Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. Digital Object Identifier 10.1109/ACCESS.2023.1120000

GReAT: A Graph Regularized Adversarial Training Method

SAMET BAYRAM¹, (Member, IEEE), KENNETH BARNER², (Fellow, IEEE)

¹Electrical and Computer Engineering Department, University of Delaware, Newark, DE 19716 USA (e-mail: sbayram@udel.edu) ²Electrical and Computer Engineering Department, University of Delaware, Newark, DE 19716 USA (e-mail: barner@udel.edu)

¹Electrical and Computer Engineering Department, University of Delaware, New ²Electrical and Computer Engineering Department, University of Delaware, New Corresponding author: Samet Bayram (e-mail: sbayram@udel.edu).
ABSTRACT This paper presents GReAT (Graph R method designed to enhance the robust classificate examples, characterized by subtle perturbations that chine learning. Although adversarial training is effect the underlying data structure. In response, GReAT is training process, leveraging the data's inherent structuring process, leveraging the data's inherent structure of the underlying data. Extensive evaluations on benchma of-the-art methods in robustness, achieving notable compared to the second-best methods, GReAT defends to unseen data. Extensive evaluations on benchma of-the-art methods in robustness, achieving notable compared to the second-best methods, GReAT achie CIFAR-10 against FGSM attack and 10.57% for SV GReAT demonstrates a performance increase of app a 5.54% increase against PGD attack. This paper princluding numerical results and comparisons with of GReAT in advancing the performance of deep lee
INDEX TERMS Adversarial examples, adversar image classification, semi-supervised learning, robust that uses artificial neural networks with multipers and neurons to analyze and learn from large a of data. Deep learning algorithms automatically learning algorithms **ABSTRACT** This paper presents GReAT (Graph Regularized Adversarial Training), a novel regularization method designed to enhance the robust classification performance of deep learning models. Adversarial examples, characterized by subtle perturbations that can mislead models, pose a significant challenge in machine learning. Although adversarial training is effective in defending against such attacks, it often overlooks the underlying data structure. In response, GReAT integrates graph-based regularization into the adversarial training process, leveraging the data's inherent structure to enhance model robustness. By incorporating graph information during training, GReAT defends against adversarial attacks and improves generalization to unseen data. Extensive evaluations on benchmark datasets demonstrate that GReAT outperforms stateof-the-art methods in robustness, achieving notable improvements in classification accuracy. Specifically, compared to the second-best methods, GReAT achieves a performance increase of approximately 4.87% for CIFAR-10 against FGSM attack and 10.57% for SVHN against FGSM attack. Additionally, for CIFAR-10, GReAT demonstrates a performance increase of approximately 11.05% against PGD attack, and for SVHN, a 5.54% increase against PGD attack. This paper provides detailed insights into the proposed methodology, including numerical results and comparisons with existing approaches, highlighting the significant impact of GReAT in advancing the performance of deep learning models.

INDEX TERMS Adversarial examples, adversarial learning, adversarial training, graph regularization, image classification, semi-supervised learning, robustness.

EEP learning is a subset of machine learning (ML) that uses artificial neural networks with multiple layers and neurons to analyze and learn from large amounts of data. Deep learning algorithms automatically learn and extract relevant features from the data to make predictions. The feature extraction process allows algorithms to achieve higher levels of accuracy and perform more complex tasks. In recent decades, deep learning has achieved impressive results in various domains, including image and text classification, speech recognition, image generation, and natural language processing [1], [2], [3]. Supervised learning methods achieved the most successful results. In this learning technique, the model is trained on a labeled dataset, meaning the input data is accompanied by its corresponding output labels. Supervised learning aims to predict new, unseen data based on the patterns learned from the labeled data. Deep neural networks adjust the weights and biases of the network through back-propagation using output labels and original labels.

Semi-supervised learning combines both supervised and unsupervised learning techniques. In this type of learning, a model is trained on a data set that has labeled and unlabeled instances. The goal is to use the labeled data to make predictions on unlabeled and new, unseen, data. This method is used often when there is a limited amount of labeled data but a larger amount of unlabeled data. There are various algorithms for propagating labels through the graph, such as label propagation [4, 5], pseudo-labeling [6], transductive SVMs [7], and self-training [8]. Label propagation provides outstanding performance for semi-supervised learning to classify graph nodes. It is based on the idea that a node's labels can be propagated to its neighbors based on the assumption that nodes with similar labels are more likely to be connected.

Despite their significant success, deep learning models are known to be vulnerable to adversarial examples. These examples are created by adding small, carefully chosen perturbations to the input data. The perturbed data remain visually similar to the original data, but are misclassified by the model [9–11]. The existence of adversarial examples has drawn significant attention to the machine-learning community. Showing the vulnerabilities of machine learning algorithms has opened critical research areas in the attack and robustness areas. Studies have shown that adversarial attacks are highly effective on many existing AI systems, especially on image classification tasks; [10, 12–18]. For instance, [12] show that even small perturbations in input testing images can significantly change the classification accuracy.

The authors of [9] attempt to explain the existence of adversarial examples and proposes one of the first efficient attack algorithms in white–box settings. [19] proposed projected gradient descent (PGD) as a universal first-order adversarial attack. They stated that network architecture and capacity play a significant role in adversarial robustness. Notable other popular adversarial attack methods are the Carlini-Wagner Attack (CW) [10], Basic Iterative Method (BIM) [16], and Momentum Iterative Attack [20]. [21] show the transferability of black-box attacks among different ML models.

Adversarial defense mechanisms can broadly be classified into three categories. The first category predominantly centers on pre-processing techniques tailored for DL models to mitigate adversarial perturbations in the adversarial examples. Methods in this category include feature denoising [22], Fourier filtering [23] and random resizing coupled with random padding [24] among others. The second category targets modifications in the architecture of neural networks, including alterations in activation functions [25], adaptations to learning processes such as distillation [26], the introduction of novel loss functions [27] and adjustments in training procedures [9]. The last category attempts to implement the combination of the first two categories.

The authors of [28] introduce two adversarial training methods, topology aligning adversarial training (TAAT) and adversarial topology aligning (ATA), which leverage topology information to maintain consistency in the topological structure within the feature space of both natural and adversarial examples, and further introduce a Knowledge-Guided (KG) training scheme to ensure stability and efficiency in topology alignment. TRADES (tradeoff-inspired adversarial defense via surrogate-loss minimization) [29] focuses on decomposing the prediction error for adversarial examples into natural error and boundary error. The authors also introduce a differentiable upper bound using the theory of classification-calibrated loss, which serves as the foundation for their defense method, TRADES, designed to trade off adversarial robustness against accuracy.

The authors of adversarial logit pairing (ALP) [30] introduce a technique called logit pairing, which encourages similarity between logits for pairs of examples, resulting in improved accuracy on adversarial examples compared to standard adversarial training [19]. Triplet Loss Adversarial (TLA) training [31] introduces a metric learning approach to regularize the representation space under attack, resulting in increased robustness of classifiers against adversarial attacks. Adversarial Contrastive Learning (ACL) [32] enhances robustness-aware self-supervised pre-training by incorporating adversarial perturbations into a contrastive learning framework. This approach improves feature consistency under both data augmentations and adversarial perturbations, leading to models that are label-efficient and robust.

The existing literature underscores the effectiveness of unlabeled samples in improving deep learning performance [4, 33-35]. Additionally, studies show that unlabeled data improve adversarial robustness, [36]. Motivated by these insights, we propose a Graph-Regularized Adversarial Training method (GReAT) to improve the robustness performance. The proposed method utilizes the structural information from the input data to improve the robustness of deep learning models against adversarial attacks. The main idea within GReAT is to construct a graph representation of clean data with an adversarial neighborhood, where each node represents a data point, and the edges encode the similarity between the nodes. This approach allows us to incorporate the structural information from the data into the training process, which helps create robust classification models. To evaluate the effectiveness of our approach, we conduct experiments on data sets: Tensor-Flow's flower data set [37] and CIFAR-10 [38]. We compare GReAT with several state-of-the-art methods. The results show that the proposed approach consistently outperforms the baselines in terms of accuracy and robustness against adversarial attacks. Our proposed GReAT graph-based semisupervised learning approach for adversarial training provides a promising direction for improving the robustness of deep learning models against adversarial attacks. We list the main contributions of our proposed method below:

- GReAT integrates graph structure into the adversarial training process.
- It improves the model's feature extraction performance by including neighboring information.
- The model enhances the learning capabilities of the model by leveraging the underlying structure of the training samples.
- The proposed method shows advantages in the adversarial training method, compared with the state-of-the-art methods, thereby improving generalization and robustness.

II. BACKGROUND AND RELATED WORKS

This section covers the relevant background and related works. In particular, we cover deep learning and semisupervised learning, adversarial learning, and graph-based semi-supervised learning.

A. DEEP LEARNING AND SEMI-SUPERVISED LEARNING

DL models are complex non-linear mapping functions between input and output. They consist of multiple layers and neurons with activation functions. They extract features from input samples and predict labels based on those features. Neural networks are trained using vast amounts of labeled data and can learn and improve their performance over time utilizing backpropagation algorithms. The following equation represents the prediction process of the classical deep learning paradigm:

$$\boldsymbol{Y}:f(\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{b}),\tag{1}$$

where X is the data fed into the neural network f, θ represent the values assigned to the connections between the neurons in the network, and b is the offsets applied to the input data. The output is the result produced by the neural network after processing the input data through its layers of neurons.

Semi-supervised learning uses labeled and unlabeled data to train a model. The following representation shows the prediction process of semi-supervised learning:

$$\boldsymbol{Y}: f(\boldsymbol{X}_{\boldsymbol{l}}, \boldsymbol{X}_{\boldsymbol{ul}}, \boldsymbol{\theta}, \boldsymbol{b}), \tag{2}$$

where X_l is the labeled data and X_{ul} is the unlabeled data. The weights and biases are the same as in supervised-learning learning. The output is the result produced by the model after processing the labeled and unlabeled data through its layers of neurons. A label assignment procedure typically exists in semi-supervised learning to annotate the unlabeled data. This procedure employs a smoothing function or similarity metrics to assign the label of the most similar labeled sample to the unlabeled sample, [5].

B. ADVERSARIAL LEARNING

A data instance x' is considered an adversarial example of a natural instance x when x' is close to x, under a specific distance metric, while $f(x') \neq y$, where y is the label of x, [39]. Formally, an adversarial example of x is can be defined as

$$\boldsymbol{x}': D(\boldsymbol{x}', \boldsymbol{x}) < \epsilon, f(\boldsymbol{x}') \neq \boldsymbol{y}, \tag{3}$$

where $D(\cdot, \cdot)$ represents a distance metric, such as the $\|\cdot\|_2$ norm, and ϵ is a distance constraint, which limits the amount of allowed perturbations. Since the existence of adversarial examples is a significant threat to DL models, adversarial attack and defense algorithms are intensively investigated to improve the robustness and security of such models.

For instance, FGSM [9] was proposed to generate adversarial examples and attack DL models. The PGD algorithm, an iterative version of the FGSM attack, was proposed to generate adversarial examples by maximizing the loss increment within an L_{∞} norm-ball, [19]. Although many defense methods have been proposed, adversarial training is the most efficient approach against adversarial attacks, [9, 19]. The authors of [9] proposed using adversarial attack samples during training so that the classifier can learn the features of adversarial examples and their perturbations. The classifier's robustness against adversarial attacks is substantially enhanced due to the integration of adversarial examples in the training phase. It effectively empowers the classifier to develop a more robust defense mechanism against adversarial instances. Formally, adversarial training is defined as

$$\boldsymbol{\theta^*} = \operatorname*{arg\,min}_{\theta \in \Theta} \frac{1}{L} \sum_{i=1}^{L} \max_{\mathsf{D}(\mathbf{x}'_i, \mathbf{x}_i) < \epsilon} \ell_{adv}(\theta, \mathbf{x}'_i, y_i). \tag{4}$$

The above equation states a min-max procedure under the specific distance constraint. In the inner maximization component, the adversarial training seeks an adversarial example \mathbf{x}'_j to maximize the loss ℓ_{adv} , under the distance metric $D(\mathbf{x}'_j, \mathbf{x}_j) < \epsilon$, given the natural sample \mathbf{x}_j . The outer minimization seeks the optimal gradient θ^* that yields the global minimum empirical loss. In their work, Madry *et al.* iteratively applies the PGD algorithm during training to search for strong adversarial examples to maximize ℓ_{adv} . This helps the model yield improved robustness against PGD and FGSM attacks. Adversarial training with PGD is considered one of the strongest defense methods, [19].

C. GRAPH-BASED SEMI-SUPERVISED LEARNING

Graph-based semi-supervised learning uses labeled and unlabeled data to train DL models, [33, 35, 40–43]. This approach uses a small amount of labeled data and a large amount of unlabeled data to learn the graph structure of the given data. A given graph can be represented as G = (V, E, W), where V indicates data points as vertices, E represents edges between data points, and W is the edge weight matrix. The edges between the vertices are created on the basis of a similarity metric between the data points. Graph-based semi-supervised learning aims to use the graph structure and the labeled data to learn the label for the unlabeled data points. This technique is typically done by propagating the labels from the labeled data points to the unlabeled data points through the similarity graph of the entire data, [5, 34, 43, 44].

In graph-based semi-supervised learning, label propagation is often used to classify nodes in a graph when only a few nodes have been labeled. This method starts with the labeled nodes and propagates their labels to their neighbors. The labels are then iteratively propagated repeatedly until the mapping function converges and the entire graph is labeled. As shown in [5], the loss function of the graph-based semisupervised learning can be represented as:

$$\sum_{i=1}^{L} \ell(\theta, \mathbf{x}_i, y_i) + \lambda \sum_{i,j} w_{i,j} \|h(x_i) - h(x_j)\|^2,$$
 (5)

where the first term represents the standard supervised loss while the second term represents the penalty of the neighborhood loss. Note that w_{ij} represents the similarity between different instances, and λ controls the contribution of neighborhood regularization. When $\lambda = 0$, the loss term becomes the standard supervised loss. The amount of penalty depends on the similarity between the instance x_i and its neighbors. In addition, *h* represents a lookup table that contains all samples and similarity weights. It can be obtained with a closed-form solution, according to [44].

The authors of [40] proposed embedding samples instead of using lookup tables by extending the regularization term in Eq. 5. The regularization term becomes $\lambda \sum_{i,j} \alpha_{i,j} ||g_{\theta}(x_i) - g_{\theta}(x_j)||^2$, where $g_{\theta}(\cdot)$ indicates the embedding of samples generated by a neural network. Transforming the regular-

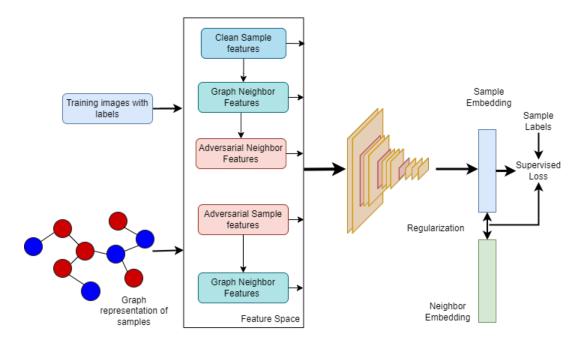


FIGURE 1. GReAT framework.

ization term by transitioning from f to g leverages stronger constraints on the neural network, according to [5].

Here, we extend Eq. 5 by replacing the lookup table term with the embedding term in the regularization component and defining a general neighbor similarity metric. This approach yields Eq. 6 as

$$\sum_{i=1}^{L} \ell(\theta, \boldsymbol{x}_i, y_i) + \lambda \sum_{i=1} w_i \mathsf{D}(g_{\theta}(x_i), \mathsf{N}(g_{\theta}(x_i))).$$
(6)

In the Eq. 6, N represents the neighbors of a given sample x_i , w_i represents the edge weight between the sample x_i and its neighbors, and D represents the distance metric between embeddings.

III. GRAPH REGULARIZED ADVERSARIAL TRAINING

In this Section, we integrate the adversarial learning process, [19], into the graph-based semi-supervised learning framework, [5, 40, 43], to take advantage of both adversarial training and semi-supervised learning. The main framework of GReAT is shown in Fig. 1.

The feature space encompasses both the labeled original training samples and the adversarial examples that are created through adversarial regularization and neighbor similarities. This feature space is crucial for identifying the nearestneighbor samples. When we feed a batch of input samples to the neural network, it includes not only the original samples but also their corresponding neighbors. In the final layer of the neural network, we derive a sample embedding for each of these samples. The training objective for regularization includes two components: the supervised loss and the label propagation loss, which accounts for neighbor-related loss. In other words, it considers the impact of neighbors on the overall training objective. Thus,

$$\mathcal{L}_{GReAT} = \mathcal{L}_{adv} + \lambda \mathcal{L}_N, \tag{7}$$

where \mathcal{L}_{adv} represents the supervised loss from training labels of clean and adversarially perturbed samples, and \mathcal{L}_N represents the neighbor loss, which includes the loss from the clean training samples and adversarially perturbed samples.

We consider similar instances as neighbors of sample x in the graph regularized semi-supervised learning case. In our case, we consider an adversarial example, x', in addition to a neighbor of sample x. Next, we extend Eq. 6 by including adversarial and adversarial neighbor losses as new regularizer terms. Formally, the unpacked form of Eq. 7 is:

$$\mathcal{L}_{GReAT}(\Theta) = \sum_{i=1}^{L} \ell(\theta, \mathbf{x}_i, y_i) + \alpha_{11} \sum_{i=1}^{L} \ell_N(y_i, x_i, \mathbf{N}(x_i)) + \alpha_{22} \sum_{i=1}^{L} \ell_N(y_i, x_i', \mathbf{N}(x_i')) + \alpha_3 \sum_{i=1}^{L} \ell(\theta, \mathbf{N}_{adv}(x_i), y_i).$$
(8)

In the above equation, N(x) represents neighbors of sample x. The neighbors could be clean or adversarially perturbed samples. Thus, N(x') represents the neighbors of adversarial example x'. Its neighbors could be clean samples and adversarial examples. Specifically, $N_{adv}(x)$ represents the adversarial neighbor of the sample x. The adversarial neighbors have

IEEE Access

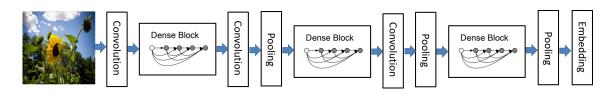


FIGURE 2. Densenet121 for generating image embeddings.

the same label as the original sample x similar to the standard adversarial training.

We obtain adversarial examples using PGD as it's described in [19]. Note that the $\alpha_{11}, \alpha_{22}, \alpha_3$ hyperparameters determine the contributions of different neighborhood types, which are shown in Fig. 6 as sub-graph types. The α terms can be tuned according to the performance on clean and adversarially perturbed testing inputs. The pseudo-code of GReAT method is given in the Algorithm 1. Furthermore, a detailed explanation of the embedding of neighbor nodes and graph construction between clean and adversarial examples is shown in Section III-B.

Algorithm 1 Graph Regularized Adversarial Training (GReAT)

- 1: Input: Labeled data X_l , unlabeled data X_{ul} , model parameters Θ , hyperparameters α_{11} , α_{22} , α_3 , λ
- 2: **Output:** Trained model Θ^*
- Train classifier f(X_l, Θ) on labeled data using supervised loss l
- 4: 2: Generate adversarial examples X_{adv}
- 5: 3: Propagate labels of X_l to X_{ul} using label propagation on graph
- 6: 4: Construct graph \mathcal{G} with nodes X_l, X_{ul}, X_{adv}
- 7: 5: Compute neighbor set N(x), N(x'), N_{adv}(x) for each sample x, x'
- 8: 6: Train model using loss \mathcal{L}_{GReAT} defined in Eq. 8
- 9: return Θ^*

A. RELATED PREVIOUS METHODS

Creating graph embeddings using Deep Neural Networks (DNNs) is a well-known method, [40]. Furthermore, the propagation of unlabeled graph embeddings using transductive methods, [5, 34], are efficient and well studied. Neural Graph Machines (NGMs), [43], are a commonly used example of label propagation and graph embeddings, along with supervised learning. The proposed training objective takes advantage of these frameworks and provides more robust image classifiers. Therefore, the training objective can be considered a combination of nonlinear label propagation and a graph-regularized version of adversarial training.

B. GRAPH CONSTRUCTION

We use a pre-trained model, DenseNet121, [45], to generate image embeddings as a feature extractor. The pre-trained model has weights obtained by training on ImageNet. The pre-trained model is more complex than the model we use to train and test the proposed regularization algorithm in our simulations. Numerous studies show that complex DNNs are better feature extractors than shallow networks, [46, 47]. Another significant advantage of using larger pre-trained models to obtain embeddings is to reduce computational costs. The process of creating embeddings is illustrated in Fig. 2.

Generating appropriate inputs to the neural network plays a significant role in yielding correct predictions. As noted above, we use a pre-trained DL model to create node embeddings. We generate embeddings of clean samples and adversarial examples to obtain the neighborhood relationship between clean and adversarially perturbed examples.

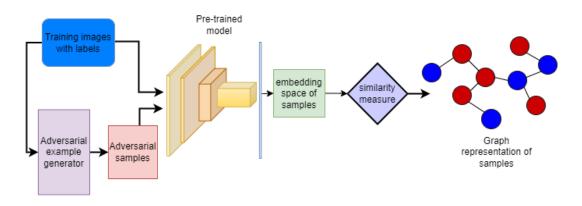


FIGURE 3. Graph creation from embedding of clean and adversarial examples.

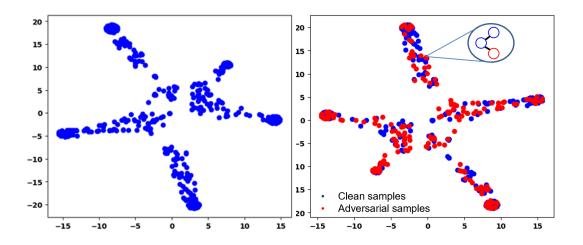


FIGURE 4. Samples in embedding space. The left figure represents all the samples in the validation data set. The right figure shows some clean samples and their adversarial neighbors.

The overall graph construction process is shown in Fig. 3. Similarly, one-dimensional embedding is a crucial process for measuring sample similarities. Since the size of the embeddings is the same, we can visualize clean and adversarial examples in the embedding space using the [48] t-distributed stochastic neighbor embedding (t-SNE) method.

In Fig. 4, we utilize t-SNE (t-Distributed Stochastic Neighbor Embedding) to create a visual representation of the validation data set obtained from TensorFlow's flower dataset. The primary purpose of this visualization is to provide insight into the distribution and relationships among the data points. The left panel of the figure is dedicated to displaying all the samples that constitute the validation data set. It is important to note that this data set encompasses samples belonging to five distinct classes. Each class represents a specific category or type of data within the dataset, and the samples within each class share certain common characteristics or features.

By visually representing the data set using t-SNE, we aim to reduce the dimensionality of the data while preserving its inherent structure and relationships. This reduction in dimensionality allows us to plot the data points in a two-dimensional space, making it easier to discern patterns, clusters, and similarities among the samples. Visualization is a valuable tool for gaining a deeper understanding of how the different classes are distributed and how they relate to each other within the validation data set. The figure panel on the right shows how adversarial examples are distributed around clean samples.

The visualization of the embeddings highlights a strong connection between individual samples and their respective neighbors, effectively distinguishing between various classes. We use the strong neighborhood connections to learn better and create more robust models. Consequently, we use these node embeddings as input features to the neural network by creating an adjacency embedding matrix, as shown in Fig. 5. In particular, we use the label propagation method, [6], to propagate the information from the labeled data points to the unlabeled instances, which improves the model's performance on both clean and adversarial examples.

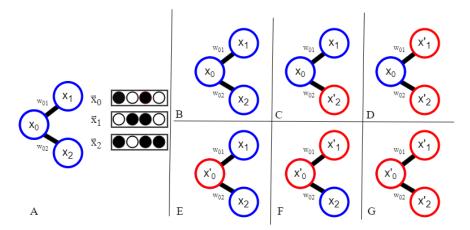


FIGURE 5. A: A sample with two neighbors showing their sub-graph and feature inputs. Blue nodes represent clean samples, and red nodes represent adversarially perturbed samples. B,C,D,E,F, and G show how clean samples and adversarial examples may link on the graph structure.

IEEE Access



FIGURE 6. From left to right: labeled sample, the first neighbour and the second neighbour. The samples are taken from Tensorflow's flowers data set.

Sample sub-graph of training instances are shown in Fig. 5. These examples might be labeled or unlabeled since we generate embeddings for each sample and create the graph based on the similarity between embeddings. A visual example of a sub-graph is demonstrated in Fig. 6. Three examples of subgraph types are shown. The first column of the figure shows labeled samples. The second and third columns show the labeled samples' two most similar neighbors. We associate these samples and their neighbors with the sub-graph examples, as noted in Fig. 5. For instance, the first row of images in Fig. 6 represents Fig. 5-D, since the labeled sample is clean and its first and second neighbors are adversarially perturbed samples. The second row of Fig. 6 represents Fig. 5-F, since the labeled sample is adversarially perturbed and its neighbors are one clean sample and one adversarially perturbed sample. Finally, the third row represents Fig. 5-C, since the labeled instance is clean and its neighbors are clean and adversarially perturbed samples.

Note that a labeled sample may have one neighbor or none, for instance, if the similarity measure of the embeddings cannot pass the similarity threshold. In that case, the labeled sample goes through the neural network as a regular input without graph regularization.

C. OPTIMIZATION

The training process begins with a minibatch of samples and their edges. Instead of using all available data at once, the training process randomly selects a subset of edges for each iteration. This helps introduce randomness and variability into the training process, which is beneficial to the learning process. Additionally, to further improve the training process, selected edges are chosen from a nearby region to increase the likelihood of some edges. This can help reduce noise and speed up the learning process. As was implemented in other benchmark models [28], The Stochastic Gradient Descent (SGD) algorithm updates network weights utilizing the crossentropy loss function.

Note that the overall open form of the cost function in the following is equivalent to Eq. 6. The cost function incorporates the cost of supervised loss from labeled clean and labeled adversarial examples and neighbor losses. That is, the cost includes different neighbor types/edges, as shown in Fig. 5. Formally,

$$\mathcal{L}_{GReAT}(\Theta) = \sum_{i=1}^{L} \ell(\theta, \mathbf{x}_i, y_i) + \sum_{i=1}^{L} \ell(\theta, \mathbf{x}'_i, y_i),$$

+ $\lambda \left[\alpha_{11} \sum_{i=1}^{L} w_a \mathbf{D}(g_{\theta}(x_i), \mathbf{N}(g_{\theta}(x_i))) + \alpha_{12} \sum_{i=1}^{L} w_b \mathbf{D}(g_{\theta}(x_i), \mathbf{N}(g_{\theta}(x'_i))) + \alpha_{21} \sum_{i=1}^{L} w_c \mathbf{D}(g_{\theta}(x'_i), \mathbf{N}(g_{\theta}(x_i))) + \alpha_{22} \sum_{i=1}^{L} w_d \mathbf{D}(g_{\theta}(x'_i), \mathbf{N}(g_{\theta}(x'_i))) \right],$

where w_a, w_b, w_c, w_d represent the similarity weights between the samples and their neighbors calculated by cosine similarity measurement.

The similarity weights are (possibly) unique for each sample and its neighbors, with a range of zero to one. A sample and neighbor candidate are dissimilar if the similarity weight is near zero. For calculating the neighbor loss, we use **D** as it represents the distance between a sample and its neighbor, where we use the norms L_1 and L_2 as distance metrics for calculating the neighbor distance. The hyperparameters $\alpha_{11}, \alpha_{12}, \alpha_{21}$ and α_{22} control the contributions of the different types of edges. For simulations, we set all α s as one to include all edges in the training. The new objective function makes SGD possible with clean and adversarial examples and their neighbors in mini-batch training.

D. COMPLEXITY ANALYSIS

The proposed method incorporates graph regularization into its training process, applying it to both labeled and unlabeled data instances within the graph, which includes benign and adversarial examples. The computational complexity of each training epoch is dependent on the number of edges in the graph, denoted as E_c . To assess the complexity of the training, we can express it as $O(count(E_c))$. It is important to note that the quantity E_c is directly proportional to several factors. First, it scales with the number of neighboring data points taken into account, indicating that more neighbors will increase the complexity. Second, it is influenced by a parameter that determines the selection of the most similar neighbors, further impacting the computational load. Furthermore, the step size used for adversarial regularization is tied to E_c .

For instance, if we opt for a single-step adversarial regularization method like FGSM, each clear example will have only one adversarial neighbor. However, when employing a multi-step adversarial regularization approach, the number of edges substantially increases, as adversarial examples are generated at each step. This type of PGD-based adversarial regularization tends to enhance the model's robustness compared to FGSM regularization. Nevertheless, it introduces a trade-off between robustness and training time. Training a model with PGD regularization demands more computational resources because of the increase in the number of edges and samples involved. This trade-off is essential when choosing the appropriate adversarial regularization method for a given application. For our simulations, we used FGSM to create adversarial examples for training and testing stages to reduce computational time.

IV. EXPERIMENTS

We conducted experiments to show the performance of the proposed GReAT method. Each experiment is carried out on clean data sets with a fixed number of epochs and training steps. The typical hyperparameters are fixed to ensure fair comparisons with other state-of-the-art methods. The base CNN model is trained and then regularized with the proposed loss function. We use the copy of the base model to obtain the regularized model each time to preserve the original base model. Once the models are trained, we test each model on the same clean and adversarially perturbed test data to measure the generalization and robustness performances.

A. DATASETS

The Canadian Institute for Advanced Research dataset (Cifar10) [38], The Street View House Numbers (SVHN) [49], and flowers [37] dataset are used to evaluate the methods. The Cifar10 dataset consists of 60,000 images with ten classes, and each class contains a fixed size of 32×32 three-channel RGB images. To further assess the robust generalization capabilities of our proposed method, we conduct evaluations using the SVHN dataset. SVHN comprises 73,257 training samples and 26,032 testing samples. The flowers dataset contains 3,670 images with five classes, each containing highresolution RGB images. The image sizes are not fixed in the flowers dataset. Resizing is, therefore, required as one of the pre-processing steps.

The image distributions of each class are balanced for both data sets. We split the dataset 80%-10%-10%, as trainvalidation-test data sets, respectively. In the simulations, we use the flowers dataset only for the ablation study since the state-of-the-art methods on the benchmark do not use flowers. In the ablation study, we reduce the training set to 20%, and 50%, to observe the model performances with fewer labeled samples.

B. PRE-PROCESSING STEPS

A few essential pre-processing steps are required to prepare the batches for training. After creating image embeddings, we measure the similarity between each embedding and create training batches based on this similarity metric.

1) Similarity measure

Identifying the closest neighbors for a given sample requires the measurement of similarity amongst the embeddings. Various metrics are available, including Euclidean distance, cosine similarity, and Structural Similarity Index Measure (SSIM). We have opted for cosine similarity due to its proven effectiveness in quantifying the similarity of image embeddings within a multidimensional space. Formally defined, the cosine similarity of two vectors can be expressed as follows:

$$Cos(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| * \|x_j\|}.$$
 (10)

The similarity weights are between 0 and 1, depending on the angle between the two vectors. Two overlapping embeddings have weight 1 when the angle between the two embeddings is zero. Conversely, if two embedding vectors are orthogonal, they are dissimilar, and the similarity weight is zero. Once all similarity weights are calculated, the most similar neighbors are identified as candidates for regularization. We pre-define a similarity threshold to consider those neighbors. Embeddings that fall under the threshold are not considered as neighboring candidates on the graph.

2) Training batches

Once the graph structure is created with clean samples and adversarial examples, we generate training batches that are fed into the neural network model. Each training batch consists of samples, their neighbors, and adversarial neighbors. The number of neighbors is predetermined, although other strategies can be utilized. In our simulations, we pick the number of neighbors as two.

3) Adversarial Examples Generation

In the training phase, we adhere to the conventional configuration of the ℓ_{∞} threat model with a radius of 8/255. Adversarial examples are crafted using the PGD attack, iterating 10 steps with a size of 2/255 for all datasets.

C. NETWORK

For our experiments, we utilize ResNet-18 [50] as the default baseline model architecture. We employ the SGD optimizer for training all models, with a momentum of 0.9 and weight decay set to 0.0005. The training batch size is set to 128. We use the same baseline model architecture and training parameters with other methods for a fair comparison. Additionally, we included the GReAT model trained with Adam optimizer to the benchmark. We noticed that using the Adam optimizer performs better than SGD during our experiments. The Adam optimizer with a 0.001 learning rate is utilized.

D. RESULTS

1) Ablation Study

As mentioned earlier, we use the flowers dataset for the ablation study. Because the dataset is relatively small, we deployed a smaller network architecture for the ablation study. This training model consists of 6 convolution layers and max-pooling layers. Dropout and batch normalization layers in the base model are deployed to minimize over-fitting. Subsequently, experiments are conducted on both clean and adversarially perturbed datasets to gauge model

generalization and robustness. We set the perturbation magnitude to 0.2 in these experiments and employ the FGSM attack method. We compare the results with the standard AT model [19] and standard graph regularized model [43] to show the performance improvements with varying training set sizes. The performance of the proposed method, along with other methods on the clean testing dataset, is summarized in Table 1.

TABLE 1.	Clean Accuracy	Results for	Flowers Data Set
----------	----------------	--------------------	------------------

Model Accuracy						
train set(%)	Base	NSL	AT	GReAT adv	GREAT	
20%	0.548	0.553	0.525	0.207	0.550	
50%	0.564	0.608	0.575	0.245	0.659	
80%	0.597	0.613	0.583	0.277	0.671	

Table 1 indicates that the NSL approach, [43], yields significantly better performance. This is largely due to its training on clean samples and their respective neighbors. For a comprehensive evaluation, we introduced $GReAT_{adv}$, which trains only adversarial examples and their neighbors to assess the impact of adversarial regularization. Given its exclusive focus on adversarial examples during training, this model faces challenges when tested on clean datasets. However, the proposed GReAT method consistently yields positive results.

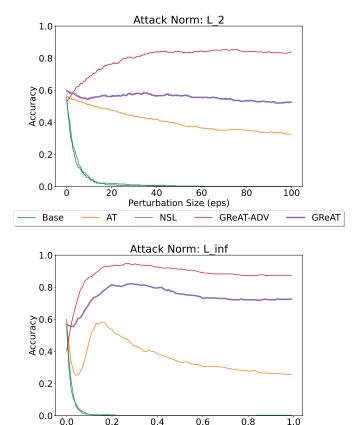
TABLE 2. Robust Accuracy for Flowers Data Set

Attack Norm	train set(%)	Model Accuracy				
		Base	NSL	AT	$\mathbf{G} ReAT_{adv}$	GReAT
L2	20%	0.011	0.011	0.450	0.836	0.605
	50%	0.014	0.024	0.496	0.854	0.647
	80%	0.016	0.063	0.526	0.891	0.668
Linf	20%	0.001	0.000	0.727	0.924	0.883
	50%	0.002	0.001	0.753	0.942	0.892
	80%	0.005	0.005	0.819	0.968	0.931

Finally, the models were evaluated using adversarially perturbed test data from the flowers data set, and the results are shown in Table 2. As the table shows, models not trained on adversarial examples, particularly the base model, exhibit diminished performance. Although the NSL model is trained only on clean samples, it still exhibits some robustness to adversarially perturbed test samples. The proposed GReAT model outperforms the other models and provides a balanced result for clean and adversarially perturbed testing data. $GReAT_{adv}$ gives the highest accuracy for perturbed test samples. This experiment shows how graph regularization with adversarial training is effective on both adversarially perturbed and clean testing samples.

2) Accuracy vs attack strength

We evaluate the robustness of the proposed methods by adjusting the step size of the perturbations, which provides insights into the model performance under varying attack strengths. As illustrated in Fig. 7, the accuracy of the base model declines sharply with increasing attack intensity. Although the model trained with standard adversarial training also exhibits a notable decrease in confidence, the proposed GReAT model consistently displays significant robustness, retaining its efficacy even under substantial perturbations.





NSL

AT

Perturbation Size (eps)

GReAT-ADV

GReAT

The perturbation sizes for adversarial training samples are 20 for L_2 norm and 0.2 for L_∞ constrained models, respectively. Training samples for L_2 norm trained model are not normalized to one since we follow the training procedures as described in [19]. As the figure shows, the models trained on adversarial examples exhibit peak performance around these specific training perturbation sizes (ϵ). This phenomenon occurs when the attack perturbation step size is smaller than the step size of the adversarial examples used for training [19]. Ideally, we aim to train a model with varying perturbation sizes to enhance its robustness against adversarial attacks, given that attack perturbation sizes might differ from the training perturbation size. However, for the sake of simulation simplicity, we utilize only a single-step perturbation size for the model training.

3) Ablation study on Cifar10 Data Set

Next, we evaluate the performance on the Cifar10 dataset, which is made up of images with lower resolution with more

classes and quantities. Tables 3 and 4 detail the performance on the clean and adversarially perturbed image datasets, respectively. The proposed GReAT methodology yields balanced results, indicating that GReAT demonstrates both generalization and robust performance. In stark contrast, alternative methodologies are significantly impacted by adversarial attacks. These simulation results underscore that regularizing the deep learning model with both benign and adversarial examples results in improved generalization *and* robustness.

TABLE 3.	Clean Accuracy	Results for	Cifar10 Data Set.
	ercun /recuracy	nesults ioi	enality bata bea

Model Accuracy						
train set(%)	Base	NSL	AT	GReAT adv	GReAT	
20%	0.522	0.523	0.296	0.227	0.560	
50%	0.612	0.648	0.437	0.285	0.649	
80%	0.701	0.713	0.688	0.327	0.731	

Compared to other methods, the proposed method shows outstanding performance on the benign testing set. This is because more training data and classes provide more underlying information between classes with graph regularization.

TABLE 4. Robust Accuracy for Cifar10 Data Set.

Attack Norm	train set(%)	Robust Accuracy					
		Base	NSL	AT	$\mathbf{G} ReAT_{adv}$	GReAT	
L2	20%	0.121	0.161	0.343	0.556	0.525	
	50%	0.090	0.172	0.367	0.594	0.567	
	80%	0.135	0.191	0.385	0.651	0.638	
Linf	20%	0.003	0.071	0.415	0.599	0.583	
	50%	0.004	0.079	0.517	0.640	0.626	
	80%	0.004	0.105	0.570	0.694	0.648	

Table 4 provides the performance of each model on adversarially perturbed testing data. As detailed in the table, the proposed method provides superior results to the NSL and standard adversarial training models. We observe similar results for $GReAT_{adv}$ in the Cifar10 data set, which shows the learning ability of GReAT over adversarial deceptive perturbations.

4) Comparison with the State-of-the-art Models

Our proposed method's robust accuracy comparison results and various baseline models are presented under different attack methods, FGSM and PGD-100, on CIFAR-10 and SVHN datasets, using the ℓ_{∞} norm with $\epsilon = 8/255$. All models are based on the ResNet-18 architecture. The best checkpoint is selected based on the highest robust accuracy on the test set.

The benchmark Table 5 illustrates the robustness results for the CIFAR-10 dataset, showcasing the performance of various methods in defending against FGSM and PGD-100 adversarial attacks. Our proposed method notably outperforms existing approaches in terms of robust accuracy. However, it is observed that the natural accuracy achieved by our method is slightly lower compared to some other methods. Despite this, the significant improvement in robust accuracy

Base

TABLE 5. Robust Benchmark under *l_{inf}* type attack for CIFAR-10

CIFAR-10, $l_{inf} = 8/$						
Method	Method Natural Acc FGSM					
AT [19]	82.97	57.77	51.35			
ALP [30]	84.86	57.55	51.57			
TLA [31]	83.49	58.17	51.96			
ACL [32]	83.26	57.54	51.51			
TRADES [29]	83.74	59.54	52.73			
ATA [28]	83.41	57.96	52.39			
TAAT [28]	83.12	59.91	54.45			
GReAT (SGD)	82.64	62.78	60.58			
GReAT (ADAM)	82.89	72.47	71.31			

demonstrates the effectiveness of our proposed method in enhancing the robustness of CIFAR-10 classification models against adversarial attacks.

TABLE 6. Robust Benchmark under *I*_{inf} type attack for SVHN

SVHN , $l_{inf} = 8/25$	SVHN, $l_{inf} = 8/255$, untargeted attack					
Method	Method Natural Acc FGSM					
AT [19]	90.5	65.08	52.87			
ALP [30]	90.67	65.51	54.07			
TLA [31]	90.63	64.66	52.96			
ACL [32]	90.33	63.57	52.07			
TRADES [29]	90.38	73.31	57.94			
ATA [28]	89.11	62.81	53.75			
TAAT [28]	90.44	72.59	59.91			
GReAT (SGD)	90.24	74.47	63.45			
GReAT (ADAM)	90.54	75.81	65.66			

The benchmark Table 6 displays the robustness outcomes for the SVHN dataset, presenting the accuracy of different methods in countering FGSM and PGD-100 adversarial attacks. The proposed method demonstrates superior performance compared to other approaches in terms of robust accuracy.

V. CONCLUSION

In this paper, we have presented a Graph Regularized Adversarial Training Method (GReAT), designed to enhance the robustness of classifiers. By leveraging classical adversarial training, the graph regularization technique enhances the robustness of deep learning classifiers. This technique employs graph-based constraints to regularize the training process, thereby bolstering the model's capacity to withstand adversarial attacks. Integrating these constraints enables the model to learn more robust features and be less prone to manipulation via adversarial examples. This strategy has demonstrated significant potential to enhance the robustness and generalization of deep learning classifiers, indicating that it is a valuable tool in adversarial training.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

IEEE Access

- [2] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings* of the IEEE conference on computer vision and pattern recognition, pages 3367–3375, 2015.
- [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139– 144, 2020.
- [4] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label Propagation and Quadratic Criterion. In Semi-Supervised Learning. The MIT Press, 09 2006.
- [5] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings, May 2016. arXiv:1603.08861 [cs].
- [6] Dong-Hyun Lee. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges* in Representation Learning (WREPL), July 2013.
- [7] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [8] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. Self-training: A survey, 2022.
- [9] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- [10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57, 2017.
- [11] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015.
- [12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.
- [13] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, Dec 2018.
- [14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.
- [15] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In

Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery.

- [16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.
- [17] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physicalworld attacks on deep learning models, 2018.
- [18] Samet Bayram and Kenneth Barner. A black-box attack on optical character recognition systems, 2022.
- [19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.
- [20] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In 2016 IEEE European symposium on security and privacy (EuroS&P), pages 372–387. IEEE, 2016.
- [21] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples, 2017.
- [22] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.
- [23] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An *l*_0-robustsparse fourier transform. *arXiv preprint arXiv:1812.05013*, 2018.
- [24] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991, 2017.
- [25] Bao Wang, Alex T Lin, Wei Zhu, Penghang Yin, Andrea L Bertozzi, and Stanley J Osher. Adversarial defense via data dependent activation function and total variation minimization. *arXiv preprint arXiv:1809.08516*, 2018.
- [26] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE symposium on security and privacy (SP), pages 582–597. IEEE, 2016.
- [27] Hao-Yun Chen, Jhao-Hong Liang, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. Improving adversarial robustness via guided complement entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4881–4889, 2019.
- [28] Huafeng Kuang, Hong Liu, Xianming Lin, and Rongrong Ji. Defense against adversarial attacks using topology aligning adversarial training. *IEEE Transactions on Information Forensics and Security*, pages 1–1,

2024.

- [29] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019.
- [30] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing, 2018.
- [31] Chengzhi Mao, Ziyuan Zhong, Junfeng Yang, Carl Vondrick, and Baishakhi Ray. Metric learning for adversarial robustness. In *NeurIPS*, pages 478–489, 2019.
- [32] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning, 2020.
- [33] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning - ICML* '05, pages 1036–1043, Bonn, Germany, 2005. ACM Press.
- [34] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. Semi-Supervised Learning with Graphs. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, USA, 2005. AAI3179046.
- [35] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006.
- [36] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled Data Improves Adversarial Robustness. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.
- [37] TensorFlow. Flowers, jan 2019.
- [38] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar10 dataset. Technical report, Canadian Institute for Advanced Research, Canada, 2009.
- [39] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.
- [40] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, page 1168–1175, New York, NY, USA, 2008. Association for Computing Machinery.
- [41] Nitin Agarwal, Huan Liu, Sudheendra Murthy, Arunabha Sen, and Xufei Wang. A Social Identity Approach to Identify Familiar Strangers in a Social Network. *Proceedings of the International AAAI Conference on Web* and Social Media, 3(1):2–9, March 2009. Number: 1.
- [42] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the* 7th ACM international conference on Web search and

data mining, pages 373–382, New York New York USA, February 2014. ACM.

- [43] Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. Neural Graph Learning: Training Neural Networks Using Graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 64–71, Marina Del Rey CA USA, February 2018. ACM.
- [44] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with Local and Global Consistency. page 8, 2004.
- [45] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.
- [46] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Canada, 2009.
- [47] Hrushikesh Mhaskar, Qianli Liao, and Tomaso A. Poggio. When and why are deep networks better than shallow ones? In Satinder P. Singh and Shaul Markovitch, editors, AAAI, pages 2343–2349. AAAI Press, 2017.
- [48] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [49] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- [50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.