# Space Objects Classification via Light-Curve Measurements: Deep Convolutional Neural Networks and Model-based Transfer Learning

Roberto Furfaro[*]
*University of Arizona, Tucson, AZ, 85721*
Richard Linares[†]
*Massachusetts Institute of Technology, Cambridge, MA 02139*
Vishnu Reddy[‡]
*University of Arizona, Tucson, AZ, 85721*

**Abstract**

Developing a detailed understanding of the Space Object (SO) population is a fundamental goal of Space Situational Awareness (SSA). The current SO catalog includes simplified characteristic for the observed space objects, mainly the solar radiation pressure and/or drag ballistic coefficients. Such simplified description limits the dynamic propagation model used for predicting the state of motion of SO to models that assume cannon ball shapes and generic surface properties. The future SO catalog and SSA systems will have to be capable of building a detailed picture of SO characteristics. Traditional measurement sources for SO tracking, such as radar and optical, provide information on SO characteristics. These measurements have been shown to be sensitive to shape, attitude, angular velocity, and surface parameters. State-of-the-art in the literature has been advanced over the past decades and in recent years seen the development of multiple models, nonlinear state estimation, and full Bayesian inversion approaches for SO characterization. The key shortcoming of approaches in literature is their overall computational cost and the limited flexibility to deal with a larger and larger amount of data.

In this paper, we present a data-driven method to classification of SO based on a deep learning approach that takes advantage of the representational power of deep neural networks. Here, we design, train and validate a Convolutional Neural Network (CNN) capable of learning to classify SOs from collected light-curve measurements. The proposed methodology relies a physically-based model capable of accurately representing SO reflected light as function of time, size shape and state of motion. The model generates thousands of light-curves per selected class of SO which are employ to train a deep CNN to learn the functional relationship between light curves and SO class. Additionally, a deep CNN is trained using real SO light curves to evaluate the performance on a real, but limited training set. CNNs are compared with more conventional machine learning techniques (bagged trees, support vector machines) and are shown to outperform such methods especially when trained on real data. The concept of model-based transfer learning is proposed as possible path forward to increase the accuracy and speed-up the training process.

## 1 Introduction

Motivated by the U.S. Airforce mission to control, protects and maintain access to space, Space Situational Awareness (SSA) has been recently become an important research topic. 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 (SO). The SO catalog maintained by JSpoC currently includes upward of 22,000 SO, with 1,100 of such objects being actively

[*]Professor, Department of Systems & Industrial Engineering, Department of Aerospace and Mechanical Engineering. Email: robertof@email.arizona.edu, Member AIAA

[†]Charles Stark Draper Assistant Professor, Department of Aeronautics and Astronautics Email: linaresr@mit.edu, Member AIAA.

[‡]Associate Professor, , Lunar and Planetary Laboratory. Email: reddy@lpl.arizona.edu

Copyright © 2018 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

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 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) 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, 7, 8], 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. Subsequently, available and labeled real light curves are employed to directly train a specified CNN architecture. The data-driven deep learning approach is compared with more conventional machine learning techniques (e.g. random forest ( [19]) and Support Vector Machine (SVM, [20]) to show that hierarchical feature learning is the key to building successful discriminative models.

Copyright © 2018 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com

## 2 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.

## 3 Light Curve and Dynamics of Space Objects Modeling

There are a number of models used for simulating light curve measurements in literature and Ref. 12 provides a good summary of the most popular ones adopted for space objects (SOs) applications. These models differ in the physics that they represent and their level of complexity, but for SOs applications the ability to model specular reflection and complex shapes while converting energy is desirable. The Ashikhmin-Shirley [22] (AS) model has all the desirable properties while producing realistic SOs light curve. This model is based on the bidirectional reflectance distribution function (BRDF) which models light distribution scattered from the surface due to the incident light. The BRDF at any point on the surface is a function of two directions, the direction from which the light source originates and the direction from which the scattered light leaves the observed surface. The model in Ref. 22 decomposes the BRDF into a specular component and a diffuse component. The two terms sum to give the total BRDF:

$$f_r = (dR_d + sR_s) \tag{2}$$

which depends on the diffuse bidirectional reflectance ($R_d$) and the specular bidirectional reflectance ($R_s$) and the fraction of each to the total ($d$ and $s$ respectively where $d + s = 1$). Where $i$ denotes the $i^{\text{th}}$ facet of the SOs. Each facet contributes independently to the brightness and total brightness is the sum over each facet's contribution. The diffuse component represents light that is scattered equally in all directions

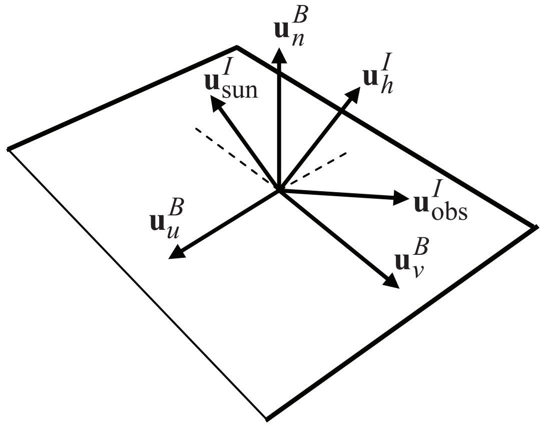


Figure 1: Reflection Geometry

(Lambertian) and the specular component represents light that is concentrated about some direction (mirror-like). Reference 22 develops a model for continuous arbitrary surfaces but simplifies for flat surfaces. This simplified model is employed in this work as shape models are considered to consist of a finite number of flat facets. Therefore the total observed brightness of an object becomes the sum of the contribution from each facet. In each model, however, $c = \mathbf{V}^T\mathbf{H}$, $\rho$ is the diffuse reflectance ($0 \leq \rho \leq 1$), and $F_0$ is the specular reflectance of the surface at normal incidence ($0 \leq F_0 \leq 1$). To be used as a prediction tool for brightness and radiation pressure calculations, an important aspect of the BRDF is energy conservation. For energy to be conserved, the integral of the BRDF times $\cos(\theta_r)$ over all solid angles in the hemisphere with $\theta_r \leq 90$ needs to be less than unity, with

$$\int_0^{2\pi}\int_0^{\pi/2} f_r \cos(\theta_r)\sin(\theta_r)\, d\theta_r d\phi = R_d + R_s \tag{3}$$

For the BRDF given in Eq. (2), this corresponds to constant values of $R_d = \rho d$ and $R_s = sF_0$. The remaining energy not reflected by the surface is either transmitted or absorbed. In this paper it is assumed the transmitted energy is zero. The diffuse bidirectional reflectance is then calculated as follows:

$$\begin{aligned} R_d = \frac{28\rho}{23\pi}(1 - sF_0)\left(1 - \left(1 - \frac{\mathbf{N}^T\mathbf{L}}{2}\right)^5\right)\cdot \\ \left(1 - \left(1 - \frac{\mathbf{N}^T\mathbf{V}}{2}\right)^5\right) \end{aligned} \tag{4}$$

where

$$F = F_0 + \left(\frac{1}{s} - F_0\right)(1 - c)^5 \tag{5}$$

The vectors $\mathbf{L}(t_i)$ and $\mathbf{V}(t_i)$ denote the unit vector from the SO to the Sun and the unit vector from the SO to the observer, respectively, and together they define the observation geometry (shown in Figure 1). In addition to $d$, $\rho$, and $F_0$, the Ashikhmin-Shirley BRDF has two exponential factors ($n_u$, $n_v$) that define the reflectance properties of each surface. The Ashikhmin-Shirley diffuse and specular reflectivities are not constant but rather complicated functions of illumination angle, exponential factor, and the diffuse and specular reflectances. In all cases, however, $R_d + R_s \leq 1$, so energy is conserved. The parameters of the Phong model that dictate the directional (local horizontal or vertical) distribution of the specular terms are $n_u$ and $n_v$. The specular bidirectional reflectance for the AS model is given by

$$R_s = \frac{F\sqrt{(n_u + 1)(n_v + 1)}}{8c\pi \max[\mathbf{N}^T\mathbf{L}, \mathbf{N}\mathbf{V}]}(\cos(\alpha))^\gamma \tag{6}$$

where $\gamma = n_u \cos^2(\beta) + n_v \sin^2(\beta)$.

The apparent magnitude of an SO is the result of sunlight reflecting off of its surfaces along the line-of-sight to an observer. First, the fraction of visible sunlight that strikes an object (and is not absorbed) is computed by

$$F_{\text{sun}}(i) = C_{\text{sun,vis}} \left(\mathbf{u}_n^I(i) \cdot \mathbf{u}_{\text{sun}}^I\right) \tag{7}$$

where $C_{\text{sun,vis}} = 1062$ W/m$^2$ is the power per square meter impinging on a given object due to visible light striking the surface. If either the angle between the surface normal and the observer's direction or the angle between the surface normal and Sun direction is greater than $\pi/2$ then there is no light reflected toward the observer. If this is the case then the fraction of visible light is set to $F_{\text{sun}}(i) = 0$. Next, the fraction of sunlight that strikes an object that is reflected must be computed:

$$F_{\text{obs}}(i) = \frac{F_{\text{sun}}(i)\rho_{\text{total}}(i)\mathcal{A}(i)\left(\mathbf{u}_n^I(i) \cdot \mathbf{u}_{\text{obs}}^I\right)}{\|\mathbf{d}^I\|^2} \tag{8}$$

The reflected light of each facet is now used to compute the total photon flux, which is measured by an observer:

$$\tilde{F} = \left[\sum_{i=1}^{N} F_{\text{obs}}(i)\right] + v_{\text{CDD}} \tag{9}$$

where $v_{\text{CDD}}$ is the measurement noise associated with flux measured by a Charge Coupled Device (CCD) sensor. The total photon flux is then used to compute the apparent brightness magnitude

$$m_{\text{app}} = -26.7 - 2.5\log_{10}\left|\frac{\tilde{F}}{C_{\text{sun,vis}}}\right| \tag{10}$$

where $-26.7$ is the apparent magnitude of the Sun.

A number of parameterizations exist to specify attitude, including Euler angles, quaternions, and Rodrigues parameters. [23] This paper uses the quaternion, which is based on the Euler angle/axis parameterization. The quaternion is defined as $\mathbf{q} \equiv [\boldsymbol{\varrho}^T \quad q_4]^T$ with $\boldsymbol{\varrho} = \hat{\mathbf{e}}\sin(\nu/2)$, and $q_4 = \cos(\nu/2)$, where $\hat{\mathbf{e}}$ and $\nu$ are the Euler axis of rotation and rotation angle, respectively. Clearly, the quaternion must satisfy a unit norm constraint, $\mathbf{q}^T\mathbf{q} = 1$. In terms of the quaternion, the attitude matrix is given by

$$A(\mathbf{q}) = \Xi^T(\mathbf{q})\Psi(\mathbf{q}) \tag{11}$$

where

$$\Xi(\mathbf{q}) \equiv \begin{bmatrix} q_4 I_{3\times3} + [\boldsymbol{\varrho}\times] \\ -\boldsymbol{\varrho}^T \end{bmatrix} \tag{12a}$$

$$\Psi(\mathbf{q}) \equiv \begin{bmatrix} q_4 I_{3\times3} - [\boldsymbol{\varrho}\times] \\ -\boldsymbol{\varrho}^T \end{bmatrix} \tag{12b}$$

with

$$[\mathbf{g}\times] \equiv \begin{bmatrix} 0 & -g_3 & g_2 \\ g_3 & 0 & -g_1 \\ -g_2 & g_1 & 0 \end{bmatrix} \tag{13}$$

for any general $3 \times 1$ vector $\mathbf{g}$ defined such that $[\mathbf{g}\times]\mathbf{b} = \mathbf{g} \times \mathbf{b}$.

The rotational dynamics are given by the coupled first order differential equations:

$$\dot{\mathbf{q}}_I^B = \frac{1}{2}\Xi(\mathbf{q}_I^B)\boldsymbol{\omega}_{B/I}^B \tag{14a}$$

$$\dot{\boldsymbol{\omega}}_{B/I}^B = J_{\text{SO}}^{-1}\left(\mathbf{T} + \mathbf{T}_{\text{srp}}^B - \left[\boldsymbol{\omega}_{B/I}^B\times\right] J_{\text{SO}}\boldsymbol{\omega}_{B/I}^B\right) \tag{14b}$$

where $\boldsymbol{\omega}_{B/I}^B$ is the angular velocity of the SO with respect to the inertial frame, expressed in body coordinates, $J_{\text{SO}}$ is the inertia matrix of the SO. The vectors $\mathbf{T}_{\text{srp}}^B$ and $\mathbf{T}$ are the net torques acting on the SO due to SRP expressed in body coordinates and the control torque, respectively.

# 4 Training Set Generation

## 4.1 Simulating Labeled Training Data

For the simulated case, the labeled training data samples are generated using the light curve model discussed above. 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 $h$.

$$h_{\text{regular}} = (h_{\min} + 0.01) + (h_{\max} - h_{\min} - 0.01)\mathcal{U}\,[0,1] \tag{15a}$$

$$s_{\text{regular}} = (s_{\min} + 0.01) + (s_{\max} - s_{\min} - 0.01)\mathcal{U}\,[0,1] \tag{15b}$$

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

$$J_{\text{triangle}} = m_{\text{SO}} \begin{bmatrix} \frac{s^2}{24} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{s^2}{24} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{s^2}{12} \end{bmatrix} \tag{16a}$$

$$J_{\text{square}} = m_{\text{SO}} \begin{bmatrix} \frac{s^2}{12} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{s^2}{12} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{s^2}{6} \end{bmatrix} \tag{16b}$$

$$J_{\text{hexagon}} = m_{\text{SO}} \begin{bmatrix} \frac{5s^2}{24} + \frac{h^2}{12} & 0 & 0 \\ 0 & \frac{5s^2}{24} + \frac{h^2}{12} & 0 \\ 0 & 0 & \frac{5s^2}{24} \end{bmatrix} \tag{16c}$$

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

$$J_{\text{cuboid}} = \frac{m_{\text{SO}}}{12} \begin{bmatrix} s_2^2 + h^2 & 0 & 0 \\ 0 & s_1^2 + h^2 & 0 \\ 0 & 0 & s_1^2 + s_2^2 \end{bmatrix} \tag{17}$$

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.

$$\begin{aligned} J_{\text{rocket}} = m_{\text{SO}} \Bigg\{ & \frac{V_{\text{cyl}}}{V_{\text{tot}}} \text{diag}\left[\frac{1}{12}(3r^2 + l^2), \frac{1}{12}(3r^2 + l^2), \frac{r^2}{2}\right] \\ & + \frac{V_{\text{top}}}{V_{\text{tot}}} \text{diag}\left[\frac{1}{12}(3r^2 + l^2), \frac{1}{12}(3r^2 + l^2), \frac{r^2}{2}\right] \\ & + \left(\frac{V_{\text{top}}}{V_{\text{tot}}}\left(\frac{l}{2} + \frac{3r}{8}\right) + \frac{V_{\text{cyl}}}{V_{\text{tot}}}\left(\frac{l}{2} - \frac{3r}{8}\right)\right)\left(I_{3\times3} - \mathbf{e}\mathbf{e}^T\right) \\ & + 2\frac{V_{\text{top}}}{V_{\text{tot}}} r^2 \text{diag}\left[\frac{83}{320}, \frac{83}{320}, \frac{2}{5}\right] \Bigg\} \end{aligned} \tag{18}$$

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 2 shows the training data generated using the process discussed above where the labels are shown using colored data points.

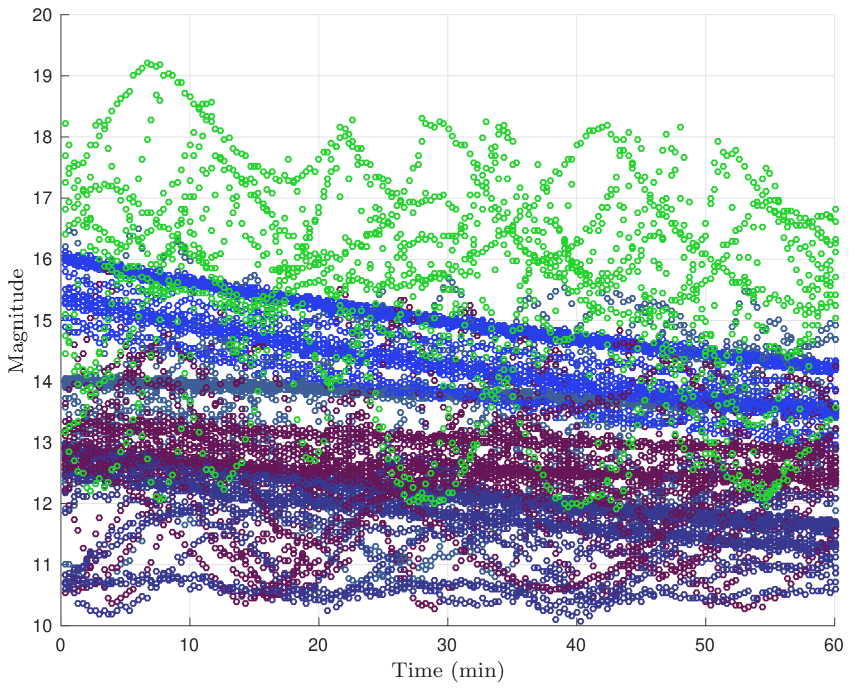


Figure 2: Simulated Labeled Training Data.

### 4.2 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 3 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.

## 5 Convolutional Neural Network Classification

### 5.1 CNN Design, Training and Classification Results: Simulated Data

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,

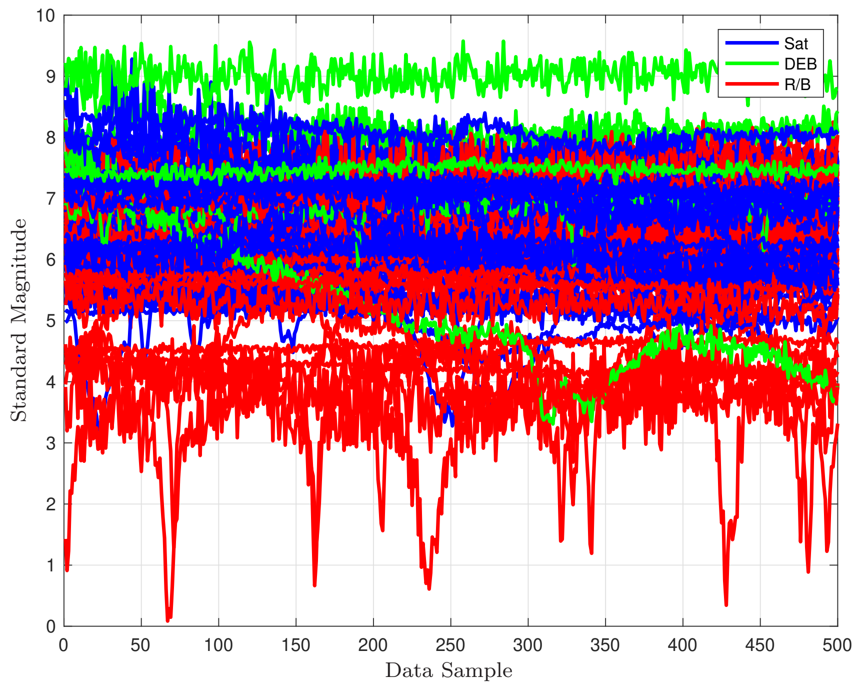


Figure 3: Labeled Real Training Data from MMT database.

max-pooling, and two fully connected layer with ReLU activation (Figure 4). The output layer uses softmax function to map to classification states. Each convolutional layer has the following form:

$$\mathbf{h}^{cov}(\mathbf{x}) = \mathbf{f}\left(\mathbf{x} * W^{cov} + \mathbf{b}^{cov}\right) \tag{19}$$

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 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 $1 \times 4$ kernel between each convolution layer is used. The max-pooling operation convolves the $1 \times 4$ kernel over the output of each convolutional layer returning the max of the outputs over the $1 \times 4$ 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

$$\mathbf{h}^{fc}(\mathbf{x}) = \mathbf{f}\left(W^{fc}\mathbf{x} + \mathbf{b}^{fc}\right) \tag{20}$$

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

$$y_j\left(\mathbf{h}^{fc}(\mathbf{x})\right) = \frac{\exp(h_j^{fc})}{\sum_i \exp(h_i^{fc})} \tag{21}$$

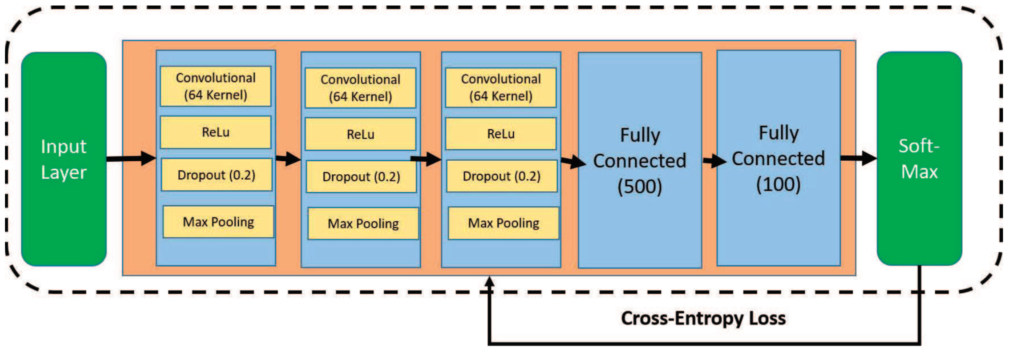


Figure 4: Network Architecture

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 the CNN outputs, and is given by:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{y},\tilde{\mathbf{y}}) \\ &= -\frac{1}{N}\sum_{i=1}^{N}[\mathbf{y}\log\tilde{\mathbf{y}} + (1-\mathbf{y})\log(1-\tilde{\mathbf{y}})] \end{aligned} \tag{22}$$

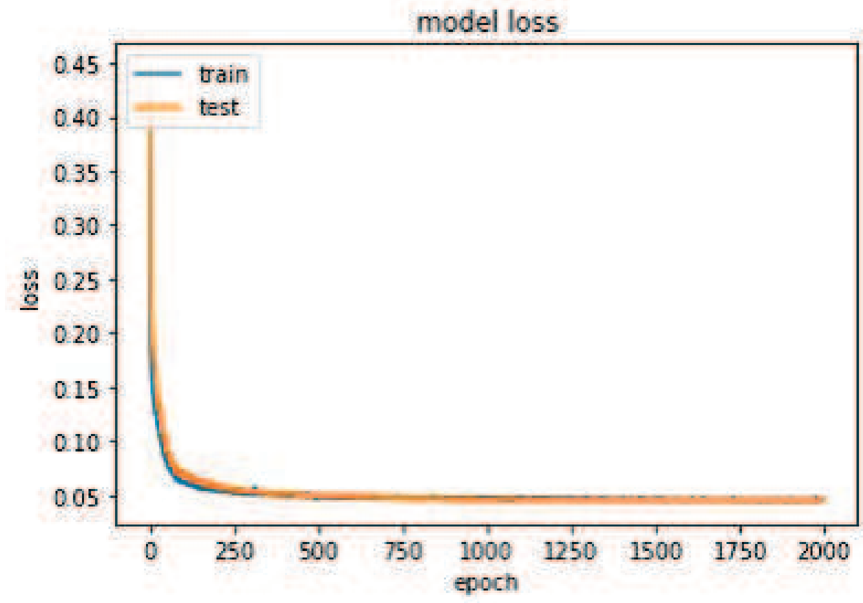
where $\tilde{\mathbf{y}}$ are the training examples and $\mathbf{y}$ are the outputs from the CNN. Then the CNN classification approach is trained by stochastic gradient descent by minimizing the cross-entropy loss from the outputs compared to the labelled data. Figure 4 shows the full architecture 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

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta\frac{\partial L}{\partial \boldsymbol{\theta}} \tag{23}$$

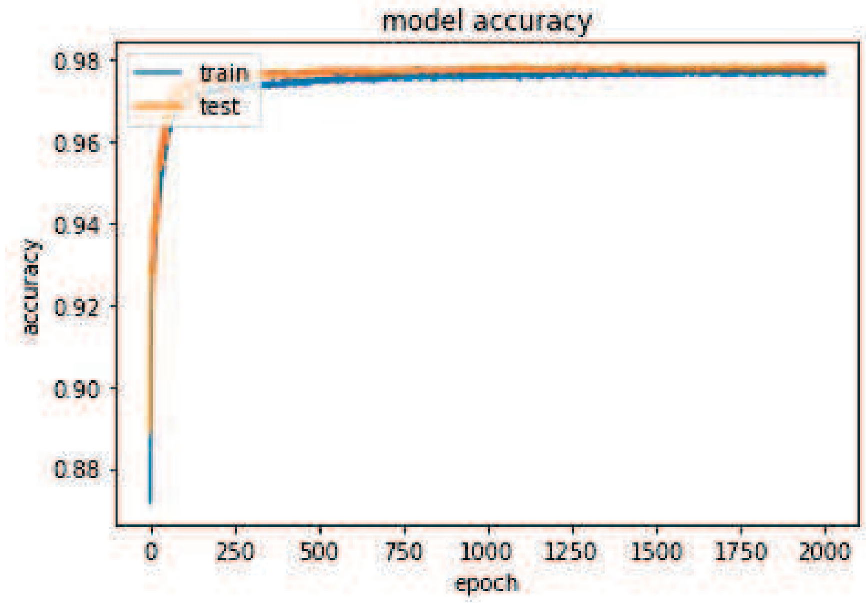
where $t$ denotes the iteration step and $\eta$ is the learning rate. The $\frac{\partial L}{\partial \boldsymbol{\theta}}$ is the gradient of the loss with respect to the overall network parameters. This update is calculated for small batches over the entire training sets. Using the small batches allows 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.

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 $\boldsymbol{\omega}_{B/I}^B = [206.26 \;\; 103.13 \;\; 540.41]^T$ 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 2). 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 4. 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



(a) CNN Loss



(b) CNN Accuracy

Figure 5: CNN Classifications Results

layers and 0.5 for fully connected layers. Training occurred for 2000 epochs using stochastic gradient descent and mini-batch of 128 samples. 5 shows the results of the training process and relative performance of the network.

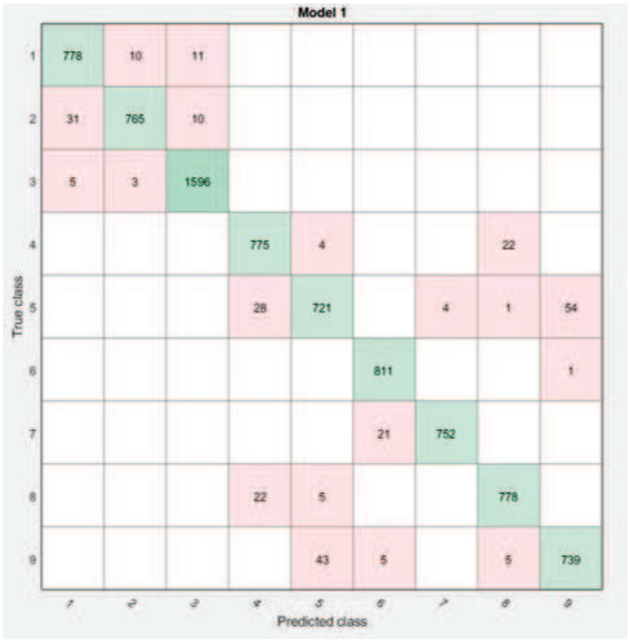
the layers are given by three convolution layers followed by a fully connected layer. The first three layers use a 32, 12, and 6 unit size kernel, respectively. Both max-pooling ($1\times4$ pooling kernel) and dropout are applied after each convolutional layer. The dropout rates used for this work are 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 5 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 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. 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 $2000 sec$. Importantly, we compared the CNN with two machine learning techniques that represent somewhat the state of the art of automatic classification techniques. First method is called Bagged Trees and it is an ensemble method. The idea behind ensemble methods is to combine many weak learners into a highly accurate single ensemble algorithm. Thus, we can consider a classification ensemble as a machine learning model comprising a weighted combination of many individual classification algorithms. Here, we consider a type of ensemble algorithm called *Bagging* (where *Bag* stands for *Bootstrap aggregation* [27]) combined with a basic decision tree algorithm [28]. The bagging process on a decision tree works by generating many bootstrap replicas of the training dataset and then grow decision trees on such replicas. Each of the bootstrap replicas are obtained by randomly choosing N observations and considering N replacements (here N is the training set size). The method works by simply training each of the individual weak learners on resampled version of the training set. The overall response of the model is obtained by average prediction over the ensemble of individual learners (For further information, see [**?**]). The second techniques is the well-known Support Vector Machines (SVM, [20]) adapted for the multi-class case. SVM generally classify data by finding the hyperplane that separate two classes by the largest margin. Non-linear transformations are generally employed when the classes are non-separable by a simple (linear) hyperplane. In this case, a variety a kernels are available to execute the transformation. For the bagged decision trees, 100 weak learners (trees) have been selected. For each of the trees, we considered 9,999 as the maximum number of possible splits. The training set has been loaded in the MATLAB classification learner app. Data are pre-processed using Principal Componenent Analysis (PCA, [29]) for dimensionality reduction. To explain 98% of the variance, ten (10) principal components have been kept. A 5-fold validation approach has been implement to protect against overfitting. The bagged model was trained in parallel fashion over an 8-core intel i7 operating @2.9Hz. The model achieved an accuracy of 96.4% over the training set. training time was recorded to be $43 sec$. Figure 6 shows the resulting confusion matrix. Similarly, an SVM with cubic kernel has been selected for training. PCA and 5-fold validation has been applied and the algorithm was run on the same machine. The SVM model achieved an accuracy of 95.3% over the training set. training time was recorded to be $32 sec$. Although the CNN is shown to be slightly more accurate, the accuracy is generally comparable.

## 5.2 CNN Design, Training and Classification Results: Real Data

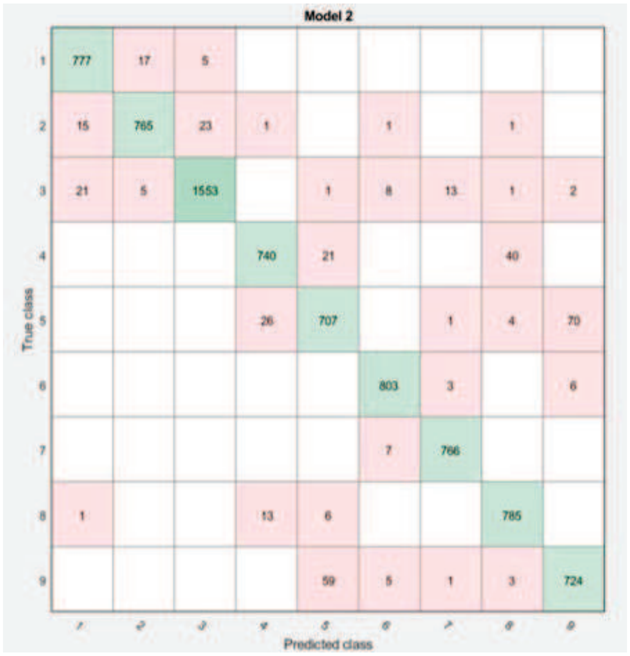
The same 1D-CNN architecture described above (see Figure 4) has been employed to train the deep network on the real data. In this case, the training set comprised 10000 samples and is subdivided in three possible classes,i.e. rocket bodies, debris, other. In the class other, it is included anything that 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 7 shows the results of the training process. Figure **??** 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 $4000 sec$ and single GPU with 1500 processors.

For comparison and similarly to the simulated light curve case, bagged trees (ensemble of 100 weak lerners) and SVM with cubic kernel are developed and trained on the same training set comprising real data. PCA has been considered to retain the principal components that can explain 98% of the covariance in the data, resulting in 29 components out of 500. For both cases, we considered a 5-fold method as protection against overfitting. Performance for both cases are reported to be rather poor. Indeed, bagged trees achieves an accuracy on the training set of 63.5% (training time: 143 sec) whereas SVM achieves an accuracy of 53.8% (training time: 1894 sec).Figure 8 show the resulting confusion matrices. 1D-CNN performs better



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 778 | 10 | 11 | | | | | | |
| 2 | 31 | 765 | 10 | | | | | | |
| 3 | 5 | 3 | 1596 | | | | | | |
| 4 | | | | 775 | 4 | | | 22 | |
| 5 | | | | 28 | 721 | | 4 | 1 | 54 |
| 6 | | | | | | 811 | | | 1 |
| 7 | | | | | | 21 | 752 | | |
| 8 | | | | 22 | 5 | | | 778 | |
| 9 | | | | | 43 | 5 | | 5 | 739 |

(a) Bagged Trees



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 777 | 17 | 5 | | | | | | |
| 2 | 15 | 765 | 23 | 1 | | 1 | | 1 | |
| 3 | 21 | 5 | 1553 | | 1 | 8 | 13 | 1 | 2 |
| 4 | | | | 740 | 21 | | | 40 | |
| 5 | | | | 26 | 707 | | 1 | 4 | 70 |
| 6 | | | | | | 803 | 3 | | 6 |
| 7 | | | | | | 7 | 766 | | |
| 8 | 1 | | | 13 | 6 | | | 785 | |
| 9 | | | | | 59 | 5 | 1 | 3 | 724 |

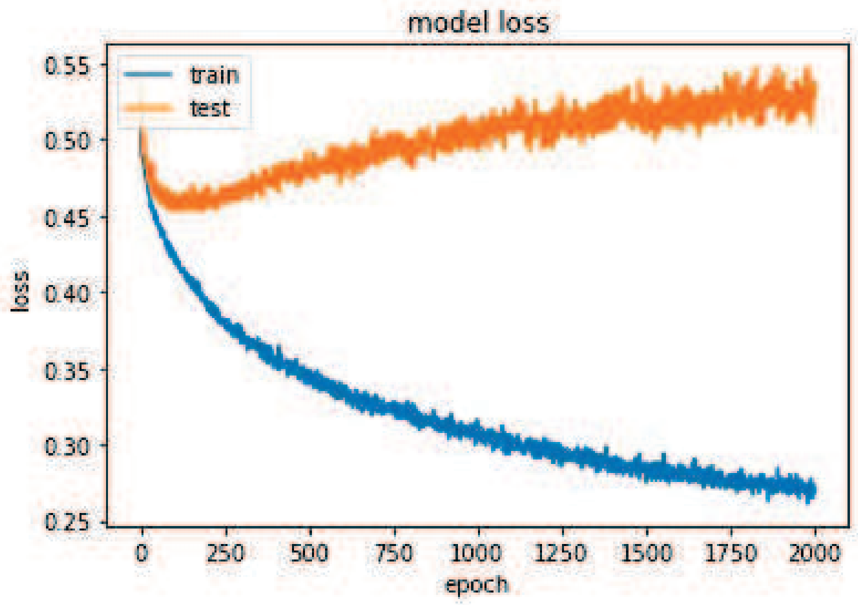
(b) SVM with cubic kernel

Figure 6: Confusion matrices for Bagged Trees and Multi-class SVM: simulated data

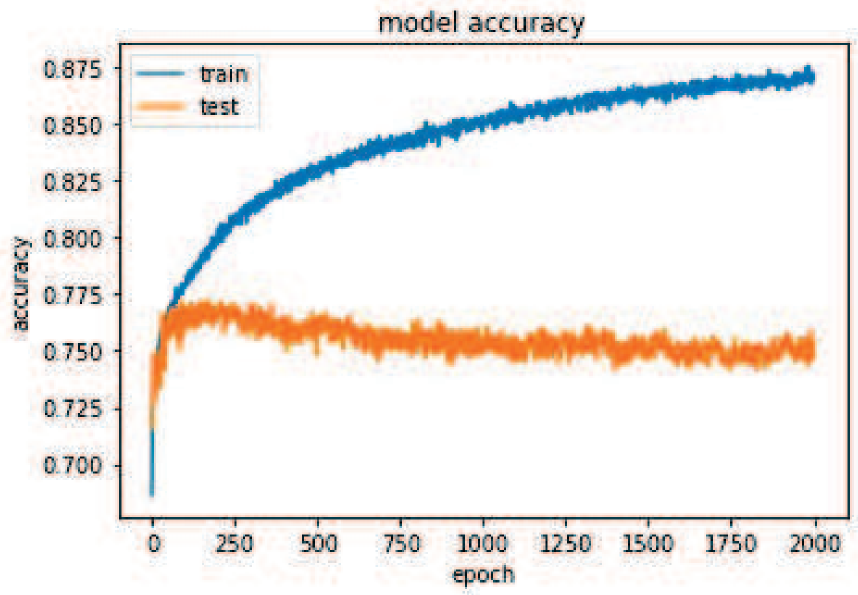
due to the superior ability to automatically extract a hierarchical set of features comprising the light curve signal.

## 6 Model-based Transfer Learning

The 1D-CNNs have been designed and trained separately on different data sets. The first dataset is comprised of simulated data generated using a physically-based reflectance model. The second dataset has been



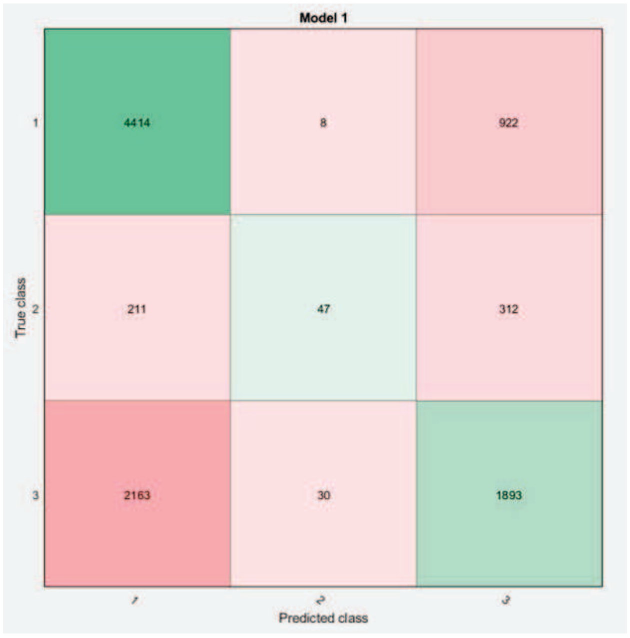
(a) CNN Loss



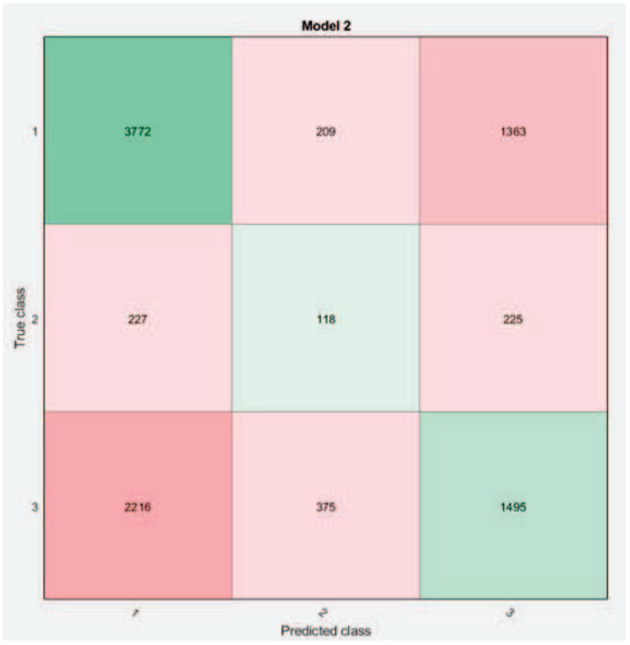
(b) CNN Accuracy

Figure 7: CNN Classifications Results: Real Data

extracted from real light curve data. Although a set of 10,000 light curves have been simulated, one can potentially generate much larger datasets comprising millions of light curves with automatic labeling. Indeed, the physically model can be employed to produce a large number of classes of SO each with light curves simulated under different conditions (e.g. observing conditions, orbital regime,material reflectance etc.). The physical model enables controlled data generation and ensure sufficient training points necessary to model complex relationship between light curves and classes. However, physically-based models are affected by *modeling error* which may be a limitation, especially when applied to classify SO based on observed light curves. it is expected that although a CNN performs well on simulated light curves, it may not perform well when, after training, it is applied to predict the SO class. Conversely, collected data are generally scarce. Since CNN are comprised of millions of trainable parameters, they tend to overfit the data (which could be



(a) Bagged Trees



(b) SVM with cubic kernel

Figure 8: Confusion matrices for Bagged Trees and Multi-class SVM: real data

a possible explanation of why the discrepancy in performance between training and test set, see Figure 7). Importantly, since deep architectures autonomously learn the relevant features in a data-driven hierarchical fashion, a CNN trained on physically-based model, may have learned the basic features. This approach is conducive to the Deep Transfer Learning, where deep networks are pre-trained on a large source of data and then applied to a new but limited target set [30]. In transfer learning, one first trains a base networks on a base dataset and task; subsequently, one transfers the learned features on a smaller target set trained on a much smaller dataset. Generally, this process works well if the learned features are general,i.e. capture the basic behavior of the system. It is conjectured that if one employs the simulated light curves as base training set to train a physically-based CNN, the learned features may capture the basic physics and may be general enough to be successfully transfered to a target network trained on a much smaller dataset, yet yielding superior accuracy. In general, training the target network may be much faster and efficient. This conjecture is current investigated and may be reported in future meetings.

## 7 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 (e.g. bagged trees, SVM). However, CNNs significantly outperform the other methods whenever real data are considered. Indeed, on real light curve test sets, CNN achieves 75% accuracy whereas bagged trees reports an accuracy of 63% and SVM an accuracy of 53%. The concept of transfer learning has been proposed as possible idea to generalize deep networks that are trained on a much smaller dataset comprising only real light curves.

## 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.

*Copyright © 2018 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com*

[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.

*Copyright © 2018 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com*

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

[30] 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.

*Copyright © 2018 Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) – www.amostech.com*