Game Theoretic Synthetic Data Generation for Machine Learning Based Satellite Behavior Detection

Dan Shen[a], Carolyn Sheaff[b], Genshe Chen[*a], Mengqing Guo[a], Nichole Sullivan[a] Erik Blasch[c], and Khanh Pham[d]

[a] Intelligent Fusion Technology, Inc., 20271 Goldenrod Lane, Germantown, MD USA 20876;
[b] Information Directorate, Air Force Research Laboratory (AFRL), Rome, NY USA 13441;
[c] Air Force Office of Scientific Research (AFOSR), Arlington, VA USA 22203;
[d] Space Vehicles Directorate, Air Force Research Laboratory (AFRL), Kirtland Air Force Base, NM USA

ABSTRACT

Space situational awareness (SSA) is needed to control satellite movement and space pervasiveness, which relies on quick and precise space object behavioral classification and discovery. Perhaps the biggest obstacle in adopting machine learning (ML) techniques and evaluating their performance in SSA applications is the lack of large, labeled datasets for training and validation. In this paper, we present a game theory enabled, data augmentation method, to produce datasets used by ML techniques for satellite behavior detection. Realistic sensor data (azimuth angle, elevation angle, range, range rate) are propagated using SGP4/SDP4. Then various maneuver strategies are produced by our space game model, played by on-ground radar and space objects. We use the two-player pursuit-evasion game to generate evasive maneuvering strategies for space objects. The game approach provides a method to solve SSA behavior detection problems, where the Resident Space Objects (RSO) exploits the sensing and tracking model to confuse the SSA observer by corrupting their tracking estimates, while the SSA observer improves tracking performance. The different cost functions generate different maneuver strategies. In addition, to simulating this rich training data we exploited generative adversarial networks (GANs) to further augment the simulated data. With simulated data plus GAN augmentation, various satellite behaviors are generated to produce synthetic datasets with labels for use by ML methods. Then a 3D convolutional neural network (3D-CNN) is provided the generated synthetic and labeled data with 143 possible satellite behaviors. The trained machine learning model efficiently and correctly classifies the satellite behaviors with a 97% rate. The model-based, game theoretic synthetic data, improves the training and validation performance of machine learning algorithms for satellite behavior classifications.

Keywords: Space situational awareness, GANs, ResNet, satellite characterization, sensor models, machine learning, general-sum games, 3D-CNN.

1. INTRODUCTION

With recent world-wide developments, society has increasingly relied on space assets to provide sensing and communications benefits in various industrial, civilian, and commercial applications. Since information from space is critical for various decisions, space science and technology is considered a new frontier. In addition to real-time and hidden information constrains, space object density significantly increases the complexity of space situational awareness (SSA). For accurate SSA, there are multiple challenges such as (*i*) partially observable movements, (*ii*) resident space objects (RSOs), (*iii*) uncertainties modeling and propagation, (*iv*) real-time respondence, and (*v*) computationally intractable algorithms.

Investigation into space access and mission trade-offs is critical for space-borne operations. This requires algorithms capable of detecting and tracking space objects, debris, and natural phenomena (such as comets, asteroids, and solar flares). Understanding the position of space objects from low-level information fusion can support high-level information fusion to support SSA missions for sensor, user, and task refinement [1]. In order to implement SSA accurately, it is possible to coordinate the evaluation of residential space objects (RSO) through user-defined operation

*gchen@intfusiontech.com; phone 1 301515-7261; fax 1 301 515-7262

pictures (UDOP) [2]. Improvements to tracking and sensor SSA include models (such as track mechanics), measurement, calculation software, and application-based system coordination (such as situations). For example, the *game theory* method can be used in SSA for pursuit and escape analysis [3].

There are several recent papers that have applied deep learning methods for space situation awareness in the area of space object detection, classification, and identification [4][5]. The National Imagery Interpretability Rating Scale (NIIRS) and video NIIRS (V-NIIRS) has been utilized to determine the dynamic behavior analysis of objects [6][7], where the V-NIIR is modified from the traditional NIIRS [8]. However, due to the data link bottleneck restrictions, large images collected and transmitted from the air or space domain to the ground domain need compression, which alters the NIIRS ratings [9-12]. Therefore, a brokering system was developed to determine the image collection parameters based on the NIIRS (or V-NIIRS) requirements [13][14]. It is similar to Light Detection and Ranging (LIDAR [15]) and thermal infrared imagery [16]. Other works, such as works by Lucas, Kyono et al [17][18], utilize deep learning neural networks to analysis Space-Object NIIRS (S-NIIRS), where a convolutional neural network was utilized to improve the classification resolution (smaller ground resolved distance due to enhanced imaging characteristics for space telescopes) of imagery from SNIIRS.

Deep learning, convolutional neural networks, as well as other machine learning tools have greatly advanced the development of the space image processing, object tracking, and information fusion technologies [19][20]. In addition, some advanced deep learning networks, such as generative adversarial networks (GAN) can provide capability for generating fake data for the same distribution of the input samples. Therefore, SSA can be also advanced by using GAN models to create data for a wide range of real-world circumstances such as jamming [21], collision avoidance [22], and coordinated collection [23] for robustness analysis. The enhanced classification neural networks [24][25] will concurrently improve the performance of SSA.

Perhaps the biggest obstacle in adopting machine learning (ML) techniques and evaluating their analytics in SSA applications is the lack of large, labeled datasets for training and validation. In the literature, there are several techniques designed to deal with small training datasets including boosting, transfer learning, and simulation. Transfer learning enables users to construct a network for a small dataset without overfitting, however, the feature detectors need to be trained on a large dataset. An autoencoder is a deep neural network (DNN) whose output and input are the same, therefore, any dataset is labeled dataset, but the events detected by trained autoencoders are limited. Usually, other algorithms are used to classify the results of autoencoders. A training set of a small size can also be made to appear larger through data augmentation. One of the challenges to use data augmentation is that in the real-world scenario, the data is captured over a limited set of conditions while machine learning algorithms require training data over as many possible operation conditions.

In this paper, we present a game theory enabled, data augmentation method, to produce datasets used by ML techniques for satellite behavior detection. Realistic sensor data (azimuth angle, elevation angle, range, range rate) are propagated using SGP4/SDP4 and our space game model produces various maneuver strategies, which is played by on-ground radar and space objectives. We use the two-player PE game to investigate the sensor management for tacking evasive space objects. The game approach provides a method to solve SSA behavior detection problems, where the Resident Space Objects (RSO) will exploit the sensing and tracking model to confuse the SA observer by corrupting their tracking estimates, while SA observer wants to improve the tracking performance. The different cost functions generate different maneuver strategies. In addition, to rich the training data, we exploited generative adversarial networks (GANs) to furtherly augment the simulated data. GAN models are used to generate more data from the simulated data to improve the robustness of our trained model as well as increase the inference accuracy of the trained networks for space behavior detection. Using the simulated data plus GANs, various satellite behaviors are simulated and augmented to generate synthetic datasets with labels for use by ML methods.

To demonstrate the proposed data generation method, we process catalog data (space-track.org) and obtain the tracks without maneuvers. Then, we simulate various satellite behaviors guided by game-theoretic maneuver strategies. The catalog data are modified based on the propagation results from SPG4/SDP4 with maneuver strategies and labeled based on the behavior simulated. To evaluate the performance, a 3D convolutional neural networks (3D-CNN) is designed and implemented for satellite behavior classification. The 3D-CNN is provided the generated synthetic and labeled data with 143 possible satellite behaviors (15 degrees for two-directions maneuvers). We generated over 72,000 tracks, 1/10 of the data is randomly selected as test set for validating the 3D-CNN performance. The convolutional filters are initially chosen arbitrarily, so the classification is done randomly. It is noted that as iteration increases, the training model converges nearly to 100% accuracy.

The remaining dataset is used for evaluation for the well-trained CNN model. The trained machine learning model can efficiently and correctly classify the satellite behaviors with a 97% rate. Meanwhile, the ResNet improved GAN model architecture can accurately perform the data augmentation, which further augment the simulated data. Overall, our model-based, game theoretic, synthetic data improves the training and validation performance of machine learning algorithms.

This paper is organized as the following. A pursuit-evasion game model is detailed in Section 2 to simulate various satellite behaviors. Section 3 introduces the architecture of the 3D-CNN. Section 4 presents architecture of Residual neural networks and its utilization in the ResNet GAN model for data augmentation for robust neural networks training. Then, the numeric results and analysis for both 3D-CNN as well as ResNet GAN model are displayed in Section 5. Finally, Section 6 summarize the whole paper with conclusion.

2. PURSUIT-EVASION GAME MODEL

2.1 System States and Dynamics for Space Objects

To perform maneuvers, the evader (space object being tracked) will apply a continuous low-thrust such as the Ion thrust. Ion thrusters tend to produce low thrust, which results in low acceleration. For example, a NASA Solar Technology Application Readiness (NSTAR) thruster producing a thrust (force) of 92 mN will accelerate a satellite with a mass of 1,000 kg by 0.092 N / 1,000 kg = 0.000092 m/s^2. The magnitude of the thrust is assumed to be fixed and small. The controls of the thrust commands are the directional angles of these thrusts.

The following equations describe the kinematics and dynamics of the spacecraft's states with continuous low-thrust:

$$\dot{r} = v \sin\gamma \tag{1}$$

$$\dot{v} = \frac{T}{m}\cos\alpha\cos\beta - \frac{\mu\sin\gamma}{r^2} \tag{2}$$

$$\dot{\gamma} = \frac{v\cos\gamma}{r} + \frac{T}{m}\frac{\sin\alpha\cos\beta}{v} - \frac{\mu\sin\gamma}{r^2 v} \tag{3}$$

$$\dot{\xi} = \frac{v\cos\gamma\cos\zeta}{r\cos\phi} \tag{4}$$

$$\dot{\phi} = \frac{v\cos\gamma\sin\zeta}{r} \tag{5}$$

$$\dot{\zeta} = \frac{T}{m}\frac{\sin\beta}{v\cos\gamma} - \frac{v\cos\gamma\sin\phi\cos\zeta}{r\cos\phi} \tag{6}$$

There are six system states: r (norm of position vector), v (norm of velocity), γ (angle between velocity vector and the local horizon plane), ζ(angle between local east and the projection of velocity vector in local horizon plane), ξ (longitude) and ϕ (latitude). α, β are the directional angles of the thrust T.

Given a local direction (α, β) and a Δt, the system states are propagated by following the steps:

1. Convert the states to the local coordinate system
2. Using numerical integration method to compute the local states at $t_0 + \Delta t$
3. Convert the new local states to ECI

The conversion between Earth-centered inertial (ECI) and local coordinate system (East-North-Up, or ENU) is detailed in the following equations, where (dx, dy, dz) is a vector in ECI and (de, dn, du) is a vector in ENU coordinate system.

$$\begin{bmatrix} de \\ dn \\ du \end{bmatrix} = \begin{bmatrix} -\sin\xi & \cos\xi & 0 \\ -\sin\phi\cos\xi & -\sin\phi\sin\xi & \cos\phi \\ \cos\phi\cos\xi & \cos\phi\sin\xi & \sin\phi \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} -\sin\xi & -\sin\phi\cos\xi & \cos\phi\cos\xi \\ \cos\xi & -\sin\phi\sin\xi & \cos\phi\sin\xi \\ 0 & \cos\phi & \sin\phi \end{bmatrix} \begin{bmatrix} de \\ dn \\ du \end{bmatrix} \tag{8}$$

2.2 Pursuit-Evasion Game Setup

In our dynamic PE game model, each player P (purser) or E (evader) has its own system states and state transitions:

$$x_p = [r_p, v_p, \gamma_p, \xi_p, \phi_p, \zeta_p]^\top = [x_1, x_2, x_3, x_4, x_5, x_6]^\top \tag{9}$$

$$x_e = [r_e, v_e, \gamma_e, \xi_e, \phi_e, \zeta_e]^\top = [x_7, x_8, x_9, x_{10}, x_{11}, x_{12}]^\top \tag{10}$$

$$u_p \in \{0, 1\} \tag{11}$$

$$u_e = [\alpha_e, \beta_e]^\top \tag{12}$$

$$\dot{x}_p = g_p(x_p) \tag{13}$$

$$\dot{x}_e = g_e(x_e) \tag{14}$$

Eq. (9-10) and Eq. (13-14) are the augmented versions of Eq. (1-6) for pursuer and evader. Since the purser (a specific ground station) will not maneuver, $T_p = \alpha_p = \beta_p = 0$. The u_p is an on-off control. When $u_p = 1$, the pursuer will spend the sensor resource to get measurements of the evader. When $u_p = 0$, the pursuer will not observe the evader. The controls of the pursuer will not affect the position and velocity states, but it will change the tracking performance in terms of entropy of the covariance, *P,* matrices in the Extended Kalman Filter (EKF) or Cubature Kalman Filter (CKF). With the covariance information, the entropy is:

$$h(x) = -\int \dots \int_{-\infty}^{\infty} f(x) \ln f(x) ds_1 \dots ds_6 \; = \frac{1}{2} ln((2\pi e)^6 det\,(P)) \; \propto \; \lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6 \tag{15}$$

where $f(x)$ is the probability density function (pdf) of satellite states x (positions and velocities), $s_1, s_2, \dots s_6$ are the components of x, λ_1, $\lambda_2 \dots \lambda_6$ which are the eigenvalues of P matrix. Intuitively, the entropy is proportional to the product of eigenvalues of P matrix. In our PE game setup, the evader will try to increase the entropy while evader wants it to be minimal. The entropy becomes the confliction parameter between the evader and the pursuer.

The cost function of evader is defined in eq. 16, where $\bar{x}_e$ is the filtered state of evader and P is the covariance matrix of a tracker (EKF or CKF).

$$\begin{aligned} J_e = h(\bar{x}_e) &= -\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} f(\bar{x}_e) \ln f(\bar{x}_e) \mathrm{d}\bar{x}_7 \dots \mathrm{d}\bar{x}_{12} \\ &= \frac{1}{2} \ln\left((2\pi e)^6 \det(P)\right) \propto \prod_{i=6} \lambda_i(P) \end{aligned} \tag{16}$$

The purser's cost function is defined in eq. 17, where $P_{\bar{m}}$ and P_m is the error covariance matrices of a tracker without measures, and with measures, respectively. $c > 0$ is a design parameter of fixed reward for saving sensor resources.

$$J_p = \begin{cases} \prod_{i=6} \lambda_i(P_{\bar{m}}) - \prod_{i=6} \lambda_i(P_m) & \text{if } u_p = 1; \\ c & \text{if } u_p = 0. \end{cases} \tag{17}$$

Then the optimal controls of two game players are $u_e^* = \arg\max_{u_e} J_e$ and $u_p^* = \arg\min_{u_p} J_p$. We can solve the game problem using a numerical method [26] as well as the Fictious Play concept [27][28].

3. 3D CONCOLUTIONAL NEURAL NETWORKS

In recent years, we have seen numerous advances in various classification tasks based on improved neural network architectures, algorithms and optimization techniques collectively known as deep learning (DL). DL has recently replaced state-of-the-art machine learning (including decision tree, support vector machine (SVM) and artificial neural

networks (ANN)) in multiple fields such as computer vision, voice, and natural language processing. DL greatly increased the capacity for feature learning directly on raw, high dimensional, input data based on high level supervised objectives, due to the newly found capacity for learning of exceptionally large neural network models with high numbers of free parameters. These achievements are possible due to the integration of the techniques including the advanced stochastic gradient descent, low-cost high-performance graphics card processing power and combining of key neural network architecture innovations such as 3D-CNNs, and rectified linear units. Based on the powerful capacities for learning and classification of DL, we developed a DL based learning module to perform the signal classification and characterizing tasks.

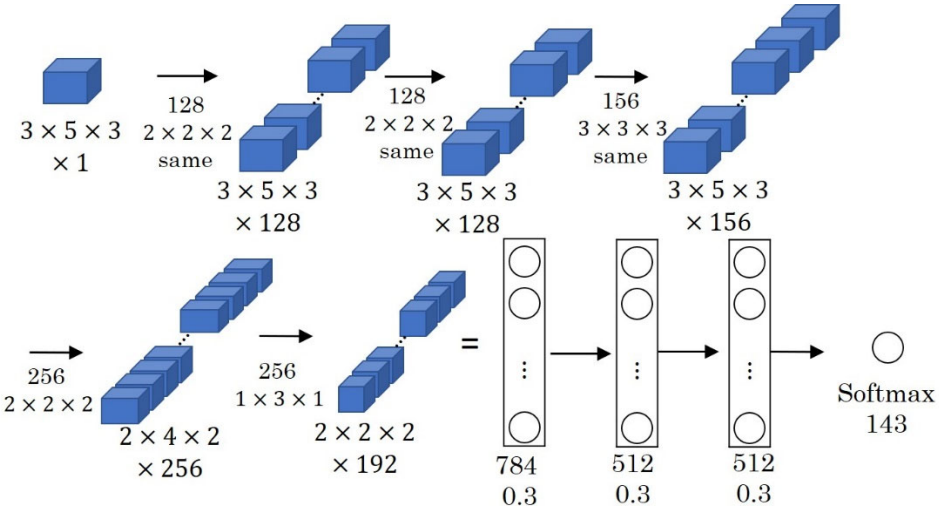

Fig. 1. The Scheme of our 3D convolutional neural networks (3D-CNN)

3D-CNNs are a class of deep neural networks, most applied to analyzing 3D visual imagery. Originally a 2D convolutional layer is an entry per entry multiplication between input and the different filters, where filters and inputs are 2D matrices. In 3D CNN layers, the same operation of the 2D convolutional layer is utilized with operations on multiple pairs of 2D matrices.

The constructed 3D-CNN structure is shown in Fig. 1. The raw data includes 4 dimensions, which are 1 dimension for location (azimuth angle, elevation angle, range), 2 dimensions for time interval, and last dimension for the channel of filters, respectively. Therefore, the raw data is reshaped to $3 \times 5 \times 3 \times 1$ dimensions with 72,000 samples as our raw dataset. In order to distinguish the 143 different satellite behaviors (15 degrees for azimuth angle and elevation angle maneuvers, respectively), we employed the following 3D-CNN architecture for classification of different behaviors.

As shown in Fig. 1, the first layer is the 3D convolutional layer with 128 $2 \times 2 \times 2$ filters and same padding afterwards. Since the GAN model will be employed in the future using the similar architecture, the max-pooling layer is excluded in our classification model, which leads to instability for the GAN model convergence. Due to the same padding's settings, the dimension of the data after first convolutional layers does not vary. Then the previous output bypass another two 3D convolutional layers with 128 $2 \times 2 \times 2$ filters and 156 $2 \times 2 \times 2$ filters together with the same paddings, respectively. Afterwards, another two 3D convolutional layers without the same padding methods were added to down-sample our data dimensions from $3 \times 5 \times 3 \times 156$ to $2 \times 2 \times 2 \times 192$ for future operations. These two layers consist of 256 $2 \times 2 \times 2$ and 192 $1 \times 3 \times 1$ filters, respectively. Finally, the $2 \times 2 \times 2 \times 192$ dimensions data was flattened to be 1 dimension using Flatten layer between the 3D convolutional layers and the following Dense layers. The flattened data is bypassed three Dense layers with 784, 512, and 512 hidden neurons, respectively. In addition, all these layers utilized 30% dropout parameters to mitigate the overfitting problem during the training process. Finally, a 143-degree SoftMax layer was attached at the last of our 3D CNN model.

For the training parameters, the weight initializers for both 3D convolutional layers as well as the Dense layers are Glorot Uniform initialization. The sparse cross entropy is selected as our loss function as shown in equation (18)

$$L_{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{18}$$

where N indicates the batch size during the training process and y_i is the true label, $\hat{y}_i$ is the predicted label, respectively. In addition, the Stochastic Gradient Descent (SGD) optimizer was employed to calculate the gradient to update the weights to minimize the loss. The utilization of SGD optimizer instead of Adam optimizer is that SGD is less prone to overfitting problems compared to the Adam optimizer, even though the convergence speed is much lower for SGD.

The details of the parameters of our deep neural networks is shown in Table 1. There are 3,081,979 parameters need to be trained for accurately detecting the different satellite behaviors. During the training process, the learning rate is decayed using the learning rate as the equation (19). For the first 300 epochs of training, the learning rate was 0.001, which can update the weights much faster. After 300 epochs, the exponentially decay of learning rate was utilized in order to find the optimized point to minimize the loss of neural networks.

$$lr = 0.001 \times e^{0.0025\times(300-epoch)} \tag{19}$$

In addition, the momentum is set to be 0.8 for the SGD optimizer to accelerate gradient descent in the relevant direction and dampen oscillations, where the value of 0.8 is optimized with trial and error with several training process to confirm its value. With higher momentum value, the training process is faster with huge overfitting problems, with lower momentum value, the training convergence process is much slower with lower accuracy after 2000 epochs. After 2000 epochs of training, the results will be shown in Section 5 in this paper.

Table 1 The parameters of 143 classification 3D-CNN model

Layer(type)	Output Shape	Param #
Conv3d	(None, 3, 15, 128)	1152
Conv3d	(None, 3, 15, 128)	131200
Conv3d	(None, 3, 15, 256)	539292
Conv3d	(None, 3, 9, 512)	319744
Conv3d	(None, 3, 3, 256)	147648
Flatten	(None, 1536)	0
Dense	(None, 784)	1205008
Dropout	(None, 784)	0
Dense	(None, 512)	401920
Dropout	(None, 512)	0
Dense	(None, 512)	262656
Dropout	(None, 512)	0
Dense	(None, 143)	73359

4. RESNET GAN ARCHITECTURE

This section describes methods to exploit Residual neural networks as well as general-sum games [29-37] to improve the performance of GANs. The purpose of deep learning is to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications. To enhance the performance of the model in Section 3, the ResNet conditional GAN (ResGAN) is utilized in the RSO behavior data augmentation. In other words, a well-trained ResGAN model can generate more data via the input data, improving the robustness of the trained model and increasing the test accuracy of input data.

4.1 Introduction of Residual Neural Networks

According to the universal approximation theorem, give enough capacity, we know that a feedforward network with a single layer is enough to represent any function. However, the layer might be massive, and the network is prone to overfitting the data. Therefore, there is a common trend in the research community that our network architecture needs to go deeper. Deep convolutional neural networks have led to series of breakthroughs for image classification. Deep networks naturally integrate low/mid/high-level features and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Since AlexNet, the state-of-art CNN

architecture is going deeper and deeper. While AlexNet had only 5 convolutional layers, the VGG network and GoogleNet had 19 and 22 layers, respectively.

However, with significance of depth, a notorious obstacle occurs named as vanishing/exploding gradients, which hampers convergence of the deep neural networks from the beginning. This problem, however, can be largely addressed by normalized initialization and intermediate normalization, which enables networks converge by utilizing backpropagation. Unexpectedly, when deeper networks are able to start converging, a degradation problem has been exposed, where accuracy gets saturated and then degrades rapidly. Such degradation is not caused by overfitting, thus, adding more layers leads to higher training error.

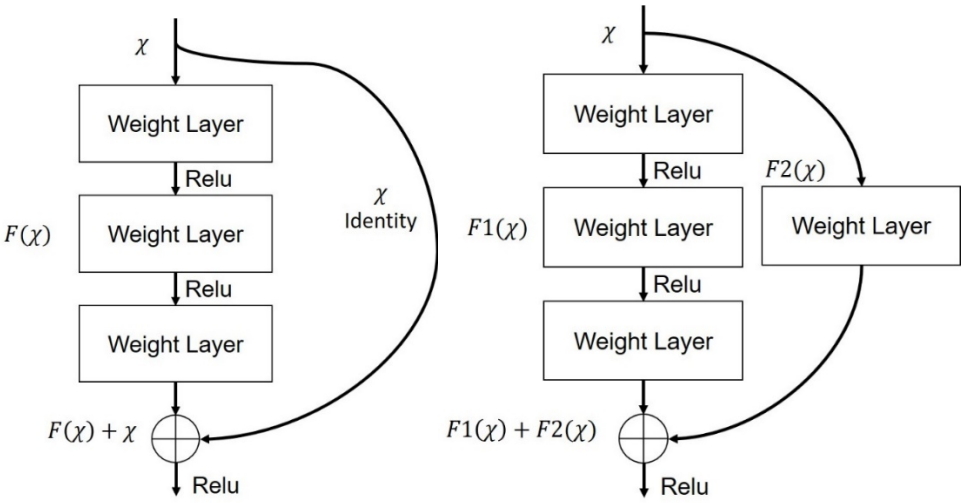

Fig. 2. Residual learning: a building block (left), block with weight layer mapping (right)

In order to solve this problem, a Residual neutral network is introduced with the core idea called "identify shortcut connection", which skips one or more layers and addresses the downgrade and vanishing problem. The idea of concept is shown in left of Fig. ***2***. Formally, the desired underlying mapping is denoted as $H(\chi) \coloneqq F(\chi) + \chi$. We hypothesize that it is much easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than fit an identity mapping by a stack of nonlinear layers. The formulation of $F(\chi) + \chi$ can be realized by feedforward neural networks with "shortcut connections". Shortcut connections are those skipping one or more layers. For some cases, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Identity short connections add neither extra parameter nor computational complexity. The entire network can still be trained end-to-end by SGD with backpropagation and can be easily implemented using common libraries without modifying the solvers.

On the other hand, the shortcut connection can be performed with weighted layers mapping as well, as shown in right of Fig. ***2***. Instead of identify mapping, one weighted layer was utilized and employed. As a matter of fact, ResNet was not the first to make use of shortcut connections, highway network introduced gated shortcut connections. These parameterized gates control how much information is allowed to flow across the shortcut. Therefore, ResNet can be thought of as a special case of highway network. However, experiments show that the highway network performs no better than ResNet, which is unexpected since the solution space of the highway network contains ResNet, therefore, it should perform at least as good as ResNet. This suggests that it is more important to keep these "gradient highways" clear rather than to go for large solution space. Following this intuition, the authors of refined the residual block and proposed a pre-activation variant of residual block, in which the gradients can flow through the shortcut connections to any other earlier layer unimpededly.

4.2 ResNet GAN models

Supervised learning often develops a model to predict a class label given an example of input variables, which is called classification, as one type of the *discriminative model*. Different than the traditional supervised learning model, there is another paradigm of learning where the model is only given the input variables (X) and the problem does not have

any output variables (Y). Alternately, this model, that summarizes the distribution of input variables, may be utilized to create or generate new examples in the input distribution, referred to *generative models.*

Generative Adversarial Networks (GANs) are a deep-learning-based generative model, which provides an architecture for training a generative model. As first introduced by Ian Goodfellow in 2014 [38], the GAN model has been developed dramatically in recent years. With two sub-models: a *generator model* for generating new examples and a *discriminator model* for classification included as the basis (as shown in Fig. 3), the GAN model has been modified to several additional GANs, such as conditional GAN (cGAN) [39], InfoGAN [40], AC-GAN [41], and semi-supervised GAN (SGAN) [42]. The details of the traditional GAN architecture have been introduced in our previous work [25].

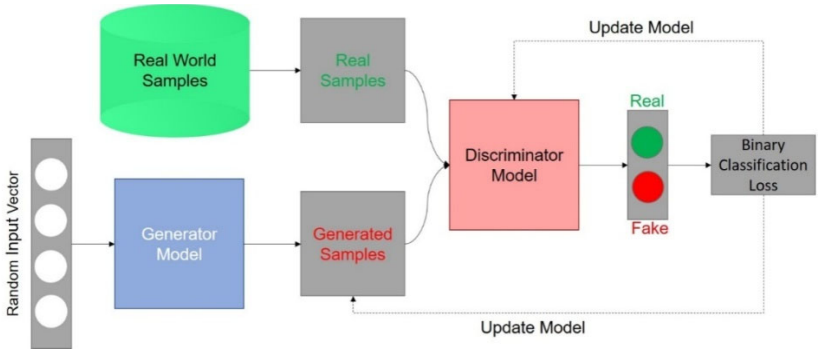

Fig. 3. Scheme of the Generative Adversarial Network Model Architecture.

In this paper, to make full usage of the class label information of different maneuvers, which is used to train our GAN model to generate different satellite behaviors as well as prohibit the mode collapse for the generated data, the *Conditional Generative Adversarial Network* (cGAN) was utilized as the base GAN architecture for our deep neural networks. The approach utilizes one hot encoded class label concatenated with the input to both the generator and discriminator models. As shown in Fig. 4, the scheme of conditional generator and conditional discriminator in a cGAN is displayed.

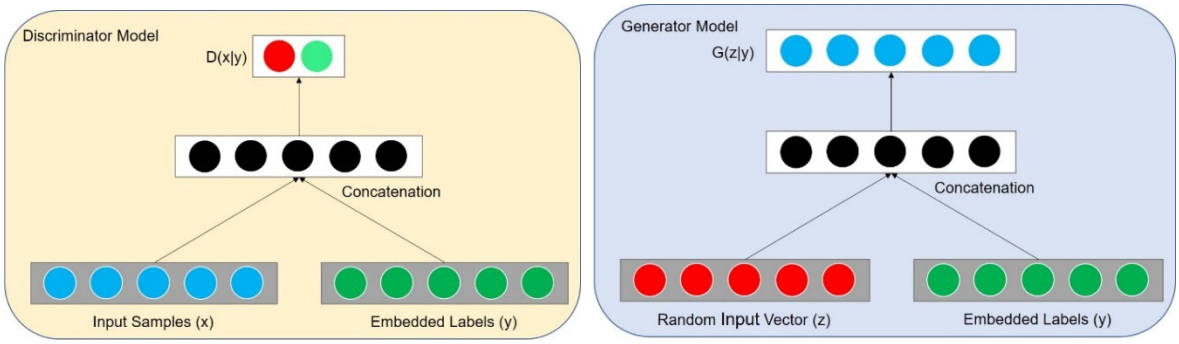

Fig. 4. Scheme of a Conditional Generator and a Conditional Discriminator in a Conditional Generative Adversarial Network.

However, based on our previous experiences, the generator in cGAN architecture needs to be much deeper with more parameters compared to the discriminator. Otherwise, the training process will be unstable and di-converge. Therefore, for 143 labels for classification, at least 3 million discriminator's parameters are needed to perform well in the deep learning model. Then, the required generator's parameters are at least 100 million with deeper networks for stable GAN training process. However, due to fact that the deeper networks have lots of paths to learn the detailed features, the original feature representations are easy to diffuse during pass through layers. The generated results thus go into under-fitting or local maximization during the GAN process, which leads to the failure of stabilization of GAN model as well as failure to generate useful fake samples to confuse the discriminator.

To overcome this issue, some researchers claimed that the ResNet proposed previously allow the features go deep in the neural networks [43], and to our knowledge the ResNet have not been found to be implemented in generative networks, such as conditional GAN models. Thus, the classified labels and latent noise are not only the input for

generator, but also pass through the channel of ResNet to push coarse noise to pass through channel to maintain the original spatial features. Based on this feature, the researcher can formulate a uniform and efficient space tracking generation networks.

In this paper, the ResNet generator in the GAN model is proposed for the task of space track restoration, which is named ResGAN based on the deep convolutional neural networks. After testing several datasets, such as MNIST, CIFAR10/100, CelebA, the results verified the lower losses and higher accuracies compared to the other state-of-the-art GANs. Therefore, the ResNet structure is more stable and efficient for generative networks.

4.3 Details of ResGAN model

Based on the previous discussion, the generator model in the GAN will be the ResNet structure due to the benefits of skipping layers to get more gradient descent backpropagation. The scheme concept of our generator is shown as the following Fig. ***5*** left. The input of generator contains two parts, which are latent noise as well as the input label for fake image generation. The latent noise contains 500 dimensions with Gaussian noise that has a 0 mean and standard deviation of 1. The input label contains 143 different numbers for generating different satellite tracks. Both the two parts bypasses embedding layers or dense layers to increase its complexity before concatenating together to a $3 \times 3 \times 784$ tensor. Afterwards, this tensor will pass several Conv2Dtranspose layers. Since identity mapping shortcuts would not change the tensor dimension, however, some of layers have different tensors to up-sample our 3×3-dimension into 3×15-dimension tensor. Therefore, instead of using identity mapping by shortcuts, we employed one Conv2Dtranspose layers.

In addition, we insert some shortcut connections as shown in Fig. ***5*** left, which turn the network into its counterpart residual version. The shortcut paths can be used when the input and output are of the same dimensions, when the dimensions increase, we added an additional up sampled layer instead. It is continued until the tensor reaches the dimension of $3 \times 15 \times 1$ for the generated satellite tracking. In the main convolutional layer part in the GAN model, the convolutional layers mostly have 2×2 filters and follow another two simple design rules: (i) the first four layers only have $1 \times x$ dimensions to upsample our tensor from 3×3 dimension to 3×15 tensor dimensions, and (ii) the filter sizes first increase and then decreases two times except for the last layers. Since for the GAN model produces the best results without the Maxpooling layer or Stride in the generator or discriminator, we perform the upsampling as well downsampling directly via filter size without "same" padding settings.

The scheme of our discriminator is shown in Fig. ***5*** right, the discriminator is based on a modified classification model for 143 labels. The same as the conditional GAN, the input has two parts: the generated sample, or original sample, and the input label. The two parts will go through several embedding and dense layers and then concatenate with the original data. Since the leakyRelu activation layer is very useful for GAN model, all the previous "relu" activation layer were changed to a "LeakyRelu" with a 0.2 slope. Then, the connection of our generator output to the input of our discriminator is built on the architecture of our ResGAN model. In order to predict a fake or real sample, there are only one set of outputs, a 1-degree sigmoid output for binary classification. The output comes from the last dense layer with 512 hidden neurons, and some dropout layers were added, which is similar with the previous 3D-CNN model.

The algorithm of our ResGAN model approach is shown as following:

1. Given real orbital data as input, D outputs one distributions D. D is optimized by minimizing a binary cross entropy loss L_D. In the case of real inputs, the gradients are generated using the following loss functions:
$$L_D = \mathbf{E}_{r\sim\mathcal{R}} \max_D \log\big(D(r)\big) + \log\left(1 - D(G(r_{fake}))\right) \tag{20}$$
2. Using the gradients from D, G is updated using the adversarial loss to produce realistic class consistent real orbital data.
$$L_G = \min_G \mathbf{E}_{z\sim \mathrm{Z}} \log\left(1 - D(G(r_{fake}))\right) \tag{21}$$

The proposed iterative optimization procedure is summarized as a pseudocode in Algorithm 1.

Algorithm 1: Iterative Training Procedure of ResNet GAN

Minibatch stochastic gradient descent training of generative adversarial nets. The batch size to apply to the discriminator, k, is a hyperparameter.

1: training iterations = N

2: for t in 1:N do

3: Sample k (batch_size) samples with labels from real dataset $\mathcal{R}$: $\{r_i, y_i\}_{i=1}^{k}$

4: Sample k random latent noise samples $\{z_i\}_{i=1}^{k} \sim \mathcal{N}(0,1)$

5: Let z_i and assigned labels $l_j (j \in (0,1,2, \dots ,10))$ be the concatenated inputs to generator

6: Let $f_i = G(z_i, l_j)$ be the generated datasets for fake data

7: Update the discriminator using the following objectives

$$L_D = L_{real} + L_{fake} \tag{22}$$

$$L_{real} = max_D \frac{1}{k} \sum_{i=1}^{k} \log\big(D(r_i, y_i)\big) \tag{23}$$

$$L_{fake} = max_D \frac{1}{k} \sum_{i=1}^{k} \log\Big(1 - D\big(G\big(z_i, l_j\big), l_i\big)\Big) \tag{24}$$

8: Update the generator, only for the fake data, through the discriminator gradients

$$L_G = min_D \frac{1}{k} \sum_{i=1}^{k} \log\Big(1 - D_{real}\big(G\big(z_i, l_j\big), l_i\big)\Big) \tag{25}$$

9: **end for**

The gradient-based updates can use any standard gradient-based learning rule.

The details of parameters for the ResNet Keras GAN model are shown in Table 2. The plain discriminator has around 5,325,406 both trainable and untrainable parameters. On the other hand, the ResNet GAN model has 109,648,974 parameters in generator, almost 20 times the plain discriminator. Uusing this type of the generator, the GAN training process is stable. Otherwise, from our experiences via trial and error, the training process of the GAN model will diverge.

Table 2 The parameters of ResNet GAN models

Name	ResNet Generator	Plain Discriminator
Generator Parameters	Trainable Parameters	Non-Trainable Parameters
109,648,974	109,648,974	0
Discriminator Parameters	Trainable Parameters	Non-Trainable Parameters
10,650,812	5,325,406	5,325,406
GAN Parameters	Trainable Parameters	Non-Trainable Parameters
114,974,380	109,648,974	5,325,406

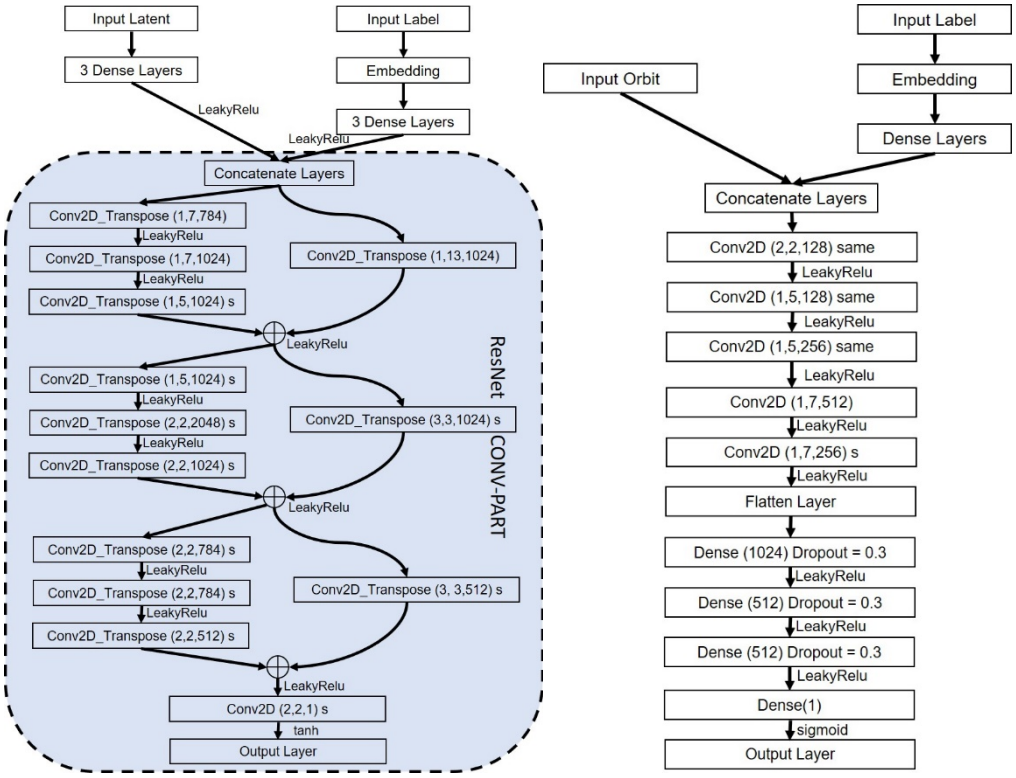

Fig. 5. The residual generator scheme (left) and the discriminator scheme (right) in our ResNet GAN model.

5. NUMERICAL RESULTS

5.1 Training Data Generation

In order to generate the training data, the tracks based on the TLEs from space-track.org were modified by adding behavioral maneuvers from Section 2. The different maneuvers indicate different labels in our training data. As an example, the maneuver to increase the orbital energy in azimuth angle by 15 degrees is labeled as 1. On the other hand, an increase in the orbital energy in both azimuth and elevation angles for 15 degree is labeled as 13. The details for maneuver addition are depicted as follows:

a. The ECI are computed from the two-line element (TLE) at 0 time-step;
b. Use the Game methods to propagate the satellites tracks;
c. Convert the 16 waypoints back to azimuth angle, elevation angle, range, range rate relative to a ground site.

Fig. **6** displays the first 10,000 tracks that are from the 72,000 generated tracks for training purposes and another 6,700 tracks for testing.

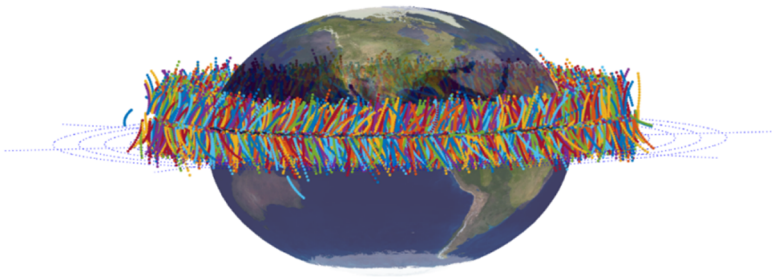

Fig. 6: The First 10,000 Training Tracks with Various Space Behaviors

The data format is listed as (Each row is an observation):

- Column 1: track id
- Column 2: observation id (from 1 to 15)
- Column 3: Azimuth angle (rad)
- Column 4: Elevation angle (rad)
- Column 5: Range (km)
- Column 6: Training label (from 1 to *m*, where *m* is total types of space behaviors)

5.2 3D Convolutional Neural Networks Results and Discussions

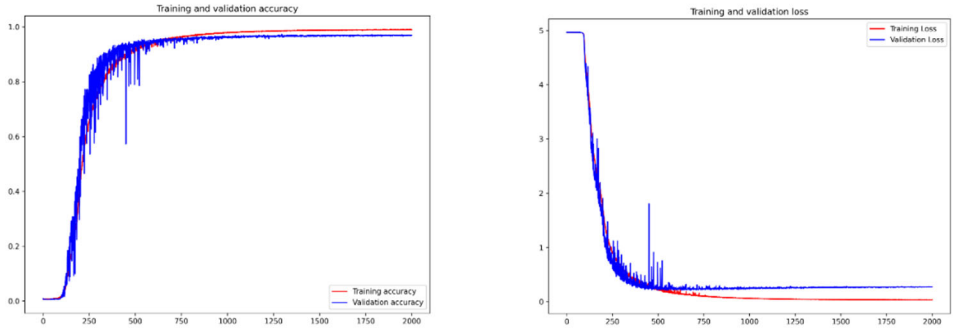

Fig. 7: Training and Validation accuracy (left) and loss (right) for 3D-CNN model

Fig. **7** represents the training process for 2000 epochs of the 3D-CNN model mentioned in Section 2. After 2000 epochs optimization using SGD, we obtained almost 99.1% training accuracy and 97.1% test accuracy, as shown in the left of Fig. **7**. Different than Adam optimizer training, the initial learning process is much slower for the first 150 epochs, shown by the plateau at the beginning of the accuracy figure. Meanwhile, due to the dropout layer and SGD optimizer, there is no overfitting before 750 epochs, even the validation accuracy is higher than the training accuracy. However, after 750 epochs of training, the accuracy of the test dataset stopped increasing with the beginning of overfitting for training datasets. Additionally, the cross-entropy loss during the training process is displayed in the right side of Fig. **7**, which has a trend similar to the training accuracy. The training set loss always decreases, while the loss of the validation set is plateaus after 750 epochs of training, indicating overfitting as well. On the other hand, there are multiple places with a large fluctuation phenomenon for both the validation accuracy and loss, indicating a larger learning rate than the requirements. The learning rate scheduler needs to be adjusted further for better performance of the 3D-CNN classification models. In addition, Fig. **8** shows the confusion matrix of our model's performance. As we can see, for both training and test datasets, the true labels are almost the same as our model's predicted labels. Therefore, our 3D-CNN model is enough accuracy to classify 143 different satellite behaviors.

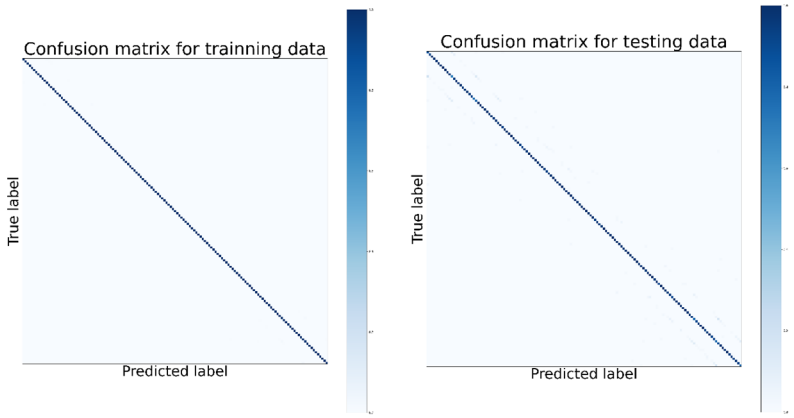

Fig. 8. Confusion Matrix of training data and testing data

As shown in Fig. ***9***, to intuitively represent these relationships between satellite behavior labels and anomaly filters, we introduce a grid-style visualization, where rows and columns represent layers and classes of behaviors, respectively. The number of rows and columns equal to the number of layers with anomaly filters and classes with anomaly iterations, respectively. In Fig. ***9***, the darker the color, the more anomaly weights and filters appear in that layer, related to that class. From this visualization, we can easily observe that the second fully connected dense layer has the most anomaly weights and filters among all the other layers. As we can see, there is a trend for most anomaly weights and filters during training processes, especially the middle layers of our deep convolutional neural networks. In addition, the anomaly class appear to have the same interval in our dataset, indicating a better network is needed.

Fig. 9. The abstract version of correlation view, where rows and columns represent layers and image classes, respectively. A sequential color scheme is used to encode the number of anomaly filters.

5.3 ResNet GAN Results and Discussions

The generated data provides inputs to the previously trained 3D-CNN classification model to determine the accuracy of the data generated by the the ResNet model. The accuracy of our ResNet generator is shown in Fig. 10. With more than 8000 epochs of training for our both the generator and discriminator model, the ResGAN system achieved around 92% accuracy. This demonstrates the performance of our trained generator, even for a large amount of different satellite behaviors.

In addition, the generated data confusion matrix is shown in Fig. 10 right. Most of the labels have an accuracy greater than 90% for every label. However, there are some small island around the diagonal, indicating some similarity for that produced data. Both visualizations of the results show a tremendous performance for our ResGAN model leading to stabilized training performance for our modified conditional GAN model.

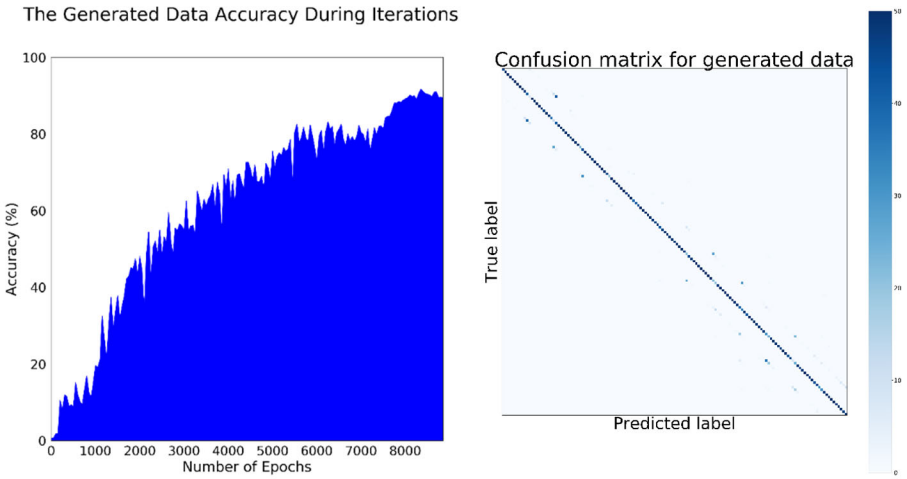

Fig. 10. The Results of our trained GAN model.

There are other ways to evaluate the performance of the ResNet GAN generator model. Many GAN practitioners rely on the evaluation of GAN generators via the manual assessment of data generated by a generator model. This involves using the generator model to create a batch of synthetic data, then evaluating the quality and diversity of the samples in relation to the target domain. However, qualitative measures are not numerical and subjective to human evaluator. Therefore, quantitative GAN generator evaluation is more robust and reasonable to use. There are several quantitative measures to evaluate the ResNet GAN model, such as converge metric, inception score (IS), Mode Score, AM Score, frechet inception distance (FID), maximum mean discrepancy (MMD), Classifier Two-sample Tests (C2ST), etc. Among these methods, IS and FID are easy to implement and reasonable to provide more information regarding the ResNet generator.

The IS involves using a pre-trained deep learning neural network model for image classification to classify the generated orbital data. Specifically, the probability of the image belonging to each class is predicted. The inception score has a lowest value of 1.0 and a highest value of the number of classes supported by the classification model. The core idea of IS is that images are classified strongly as one class over all other classes indicting a high quality. As such, the conditional probability of all generated data in the collection should have a low entropy. The entropy is calculated as the negative sum of each observed probability multiplied by the log of the probability. The marginal probability is utilized here for all generated samples. Therefore, integral of the marginal probability distribution needs to have a high entropy. These elements are combined by calculating the Kullback-Leibler divergence between the conditional and marginal probability distributions. The equation of this divergence is shown below:

$$KL\ divergence = p(y|x) \times (\log(p(y|x)) - \log(p(y))) \tag{26}$$

The KL divergence is then summed over all images and averaged over all classes and the exponent of the result is calculated to give the final inception score. In our results, the inception score is ~45.55, which has room for improvement. However, this number still shows that the ResNet GAN generator can produce a great distribution near our original orbital data.

FID is a metric that calculates the distance between feature vectors for real and generated images, which summarize how similar two groups are in terms of features of samples. Thus, if two groups of the data are similar, the FID score will be smaller. 0.0 FID score indicates the identical of two groups of the data.

By first loading the pre-trained classifier to calculate the FID score, the output of the model can be removed, and the output converted as the activation from the last pooling layer (a global spatial pooling layer). Then, for the real dataset of the problem domain, predict the features. Then, the feature vector can be calculated for the synthetic data. Then use the following formula to calculate the FID score:

$$d^2 = ||mu_1 - mu_2|| + Tr(C_1 + C_2 - 2 \times \sqrt{C_1 \times C_2}) \tag{27}$$

among the equation, mu_1 and mu_2 refer to the sequence of real data and generated data based on characteristics. The C_1 and C_2 are the covariance matrix for the real and generated feature vectors. The popularity of FID score of 0.013 indicates the excellent performance of our ResNet GAN training process.

6. CONCLUSIONS

The goal of the paper is to demonstrate the increased accuracy of deep learning classification models, with game theory enabled, data augmentation, generated training data for space object behavior detection. The paper has described a residual skipping method employed by 3D-CNNs to enhance the performance of deep learning to solve the general-sum games. To evaluate the deep learning performance, a 3D-CNN is designed and implemented for satellite behavior classification using the generated synthetic data to classify 143 different satellite behaviors. The convolutional filters are initially chosen arbitrarily, so the classification is done randomly. It is noted that as iteration increases, the training model converges nearly to 100% accuracy. Meanwhile, with 92% classification accuracy as well as 0.013 FID score in generator performance, the ResNet improved GAN generator architecture to accurately perform the data augmentation. Overall, our model-based, game theoretic, synthetic data improves the training and validation performance of machine learning algorithms.

Acknowledgements: The work was supported under contracts FA9453-15-C-0423, FA8750-19-C-1000, and FA8750-18-C-0106. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of AFRL, or the U.S. Government.

REFERENCES

[1] Blasch, E., Bosse, E., Lambert, D. A. *High-Level Information Fusion Management and Systems Design*, Artech House, Norwood, MA, 2012.

[2] Blasch, E., Enhanced Air Operations Using JView for an Air-Ground Fused Situation Awareness UDOP, *AIAA/IEEE Digital Avionics Systems Conference*, Oct. 2013.

[3] Shen, D., Pham, K, Blasch, E., Chen, H., Chen, G, Pursuit-evasion orbital game for satellite interception and collision avoidance, *Proc. SPIE*, Vol. 8044, 2011.

[4] McQuaid , I., Merkle, L. D., Borghetti, B., Cobb, R., Fletcher, J., Space Object Identification Using Deep Neural Networks, *AMOS*, 2018.

[5] Kyono, T., Lucas, J., Werth, M., Flecther, J., McQuaid, I., Determining Multi-frame Blind Deconvolution Resolvability using Deep Learning, *AMOS* (2019).

[6] Kahler, B., Blasch, E., Predicted Radar/Optical Feature Fusion Gains for Target Identification, *Proc. IEEE Nat. Aerospace Electronics Conf (NAECON)*, 2010.

[7] Blasch, E., Kahler, B., V-NIIRS Fusion Modeling for EO/IR Systems, *IEEE Nat. Aerospace and Electronics Conf*., 2015.

[8] Blasch, E., Kahler, B., Application of VNIIRS for target tracking, *Proc. SPIE*, Vol. 9473, 2015.

[9] Chen, H-M., Blasch, E., Pham, K., *et al*., An investigation of image compression on NIIRS rating degradation through automated image analysis, *Proc. SPIE*, Vol. 9838, 2016.

[10] Blasch, E., Chen, H., Wang, Z., *et al*., Target Broker Compression for Multi-Level Fusion, *IEEE National Aerospace and Electronics Conference*, 2016.

[11] Zheng, Y., Chen, G., Wang, Z., *et al*., Image quality (IQ) guided multispectral image compression, *Proc. SPIE* 9871, 2016.

[12] Blasch, E., Chen, H-M., Wang, Z., *et al*., Compression induced image quality degradation in terms of NIIRS, *IEEE Applied Imagery Pattern Recognition Workshop*, 2016.

[13] Chen, H-M., Wang, Z., Chen, G., J. M. Irvine, *et al*., Tailoring image compression to mission needs: Predicting NIIRS loss due to image compression. *Proc SPIE*, 2018.

[14] Blasch, E., Chen, H-M., Irvine, J. M., et al., Prediction of compression induced image interpretability degradation, *Opt. Eng.* 57(4), 043108 2018.

[15] Duan, Y., Irvine, J. M., Chen, H-M., *et al*., Feasibility of an Interpretability Metric for LIDAR Data, *Proc SPIE*, 2018.

[16] Chen, H-M., Irvine, J. M., Wang, Z., *et al*., Predicting Interpretability Loss in the Thermal IR Imagery due to Compression, *IEEE Applied Imagery Pattern Recognition Workshop*, 2018.

[17] Lucas, J., Kyono, T., Werth, M., Flecther, J., McQuaid, I., Automated Resolution Scoring of Ground-Based LEO Observations Using Convolutional Neural Networks, *AMOS*, 2019.

[18] Kyono, T., Lucas, J., Werth, M., Claef, B., McQuaid, I., Flecther, J., Machine learning for quality assessment of ground-based optical images of satellites, *Opt. Eng. 59(5), 051403* 2020.

[19] Jia, B., Pham, K. D., Blasch, E., *et al*. Cooperative space object tracking using space-based optical sensors via consensus-based filters, *IEEE Transactions on Aerospace and Electronic Systems* 52 (4), 1908-1936, 2016.

[20] Blasch, E., Ravela, S., Aved, A. (eds.), *Handbook of Dynamic Data Driven Applications Systems*, Springer, 2018.

[21] Wei, M., Chen, G., Cruz, J. B., *et al*., Multi-pursuer multi-evader pursuit-evasion games with jamming confrontation, *Journal of Aerospace Computing, Information, and Communication* 4 (3), 693-706, 2007.

[22] Shen, D., Pham, K., Blasch, E., et al., Pursuit-evasion orbital game for satellite interception and collision avoidance, *Proc. SPIE*, Vol. 8044, 2011.

[23] Jia, B., Blasch, E., Pham, K. D., *et al*., Space Object Classification using Fused Features of Time Series Data, *Advanced Maui Optical and Space Surveillance Technologies* (*AMOS)*, 2017.

[24] Shen, D., *et al*. Adaptive Markov inference game optimization (AMIGO) for rapid Discovery of satellite behaviors. Sensors and Systems for Space Applications XII. Vol. 11017. International Society for Optics and Photonics, 2019.

[25] Shen, D., Sheaff, C., Guo, M., Blasch, E., Pham, K., & Chen, G. (2020, May). Enhanced GANs for satellite behavior discovery. In Sensors and Systems for Space Applications XIII (Vol. 11422, p. 114220F). International Society for Optics and Photonics.

[26] Shen, Dan, Khanh Pham, and Genshe Chen. "A Numerical Solution to Orbital Pursuit-Evasion Games." amos (2018): 61.

[27] Brown, G. W. Iterative Solutions of Games by Fictitious Play, In *Activity Analysis of Production and Allocation*, T. C. Koopmans (Ed.), New York: Wiley, 1951.

[28] Berger, U., Brown's original fictitious play, *Journal of Economic Theory* 135:572–578, 2007.
[29] Shen, D., *et al*. Computer vision and pursuit–evasion game theoretical controls for ground robots. *Advances in Mechanical Engineering* 11.8, 1687814019872911, 2019.
[30] Shen, D., Pham, K., Chen. G., "A Numerical Solution to Orbital Pursuit-Evasion Games," *The Advanced Maui Optical and Space Surveillance Technologies Conference*, 2018.
[31] Pham, K. D., *et al*. Methods and devices for demonstrating three-player pursuit-evasion game, *U.S. Patent* No. 9,737,988. 22 Aug. 2017.
[32] Shen, D., *et al.* A game theoretic DRA approach for improved spread spectrum frequency hopped waveforms performance in the presence of smart jammers, *Cognitive Communications for Aerospace Applications Workshop* (CCAA). IEEE, 2017.
[33] Shen, D., *et al*., Game optimal sensor management strategies for tracking elusive space objects. *IEEE Aerospace Conference*. 2017.
[34] Shen, D., *et al*., Pursuit-evasion game theoretic uncertainty oriented sensor management for elusive space objects, *IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*. IEEE, 2016.
[35] Shen, D., *et al*., Network survivability oriented Markov games (NSOMG) in wideband satellite communications, *IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*. IEEE, 2014.
[36] Chen, G., Shen, D., Kwan, C., Cruz, J., Kruger, M., Blasch, E., Game Theoretic Approach to Threat Prediction and Situation Awareness, *Journal of Advances in Information Fusion*, 2(1). 35-48, 1 June 2007.
[37] Blasch, E., *et al*. Review of game theory applications for situation awareness. *Sensors and Systems for Space Applications VIII. Vol. 9469. International Society for Optics and Photonics*, 2015.
[38] Goodfellow, I., *et al*. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.
[39] Mehdi Mirza, Simon Osindero, Conditional Generative Adversarial Nets, *ArXivs* Nov 2014
[40] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, *30th Conference on Neural Information Processing Systems (NIPS),* Barcelona, Spain, 2017
[41] Odena, Augustus, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans." in *International Conference on Machine Learning*, pp. 2642-2651. 2017..
[42] Abhishek Kumar, Prasanna Sattigeri, P. Thomas Fletcher, Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference. *31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA,* 2017
[43] Wang, M., Li, H., & Li, F. Generative Adversarial Network based on Resnet for Conditional Image Restoration. *ArXiv, abs/1707.04881*. 2017