Improving Adversarial Robustness of Few-shot Learning with Contrastive Learning and Hypersphere Embedding

Madhushan Buwaneswaran*, Tehara Fonseka*, Apurva Narayan*†, and Katarina Grolinger*
* Department of Electrical and Computer Engineering, Western University, London, Ontario, Canada
† Department of Computer Science, Western University, London, Ontario, Canada
{mbuwanes, tfonsek, apurva.narayan, kgroling}@uwo.ca

***Abstract*—Few-shot image classification (FSIC) is a computer vision task from the few-shot learning (FSL) category in which the model learns to classify images using only a few training samples. It has been demonstrated that even neural networks trained on large scale datasets are vulnerable to adversarial samples. This vulnerability is magnified in FSIC due to the low volume of training data. This paper proposes the use of hypersphere embedding and supervised contrastive learning to improve the adversarial robustness of representation learning-based FSIC. Contrastive learning contributes through its ability to bring together similar samples while pushing away dissimilar ones. On the other hand, hypersphere embedding has been successful in the representation learning tasks by restricting the embeddings to a hypersphere manifold. The proposed approach was evaluated on both 5-shot and 1-shot learning using two standard FSL networks and the standard Mini-ImageNet benchmark dataset. The evaluation shows that supervised contrastive training provides inherent adversarial robustness to the FSIC model while hypersphere embedding with cosine distance metrics improves the accuracy of the FSIC model and, when used in conjunction with an adversarial defense mechanism, boosts the adversarial performance.**

***Index Terms*—Few-shot learning, Adversarial defense, Contrastive Learning, Hypersphere embedding**

I. Introduction

Neural networks have become the de-facto standard for many computer vision tasks due to their state-of-the-art performance. Training a deep neural network requires abundant data [1], but such data are often not readily available, and, at times, obtaining large labeled data is costly. This gave rise to Few-Shot Learning (FSL), a sub domain of machine learning that tries to develop algorithms that are able to learn from fewer training samples. FSL is predominantly present in the computer vision domain, specifically in image classification where Few-Shot Image Classification (FISC) aims to train an image classifier with only a few training samples.

FSL approaches can be categorized into two main classes: representation learning approaches and meta-learning approaches. Representation learning approaches learn a latent representation in embedding hyperspace where different classes are expected to be separable [2]. On the other hand, meta-learning approaches train a meta model with multiple FSL tasks and then fine tune that meta model to adapt it for a new FSL task at hand [3]. Both, representation and meta-learning approaches, use a neural network as the base learner.

Several studies have shown that neural networks are vulnerable to adversarial samples [4], [5]. Adversarial samples are carefully crafted input data instances created to fool a model and cause it to output a wrong prediction; however, these samples look normal to the human eye due to the imperceptible nature of the perturbations. Neural networks trained under FSL settings are more prone to adversarial attacks due to the reduced number of training samples [6] which makes FSL models an easy target for adversarial attacks. Hence, it is essential to investigate and design adversarially robust FSL models that are robust against adversarial attacks while maintaining the ability to learn from a few samples. Adversarially robust FSIC is a very challenging area since it combines two challenging tasks, namely training a model using less data and making that model robust to adversarial attacks.

A few studies have been conducted in the domain of adversarially robust FSIC [6]–[8]. Most of them are based on the meta-learning [7], [9], except for Dong et al. [8] who focused on the representation learning class of algorithms. While these works have made significant progress towards the adversarial robustness of FSIC models, given the hard nature of the problem, further advancements are needed to ensure the adversarial robustness of FSIC models.

The concept of representation learning is not restricted to FSIC; it can be seen in other domains such as face recognition [9] and person re-identification [10]. For these tasks, prior works have shown that constraining the representation embedding space to a hypersphere manifold can improve the performance [11]–[13]. Furthermore, Pang et al. [14] have demonstrated that in image classification tasks hypersphere embeddings (HE) can boost the adversarial robustness when used together with adversarial training.

Another technique widely used in representation learning is contrastive learning (CL) [15]. The goal in CL is to learn a representation of data such that similar samples are close together in the embedding space, while dissimilar samples are far apart. More specifically, in supervised CL [16], the goal is to ensure that the intra-class distance is lower than the inter-

This work was supported in part by NSERC under grant RGPIN-2018-06222 (K. Grolinger) and in part by the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute (M. Buwaneswaran).

2023 International Conference on Machine Learning and Applications (ICMLA) DOI 10.1109/ICMLA58977.2023.00041
https://ieeexplore.ieee.org/document/10459996

class distance. By training with a target of increasing inter-class distances, CL is able to create a buffer zone between class boundaries in the representation space. Intuitively, this buffer zone could make it more difficult to create adversarial samples that can crossover to the other side.

Therefore, given the success of HE and CL in other representation learning tasks, this paper proposes HE latent space mapping and supervised CL for representation learning-based FSIC, with the aim of improving the adversarial robustness of the FSIC models. This study answers the following questions: Does CL or HE provide adversarial robustness to FSIC without requiring an explicit adversarial defense mechanism? Does CL or HE boost the performance of adversarial defense techniques in FSIC? Can the use of HE improve the (natural) accuracy of the representation learning FSIC techniques? The main contributions are i) integrating CL and HE into FSIC ii) showing that CL increases the FSIC accuracy and provides adversarial robustness, and iii) demonstrating that HE improves FSIC when used with adversarial training.

This paper is organized as follows: Section II provides background information, Section III reviews related works, Section IV presents the proposed techniques, and Section V discusses results. Finally, Section VI concludes the paper.

II. Background

This section presents an overview of a few-shot learning, adversarial sample generation, and adversarial training.

A. Few-Shot Learning

Few-shot image classification (FSIC) is a task of training an image classifier using only a few training samples per class, typically 1,5 or 10. The data for the FSL model consists of two disjoint datasets D_{train} and D_{test} containing non-overlapping sets of classes with K_{train} and K_{test} classes respectively. The model is trained by simulating multiple few-shot tasks. First, from D_{train} dataset, k classes are sampled, and then from each of these k classes, n sample (support) images are randomly selected to train the model and q query images to calculate the loss. One round of training and calculating the loss is considered one training *episode* of a *k-way n-shot* classification task. The model is trained on multiple such episodes, each time randomly sampling k new classes and n samples for each class. During the evaluation, k classes are sampled from D_{test}, and n support images are sampled for each of these classes – These data are used to "fine-tune" the model. Next, the accuracy of this model is calculated using q query images belonging to the selected k classes. The fine-tuning together with accuracy calculation is known as a test episode. The final reported accuracy is the accuracy over multiple test episodes.

As already mentioned, meta-learning and representation learning are the main categories of FSL approaches. Meta-learning approaches train a meta model with multiple few-shot learning tasks and then adapt that meta model for the new FSL task: Model-Agnostic Meta-Learning (MAML) [3] is a widely adapted approach from this category. In contrast, representation learning approaches learn a latent embedding space in which embeddings of a class are expected to be clustered together: the prototypical network [2] is a well known approach from this category.

The prototypical network passes each of the images through a neural network and generates a latent representation of the particular image. Resulting latent representations of n support images are used to generate the prototype for each class by taking the mean of the image representations. Additionally, q query images are also embedded in the latent representation space. The network trains over many episodes and multiple epochs with the objective of minimizing the distance between the prototype and the queries of a particular class. During the evaluation, the n support images from each of the k classes from D_{test} dataset are used to generate the prototypes and the q query images are used to test the classification accuracy. As the goal of our study is to improve the adversarial robustness of representation learning-based FSL models, the experiments are conducted using prototypical networks [2].

B. Adversarial Samples and Adversarial Training

Adversarial samples are created by adding small noise, non-perceptible to a human, to the original image with the objective of making a trained neural network misclassify a given input image. Hence, to the human eye, the image still appears the same. Szegedy et al. [4] were the first to report the vulnerability of the neural networks to such attacks. Since then many adversarial sample generation techniques have been proposed such as Fast Gradient Sign Method (FGSM) [5] and Projected Gradient Descent (PGD) attack [17].

Adversarial samples are commonly used to evaluate the adversarial robustness of a model: in our study samples were generated with PGD attack [17]. PGD was selected as it is a strong multi-step whitebox attack that has been used for evaluation in previous works [6], [8]. In PGD, adversarial examples x_{adv} are generated by iteratively altering the legitimate sample x; starting with a uniform random perturbation on l_∞-norm hypersphere S around the legitimate sample. In each iteration, the adversarial samples are altered with an α step size based on the gradients ∇ of loss function $\mathcal{L}_\theta$ between target output y and the network output generated from current adversarial sample x^t_{adv}, while restricting the perturbations to a maximum allowed ε perturbation in the l_∞-norm bound. One timestep of the iterative PGD alteration is expressed as follows:

$$x^{t+1}_{adv} = \Pi_S\Big(x^t_{adv} + \alpha \cdot sign\big(\nabla_{x^t_{adv}} \mathcal{L}_\theta(x^t_{adv}, y)\big)\Big) \quad (1)$$

Adversarial samples can be repurposed to make a model robust through adversarial training which is one of the basic and intuitive defense mechanisms against adversarial attacks [5]. In adversarial training, adversarial samples with correct labels are available during training to ensure that the model learns to classify the adversarial samples by learning adversarially robust features. However, the main drawback of such training is that it only defends the network against a particular type of adversarial attack that was used in training, leaving the network vulnerable to other types of adversarial attacks. Adversarial training can be viewed as a min-max problem [17]:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\max_{\|\delta\|_\infty<\varepsilon} \mathcal{L}_\theta(x+\delta, y)\Big] \quad (2)$$

Here, $\mathcal{D}$ is the data distribution with pairs of input samples x and labels y. The perturbation δ is added to the legitimate sample x, and it is constrained in an l_∞-norm hypersphere of radius ε. The neural network parameters are θ while $\mathcal{L}$ is the loss. First, the adversarial samples are chosen by maximizing the loss, and then the loss is minimized with respect to the generated adversarial samples and correct labels y.

III. Related Work

This section reviews recent works in adversarially robust FSL, contrastive learning, and hypersphere embedding.

Adversarially Robust FSL. Adversarially robust FSL tries to correctly classify both legitimate and adversarial samples in a few-shot learning scenario. ADversarial Meta-Learner (ADML) [7], a MAML variant specifically designed for robustness, was the first work to introduce the concept of adversarially robust FSL. ADML integrated meta-learning and adversarial training; specifically, Fast Gradient Sign Method (FGSM) was employed for adversarial sample generation and adversarial training. Wang et al. [18] proposed an approach for improving adversarial robustness using unlabelled (semi-supervised) adversarial sample generation at the meta-update level and showed that this technique offers fast and effective robustness adaptation for MAML-based models. They also showed that introducing an auxiliary CL task can boost adversarial performance. Another recent technique for MAML-based models is ITS-MAML [19]: ITS-MAML showed that introducing adversarial loss into the MAML framework reduces the intrinsic dimension of features, which results in the drop of clean accuracy. They proposed the increase of n in n-shot to mitigate this drop; however, that moves the solution away from a typical fixed-shot FSL.

While approaches discussed so far build on the meta-learning paradigm, Goldblum et al. [6] considered both meta-learning-based FSL, and representation learning-based FSL. First, through extensive experiments using multiple attacks, they showed that naturally trained FSL models collapse under adversarial attacks. Then, they proposed to do adversarial learning by introducing adversarial samples in the querying stage of meta-learning. The work of Dong et al. [8] is the first work that explicitly tries to make representation learning-based FSL robust to adversarial attacks. They proposed an approach that learns an adversarially robust representation by using an auxiliary adversarial aware classification module, a feature purification module, and adversarial (instance-wise) re-weighted training.

Contrastive Learning (CL). CL helps the model to differentiate between classes by contrasting representations of samples in a latent space. Even in a self-supervised setting, a CL-based method [15] outperformed supervised image classifiers by a great margin with the use of a framework for CL of visual representations (SimCLR). Li et al. [20] bridged CL with clustering by introducing a new loss that encourages representations to be closer to their assigned prototypes. Also, recent works [21], [22] proposed CL-based frameworks to improve the performance of FSIC. However, these works have not evaluated the robustness against adversarial attacks. Jiang et al. [23] presented three variants of adversarial CL frameworks by learning representations that are consistent under adversarial perturbations. However, they have not evaluated the applicability of their framework for the FSL paradigm. The exploration of CL techniques for improving adversarial robustness in FSL problems has been very limited and further investigation is needed due to the potential of CL to increase distance between classes.

Hypersphere Embedding (HE).The concept of restricting the representation of an image to a hypersphere manifold can be seen in a few works which try to learn a feature representation of an image such that representations belonging to the same class are clustered together. SphereFace [11] embeds a facial feature representation on a hypersphere and uses angular softmax to achieve better performance than existing works in face recognition. Arcface [12] improved the SphereFace approach by introducing additive angular margin to the HE which increased intra-class compactness. HyperSphere Manifold Embedding (HSME) [10] and SphereReID [13] proposed two approaches for the person re-identification task which entails learning a discriminatory feature vector given an image of a person. Both approaches [10], [13] include the concept of transforming the representations to ensure that the feature representation lies on a hypersphere manifold. The work of Pang et al. [14] is especially relevant in our context as they have shown that using HE together with adversarial training can boost the performance of adversarial training in regular image classification tasks; however, they did not consider FSL.

While adversarially robust FSL studies dominantly focus on the meta-learning category, we consider representation learning-based FSL. Although CL improved the performance of FSIC [21], [22] and demonstrated increased robustness against attacks with large training datasets [23], these studies did not examine CL's ability to increase robustness against attacks in FSIC. Moreover, we further add HE as it demonstrated success in other representation learning tasks.

IV. Methodology

This section presents the proposed approaches, CL and HE for FSL adversarial robustness

A. Contrastive Learning for Adversarial Robustness

SimCLR [15] contrasts the representations of samples in a latent space, by contrasting a single positive (an augmented view of an image) against a set of negatives consisting of the entire remainder of the batch. The SimCLR approach was later extended to the fully-supervised setting [16] by leveraging the additional label information resulting in improved accuracy. Supervised CL (SupCL) [16] uses multiple positives to contrast in addition to multiple negatives unlike SimCLR [15]. Moreover, the label information makes sure that positive samples are not considered as negatives, which could happen

in SimCLR. The idea behind SupCL is to pull together samples belonging to the same class and to push apart clusters of samples from different classes. This property of increasing inter-class distance to a value greater than intra-class distance is leveraged in our work for FSIC and we examine if it could also improve adversarial robustness. Since FSIC is treated as a supervised learning problem, SupCL is better suited for our study than SimCLR.

In original SupCL [16], given an input batch of images, authors create two copies of batches by augmenting the samples, and both copies are forward propagated through the encoder network which generates 2048 dimensional embedding for each sample. These embeddings are further passed through a projection network which creates a 128-dimensional representation and the supervised loss is computed on these representations.

In our method, an encoder model *Enc(.)* is used to generate a latent representation for the input sample x as $r = Enc(x) \in \mathbb{R}^{D_E}$, where D_E is the dimension of the embedding. The encoder is followed by a projection network *Proj(.)* as in SupCL [16], with two linear layers and ReLU non-linearity followed by an l_2 normalization which ensures that the vector $z = Proj(r) \in \mathbb{R}^{D_P}$, where D_P is the dimension of the projection head output, lies on the surface of a unit hypersphere. Similar to SupCL [16], this projection network is employed only during the training and discarded during inference. Thus, there are no architectural changes to the *Enc(.)*.

In a k-way n-shot FSIC episode, the support set is used to create class prototypes, and the embeddings generated from the query set are used to calculate the loss. In contrast to SupCL [16], we do not use an augmentation module; instead, a batch which consists of the embeddings generated from the query set is passed through *Proj(.)* together with k class prototypes, to calculate the SupConLoss. We use projected class prototypes as *anchors*, projected query embeddings generated from the same class as *positives*, and projected query embeddings generated from different classes as *negatives*. The intuition behind this is to learn an embedding model which can gather query samples around their corresponding class prototypes while pushing away the prototypes which do not belong to the same class as shown in Figure 1, creating a buffer zone between class boundaries. Thus, SupConLoss can be reformulated as:

$$\mathcal{L}^{sup} = \sum_{i \in I} \mathcal{L}_i^{sup} = \sum_{i \in I} \frac{-1}{|P|} \sum_{p \in P} log \frac{e^{z_i \cdot z_p / \tau}}{\sum_{q \in Q} e^{z_i \cdot z_q / \tau}} \tag{3}$$

For a k-way n-shot problem, i, referred to as an anchor, is selected from the set of k class prototypes, Q is the query set, and $P \equiv \{p \in Q : y_p = y_i\}$, where $|P| = k$ is the set of positives in the query set Q which has the same class label as anchor i. The projection of embedding generated from i-th sample is z_i, symbol $\cdot$ denotes the inner (dot) product, and τ is a positive scalar temperature parameter.

Inspired by the work of Yang et al. [22], instead of backpropagating the pure Cross Entropy Loss (CEL) as in prototypical network [2] to train the embedding model, we

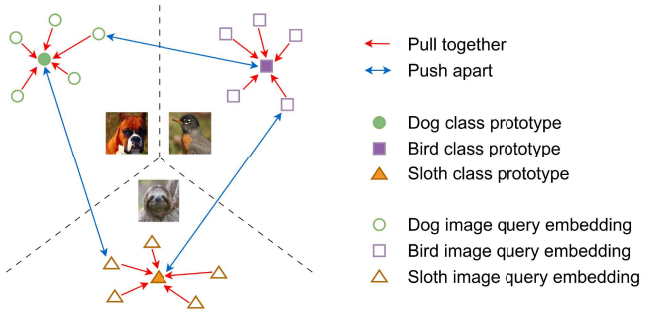

Fig. 1. Pulling together embedding representations of the same class and pushing apart representations of different classes.

backpropagate a weighted sum of CEL and the SupConLoss, to add contrastive power to the classifier. The total loss is:

$$\mathcal{L}^{total} = \alpha \mathcal{L}^{sup} + \beta \mathcal{L}^{cel} \tag{4}$$

Here α, β are weights to prioritize one loss over the other.

B. Hypersphere Embedding Latent Space

When proposing the prototypical networks, Snell et al. [2] evaluated cosine distance and Euclidean distance in the latent space as the distance metrics. They empirically showed that the Euclidean distance performed better, but it is important to note that the latent space in the original prototypical networks is unconstrained space as shown in Figure 2 (Left). Although Euclidean distance can better separate classes in an unconstrained latent space, studies in the face recognition domain [11], [12] demonstrated that cosine distance, when used with a latent HE space can outperform Euclidean distance in an unconstrained space. Hence, we combine HE and prototypical networks with cosine distance as the distance metric to examine if such architecture can improve adversarial performance in adversarial training. Moreover, this study examines if restricting the embeddings to a hypersphere manifold and using cosine distance improves the natural accuracy of prototypical networks.

Restricting the embeddings to a hypersphere latent space shown in Figure 2 (Right) is achieved by following Sphereface [11] and Arcface [12]. First, the bias of the last layer b^L is set to 0 to ensure that the embeddings are distributed around the origin. Then, the transformation of the last layer is modified

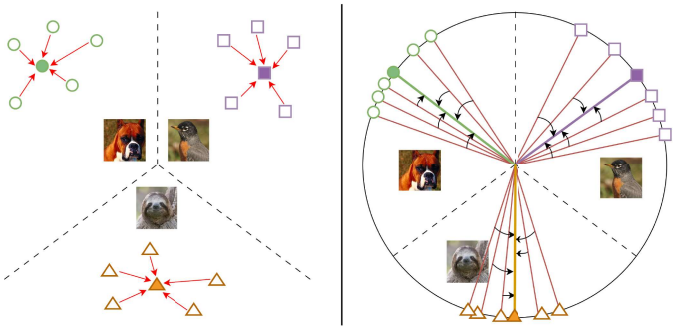

Fig. 2. **Left**: Image embeddings in a 2D unconstrained latent space. **Right**: Image embeddings in a 2D hypersphere (circle) latent space.

such that it depends on the angles between weights and the input features. Let the output of the penultimate layer be z_i^{L-1} for i-th sample, $z_i^{L-1} \in \mathbb{R}^d$ where d is the dimension of the embedding. Let the weight matrix of the last layer be W^L where W_j^L is the j-th column of the weight matrix ($W_j^L \in \mathbb{R}^d$). The vector multiplication operation of the last linear layer ${W_j^L}^\top \cdot z_i^{L-1}$ is modified as $\|W_j^L\|\|z_i^{L-1}\| \cos\theta_j$, where θ is the angle between W_j^L and z_i^{L-1}. The individual columns of the last layer weight matrix and the input features to that layer are restricted to a magnitude of 1 through l_2 normalization. The normalization makes $\|W_j^L\| = 1$ and $\|z_i^L\| = 1$; hence, this ensures that the predictions only depend on the angle between the feature and the weight. Thus, the learned embedding features are distributed on a hypersphere manifold with a radius of one. The loss of the HE approach is:

$$L_{HE} = -log\frac{e^{\cos\theta_{y_i}}}{\sum_{j=1}^{N} e^{\cos\theta_{y_j}}} \tag{5}$$

However, the above mentioned steps cannot be directly applied to prototypical networks because the conventional network architecture in prototypical networks (Conv4-64) is fully convolutional and the output is a flattened feature representation of the last convolution block. Hence, the weight matrix normalization cannot be done at the last layer. To address this issue, we keep the embedding network as it is, treat its output as z^{L-1}, and introduce a fully connected layer with weight matrix W^L. This newly introduced fully connected layer is treated as the last layer of the new configuration and we proceed with the previously mentioned steps.

While the vanilla HE approach ensures that the embeddings lie on a hypersphere manifold, it does not guarantee intra-class compactness. To ensure intra-class compactness, we follow Arcface [12] and introduce an additive angular margin. As shown in Equation 6, the penalty is applied to the correct class, hence the network is forced to learn a representation that holds embeddings belonging to each class within the bound of m, thus ensuring intra-class compactness during training.

$$L_{arc} = -log\frac{e^{\cos(\theta_{y_i}+m)}}{e^{\cos(\theta_{y_i}+m)} + \sum_{j=1;j\neq i}^{N} e^{\cos\theta_{y_j}}} \tag{6}$$

Both L_{HE} and L_{arc} limit the radius of the hypersphere to one which can result in computation limitations when calculating derivatives due to the finite numerical precision of discrete computing platforms. Hence, to circumvent this, we multiply the output logits of the last layer by a scale factor s before applying Softmax as done by Deng et al. [12]. The scaled version of the loss function becomes:

$$L_{arc-scaled} = -log\frac{e^{s\cos(\theta_{y_i}+m)}}{se^{\cos(\theta_{y_i}+m)} + \sum_{j=1;j\neq i}^{N} e^{s\cos\theta_{y_j}}} \tag{7}$$

The additive angular margin helps achieve intra-class compactness. In an attempt to further improve the model, the third variant $L_{lin-arc}$ is introduced where the margin m is linearly increased from 0 to an upper bound M during the training process. This ensures that the compactness constraint is enforced gradually, making it easier for the network to learn. In experiments, the modified prototypical network (protonet) is trained with the three losses L_{HE}, L_{arc}, $L_{lin-arc}$, and their scaled variants to investigate if these losses improve the performance of prototypical networks.

As we are interested in determining if HE can boost the performance of the adversarial defense, adversarial training was employed. Adversarial samples were generated using PGD attack [17] and then used as training query samples similar to the strategy presented by Goldblum et al. [6]. Specifically, two strategies were considered: In the first adversarial training strategy, only adversarial samples are used as queries to train the network while in the second strategy, adversarial samples together with legitimate samples are used as queries to train the network. In the first strategy, complete priority is given to adversarial performance and corresponding loss L_{adv}. While this is expected to improve the adversarial accuracy of the model, it will also reduce the clean accuracy (accuracy when evaluated using legitimate samples) as shown by previous works [6], [8]. In the second strategy, joint loss $L_{joint-adv}$ can achieve a trade off between clean accuracy and adversarial accuracy by varying the weight parameter λ. The two losses for a single episode are formulated as:

$$L_{adv} = -log\frac{e^{f_\theta(x+\delta)_y}}{\sum_{j=1}^{N} e^{f_\theta(x+\delta)_y}} \tag{8a}$$

$$L_{joint-adv} = -log\frac{e^{f_\theta(x)_y}}{\sum_{j=1}^{N} e^{f_\theta(x)_y}} + \lambda \cdot L_{adv} \tag{8b}$$

where x and y are the training samples and corresponding labels respectively. The transformation function of the neural network parameterized by θ is given by f_θ and the adversarial perturbation added to the training samples is given by δ.

V. Evaluation

This section introduces the dataset and the experimental setup followed by the corresponding results and discussion.

A. Dataset and Experiment Setup

Dataset. The presented techniques were evaluated on the widely-used Mini-ImageNet few-shot benchmark dataset [24] which consists of 100 classes, each one with 600 RGB images of size 84×84. From those classes, 80 random classes are assigned to D_{train} and the remaining 20 to D_{test}.

Neural Networks. The evaluation was carried out on the two standard FSL evaluation networks: Conv4-64 and Resnet 12. Conv4-64 network is a CNN network with four convolutional blocks. Each convolutional block has a 2D convolutional layer (with 64 output channels, kernel size 3, padding of 1, stride of 1), a batch normalization layer, ReLU activation, and a 2×2 max-pooling layer. As described in Section IV-B, we append a fully connected (FC) layer, with the same output and input dimensions, to the end of this network for HE evaluations.

The second network architecture, Resnet 12, is the 12 layer Resnet variant [25]. It has four Resnet blocks. Each Resnet block has three 3×3 convolution layers with varying number

of output channels (64, 160, 320, 640) in each block. Skip connections are present between start and end of a block. We introduced an additional FC layer similar to the one added to Conv4-64 for HE evaluations. This additional FC layer gives the HE model the advantage of having more trainable parameters; hence, to ensure a fair comparison, for experiments without HE, additional architecture is considered: base architecture + FC layer (without HE constraints).

FSL Setup. All architectures were evaluated under both 5-way 5-shot and 5-way 1-shot settings. The training and testing settings were the same as in the original prototypical networks [2]. For the 5-way 5-shot training, a support set of 20-way ($k_{train} = 20$) episodes for 5-shot ($n_{train} = 5$) classification tasks and a query set with each class consisting of 15 query images ($q_{train} = 15$) were used. For testing, a 5-way ($k_{test} = 5$), 5-shot ($n_{test} = 5$) setting with a single query sample ($q_{test} = 1$) was employed. Similarly for the 5-way 1-shot training, a support set of 20-way ($k_{train} = 20$) episodes for 1-shot ($n_{train} = 1$) classification tasks and a query set with each class consisting of 15 query images ($q_{train} = 15$) were used. For testing, a 5-way ($k_{test} = 5$), 1-shot ($n_{test} = 1$) setting with a single query sample ($q_{test} = 1$) was employed. For both setups, each training epoch consisted of 100 train episodes, and in the evaluation, the network was tested on 2000 test episodes and the average accuracy is reported.

Other Implementation Details. For adversarial training, the PGD parameters were $\varepsilon = 8/255$, $\alpha = 2$, and $steps = 7$, and for adversarial attack $\varepsilon = 8/255$, $\alpha = 2$, and $steps = 20$ which are the same as in the previous works [6], [8]. For CL, the temperature parameter was $\tau = 0.07$ and weights of Equation 4 were $\alpha = 1$ and $\beta = 1$. The additive angular margin was $m = 0.5rad$ and for scaling the the hypersphere radius was $s = 64$ as in the study of Deng et al. [12]. In joint adversarial training, the scaling coefficient λ was $\lambda = 0.5$. While λ is a hyperparameter to obtain a trade-off between clean and adversarial accuracy, experiments evaluate only one value to show that the results lie between natural training and adversarial training cases. In all experiments, the networks were trained for 80 epochs, with a learning rate of 0.001 and Adam optimizer [26].

All the networks and the FSL setups were evaluated under three settings:

1) Natural training – training using legitimate samples in a traditional way.
2) Adversarial training – training using PGD adversarial samples as queries as shown in Equation 8a.
3) Joint adversarial training – training using both legitimate and PGD adversarial samples as queries as shown in Equation 8b.

The accuracy on legitimate test images is referred to as *clean accuracy* and the accuracy under PGD attack as *adversarial accuracy*.

B. Results

Tables I and II show results for Conv4-64 and Resnet 12 respectively. All the results are presented as percentages (%).

1) Contrastive learning results: The first two rows of Table I compare the supervised CL (Protonet+CL) to a vanilla prototypical network for Conv4-64 architecture (Protonet). Protonet achieved a clean accuracy of 63.10% [1]; however, under PGD attack, vanilla protonet completely breaks down and achieves 0% adversarial accuracy. Adding CL increases clean accuracy to 64.05% which is about 1% increase. More importantly, the adversarial accuracy improves from 0% to 5.19% without any explicit adversarial defense mechanism. When adversarial training is introduced, both the vanilla protonet and the protonet with CL improve in terms of adversarial accuracy but this comes with a drop in clean accuracy. Furthermore, it can be observed that the results of joint adversarial training lie in between natural training and adversarial training results. Varying the λ parameter in Equation 8b can control the trade off between clean and adversarial accuracy.

Similar trends can be observed for a 5-way 1-shot setting as well. Vanilla protonet does not offer a significant adversarial accuracy (0.08%) but introducing CL increases adversarial accuracy to 25.12% while also marginally improving clean accuracy. Unlike the 5-way 5-shot setting, in the 5-way 1-shot setting, protonet with CL and adversarial training outperforms vanilla protonet with adversarial training in terms of both clean and adversarial accuracy. However, protonet with CL and adversarial training does not outperform naturally trained protonet with CL in either metric. Further, as expected, the results of joint adversarial training lie in between natural training and adversarial training results as in the 5-way 1-shot setting.

Similar observations can be made from Resnet 12 results presented in Table II. Under both 5-way 5-shot and 5-way 1-shot settings, the introduction of CL in the natural training setting improved the natural accuracy from 61.90% to 66.88% and from 34.85% to 48.12% respectively, and significantly improved the adversarial accuracy from 0.00% to 48.69% and from 2.15% to 37.94% respectively. Further, combining CL with adversarial training does not provide any additional benefit over natural training with CL.

2) Hypersphere embedding results: The second part of Table I examines the performance of HE with Conv4-64. As mentioned in Subsection V-A, this method is compared to a prototypical network with a fully connected layer at the end for a fair comparison. Examining 5-way 5-shot results, the protonet with FC layer and Euclidean distance (ProtoNet+FC in the table) achieves a clean accuracy of 64.3%, but as in the case of vanilla protonet, it completely fails under PGD attack. Introducing HE achieved a clean accuracy of 59.48% which does not outperform the protonet with FC layer and Euclidean distance on its own. Adding angular margin (Protonet+Arc) resulted in a minimal change while with additive angular margin and scaling (Protonet+Arc+Scale), protonet achieved

[1]The original paper reported 68.20% 5-shot accuracy. Our reproduction achieved 63.10% which could be due to the hardware platform, library versions, or the fact that we did not do model selection using a validation set. In this work, we compare our proposed approaches (trained under the same conditions) against our reproduction.

TABLE I
COMPARISON OF THE PROPOSED APPROACHES WITH BASELINES – NETWORK ARCHITECTURE: **CONV4-64**

Method	Distance	5-way 5-shot						5-way 1-shot					
		Natural Training		Adversarial Trainng		Joint adv. Training		Natural Training		Adversarial Trainng		Joint adv. Training	
		Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.
Protonet	Euclid.	63.10	0.00	37.85	22.16	50.78	15.24	41.98	0.08	26.11	15.79	30.80	13.52
Protonet + CL	Euclid.	64.05	**5.19**	41.46	25.52	51.16	16.92	42.72	**25.12**	29.18	17.91	28.99	11.61
Protonet + FC	Euclid.	64.30	0.00	43.51	27.09	51.68	13.88	42.80	0.03	28.70	17.21	32.43	14.98
Protonet + HE	Cosine	59.48	0.00	49.96	29.53	55.10	16.90	41.86	0.01	33.57	21.48	40.50	10.88
Protonet + Arc	Cosine	59.62	0.00	49.93	29.89	54.70	15.97	42.65	0.02	**34.25**	21.26	39.44	10.46
Protonet + Arc + Scale	Cosine	69.23	0.00	49.83	29.16	56.71	15.30	**46.46**	0.04	32.65	**21.86**	40.26	12.67
Protonet + Lin Arc	Cosine	59.33	0.00	49.69	29.65	54.47	16.31	42.45	0.01	33.25	20.89	39.97	11.89
Protonet + Lin Arc + Scale	Cosine	**69.68**	0.00	**51.11**	**29.90**	57.46	14.56	46.24	0.02	33.02	21.79	40.48	12.69

Legend: CL – contrastive Learning; FC – fully connected layer; HE – Hypersphere embedding; Arc – additive angular margin; Scale – logits scaled after adding the margin; Lin Arc – linearly scheduled additive angular margin;

TABLE II
COMPARISON OF THE PROPOSED APPROACHES WITH BASELINES – NETWORK ARCHITECTURE: **RESNET 12**

Method	Distance	5-way 5-shot						5-way 1-shot					
		Natural Training		Adversarial Trainng		Joint adv. Training		Natural Training		Adversarial Trainng		Joint adv. Training	
		Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.
Protonet	Euclid.	61.90	0.00	37.97	17.15	52.95	10.15	34.85	2.15	26.29	10.89	31.19	6.70
Protonet + CL	Euclid.	**66.88**	**48.69**	41.98	14.14	56.46	7.12	48.12	**37.94**	31.49	7.21	34.99	5.37
Protonet + FC	Euclid.	48.27	0.02	41.93	1.08	42.51	0.15	29.53	0.77	28.92	0.08	23.95	0.09
Protonet + HE	Cosine	54.21	0.00	51.69	26.96	54.61	13.32	45.12	0.00	34.62	21.22	41.54	9.85
Protonet + Arc	Cosine	55.00	0.00	51.15	26.25	54.92	13.79	45.90	0.00	34.47	20.05	39.43	11.83
Protonet + Arc + Scale	Cosine	65.27	0.00	53.30	31.62	57.97	10.54	**52.25**	0.00	36.98	22.53	43.79	12.44
Protonet + Lin Arc	Cosine	55.10	0.00	52.34	26.29	55.69	12.66	46.25	0.00	34.87	21.18	41.90	10.90
Protonet + Lin Arc + Scale	Cosine	65.46	0.00	**54.08**	**31.89**	54.45	6.28	51.81	0.00	**37.20**	**22.89**	45.40	12.14

69.23% clean accuracy and outperformed other variants in terms of clean accuracy. Replacing additive angular margin with linearly scheduled additive margin (Lin) achieved very similar results.

None of the HE variants provided any inherent adversarial robustness like CL did; however, our quest was to also examine if HE can boost other adversarial defense mechanisms. As expected, when used together with adversarial training, the Protonet+HE model achieved a clean accuracy of 49.96% and adversarial accuracy of 29.53% compared to 43.51% clean accuracy and 27.09% adversarial accuracy of the non HE variant. This is $\sim$ 6.5% and $\sim$ 2.5% improvement in terms of clean and adversarial accuracies respectively. These results show that HE does boost the adversarial training performance under the FSL setting.

Looking at the 5-way 1-shot results in Table I, similar performance gains can be observed. While protonet with HE does not outperform protonet with FC layer in natural training, it boosts both clean and adversarial accuracies by $\sim$ 5% and $\sim$ 4% under adversarial training. When additive angular margin and scaling are introduced to the HE variant, the model outperforms the FC counterpart.

The second part of Table II shows the performance of HE with Resnet 12. The main difference in comparison to Table I is that introducing an additional FC layer results in a significant drop in naturally trained clean accuracy compared to the vanilla protonet. This drop is observable in both 5-shot and 1-shot settings. However, with HE constraints, additive angular margin, and scaling, the model improves and outperforms the vanilla protonet variant by achieving 65.27% and 52.25% for 5-shot and 1-shot respectively. Similar to the Conv4-64 network, introducing HE constraints significantly boosts both clean and adversarial accuracies under adversarial training in Resnet 12 as well.

A common observation from both tables I and II is that linearly scheduling the additive angular margin (Lin Arc) does not offer any clear advantage over the rigid additive angular margin (Arc). Another common observation is that joint adversarial training in most of the tested scenarios achieves results that lie in between natural training and adversarial training results just like in the CL setting.

C. Discussion

The goal of this study is to examine CL and HE abilities to inherently provide adversarial robustness or to boost adversarial performance when used in conjunction with a defense mechanism. The proposed techniques were examined using protonet [2] with two different neural network architectures and under two FSL settings.

The first key takeaway from the CL results is that CL improves the clean accuracy of the prototypical networks and also provides inherent adversarial robustness to the prototypical networks. The reason for this is the CL algorithm's ability to increase the distance between an anchor and a negative further than the distance between the anchor and the positive. Through this step, the algorithm is essentially increasing inter-class distances, which is making it difficult for the PGD attack to fool the network. The next takeaway is that combining adversarial training with CL does not outperform natural training with CL. Also, we showed that joint adversarial training can be used when a trade off has to be achieved between clean and adversarial accuracy.

The main takeaway from HE results is that HE does not provide inherent adversarial robustness as CL does. However, introducing HE constraints improves both clean and adversarial accuracies when used with a defense mechanism such as adversarial training. Next, introducing HE constraints and additive angular margin with scaling improves the performance of vanilla protonet, and finally, joint adversarial training can be used when a trade off needs to be achieved between clean and adversarial accuracy.

The research direction of our work is orthogonal to the works discussed in Section III; hence, we believe, that our techniques can be combined with other defense mechanisms (that can be applied in a representation learning context). Further investigation is needed to explore the presented techniques with other defense mechanisms.

VI. Conclusion

Ensuring adversarial robustness in FSL is a critical but challenging problem due to the small number of training samples. This paper improves the adversarial performance of representation learning-based FSL by incorporating CL and HE with protonet, driven by the success of those approaches in related representation learning tasks. Supervised CL increases inter-class distance more than the intra-class spread, while HE with additive angular margin ensures intra-class compactness. The evaluation answers the research questions stated in Section I. Experiments showed that CL provides inherent adversarial robustness to the examined FSL models, while HE does not. However, the evaluation also demonstrated that HE, when used in conjunction with an adversarial defense mechanism, boosts the performance of the defense mechanism. Finally, we showed that HE with cosine distance improves the natural accuracy of the considered representation learning-based FSL technique, the protonet.

Future work will examine combining advanced defense mechanisms with the presented techniques to boost adversarial robustness. Moreover, CL and HE will be examined on other datasets and with other adversarial attacks.

References

[1] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[2] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *International Conference on Neural Information Processing Systems*, 2017.

[3] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, 2017.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[5] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[6] M. Goldblum, L. Fowl, and T. Goldstein, "Adversarially robust few-shot learning: a meta-learning approach," in *International Conference on Neural Information Processing Systems*, 2020.

[7] C. Yin, J. Tang, Z. Xu, and Y. Wang, "Adversarial meta-learning," *arXiv preprint arXiv:1806.03316*, 2018.

[8] J. Dong, Y. Wang, J.-H. Lai, and X. Xie, "Improving adversarially robust few-shot image classification with generalizable representations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[9] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021.

[10] Y. Hao, N. Wang, J. Li, and X. Gao, "HSME: Hypersphere manifold embedding for visible thermal person re-identification," in *AAAI conference on artificial intelligence*, 2019.

[11] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[13] X. Fan, W. Jiang, H. Luo, and M. Fei, "Spherereid: Deep hypersphere manifold embedding for person re-identification," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 51–58, 2019.

[14] T. Pang, X. Yang, Y. Dong, K. Xu, J. Zhu, and H. Su, "Boosting adversarial training with hypersphere embedding," in *International Conference on Neural Information Processing Systems*, 2020.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning*, 2020.

[16] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *International Conference on Neural Information Processing Systems*, 2020.

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

[18] R. Wang, K. Xu, S. Liu, P.-Y. Chen, T.-W. Weng, C. Gan, and M. Wang, "On fast adversarial robustness adaptation in model-agnostic meta-learning," in *International Conference on Learning Representations*, 2021.

[19] X. Duan, G. Kang, R. Wang, S. Han, S. Xue, T. Wang, and B. Zhang, "Rethinking the number of shots in robust model-agnostic meta-learning," *arXiv preprint arXiv:2211.15180*, 2022.

[20] J. Li, P. Zhou, C. Xiong, and S. Hoi, "Prototypical contrastive learning of unsupervised representations," in *International Conference on Learning Representations*, 2021.

[21] Z. Yang, J. Wang, and Y. Zhu, "Few-shot classification with contrastive learning," in *European Conference on Computer Vision*, 2022.

[22] S. Yang, F. Liu, and Z. Chen, "Feature hallucination in hypersphere space for few-shot classification," *IET Image Processing*, vol. 16, no. 13, pp. 3603–3616, 2022.

[23] Z. Jiang, T. Chen, T. Chen, and Z. Wang, "Robust pre-training by adversarial contrastive learning," in *International Conference on Neural Information Processing Systems*, 2020.

[24] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *International Conference on Neural Information Processing Systems*, 2016.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.