Space Debris Identification and Characterization via Deep Meta-Learning

Roberto Furfaro(1), Richard Linares(2), and Vishnu Reddy(3)

(1) Department of Systems and Industrial Engineering, Department of Aerospace and Mechanical Engineering, University of Arizona, Tucson, Arizona 85721, USA, robertof@email.arizona.edu
(2) Department of Department of Aeronautics and Astronautics, Massachusetts Institute of Technology Cambridge, MA 02139, USA, linaresr@mit.edu
(3) Lunar and Planetary Laboratory, University of Arizona, Tucson, Arizona 85721, USA, reddy@lpl.arizona.edu

ABSTRACT

We describe a new class of deep learning algorithms that can discriminate debris from non-debris objects using light curve real and simulated data. Recently our team demonstrated that deep neural networks (e.g. Convolutional Neural Networks or CNN), trained on real and simulated light curves can be effectively employed in discriminating between active satellites, debris and rocket bodies. However, such algorithms are computationally expensive to train and require a large number of available labeled data. The latter are generally time consuming to obtain. Recently, a new paradigm within the machine learning community emerged, where deep networks are designed with procedures that can teach the system to "learn-to- learn." Named Meta-Learning, it relies on the assumption that a deep learning system can mimic the ability of humans to efficiently learn to recognize objects from a few examples. Indeed, meta-learning is implemented by defining a learning procedure that can be carried along a distribution of the multiple tasks, each with a limited number of examples. Here, we train deep networks with meta-learning (e.g. Model-Agnostic Meta-Learning, MAML) to show that such class of algorithms can effectively solve the debris/non-debris problem by processing light curves. Preliminary results show that meta-learning framework can efficiently and quickly learn to discriminate debris from non- debris object under a variety of observational conditions.

1 INTRODUCTION

Space debris identification and characterization is a problem of paramount importance in Space Situational Awareness (SSA). Currently, the Joint Space Operation Center (JSpOC) Mission System (JSM) is tracking more than 29,000 resident space objects greater than 10 centimeter in size. It is estimated that more than 500,000 objects larger than 1 cm are currently orbiting Earth in the LEO/MEO/GEO regime. As a result, new methods are needed to effectively identify, characterize and concurrently track such large number of objects.

Optical sensors are generally employed to track near-geosynchronous Space Objects (SO) Such sensors provide both astrometric and photometric measurements. Consequently, SO properties (e.g. trajectories, shape and attitude) can be extracted from astrometry and photometric data. More specifically, light curves, defined as the flux of photons across a wavelength reflected by the SO and collected by optical sensors, play a crucial role in determining the SO attitude and state of motion. Importantly, estimation of attitude as well as other SO properties via light curve measurements has been demonstrated [1,2,3,4,5].

Traditional measurement sources for SO tracking such as radar and/or optical measurements, have shown to be sensitive to shape [4, 6], attitude [4], angular velocity [9], and surface parameters [10, 11]. SO properties determination may heavily rely on estimation theory and include the development of multiple model [4, 12], nonlinear state estimation [7,8,9], and full Bayesian inversion [13] approaches for SO characterization. Although grounded in a solid theoretical background and physical first principles, the abovementioned methods tend to be computationally expensive. New techniques are sought that can provide a higher degree of accuracy, computational efficiency and reliability.

Generally, classifying SO, especially trying to resolve the dichotomy debris/not debris, is a challenging task. State-of-the-art methods rely on well established physical models that are embedded in an inversion scheme capable of processing the data and estimate the model parameters. For example, the Multiple Model Adaptive Estimation (MMAE) classification algorithm [3] has been employed to model the SO dynamics, estimate relevant parameters and finally classify SOs. Although such method is one of the most promising available in the literature, the full-scale inversion process requires the estimation of a large number of parameters. As a result, the computational burden is

large and may not be practical for a catalog comprising a large number of objects. Conversely, one may be interested in exploring a data-driven classification approach that employs both simulated and real-data to learn the functional relationship between observed light curves and SO classes.

Recent advancements in machine learning have included deep learning as critical and breakthrough technology. Indeed, deep learning methods [14] have shown ground-breaking results across a large number of domains. Deep networks are neural networks that comprises more than hidden layers of neurons in their architecture. In such multi-layer neuronal arrangement, deep learning approaches are designed to mimic the function of the brain by learning nonlinear hierarchical features from data that build in abstraction [15]. The latter enabled a higher level of accuracy in typical classification tasks. Importantly, Convolutional Neural Networks (CNNs) have been recently explored as a potential class of deep learning approaches for photometric data processing. More specifically, CNNs architectures have been devised and trained to classify SOs based light-curves information collected by electro-optical sensors. CNNs have achieved remarkable performance on image processing tasks. CNNs have achieved remarkable performance on image processing tasks. Examples include 1) object classification [16], 2) scene classification [17], and 3) video classification [18]. Importantly, the key enabling factor for the success of CNN is the development of techniques that can optimize large-scale networks, comprising tens of millions of parameters, as well as massive labeled datasets. Inspired by these results, researchers have finally started exploring the application of such technologies for SSA. Linares and Furfaro [19] first proposed a Deep CNN for light-curve analysis using simulated data, i.e. using a bidirectional reflectance model that simulates the light-curve measured from a ground sensor as function of orbital and rotational state, size and shape. Subsequently, Furfaro et al. [20] investigated the use of CNNs algorithms and related performances in classifying SOs based on both simulated and real-data, and explored the potential application to Space Traffic Management. More recently, Furfaro et. al [21] studied the SO shape recognition problem and investigated the use of deep networks to solve the light-curve inverse problem.

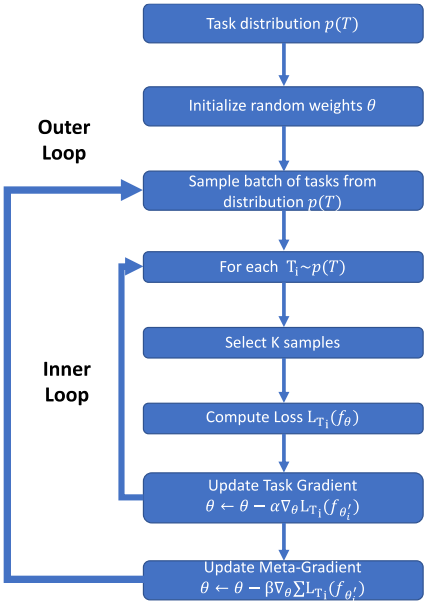
In this paper, in the framework of SO identification and characterization via light curves, we explore the use of a new and exciting AI technology called Deep Meta-Learning (DML) and evaluate its potential SSA applications. One of the major downsides of deep neural networks algorithms is that in order to train deep models, one needs to have available a large training set. Indeed, such models tend to fail abruptly when one has only few data points available. Although a large amount of data can be collected by SSA-based sensors, the application of supervised methods requires extensive labelling of such observations which may come to a major cost. Indeed, labeling of data by a human operator is time consuming and dramatically affected by uncertainties as well as prone to errors (i.e. causing mis-labelling). Simulations may help augment the data set with physics-based data, but nevertheless the approach may suffer from modeling mismatch and requires extensive calibration effort. Meta-learning or "learn-to-learn" [22] may get AI systems closer to how humans learn. Indeed, we tend to generalize our learning to multiple concepts and use it as springboard to learn the next task. Current deep learning algorithms are generally trained to learn a specific task and as consequence they tend to master one task at one time, with the additional requirements of iterating across many epochs in a training database comprising millions of training points. Conversely, DML usually may call for the design of neural network models that can learn to perform a variety of tasks without the need to start the training from scratch. Consequently the model may be trained using a fewer training points. In the context of SO identification and characterization via light curves, the goal is to define a deep meta-learning model and train it on a variety of tasks using only few training points. Whenever applied to a new and related task, the network may quickly learn the new tasks in a few-shot sequence. There have been a lot of recent effort in modeling meta-learning for deep networks mostly based on learning an update function or learning rule. Here we focus on a recently developed meta-learning approach named Model-Agnostic Meta-Learning (MAML,[23]). MAML is general, model-agnostic and it is trained with a gradient descent procedure. When applied to currently developed CNNs architectures for light curves processing, it implements the key idea of training the meta-learner to learn the model initial parameters so that it has maximum performances on a new task, i.e. CNNs processing light curves can be quickly trained to learn a new but related task. A set of selected examples using CNNs architecture are shown to demonstrate the proposed approach.

2 DEEP META-LEARNING FOR SPACE OBJECTS AND DEBRIS CLASSIFICATION

The goal of this section is to describe the fundamentals of meta-learning and the types of algorithms that can be explored for SO characterization and identification using light curves measurements. Here, the goal is to show that deep networks architectures can be employed for SSA-based classification tasks using a relatively low number of data (training) points available for each class. Within this context, one wishes to explore the possibility of acquiring knowledge from a few examples in an efficient and accurate fashion. However, the learning process is spread across a set of tasks that are both different and related. The overall process can be implemented through meta-learning [22]

in which the learning problem is framed at two levels, i.e. 1) quick acquisition of knowledge within each of the separate task presented to the network; and 2) extraction of information across all the tasks. Clearly, the meta-learner (i.e. the second level) guides the overall process. The problem formulation for meta-learning is somewhat different than conventional machine learning. In a typical ML setting, one has available a dataset D which is usually split as $D \equiv D_{train} \cup D_{test}$ where D_{train} and D_{test} are the training and testing datasets employed to optimized the network parameters, respectively. Conversely, in a typical meta-learning setting, one deals with meta-sets $\mathcal{D}$ which contains all datasets D_i associated to the i^{th} tasks. Typically, for SO classification problem, we consider the k-shot, N-class classification task. In this framework, for each dataset D_i, the training set $D_{i_{train}}$ consist of k labelled examples for each of the N classes, comprising overall kN examples. Conversely, the $D_{i_{test}}$ comprises a few examples, generally employed for evaluation.

Within the meta-learning framework, a variety of techniques have been proposed to effectively solve the k-shot, N-class classification problem via meta-learning. One of the most notable approach is presented in [24] where a Short-Long Term Memory (LSTM) recurrent architecture is employed as a meta-learner to optimize a neural network classifier. The meta-learner effectively captures the short-term knowledge within each individual task and a long-term knowledge distributed across the tasks. The approach has been successfully adapted to the guidance landing problem [25]. However, for SSA application, we will focus on the gradient-based MAML [23] which is described in the next section.



Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

Fig. 1. Meta-Agnostic Meta-Learning (MAML, adapted from [23])

2.1 Model-Agnostic Meta-Learning (MAML) Algorithm

MAML addresses the problem of learning to learn in a flexible and effective fashion. The basic idea behind MAML is to determine a set of "best", optimized network parameters so that the model can learn quickly on new related tasks, i.e. using less gradient steps and thus resulting in less iterations. The goal is to learn internal features/parameters that are not specialized for a single tasks but that are broadly applicable across all tasks. To encourage a general-purpose representation, MAML takes an explicit approach to the problem by learning a model in such a way that the gradient-based learning rule can make quick and rapid progresses toward new tasks that are drawn out of a distribution of tasks. The objective of this gradient-based procedure is to find the network parameters that are the most sensitive to changes in task, i.e. parameters that produce the greatest improvements in loss and prediction accuracy on any tasks on the distribution.

Within the few-shot, meta-learning problem, whose goal is to train a deep network model to quickly adapt to a new task using only few data points, we consider a deep network model parametrized as $f_\theta(\cdot)$ which maps an observation (light curves) into a class (e.g. rocket body). Additionally, one has a distribution over tasks $p(T)$, i.e. the network can

sample individual tasks and train on k examples taken out of the distribution. Initially, the parameters ϑ are taken our of random distribution (e.g. uniform). Then the training process starts by sampling batches of tasks T_i out of the associated distribution $T_i \sim p(T)$. For each of the task T_i we sample k data points (shots) and start the training process. The model $f_\theta(\cdot)$ is trained by computing the loss $L_{T_i}(f_\theta)$ associated with the i-th task, typically a cross-entropy loss for classification, and we aim at minimizing the loss via gradient descent, as follow:

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$$

Here, for one-step gradient update, θ_i' is the one-step optimal parameter for task T_i, θ is the initial network parameter, α is the internal hyperparameter (learning rate) and $\nabla_\theta L_{T_i}$ is the gradient computed on the task T_i. Before moving to sampling the next batch of tasks, we perform what is properly named meta-update or meta-optimization. The model parameters are trained by optimizing $f_{\theta_i'}$ with respect to θ across all tasks that have been sampled out of the distribution. In this specific case, one define a meta-objective (or meta-loss) as a minimization problem, i.e.:

$$\min_\theta \sum_{T \sim p(T)} L_{T_i}(f_{\theta_i'})$$

The meta-step or meta-update across all tasks is generally performed via stochastic gradient descent:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T \sim p(T)} L_{T_i}(f_{\theta_i'})$$

Here, β is the meta-learning rate (hyperparameter) and $\nabla_\theta \sum_{T \sim p(T)} L_{T_i}(f_{\theta_i'})$ is the gradient for each of the new tasks T_i with respect to the parameter θ_i'. The meta-update equation implies that one updates the model parameters θ by taking the average of the gradients of each new tasks with respect to its optimal parameters.

Figure 1 shown the workflow of the complete MAML algorithm. The flow consists of two loops, i.e. inner and outer loop. In the inner loop, the algorithm computes the optimal parameters θ_i' for each of the T_i tasks. Conversely, the outer loop updates the randomly initiated parameters θ by computing the average gradient computed over the optimal parameters θ_i'. Importantly, in a set of new tasks T_i, one must not use the same set of tasks employed to find the optimal parameters θ_i'.

3 SELECTED RESULTS

In this section, we provide selected results associated with training CNNs using 1) simulated data, 2) real data; and 3) preliminary results with meta-learning.

3.1 Simulated and Real Light Curves

For the simulated case, the labeled training data samples are generated using the light curve models [2]. The parameters required to define the light curve model are sampled. We considered four categories, i.e. fragments, rocket bodies, regular polygon prisms and rectangular cuboids. The SO parameter models associated with shape and surface are randomly generated out of a uniform distribution. Importantly, the regular polygon prisms are then further divided into equilateral triangular prisms, square prisms and regular hexagonal prisms. Models are generated by sampling side lengths and heights from a uniform distribution on the interval [0.01, 5] m. For the regular polygon prisms, the number of sides are also selected randomly on the interval [3, 6], with all instances of 5 sides being set to 4 as pentagonal prism models are not included. In addition to the model geometry, the material properties also need be defined. For each model, all facets are assumed to have the following: $R_{spec} = 0.7$, $R_{diff} = 0.3$, $\varepsilon = 0.5$. The Phong parameters

are each taken to be equal to 1000 for all facets of every model. Rocket body models are generated using octant triangulation of a sphere as discussed in [26] which divided the surface of a sphere into *N* facet normal. Then rocket body models are generated by connecting two hemisphere ends of radius *r* with cylinder of height *l*. Finally, the fragment shapes use the cuboid model but with much small aspect ratios than payload shapes. Figure 2A shows the training data generated using the process discussed above where the labels are shown using colored data points.

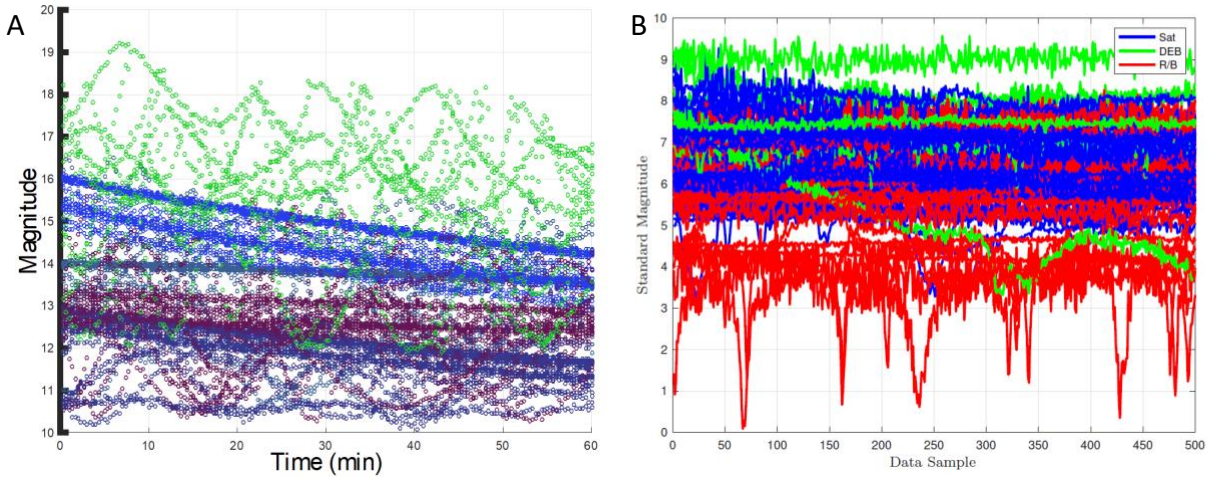


Fig. 2. Examples of A) Simulated data; and B) Light Curve Measurements taken from the Multichannel Monitoring Telescope (MMT). Both data are employed to train CNNs.

For the real light curve SO classification, photometric observations taken from the Multichannel Monitoring Telescope (MMT) are employed as training set. This data source is publicly available through astroguard.ru. For this work, the training data was developed using segments of 500 measurement samples taken from the MMT dataset for objects with TLE information. Their TLE numbers and TLE names label the objects in the MMT dataset, and from the TLE names, class information can be extracted. Three classes are used in this work, and these classes are Debris, Rocket Bodies, and Satellite. Figure 2B shows some representative examples of the MMT data used for training. From this figure, a clear difference can be seen from Debris, Rocket Bodies, and Satellite classes.

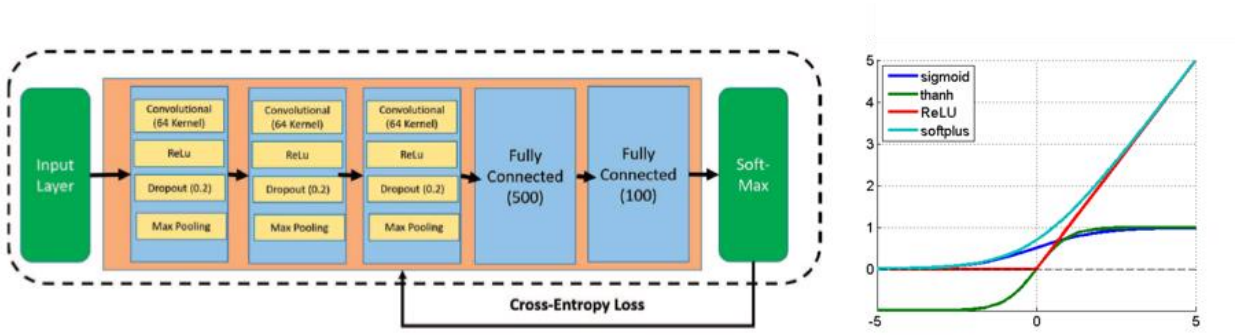


Fig. 3. CNN architecture and possible activation fucntions.

3.2 CNN Architecture, Training and Validation Process

The CNN is designed to learn the functional relationship between simulated and real measurements and SO classes consists of 1-D convolutional layers with rectified linear unit (ReLU) activation, dropout, max-pooling, and two fully connected layer with ReLU activation. The output layer uses softmax function to map to classification states. Each convolutional layer has the following form:

$$\boldsymbol{h}^{cov}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x} * W^{conv} + \boldsymbol{b}^{conv})$$

Where $*$ denotes the convolution operator, W^{conv} denotes the convolution kernel, and $\boldsymbol{f}$ is the activation function for each layer that adds nonlinearity to the feature vector. Here, we use ReLU for convolutional layer activation. The ReLU function is given by $\boldsymbol{f}(\boldsymbol{x}) = \max(\boldsymbol{0}, \boldsymbol{x})$, where it is zero for negative input values and linear for positive input values. The convolution of $\boldsymbol{x}$ defines the output feature map $\boldsymbol{h}^{cov}(\boldsymbol{x})$. The number of output maps is determined by the number of convolution filters for each convolutional layer. Each convolution layer has a collection of kernels of given size that are learned directly from the data. For the light curve problem the convolutions are of one dimensional time-series data. Then a CNN applies a series of these kernels W^{conv}in a layered fashion where each layer has a different size kernel that learns features on a given scale. To simplify the information in the output of each convolutional layer, max-pooling is used. In this work max-pooling with 1 × 4 kernel between each convolution layer is used. The max-pooling operation convolves the 1 × 4 kernel over the output of each convolutional layer returning the max of the outputs over the 1 × 4 region. Finally, at the final layer a nonlinear function is applied to the output using a traditional neural network. This final layer uses a fully connected neural network layer with a softmax function.

The convolutional kernel and fully connected output layer parameters are cased into the vector θ. The cost function used for this work is the cross-entropy loss. This loss function minimizes the cross-entropy loss between training outputs and the CNN output and can be described as follows:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}[\mathbf{y}\ \log\ (\tilde{\boldsymbol{y}}) - (1-\boldsymbol{y})\ \log\ (1-\tilde{\boldsymbol{y}})]$$

Where $\tilde{\boldsymbol{y}}$ are the training examples and $\boldsymbol{y}$ are the outputs from the CNN. Then the CNN classification approach is trained by stochastic gradient descent by minimizing the cross-entropy loss from the outputs compared to the labelled data. The proposed CNN architecture is shown in Figure 3. The CNN comprises a seven (7) layer structure, including input layer, three convolutional layers, two fully connected layers and one softmax layer. The input layers manages a light curve input vector comprising 182 data points. The convolutional layers have 64, 32 and 64 filters, respectively. Both max-pooling (1×2 pooling kernel) and dropout are applied after each convolutional layer.Dropout regularization rates are set to be 0.2 for the convolutional layers and 0.5 for fully connected layers. Training occurred for 2000 epochs using stochastic gradient descent and mini-batch of 128 samples.

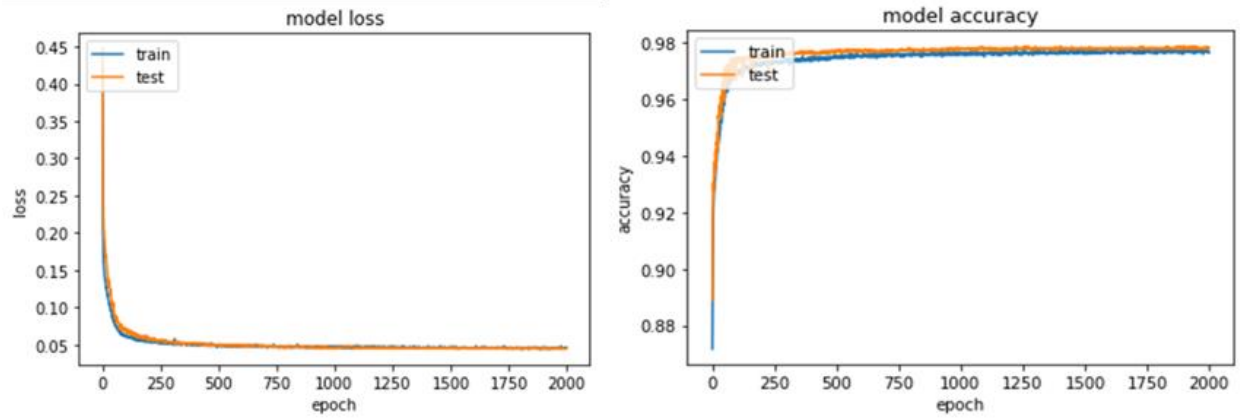


Fig. 4. Loss function and Accuracy during the training of the CNNs using simulated light curves

3.3 CNN on Simulated Light Curves

The CNN is trained over 10,000 randomly generated scenarios comprising nine (9) possible classes of SO. During the training, the CNN is validated against 5000 data scenarios not used in the training set. For all training scenarios, an SO is in near geosynchronous orbit with orbital elements given by a = 42, 364.17 km, e = 2.429 × 10−4, i = 30 deg, ω = Ω = 0.0 deg and M_0 = 91.065 deg. Brightness magnitude are simulated using a ground station located at 20.71◦ North, 156.26◦ West longitude and 3,058.6 m altitude. Measurements constructed using instantaneous geometry are

corrupted by zero-mean Gaussian white noise with standard deviations of 0.1 for the brightness magnitude. Observations are available every 5 seconds for one hour (Figure 2A). All training samples are generated using the same orbit during the sample simulation time interval. Each sample has different shape model, rotational initial condition, and control profile. The Keras (Python) library wit Tensorflow as backend are employed to design and train the network. The 1D-CNN has been trained using a single GPU with 1500 processors. Figure 4 shows the results of the training process. The CNN has been trained on 8000 samples comprising the training set and tested on 2000 samples during the training process. Figure 4A shows the behavior of the cross-entropy loss as function of the epoch. Figure 4B shows the model accuracy as function of the epoch. Both test and training sets have a similar accuracy result. We report that the CNN exhibits an accuracy of 97.83% on the test set. Training time is reported to be 2000*sec*.

3.4 CNN on Real Light Curves

The same 1D-CNN architecture described above (see Figure 3) has been employed to train the deep network on the real light curve observars. In this case, the training set comprised 118,400 samples and is subdivided in three possible classes,i.e. rocket bodies, debris, other. In the class other, it is included anything that is not either a rocket body or a debris. A set of 29,993 light curves samples is employed as test set during the training. The input light curve vector comprises 500 points. The number of epochs is set to be 2000. Figure 5 shows the results of the training process. Figure 5B shows the behavior of the cross-entropy loss as function of the epoch. Figure 5A shows the model accuracy as function of the epoch. The real set of data is more comprehensive as it accounts for classes of objects with different observing conditions and different orbits (i.e. GEO, LEO and MEO). Thus, it is expected that the separation boundaries between classes is highly non-linear. Here, we show that accuracy over the training set and test set is markedly different. At the end of the training, accuracy on sequence of minibatches is as large as 81%. Conversely, final accuracy on the test set is 77%. Training time is about 8*hrs* and single GPU with 1500 processors.

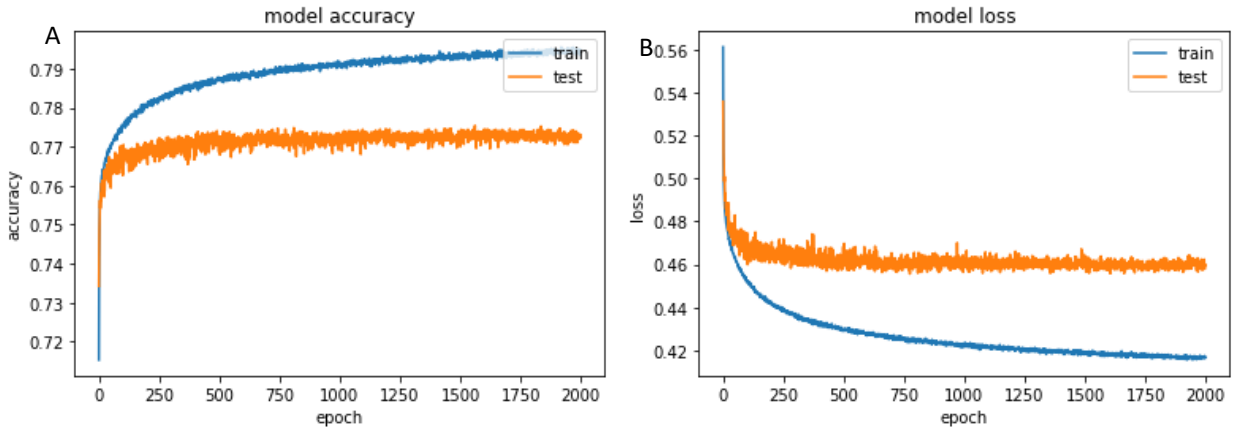


Fig. 5. Loss function and Accuracy during the training of the CNNs using MMT real light curves.

3.5 Preliminary Results with MAML-based Algorithm

A preliminary implementation of the MAML algorithm for SO classification has been executed and initial results reported. In this case, we considered a dense deep network capable of processing the real light curve data out output three possible classes (i.e. debris, rocket bodies and satellites). The selected architecture is 5-layer deep network 500-2000-1000-500-3 with an input vector as an element of $\mathbb{R}^{500}$ and an output corresponding to the three abovementioned classes. We considered a meta-training set $\mathcal{D}$ comprising 10,000 samples of real light curves (Figure 2B). We considered the k-shot, N-class problem with $k = 10$ and $N = 3$. In this case, each task T_i is sampled out of a distribution $T_i \sim p(T)$ where the task training D_{train_i} is randomly selected out the meta training set $\mathcal{D}$ and contains 10 light curves measurements. Conversely, the task test set D_{test_i} comprises 5 randomly selected light curves sampled out of D. Training occurs for one epoch with 1000 gradients meta-updates. After the training is complete, the trained network is further trained on a new similar task (classification of space objects with $N = 3$) using two data sets comprising $k_1 = 10$ and $k_2 = 20$ shots, respectively. Testing occurs of a test set comprising $k_{test} = 10$ individual light curves. Results are reported in Table 1. It is noted an improvement of the performance of the deep network with

respect to the CNN trained on the full set of real data (Figure 5). These results are considered to be very preliminary and are the subject of further investigation.

Table 1. MAML algorithm preliminary performances

	10-shot ($k = 10$)	**20-shot ($k = 20$)**
DNN Accuracy	~82%	~85%

4 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced the concept of meta-learning for application to SSA. More specifically, we investigated the problem of training deep networks architectures to discriminate between different SO using light curves measurements and explored the meta-learning approach as a possible training strategy. It is demonstrated that CNNs can be trained both on simulated and real light curves observations to classify space objects and that meta-learning may provide a framework for learning using a very limited amount of data (few-shot learning problem). The presented results are preliminary and still subject to further investigation. Future work will solidify the initial findings as further analysis of the MAML algorithm for light-curve classification is currently undergoing at University of Arizona.

5 REFERENCES

1. Linares, R., Jah, M.K., Crassidis, J.L., Leve, F.A. and Kelecy, T., 2014. Astrometric and photometric data fusion for inactive space object mass and area estimation. *Acta Astronautica*, *99*, pp.1-15.
2. Linares, R., Jah, M.K., and Crassidis, J. L. , "Inactive Space Object Shape Estimation via Astrometric And Photometric Data Fusion," *AAS/AIAA Space Flight Mechanics Meeting*, No. AAS Paper 2012-117, Charleston, SC, Jan.-Feb 2012.
3. Linares, R., Jah, M. K., and Crassidis, J. L., "Space Object Area-To-Mass Ratio Estimation Using Multiple Model Approaches," *Advances in the Astronautical Sciences*, Vol. 144, 2012, pp. 55–72.
4. Linares,R., Jah, M. K., Crassidis, J. L., and Nebelecky, C. K., "Space Object Shape Characterization and Tracking Using Light Curve and Angles Data," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, 2013, pp. 13–25.
5. Hinks, J. C., Linares, R., and Crassidis, J. L., "Attitude Observability from Light Curve Measurements," *AIAA Guidance, Navigation, and Control (GNC) Conference*, No. 10.2514/6.2013-5005, AIAA, Boston, MA, August 2013.
6. Hall, D., Calef, B., Knox, K., Bolden, M., and Kervin, P., "Separating Attitude and Shape Effects for Non-resolved Objects," *The 2007 AMOS Technical Conference Proceedings*, 2007, pp. 464–475.
7. Jah, M. and Madler, R., "Satellite Characterization: Angles and Light Curve Data Fusion for Spacecraft State and Parameter Estimation," *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, Vol. 49, Wailea, Maui, HI, Sept. 2007, Paper E49.
8. Holzinger, M. J., Alfriend, K. T., Wetterer, C. J., Luu, K. K., Sabol, C., and Hamada, K., "Photometric attitude estimation for agile space objects with shape uncertainty," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 921–932.
9. Linares, R., Shoemaker, M., Walker, A., Mehta, P. M., Palmer, D. M., Thompson, D. C., Koller, J., and Crassidis, J. L., "Photometric Data from Non-Resolved Objects for Space Object Characterization and Improved Atmospheric Modeling," *Advanced Maui Optical and Space Surveillance Technologies Conference*, Vol. 1, 2013, p. 32.
10. Linares, R., Jah, M. K., Crassidis, J. L., Leve, F. A., and Kelecy, T., "Astrometric and photometric data fusion for inactive space object feature estimation," *Proceedings of 62nd International Astronautical Congress, International Astronautical Federation*, Vol. 3, 2011, pp. 2289–2305.

11. Gaylor, D. and Anderson, J., "Use of Hierarchical Mixtures of Experts to Detect Resident Space Object Attitude," *Advanced Maui Optical and Space Surveillance Technologies Conference*, Vol. 1, 2014, p. 70.
12. Wetterer, C. J., Linares, R., Crassidis, J. L., Kelecy, T. M., Ziebart, M. K., Jah, M. K., and Cefola, P. J., "Refining space object radiation pressure modeling with bidirectional reflectance distribution functions," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, 2013, pp. 185–196.
13. Linares, R. and Crassidis, J. L., "Resident Space Object Shape Inversion via Adaptive Hamiltonian Markov Chain Monte Carlo," *AAS/AIAA Space Flight Mechanics Meeting*, No. AAS Paper 2016-514, Napa, CA, Feb 2016.
14. LeCun, Y., Bengio, Y., and Hinton,G.,"Deep learning,"*Nature*,Vol.521,No.7553,2015,pp.436–444.
15. Lee, H., Pham, P., Largman, Y., and Ng, A. Y., "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in neural information processing systems*, 2009, pp. 1096–1104.
16. Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012, pp. 1097–1105.
17. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A., "Learning deep features for scene recognition using places database," *Advances in neural information processing systems*, 2014, pp. 487–495.
18. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L., "Large-scale video classification with convolutional neural networks," *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.
19. Linares, R. and Furfaro, R., 2016, July. Space object classification using deep convolutional neural networks. In *2016 19th International Conference on Information Fusion (FUSION)* (pp. 1140-1146). IEEE.
20. Furfaro, R., Linares, R., & Reddy, V. (2018, September). Space Objects Classification via Light-Curve Measurements: Deep Convolutional Neural Networks and Model-based Transfer Learning. In *AMOS Technologies Conference, Maui Economic Development Board, Kihei, Maui, HI*.
21. Furfaro, R., Linares, R., & Reddy, V. (2019, September). Shape Identification of Space Objects via Light Curve Inversion using Deep Learning Models. In *AMOS Technologies Conference, Maui Economic Development Board, Kihei, Maui, HI*.
22. Thrun, S. and Pratt, L. eds., 2012. *Learning to learn*. Springer Science & Business Media.
23. Finn, C., Abbeel, P. and Levine, S., 2017, August. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 1126-1135). JMLR. org.
24. Ravi, S., & Larochelle, H. (2016). Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.
25. Gaudet, B., Linares, R. and Furfaro, R., 2019. Adaptive Guidance and Integrated Navigation with Reinforcement Meta-Learning. *arXiv preprint arXiv:1904.09865*.
26. Kaasalainen, M. and Torppa, J., 2001. Optimization methods for asteroid lightcurve inversion: I. shape determination. *Icarus*, *153*(1), pp.24-36.