Hyperspherical class prototypes for adversarial robustness

Vasileios Mygdalis, Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{*mygdalisv, pitas*} *csd.auth.gr*

Author pre-print version, submitted to Elsevier Pattern Recognition.
Please cite the publisher maintained version in your work.
DOI : https://doi.org/10.1016/j.patcog.2022.108527

Abstract

This work addresses the problem of adversarial robustness in deep neural network classification from an optimal class boundary estimation perspective. It is argued that increased model robustness to adversarial attacks can be achieved when the feature learning process is monitored by geometrically-inspired optimization criteria. To this end, we propose to learn hyperspherical class prototypes in the neural feature embedding space, along with training the network parameters. Three concurrent optimization functions for the intermediate hidden layer training data activations are devised, requiring items of the same class to be enclosed by the corresponding class prototype boundaries, to have minimum distance from their class prototype vector (i.e., hypersphere center) and to have maximum distance from the remainder hypersphere centers. Our experiments show that training standard classification model architectures with the proposed objectives, significantly increases their robustness to white-box adversarial attacks, without adverse (if not beneficial) effects to their classification accuracy.

Keywords: Adversarial Defense, Adversarial Robustness, Hypersphere Prototype loss, HCP loss

1. Introduction

The problem of adversarial robustness in deep neural network classification is a relatively new research area that has attracted significant attention in the past few years. It involves studying and addressing the inherent model weaknesses that allow adversaries to easily fool a neural network classifier by carefully crafting input perturbations, the so-called adversarial attacks. The root causes of these weaknesses have not been properly identified yet, however, they are

somehow related with the unified deep learning optimization procedure that involves feature learning and classifier learning at the same time. This problem was not very popular during the previous machine learning/computer vision era of hand-crafted features and kernel-based classification models, when crafting such adversarial attacks was not as straightforward as it is today.

Adversarial attacks work by the following principle; significant changes to the feature vector that is fed to a classifier, strong enough to change the output of the decision function, may correspond to minor or even humanly imperceptible changes to the input space (e.g., to an input image). Adversarial attacks generate a perturbation capable of fooling a classification model, by exploiting a neural network backward pass to obtain gradient flow from the activations of the final (or even some intermediate) layer towards the input, using some loss function. Depending on the adversary knowledge and access to the model architecture and parameters, adversarial attacks are classified as "white-box", where an adversary has full access to the model architecture and parameters, "black-box/transferability", where the adversary employs a different model to attack the target architecture and only knows its outputs for some given input, while anything in between is considered "grey-box". Up-to-date, there is a wealth of literature describing different forms and types of adversarial attacks described in review papers [1, 2], where the reader is referred to for more details.

Methods designed to counter adversarial attacks are commonly called adversarial defenses. Unlike adversarial attacks, the working principles of recently proposed adversarial defenses vary, depending on the forms of adversarial attack they are trying to address. Perhaps the most difficult problem is to defend for the white-box attack scenario, where the goal is to eventually design a model that cannot be fooled by perturbations (in the input space) within some specific noise margin. There are many ways to achieve this goal, each having their own advantages and disadvantages; one being to modify the network architecture by e.g., leaving out some layers that increase adversarial attack susceptibility, e.g., the batch normalization layer [3]. However, such defenses cannot be applied in already deployed pretrained models. Another way of modifying the network architecture is by integrating some filtering type layers, either to the input [4] or intermediate layers [5], so that adversarial perturbations are deteriorated by some transformation. The disadvantage of such methods is that they do not explain why adversarial attacks exist in the first place. White-box adversarial attacks can also be disabled by concealing the gradient flows towards the inputs [6], with the disadvantage of leaving the model susceptible to transferability attacks. Other approaches focus on detecting adversarial attacks [7], to be applied as an input verification mechanism in parallel with the classification model [8, 9]. Finally, another line of work is to only make changes to the model's learning process, so that adversarial attack generation is not disabled, data inputs are not filtered, but the noise level required to fool the model, increases.

In the latter direction, the most prominent defenses so far are based on the so-called "adversarial training" [10, 11], which in simplified terms, involves training a deep neural network with adversarial examples of predefined noise margins, calculated implicitly or explicitly. The main disadvantages of such

approaches, is that they add significant workflow during the model training process for generating and training with these attacks and in addition, they seem to decrease the classification accuracy in clean data. Another recently proposed method [12] assumes that the problem of adversarial attack generation lies in the properties of the learned feature space. To this end, distance-based optimization criteria are employed in the the model learning process, inspired by e.g., the Nearest Centroid Classifier [13] and/or combining ideas from the triplet-loss [14] and center-loss [15] functions, so that the learned representation has decreased within-class dispersion and increased between-class separation in the feature space, which can be used together with adversarial training. We argue that their approach is on the right direction, however, the specific distance-based optimization criteria that have been proposed are sub-optimal, do not fully exploit data discrimination criteria and suffer from practical limitations, as will be discussed in detail in the subsequent sections.

This work addresses the problem of adversarial robustness in deep neural network classification from a rather traditional optimal class boundary estimation perspective. It is argued that increased model robustness in adversarial attacks can be achieved when the feature learning process is monitored by geometrically-inspired optimization criteria. The contributions of this paper are the following:

- A neural parameter set for learning hyperspherical class prototypes in the feature space is proposed.
- Three geometrically-inspired optimization criteria for the training data representation are devised, requiring them to: a) lie within their corresponding class prototype margins (classification margin loss), b) minimize their distances from the corresponding hypersphere center (prototype proximity loss), c) maximize their distance from the hypersphere centers of the remaining classes (negative proximity loss).
- These criteria are implemented with three loss functions, optimized concurrently with a standard task-objective loss function (e.g., cross-entropy loss for classification), involving no changes to the standard model inference architecture, or any kind of input manipulation.
- Our experiments show that standard classification model architectures trained using the proposed learning process: a) are more robust to adversarial attacks within some predefined noise margin, b) require stronger input perturbations in order to be fooled, c) perform comparably with softmax-only pretrained models in clean data, d) consistently outperform related adversarial defenses in both clean data classification and adversarial robustness.

The rest of the paper is structured as follows. Section 2 formally states the adversarial robustness problem from our perspective. Section 3 overviews the present state-of-the-art in Adversarial Defenses. Section 4 analytically describes the components and intuitions of the proposed method. Experiments conducted

to evaluate the robustness of the proposed method against related work in white-box and transferability adversarial attacks are presented in Section 5. Finally, conclusions are drawn in Section 6.

2. Adversarial robustness in classification

Let $\mathbf{x} \in \mathbb{R}^D$ be a data sample (e.g., an image) having a true label index y from a set $\mathcal{Y} = \{y|\ y \in \mathbb{N}, 1 \leq y \leq C\}$ that corresponds to semantic information of some discrete label set of cardinality C. The aim of a classification model is to learn to identify the semantic information in domain $\mathbb{R}^D$ by training the operation $\mathcal{X} \mapsto \mathcal{Y}$ in a representative dataset containing N samples ($\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}, |\mathcal{S}| = N,\ \mathcal{X} = \{\mathbf{x}|\mathbf{x} \in \mathbb{R}^D\}$). To date, deep neural networks are perhaps the most successful classification models.

The operation of a deep neural network with L layers can be viewed as a composition of functions applied sequentially to the input data, deriving hidden space data representations such that $\mathbf{g}_k(\cdot) : \mathbb{R}^{L_{k-1}} \mapsto \mathbb{R}^{L_k}, k = 1, \ldots, L$, (obviously $\mathbb{R}^{L_0}$ represents the input space i.e., $\mathbf{g}_1(\mathbf{x}) : \mathbb{R}^D \mapsto \mathbb{R}^{L_1}$). In classification problems, the final layer of the neural network $\mathbf{g}_L : \mathbb{R}^{L_{L-1}} \mapsto \mathbb{R}^C$ maps the hidden data representations to decision values, corresponding to each class. By assuming a softmax activation function, the network performs classification with the operation $\text{argmax}((\mathbf{g}_1 \circ \cdots \circ \mathbf{g}_L)(\mathbf{x}))$. This whole process is typically summarized by the notation $f(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the model trainable parameters. The parameters $\boldsymbol{\theta}$ are trained from the sets $\mathcal{X}, \mathcal{Y}$, by gradient back-propagation derived from a classification loss function $\mathcal{L}$, e.g., cross entropy loss, that is applied in the final layer. Finally, sample $\mathbf{x}$ is classified correctly by the neural network if $f(\mathbf{x}; \boldsymbol{\theta}) = y$. Hereafter, we will employ the notation $\mathbf{g}_k(\mathbf{x})$ to denote the intermediate hidden data representations for sample $\mathbf{x}$ in layer k and the notation $f(\mathbf{x}; \boldsymbol{\theta})$ to denote the complete model classification process.

The goal of adversarial attacks is to determine a perturbation vector $\mathbf{p} \in \mathbb{R}^D$ within a noise margin ϵ, so that to change the trained classifier decision for sample $\mathbf{x}$ i.e.:

$$\begin{aligned} &\min_{|\mathbf{p}|}: f(\mathbf{x} + \mathbf{p}; \boldsymbol{\theta}) \neq y, \\ &\text{s.t.:}\ \ |p_i| < \epsilon, \forall i = 1, \ldots, D, \end{aligned} \tag{1}$$

where p_i are elements of vector $\mathbf{p}$ and the adversarial sample $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{p}$ must remain in the same input space domain as $\mathbf{x}$ (i.e., if $\mathbf{x}$ is an image, $\tilde{\mathbf{x}}$ is also an image). White-box adversarial attacks determine the perturbation in a similar fashion to training the network, by back-propagating gradient flow from the model f towards the input, instead of updating the network parameters $\boldsymbol{\theta}$. Black-box or transferability attacks commonly work using the same principle, but without having access to the network parameters $\boldsymbol{\theta}$, e.g., they might employ a different function $\tilde{f}$, and propagate gradients towards the input with the aim of attacking f.

The robustness of the classification model f to adversarial attacks at noise margin ϵ can be estimated by the corresponding failure rate of some adversarial

attack methodology. That is, assuming that we have access to a dataset $\mathcal{S}$, we employ an adversarial attack to construct an adversarial dataset $\tilde{\mathcal{S}} = \{\tilde{\mathcal{X}}, \mathcal{Y}\}$, perturbing the data samples but keeping their original labels, and measure the classification accuracy of the model in the perturbed dataset:

$$ACC = \frac{1}{N} \sum_{i=1}^{N} acc(\tilde{\mathbf{x}}_i), \tag{2}$$

where:

$$acc(\mathbf{x}) = \begin{cases} 1, & \text{if } f(\mathbf{x}; \boldsymbol{\theta}) = y, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

A complementary way of measuring model robustness to adversarial attacks is by determining the minimum amount of perturbation required so that the accuracy of the model in the adversarial dataset drops to 0%, i.e.: $f(\tilde{\mathbf{x}}; \boldsymbol{\theta}) \neq y, \forall \tilde{\mathbf{x}}, y \in \tilde{\mathcal{S}}$, for all adversarial dataset items. To this end, one method is to employ some iterative adversarial attack e.g., projected gradient descent (PGD) [10] or the Basic Iterative Method [16] (BIM), by allowing a high enough noise margin ϵ. Thereby, we measure the average mean squared errors between the corresponding samples of the perturbed and the original set, in the input space:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2, \tag{4}$$

and higher MSE values indicate increased model robustness.

Adversarial defenses that cannot be evaluated by this metric i.e., white-box adversarial attacks fail to produce a dataset $\tilde{\mathcal{S}}$ for which the model has 0% classification accuracy, are generally considered unreliable [17, 18]. Obviously, if the gradient flow towards the input is deteriorated by some carefully crafted process, white-box adversarial attacks will be hindered as well. This property is an indication that an adversarial defense might provide a false sense of security, a phenomenon called "gradient masking/obfuscation". In fact, it has been shown that a number of adversarrial defenses proposed recently, distillation-based [19], auto-encoder based [20] or GAN-based [21, 22] ultimately rely on obfuscating the gradient flow towards the input samples, while providing limited or even no security gains against strong black-box or carefully crafted adversarial attacks [23]. Therefore, reliable adversarial defenses should have the following properties:

- White-box adversarial attacks should be more effective than transferability attacks.
- Iterative adversarial attacks should be more effective than single-step attacks.
- Increasing the noise margin ϵ should drop the model's classification in accuracy in the adversarial dataset $\tilde{\mathcal{S}}$.

3. Adversarial Defenses

The main motivation behind adversarial defenses is that currently, the noise required to fool neural network classifiers with adversarial attacks is very low, i.e., perturbed images are almost indistinguishable from the original ones to the human eye. Adversarial defense methods approach the problem from different perspectives. Following our notation, they might be grouped by the element of their focus.

One research direction is to make changes to the model architecture by e.g., replacing the mapping functions $\mathbf{g}_k$ themselves or add new ones prior to their inputs. Input filtering approaches [4, 5] fall into the category of altering the model architecture. They might introduce iterative functions $\mathbf{h}(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}^D$ that replace the inputs $\mathbf{x}$, where $\mathbf{h}(\mathbf{x})$ might be a low-pass filter or even a learnable transformation. This process has been applied to intermediate layer representations i.e., $\mathbf{h}(\cdot) : \mathbb{R}^{L_k} \mapsto \mathbb{R}^{L_k}$ as well. In a similar fashion, DnDNet [6] developed a defense layer with activation functions that conceal the gradient flow towards the input, thus disabling white-box adversarial attacks. Such approaches may be applied to already pretrained models, thus being relatively easy to implement from an engineering point of view and provide very good performance in terms of robustness. However, methods that fall under this category do not explain or address the inherent model weakenesses, that allow adversarial attacks to be so effective in the first place.

Another research direction is to maintain the same neural network architecture, only by trying to derive in different parameters i.e., $f(\mathbf{x}; \tilde{\boldsymbol{\theta}})$. The most obvious approach is to fine-tune or train the model by exploiting adversarial datasets $\tilde{\mathcal{S}}$, constructed by one or more adversarial attack methods [10, 11]. This process can be applied during training by employing an additional objective function inspired by adversarial attacks. For instance, the Fast Gradient Sign [10] objectives have been employed for adversarial training in the following manner:

$$\mathcal{L}_{AT} = \lambda \mathcal{L}_{CE}(f(\mathbf{x}; \boldsymbol{\theta}), y) + (1 - \lambda)\mathcal{L}_{CE}(f(\tilde{\mathbf{x}}; \boldsymbol{\theta}), y), \tag{5}$$

where $\tilde{\mathbf{x}}$ is an adversarial sample derived from the clean sample $\mathbf{x}$, $\mathcal{L}_{CE}$ is the standard cross entropy loss function and $0 \leq \lambda \leq 1$ is hyperparameter that controls the learning balance between clean and adversarial samples (a value equal to 0.1 has been proposed showing good results [10]). A more sophisticated variant [11] generalized the adversarial training approach by incorporating combinations of general adversarial attacks and remains up to date, the most efficient defense mechanism. Another approach treats the problem of adversarial defense from a domain adaptation point of view [24]. To this end, intermediate layer clean and adversarial data representations are projected to a subspace by employing a Graph Neural Network [25], and the divergence between them is minimized by computing an approximation of the Wasserstein distance [26]. The main disadvantages of these approaches are the introduced workflow for calculating the adversarial examples, while at the same time, model classification accuracy in clean data is negatively affected. Moreover, due to the

adversarial attack-specific nature, there is no guarantee [27] that such defenses remain effective against different types of adversarial defense.

Ultimately, adversarial defense methods that fall into the above category seem to be effective when they produce similar intermediate data representations for both clean and adversarial images belonging to the same specific class. One recently proposed adversarial defense [12] showed that incorporating distance-based optimization criteria might achieve this goal, without requiring re-training the model with adversarial examples. Inspired by the Nearest Centroid Classifier [13] and combining ideas related to the triplet-loss [14] and center-loss [15] functions, the classification model is encouraged to produce class data representations that lie close to some learned class prototype vectors, leading to increased robustness in adversarial attacks, by only having minor degradation in classification accuracy for clean samples. More specifically, one prototype vector is learned for each class in the intermediate hidden layer spaces (let $\mathbf{A}^{(k)}$ be the matrix of dimensions $\mathbb{R}^{L_k \times C}$ containing the prototype vectors) and the distances between the class data representations and the prototype vectors is minimized. To this end, the following optimization procedure is employed:

$$\mathcal{L} = \lambda \mathcal{L}_{CE}(f(\mathbf{x};\boldsymbol{\theta}), y) + (1-\lambda) \sum_{k \in \mathcal{K}} \mathcal{L}_{PCL}(\mathbf{g}_k(\mathbf{x};\boldsymbol{\theta}), \mathbf{A}^{(k)}, y), \tag{6}$$

where $\mathcal{L}_{PCL}$ is the so-called prototype conformity loss that incroporates the criteria of the NCC classifier for the intermediate layers denoted in set $\mathcal{K}$. More specifically, assuming sample $\mathbf{x}$ belongs to class y, $\mathcal{L}_{PCL}$ for this item represented in this $k-$th layer is given by:

$$\mathcal{L}_{PCL} = \|\mathbf{g}_k(\mathbf{x};\boldsymbol{\theta}) - \mathbf{a}_y\|_2^2 - \frac{1}{C-1} \sum_{j \neq y} \left(\|\mathbf{g}_k(\mathbf{x};\boldsymbol{\theta}) - \mathbf{a}_{kj}\|_2^2 + \|\mathbf{a}_{kj} - \mathbf{a}_{ky}\|_2^2 \right), \tag{7}$$

where $\mathbf{a}_{kj}$ is the prototype vector of class j in the hidden data space derived in the $k-$th layer. Thus, the representations of items belonging to class y are encouraged to lie closer to prototype vectors of their own class and lie further away from other prototype vectors. Such methods might also apply an adversarial training step to further increase model robustness against specific type of adversarial attacks. Our approach builds on and extends this research direction.

4. Hypersherical Class Prototype loss

The proposed method is a modification of the standard classification model learning process for increasing its robustness to adversarial attacks. In order to achieve this goal, geometrically-inspired optimization criteria are introduced to the model optimization phase, by employing hypersherical class prototypes in the deep representation spaces, defined by learnable centers and radii. The optimization phase focuses on two goals; activations of data belonging to the same class are enclosed by a bounding hypersphere prototype, while at the same

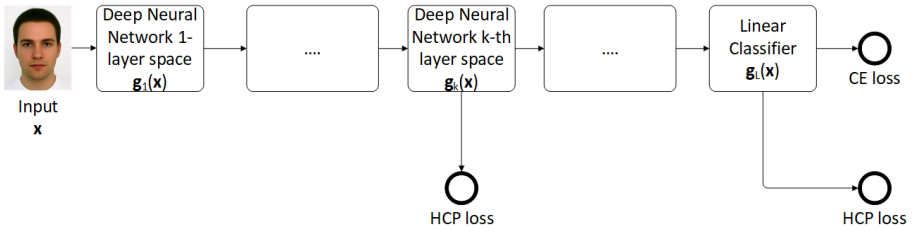


Figure 1: Conceptual diagram of the proposed M-SVDD-D. HCP loss may be applied in intermeddiate layers as well as the final layer, together with task-specific loss terms.

time, distances between data activations and from irrelevant hypesphere class prototypes, are maximized. These criteria are applied in intermediate as well as the final layer data activations, in the form of loss functions, as shown in Figure 1. The optimization procedure can be employed to untrained models, or can be used to pretrained models, by employing a dataset $\mathcal{S}$.

During model deployment phase, the derived robust model remains adversarial attack agnostic in principle. Nevertheless, the proposed procedure does not prohibit the application of other type of adversarial defenses (e.g., adversarial training) in conjunction with the proposed method.

The first objective of the proposed learning process is to derive tight class boundaries in the deep representation space. Based on our previous experience in one-class classification [28, 29], we consider that the optimal tight class boundaries can be determined by enclosing class data representations of items belonging each class with hyperspheres, and thereby minimize the respective volumes. In fact, one-class classification methods are even more relevant in the adversarial robustness problem, if we consider clean data samples as "target" data and adversarial samples as "anomalies". The optimization problem is very well defined in the traditional Support Vector Data Description method [30]. Extensions of the vanilla method include training from negative [31] or unlabeled examples [32]. Finally, one-class classification criteria inspired by this method have been successfully applied in the deep learning case [8, 9], as well. In the following, we show how one-class classification criteria can now be applied in the multi-class classification case.

Let $\mathcal{K}$ be the set of layers on which the proposed objectives will be applied to. Formally, the proposed method aims to learn hypersherical prototypes in the k-th layer defined by the prototype matrices $\mathbf{A}^{(k)} \in \mathbb{R}^{C \times L_k}$, and radii $\mathbf{R}^{|\mathcal{K}| \times C}$ that will act as one-class classifiers, verifying data sample activations belonging to the j-th class. To this end, the relevant optimization terms for each sample $\mathbf{x}_i$ are the following:

$$\begin{aligned} \min_{\mathbf{R},\boldsymbol{\Xi},\boldsymbol{A}^{(k)}} \quad & \sum_{k\in\mathcal{K}}\sum_{j=1}^{C} r_{kj}^2 + \sum_{k\in\mathcal{K}} c_k \sum_{i=1}^{N} \xi_{ki} \\ \text{s.t.:} \quad & \sum_{k\in\mathcal{K}}\sum_{j=1}^{C} \left(-y_{ij}\left(r_{kj}^2 - \|\mathbf{g}_k(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{a}_j^{(k)}\|^2\right) \leq \xi_{ki}\right), \\ & \xi_{ki} \geq 0 \end{aligned} \tag{8}$$

where $\mathbf{Y} \in \{-1, 1\}^{N\times C}$ is a slight different definition of the one-hot labels ($y_{ij} = 1$ if sample $\mathbf{x}_i$ belongs to class j, $y_{ij} = -1$, otherwise), ξ_{ki} are the slack variables and $c_k \geq 0$ is a hyperparameter that allows some training error (i.e., soft margin formulation). A value of $c_k = 0$ denotes the hard margin formulation, allowing no training error. Since both the feature vectors and the prototype vectors are trainable, we consider that a value of $c_k = 0$ is a valid choice; we are not worried about overfitting the class centers to the training data.

The constraints of the above optimization problem can be optimized by applying the following hinge loss function in every layer selected in $\mathcal{K}$:

$$\mathcal{L}_M = \sum_{j}^{C} \max\left(c_k, -y_{ij}\left(r_{kj}^2 - \|\mathbf{g}_k(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{a}_j^{(k)}\|^2\right)\right). \tag{9}$$

As can be observed, if $\|\mathbf{g}_k(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{a}_j^{(k)}\|^2 < r_{kj}^2$, then the data representation of $\mathbf{x}_i$ falls inside the j-th class hypersphere, while otherwise, the item lies outside the j-th hypersphere. The loss value is equal to the distance of the data representations in the feature space from the closest hypersphere outer boundary, and $\mathcal{L}_M > 0$ if and only if the one-class classifier decision function misclassifies $\mathbf{x}_i$.

Loss function (9) introduces the one-class classification criteria to the multi-class classification case. The compactness of class representations is proportional to the learned value of the corresponding radius r_{kj}. An obvious drawback of this optimization option is that very little or almost no loss is produced for marginal data items (lying very close to the hypersphere boundaries, either inside when they belong to the class or outside, if they do not). Thus, this optimization term is not enough for achieving adversarial robustness.

In order to achieve robustness, we would require data representations belonging to some specific class to lie very close with each other, without minimizing the volume of the enclosing hypersphere. Instead, we would wish that the learned hypershere to capture enough volume so that potential adversarial examples will still be enclosed. If the hypersphere volume is high enough and the classification criteria defined by loss term $\mathcal{L}_M$ are met, then we can also assume that the between class distances will also be maximized as well, by implication.

To this end, we introduce the following loss function:

$$\mathcal{L}_P = \sum_{j}^{C} \frac{y_{ij}+1}{2} \frac{\|\mathbf{g}_k(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{a}_j^{(k)}\|^2}{r_{kj}^2 + e}, \tag{10}$$

where e is a regularization hyperparameter that avoids divisions with zero (typically set to very low values, e.g., $e = 0.001$). The loss value is non-zero only for data representations belonging to the corresponding hyperspherical prototype (positive items). It is equal to the distance from the center, divided by the hypersphere radius. Therefore, it complements $\mathcal{L}_M$ by enforcing even tighter class representations, while at the same time, it allows for the radius to grow in order to capture more space, thus maximizing the volume of the hypersphere. In every case, the term $\mathcal{L}_P$ does not consider negative data items, at all. In order to obtained balanced optimization terms, we require to achieve opposing goals for the negative items: i.e., maximize their distance between all non corresponding hyperspheres, proportionally to their radii. More formally, the following loss function is proposed:

$$\mathcal{L}_{NP} = -\sum_{j}^{C} \frac{y_{ij}-1}{2} \frac{r_{kj}^2}{\|\mathbf{g}_k(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{a}_j^{(k)}\|^2 + e}. \tag{11}$$

The loss value is non-zero only for items not belonging to the corresponding hypersphere prototype, and enforces increasing their distance from the prototypes of other classes.

Finally, the proposed Hypersherical Class Prototype (HCP) loss function includes the combination of the constraints of (9), (10) and (11), as follows:

$$\mathcal{L}_{HCP} = \mathcal{L}_M + \mathcal{L}_P + \mathcal{L}_{NP}, \tag{12}$$

where relevant weighting hyperparameters could be considered as well i.e., $\mathcal{L}_{HCP} = \mu_1\mathcal{L}_M + \mu_2\mathcal{L}_P + \mu_3\mathcal{L}_{NP}$, for adjusting the contribution of each term to the overall loss. Their tuning using traditional grid search is possible but not advised in these types of problems, since it would severely increase the complexity of the optimization procedure while on the other hand, it has been shown that automatic hyperparameter tuning is more effective [33] in classification problems. In the experiments presented in this work, weighting parameters were not employed (i.e., $\mu_1 = \mu_2 = \mu_3 = 1$).

Why apply the proposed loss function in intermediate layers and not the final layer only or the first? Assuming the proposed loss function is applied in the input space for an element $\mathbf{x} \in \mathbb{R}^D$ belonging to class c, the model should be able to tolerate data perturbations of noise margins equal to the corresponding radius i.e,. $|p_i| < r_c, i = 1, \ldots, D$, by definition. However, if this function was applied in the input space only, then the whole deep learning process is not utilized, thus the classification accuracy of the model would be sub-optimal. On the other hand, if we apply the proposed term only in the final layer, the full deep learning process is utilized, though the connection between the radius

and the perturbation values is lost. Therefore, where to apply the proposed loss terms is not trivial and can be viewed as a trade-off between classification performance and robustness. The optimal place to apply the proposed loss term is definitely in the final layer, together with some intermediate layers as close to the input space as possible, such that the classification accuracy of the model is not hindered.

5. Experiments

In this section, we describe the experiments conducted in order to evaluate the performance of the proposed optimization scheme. In all our experiments, we have employed the ResNet-101 [34] architecture, which is typically employed in image classification problems and produces close to state-of-the-art results. In terms of datasets, we have employed the publicly available CIFAR-10, CIFAR-100 [35] and SVHN [36] datasets belonging to 10, 100 and 10 classes, respectively. Unless otherwise stated, the ResNet model was pretrained in Imagenet dataset [37] and fine-tuned for 400 epochs at the task at hand, using different loss functions and optimization options, according to the ones proposed by related adversarial defense methods. For comparisons reasons, we have employed the proposed method (HCP), along with the Vanilla softmax optimization function (SM), the recently proposed PCL adversarial defense [12] (PCL), the closely related center loss function [15] (CL) and Adversarial Training [11] (AT). Hereafter, we will refer to the compared methods by their respective acronyms. Wherever publicly available, we have employed the source code from the paper authors (i.e., PCL, CL). We implemented AT using the code from [12]. The hyperparameter settings of the proposed method include the following: the regularization parameter $e = 0.001$, learning rate for the model $lr = 0.01$ and learning rate for the HCP parameters $lr_{HCP} = 0.05$, multiplied by 0.1 at epochs $142, 230$ and 360, respectively. All prototype vectors $\boldsymbol{A}$ were initiated with random values, while all radii r_{kj} were initiated with ones. The hyperparameters settings for the remainder methods were set according to the details found in the publicly available codes provided by the authors and the referenced papers. The loss functions for HCP, PCL and CL and were applied in the same ResNet layers (i.e., the 256-dimensional layer-3 and 1024-dimensional layer-4), similar with [12]. All experiments were implemented in Pytorch 1.6.0.

In Subsection 5.1, we report the classification accuracy of the different optimization options, in clean data. In Subsection 5.2, we describe our experiments for evaluating the robustness of the different models in adversarial attacks. In Subsection 5.3, we qualitatively analyze the reasons why the proposed method shows more effectiveness against adversarial attacks when compared to other optimization options. Finally, in Subsection 5.4, we conduct an ablation study the contribution of each of the proposed loss functions.

5.1. Classification accuracy and training performance

In our first set of experiments, we report the classification accuracy of the different methods in the employed datasets. Two different intitialization modes

Table 1: Classification accuracy of the employed architectures

Method/Dataset	**CIFAR-10**	**CIFAR-100**	**SVHN**
SM (FS)	93.36	**74.04**	96.23
AT (SM-PGD) [11]	81.49	54.01	91.46
CL (SM) [15]	93.77	69.75	95.90
PCL (SM) [12]	92.30	68.19	95.37
HCP - (SM)	93.31	72.83	95.85
HCP - (FS)	**94.69**	72.56	**96.38**

(SM) the architecture was pretrained with softmax-only for 200 epochs.
(PGD) training by employing the PDG attack.
(FS) training from scratch.

for the competed methods were employed, i.e., a softmax-only pretraining step for 200 epochs, denoted by (SM), and alternatively, without any pretraining step, denoted by (FS). In both cases, the models were trained for 400 epochs. For instance, PCL (SM) denotes 200 epochs of pretraining with softmax only, and 400 epochs training with the PCL method. Table 1 reports the accuracy obtained in the employed datasets by the different methods. As can be observed, the proposed method remains very competitive against the vanilla SM optimization function in clean data and in the CIFAR-10 and SVHN cases, it even provides better accuracy, when applied from scratch. In all cases, the proposed method outperformed the PCL defense, whether pre-trained or trained from scratch.

Here, it should be noted that in our experiments, we did not manage to obtain consistent good results for AT, CL and PCL when trained from scratch, thus their results are not reported. In contrast, the proposed method is very effective in the classification problem, in both cases. We will make an attempt to explain the reasons behind this finding. For the AT case for instance, employing the adversarial training step without having a well initialized model first, hinders the ability of the model to converge to good values, as the adversarial examples and the clean examples seem to produce gradients in competing directions and slow down the learning process for the clean data. This may be solved by exploring different hypeparameters λ for the learning rates for the clean data and adversarial data, but in every case, this was not needed when the model was initialized with softmax only training. For the CL and PCL cases, we were not able to obtain consistent results in the FS mode. This can be explained by the initialization options (we employed random initialization for the class prototypes). Class prototypes and network parameters are optimized at the same time. We examine the case where the initial prototype vectors happen to lie very close to each other, while the network parameters are untrained, thus are not discriminant. Essentially, this optimization term essentially forces data belonging to the same class to be mapped very close to the corresponding prototype vectors, and if the prototype vectors happen to lie are very close

to each other, the optimization term forces the network representations to lie very close for all training data, by implication. This hinders the ability of the network parameters to be trained to the most discriminant direction. Nevertheless, initializing the model with a softmax function solves this issue, since the gradient values returned to the prototype centers, are stronger than the ones returned to the network parameters. In contrast with the above, the proposed method clearly enforces classification criteria by the function (9) along with the distance-based criteria, thus updating the prototype centers and the model parameters with the proposed loss term is more compatible; it allows the model to reach optimality regardless of the initialization mode and does not require such an extensive hyperparameter search.

5.2. Robustness in white-box attacks

Table 2: Robustness in white-box attacks

Method/Dataset	**CIFAR-10** ($\epsilon = 0.03$)			**CIFAR-100** ($\epsilon = 0.01$)			**SVHN** ($\epsilon = 0.03$)		
Attack type	FGSM	BIM	MIM	FGSM	BIM	MIM	FGSM	BIM	MIM
SM (FS)	24.73	00.00	00.06	18.25	04.60	06.29	48.44	02.50	05.90
AT (SM-PGD) [11]	53.74	48.66	49.00	41.78	**41.21**	**41.40**	**90.23**	**77.33**	73.12
CL (SM) [15]	57.74	41.07	41.34	38.98	27.68	28.38	77.72	64.58	64.53
PCL (SM) [12]	50.62	27.23	27.63	38.92	29.08	29.09	68.76	40.95	41.17
HCP- (SM)	60.28	55.51	57.57	**48.28**	40.36	41.37	76.37	73.38	74.10
HCP – (FS)	**72.26**	**63.51**	**64.76**	35.87	15.40	18.3	80.83	76.10	**76.25**

In our second set of experiments, we report the Robustness of the models to three different types of adversarial attacks, namely to the Fast Gradient Sign Method (FGSM) [10], Basic Iterative Method (BIM) [16] and Momentum Iterative Method (MIM) [38]. We have employed these adversarial attacks to generate one adversarial dataset $\tilde{\mathcal{S}}$ for each combination of attack/competing method, respectively. In order to measure robustness, we have exploited the accuracy and MSE metrics as defined in Section 2 in equations (2) and (4), respectively.

Table 3: MSE values and standard deviations of white box attacks in the competing methods

Method/Dataset	**CIFAR-10**	**CIFAR-100**	**SVHN**
AT (SM-PGD) [11]	0.07 ± 0.028	0.08 ± 0.04	0.16 ± 0.02
CL (SM) [15]	0.10 ± 0.035	0.05 ± 0.04	0.21 ± 0.04
PCL (SM) [12]	0.09 ± 0.028	0.04 ± 0.03	0.10 ± 0.06
HCP - (SM)	0.19 ± 0.052	$\mathbf{0.10} \pm 0.06$	$\mathbf{0.26} \pm 0.04$
HCP - (FS)	$\mathbf{0.26 \pm 0.037}$	0.02 ± 0.01	0.25 ± 0.04

All architectures reached 0% accuracy by using a value of $\epsilon = 5$.

Table 2 summarizes the obtained results. Regarding the introduced noise, we have employed a value of $\epsilon = 0.03$ in the $10-$class datasets (CIFAR-10, SVHN) and a value of $\epsilon = 0.01$ in the 100-class CIFAR-100 dataset. As can be seen, the proposed method outperformed the competition in CIFAR-10 dataset, was very close with AT in CIFAR-100 when using the SM mode, and was also

very comparable in SVHN dataset, in both FS and SM modes. The only case where the proposed method did not work as well as the competition was on the CIFAR-100 dataset, in the FS mode. This can be explained by the fact that the model had 200 epochs less to train in total when compared to the competition, which although did not make a difference in the 10-class datasets, it presented itself in the 100-class case. As can be seen from the classification accuracies in Table 1, the model was able to separate the classes well enough, but seems to have failed to reach sufficient robustness levels. Nevertheless, this setback was resolved when a softmax pre-training step was applied. In fact, the proposed method in the SM mode greatly outperformed the distance-based methods (PCL, CL) in every case.

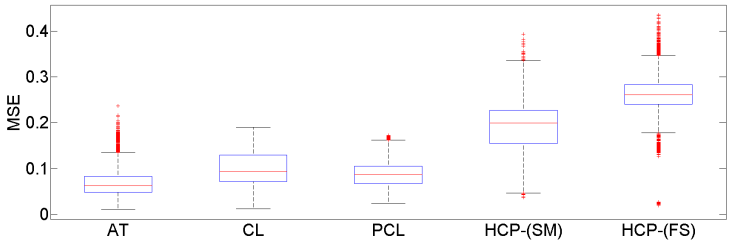
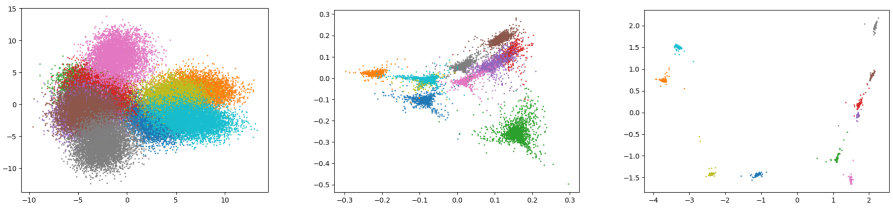


Figure 2: Noise distribution of each method in CIFAR-10 Dataset. The mean values and standard deviations are reported in Table 3. As can be observed, employing the MIM adversarial attack to the architecture trained using the proposed method, produces more noisy images when compared to other training methods, using the same settings ($\epsilon = 5$, number of iterations $= 100$).

These obtained difference between the models can be observed by analysing the derived adversarial datasets, from a statistical point of view. To this this end, MSE values as well as the corresponding standard deviations in the respective adversarial datasets, as drawn Table 3. In order to derive these data, we have employed the MIM white-box iterative adversarial attack and allowed very high perturbation levels (i.e., $\epsilon = 5$) for 100 iterations so that all competed models reached 0% accuracy. Thereby, we measured the introduced perturbation by comparing the MSE values between the obtained adversarial dataset items $\tilde{\mathcal{X}}$ with the original ones $\mathcal{X}$. The datasets derived by the adversarial attacks on architectures trained using the proposed method, contained more noise than the competition, in almost every case, expect the CIFAR-100 FS mode. This is aligned with the reported results in Table 2, indicating that the proposed adversarial defense method is more effective than the competition in 10-class classification problems, and requires pre-training to be more competitive for the 100-classes case (CIFAR-100). For demonstration reasons, we also plot the obtained noise distribution for each method in CIFAR-10 dataset, using a box plot (Figure 2).



(a) SM (b) PCL (c) HCP

Figure 3: PCA plots for the datamatrices of the CIFAR-10 dataset by extracting the final layer of SM, PCL and HCP trained models. The proposed HCP model exhibits more discriminant power in the first two dimensions of the eigenvector basis. That is, the model was able to encode class information in the strongest eigenvalues, which explains why the proposed model requires stronger perturbations in order to be fooled by adversarial attacks. Best viewed in color.

5.3. Qualitative and statistic evaluation

In an attempt to explain the obtained results, we have plotted the obtained network representations in the final layer ($1024-$dimensional) for the vanilla SM model, the PCL-trained model and the proposed HCP-trained one, in the CIFAR-10 dataset. In order to visualize in 2-dimensions, we have employed Principal Component Analysis (PCA) using the scikit-learn 0.24.2 tool [39] in the obtained datamatrices, and kept the 2 eigenvectors corresponding to the largest eigenvalues. As depicted in Figure 3, the proposed HCP model exhibits more discriminant power in this basis, when compared with the SM and PCL-trained models. That is, the HCP-trained model was able to encode class information in the strongest eigenvalues, which explains why the proposed model requires stronger perturbations in order to be fooled by adversarial attacks.

In addition, we conducted a statistical analysis to determine whether the observed performance differences between competing methods are statistically significant, or not. To this end, we have employed the Friedman test with the Nemenyi post-hoc procedure [40], for testing the null hypotheses that all competing methods perform the same. To this end, we have ranked AT, CL, PCL and HCP methods according to their performances in each experiment. Using 15 different experiments and 4 different competing methods, the degrees of freedom are 42. The obtained mean ranks were 2.40, 2.73, 3.60 and 1.26, respectively. The statistic $\chi^2 = 25.16$ with a critical value of 6.2514. Thus, the critical distance was 1.08. According to this metric, the proposed method performs significantly different (better) than the competing methods.

5.4. Ablation study

Finally, we conducted an ablation to study in order to determine the effect of each of the proposed loss terms in the optimization process of the model. Table 4, we have employed different combinations of the proposed $\mathcal{L}_{HCP}$ loss in the softmax pretrained mode, due to the performance instability issues of some variants in the FS mode for the reasons explained in Subsection 5.1. The

difference in classification accuracy in all terms seems is very small for every case, very comparable to the standard SM loss, that indicates that none of the introduced terms hinders the ability of the model to reach close to optimal results, in the classification problem. On the robustness side however, the most stable performance is obtained when all three of proposed terms are employed at the same time. In addition, we also report the performance of the model when AT is employed together with the proposed loss functions. As can be seen, the combination of the proposed method with AT leads to better classification accuracy when compared with the standard AT method and in most cases, increased robustness as well.

Table 4: Classification accuracy and robustness of different variants of the proposed method

	Classification Accuracy			**Robustness (BIM)**		
$\mathcal{L}_{HCP}$/Dataset	CIFAR-10	CIFAR-100	SVHN	CIFAR-10	CIFAR-100	SVHN
$\mathcal{L}_M + \mathcal{L}_P$	92.65	73.02	95.95	45.03	19.30	69.52
$\mathcal{L}_M + \mathcal{L}_{NP}$	92.85	**73.18**	95.89	**55.88**	19.53	**78.04**
$\mathcal{L}_P + \mathcal{L}_{NP}$	**93.53**	72.93	**95.91**	52.70	39.97	69.64
$\mathcal{L}_M + \mathcal{L}_P + \mathcal{L}_{NP}$	93.31	72.83	95.85	55.51	**40.36**	73.38
AT [11]	81.49	54.01	91.46	48.66	41.21	77.73
$\mathcal{L}_{HCP}$ + AT	81.31	56.08	93.07	50.45	**43.85**	70.19

6. Conclusion

This work presented a deep neural network learning process that incorporates geometrically-inspired optimization criteria in classification problems, by introducing the concept of hyperspherically-shaped class prototypes. Three concurrent optimization criteria for the intermediate hidden layer training data activations were devised and were implemented with corresponding loss functions to be optimized concurrently with the standard task-objective loss function (e.g., cross-entropy loss for classification), involving no changes to the standard model inference architecture, or any kind of input manipulation. It was shown that incorporating these criteria in the model training processes increases their robustness to adversarial attacks by a large extent, without hurting their classification performance in clean data. These attributes were showcased in image-classification problems.

More specifically, in was shown that the models trained with the proposed learning process were significantly more robust to 3 different types of white-box adversarial perturbations, both in terms of introduced noised and classification accuracy. Especially for the case of 10-class classification problems, unlike other state-of-the-art adversarial robustness learning schemes, the proposed method does not require a neural architecture pretraining step with clean examples to provide acceptable results. However, this was not the case in the 100-class classification problem, where potential adopters are advised to employ neural network pretraining steps. The findings of this work suggest important potential benefits in image classification problems and related applications where adver-

sarial robustness is very much needed, e.g., biometric data identification, fault detection, medical image classification etc.

This work could be extended in the following research directions. First, it is straightforward to be implemented in modalities other than image (e.g., sound, natural language processing etc). Next, the proposed optimization criteria could be generalized to other geometrical shapes, e.g., hyper-ellipsoids or cubes. The proposed methodology and findings can be employed towards achieving a deeper understanding of adversarial robustness in general and can be linked with works conducted in certifiable adversarial robustness. Another research direction is to employ the hypersherical prototype centers for designing novel adversarial attacks by exploiting e.g., their distance with the intermediate-layer data feature representations. Finally, this learning process could also be examined in retrieval problems, or combinations of classification-regression problems (e.g., face detection).

Acknowledgment

This work has received funding from the European Union's European Union Horizon 2020 research and innovation programme under grant agreement 951911 (AI4Media). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] X. Yuan, P. He, Q. Zhu, X. Li, Adversarial examples: Attacks and defenses for deep learning, IEEE Transactions on Neural Networks and Learning Systems 30 (9) (2019) 2805–2824. doi:10.1109/TNNLS.2018.2886017.

[2] K. Ren, T. Zheng, Z. Qin, X. Liu, Adversarial attacks and defenses in deep learning, Engineering 6 (3) (2020) 346–360. doi:https://doi.org/10.1016/j.eng.2019.12.012.
URL https://www.sciencedirect.com/science/article/pii/S209580991930503X

[3] A. Galloway, A. Golubeva, T. Tanay, M. Moussa, G. W. Taylor, Batch normalization is a cause of adversarial vulnerability, arXiv preprint arXiv:1905.02161.

[4] C. Guo, M. Rana, M. Cisse, L. van der Maaten, Countering adversarial images using input transformations, in: International Conference on Learning Representations, 2018.

[5] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, K. He, Feature denoising for improving adversarial robustness, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 501–509.

[6] A. Goel, A. Agarwal, M. Vatsa, R. Singh, N. K. Ratha, Dndnet: Reconfiguring cnn for adversarial robustness, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 22–23.

[7] G. Cohen, G. Sapiro, R. Giryes, Detecting adversarial samples using influence functions and nearest neighbors, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14453–14462.

[8] V. Mygdalis, A. Tefas, I. Pitas, K-anonymity inspired adversarial attack and multiple one-class classification defense, Neural Networks 124 (2020) 296–307.

[9] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, M. Kloft, Deep one-class classification, in: International conference on machine learning, PMLR, 2018, pp. 4393–4402.

[10] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018.

[12] A. Mustafa, S. H. Khan, M. Hayat, R. Goecke, J. Shen, L. Shao, Deeply supervised discriminative learning for adversarial defense, IEEE transactions on pattern analysis and machine intelligence.

[13] T. Mensink, J. Verbeek, F. Perronnin, G. Csurka, Distance-based image classification: Generalizing to new classes at near-zero cost, IEEE transactions on pattern analysis and machine intelligence 35 (11) (2013) 2624–2637.

[14] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.

[15] Y. Wen, K. Zhang, Z. Li, Y. Qiao, A discriminative feature learning approach for deep face recognition, in: European conference on computer vision, Springer, 2016, pp. 499–515.

[16] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial examples in the physical world, arXiv preprint arXiv:1607.02533.

[17] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: International Conference on Machine Learning, PMLR, 2018, pp. 274–283.

[18] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, A. Kurakin, On evaluating adversarial robustness, arXiv preprint arXiv:1902.06705.

[19] N. Papernot, P. McDaniel, I. Goodfellow, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277.

[20] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1778–1787.

[21] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: Protecting classifiers against adversarial attacks using generative models, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=BkJ3ibb0-`

[22] G. S. Dhillon, K. Azizzadenesheli, J. D. Bernstein, J. Kossaifi, A. Khanna, Z. C. Lipton, A. Anandkumar, Stochastic activation pruning for robust adversarial defense, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=H1uR4GZRZ`

[23] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 39–57.

[24] X. Zhang, J. Wang, T. Wang, R. Jiang, J. Xu, L. Zhao, Robust feature learning for adversarial defense via hierarchical feature alignment, Information Sciences 560 (2021) 256–270.

[25] Y. Chen, M. Rohrbach, Z. Yan, Y. Shuicheng, J. Feng, Y. Kalantidis, Graph-based global reasoning networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 433–442.

[26] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Advances in neural information processing systems 26 (2013) 2292–2300.

[27] N. Jovanović, M. Balunović, M. Baader, M. Vechev, Certified defenses: Why tighter relaxations may hurt training?, arXiv preprint arXiv:2102.06700.

[28] V. Mygdalis, A. Iosifidis, A. Tefas, I. Pitas, Graph embedded one-class classifiers for media data classification, Pattern Recognition 60 (2016) 585–595.

[29] V. Mygdalis, A. Iosifidis, A. Tefas, I. Pitas, Kernel subclass support vector description for face and human action recognition, in: 2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE), IEEE, 2016, pp. 1–5.

[30] D. M. Tax, R. P. Duin, Support Vector Data Description, Machine learning 54 (1) (2004) 45–66.

[31] M. Wu, J. Ye, A small sphere and large margin approach for novelty detection using training data with outliers, IEEE transactions on pattern analysis and machine intelligence 31 (11) (2009) 2088–2092.

[32] V. Mygdalis, A. Iosifidis, A. Tefas, I. Pitas, Semi-supervised subclass support vector data description for image and video classification, Neurocomputing 278 (2018) 51–61.

[33] W. Xie, W. Chen, L. Shen, J. Duan, M. Yang, Surrogate network-based sparseness hyper-parameter optimization for deep expression recognition, Pattern Recognition 111 (2021) 107701.

[34] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[35] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Master's thesis, Department of Computer Science, University of Toronto.

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, In: Neural Information Processing Systems (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[38] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 9185–9193.

[39] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[40] A. Ulaş, O. T. Yıldız, E. Alpaydın, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recognition 45 (4) (2012) 1772–1781.