Feb 26th, 2:00 PM

Space Objects Classification and Characterization via Deep Learning and Light Curves: Applications to Space Traffic Management

Roberto Furfaro
University of Arizona, robertof@email.arizona.edu

Richard Linares
Massachusetts Institute of Technology, linaresr@mit.edu

Vishnu Reddy
University of Arizona, reddy@lpl.arizona.edu

Space Objects Classification and Characterization via Deep Learning and Light Curves: Applications to Space Traffic Management

Roberto Furfaro*
University of Arizona, Tucson, Arizona, USA

Richard Linares†
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Vishnu Reddy‡
University of Arizona, Tucson, Arizona, USA

Recent advancements in deep learning (e.g. Convolutional Neural Networks (CNN), Recurrent Neural networks (RNN)) have demonstrated impressive results in many practical and theoretical fields (e.g. speech recognition, computer vision, robotics). Whereas deep learning methods are becoming ubiquitous, they have been barely explored in SSA applications, in particular with regard to object characterization for Space Traffic Management (STM). In this paper, we report the results obtained in designing and training a set of CNNs and RNNs for Space Object (SO) classification and characterization using light-curve measurements. More specifically, we provide a comparison between deep networks trained on both physically-based models (i.e. reflectance models) and real light-curve data (data-driven approach). Physically-based models allow the generation of synthetic light-curves as function of the SO parameters (e.g. shape, size, material). Such models have the ability to yield a large number of training points which make them suitable to train deep networks. Nevertheless, they suffer from modeling errors. Conversely, real light-curve measurements can be employed to directly train CNNs and/or RNNs, yet with a relatively limited data set. Insight in how to design such networks and expected performances will be discussed to highlight the power of the proposed methodology. Additionally, a cluster analysis conducted via data dimensionality reductions coupled with unsupervised feature extractions for SO characterization is reported.

I. Introduction

Space Traffic Management (STM) is becoming an area of utmost interest. The latter is due to a dramatic increment of the resident Space Objects (SO) as well as the continuously increasing amount of international space actors, including state and private companies. To help and support STM, Space Situational Awareness (SSA) has been recently become an important research topic especially when dealing wiht the problem of SO identification, characterization and behavioral understanding. Enabled by the most recent advancement in sensor technology, researches and operational engineers rely on a large amount of tracking data that can be processed to identify, characterize and understand intention of space objects. The SO catalog maintained by JSpoC currently includes upward of 29,000 SO, with 1,100 of such objects being actively controlled and operated. Researchers working on SSA are interested in providing a detailed understanding of the SO population behavior which must go beyond the currently SO catalog comprising simplified SO characteristics such as solar radiation pressure and drag coefficients. To provide a more realistic and reliable understanding of the SO dynamics, future catalogs that support STM must include detailed SO characteristics (e.g. shape and state of motion). The latter can be employed in dynamical propagation models to accurately predict SO trajectory and behavior.

Optical sensors are generally employed to track near-geosynchronous SO. Such sensors provide both astrometric and photometric measurements. Consequently, SO properties can be estrated from astrometry pipelines (e.g. trajectories)

*Professor, Department of Systems and Industrial Engineering, Department of Aerospace and Mechanical Engineering, Tucson, Arizona, 85721

†Charles Stark Draper Assistant Professor, Department of Aeronautics and Astronautics, Massachussets Institute of Technology, Cambridge, MA, 02139

‡Associate Professor, Lunar and Planetary Laboratory, Tucson, Arizona, 85721

and photometric data (e.g. shape and state of motion). More specifically, light curves, i.e. flux of photons across a wavelength reflected by the SO and collected by optical sensors, play an important role in determining the SO attitude and state of motion. Indeed, attitude estimation and extraction of other characteristic using light curve data has been demonstrated in Ref. 1–5.

Traditional measurement sources for SO tracking (e.g. radar and/or optical measurements)have been shown to be sensitive to shape [4, 6], attitude [4], angular velocity [9], and surface parameters [10, 11]. A literature review shows that recent advancement have been made to estimate SO properties. Such techniques heavily rely on estimation theory and include the development of multiple model [4, 12], nonlinear state estimation [7–9], and full Bayesian inversion [13] approaches for SO characterization. Although grounded in a solid theoretical background, the above mentioned 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 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, Reference 4 used a Multiple Model Adaptive Estimation classification approach to model the dynamics and physics, estimate relevant parameters and finally classify SOs. Although such method is one of the most promising available in the literature, the inversion process require 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. Here, we are 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 class.

Recent advancements in machine learning have included deep learning as critical 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. In this paper, we explore deep learning methods to classify SOs trained on their simulated and real data. More specifically, we investigate Convolutional Neural Networks (with max-pooling and dropout) [15] for supervised classification of SO observational data. Here, we demonstrate the design CNNs architectures trained both on simulated and real data, and demonstrate the effectiveness of the proposed methodology in classifying SOs. 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, this paper studies the classification of observational data generated by both simulated and real dynamical systems. The dynamical system investigated here is rotational dynamics of SOs. The physical attributes of the SOs, such as shape and mass distribution, are also included in the classification process. The challenging aspect of the application of CNNs to physical dynamical systems is the generation of labeled training data. In a dual-fold fashion, we first use physical models to simulate observations by sampling randomly from a distribution of physical attributes and dynamic states. Light curve measurements are used as inputs, and classes of the SOs are used as outputs for training the CNN approach. The 1D-CNN then learns convolutional kernels that look for characteristic features in the light curve data. As opposed to manually specified features, the features are adaptively learned given the training data.

II. Deep Learning Methods: Convolutional Neural Networks

Over the past few years, there has been an explosion of machine learning algorithms. Such algorithms can learn from data to accomplish specific tasks (e.g. image recognition, object identification, natural language process, etc.). Among the various available techniques, deep learning, comprising methods and techniques to design and train multi-layer neural networks, has been playing a dominant role. In contrast to shallow networks, deep networks refers to a class of neural networks comprising a number of hidden layers greater than one. Among all possible systems, one of the most powerful deep architectures is called Convolutional Neural Networks (CNN). CNNs, which are nominally applied to input images, have been used to classify an image or determining the particular content of an image by transforming the original image, through a set of layers with specialized architecture, to a class scores. Indeed, the architecture of a CNN is designed to take advantage of the 2D or 3D $[width, height, depth]$ structure of an input image which is processed as pixels values. The basic CNN structure uses many types of layers which are 1) Convolutional Layer, 2) Pooling Layer, 3) Fully Connected Layer and 4) Output Layer. The core layer, i.e. the Convolutional layer, extract features from the

input volume by applying filters on the image. It is the most demanding layer in terms of computations of a CNN and the layer′s parameters consist of a set of learnable filters. Each filter is basically a matrix spatially smaller than the image which is scanned along width and height (2D case). Importantly, filters (or kernels) are the weights of this layer. In fact, as the filter is sliding on the input image, it multiplies its values with the original pixel values of the image and these multiplications are all summed up giving only one number as output. Repeating this procedure for all the regions on which filter is applied, the input volume is reduced and transformed and then passed to the *Max-pooling layers*. Matematically, the convolutional layer can be described as follows:

$$(X * Y)(i,j) = \sum_{n=0}^{N} \sum_{m=0}^{M} X_{m,n} * W_{i-m,j-n} \tag{1}$$

Here, W is the convolution kernel corresponding to the randomly initialized weights, and X is the image with indices (m,n). CNN typically employs the nonlinear activation function called $ReLU$ function (i.e., Rectified Linear Unit) described as $f(x) = max(0, x)$. Spatial pooling layers group local features from spatially adjacent pixels to improve robustness. A set of convolutional layers are generally stacked below a fully connected layer that feeds a soft-max layer ([21]), which outputs the probability of the image belonging to one of the classes. CNN are easier to train due to inherent parameter sharing in the convolutional layer. For a classification task, the cross-entropy function [21] is generally employed as cost to minimize.CNNs are trained in batch mode via Stochastic Gradient Descent (SDG) with variable size of mini-batches. The dropout technique improves generalization and avoids overfitting.

III. Training Set Generation

A. Simulating Labeled Training Data

For the simulated case, the labeled training data samples are generated using the light curve models ([2]). The parameters required to define the AS 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. The regular polygon prisms are prisms whose ends (i.e. top and bottom) are regular shapes. The shape of a regular polygon prism is defined by the number of sides n, side length s and height h.

$$h_{\text{regular}} = (h_{\min} + 0.01) + (h_{\max} - h_{\min} - 0.01)\,\mathcal{U}[0,1] \tag{2a}$$

$$s_{\text{regular}} = (s_{\min} + 0.01) + (s_{\max} - s_{\min} - 0.01)\,\mathcal{U}[0,1] \tag{2b}$$

Assuming constant density throughout the shape model, the moment of inertia matrices for each of the regular polygon models are given by

$$J_{\text{triangle}} = m_{\text{SO}} \begin{bmatrix} \frac{s^2}{24} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{s^2}{24} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{s^2}{12} \end{bmatrix} \tag{3a}$$

$$J_{\text{square}} = m_{\text{SO}} \begin{bmatrix} \frac{s^2}{12} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{s^2}{12} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{s^2}{6} \end{bmatrix} \tag{3b}$$

$$J_{\text{hexagon}} = m_{\text{SO}} \begin{bmatrix} \frac{5s^2}{24} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{5s^2}{24} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{5s^2}{24} \end{bmatrix} \tag{3c}$$

The rectangular cuboids are prisms defined by two side lengths s_1 and s_2 as well as the height h. The moment of inertia matrix for the cuboids are given by

$$J_{\text{cuboid}} = \frac{m_{\text{SO}}}{12} \begin{bmatrix} s_2^2 + h^2 & 0 & 0 \\ 0 & s_1^2 + h^2 & 0 \\ 0 & 0 & s_1^2 + s_2^2 \end{bmatrix} \tag{4}$$

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: $R_{\text{spec}} = 0.7$, $R_{\text{diff}} = 0.3$, $\epsilon = 0.5$. The Phong parameters n_u and n_v are each taken to be equal to 1000 for all facets of every model. The mass of the SO is randomly sampled using the following $m_{\text{SO}} = m_{\min} + (m_{\max} - m_{\min})\mathcal{U}[0, 1]$.

Rocket body model are generated using octant triangulation of a sphere discussed in Ref. 24 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. This model is not exact for all rocket bodies but is close enough to approximate the types of light curves seen for rocket bodies.

$$\begin{aligned} J_{\text{rocket}} = m_{\text{SO}} \Bigg\{ & \frac{V_{\text{cyl}}}{V_{\text{tot}}} \text{diag}\left[\frac{1}{12}(3r^2 + l^2), \frac{1}{12}(3r^2 + l^2), \frac{r^2}{2}\right] \\ & + \frac{V_{\text{top}}}{V_{\text{tot}}} \text{diag}\left[\frac{1}{12}(3r^2 + l^2), \frac{1}{12}(3r^2 + l^2), \frac{r^2}{2}\right] \\ & + \left(\frac{V_{\text{top}}}{V_{\text{tot}}}\left(\frac{l}{2} + \frac{3r}{8}\right) + \frac{V_{\text{cyl}}}{V_{\text{tot}}}\left(\frac{l}{2} - \frac{3r}{8}\right)\right)\left(I_{3\times 3} - \mathbf{e}\mathbf{e}^T\right) \\ & + 2\frac{V_{\text{top}}}{V_{\text{tot}}} r^2 \text{diag}\left[\frac{83}{320}, \frac{83}{320}, \frac{2}{5}\right] \Bigg\} \end{aligned} \tag{5}$$

Where $\mathbf{e} = [0, 0, 1]^T$ and the volume of the top hemisphere is given by $V_{\text{top}} = 2/3\pi r^3$ and it is assumed the bottom volume is $V_{\text{bot}} = V_{\text{top}}$. The volume of the cylinder is given by $V_{\text{cyl}} = \pi r^2 l$ and the total volume is $V_{\text{tot}} = V_{\text{top}} + V_{\text{bot}} + V_{\text{cyl}}$. Finally, the fragment shapes use the cuboid model but with much small aspect ratios than payload shapes. Figure 1 shows the training data generated using the process discussed above where the labels are shown using colored data points.

B. Training Set on Real Data

This work investigates using real light curve observations taken from the Multichannel Monitoring Telescope (MMT). 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 2 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.

IV. Convolutional Neural Network Classification

A. CNN Design, Training and Classification Methods

As previously discussed, the training data set consists of pairs of simulated light curve measurement vectors and class vectors. Here, The input light curve and output class vector are denoted by $\mathbf{x} \in \mathbb{R}^{1\times m}$ and $\mathbf{y} \in \mathbb{R}^{1\times n_c}$, respectively, where m and n_c denote the number of light curve measurements and number of classes, respectively. Here,a CNN is trained to map the measurement vector, $\mathbf{x}$, to classes, $\mathbf{y}$, using a set of training examples. The CNN designed to learn the functional relationship between simulated 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 (Figure 3). The output layer uses softmax function to map to classification states. Each convolutional layer has the following form:

$$\mathbf{h}^{cov}(\mathbf{x}) = \mathbf{f}\left(\mathbf{x} * W^{cov} + \mathbf{b}^{cov}\right) \tag{6}$$

Where $*$ denotes the convolution operator, W^{cov} denotes the convolution kernel, and $\mathbf{f}$ is the activation function for each layer that adds nonlinearity to the feature vector. This work uses ReLU for convolutional layer activation. The ReLU function is given by $\mathbf{f}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$, where it is zero for negative input values and linear for positive input values. The convolution of $\mathbf{x}$ with W^{cov} defines the output feature map $\mathbf{h}^{cov}(\mathbf{x})$. The number of output maps is determined by

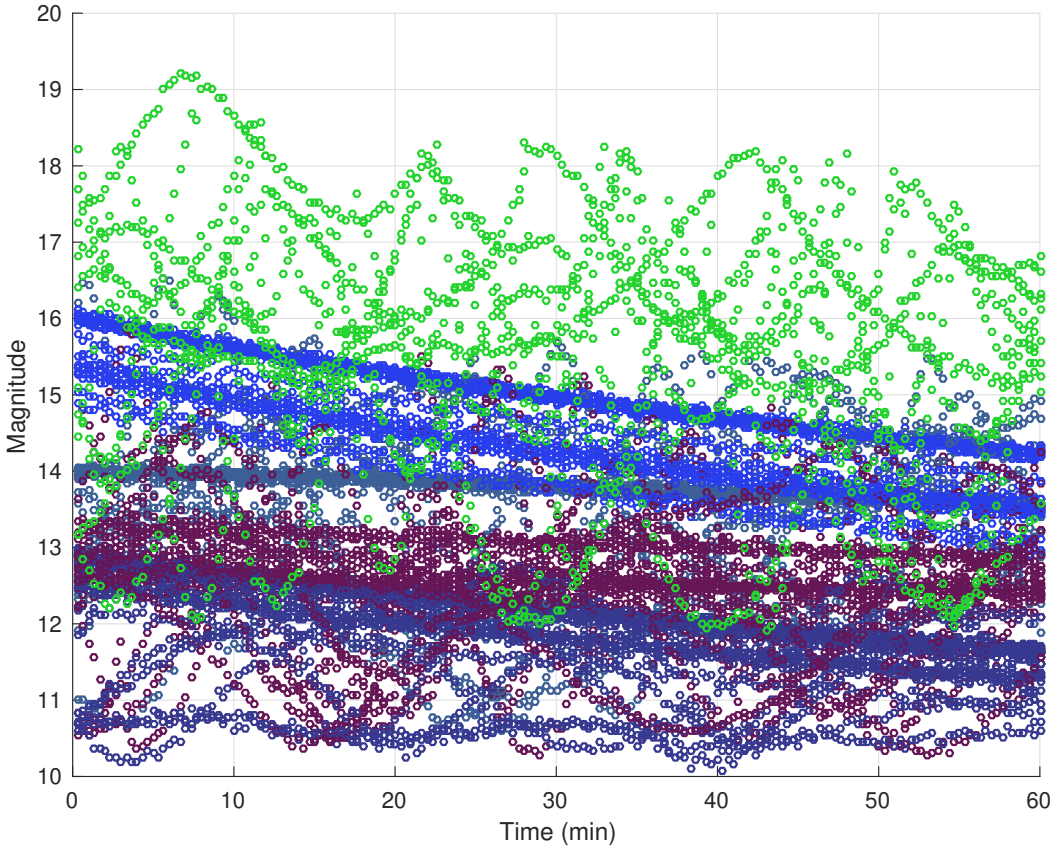


Fig. 1 Labeled Training Data.

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 for input vectors having size $(1, s_x)$ the output vector is given by $(1, s_y)$ and the output vector size can then be calculated from the size of the kernel, s_k, and is given by $s_y = s_x - s_k + 1$. After the convolution is applied the output vector is reduced in size but zero padding is used in this work to keep the size of the feature vectors constant through each convolutional layers.

Then a CNN applies a series of these kernels W^{cov} 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 and is given by

$$\mathbf{h}^{fc}(\mathbf{x}) = \mathbf{f}\left(W^{fc}\mathbf{x} + \mathbf{b}^{fc}\right) \tag{7}$$

After the fully connected layer a softmax function is used to provide outputs in the ranges $(0, 1)$ that add up to 1. The softmax function is defined by

$$y_j\left(\mathbf{h}^{fc}(\mathbf{x})\right) = \frac{\exp(h_j^{fc})}{\sum_i \exp(h_i^{fc})} \tag{8}$$

The convolutional kernel and fully connected output layer parameters are cased into the vector $\boldsymbol{\theta}$. The cost function used for this work is the cross-entropy loss. This loss function minimizes the cross-entropy loss between training outputs and

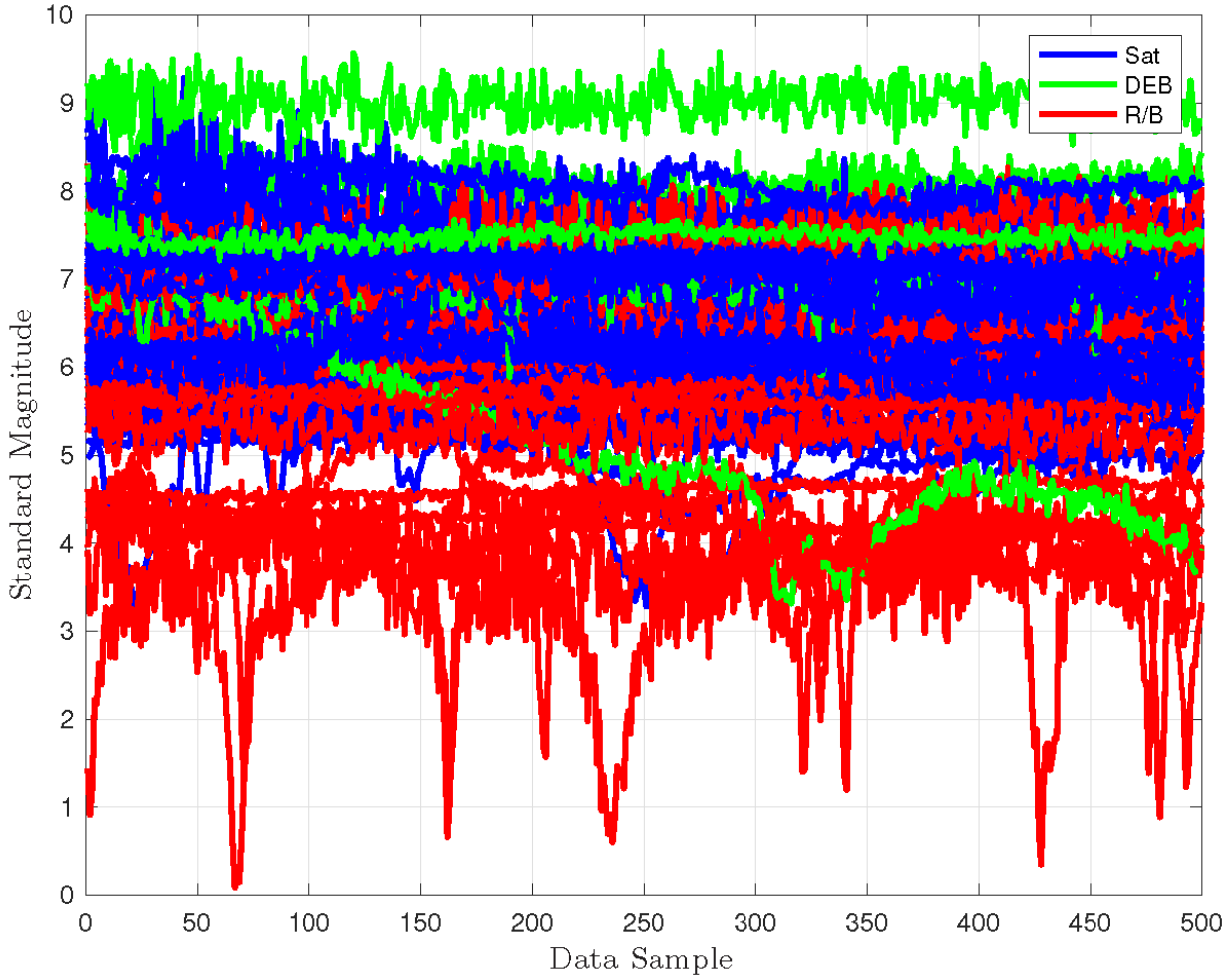


Fig. 2 Labeled Real Training Data from MMT database.

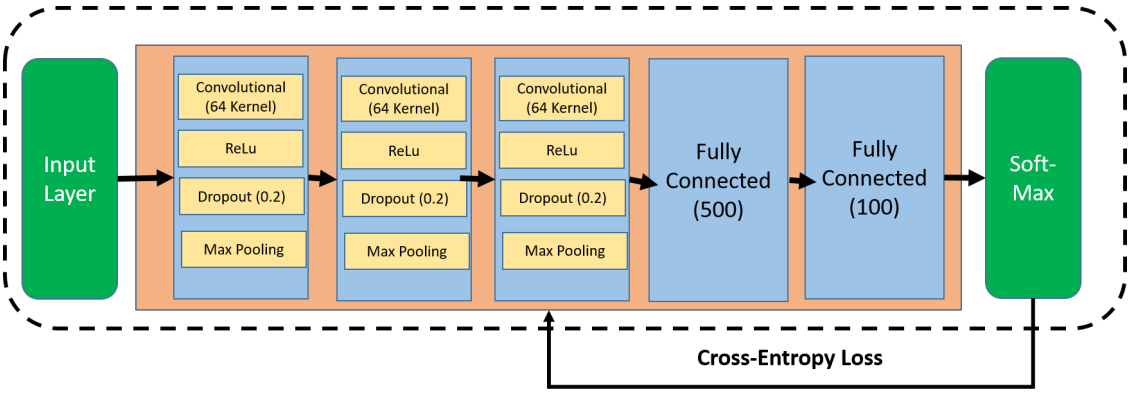


Fig. 3 Network Architecture

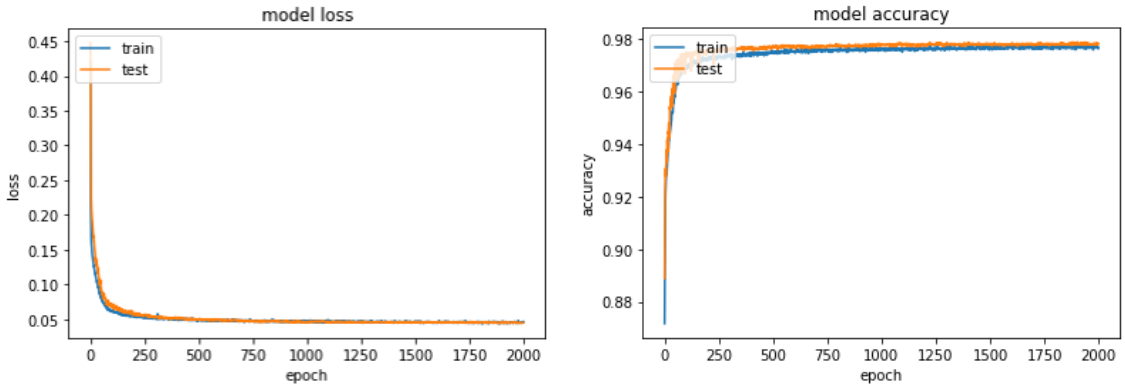
the CNN outputs, and is given by:

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{y},\tilde{\mathbf{y}}) \\
&= -\frac{1}{N}\sum_{i=1}^{N} [\mathbf{y}\log\tilde{\mathbf{y}} + (1-\mathbf{y})\log(1-\tilde{\mathbf{y}})]
\end{aligned}
\tag{9}
$$

where $\tilde{\mathbf{y}}$ are the training examples and $\mathbf{y}$ 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. Figure 3 shows the full architecture of the network used for this work. LeCun [21] showed that stochastic online learning is superior against the full batch mode as it is faster and results in more accurate solutions. The weights for the output layer and the convolutional layer are updated using the following relationship

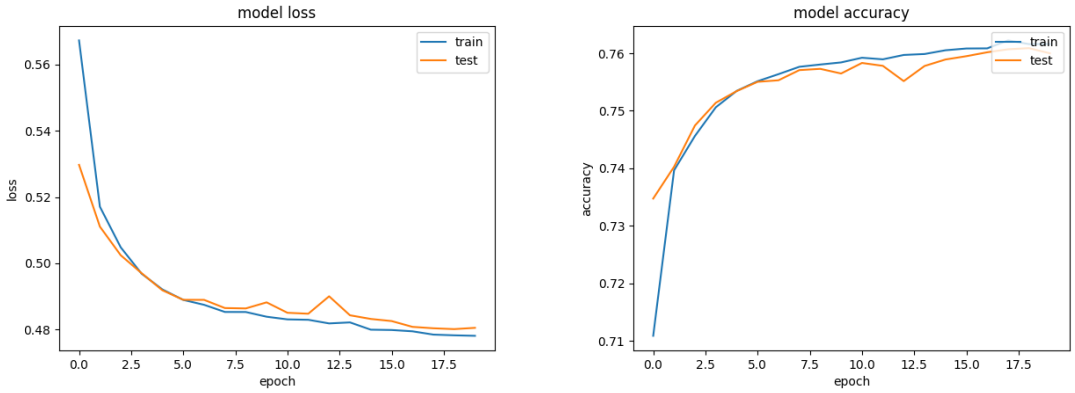
$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \frac{\partial L}{\partial \boldsymbol{\theta}} \tag{10}$$

where t denotes the iteration step and η is the learning rate. The $\frac{\partial L}{\partial \boldsymbol{\theta}}$ is the gradient of the loss with respect to the overall network parameters. This update is calculated for small batches over the entire training sets. Using the small batches allows for small updates to the gradient while reducing the noise in the gradient of individual training samples. This method of updating the parameters is referred to as stochastic gradient descent and the gradient is calculated with error back propagation.



(a) CNN Loss (b) CNN Accuracy

Fig. 4 CNN Classifications Results Simulated Data Case.



(a) CNN Loss (b) CNN Accuracy

Fig. 5 CNN Classifications Results Real-Data Case.

V. Results

A. CNN Design, Training and Classification Results: Simulated Data

The CNN is trained over 8000 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, $e = 2.429 \times 10^{-4}$, $i = 30$ deg, $\omega = \Omega = 0.0$ deg and $M_0 = 91.065$ deg. The simulation epoch is 15-March-2010 at 04:00:00 GST. The initial quaternion and angular rate of the SO are given by

$$\mathbf{q}_I^B \equiv [0.7041 \;\; 0.0199 \;\; 0.0896 \;\; 0.7041]^T$$

and

$$\boldsymbol{\omega}_{B/I}^B = [206.26 \;\; 103.13 \;\; 540.41]^T \text{ deg/hr}$$

.

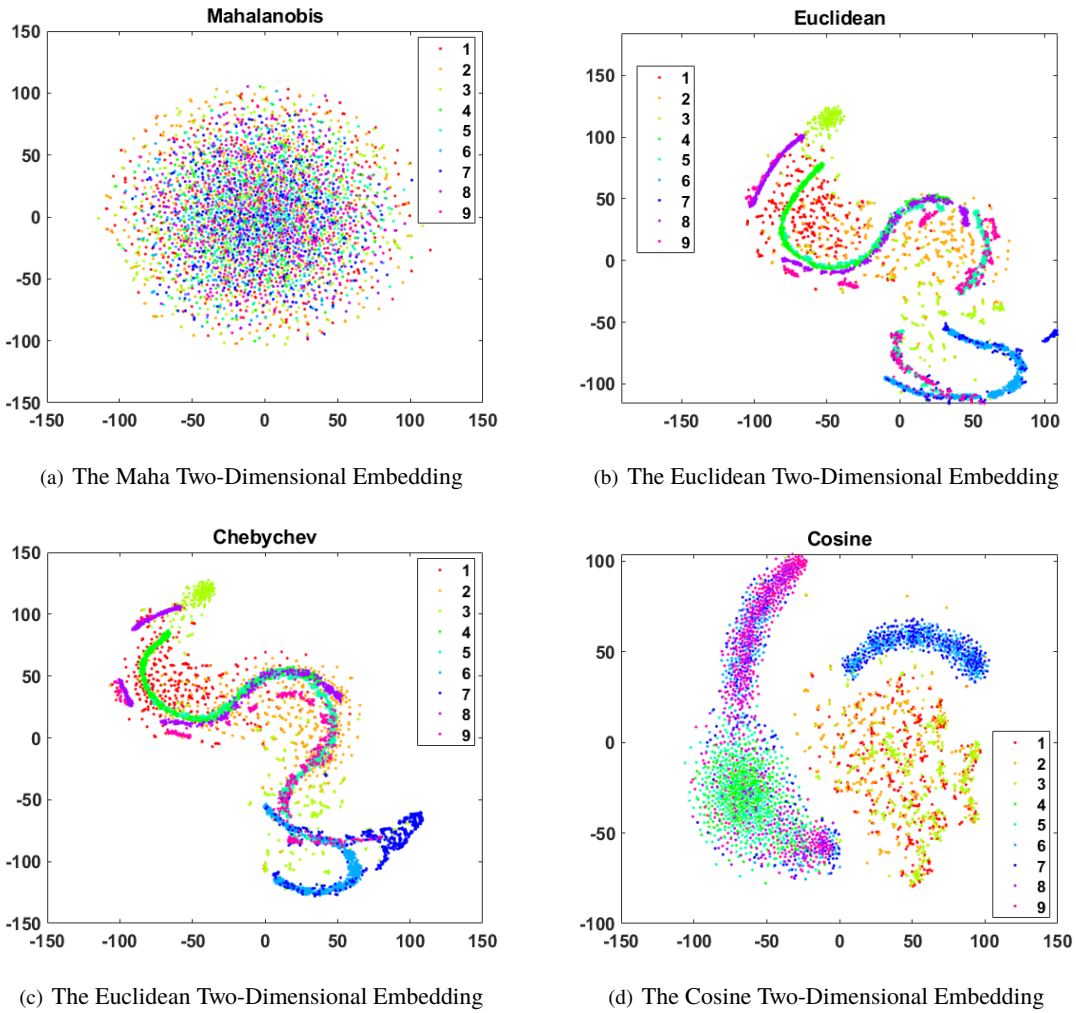
Brightness magnitude are simulated using a ground station located at 20.71° North, 156.26° West longitude and 3,058.6 m altitude. Measurements constructed using instantaneous geometry are corrupted by zero-mean Gaussian white noise with standard deviations of 0.1 for the brightness magnitude[25]. Observations are available every 5 seconds for one hour (Figure 1). 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[26] as backend are employed to design and train the network. The proposed CNN architecture is shown in 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. 4 shows the results of the training process and relative performance of the network.

the layers are given by three convolution layers followed by a fully connected layer. The first three layers use a 32, 12, and 6 unit size kernel, respectively. Both max-pooling (1×4 pooling kernel) and dropout are applied after each convolutional layer. The dropout rates used for this work are 0.7 and 0.5 for the convolutional and fully connected layers respectively. Then the training data consists of simulated light curve measurements as inputs and class states as outputs. For this study we only consider shape classes with one control class, but other classes can be added in the same CNN or with independent CNNs for each class. The classes considered are rocket bodies, controlled payload, uncontrolled payload, and debris.

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 4(a) shows the behavior of the cross-entropy loss as function of the epoch. Figure 4(b) 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 $2000sec$.

B. CNN Design, Training and Classification Results: Real Data

The same 1D-CNN architecture described above (see Figure 3) has been employed to train the deep network on the real data. In this case, the training set comprised 10000 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 2261 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 5(a) shows the behavior of the cross-entropy loss as function of the epoch. Figure 5(b) 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 as 87.5%. Conversely, final accuracy on the test set is 75.4%. Training time is about $4000sec$ and single GPU with 1500 processors.



(a) The Maha Two-Dimensional Embedding

(b) The Euclidean Two-Dimensional Embedding

(c) The Euclidean Two-Dimensional Embedding

(d) The Cosine Two-Dimensional Embedding

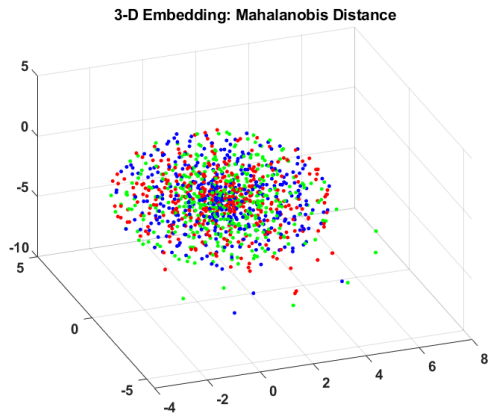
Fig. 6 The Two-Dimensional Embedding of Simulated Case.

VI. Training Set Visualization and Clustering

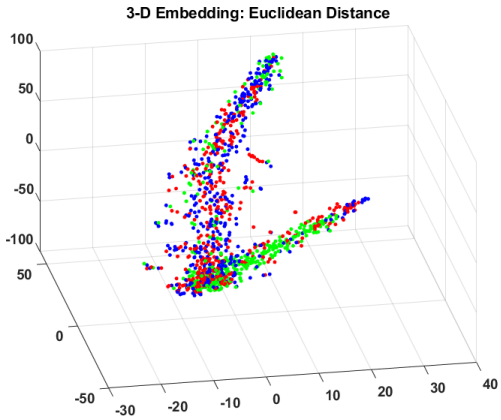
Understanding the training data structure may provide insight into the ability of the deep classifier to recognize the right SO after training. However, the visualization of high dimensional data sets is generally a difficult problem. Indeed, the measured light curves employed in the training set are represented as one-dimensional vectors comprising 500 components. Likewise, the simulated data include 182 components. A conventional approach to data reduction and visualization considers processing the data using PCA, where the most significant two or three components can be visualized in two-dimensional and three dimensional spaces. However, as a linear technique, PCA captures only the linear structures in the data and generally linearly maps high dimensional spaces into low dimensional subspaces. Recently, new probabilistic approaches that preserve the local distance between data while mapping the high-dimensional data into a low-dimensional subspace. One of the most popular appoach to such non-linear data dimensionality reduction is the t-Distributed Stochastic Neighbor Embedding (t-SNE, [30]). We have considered such technique for the visualization of the training data in two and three dimension to understand the structure and clustering of the different RSO classes employed during the learning processes by our deep networks.

A. t-SNE visualization of the measured and simulated training data

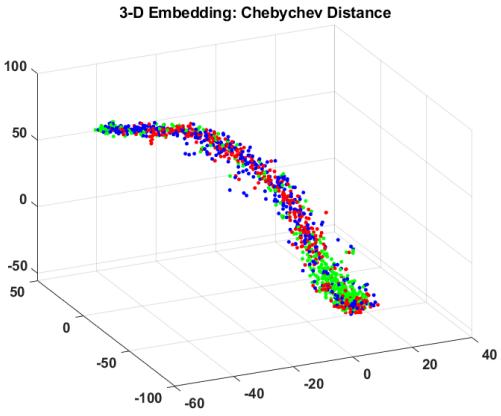
We considered 2-D and -D visualization of he simulated and measured light curves employed in the training of the deep networks. First, the 2-D embedding of sampled simulated training points is considered for visualization. In this



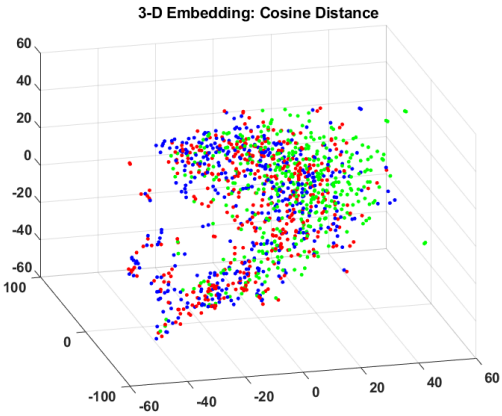
(a) The Mahalanobis Three-Dimensional Embedding



(b) The Euclidean Three-Dimensional Embedding



(c) The Chebychev Three-Dimensional Embedding



(d) The Cosine Three-Dimensional Embedding

Fig. 7 The Three-Dimensional Embedding of the Real-Data Case.

case, the training set comprises 250, 000 simulated light curves distributed among 9 possible classes. Each light curve is represented by a vector of dimension $D = 182$. We sampled 3600 vectors (i.e. 400 per class) and processed the sample via the t-SNE algorithm for four selected distances, i.e. mahalanobis, cosine, chebyshev and euclidean. The resulting 2-D embedding is visualized in figures 6(c),6(d),6(a),6(b). Structure of the data and clustering is readily apparent in all cases but the one employing the Mahalanobis distance. All classes seem to be fairly separated in the 2-D embedding which reflects the clustering the 182-dimensional space of the original light curves. The separation between the different classes may be responsible for the high performances achieved with a non-deep machine learning technique (e.g. bagged decision trees) which is is able to easily find the decision boundaries between the different classes. Although the deep CNN achieves similar performances, the deep architecture does not yield any significant advantage for the case generated by the simulated data. The situation is completely different for the training set generated from measured light curves. Here, the 3-D embedding of sampled measured training points is considered for visualization. In this case, the training set comprises 150, 000 measured light curves distributed among 3 possible classes. Each light curve is represented by a vector of dimension $D = 500$. We sampled 1200 vectors (i.e. 400 per class) and processed the sample via the t-SNE algorithm for four selected distances, i.e. mahalanobis, cosine, chebyshev and euclidean. The resulting 3-D embedding is visualized in figures 7(c),7(d),7(a),7(b). As in the previous case, the structure of the data and the corresponding clustering in 3-D is readily apparent in all cases but the one employing the Mahalanobis distance. Here the clustering of data in the 3D embedding is much more complex and decision boundaries are not well defined as in the previous case

VII. Conclusion

In this paper, a data-driven classification approach based on the Convolutional Neural Network (CNN) scheme is used for Space Object (SO) classification using light curve data. The classification approach determines the shape class from light curve observations of a SO. These classes are rocket bodies, payload, and debris. A set of 1D-CNN capable of ingesting light curves, are separately trained on both simulated data (generated by a physically-based model) and real, observed light curves and performance reported. It is shown that CNN are highly accurate classifiers whenever trained on simulated (controlled) data,yielding 98% accuracy for the selected test set. However, whenever trained on simulated light curves, the CNN tends to lose the advantage over more conventional state-of-the-art machine learning methods . Nevertheless, CNNs significantly outperform the other methods whenever real data are considered.

VIII. Acknowledgment

The first authors wishes to acknowledge support of this work by the Air Force's Office of Scientific Research under Contract Number FA9550-18-1-0115.

References

[1] Linares, R., Jah, M. K., Leve, F. A., Crassidis, J. L., and Kelecy, T., "Astrometric and Photometric Data Fusion For Inactive Space Object Feature Estimation," *Proceedings of the International Astronautical Federation*, Cape Town, South Africa, Sept. 2011, Paper ID: 11340.

[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] Breiman, L., "Random forests," *Machine learning*, Vol. 45, No. 1, 2001, pp. 5–32.

[20] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B., "Support vector machines," *IEEE Intelligent Systems and their applications*, Vol. 13, No. 4, 1998, pp. 18–28.

[21] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, 1998, pp. 2278–2324.

[22] Ashikmin, M. and Shirley, P., "An Anisotropic Phong Light Reflection Model," Tech. Rep. UUCS-00-014, University of Utah, Salt Lake City, UT, 2000.

[23] Shuster, M. D., "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, Oct.-Dec. 1993, pp. 439–517.

[24] Kaasalainen, M. and Torppa, J., "Optimization methods for asteroid lightcurve inversion: I. shape determination," *Icarus*, Vol. 153, No. 1, 2001, pp. 24–36.

[25] Hall, D. T., Africano, J. L., Lambert, J. V., and Kervin, P. W., "Time-Resolved I-Band Photometry of Calibration Spheres and NaK Droplets," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, July 2007, pp. 910–919.

[26] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016.

[27] Dietterich, T. G., "Ensemble methods in machine learning," *International workshop on multiple classifier systems*, Springer, 2000, pp. 1–15.

[28] Quinlan, J. R., "Induction of decision trees," *Machine learning*, Vol. 1, No. 1, 1986, pp. 81–106.

[29] Jolliffe, I., "Principal component analysis," *International encyclopedia of statistical science*, Springer, 2011, pp. 1094–1096.

[30] van der Maaten, L. and Hinton, G., "Visualizing High-Dimensional Data Using t-SNE," 2008.

[31] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, 2014, pp. 3320–3328.