 0.1$	0.89	0.91	0.86	0.87

5. REGULAR TIME SERIES CLASSIFICATION

In this section, a regular time series based ML classifier is tested. This classifier was attempted because the inclusion of the time interval between two consecutive measurements led to the results having the highest accuracy among the conventional ML techniques. Also, we are interested in the early classification of RSOs, i.e. we want to use only a short sequence of measurements to predict an RSO's class. Specifically, we choose the InceptionTime [7] algorithm, which is inspired by the Inception-v4 architecture for image classification. InceptionTime uses inception modules which are essentially convolutional neural network (CNN) layers with multiple differing kernel sizes. This allows handling of different patterns in the data. The initial choice of InceptionTime is due to its influential role in DL for time series classification, and its potential for strong performance, as demonstrated in benchmark studies [7]. The regular time series classification technique, owing to the non-uniform time intervals between subsequent VCM measurements, uses a special-perturbations propagator to interpolate astrometric data with evenly spaced time steps. Moreover, the time regularization produces synthetic data which is used in data augmentation for the Rocket Body class. This helps with the aforementioned data imbalance issue.

5.1 Orbit Propagator Setup

The Orekit space dynamics library [13] is used in a Python [14] environment to generate the astrometric data with evenly spaced time steps. To make sure the orbit propagator matches as close as possible the one originally used for the VCM by the USSF 18th SDS, the following perturbing accelerations, that are implemented in the Orekit space dynamics library, are considered in the analysis [12, 15, 16]: the Earth's non-spherical gravity field, atmospheric drag, solar radiation pressure, lunisolar third body accelerations, and Earth solid tides. Specifically, the Earth's non-spherical gravity field is modeled up to degree and order 70. Regarding the atmospheric drag, a drag coefficient equal to 2.2 is assumed, whereas the Jacchia-Bowman 2008 (JB2008) empirical density model [17] is used to retrieve the value of the atmospheric density, with solar and geomagnetic data provided by CelesTrak [18]. For the solar radiation pressure, a constant coefficient of reflectivity equal to 1.0 is assumed for each RSO. Furthermore, the values of the cross-sectional

area A, and the sun-exposed area $A_\odot$ associated to each RSO can be computed based on BC and C_SRP extracted from the VCM data:

$$A = \frac{m \cdot BC}{c_D} \tag{6}$$

$$A_\odot = m \cdot C\_SRP \tag{7}$$

where the RSO mass is:

- extracted from the ESA's DISCOS (European Space Agency's Database and Information System Characterising Objects in Space) [19] catalogue, if available, or
- approximated as $m = avg\_cross\_section \cdot 100$ (kg/m^2), with $avg\_cross\_section$ the average cross sectional area (m^2) extracted from the ESA's DISCOS catalogue, if available, or
- assumed as a default value of $m = 100.0$ kg for Payload, $m = 1000.0$ kg for Rocket Body, and $m = 1.0$ kg for Debris.

5.2 Dataset Preparation

The dataset for the regular time series classifier is composed of a 3D array of size $n_S \times n_V \times n_{ST}$, where:

- Total number of samples, n_S: given an RSO, the orbital period T_P is computed and divided into n_{S_P} points, then the closest VCMs data to these points are found, while discarding the repeated points, since sometimes the same VCM data can be closer to multiple desired points. The value of n_{S_P} corresponds to number of samples that are extracted from each RSO, e.g. $n_{S_P} = 4$ means four samples for one RSO. In particular, considering the measurements epochs associated to one RSO, t_{MS}, the first sample associated to n_{S_P} is the first available RSO's measurement date, $t_{MS}(0)$, belonging to the first orbital period T_P. Instead, the second sample n_{S_P} is extracted after summing one T_P to the fraction of the orbital period corresponding to the second n_{S_P}. This is repeated for all the remaining n_{S_P}, as described by the following relation:

$$t_{MS}(i) = T_P + \frac{i}{n_{S_P}(i)} T_P, \quad \text{for } i \in \{1, 2, \cdots, n_{S_P}\} \tag{8}$$

  In this way, we ensure the utilization of measurements epochs that span over the full VCM period. It must be noted that the 0 index is considered as the first index because Python uses a zero-based indexing.

- Total number of variables (or features), n_V: which are included in each n_S, constituted by at least the position and velocity vectors *J2K Pos/Vel X-Y-Z*, with the addition of parameters such as BC, C_SRP, and the COEs.
- Total number of steps, n_{ST}: given a sample n_S, the state vector associated to a measurement epoch t_{MS} is propagated backward and forward in time by a selected time interval Δt_{ST}, e.g., every 5, 10, or 15 minutes, for a total propagation time of a few hours. This value is based on the findings related to how the time interval affects the accuracy of the orbit propagator, and on the early classification requirement of this study. Hence, astrometric data with evenly spaced time steps are generated.

The dataset preparation is conducted on every class in the VCM dataset. A 80/20 train-test split is adopted after randomizing and normalizing the data, where the testing dataset is composed of RSOs with NORAD ID that are not included in the training dataset. To improve the performance of the ML model, a normalization per channel using the whole dataset is adopted, meaning that the normalization process is applied independently to each feature/variable n_V, in order to ensure that different features have comparable scales. The data augmentation for the Rocket Body is performed by considering more n_{S_P} for each RSO of that class, by mean of a data augmentation multiplication factor n_{AUG}, and, consequently, retrieving more samples n_S of the same RSO.

5.3 Results and Discussion

The InceptionTime algorithm is provided by Tsai [20], an open-source DL package based on Pytorch and fastai [21]. Specifically, we use InceptionTimePlus, which builds upon the InceptionTime architecture and adds additional features, such as dropout and batch normalization, enhancing its capabilities. The configuration of InceptionTimePlus considers a batch size of 64 for the training data and 128 for the testing data, since the larger batch size can help to speed up the testing process, as parameter updates are not performed during testing. If not stated elsewhere, a default value for the convolutional dropout of 0 and a NN's depth of 6 are used. Dropout is a regularization technique used in NN to prevent overfitting, and it works by randomly dropping out (setting to zero) some of the extracted features during each training step, forcing the model to learn more generalized representations and reducing overfitting. Instead, the depth of NNs refers to the number of layers in the network. Increasing depth generally increases the representational capacity of the model, allowing it to learn more intricate relationships in the time series data; but deeper NN are more computationally expensive to train and might require more data to avoid overfitting. The fastai's learner rate finder function is used to set the maximum learning rate, which is based on [22]. Essentially, it employs a cyclical learning rate approach, gradually increasing the learning rate during a training cycle and plotting the training loss against it. The resulting plot reveals a sweet spot where the loss decreases rapidly before surging. This spot indicates the optimal learning rate range. For the learning rate schedule, a dynamic learning rate with a one cycle policy is adopted [23], where it manages the learning process over the entire training duration (epochs), making adjustments at each training step to optimize learning rate, momentum, and weight decay for improved convergence and generalization. For the loss function, the Cross-Entropy Loss Flat is adopted, i.e. the Cross-Entropy Loss with flattened input and target [20], a commonly used loss function in ML for classification tasks. It measures the difference between the model's predicted probability distribution and the true (target) probability distribution of the data. The model makes more confident and accurate predictions by being penalized more heavily for incorrect predictions with high confidence, and leading to better learning and improved classification performance [21].

The results of the regular time series classification algorithm in terms of F1 score are shown in Fig. 1, where several combinations of number of steps n_{ST}, propagation time interval Δt_{ST}, orbital period's sample points n_{S_P}, number of variables n_V, and augmentation factor n_{AUG} are analyzed. For all cases, the data augmentation of the Rocket Body class is included. From the analysis of the results, InceptionTimePlus is not able to retrieve a F1 score higher than 0.9 for Payload and Debris classes, and higher than 0.8 for the Rocket Body class. The overall best F1 score is achieved by the case with $n_{ST} = 24$, $\Delta t_{ST} = 5$, for a total backward and forward propagation time of 2 hours, $n_{S_P} = 12$, an augmentation multiplication factor $n_{AUG} = 6$, and the RSOs state vector as n_V. In general, better results are achieved when considering only the state vector or only the COEs as features. The inclusion of *BC*, and *C_SRP* doesn't seem to improve the score, but actually degrades the performances for the Rocket Body class. This is probably due to the fact that *BC* and *C_SRP* are kept constant during the data generation, and are not bringing useful information to the model to learn. However, when they are not included in the dataset, the orbit propagator seems to generate sufficiently accurate data. By looking at the two cases with all the variables included in the dataset, it can be seen that a larger data augmentation factor improves the results for the Rocket Body class. Moreover, a higher number of orbital period's sample points n_{S_P} is beneficial, meaning that having more samples related to different part of the SOs orbit allows the model to improve the learning, and predict more accurately.

It must be noted that in these tests, the training losses are reducing throughout the training epochs, suggesting that the model is learning, contrary to the testing (or validation) losses which are slightly increasing, meaning that the model is not generalizing well and is overfitting. For this reason, the values of the convolutional dropout and NN's depth are tuned to reduce the testing (or validation) losses. Specifically, a convolutional dropout of 0.6 and NN's depth of 3 are able to reduce the validation losses, and consequently the overfitting, to the same level of the training losses, while maintaining similar F1 scores, even when tested on longer training epochs. In future works, we will conduct additional analyses to effectively tune the model's hyperparameters.

In addition, we checked the model classification's performances after removing RSOs with altitude greater than 45,000 km. The motivation is related to the fact that fewer sensor observations can be made of these RSOs and, consequently, classification is more challenging. The following case $n_{ST} : 12, \Delta t_{ST} : 10, n_{S_P} : 4, n_V : 6\ (Pos, Vel), n_{AUG} : (0,6,0)$ is considered, and the result presents a slight increase in the F1 scores, passing from $P : 0.78$, $R : 0.68$, $D : 0.82$, to $P : 0.80$, $R : 0.71$, $D : 0.83$, and a decrease in the training/testing losses.

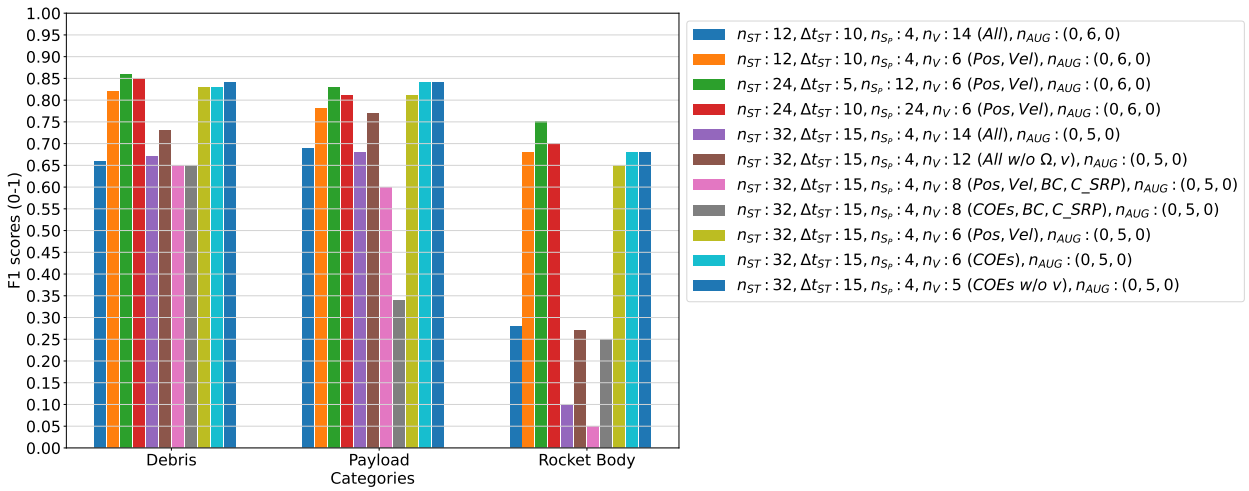

Fig. 1: Regular time series classification results based on several combinations of number of steps n_{ST}, propagation time interval Δt_{ST}, orbital period's sample points n_{S_p}, number of variables n_V, and augmentation factor n_{AUG} for (P, R, D). If the same n_V is indicated, details about the n_V types are reported.

6. IRREGULAR TIME SERIES CLASSIFICATION

The use of synthetic data generated with the orbit propagator is not able to achieve a satisfactory F1 score for the Rocket Body class in the regular time series classification. This is probably due to the large orbit propagation times and to the fact that the special perturbation propagator is not the same as the one originally used for the VCMs by the USSF 18th SDS. For this reason, we explore the irregular time series classification technique to try to overcome this issue, by recognizing the time irregularity of the VCMs. In particular, we select TSiT [8] algorithm, or Time Series Transformer model based on ViT (Vision Transformer), which is based on the Transformer model [24], highly successful in natural language processing tasks, and employs self-attention mechanisms that allow it to focus on important parts of the time series without being constrained by fixed windows or intervals.

6.1 Dataset Preparation

The dataset for the irregular time series classifier is composed of a 3D array of size $n_S \times n_V \times n_{ST}$, where:

- Total number of samples, n_S: each n_S is a RSO from the VCMs.
- Total number of variables (or features), n_V: which are included in each n_S, constituted by at least measurements *Epoch time* and the position and velocity vectors *J2K Pos/Vel X-Y-Z*, with the addition of parameters such as *BC*, *C_SRP*, and the COEs (excluding the true anomaly, ν). The epochs are considered as the number of seconds starting from the first measurement of the sequence.
- Total number of steps, n_{ST}: representing a sequence (or slice) of measurements extracted from each RSO. The shorter the sequence, the shorter is the total time span covered, allowing to investigate the early classification requirement of space objects. After selecting the value of n_{ST}, all the RSOs with a total number of measurements less than n_{ST} are disregarded. Usually, the larger n_{ST}, the larger is the number of RSOs to disregard, as shown in Table 7. This is an important aspect to consider since we don't want to lose too many useful information when training the ML model.

The dataset preparation is performed on every class in the VCM dataset. A 80/20 train-test split is adopted after randomizing and normalizing the data, where the testing dataset is composed of RSOs with NORAD ID that are not included in the training dataset. To improve the performance of the ML model, a normalization per channel using the whole dataset is adopted, meaning that the normalization process is applied independently to each feature/variable n_V, in order to ensure that different features have comparable scales.

The data augmentation is carried out by considering multiples n_{ST} for each RSO of that class without overlapping, and, consequently, retrieving more samples n_S of the same RSO. We refer to the data augmentation multiplication factor as n_{AUG}. Different options to extract n_{AUG} and multiple sequence of measurements n_{ST} from the same RSO are tested:

- Opt. 1: given an RSO, consecutive measurements slices of length n_{ST} are extracted, starting from the RSO's first available measurement without randomizing their order.
- Opt. 2: given an RSO, consecutive measurements slices of length n_{ST} are extracted, starting from the RSO's first available measurement, and then we randomize their order.
- Opt. 3: given an RSO, consecutive measurements slices of length n_{ST} are extracted, with randomized starting point, and an automatic loop search is performed to include additional n_{AUG} slices. The loop search stops after a maximum number of attempts, e.g. 10^3-10^4, allowing for less than n_{AUG} slices. This is necessary to avoid the algorithm from being stuck in the `while` loop, since every RSO has different total number of measurements, and the multiple of n_{ST} might be a value very close to the total measurements, making the randomized search without overlapping occasionally difficult to be accomplished.
- Opt. 4: it is a combination of Opt. 3 with the addition of Opt. 2 whenever the maximum number of attempts in the loop search is reached, to prevent some of the n_{AUG} slices from being dropped.

Moreover, we implemented a maximum time interval cut-off (Δt_{cut})—including only measurement sequences with intervals shorter than this threshold. It must be noted that Opt. 3 is the most generic one, due to the randomization of both the starting point and the slices' order. Table 8 reports an explanatory example of the four options considering an RSO with 100 measurements per class, $n_{ST} = 30$, $n_{AUG} = 1$ for Payload and Debris, and $n_{AUG} = 3$ for Rocket Body.

Table 7: Number of disregarded RSOs in the VCMs with total number of measurements less than a cutoff threshold n_{ST} for Payload, Rocket Body, and Debris classes.

# RSOs with $n_{ST} <$	30	75	100	150	200
Payload	132	205	223	419	583
Rocket Body	7	15	42	103	219
Debris	778	1,982	2,623	3,655	4,621

Table 8: Example of the data augmentation options considering an RSO with 100 measurements per each class, $n_{ST} = 30$, $n_{AUG} = 1$ for Payload and Debris, and $n_{AUG} = 3$ for Rocket Body.

	Sequence of measurements of length n_{ST}		
Option	Payload	Rocket Body	Debris
1	(0-30)	(0-30) (30-60) (60-90)	(0-30)
2	(30-60)	(30-60) (0-30) (60-90)	(60-90)
3	(30-60)	(48-78) (17-47) (missing due to max attempts)	(60-90)
4	(30-60)	(67-97) (34-64) (0-30)	(60-90)

6.2 Results and Discussion

In this work, we use the TSiTPlus algorithm, an enhanced version of TSiT with additional features and provided by the library Tsai [20]. Similarly to the regular time series classification algorithm, the configuration of TSiTPlus considers a batch size of 64 for the training data and 128 for the testing data, a default value for the convolutional dropout of 0 and a NN's depth of 6, the fastai's learner rate finder function, the dynamic learning rate with a one cycle policy, and the Cross-Entropy Loss Flat as loss function.

First, a comparison between two data augmentation strategies is carried out. In particular, two models are trained with the data augmentation options 1 and 3, and then tested with a dataset which has been created with the same options,

as depicted in Table 9. When the trained model is tested against the dataset created with the same data augmentation option, good F1 scores are obtained, as expected. However, when the trained model is tested against the dataset based on the other data augmentation option, option 3 achieves more consistent F1 scores, meaning that it is capable of better generalization, due to the randomization of both the starting point and the slices' order. For this reason, the data augmentation opt. 3 is chosen for the next analysis.

Table 9: Results of comparison between data augmentation strategy options 1 and 3 in terms of F1 scores, for the following case, $n_{ST}:30, n_V:14, n_{AUG}:(0,5,0), depth:6$.

	Trained model w/ opt. 1	Trained model w/ opt. 3
Testing dataset w/ opt. 1	P: 0.90, R: 0.87, D: 0.95	P: 0.85, R: 0.82, D: 0.93
Testing dataset w/ opt. 3	P: 0.80, R: 0.79, D: 0.92	P: 0.84, R: 0.80, D: 0.94

The results of the irregular time series classification algorithm in terms of F1 score are shown in Fig. 2, where several combinations of number of steps n_{ST}, number of variables n_V, augmentation factor n_{AUG}, and NN's depth are analyzed. When $n_V = 14$, all the variables are considered, as described in the dataset preparation, whereas $n_V = 9$ refers to the cases where the COEs are not included in the dataset. Based on our analysis, TSiTPlus demonstrates superior classification accuracy compared to InceptionTimePlus, achieving higher accuracy for all three classes. This confirms how the use of synthetic data in addition to the real one can create problems in the classification accuracy of the ML algorithms. Moreover, the inclusion of *BC*, and *C_SRP* does not decrease the classification performances of the algorithm, since they are directly extracted from the VCMs data at the specific *Epoch time*. They are actually quite helpful at characterizing the RSOs' behavior, since they provide information about their mass and size. The two cases with no data augmentation for the Rocket Body class have very low F1 scores of about 0.55, even if Payload and Debris classes are able to achieve two of the highest scores. This trend highlights once more the problem of the data imbalance between classes, and the need of more data to achieve better accuracies. In general, it is evident that an increase in performances is reached by increasing n_{ST}. However, promising results related to the early classification of RSOs are shown by the following two cases $n_{ST}:5, n_V:9, n_{AUG}:(5,25,5), depth:3$ and $n_{ST}:15, n_V:14, n_{AUG}:(0,5,0), depth:3$, with only 5 and 15 measurements, respectively.

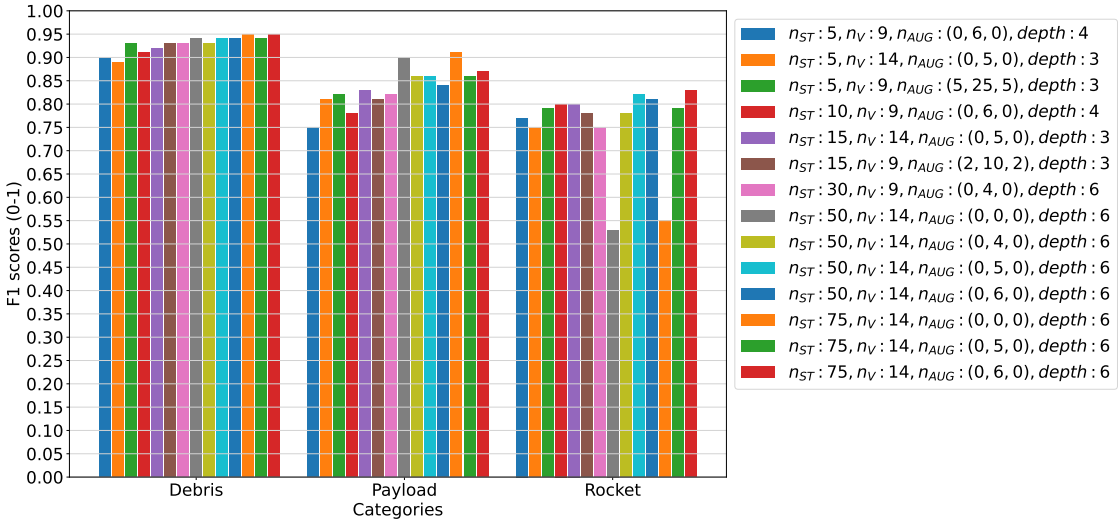

Fig. 2: Irregular time series classification results based on several combinations of number of steps n_{ST}, number of variables n_V, augmentation factor n_{AUG} for (P, R, D), and NN's depth.

It must be noted that in these tests, similar to the regular time series classification, the training losses are reducing throughout the training epochs, suggesting that the model is learning, whereas the testing (or validation) losses might

occasionally increase, meaning that the model is not generalizing well and is overfitting. For this reason, different value of the NN's depth are tested and, in some cases, they help to reduce the losses, but additional analysis are required to effectively tune the model's hyperparameters.

In addition, we examine the effects on the classification performances of a maximum time interval cut-off Δt_{cut}, expressed in days, as depicted in Fig. 3. In these test, opt. 3 for the data augmentation and a NN's depth of 6 are chosen. Overall, the F1 scores seem improving when constraining the maximum time interval between measurements. Indeed, even with n_{ST} of 5 and 30, the Payload and the Rocket Body class are able to more easily reach F1 scores greater than or equal to 0.85 and 0.80, respectively. This is due to the Payload and Rocket Body classes having less outliers greater than Δt_{cut}, i.e. including a maximum time interval cut-off allows to have more consistent data slices representative of the behavior of the class. Contrary to D, which has more measurements with a maximum time interval between measurements greater than Δt_{cut}.

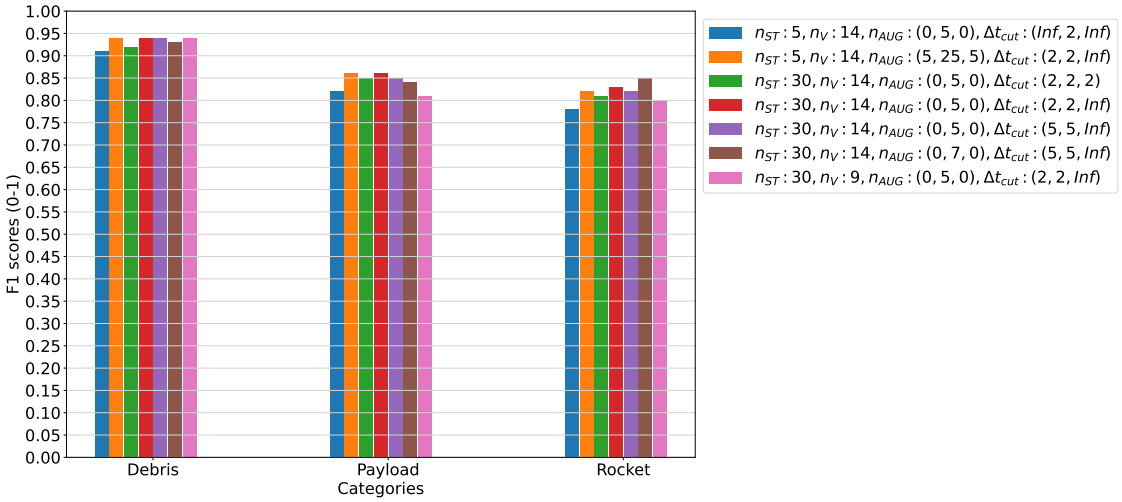

Fig. 3: Irregular time series classification results with maximum time interval cut-off Δt_{cut}, based on several combinations of number of steps n_{ST}, number of variables n_V, and augmentation factor n_{AUG} for (P, R, D).

7. CONCLUSIONS

In this work, we leverage a VCM dataset—the first large-scale and long-time coverage dataset comprising 22,303 RSOs over a period of six months—to perform data classification in space applications. A preliminary comparison of 15 AI techniques, composed of conventional ML and DL classification algorithms, and advanced ML classifiers for regularly and irregularly time series is carried out. The comparison explores several datasets structures, input combinations, and trade-off for early classification of RSOs. Results show that the irregular time series classifier is able to obtain higher F1 scores than the regular time series classification algorithm. Specifically, F1 scores higher than 0.80 are obtained for every class with peaks of almost 0.95 for the Debris class, and depending on the number of measurements included in each sample. In this case, the data augmentation is performed by considering more samples, i.e. a sequence of real measurements, belonging to the same VCM data, instead of synthetic data generated by the orbit propagator. In addition, it is evident that the Rocket Body class suffers from class imbalance issues, resulting in a bias towards the majority class, and in poor performance for the minority class, when data augmentation is not performed. Promising results for the early RSOs' classification are demonstrated, with satisfactory F1 scores achieved with as little as 5, 10 or 15 measurements.

Future work will focus on several key areas to further enhance model performance. First, rigorous hyperparameter tuning will be conducted to optimize classification accuracy and minimize validation loss. Additionally, an analysis of the model's training computational time will be undertaken to assess efficiency. Furthermore, the current model will be compared against other state-of-the-art regular and irregular time series classification algorithms. Finally, a more recent collection of VCMs will be integrated to augment the existing VCM dataset. This particularly addresses the need for more samples within the Rocket Body class.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

CODE AVAILABILITY

The irregular time series trained model used to compute the results in this paper is provided in `https://github.com/ARCLab-MIT/ML-RSO-Classification`.

COMPETING INTERESTS

The authors declare no competing interests.

ACKNOWLEDGMENTS

The authors thank Deshaun Hutchinson and Amanda Harman at the US Space Force 18 SDS for their efforts in collecting and providing the VCMs used in this study.

Research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

8. REFERENCES

[1] B. Jia, K. D. Pham, E. Blasch, D. Shen, Z. Wang, and G. Chen. Space Object Classification Using Fused Features of Time Series Data. In S. Ryan, editor, *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, page 93, January 2017.

[2] Bin Jia, Khanh D. Pham, Erik Blasch, Zhonghai Wang, Dan Shen, and Genshe Chen. Space object classification using deep neural networks. In *2018 IEEE Aerospace Conference*, pages 1–8, 2018.

[3] Roberto Furfaro, Richard Linares, and Vishnu Reddy. Space Objects Classification via Light-Curve Measurements: Deep Convolutional Neural Networks and Model-based Transfer Learning. In S. Ryan, editor, *The Advanced Maui Optical and Space Surveillance Technologies Conference*, page 11, September 2018.

[4] Marcos Damian Perez, Lmo M. A. Musallam, A. Garcia, E. Ghorbel, K. A. Imsaeil, D. Aouada, and Perig Le Henaff. Detection & identification of on-orbit objects using machine learning. 2021.

[5] Gilles Ottervanger, Mitra Baratchi, and Holger H. Hoos. Multietsc: automated machine learning for early time series classification. *Data Min. Knowl. Discov.*, 35(6):2602–2654, nov 2021.

[6] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315, 2016.

[7] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and Franccois Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34:1936 – 1962, 2019.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[9] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 345–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, Michael Jones, Anna Klein, Lauren Milechin, Julia Mullen, Andrew Prout, Antonio Rosa, Charles Yee, and Peter Michaleas. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2018.

[12] David A. Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm Press, 2013.

[13] Luc Maisonobe, Véronique Pommier, and Pascal Parraud. Orekit: An open source library for operational flight dynamics applications. In *ICATT 2010, ESAC, Madrid*, 04 2010.

[14] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[15] Giovanni Lavezzi, Miles Lifson, Simone Servadio, and Richard Linares. An analysis of orbital separation distances to support space traffic management. In *2023 AAS/AIAA Astrodynamics Specialist Conference, August 13-17 2023, Big Sky, Montana*, 08 2023.

[16] Giovanni Lavezzi, Miles Lifson, Simone Servadio, and Richard Linares. Orbital tolerance and intrinsic orbital capacity for electric propulsion constellations. *Journal of Spacecraft and Rockets*, 0(0):1–15, 0.

[17] Bruce Bowman, W Kent Tobiska, Frank Marcos, Cheryl Huang, Chin Lin, and William Burke. A new empirical thermospheric density model jb2008 using new solar and geomagnetic indices. In *AIAA/AAS astrodynamics specialist conference and exhibit*, page 6438, 2008.

[18] T.S. Kelso. Celestrak. `https://celestrak.com/SpaceData/`, 2023.

[19] H. Klinkrad. Discos - esa's database and information system characterising objects in space. *Advances in Space Research*, 11:43–52, 1 1991.

[20] Ignacio Oguiza. tsai - a state-of-the-art deep learning library for time series and sequential data. Github, 2023.

[21] J. Howard and S. Gugger. *Deep Learning for Coders with fastai and PyTorch*. O'Reilly Media, 2020.

[22] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018.

[23] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.