Adaptive-Gravity: A Defense Against Adversarial Samples

Ali Mirzaeian*, Zhi Tian*, Sai Manoj P D*, Banafsheh S. Latibari‡, Ioannis Savidis†, Houman Homayoun‡, Avesta Sasan‡
*Department of Electrical and Computer Engineering, George Mason University
†Department of Electrical and Computer Engineering, Drexel University
‡Department of Electrical and Computer Engineering University of California, Davis
{amirzaei, ztian1, spudukot}@gmu.edu, is338@drexel.edu, {bsaberlatibari, hhomayoun, asasan}@ucdavis.edu

2022 23rd International Symposium on Quality Electronic Design (ISQED) | 978-1-6654-9466-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISQED54688.2022.9806214

Abstract— **This paper presents a novel model training solution, denoted as Adaptive-Gravity, for enhancing the robustness of deep neural network classifiers against adversarial examples. We conceptualize the model parameters/features associated with each class as a *mass* characterized by its centroid location and the spread (standard deviation of the distance) of features around the centroid. We use the centroid associated with each cluster to derive an anti-gravity force that pushes the centroids of different classes away from one another during network training. Then we customized an objective function that aims to concentrate each class's features toward their corresponding new centroid, which has been obtained by anti-gravity force. This methodology results in a larger separation between different *masses* and reduces the spread of features around each centroid. As a result, the samples are pushed away from the space that adversarial examples could be mapped to, effectively increasing the degree of perturbation needed for making an adversarial example. We have implemented this training solution as an iterative method consisting of four steps at each iteration: 1) centroid extraction, 2) anti-gravity force calculation, 3) centroid relocation, and 4) gravity training. Gravity's efficiency is evaluated by measuring the corresponding fooling rates against various attack models, including FGSM, MIM, BIM, and PGD using LeNet and ResNet110 networks, benchmarked against MNIST and CIFAR10 classification problems. Test results show that Gravity not only functions as a powerful instrument to robustify a model against state-of-the-art adversarial attacks but also effectively improves the model training accuracy.**

Index Terms—**Adversarial Example, Convolutional Neural Network, Latent Space Separation**

I. Introduction and Background

Despite the evolution in neural networks model structures and their performance, it is illustrated that CNNs and DNNs are prone to adversarial attacks through simple perturbation of their input images [1]–[6]. These algorithms have demonstrated how easily the normal images can be perturbed by adding a small amount of noise to degrade the performance of neural networks. The vulnerability of deep neural networks to adversarial examples was first investigated in [7]. Since this early work, many new algorithms for generating adversarial examples and various solutions for defending against these attacks are proposed.

Adversarial sample is introduced as an optimization problem, mathematically defined as follows [7]:

$$\underset{\epsilon}{\operatorname{argmin}}\, f(x+\epsilon)=t \qquad s.t. \begin{cases}(x+\epsilon)\in D,\\ f(x+\epsilon)\neq f(x)\end{cases} \tag{1}$$

In this optimization problem, f is a classifier that maps image pixel vectors x to a discrete k-label set t, i.e., $f:\mathbb{R}^m\rightarrow\{1...k\}$. The goal of this optimization formula is to find the minimum perturbation ϵ, such that by applying it to the original data sample x, the under-attack machine learning model f misclassifies the perturbed sample $x+\epsilon$ as the target class t, $f(x+\epsilon)=t$. The obtained perturbed sample $x+\epsilon$ also needs to remain in the acceptable input domain i.e., $D\in[0,1]^m$. In Szegedy and et al. [7], this problem was solved using LBFGS algorithm. Although their offered solution is effective, it is a time-extensive process to achieve the adversarial perturbation.

In [1], Goodfellow and et al, introduced Fast Gradient Sign Method (FGSM) that, unlike LBFGS, was fast and effective. However, it perturbs all the input pixels for obtaining the adversarial example. But only a subset of input pixels can be found that has a similar effect and at the same time leads to a more imperceptible adversarial perturbation. Soon after FGSM, many algorithms like Basic Iterative Method (BIM) [2], Momentum Iterative Method (MIM) [3], and Projected Gradient Descent (PGD) [8] have introduced. In which, through an iterative procedure a minimum amount of adversarial perturbation is generated that leads to a successful adversarial attack.

In essence, adversarial examples could be constructed due to the underlying model's lack of adequate generality. By this intuition, many defenses against adversarial examples have been introduced that we illustrate some of the most notable ones as follow:

Adversarial training [1]: is an iterative procedure that at each iteration, the target model is being trained based on the training dataset. Different attacks are then applied to the model, and the extracted adversarial examples are added to the training dataset. This procedure continues till reaching an acceptable level of robustness. This method has two drawbacks: 1) it can only make the model robust against the assistant attacks; 2) It also increases the training time significantly.

Defensive Distillation: In [9], distilling was used to propose the defense method. For defensive distillation, the second network is the same size as the first network [9]. The main idea is to hide the gradients between the pre-softmax and softmax layers to make the attacker's job more difficult. However, it was illustrated in [10] that this defense could be broken by using the pre-softmax layer outputs in the attack algorithm and/or choosing a different loss function.

Gradient Regularization: Input gradient regularization was first introduced in [11] to improve the generalization of neural networks training by a double backpropagation method. The work in [9] mentions the double backpropagation as a defense, and [12] evaluates the effectiveness of this idea to train a more robust neural network. This approach intends to ensure that if there is a small change in the input, the change in Kullback-Leibler (KL) divergence between the predictions and the labels will also be small. However, this approach is sub-optimal because of the blindness of the gradient regulation.

Adversarial Detection: Another approach is to detect adversarial examples before feeding them to the network [13], [14]. [13] tries to find a decision boundary to separate adversarial and clean inputs. [14] deploys the fact that the perturbation of pixel values by adversarial attack alters the dependence between pixels. By modeling the differences between adjacent pixels in natural images, deviations due to adversarial attacks can be detected.

Autoencoders: Authors in [15] analyze the use of normal and denoising autoencoders as a defense method against adversarial samples. [16], uses a two-level module and uses autoencoders to detect and reform adversarial images before feeding them to the target classifier. However, this method may change the clean images and add a computational overhead to the whole defense-classifier module. To improve the method introduced in [16], the work in [17] presents an efficient auto-encoder with a new loss function, which was learned to preserve the local neighborhood structure on the data manifold.

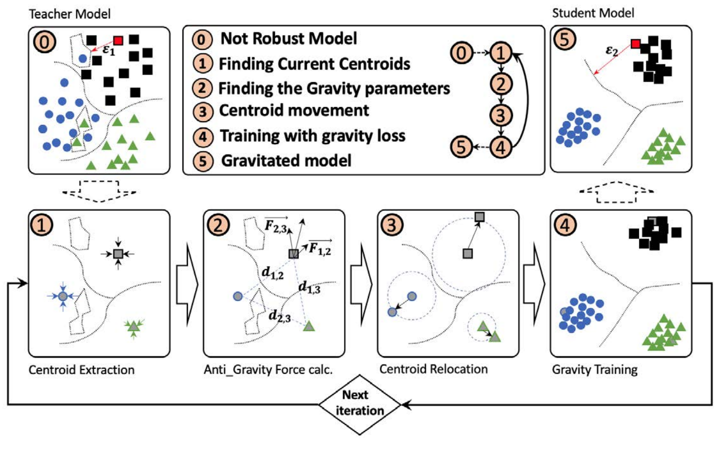


Fig. 1. Gravitation process for robustifying a given model, indicated with step zero, through an iterative procedure, showed with steps one to four, to generate a resilience model against adversarial attacks, showed with step five.

II. Adaptive-Gravity: Proposed Adversarial Defense

In the previous sections, we have investigated different attacks and defenses related to the adversarial examples. One of the primary reasons for the attacks' existence is the low generality of the underlying model, i.e., the underlying model has trained on a dataset that is a subset of all the feasible data samples in the input domain. In this section, we present the proposed technique, termed Adaptive-Gravity, as an effective method that increases the underlying model's generality and simultaneously preserves (or increases) the accuracy of the ML classification model. Further, we combine the proposed gravity method with adversarial training to enhance the model robustness against adversaries.

An abstract view of the proposed procedure is depicted in Fig. 1. Let's assume that Fig. 1-⓪ shows a latent space of a hypothetical model with three categories, indicated by black squares, blue circles, and green triangles. Due to the inherent high non-linearity of the model, there is a boundary of the class blue-circles inside the boundary of black-squares, making it much easier for an adversary to find a small adversarial perturbation ϵ_1 which can lead to an imperceptible adversarial example.

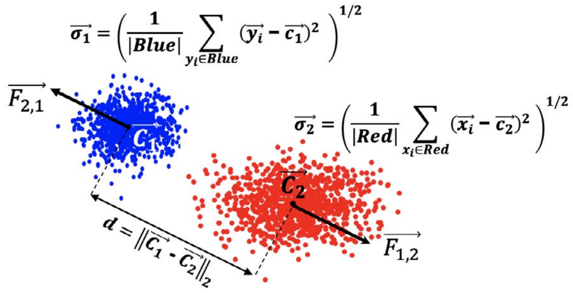
In order to make it much harder for the adversary to find a perturbation, Adaptive-Gravity employs an iterative procedure comprised of four steps which are marked with ① to ④ in Fig. 1. By following these steps a robust model can be obtained as shown in Fig. 1-⑤ . These steps will be elaborated on in this section. With this resultant model, the adversary needs a much larger perturbation ϵ_2 to deceive the underlying model to misclassify the sample marked with the red rectangle in Fig. 1-⑤ compared with Fig. 1-⓪ i.e., $\epsilon_2 >> \epsilon_1$, which means the resilience of the gravitated model is much higher compared to the non-robust model.

To understand the intuition behind Adaptive-Gravity, we first illustrate the concept of traditional gravity force between two masses m_1 and m_2 which are at distance of d from each other as $\overrightarrow{F}_{m_1,m_2} = G\frac{m_1 m_2}{d^2}$, in which G is constant [18]. In this equation, the gravity force $\overrightarrow{F}_{m_1,m_2}$ increases linearly by increasing the mass of either m_1 or m_2, or it increases quadratically by reducing the distance between these two masses. With a similar intuition to the gravity force between two masses, we analogize each class as a mass at the centroid of that particular class. The standard deviation of each class, denoted by σ, is considered as the size of the mass. The L_2 norm between centroids of classes depicts the distance between their corresponding latent spaces. These parameters are shown in Fig. 2 for two classes with a different distribution of their latent spaces. By defining these parameters, we formulate the anti-gravity force between two classes as described in Eq. (2) below.

$$\overrightarrow{F}_{1,2} = \frac{\overrightarrow{\sigma}_1 \overrightarrow{\sigma}_2}{d^2}(\overrightarrow{c_1} - \overrightarrow{c_2}) \qquad (2)$$

The anti-gravity force shows the direction and magnitude of the force that classes impose on each other to keep the different classes away from themselves. This formula serves the following purposes: 1) those classes that have a lower L_2 distance have a larger anti-gravity force, i.e., those classes that are already far apart from each other, having a higher L_2 distance, have a minimum effect on other classes. Conceptually, those classes that are closer to each other are more vulnerable to adversarial attacks. 2) Classes with a bigger mass (larger standard deviation) are more susceptible to adversarial attacks. Because samples of these classes are more likely to locate inside other classes' boundaries. This, in turn, lowers the trained model's generality and, consequently, makes it more vulnerable against adversarial attacks.

By this definition, Adaptive-Gravity iterates over four steps 1) Centroid Extraction, Fig. 1-① , 2) Anti-Gravity force calculation, Fig. 1-② , 3) Centroid relocation, Fig. 1-③ , 4) Gravity Training, Fig. 1-④ .



$\overrightarrow{\sigma_1} = \left(\frac{1}{|Blue|}\sum_{y_i \in Blue}(\overrightarrow{y_i} - \overrightarrow{c_1})^2\right)^{1/2}$
$\overrightarrow{\sigma_2} = \left(\frac{1}{|Red|}\sum_{x_i \in Red}(\overrightarrow{x_i} - \overrightarrow{c_2})^2\right)^{1/2}$
$\overrightarrow{F_{2,1}}$
C_1
C_2
$\overrightarrow{F_{1,2}}$
$d = \|\overrightarrow{C_1} - \overrightarrow{C_2}\|_2$

Fig. 2. Definition of parameters in a model with two classes in which G is gravitization constant, $\overrightarrow{\sigma_1}$ and $\overrightarrow{\sigma_2}$ are the standard deviation of class 1 and class 2 respectively, d is the L_2 norm distance between two centroids c_1 and c_2 which are associated to class 1 and class 2 respectively

A. Centroid Extraction

At this step, Fig. 1-① , the mean of the embedded features in each class is considered as the corresponding centroid for that particular class. In this paper, we assume that each class has only one centroid, which can be obtained by averaging each class's embedded features. It is possible to consider more than one centroid for each class by employing the K-Means algorithm to locate multiple centroids for each class. The obtained centroid for i^{th} class is denoted by $\overrightarrow{c_i} = \frac{1}{M_i}\sum_{j=1}^{M_i}\overrightarrow{x_j}$ in which M_i is the cardinality of the i^{th} class and $\overrightarrow{x_j}$ is the j^{th} member of the i^{th} class. Each centroid is considered as a representative for a class that the whole mass of that class is concentrated on that centroid. Because all the steps of Gravity rely on the centroids of classes, the proper centroids can significantly change the performance of Gravity.

B. Anti-Gravity Force

The second step, Fig. 1-② , is to obtain the total force on the ith centroid ($\overrightarrow{c_i}$) on a model with N classes. According to Eq. (2), the anti-gravity force $\overrightarrow{F}_{i,j}$ between two classes i and j is determined by three sets of quantities: $\overrightarrow{\sigma_i}$, d_{ij}, and $\overrightarrow{c_i}$. The standard deviation of the latent space related to ith class, ϕ_i, is obtained by $\overrightarrow{\sigma_i} = \sqrt{\frac{1}{M_i}\sum_{j=1}^{M_i}\left(\overrightarrow{x_j} - \overrightarrow{c_i}\right)^2}$ in which $\overrightarrow{x_i} \in \phi_i$ and M_i is the cardinality of the i^{th} class. The L_2 distance between $\overrightarrow{c_i}$ and the rest of the centroids ($\vec{c_j}$) is calculated by $d_{ij} = \| \vec{c_i} - \vec{c_j} \|_2$. Lastly, the total anti-gravity force on $\vec{c_i}$ is calculated by summing up all the anti-gravity forces on this centroid as shown in Eq. (3).

$$\vec{F_i} = \sum_{j=1, j \neq i}^{N} \overrightarrow{F_{i,j}} = \sum_{j=1, j \neq i}^{N} \frac{\vec{\sigma_i}\vec{\sigma_j}}{d_{ij}^2} (\vec{c_i} - \vec{c_j}) \tag{3}$$

The anti-gravity force for each centroid is different from others, meaning centroids are forced to move in different directions with different step sizes. Those centroids that are far from other centroids are subject to a weaker anti-gravity force than those centroids that are close to each other. That is because those classes that are close are more susceptible to adversarial attacks, so those should be separated more aggressively.

For instance, consider a simple example of three classes with centroids $\vec{c_1}, \vec{c_2}, \vec{c_3}$ and standard deviations $\vec{\sigma_1}, \vec{\sigma_2}, \vec{\sigma_3}$ which are located at L_2 distances d_{12}, d_{13}, d_{23} from each other as shown in Fig. 3-left. In Fig. 3-right, the anti-gravity force at each centroid has been shown with a red arrow. As shown, the anti-gravity between classes two and three is much larger than the anti-gravity force toward class one. This is because classes two and three are close to each other and have a larger mass, while class one is far apart from the two others and has a smaller mass. Consequently, class one imposes the minimum anti-gravity force toward the other two classes, i.e., the other two classes have the minimum anti-gravity force toward class one.

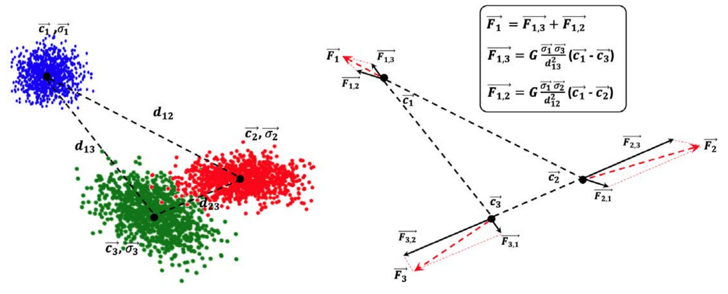

$\vec{c_1}, \vec{\sigma_1}$
d_{12}
d_{13}
$\vec{c_2}, \vec{\sigma_2}$
d_{23}
$\vec{c_3}, \vec{\sigma_3}$
$\vec{F_1}$
$\overrightarrow{F_{1,3}}$
$\overrightarrow{F_{1,2}}$
$\vec{c_1}$
$\vec{F_1} = \overrightarrow{F_{1,3}} + \overrightarrow{F_{1,2}}$
$\overrightarrow{F_{1,3}} = G\frac{\vec{\sigma_1}\vec{\sigma_3}}{d_{13}^2}(\vec{c_1} - \vec{c_3})$
$\overrightarrow{F_{1,2}} = G\frac{\vec{\sigma_1}\vec{\sigma_2}}{d_{12}^2}(\vec{c_1} - \vec{c_2})$
$\overrightarrow{F_{2,3}}$
$\vec{F_2}$
$\vec{c_2}$
$\overrightarrow{F_{2,1}}$
$\vec{c_3}$
$\overrightarrow{F_{3,2}}$
$\overrightarrow{F_{3,1}}$
$\vec{F_3}$

Fig. 3. Total anti-gravity forces $\vec{F_1}$, $\vec{F_2}$, $\vec{F_3}$ corresponding to each one the classes with centroids $\vec{c_1}$, $\vec{c_2}$, $\vec{c_3}$ and standard deviations $\vec{\sigma_1}$, $\vec{\sigma_2}$, $\vec{\sigma_3}$, respectively.

C. Centroid Relocation

At this step, Fig. 1-3 , the obtained centroids in Section II-A are relocated in the direction of the total anti-gravity forces obtained in section II-B. Because there is no constraint on the magnitude of each centroid's total anti-gravity forces, the new centroids' locations can be too far from their current locations. So it is hard for the underlying model to get trained to converge the latent spaces of each class to their corresponding new centroid, which leads to a dramatic drop in the model's accuracy. To prevent this phenomenon, we define a normalized anti-gravity force $\vec{\alpha_i}$ for each class that constraints the maximum distance of the next centroid of a class to its current location. $\vec{\alpha_i}$ is a vector in the same direction of $\vec{F_i}$ with a hyper-parameter G to adjust the magnitude of it. This procedure is illustrated in Alg. 1.

In Alg. 1, the inputs are anti-gravity forces $\vec{F}$, Gravity constant G, and the list of classes' centroids c. In line 2, the anti-gravity force that has the largest magnitude is selected then as in line 4, the normalized anti-gravity force $\vec{\alpha_i}$ for each one of the centroids is calculated, and finally, in line 5, each centroid is updated by the calculated $\vec{\alpha_i}$. With this algorithm, each centroid at the kth iteration of Gravity i.e., $\vec{c}_i^{\,k}$, can be relocated within the L_2 distance range $(0, G]$ in the same direction of $\vec{\alpha_i}$.

Algorithm 1 Centroid Modification

```
1: Inputs: $\vec{F}$: list of anti-gravity forces, $G$: constant, $C$: list of classes' centroid
2: $F_M = max(|\vec{F}|_2)$
3: for ($i = 1$; $i \leq N$ ; $i$++) do
4:     $\vec{\alpha_i} = G * (\vec{F}_i / F_M)$
5:     $\vec{c}_i^{\,k+1} = \vec{\alpha_i} + \vec{c}_i^{\,k}$
6: end for
```

Using this simple method, Adaptive-Gravity adds a priority measure to each centroid to indicate how aggressively each centroid should move. Adaptive-Gravity assigns a higher priority to separate classes close to one by using a maximum step size G, while spending less effort in separating classes that are far apart. For example, in Fig. 3, $\vec{F_1}$ has the lowest magnitude, so at the next iteration $\vec{c_1}$ has the smallest relocation compared to other two centroids $\vec{c_2}$ and $\vec{c_3}$.

D. Gravity Training

Gravity training, Fig. 1-4 , is an iterative procedure that performs in a teacher/student fashion. The student model has the same structure as the teacher model. During this process, a student model is trained such that its accuracy follows the teacher's accuracy but covers the teacher model's weakness against adversarial attacks.

1) Gravity Force: So far, the formulated training process pushes the centroids of different clusters away from one another. To further increase the separation, we also need to reduce each mass's spread (standard deviation of the distance between features and centroid in each class). By reducing the mass of the classes, they become more concentrated toward their centroids. It needs to be noted that, by decreasing the standard deviation of classes, the latent spaces of classes are less likely to overlap, and this leads to a model with a higher generality that is less vulnerable toward adversarial examples.

For this purpose, gravity force has been defined to reduce the standard deviation of classes. Expressly, we represent the gravity force as Mean Square Error (MSE) between each class's latent spaces and its corresponding centroids. Conceptually, minimizing the gravity force of a class leads to a more concentrated latent space around its centroid. To model the force of gravity in our optimization problem, we formulate the gravity force using the *gravity loss function*, details of which are discussed in the next section.

2) Gravity Loss Function: At this step, the model is trained to minimize the gravity loss using the Eq. (4) through an iterative process. This equation consists of two parts: 1) The first part that targets accuracy is annotated with $L_{ce}^{(k)}$ and defines the cross-entropy loss, Eq. 5, between the ground truths Y and the outputs of the student model i.e., $S(X_{m\times n}, Y)$, at the kth iteration. 2) The second part that targets security aims to minimize the gravity force, i.e., converging the latent space, ϕ_i, toward their corresponding centroids $\vec{c_i}$. This structure is similar to the defensive distillation method discussed earlier. Despite the similar structure, Gravity uses two layers, the head, and tail of the teacher model, to transfer their knowledge to the student model's corresponding layers.

$$L_{Gr}^{(k)} = \underbrace{(1-\gamma_{k-1})L_{ce}^{(k)}}_{targets\,the\,accuracy} + \underbrace{\gamma_{k-1}(L^{(l=-1,k)} + L^{(l=i,k)})}_{targets\,the\,security} \tag{4}$$

$$L_{ce}^{(k)} = -\sum_{i=1}^{i=m\times n} Y_i \log(S^k(X_i)) \tag{5}$$

$L^{(l=-1,k)}$ and $L^{(l=i,k)}$ are associated to the student's head and tail objective function at the kth iteration and are defined as in Eq. (6) with $l=-1$ and $l=i$, respectively. In these equations, Mean Square Error (MSE) calculates the likelihood between the latent spaces of the head (or tail) layer of the student model, $S_{\phi_{m\times n}}^{(l=-1,k)}$, and the corresponding centroids of the teacher model, $S_{c_m}^{(l=-1,k-1)}$.

$$L^{(l,k)} = \frac{1}{m\times n} \sum_{j=1}^{m} \sum_{f=1}^{n} \left(S_{c_j}^{(l,k-1)} - S_{\phi_{(f,j)}}^{(l,k)} \right)^2 \tag{6}$$

TABLE I
LENET AND RESNET-110 STRUCTURES ARE USED FOR TRAINING MNIST AND CIFAR10 DATASETS, RESPECTIVELY. THE HEAD AND TAIL LAYERS FOR EACH MODEL HAVE BEEN HIGHLIGHTED. SYSTEM CONFIGURATION AND TRAINING HYPERPARAMETERS ALSO HAVE SHOWN ON THE BOTTOM.

LeNet	ResNet-110
Conv(6, 5×5)	Conv(16, 3×3)+BN
ReLu(2×2)	ReLu(2×2)

Conv(16, 5×5)
PReLu(2×2)
	$\begin{bmatrix} \text{Conv}(16, 1 \times 1)\text{+BN} \\ \text{Conv}(16, 3 \times 3)\text{+BN} \\ \text{Conv}(64, 1 \times 1)\text{+BN} \end{bmatrix} \times 12$

FC(120) Tail	$\begin{bmatrix} \text{Conv}(32, 1 \times 1)\text{+BN} \\ \text{Conv}(32, 3 \times 3)\text{+BN} \\ \text{Conv}(128, 1 \times 1)\text{+BN} \end{bmatrix} \times 12$
FC(84)	$\begin{bmatrix} \text{Conv}(64, 1 \times 1)\text{+BN} \\ \text{Conv}(64, 3 \times 3)\text{+BN} \\ \text{Conv}(256, 1 \times 1)\text{+BN} \end{bmatrix} \times 12$
FC(10) Head	FC(1024) Tail
	FC(10) Head

System Configuration and training hyper parameters
OS: Red Hat 7.7, Pytorch: 1.3, AdverTorch: 0.2, GPU: Nvidia Tesla V100, EPOCH: 100, MNIST Batch Size: 64, CIFAR10 Batch Size:128, Optimizer: ADAM, MNIST learning rate: 1e-4, CIFAR10 learning rate: 5e-3

Several remarks are in order. First, at the first iteration, Adaptive-Gravity transfers as much as possible knowledge from the teacher model to the student model. But at the later iterations, Adaptive-Gravity focuses on securing the student model, i.e., removing those vulnerabilities transferred from the teacher model during the first iteration. Second, γ_i contains the accuracy of the teacher model at each iteration. So if this accuracy is high, the student model relies more on the teacher model, which brought in the Gravity Loss by the second term. However, if the teacher model's accuracy is low, the student model relies on itself, which is the first term. Using this strategy, the student model adds more intelligence to its learning process to not follow the teacher model if performance degradation occurs.

III. EXPERIMENTS

In order to assess the performance of the Adaptive-Gravity method we used two architectures LeNet [19] and ResNet-110 [20], see Table I, for training and evaluating on two datasets MNIST [19] and CIFAR10 [21], respectively. In this section, we evaluate our results in three sections 1) performance of proposed gravity technique, the impact of Gravity on 2) White-box attacks, 3) Black-box attacks. The first section shows how effective Gravity separates the latent spaces of a target layer of an underlying model. Sections two and three show how resistant the gravitated model is against different adversarial examples in the white-box and black-box scenarios.

A. *Gravity Performance*

In this section, we measure three metrics 1) Inter-Class Convergence (ICC), that indicates how far, L_2 norm, are the latent spaces of the head or tail layer from their designated centroid at each iteration. ICC is obtained by summing masses of all classes, see Fig. 2, i.e., $\sum_{i=1}^{N} \epsilon_i$. 2) Intra-Class Divergence (ICD), measures the relative distance, L_2 norm, of the designated centroids for the head and tail layers at each iteration, see Fig. 2. ICD contains three values I) Average distance i.e., $1/N \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}$, II) Maximum distance i.e., $\max(d_{ij})$, and III) Minimum distance i.e., $\min(d_{ij})$. In which d_{ij} indicates the distance between centroids of i^{th} and j^{th} classes. 3) Evaluation accuracy, that measures the student evaluation accuracy at each iteration.

Figure 4 shows the evaluation of these three metrics during 50 iterations of Gravity for training the models ResNet110 and LeNet, see the table I, on two datasets CIFAR10 and MNIST, respectively. By studying Fig. 4 we have the following observations:

- The first row shows a different pattern for the ICD metric associated with the head and tail layers of LeNet compared with ResNet110. In the LeNet, the ICD of the head and tail layers show a similar behavior by increasing in a logarithmic form. Meaning, in the early iterations of Gravity, the ICD increases with a larger magnitude than later iterations. Unlike LeNet, in the ResNet110 after the 5th iteration, the ICD of the tail layer comes under the head layer. One interpretation is that in the LeNet, Gravity successfully moves the centroids further apart because MNIST is a simple dataset. Increasing the ICD of the head layer has no contradictory impact on the ICD of the tail layer. However, CIFAR10 is a much more complex dataset, and consequently, after 5th iteration increasing the ICD of the head layer leads to decreasing the ICD of the tail layer.
- The second row, associated with the ICC metric for the tail and head layers, shows different behavior regarding the LeNet and ResNet110. For the LeNet, ICC increases rapidly at the early iterations, but later, this growth is diminishing in both head and tail layers. For the ResNet110, the ICC of the head layer continually increases (more rapidly at the early iterations), but the ICC of the tail layer rapidly increases till the 7th iteration then start a diminishing decline.
- The third row, which is related to the evaluation of the student model's accuracy, confirms that Gravity can increase the accuracy of the underlying models while separating the latent spaces of the student's head and tail layers. Noted that the reported accuracy at the 0th iteration indicates the original accuracy of the model we wanted to harden it. One reason for improving accuracy is that Gravity helps increase the generality of the models, i.e., models show a higher accuracy at the evaluation phase.

A secure model (against adversarial input perturbation attacks) should resist adversarial examples and generate the correct classification outcome maintaining a high classification accuracy for both adversarial and normal inputs. By increasing ICD and decreasing ICC, and preserving (or increasing) the accuracy, we expect the underlying model's robustness to increase. However, from our observations presented in Fig. 4, these conditions are not always satisfied during different iterations. Hence we need to embrace a strategy to select one iteration of Gravity that is the most suitable for the robustifying task. For this purpose, we establish an acceptable threshold for accuracy at the first step, which is shown with dash lines in the third row of fig. 4. In the next step, we measure the relation between the normalized ICC which is defined as $\frac{\text{MinICC}_{\text{head}} \times \text{MinICC}_{\text{tail}}}{\text{Max}(\text{MinICC}_{\text{head}}) \times \text{Max}(\text{MinICC}_{\text{tail}})}$, and the normalized ICD which is defined as $\frac{\text{ICD}_{\text{head}} \times \text{ICD}_{\text{tail}}}{\text{Max}(\text{ICD}_{\text{head}}) \times \text{Max}(\text{ICD}_{\text{tail}})}$, for those iterations resulting in an accuracy level above the threshold. In the third step, we obtain the Pareto-front between those iterations that satisfying the threshold requirement. Noted that each Pareto-front is associated with one iteration, so in the fourth step, between these Pareto-fronts, we pick one iteration that leads to the best resistance against adversarial attack. Here we used the PGD attack as one of the most powerful first-order adversarial attacks. Finally, The model parameters generated in the selected iteration are then used in the student model.

We set the threshold to 0.9965 for the MNIST dataset and 0.92 for the CIDAR10 dataset, as shown with the red-dash line in Fig. 4 third row. Then we obtained the Pareto-fronts for those selected iterations as shown in Fig. 4 fourth row. Between all the Pareto-fronts, the 10th and 8th iterations show the most resilience against PGD attack for the LeNet and ResNet110, respectively. So for the next evaluations, we use the parameters associated with these iterations.

Figures 5-(A) and (B) illustrate the trajectory of the centroid movement of the latent spaces associated with the head and tail layers of LeNet, marked with blue and green in Table I, on the MNIST dataset. In these figures, the centroids of classes at each iteration are shown with a different color. For example, the red color is associated with the ten centroids at the first iteration,

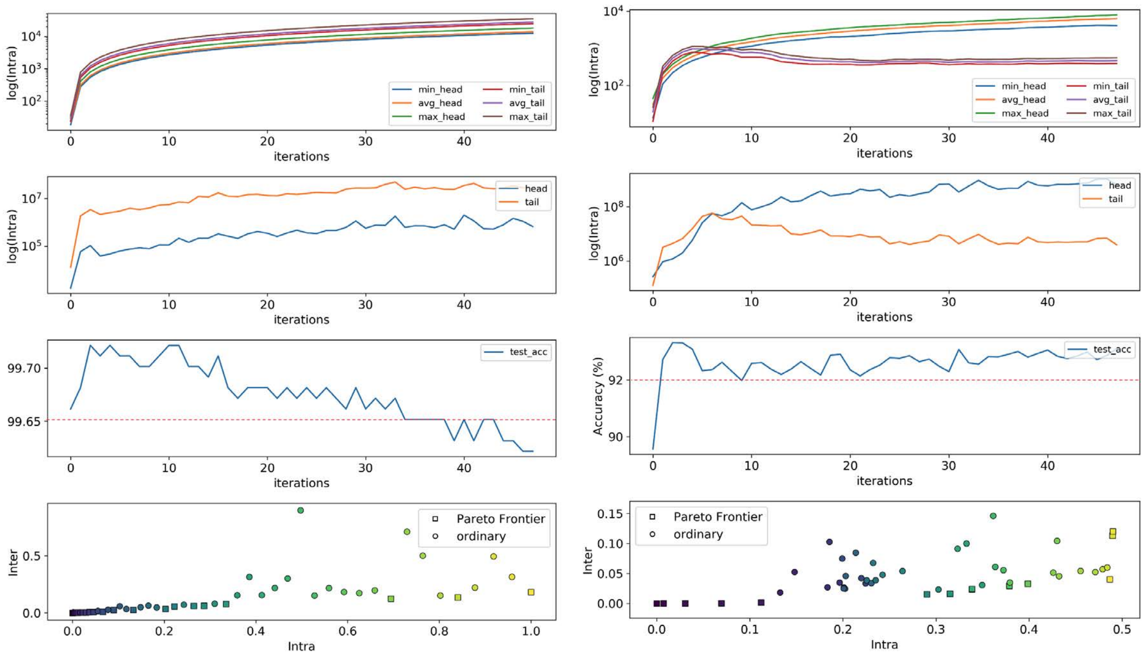


Fig. 4. ICD (first row), ICC (second row), accuracy (thrid row), Pareto metrics of gravity during different iterations. The fourth row shows the relation between normalized ICC and ICD. In the fourth row, purple is related to the early iteration, and yellow is related to the 50th iteration. Left and right figures are related to the LeNet and ResNet110 models trained on MNIST and CIFAR10 datasets, respectively.

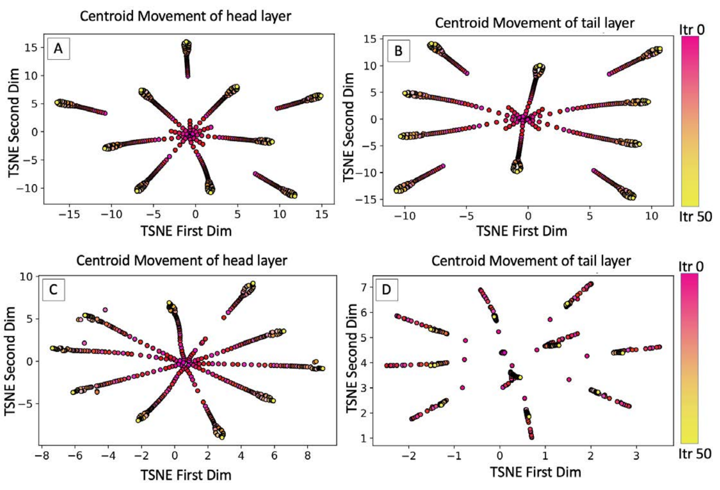


Fig. 5. (A), (B) shows the LeNet centroid movement of the head and tail layers with G_{head} = 100 and G_{tail} = 200 on MNIST dataset. (C), (D) shows the ResNet110 centroid movement of the head and tail layers with G_{head} = 200 and G_{tail} = 300 on CIFAR10 dataset. Iterations are color-coded with red as the 0th iteration to yellow as the 50th iteration.

and similarly, the yellow ones are corresponding to 50th iteration. This figure confirms that by increasing the number of iterations at the Adaptive-Gravity, the classes and subsequently their related centroids are moving away from each other. Figure 5 captures similar information for the head and tail layers of ResNet110 that was trained on the CIFAR10 dataset. Unlike LeNet, the trajectory of centroid movement of head and tail layers of ResNet110 shows different behavior, as explained earlier in this section.

B. Performance against White-Box Attacks

In white-box attack the attacker has access to the trained model, i.e., $S(.)$, and the training dataset. Thus, by having these two components, we assess how many of the input samples can be misclassified by the adversary, i.e., the accuracy of the underlying model in the existence of different adversarial attacks indicates the robustness of the model.

The structure of the used model is shown in Table I in which colored rows show the location of the tail and head in the Adaptive-Gravity method. We also combined the Adaptive-Gravity with adversarial training techniques for boosting the model robustness. For adversarial training, we used two different attacks FGSM and PGD, and their results are reported under the names $\text{AdvTrain}_{\text{FGSM}}$ and $\text{AdvTrain}_{\text{PGD}}$, respectively. We have also compared the proposed Adaptive-Gravity technique's performance with the existing aforementioned adversarial defenses in Tables II and III on datasets CIFAR10 and MNIST, respectively.

By investigating the white-box scenario's result, we have observed that: 1) The accuracy of the gravitated model is higher than the baseline. This shows that Adaptive-Gravity has spaced the classes far apart, allowing for better generalization by simplifying the class mapping boundaries, resulting in the correct classification of near-boundary samples misclassified by the baseline model. 2) Combining adversarial training with Gravity has a trivial impact on the accuracy of the underlying model. This is because Adaptive-Gravity has already generalized the model, and combining more samples (adversarial samples) into the training dataset will not significantly change the model's performance. 3) Compared to the other defenses, Adaptive-Gravity shows a much better resilience against adversarial attacks.

C. Adaptive-Gravity against Black-Box Attacks

In The black-box attacks, the attacker only has access to the trained data and can only feed input samples to the model under attack and observe its output. For launching this scenario, we have assumed the attacker has chosen the VGG-19 model to generate adversarial examples and feed them to the model which has trained with the Adaptive-Gravity strategy. We report the results of a black-box attack on a trained model based on Adaptive-Gravity and adversarial training in Table IV.

IV. Conclusion

This paper introduces Adaptive-Gravity, a powerful training solution for hardening Neural Network models against adversarial attacks. Adaptive-Gravity is an iterative method. The corresponding latent spaces related to each head and tail layer are pushed further apart and concentrated toward their corresponding centroid at each iteration. To find the direction for moving the latent spaces apart, we introduced the anti-gravity force. We stipulate a new location for the centroids of each set of latent class features in the tail and head layers' latent spaces using the anti-gravity force. Subsequently,

TABLE II
COMPARISON OF GRAVITY METHOD TO OTHER DEFENSE METHODS ON CIFAR10 DATASET. FOR GRAVITY, WE REPORTED RESULTS WITHOUT ADVERSARIAL TRAINING (GRAVITY) AND WITH ADVERSARIAL TRAINING USING FGSM (GRAVITY$_f$) AND PGD (GRAVITY$_p$) ATTACKS.

Attacks	Params.	Baseline	AdvTrain [22]	Ye et al. [23]	Ross et al. [24]	Pang et al. [25]	Madry et al. [26]	Mustafa et al. [27]	Gravity	Gravity$_f$	Gravity$_p$
No Attacks	-	89.0	84.5	83.1	86.2	90.6	87.3	90.5	**93.0**	91.9	91.32
FGSM	$\epsilon = 0.02$	74.0	44.3	48.5	39.5	61.7	71.6	72.5	90.5	**90.94**	89.57
	$\epsilon = 0.04$	46.0	31.0	38.2	20.8	46.2	47.4	56.3	88.57	**90.12**	87.79
BIM	$\epsilon = 0.01$	82.57	22.6	62.7	19.0	46.6	64.3	62.9	**92.45**	91.79	91.31
	$\epsilon = 0.02$	77.11	7.8	39.3	6.9	31.0	49.3	40.1	**92.28**	91.77	91.25
MIM	$\epsilon = 0.01$	76.41	23.9	-	24.6	52.1	61.5	64.3	90.46	**91.38**	90.66
	$\epsilon = 0.02$	53.93	9.3	-	9.5	35.9	46.7	42.3	88.58	**90.96**	89.67
PGD	$\epsilon = 0.01$	82.8	24.3	-	24.5	48.4	67.7	60.1	91.12	**91.7**	91.09
	$\epsilon = 0.02$	74.38	7.8	-	8.5	30.4	48.5	39.3	86.21	**91.29**	90.47
C&W	c=0.001	52.3	67.7	82.5	72.2	80.6	84.5	91.3	90.82	**91.75**	91.09
	c=0.01	47.38	40.9	62.9	47.8	54.9	65.7	73.7	85.24	87.27	**89.48**
	c=0.1	10.38	25.4	40.7	19.9	25.6	47.9	60.5	83.23	**85.29**	84.17

TABLE III
COMPARISON OF GRAVITY METHOD TO OTHER DEFENSE METHODS ON MNIST DATASET. FOR GRAVITY, WE REPORTED RESULTS WITHOUT ADVERSARIAL TRAINING (GRAVITY) AND WITH ADVERSARIAL TRAINING USING FGSM (GRAVITY$_f$) AND PGD (GRAVITY$_p$) ATTACKS.

Attacks	Params.	Baseline	AdvTrain [22]	Ye et al. [23]	Ross et al. [24]	Pang et al. [25]	Mustafa et al. [27]	Gravity	Gravity$_f$	Gravity$_p$
No Attacks	-	98.63	99.1	98.4	99.2	99.5	99.5	**99.69**	99.68	99.59
FGSM	ϵ=0.1	91.23	73.0	91.6	91.6	96.3	97.1	91.53	97.2	**97.67**
	ϵ=0.2	9.7	52.7	70.3	60.4	52.8	70.6	70.68	68.09	**73.06**
BIM	ϵ=0.1	93.72	62.0	88.1	87.9	88.5	90.2	98.35	98.67	**98.72**
	ϵ=0.15	22.2	18.7	77.1	32.1	73.6	76.3	97.54	98.56	**98.66**
MIM	ϵ=0.1	90.9	64.5	-	83.7	92.0	92.1	97.07	97.28	**98.61**
	ϵ=0.15	7.6	28.8	-	29.3	77.5	77.7	81.49	**84.44**	83.34
PGD	ϵ=0.1	20.15	62.7	-	77.0	82.8	83.6	94.95	**98.46**	98.45
	ϵ=0.15	6.99	31.9	-	44.2	41.0	62.5	68.26	96.97	**97.26**
C&W	c=0.1	60.15	71.1	89.2	88.1	97.3	97.7	96.55	97.3	**98.05**
	c=1	36.99	39.2	79.1	75.3	78.1	91.2	95.6	97.87	**97.92**
	c=10	6.01	17.0	37.6	20.0	23.8	46.0	85.26	**86.97**	85.96

TABLE IV
ROBUSTNESS OF OUR MODEL IN BLACK-BOX SETTINGS. ϵ_{FGSM}, ϵ_{BIM}, ϵ_{MIM}, ϵ_{PGD} SHOWS THE EQUIVALENT ϵ FOR EACH ONE OF THE ATTACKS FGSM, BIM, MIM, AND PGD, THAT LEAD TO THE SAME PSNR IN WHITE-BOX SETTINGS. WE ALSO INVESTIGATE THE IMPACT OF COMBINING GRAVITY WITH ADVERSARIAL TRAINING.

Training	No Attack	FGSM	BIM	MIM	PGD
MNIST ($\epsilon_{\text{FGSM}} = 0.225, \epsilon_{\text{BIM}} = 0.1, \epsilon_{\text{MIM}} = 0.2, \epsilon_{\text{PGD}} = 0.265$)					
Baseline	99.3	39.2	20.15	16.28	20.1
Gravity	99.69	87.33	94.95	91.05	94.0
Gravity+AdvTrain$_{\text{FGSM}}$	**99.76**	**96.41**	**97.92**	**96.93**	97.0
Gravity+AdvTrain$_{\text{PGD}}$	**99.76**	96.04	97.69	96.52	**97.65**
CIFAR10 ($\epsilon_{\text{FGSM}} = 0.01, \epsilon_{\text{BIM}} = 0.01, \epsilon_{\text{MIM}} = 0.01, \epsilon_{\text{PGD}} = 0.03$)					
Baseline	92.4	83.8	85.6	84.1	85
Gravity	**92.5**	87.7	88.9	88.2	88.9
Gravity+AdvTrain$_{\text{FGSM}}$	91.81	**89.45**	**89.9**	**89.5**	**90.0**
Gravity+AdvTrain$_{\text{PGD}}$	92.4	87.7	88.6	87.9	88.6

we deploy a training procedure to adjust the underlying model's parameters to achieve two objectives 1) improving the accuracy of the underlying model 2) concentrating the latent spaces of the tail and head layers to their projected centroids. We evaluated our method against diverse attack scenarios on CIFAR10 and MNIST datasets. Our experimental results indicate that Adaptive-Gravity is an extremely effective and resilient solution for resisting the existing adversarial attacks.

V. ACKNOWLEDGMENT

This work was supported by National Science Foundation (NSF) under NSF award number 1718538.

REFERENCES

[1] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[2] A. Kurakin *et al.*, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[3] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[4] Moosavi-Dezfooli *et al.*, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[5] A. Mirzaeian *et al.*, "Diverse knowledge distillation (dkd): A solution for improving the robustness of ensemble models against adversarial attacks," in *International Symposium on Quality Electronic Design (ISQED) 2021*, 2021.

[6] F. Behnia et al., "Code-bridged classifier (cbc): A low or negative overhead defense for making a cnn classifier robust against adversarial attacks," in *International Symposium on Quality Electronic Design (ISQED) 2020*, 2020.

[7] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[8] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[9] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[10] N. Carlini *et al.*, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[11] H. Drucker and Y. Le Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. on Neural Networks*, vol. 3, no. 6, 1992.

[12] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-second AAAI Conf. on Artificial Intelligence*, 2018.

[13] T. Pang *et al.*, "Towards robust detection of adversarial examples," in *Advances in Neural Information Processing Systems*, 2018, pp. 4579–4589.

[14] J. Liu *et al.*, "Detection based defense against adversarial examples from the steganalysis point of view," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognitionn*, 2019, pp. 4825–4834.

[15] I.-T. Chen and B. Sirkeci-Mergen, "A comparative study of autoencoders against adversarial attacks," in *Proceedings of the Int. Conf. on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[16] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[17] M. Sabokrou *et al.*, "Self-supervised representation learning via neighborhood-relational encoding," in *ICCV*, 2019, pp. 8010–8019.

[18] I. Newton, *Philosophiae naturalis principia mathematica*. G. Brookman, 1833, vol. 1.

[19] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] K. He *et al.*, "Deep residual learning for image recognition," in *proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[22] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[23] F. Yu, C. Liu, Y. Wang, L. Zhao, and X. Chen, "Interpreting adversarial robustness: A view from decision surface in input space," *arXiv preprint arXiv:1810.00144*, 2018.

[24] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," *arXiv preprint arXiv:1711.09404*, 2017.

[25] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," *arXiv preprint arXiv:1901.08846*, 2019.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[27] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Adversarial defense by restricting the hidden space of deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3385–3394.