



LSTM-based maneuver detection for resident space object catalog maintenance

Riccardo Cipollone¹  · Carolin Frueh² · Pierluigi Di Lizia¹

Received: 28 May 2024 / Accepted: 5 March 2025

© The Author(s) 2025

Abstract

Maintaining a catalog of Resident Space Objects involves a wide range of activities, exploiting measurement processing to estimate and update orbits, in an effort to achieve a comprehensive picture of the Near-Earth environment status in terms of human-made objects. The first step required to update a cataloged orbit is to correctly associate it with new observations. This task is particularly hampered by the growing presence of active satellites that are capable of maneuvering. Nonetheless, a large part of them perform routine maneuvers to preserve their orbit, defining very regular patterns. Given the significant quantity of past orbital and maneuvering data about tracked objects as the main by-product of catalog maintenance, the main focus of this work is to effectively exploit them by training Machine Learning models. The objective is to infer the probability of a target maneuver within a specific time horizon, using a Long Short-Term Memory Recurrent Neural Network, specialized in processing time sequences. Two distinct maneuver detection modules are developed and tested on real LEO object data to, respectively, extend a target's control history and predict how likely a maneuver will happen in the immediate future.

Keywords Long Short-Term Memory · Maneuver detection · Data-driven classification · Space Surveillance and Tracking

1 Introduction

As Near-Earth space becomes progressively more accessible and used for both commercial services and scientific missions, a newfound interest in its monitoring and modeling has spread to improve safety in space operations. In this framework, Space Traffic Management plays a key role, tracking the evolving population of Resident Space Objects (RSO) and contributing to a comprehensive picture of its status, namely Space Situational Awareness [1]. The main reason behind this effort consists in avoiding catastrophic collisions and fragmentations that would impair any further human activity in the Near-Earth environment. A significant part of this discipline focuses on intricate pipelines of measurement processing aimed at building and maintaining an RSO catalog by continuously updating target orbits. Several international agencies keep some of these catalogs publicly available, often

Carolin Frueh and Pierluigi Di Lizia contributed equally to this work.



employing standard data types such as Two-Line Elements (TLE) as a baseline to promote sharing and information exchange. Some popular examples are Celestrak¹ and the Space-Track website.²

Once a new observation is acquired and properly processed, the first problem to face is understanding whether it belongs to a known, cataloged object or to a target that has never been observed before. In order to do so, an association test is set up between a new observation and a set of candidate targets. This process, called track-to-orbit correlation, is usually based on metrics for which a correlation hypothesis is accepted or discarded. As a consequence, selecting the right quantity to formulate a score is fundamental to the process's robustness. Traditional correlation metrics are defined via statistical distances between distributions in a common space, often being the measurement one (such as the squared Mahalanobis distance-based correlation index reported in [2]).

By using this kind of approach, active objects are not specifically taken into account, neglecting the possibility of a maneuver happening between the last available orbital information of a target and the observation epoch. When operational objects are included in the analysis, in fact, maneuver detection becomes a key task to embed in a Space Surveillance and Tracking (SST) pipeline. Its outcome often provides crucial context on cataloged objects, resulting in increased robustness of the correlation phase. Most techniques built around track-to-orbit correlation aim at testing the hypothesis that the target has performed a maneuver in the time span between its last available state and an incoming observation, ranking them according to the cheapest hypothetical maneuver. In order to constrain the problem as much as possible, some strong assumptions are usually made on the control policy, supposed low thrust [3, 4] or impulsive energy optimal [5]. In this way, optimal control problems can be solved between well-defined boundary conditions and the resulting cost can be used as an alternative correlation metric.

An easily forgotten additional source of information in this phase is the history of each correlation candidate, including its orbit and pattern of life, if available. Exploiting past data of a known RSO helps in providing contextual information on its maneuvering probability within a specific time frame, given a maneuvering pattern. There are several works focusing on a statistical analysis performed on orbital data histories to look for events that can be related to maneuvers. Some of them leverage traditional statistics-based techniques and Keplerian osculating elements associated with orbit shape and orientation to retrieve a metric, usually chosen as a deviation of the Keplerian element of interest from a local curve fit, or its time derivative, and perform a thresholding process [6, 7]. The main limitation of these approaches resides in the case-dependent tuning of the threshold used to distinguish the target events from the background, requiring constant monitoring. The method developed in [8] tries to mitigate this shortcoming via automatic policies for threshold update. This problem seems to be partially solved by adopting Machine Learning algorithms exploiting a great deal of cataloged information about known RSOs and intrinsically encoding their maneuver-related variations and patterns to classify new orbital data samples. Both unsupervised learning [9, 10], involving clustering linked to the variation magnitude of the monitored parameters, and supervised techniques spanning from basic Support Vector Machine and Random Forest algorithms [11, 12] to Convolutional [13, 14] and Recurrent Neural Networks (RNN) [15], have been investigated showing promising results, especially when RNNs prediction capability is combined with a detection or classification step to understand whether a maneuver has to be expected within a target time horizon. This work stems from this last research path, exploiting Long Short-Term Memory (LSTM) neural networks, specifically designed to process sequential data and to effectively map their time dependence, to investigate their detection capabilities and how they can be turned into a reliable maneuver prediction tool. This is accomplished by designing two distinct models: the former aims at augmenting an already available maneuver history with new, accurately labeled, data as they are progressively published, while the latter focuses on a more real-time application consisting in inferring how likely a maneuver will happen in a time horizon spanning from the latest available RSO orbital information and the immediate future. The target maneuvers considered for the reported methods are limited to the typical ones performed by Low Earth Orbit (LEO) objects, mostly aimed at orbit

¹ <https://celestrak.org>, last accessed: 2024-05-28.

² <https://www.space-track.org>, last accessed: 2024-05-28.

raising and plane preservation. This specific choice is due to the restrained set of publicly available maneuvering histories, basically consisting in the International Laser Ranging System (ILRS) database.³

The structure of the paper is the following: Sect. 2 delves into the mathematical background and methods utilized to shape the technique. Therefore, Sect. 3 outlines the approach, detailing the pre-processing pipeline followed to generate the datasets used to train the mentioned maneuver detection models, with a particular focus on the structure design choices made to build effective networks. Section 4 thoroughly describes the method's testing procedure, assessing its performance through the typical metrics used for classification tasks. Finally, Sect. 5 summarizes and discusses the findings, highlighting the existing limitations of the technique and elaborating on the current efforts to overcome them.

2 Data and model description

This section describes the theoretical content behind the orbital data format at the base of the maneuver detection models' training process, how spacecraft maneuvers affect a target trajectory, and how this effect can be spotted thanks to a convenient set of coordinates. Besides, some notions about the kind of sequential neural network architecture employed to encode the RSO trajectory evolution are also dealt with, to give a comprehensive overview of the concepts used to devise the method.

2.1 Orbital data and propagation

The TLE is a data format that encompasses orbital elements and information pertaining to an RSO at a specific moment in time. It is the most publicly available data format and, despite its poor accuracy and the constraints related to the method they are derived from, it is by far the most widely used.

A typical TLE is composed of two 69-character lines of data, which can be processed to derive the RSO's mean orbital parameters generated in the geocentric coordinate system True Equator Mean Equinox (TEME). A piece of additional information is also provided by this file format, namely B^* (BSTAR). It can be defined as an adjusted value of the ballistic coefficient B using the reference atmospheric density value, denoted as ρ_0 at one Earth radius R_E :

$$B^* = \frac{R_E \rho_0}{2B} \quad (1)$$

This quantity is not physical and usually includes every contribution that is not properly modeled by the dynamics.

As reported in [16], TLE files stem from an Orbit Determination (OD) process that exploits observations, coming from sensors of different nature and belonging to a known RSO, that are usually collected several times a day. In this way, the target orbit contained in the catalog is updated. This procedure must rely on a specific propagator and a dynamical model choice to propagate a reference state, representing the latest orbital information on the object, to the observation epochs. The one chosen for most catalog maintenance activities is the Simplified General Perturbations-4 (SGP4) [17, 18]. This means that whenever TLEs are used to retrieve any orbital state information, this represents the optimal and most consistent choice to propagate it, constraining the operators to stick with its limitations. The involved dynamical model together with the dependency on both observation and OD process quality causes this kind of orbital data to feature inconsistent accuracy, also affected by the limited number of digits available. Moreover, since no uncertainty information is usually provided when published, there is little that can be done to try and quantify it. A valid alternative to this widely available file

³ <https://ilrs.gsfc.nasa.gov/>, last accessed: 2024-02-09.

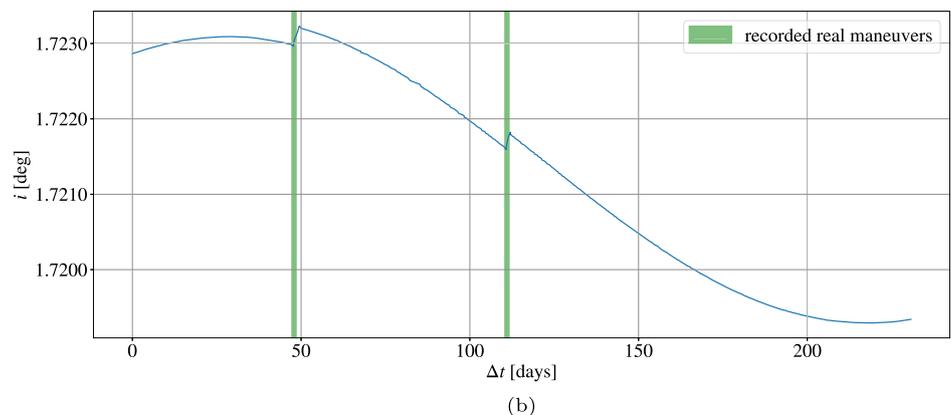
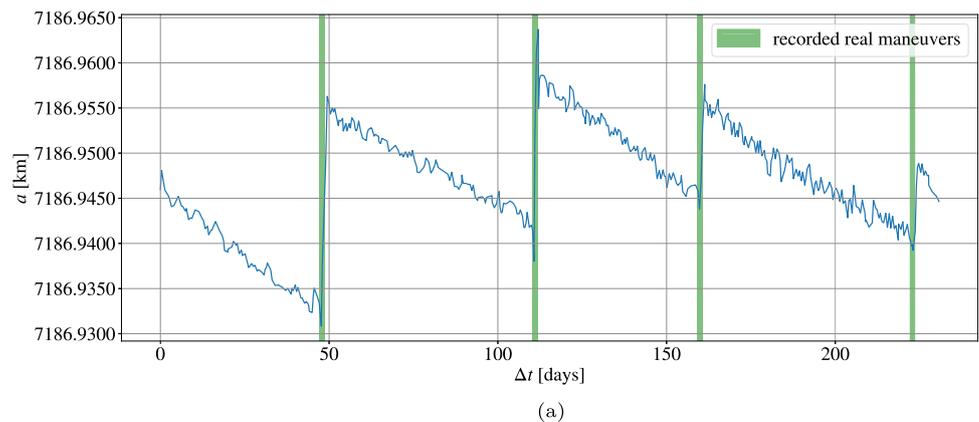
format is represented by the Orbital Mean Message (OMM), developed by the Consultative Committee for Space Data Systems (CCSDS) [19], which can provide the same information as a TLE without any limitation on its digits. Moreover, even if covariance information can be easily included, it is not provided in most publicly available data.

Focusing on the maneuver detection task, not only the quality but also the sampling frequency of TLEs is a factor to take into account. If orbital data are published often enough, high-frequency contributions to the target dynamics start to show, causing maneuver-related peaks usually characterizing orbital parameter trends to sink into high-amplitude oscillations. On the other hand, if states are provided with lower frequency, an averaged version of the target orbital parameters trend may be the only information available, consequently smoothing out maneuvering peaks.

2.2 Maneuvers and orbital parameters

TLE files are thus used to extract the target orbit history and evolution, embedding both the ballistic dynamics, with its major orbital perturbations, and external contributions that cannot be properly modeled without having a deeper knowledge of the object at hand. Maneuvers are the perfect example of the latter, as well as the main reason why traditional propagation could benefit some data-driven assistance. Active control can be usually detected by looking at the evolution of some specific Keplerian elements of the target. They can be grouped into two categories: the shape-related ones, being semi-major axis a and eccentricity e , and the ones affecting the orientation of the orbit, both in plane (anomaly of the pericenter ω) and out of plane (inclination i and right ascension of the ascending node Ω). According to the kind of maneuvers performed, some or all of them could present peaks or sudden trend variations standing out from the usual behavior: orbit-raising maneuvers mostly affect a , plane-preserving maneuvers can be detected by looking at i and Ω . An example of both kinds of control

Fig. 1 Semi-major axis trend as a function of time, reported in days from January 1st, 2020, is reported in (a) showing a correlation between recorded past maneuvers and sudden positive variations. The same can be noticed in (b), where the inclination trend for the same time span highlights peaks in inclination corresponding to the only orientation-changing maneuvers



actions is reported in Fig. 1, showing the semi-major axis and inclination trend variations correlated with real past maneuvers performed by Sentinel-3B in 2020. The maneuver events reported in the figure are taken from the ILRS website just as the information used to build the labels and to train the models shown in this paper.

Specifically for maneuver detection purposes, a combination of these parameters can also be adopted to better highlight these variations, using angular momentum \mathbf{h} norm and direction, or specific orbital energy E . Alternative coordinate systems can be employed as well according to the peculiar operational objectives of a target maneuver. A clear example is represented by station-keeping maneuver detection in the geosynchronous orbital regime [20, 21], where active control focuses on keeping the target within specific ranges of geographic longitude and latitude.

2.3 Recurrent neural networks

Whenever a meaningful amount of structured data is available, and their features and distribution are too intricate to model, Machine Learning becomes a convenient option, offering a wide range of techniques to retrieve underlying patterns and information. Most of them rely on an optimization phase, usually referred to as training, in which the parameters of a model are adjusted to fit the available data as much as possible. Once this optimal tuning is accomplished, given that the dataset is representative enough of the underlying distribution, the model can infer information about never-seen data from the same distribution. In particular, this work focuses on the application of Neural Networks (NN), a specific kind of method consisting in building a modular function in terms of neurons and layers, to fit a data distribution. Even if the two basic tasks that NN accomplish are regression and classification, a wider range of complex activities can be derived from them, including object detection [22] and semantic segmentation [23] when dealing with Computer Vision, or anomaly detection [24] and error prediction [25] as far as Sequential Models are concerned. A neuron represents the basic unit of a neural network, embedding a linear combination of a given input with weights \mathbf{W} and biases \mathbf{b} , being the parameters that are updated throughout the training process, and a nonlinear modulation of the output via an activation function $\mathbf{g}(\mathbf{W}\mathbf{x} + \mathbf{b})$ aimed at transmitting the information to the following unit. Neurons are usually stacked in layers of different sizes and roles, each one embedding features at different scales and detail level. Stacking layers together allows for better coverage of the data distribution nonlinear features at the expense of higher structural intricacy and computational resources needed to train such a model. This kind of complex model flows into the category of deep learning, one of the current state-of-the-art data-driven techniques. To exploit the sequential structure of the data composing the orbital history of a target, the choice narrows down to the Recurrent Neural Network (RNN) category, capable of embedding a sense of time sequentiality when processing an input sample [26]. This kind of supervised NN is based on an array of cells exploited to transmit knowledge about the history within the input sequence through a hidden state \mathbf{h} . The output of a generic cell \mathbf{y}_t as well as its hidden state \mathbf{h}_t is then affected by both the current input value \mathbf{x}_t through the weights $\mathbf{W}_{y,x}$ and $\mathbf{W}_{h,x}$ and \mathbf{h}_{t-1} , derived from the past time step, via $\mathbf{W}_{y,h}$ and $\mathbf{W}_{h,h}$ as follows:

$$\mathbf{y}_t = \mathbf{f}_y(\mathbf{W}_{y,x}\mathbf{x}_t + \mathbf{W}_{y,h}\mathbf{h}_{t-1} + \mathbf{b}_y) \tag{2}$$

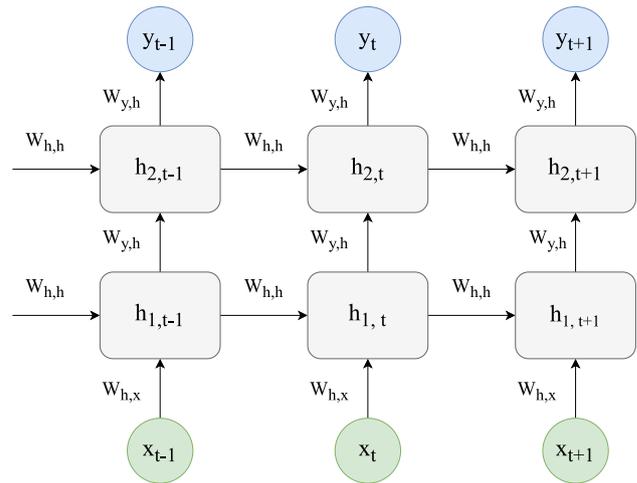
$$\mathbf{h}_t = \mathbf{f}_h(\mathbf{W}_{h,x}\mathbf{x}_t + \mathbf{W}_{h,h}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{3}$$

where \mathbf{b}_y and \mathbf{b}_h are the biases.

The working principle of a generic multiple-input multiple-output RNN is displayed in Fig. 2.

A known limitation of traditional RNNs is that they are not able to encode long-term time dependencies due to errors used to build the loss function vanishing when they are back-propagated over a certain number of time steps. As information about them is progressively lost, the associated training times get increasingly longer. One of the most popular solutions to this issue consists of the Long Short-Term Memory cells, an alternative building

Fig. 2 Architecture scheme of a stacked RNN



block to design an RNN, enabling the preservation of both long- and short-term contributions to the current time step by using specialized gates (with their own weights and biases) and a memory cell \mathbf{c} to filter and modulate the information according to its source and distance in time [27]. The LSTM cell internal structure is composed of 3 gates with different roles, interacting with each other: the input gate \mathbf{i}_t , taking information from the previous hidden state \mathbf{h}_{t-1} and current input \mathbf{x}_t , the output gate \mathbf{o}_t , with a similar structure but independent weight and bias terms, and the forget gate \mathbf{f}_t , modulating information according to hidden state distance in time:

$$\mathbf{i}_t = \sigma_i(\mathbf{W}_{i,x}\mathbf{x}_t + \mathbf{W}_{i,h}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{4}$$

$$\mathbf{o}_t = \sigma_o(\mathbf{W}_{o,x}\mathbf{x}_t + \mathbf{W}_{o,h}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{5}$$

$$\mathbf{f}_t = \sigma_f(\mathbf{W}_{f,x}\mathbf{x}_t + \mathbf{W}_{f,h}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{6}$$

where σ are sigmoid activation functions, ranging between 0 and 1, expressed as:

$$\sigma = \frac{1}{1 + e^{-x}} \tag{7}$$

and the \mathbf{b} terms are biases. The above gates define how \mathbf{h} and \mathbf{c} propagate through time. The long- and short-term memory contributions are defined as follows:

$$\mathbf{c}_{t-1}^S = \tanh(\mathbf{W}_s\mathbf{c}_{t-1} + \mathbf{b}_s) \tag{8}$$

$$\mathbf{c}_{t-1}^L = \mathbf{c}_{t-1} - \mathbf{c}_{t-1}^S \tag{9}$$

while the candidate memory cell for the current time step is:

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{c,x}\mathbf{x}_t + \mathbf{W}_{c,h}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{10}$$

where \tanh is the hyperbolic tangent activation function:

$$\tanh(\mathbf{x}) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{11}$$

The current memory cell and hidden state are then computed by using the gates in Eqs. 4 to 6:

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tilde{\mathbf{c}}_t \tag{12}$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t) \tag{13}$$

The general structure of an LSTM cell is shown in Fig. 3.

3 Data-driven maneuver detection

The topic of this section consists in the explanation of the two methods developed to detect maneuvers performed by an active RSO, given its history and a sequence of past ones. The whole pipeline is analyzed in both cases, from dataset generation to the training phase setup.

3.1 Dataset generation and pre-processing

As with every Machine Learning application, the first step is to build a dataset to train the network to achieve its designed task. In this case, the raw materials are TLEs and maneuver histories in the form of a sequence of time windows composed of an initial and final date in Coordinated Universal Time (UTC). The task to be accomplished is the same for both models, and they mostly share the same kind of input, so the processing to generate them is detailed as a common first step while the slight differences will be highlighted throughout the pipeline. The description is structured following the flow reported in the block diagram displayed in Fig. 4, reporting the overall process.

In order to properly exploit the advantages of an LSTM-based RNN, the sample fed to the network shall be a time sequence of elements; thus, the first step consists in processing the whole series of TLEs (block II of the diagram) to build a sample of fixed size. The coordinates system chosen to translate TLE data can be different according to the task: while Cartesian states can help with the task of state prediction due to their strongly periodic trends, the models developed in the case at hand are based on Keplerian elements (extracted as part of block P1), showing more pronounced maneuvering peaks in some of them, as already stated in Sect. 2.2. Considering the specific two-body application of the method, the more general Cartesian coordinates advantages, such as their lack of singularities and capability to describe the system regardless of its dynamics, are traded for the capability to highlight control actions. The structure of the arrays composing the fixed-size input sequence represents the major difference between the two maneuver detection algorithms. The first, called Net_{off} , is based

Fig. 3 Architecture scheme of an LSTM cell

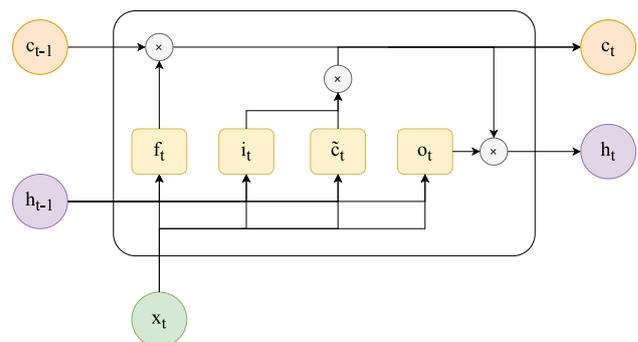
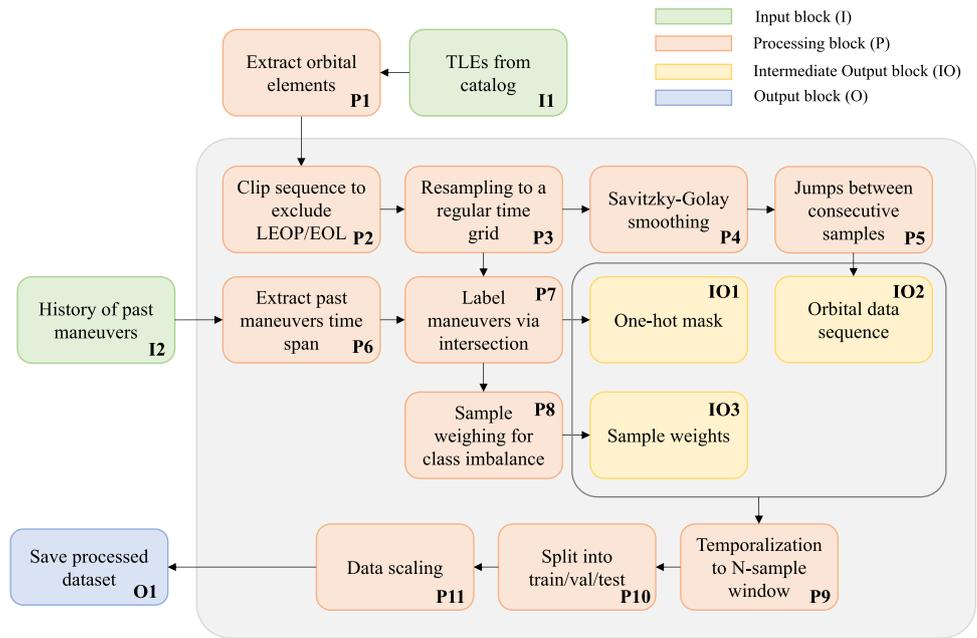


Fig. 4 Workflow diagram representing the data pre-processing pipeline used to generate the dataset for the maneuver detection task



on an array of elements augmented with their variation between consecutive epochs (t_k, t_{k+1} for a generic k -th sample), similar to the one reported in [15]. This implies that, given a time span composed of TLEs from 1 to d , complete knowledge of TLEs from 1 to $d + 1$ is required to properly define the input structure. As a consequence, this version of the model is suitable for offline use only, since no information about a hypothetical $d + 1$ TLE would be available if the previous d TLEs were the latest ones available for a specific active RSO. No maneuver prediction in the immediate future could thus be performed. Conversely, the latter algorithm, called Net_{on} , does not rely on any variation across time, enabling the detection of maneuvering events from the latest orbital state onwards, with no requirements on the knowledge of future TLEs.

The whole orbital history of the target is then represented as a $(10 \times \bar{N})$ time series in the first case (excluding TLE B^* and true anomaly θ due to its marginal role in this specific application), with

$$\mathbf{e}_{kep,\Delta}^i = [a^i, \delta a^i, e^i, \delta e^i, i^i, \delta i^i, \Omega^i, \delta \Omega^i, \omega^i, \delta \omega^i] \tag{14}$$

being its i -th element and \bar{N} being the size of the original sequence. As for the second model, the basic unit shaping the $(5 \times \bar{N})$ original sequence is composed of the only Keplerian elements (excluding again the true anomaly θ):

$$\mathbf{e}_{kep}^i = [a^i, e^i, i^i, \Omega^i, \omega^i] \tag{15}$$

The following step consists in discarding inaccurate TLEs by keeping only the ones being more than half the orbital period apart from each other. As stated in [8], they are usually two consecutive outcomes of the same orbit determination process, and the second one in chronological order represents a correction of the first one. Once this step is performed, every sample belonging to the satellite Launch and Early Orbit Phase (LEOP) and/or to its End Of Life (EOL) is filtered out as well, as they represent the boundaries of the operational life targeted by the proposed method (block P2).

As highlighted in [28], LSTM-based neural networks usually make the strong assumption of dealing with uniformly sampled data so, as far as a sparse and irregularly sampled dataset is concerned, this assumption becomes an intrinsic source of error if ignored. A common workaround to this issue consists in resampling the original time series and turning it into a hybrid dataset (as shown in block P3). Consequently, a uniform time grid

has been defined with a time step equal to 21,600 s (0.25 days), using Piece-wise Cubic Hermite Interpolating Polynomials (PCHIP [29]), particularly effective in preserving the shape of the original data as much as possible, to generate a \mathbb{C}^1 resampled signal. This method is preferred to an approach based on propagation, meaning using SGP4 due to the TLEs composing the dataset, that would imply two main drawbacks due to sensitivity to the time grid density: the former is linked to higher-frequency contributions of the two-body perturbed dynamics showing up, with their magnitude causing active control-related spikes to sink in artificial oscillations; the latter instead lies in how the original orbital information is deleted or replaced by synthetic samples due to the initial state selected for propagation (usually the closest to the desired time label). Moreover, PCHIP interpolation is particularly consistent with the very nature of the NN, since it does not necessarily learn the physics and the dynamics used to propagate a state to a given epoch, but it just tries to predict what the next orbit determination process outcome, namely the TLE state, will be.

In order to reduce the intrinsic noise of the sequence and to restrain the influence of the outliers on the network training process, a smoothing step is added to the pipeline exploiting Savitzky–Golay polynomial filters [30] with a window dimension small enough not to lose local features (block P4 of the chart). In this case, 5 consecutive samples have been chosen and a polynomial of degree 2 has been used to fit them.

Once the uniform orbital history covering the whole target’s lifespan is obtained (block IO2), it is scanned via a sliding window algorithm, usually referred to as temporalization (reported in block P9), where window size d and stride s are hyperparameters, to obtain a sequence of constant length.

The samples belonging to the processed version of the two datasets take the following shape:

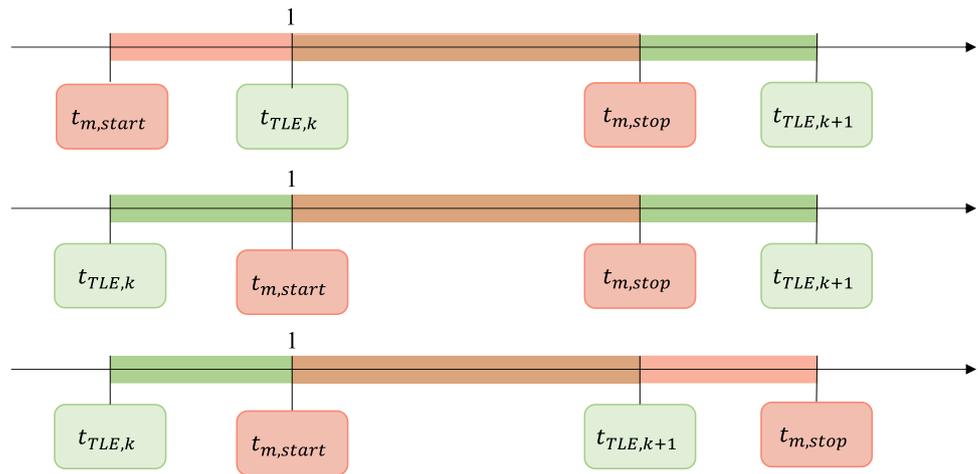
$$\begin{aligned}
 \mathbf{E}_{off,i} &= \begin{bmatrix} a_1^i & a_2^i & \dots & a_k^i & \dots & a_d^i \\ \delta a_1^i & \delta a_2^i & \dots & \delta a_k^i & \dots & \delta a_d^i \\ \omega_1^i & \omega_2^i & \dots & \omega_k^i & \dots & \omega_d^i \\ \delta \omega_1^i & \delta \omega_2^i & \dots & \delta \omega_k^i & \dots & \delta \omega_d^i \\ i_1^i & i_2^i & \dots & i_k^i & \dots & i_d^i \\ \delta i_1^i & \delta i_2^i & \dots & \delta i_k^i & \dots & \delta i_d^i \\ \Omega_1^i & \Omega_2^i & \dots & \Omega_k^i & \dots & \Omega_d^i \\ \delta \Omega_1^i & \delta \Omega_2^i & \dots & \delta \Omega_k^i & \dots & \delta \Omega_d^i \end{bmatrix} \\
 \mathbf{E}_{on,i} &= \begin{bmatrix} a_1^i & a_2^i & \dots & a_k^i & \dots & a_d^i \\ e_1^i & e_2^i & \dots & e_k^i & \dots & e_d^i \\ i_1^i & i_2^i & \dots & i_k^i & \dots & i_d^i \\ \Omega_1^i & \Omega_2^i & \dots & \Omega_k^i & \dots & \Omega_d^i \\ \omega_1^i & \omega_2^i & \dots & \omega_k^i & \dots & \omega_d^i \end{bmatrix}
 \end{aligned} \tag{16}$$

where $i = 1, \dots, N$ and N is the new size of the dataset, which can be computed with a simple rule as a function of window size d , stride s , and the original number of samples \bar{N} :

$$N = \frac{\bar{N} - d + 1}{s} \tag{17}$$

The same temporalization process is applied to the labels corresponding to each sample. They are defined as one-hot vectors (block IO1) to distinguish between the positive and negative categories involved in the binary classification task, being “maneuver happening” or “no maneuver happening” in this case. The process to define

Fig. 5 Algorithm designed for the labeling process of the orbital state sequence to perform classification. The green window represents the time spanning between the two consecutive TLEs of the couple k , while the red one covers the firing arc associated with the maneuver m . The 3 reported timelines show the cases in which a positive label is assigned to the k -th state



the dynamical states involved in an active control arc consists in scanning each couple of consecutive samples from both original states sequences, reported in Eqs. 13 and 14, and comparing their timestamps in terms of UTC epochs, with the time spans reported in the maneuver histories from ILRS data (extracted in block P6 of the chart). In order to label a sample k , a loop over the list of past maneuvers is designed to look for intersections between the selected time window $[t_{TLE,k}, t_{TLE,k+1}]$ and any m -th control event time window $[t_{m,start}, t_{m,stop}]$ (block P7 of the diagram). Figure 5 shows the 3 possible intersection outcomes, leading to a positive label 1 to be assigned to sample k . Using this logic, the value assigned to each sample refers to its immediate future behavior. Shifting this to an operative scenario, it translates to assessing if the time spanning from the latest TLE of a cataloged RSO to the next, not yet recorded one contains a maneuver.

One last step is mandatory to shape the specific maneuver detection application into a classification problem. As a matter of fact, when comparing satellite routine maneuvering frequency with the one related to the TLE generation process, maneuvers can be considered “rare” events. Taking Sentinel-3A dataset as an example, on average only 1 in every 152 samples is involved in a maneuver (according to the mentioned logic), so a clear case of class imbalance is embedded in the problem. One way to address this issue is relying on class weights and treating samples differently during the whole training process according to the kind of class they are associated with (block P8). In order to define the two weights in this case, the majority and minority classes have to be identified. Being w_1 and w_0 the weights assigned, respectively, to the positive and negative class, they are computed through the following formula, implemented in the sci-kit learn library from the work in [31]:

$$\mathbf{w}_c = \frac{n_s}{n_c \mathbf{f}_c} \tag{18}$$

where n_s is the number of samples, n_c represents the number of classes, and \mathbf{f}_c is a vector containing the occurrence of each class as its elements. The operation used to obtain the resulting class weight vector \mathbf{w}_c is thus performed element-wise for each class contained in \mathbf{f}_c .

Furthermore, due to the kind of sequential NN involved, class weights have to be turned into sample weights (block IO3), so that the samples composing each input sequence can be weighted accordingly when plugging them into the corresponding loss function term. The sample weight is then shaped as the label assigned to a generic sample i , mirroring the disposition of the two classes. For the sake of clarity, given the i -th one-hot vector:

$$\mathbf{o}_i = [0, 1, \dots, 0, \dots, 1] \tag{19}$$

The corresponding weight mask will be:

$$\mathbf{w}_i = [w_1, w_0, \dots, w_1, \dots, w_0] \tag{20}$$

Once this mask and weights setup is performed, the whole dataset can be split into training, validation, and test sets (as reported in block P10 of the chart). In this case, 70% of the samples are devoted to the training process, 20% to the validation, while the remaining 10% is used for the testing phase.

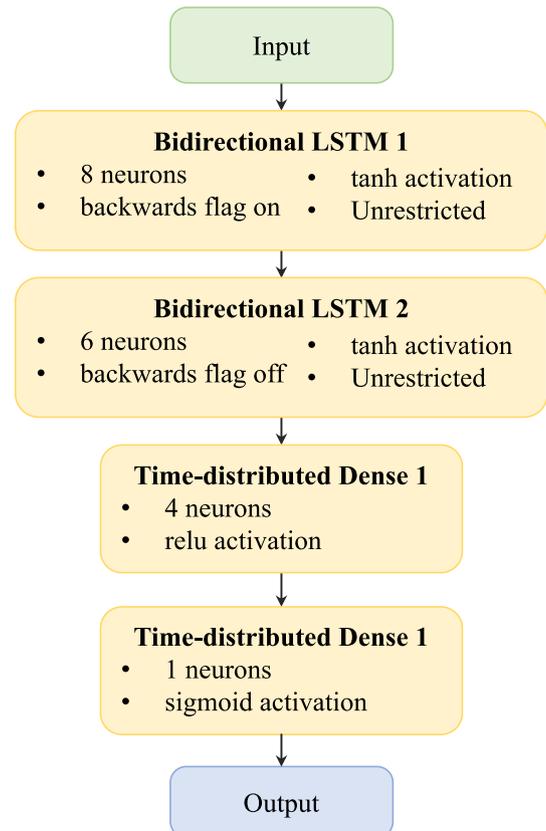
The last fundamental step to obtain a suitable dataset for a NN is data scaling, so that the model can work with values with limited span, usually between 0 and 1 or within a symmetric interval around 0. For this specific application, a standard scaler has been used to normalize the inputs with the following process (block P11):

$$e_{i,scaled} = \frac{e_i - \mu_e}{\sigma_e} \tag{21}$$

where e_i is a generic i -th sample component, normalized over its mean μ_e and standard deviation σ_e . An important remark for this algorithm is that the statistics for each component of a sample are computed just by using the training set, not to include information that should not be available in an operational scenario. This can make the difference if the distribution is subject to abrupt enough changes between training and test set samples, since the dimensionalization of the outputs, being the inverse transform of Eq. 20, is based on the only information contained in the training set.

The target selected to generate the results reported in the paper is Sentinel-3A, a LEO active satellite, considered a representative sample of the small population of objects provided with a maneuver history in the ILRS public database. As for all of its past TLE orbital data, they have been instead downloaded from Space-Track website.

Fig. 6 Network structure used to accomplish the maneuver detection task



3.2 Neural network structure

The maneuver detection task is based on the analysis of a fixed-size sequence of orbital parameters (and their variations over a time step) to output a one-hot vector of the same size representing the classification of each element belonging to the sequence. To achieve that, downstream to an iterative procedure to tune the network layers and parameters, an optimal structure has been found to be the one reported in Fig. 6, leveraging several tweaks and improvements in the standard LSTM cell described in Sect. 2.3. The first one is the concept of Bidirectional LSTM cells, introduced in [32], consisting in enabling a traditional cell, able to just learn the temporal patterns in a forward direction—going from past to future, to learn the other way around too. This is possible by adding a mirror RNN layer to the traditional LSTM one, doubling the corresponding parameters. While the original input sequence is passed unchanged to the first layer, its flipped version is fed to the mirror one so that the time pattern is learned symmetrically. This is proved to outperform traditional LSTM-based NN in speech recognition tasks [33]. The second adjustment consists in reversing the order of the input fed to the first layer of the network which, as detailed in [34], is proved to improve the training performance by establishing short-term dependencies between consecutive elements of a sample. This modification is expected to improve performance during back-propagation, when the initial input features, usually mitigated by consecutive gates between LSTM cells, are moved to the end of the sequence, preserving upstream information. An additional peculiar characteristic is referred to with the term “unrestricted” in the diagram, meaning that the entire output time sequence of an LSTM layer is given as input to the following one. A “restricted” layer instead provides the output corresponding to the only final time step. The former is used for the first two layers to transmit intermediate features to every LSTM cell of the following layers. The reason behind this resides in the encoder logic used to achieve element-wise classification, leading to the stack of two time-distributed layers of fully connected neurons. The time-distributed shell to these two dense layers allows the network to perform the same operations of a single dense layer to each temporal slice of the mentioned unrestricted input, providing the desired type of classification via sigmoid activation function (defined as in Eq. 7).

The expected input size for Net_{off} model is $(10 \times TS)$, where TS stands again for the number of fixed time steps. The neurons used to encode the 5 Keplerian elements and the jumps between consecutive epochs in the first LSTM-based layer are 8, decreasing in size with the following one, with an expected output size of $(6 \times TS)$. The two time-distributed dense layers scale it down even more, with sizes $(4 \times TS)$ and $(1 \times TS)$, finally giving a scalar score for each epoch comprised in the input time window. The only modification applied to the online model Net_{on} is a different input layer size of $(5 \times TS)$, excluding Keplerian element variations in time.

3.3 Training setup

Concerning the training phase, given the kind of task at hand, the Binary Cross-Entropy (BCE) loss function is chosen as the one to optimize. Due to the imbalance between the maneuvering and non-maneuvering classes, the sample weights described in the previous section are integrated into the loss function formulation, leading to a weighted version of BCE. The weighted loss function for a generic sample k is then defined as:

$$L_{BCE,w}(\mathbf{y}_k) = - \sum_{i=1}^N (w_1 y_{i,k} \log(p(y_{i,k})) + w_0 (1 - y_{i,k}) \log(1 - p(y_{i,k}))) \tag{22}$$

where N is the number of samples, $p(y_{i,k})$ represents the probability of the positive class (corresponding to a maneuvering sample) referring to the element i of sample k , w_1 being the corresponding weight used to rescale it, while the negative probability one is modulated using the weight w_0 .

The choice for the optimization algorithm lies on ADAM, representing the state of the art in terms of Gradient Descent-based techniques for NNs, as implemented in Keras library following the work in [35]. The batch size is

selected as equal to 1 in order to maximize the potential to find a meaningful minimum, investing more processing time in the training session to update the model weights with each single sample (adopting a pure Stochastic Gradient Descent). The number of epochs for each training session is set to an initial value of 50, but this represents an additional hyperparameter through which the model performance can be tuned. An early stopping policy is foreseen too, in order to prevent useless iterations in case the loss function value does not change enough between an arbitrary number of consecutive iterations, in this case set to 5.

Due to the kind of task, the loss function value does not give enough information about how the training is proceeding, so diverse metrics are monitored across the iterations. The ones chosen for the case at hand are binary accuracy (A), precision (P), and recall (R), being the same ones used to assess the model performance during testing with never-before-seen data. They are all defined in their standard version as:

$$\begin{aligned} A &= \frac{TP + TN}{TP + TN + FP + FN} \\ P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{23}$$

where TP stands for true positives, FP is the false positives, while TN and FN are true and false negatives. Accuracy is the most straightforward metric, showing how many correct predictions the model was able to make; precision hints at how many of the detected positives are real while recall represents the sensitivity of the model, quantifying how many of the real positives were detected.

As a consequence of the problem’s class imbalance, their weighted version has to be used as well in order to make non-maneuvering samples, being the strong majority, weigh less than maneuvering ones. They can be formulated as follows:

$$\begin{aligned} A_w &= \frac{w_1 TP + w_0 TN}{w_1 (TP + FN) + w_0 (TN + FP)} \\ P_w &= \frac{w_1 TP}{w_1 TP + w_0 FP} \\ R_w &= \frac{w_1 TP}{w_1 TP + w_1 FN} \end{aligned} \tag{24}$$

where, according to this definition, R_w coincides with the vanilla version R . A level of confidence equal to 99% is set to establish the minimum score threshold for which an element composing a sample is considered a true positive. A high value of confidence allows for more robustness to false positives, decreasing the sensitivity of the method, and corresponding to lower R . The lower the threshold is set the higher the risk of false positives will be, with a consequent decrease in P . The output score provided by the last layer sigmoid activation function (made binary by the above-mentioned thresholding process) can be considered as a reference value to compute a maneuvering probability value between 0 and 1. This alternative approach could prove useful in conditioning further sources of information to combine them into a comprehensive index.

As already stated in Sect. 3.1, the model is trained with Sentinel-3A data, consisting of 5404 samples (from February 16th 2016, 20:59:45.170 to July 11th 2023, 03:23:06.021). The PCHIP-resampled sequence of its Keplerian parameters is reported in Fig. 7.

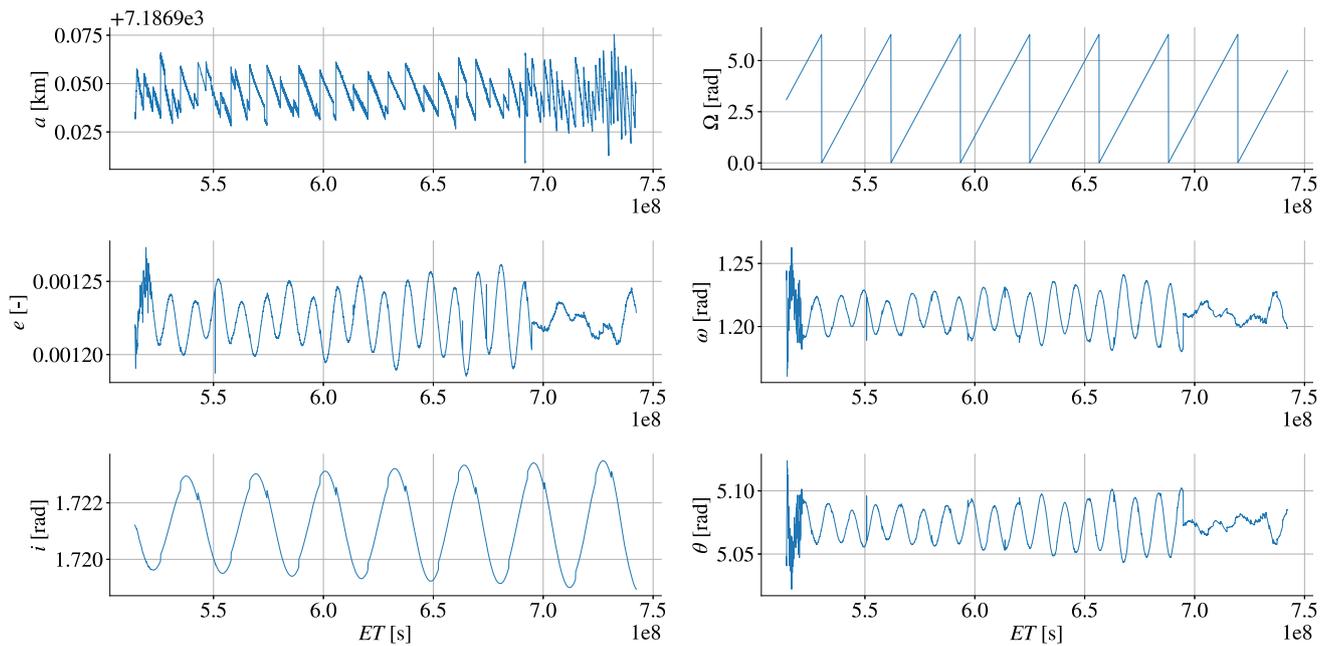


Fig. 7 Sentinel-3A Keplerian elements after the PCHIP-based resampling step are represented as a function of time in ET, showing how the shape of the original sequence is preserved while granting a denser time grid

4 Results

The testing phase for a standard classification task mainly consists in assessing how precise and sensitive the trained model is to data that it has never seen before. In the specific case of binary classification, such as the maneuver detection algorithm developed for this work, the metrics to take into account are the already mentioned weighted accuracy, precision, and recall (as defined in Eq. 23).

A total of two testing scenarios are detailed in this section, the only difference between them being the amount of information given to the network as input. The former is fed with a set of 5 orbital parameters together with the corresponding jumps between consecutive samples, as showcased in Eq. 13. The models included in this scenario are designed for stand-alone usage, enabling a swift labeling process of new TLE data belonging to a single cataloged object. As for the latter, the input consists of the only 5 orbital parameters involved in the definition of Eq. 14, without any jump, in order to mimic operational conditions, in which there is no information about the jump between the very last element and a future unknown one.

4.1 Offline maneuver detection model

The first testing session focuses on the model trained via Keplerian elements and their variations. For the maneuver detection task, the only resampling technique that is considered viable is the PCHIP interpolation. The trends for each Keplerian element downstream to the resampling process are displayed in Fig. 7, preserving the parameter trends shape with respect to the original state sequence. The campaign is organized as a sensitivity analysis to the model’s input size d (resulting in the time steps TS in the network structure), spanning between 32 and 512 consecutive samples to understand how to trade off between network complexity, meaning number of parameters and past information provided, and inference performance. The outcome of the 5 training sessions performed on Sentinel-3A data is reported in Table 1.

From the values reported in the table, it is easy to notice the ascending trend in performance as the input size increases. This suggests that long-term connections between the samples composing a single input sequence are effectively modeled by the NN, helping it discern true positives among different features contained in the element

Table 1 Weighted binary accuracy A_w , precision P_w , and recall R_w values for input sizes from 32 to 512 samples, on Sentinel-3A satellite test set. In this case, the samples are composed of both Keplerian elements and their variations in time

Input size	A_w	P_w	R_w
Input size 32	99.87%	90.11%	89.90%
Input size 64	99.95%	95.83%	97.15%
Input size 128	99.98%	98.63%	98.63%
Input size 256	99.99%	99.43%	99.38%
Input size 512	99.99%	99.41%	99.53%

trends. In order to better visualize true positives, true negatives, false positives, and false negatives, the confusion matrices for each model are reported in Fig. 8.

Besides, a representation of the detections on the semi-major axis a trend corresponding to the best-performing model (512 input size) is displayed in Fig. 9, to visually understand what the models are actually computing.

4.2 Online maneuver detection model

The second bench test consists in assessing the performance of the mentioned 5 models trained with the same Sentinel-3A data history in terms of Keplerian parameters, resampled with PCHIP interpolation. This time though the variations between consecutive samples are excluded from the input. The input sizes, ranging again from 32 to 512 elements, are included in a sensitivity analysis leading to the figures listed in Table 2.

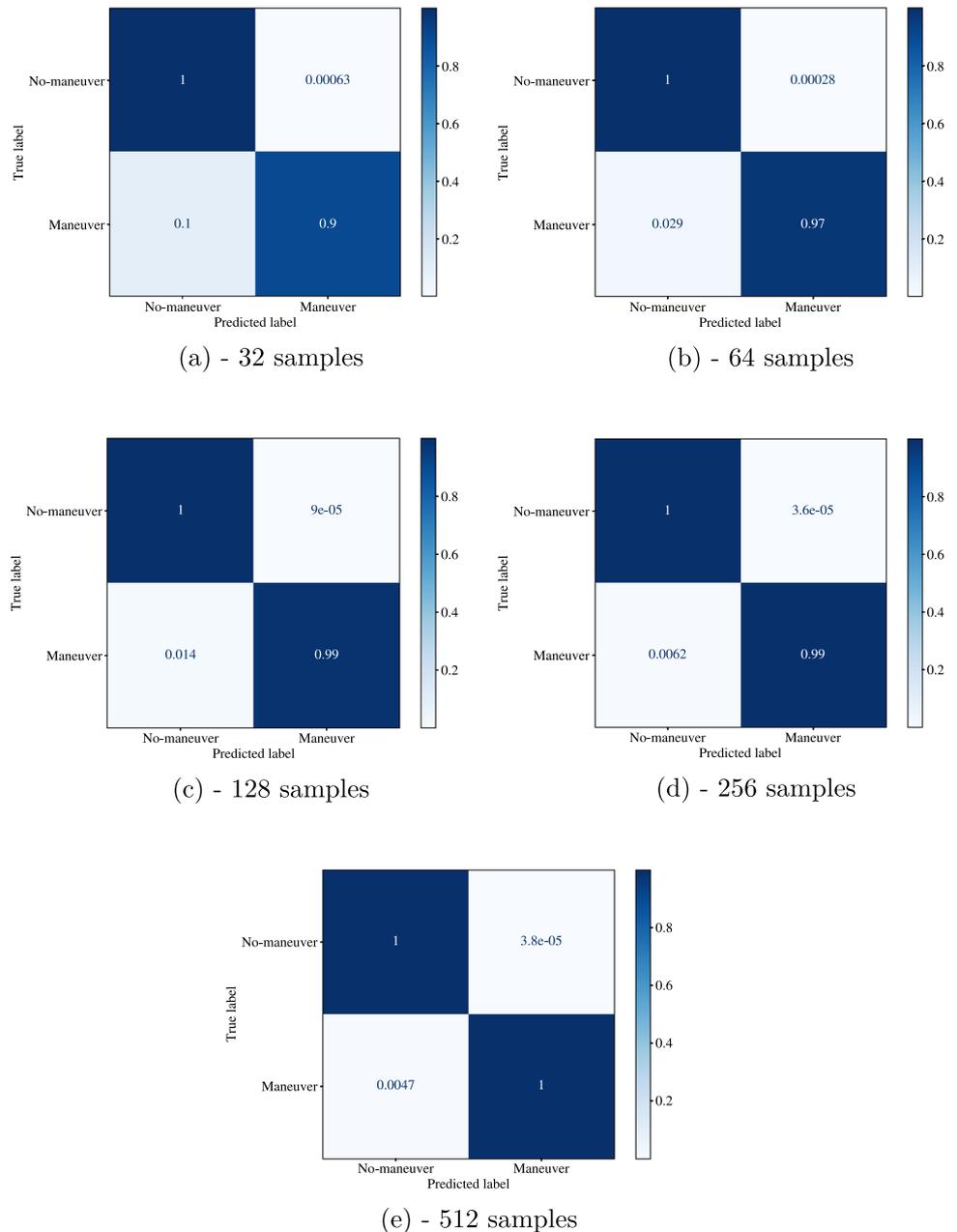
The tests confirm once again that increasing the amount of prior information embedded in the input sequence helps the network build long-term connections, allowing it to better detect the peaks associated with routine maneuvers. As a consequence, the scores increase with the highest number of samples (512). Comparing the outcomes with the previous stand-alone version of the algorithm, they exhibit drops of some percentage points, given that fewer encoded features are available in this case, providing nonetheless satisfactory results for the application.

The proportion between true positives, true negatives, false positives, and false negatives are reported once again in the form of confusion matrices for each of the trained models in Fig. 10.

4.3 Generalization capabilities

An additional testing campaign is conducted to assess generalization capabilities in the online maneuver detection model Net_{on} , considered the operational one in case maneuvers must be detected in newly published TLEs. The presented LSTM-based models are in fact designed to infer from data describing the orbit and maneuvering history of a single target, resulting in several training sessions for the method to work on multiple cataloged objects. This object-tailored approach is deemed the best under the assumption of no processing power or storage limitation constraining the training phase of the algorithm. The mentioned analysis is thus used to understand whether the model performance is preserved when data coming from satellites that are different from Sentinel-3A are used to train a fixed LSTM-based architecture. For this purpose, 3 additional space objects from the ILRS

Fig. 8 Plots show the performance of every model of the offline version of the algorithm, with varying input size (from 32 to 512 samples) and trained on Sentinel-3A data, in terms of confusion matrices, showing the TP, TN, FP, and FN cases



database are selected, spanning different levels of similarity to the baseline: Sentinel-3B, Jason-3, and Haiyang-2A. In order to understand how significant the generalization of the model is, it is deemed important to underline their similarities and differences from Sentinel-3A. While Sentinel-3B [36] and Haiyang-2A [37] feature very similar mission profiles, consisting of sun-synchronous orbits, at around 800 and 970 km altitude, respectively, and both around 98° inclination, Jason-3 follows a non-sun-synchronous orbit at about 1300 km altitude and 66° inclination [38]. As for their maneuver patterns, the average velocity expenses ΔV (stored in the database together with maneuvering epochs) are similar among Sentinel-3A, Sentinel-3B and Jason-3 (roughly 0.60 m/s, 0.65 m/s, and 0.44 m/s, respectively), while Haiyang-2A performs cheaper maneuvers (averaging around 0.0064 m/s). In general, the control actions are aimed at orbit raising and plane preservation, and their effect in terms of jumps in orbital parameters can be detected given the accuracy level of the TLE data used for each of the selected targets, regardless of the ΔV magnitude involved.

Fig. 9 A subset of the ground-truth maneuvers (in green) together with the ones predicted by the best-performing offline model (in red) is reported in the semi-major axis as a function of time in ET. The graph shows a false positive case as well, represented as a red band not intersecting with any green

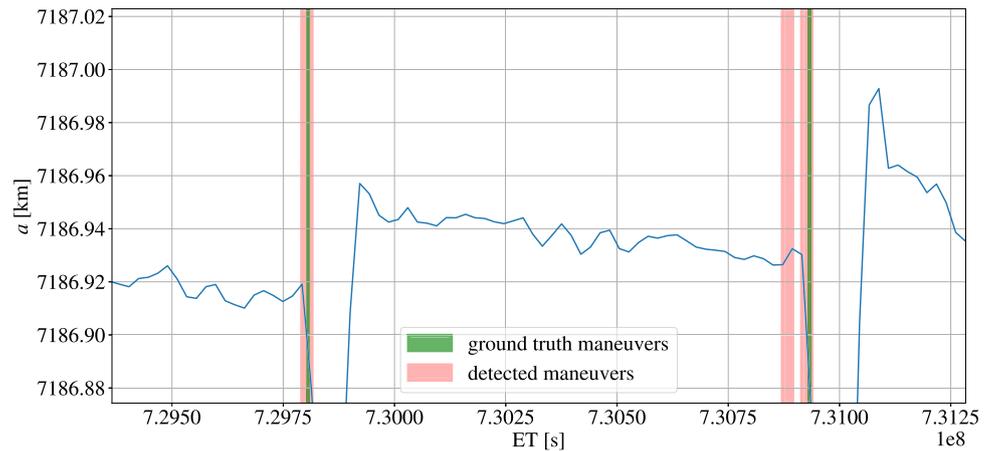


Table 2 Weighted binary accuracy A_w , precision P_w , and recall R_w values for input sizes from 32 to 512 samples, on Sentinel-3A satellite test set. In this case, the samples are composed of Keplerian elements only

<i>Input size 32</i>		
A_w	P_w	R_w
99.43%	54.17%	64.08%
<i>Input size 64</i>		
A_w	P_w	R_w
99.90%	94.45%	89.28%
<i>Input size 128</i>		
A_w	P_w	R_w
99.88%	90.23%	92.18%
<i>Input size 256</i>		
A_w	P_w	R_w
99.98%	98.30%	98.03%
<i>Input size 512</i>		
A_w	P_w	R_w
99.99%	98.21%	99.52%

The test is performed by training the 512-sample version of Net_{on} using data from the mentioned targets, pre-processed following the pipeline reported in Sect. 3.1, and assessing the models’ weighted accuracy, precision and recall. The scores for each target are displayed in Fig. 11, while the corresponding numerical values are reported in Table 3.

The similarity in terms of scores proves that the model manages to generalize to objects belonging to the same orbital regime as the baseline with drops in performance that are considered reasonable for the application. This is mostly due to the fact that the selected targets feature similar maneuvering patterns and orbital history, allowing for a single network design to effectively learn correlation between specific features in the time sequence and the presence of a maneuver. This behavior of course changes when moving to higher orbital regimes like MEO or GEO, where TLE accuracy may not be enough to make low-magnitude station-keeping maneuvers show as peaks in Keplerian elements. In this case, shifting to a more effective coordinate system (e.g., geographic coordinates for GEO space objects as reported in [14]) could highlight recurrent and observable patterns for the network.

4.4 Processing time

As widely known, the processing times associated with a neural network-based algorithm are unevenly distributed between training and inference. The former phase is the most demanding in terms of computational

Fig. 10 Performance of every model of the online version of the algorithm, with varying input size (from 32 to 512 samples) and trained on Sentinel-3A data, in terms of confusion matrices, showing the TP, TN, FP, and FN cases

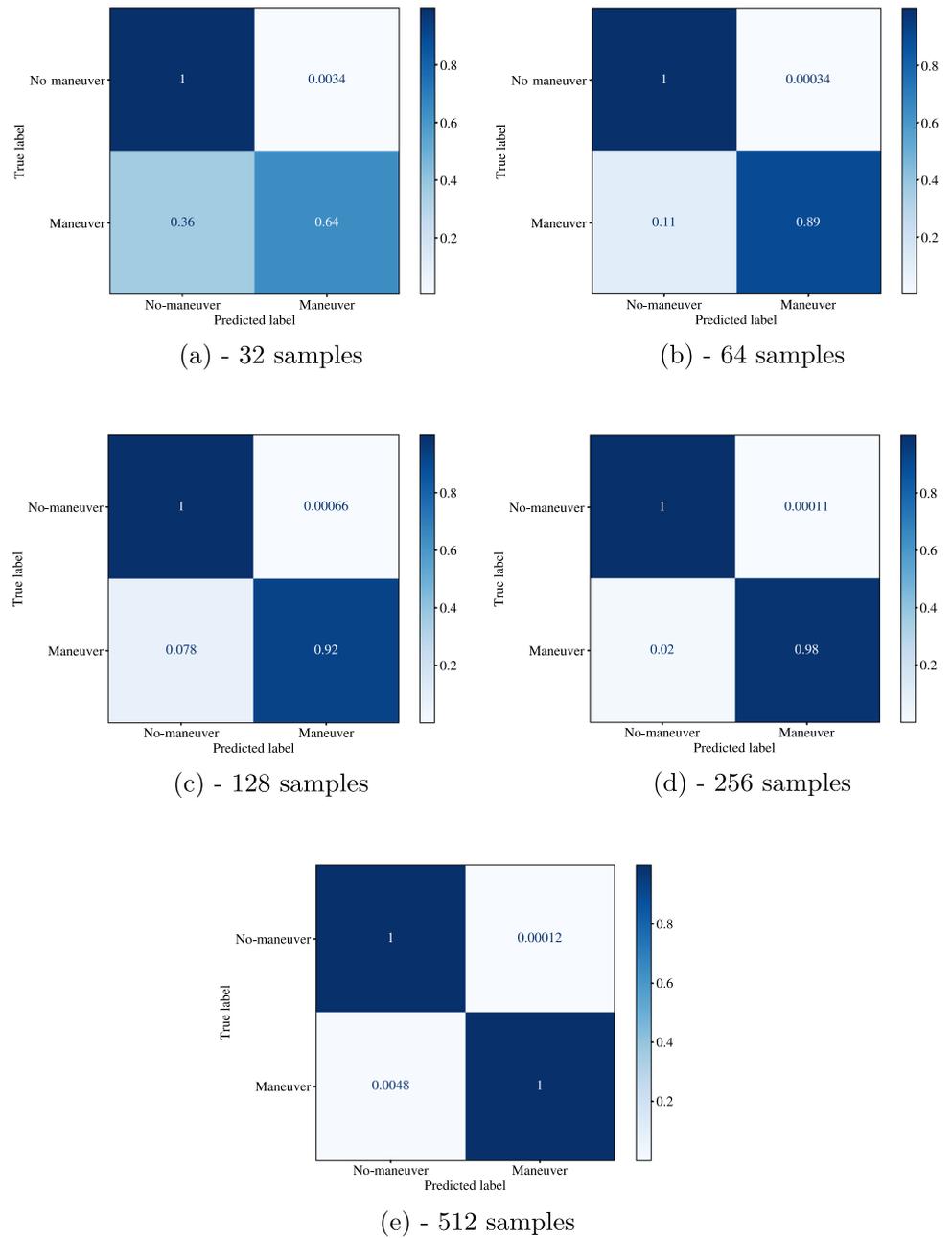


Fig. 11 A histogram showing the comparison in model performance on Sentinel-3A, Sentinel-3B, Jason-3 and Haiyang-2A data, showing weighted accuracy, precision and recall of the 512-sample online maneuver detection model

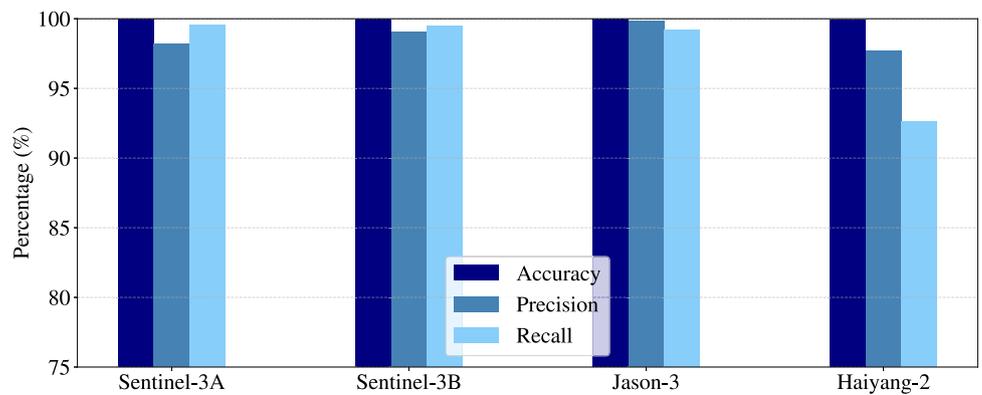


Table 3 Numerical values for each weighted metric used to assess model performance on the selected space objects, using the 512-sample online maneuver detection model

	A_w	P_w	R_w
Sentinel-3A	99.99%	98.21%	99.52%
Sentinel-3B	99.98%	99.05%	99.51%
Jason-3	99.99%	99.80%	99.22%
Haiyang-2A	99.93%	97.66%	92.60%

power, due to the involved loss optimization process generally running over large datasets. As for the inference phase, it usually exploits trained models, with weights that are simply loaded beforehand, resulting in swift prediction capabilities. This can be confirmed by analyzing the processing times associated with each of the models described in this work. The runs are carried out on a Windows machine featuring an Intel Core i7-10700, 16 GB RAM and no dedicated GPU.

The training times related to Net_{off} and Net_{on} are, respectively:

- 61 s/epoch and 62 s/epoch for an input size of 32 samples, with $9.596e-1$ ms and $5.982e-1$ ms as corresponding inference times;
- 106 s/epoch and 106 s/epoch for an input size of 64 samples, with 1.104 ms and 1.076e ms as corresponding inference times;
- 191 s/epoch and 194 s/epoch for an input size of 128 samples, with 1.346 ms and 1.346 ms as corresponding inference times;
- 381 s/epoch and 378 s/epoch for an input size of 256 samples, with 1.863 ms and 1.857 ms as corresponding inference times;
- 753 s/epoch and 756 s/epoch for an input size of 512 samples, with 2.980 ms and 2.861 ms as corresponding inference times.

These results show how time performance strongly depends on the number of weights operations involved in both training (ranging from around 20 min to several hours) and inference, implying that larger input sizes (both in terms of time steps and features involved) mean higher processing times.

5 Conclusions

This work shows how sequential models represent a suitable method to exploit a target pattern of life, in terms of orbital and maneuver history, in order to detect new maneuvers that comply with the satellite’s operational routine. Through the specific labeling logic employed to pre-process raw data, the probability of a maneuver happening in the immediate future can be inferred by spilling the score coming out of the last layer of the network, as the output of the mentioned sigmoid function. This kind of information could assist model-based methods, leveraging the strong assumption of optimal policy, in understanding how far the optimal solution is from the usual behavior of the target. This same value, processed via a thresholding process, is used to categorize orbital data as well, as extensively shown in this paper. There are still limitations to the pipeline reported in this paper. The main one lies in the scarce amount of publicly available labeled data. They are currently limited to LEO targets with very similar orbits, restraining the testing of the networks’ generalization capabilities to a single orbital regime. This is the reason why the offline version of the maneuver detection model (Net_{off}) was conceived: assisting the maneuver labeling process, in terms of involved time windows, in a way to extend already existing datasets. An additional aspect to take into account is the fact that the approach relies on an RSO-specific architecture, requiring an offline training session for each target of interest. This is proved not to be necessary for multiple LEO objects in [15], but it is still considered to fit as baseline solution when no training time constraints have to be complied with.

Ongoing developments are focusing on increasing the level of data pre-processing to improve models' generalization capabilities, currently limited to the object or set of objects involved in the training: even in case multiple similar satellites are used to build the training set, the detection performance drops in case orbital data from a new target are used. Moreover, the models presented are just tested against maneuvers that affect a or E enough to stand out from TLE-derived trends, such as the case of orbit-raising maneuvers performed in LEO.

Different strategies can be adopted to address these shortcomings. A possible one that is investigated consists of an autoencoder, trained as the backbone of the NN architecture shown in the current method, in order to identify a latent space tailored to this specific classification task. An alternative could be instead acting on the training process, using transfer learning to retrieve the subset of model weights tuned on the general features of an orbital data sequence and then specialize the remaining ones to focus on a target of interest.

Acknowledgements The authors would like to thank Dr. Alessandra Di Cecco and Marco Castronuovo from the Italian Space Agency (ASI) for the simulating discussions and precious exchange of opinions that enriched this research work.

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. This publication was the result of the research activity conducted as part of the grant agreement n.2023-37-HH.0 for the project "Attività tecnico-scientifiche di supporto a C-SSA/ISOC e simulazione di architetture di sensori per SST," established between the Italian Space Agency (ASI) and Politecnico di Milano (PoliMi).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Data availability The raw data leading to the results obtained in the present work are openly available online, and cited in the paper: the Space-track website was used to obtain Sentinel-3A, Sentinel-3B, Haiyang-2A, and Jason-3 orbital history while their past control history was retrieved from the ILRS website. As for the processed dataset, it is not public but available upon reasonable request from the corresponding author.

References

1. Office ESD (2022) Esa's annual space environment report. Technical report, European Space Agency (April 2022)
2. Montaruli MF, Purpura G, Cipollone R, Vittori AD, Facchini L, Di Lizia P, Massari M, Peroni M, Panico A, Cecchini A et al (2024) An orbit determination software suite for space surveillance and tracking applications. *CEAS Sp J* 16:1–15
3. Holzinger MJ, Scheeres DJ, Alfriend KT (2012) Object correlation, maneuver detection, and characterization using control distance metrics. *J Guid Control Dyn* 35(4):1312–1325. <https://doi.org/10.2514/1.53245>
4. Cipollone R, Di Lizia P, et al (2023) A back-propagated effort metric for maneuvering space objects correlation. In: 2023 AAS/AIAA astrodynamics specialist conference, pp. 1–17
5. Pastor A, Escribano G, Escobar, D (2020) Satellite maneuver detection with optical survey observations. *Advanced Maui optical and space surveillance technologies conference*
6. Patera RP (2008) Space event detection method. *J Spacecr Rocket* 45(3):554–559. <https://doi.org/10.2514/1.30348>
7. Kelecy T, Hall D, Hamada K, Stocker MD (2007) Satellite maneuver detection using two-line element (TLE) data. In: (2007) advanced maui optical and space surveillance technologies conference (AMOS) proceedings, Maui. Hawai'i, Maui Economic Development Board
8. Lemmens S, Krag H (2014) Two-line-elements-based maneuver detection methods for satellites in low earth orbit. *J Guid Control Dyn* 37(3):860–868. <https://doi.org/10.2514/1.61300>
9. Bai X, Liao C, Pan X, Xu M (2019) Mining two-line element data to detect orbital maneuver for satellite. *IEEE Access* 7:129537–129550. <https://doi.org/10.1109/ACCESS.2019.2940248>
10. Mital R, Cates K, Coughlin J, Ganji G (2019) A machine learning approach to modeling satellite behavior. In: 2019 IEEE international conference on space mission challenges for information technology (SMC-IT), pp. 62–69. IEEE, Pasadena, CA, USA. <https://doi.org/10.1109/SMC-IT.2019.00013>. <https://ieeexplore.ieee.org/document/8863858> Accessed 2022-12-29
11. Shabarekh C, Kent-Bryant J (2016) A novel method for satellite maneuver prediction. In: advanced Maui optical and space surveillance technologies conference (AMOS) proceedings, Maui. Hawai'i, Maui Economic Development Board

12. DiBona P, Foster J, Falcone A, Czajkowski M (2019) Machine learning for RSO maneuver classification and orbital pattern prediction. In: 2019 advanced Maui optical and space surveillance technologies conference (AMOS) proceedings, Maui. Hawai'i, Maui Economic Development Board
13. Cipollone R, Setya Ardi N, Di Lizia P (2022) A supervised learning-based approach to maneuver detection through TLE data mining. In: international conference on applied intelligence and informatics, pp. 419–434. Springer
14. Roberts T (2021) Geosynchronous satellite maneuver classification and orbital pattern anomaly detection via supervised machine learning. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics
15. Cipollone R, Leonzio I, Calabrò G, Lizia PD (2023) An LSTM-based maneuver detection algorithm from satellites pattern of life. In: 2023 IEEE 10th international workshop on metrology for aerospace (Metroaerospace), pp. 78–83. <https://doi.org/10.1109/MetroAeroSpace57412.2023.10189993>
16. Vallado DA, Bastida B, Flohrer T (2013) Improved SSA through orbit determination of two-line element sets. In: 6th European conference on space debris proceedings, Darmstadt, Germany. European Space Agency. <https://doi.org/10.13140/2.1.4644.2241>
17. Hoots FR, Schumacher PW Jr, Glover RA (2004) History of analytical orbit modeling in the us space surveillance system. *J Guid Control Dyn* 27(2):174–185
18. Vallado D, Crawford P, Hujsak R, Kelso T (2006) Revisiting spacetrack report# 3. In: AIAA/AAS astrodynamics specialist conference and exhibit, p. 6753
19. Berry D, Finkleman D (2010) The CCSDS orbit data messages blue book version 2: Status, applications, issues. In: SpaceOps 2010 conference delivering on the dream hosted by NASA Marshall Space Flight Center and Organized by AIAA, p. 2282
20. Roberts T, Linares R (2022) A survey of longitudinal-shift maneuvers performed by geosynchronous satellites from 2010 to (2021) In: 73rd international astronomical congress (IAC 2022) proceedings, Paris. France, International Astronomical Federation
21. Roberts TG, Linares R (2021) Geosynchronous satellite maneuver classification via supervised machine learning. In: 2021 advanced Maui optical and space surveillance technologies conference (AMOS) proceedings, Maui. Hawai'i, Maui Economic Development Board
22. Calvi J, Panico A, Cipollone R, Vittori AD, Lizia PD (2021) Machine learning techniques for detection and tracking of space objects in optical telescope images. In: aerospace Europe conference 2021 (AEC-21) Proceedings, pp. 1–17
23. Vittori AD, Cipollone R, Lizia PD, Massari M (2022) Real-time space object tracklet extraction from telescope survey images with machine learning. *Astrodynamics* 6(2):205–218. <https://doi.org/10.1007/s42064-022-0134-4>
24. Lindemann B, Maschler B, Sahlab N, Weyrich M (2021) A survey on anomaly detection for technical systems using LSTM networks. *Comput Ind* 131:103498
25. Pihlajasalo J, Leppakoski H, Ali-Loytty S, Piche R (2018) Improvement of GPS and BeiDou extended orbit predictions with CNNs. In: 2018 European navigation conference (ENC), pp. 54–59. IEEE, Gothenburg, Sweden. <https://doi.org/10.1109/EURONAV.2018.8433244>. <https://ieeexplore.ieee.org/document/8433244> Accessed 2023-10-15
26. Rumelhart DE, McClelland JL (1987) Learning internal representations by error propagation. In: parallel distributed processing: explorations in the microstructure of cognition: foundations, The MIT Press, pp. 318–362
27. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
28. Baytas IM, Xiao C, Zhang X, Wang F, Jain AK, Zhou J (2017) Patient subtyping via time-aware LSTM networks. In: proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 65–74. ACM. <https://doi.org/10.1145/3097983.3097997>. Accessed 2023-05-16
29. Fritsch FN, Butland J (1984) A method for constructing local monotone piecewise cubic interpolants. *SIAM J Sci Stat Comput* 5(2):300–304. <https://doi.org/10.1137/0905021>
30. Savitzky A, Golay MJE (1964) Smoothing and differentiation of data by simplified least squares procedures. *Anal Chem* 36(8):1627–1639. <https://doi.org/10.1021/ac60214a047>
31. King G, Zeng L (2001) Logistic regression in rare events data. *Polit Anal* 9(2):137–163. <https://doi.org/10.1093/oxfordjournals.pan.a004868>
32. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. *IEEE Trans Signal Process* 45(11):2673–2681. <https://doi.org/10.1109/78.650093>
33. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw* 18(5):602–610. <https://doi.org/10.1016/j.neunet.2005.06.042>
34. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. *Advances in neural information processing systems* 27
35. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Bengio Y, LeCun Y. (eds.) 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, conference track proceedings (2015). <http://arxiv.org/abs/1412.6980>
36. Fernández J, Fernández C, Féménias P, Peter H (2016) The copernicus sentinel-3 mission. In: ILRS workshop, pp. 1–4

37. Zhang S, Andersen OB, Kong X, Li H (2020) Inversion and validation of improved marine gravity field recovery in south china sea by incorporating HY-2A altimeter waveform data. *Remote Sens* 12(5):802. <https://doi.org/10.3390/rs12152470>
38. Vaze P, Neeck S, Bannoura W, Green J, Wade A, Mignogno M, Zaouche G, Couderc V, Thouvenot E, Parisot F (2010) The jason-3 mission: completing the transition of ocean altimetry from research to operations. *Sensors, systems, and next-generation Satellites XIV* 7826:264–268. SPIE. <https://doi.org/10.1117/12.868543>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Riccardo Cipollone¹  · Carolin Frueh² · Pierluigi Di Lizia¹

✉ Riccardo Cipollone
riccardo.cipollone@polimi.it

Carolin Frueh
cfrueh@purdue.edu

Pierluigi Di Lizia
pierluigi.dilizia@polimi.it

¹ Department of Aerospace Science and Technology, Politecnico Di Milano, Via Giuseppe La Masa 32, 20156 Milan, Lombardy, Italy

² School of Aeronautics and Astronautics, Purdue University, 701 W Stadium Ave, West Lafayette, IN 47907, USA