Machine Learning (2023) 112:2619–2651
https://doi.org/10.1007/s10994-023-06307-y

Reducing classifier overconfidence against adversaries through graph algorithms

Leonardo Teixeira[1] · **Brian Jalaian**[2] · **Bruno Ribeiro**[1]

Received: 15 February 2022 / Revised: 16 November 2022 / Accepted: 23 January 2023 /
Published online: 14 March 2023

Abstract
In this work we show that deep learning classifiers tend to become overconfident in their answers under adversarial attacks, even when the classifier is optimized to survive such attacks. Our work draws upon stochastic geometry and graph algorithms to propose a general framework to replace the last fully connected layer and softmax output. This framework (a) can be applied to any classifier and (b) significantly reduces the classifier's overconfidence in its output without much of an impact on its accuracy when compared to original adversarially-trained classifiers. Its relative effectiveness increases as the attacker becomes more powerful. Our use of graph algorithms in adversarial learning is new and of independent interest. Finally, we show the advantages of this last-layer softmax replacement over image tasks under common adversarial attacks.

Keywords Adversarial robustness · Overconfidence · Gossip algorithm

1 Introduction

In critical applications, it is already hard to be sure if we can trust the predictions of existing neural network models (Guo et al., 2017). The potential presence of an adversary makes it even harder to trust these systems. *Is there a method to create classifiers that are more robust, and less overconfident?* In existing adversarial defenses, we show that the classifier softmax output overestimates the probability its predicted class is correct, specially for powerful attackers (see Fig. 1).

Editors: Krzysztof Dembczynski and Emilie Devijver.

✉ Leonardo Teixeira
lteixeir@purdue.edu

Brian Jalaian
brian.a.jalaian.civ@mail.mil

Bruno Ribeiro
ribeiro@cs.purdue.edu

[1] Department of Computer Science, Purdue University, West Lafayette, IN, USA

[2] U.S. Army Research Laboratory, Adelphi, MD, USA

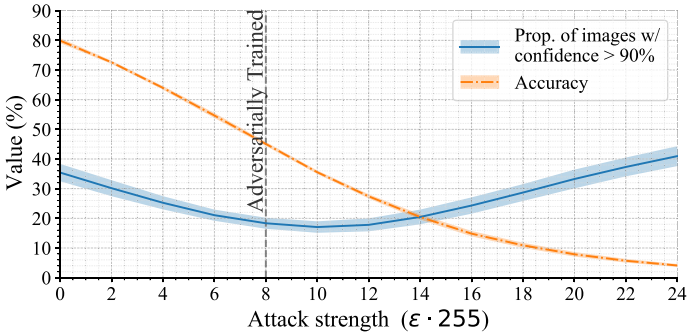


Fig. 1 (CIFAR-10, ResNet-18) Confident predictions of an adversarially trained model (Mądry et al. (2018) with $\epsilon = {}^{8}/_{255}$) surprisingly increases as attacks become stronger

A *robust* classifier must be (approximately) invariant to any transformation of the inputs which doesn't change the true label. Another desirable property is to avoid the overconfidence of its predictions by not overestimating the probability that its predicted class is correct—for example, whenever the classifier's softmax output makes a prediction with confidence of 0.9, then it must be correct at least 9 out of 10 times.

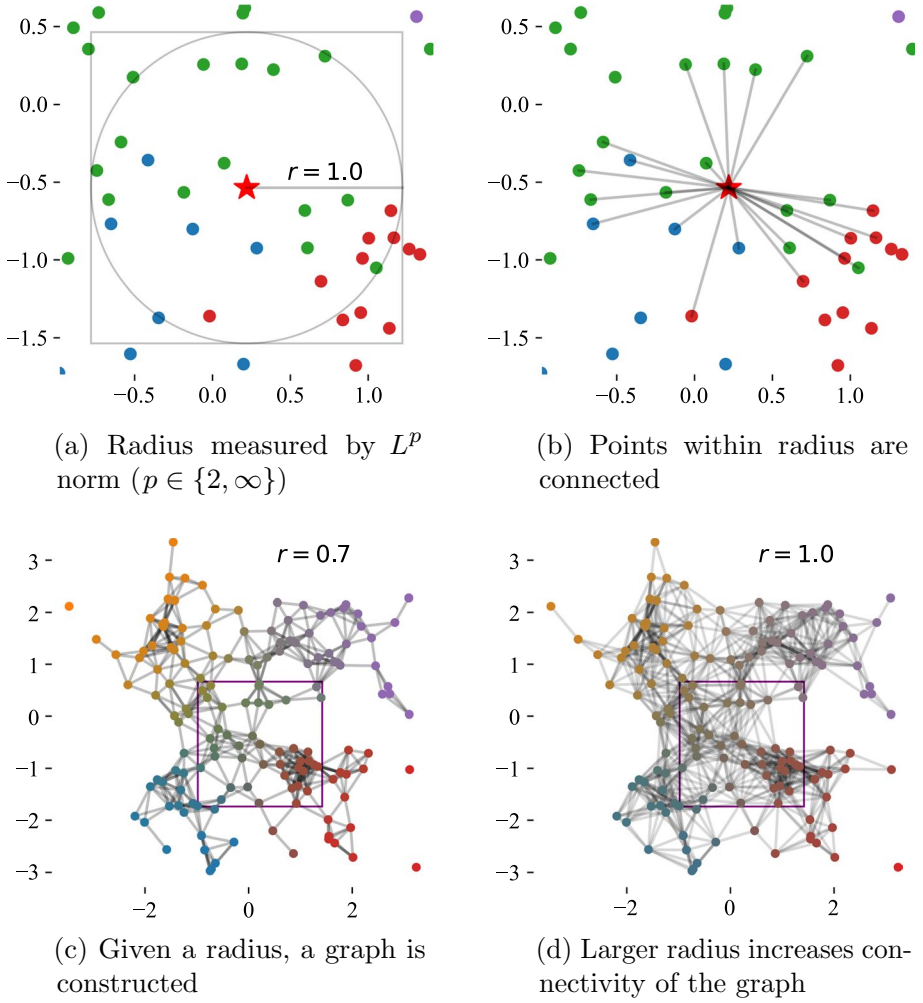
This is particularly important in the context of *adversarial machine learning*, where the neural network's failure to learn the correct input invariances enables an adversary to craft small perturbations in the images which change the classifier's predictions. This has created a perpetual arms race of new methods of crafting *adversarial examples* (Kurakin et al., 2017; Mądry et al., 2018; Carlini & Wagner, 2017b) for which new defenses are proposed (Papernot et al., 2017; Guo et al., 2018; Dhillon et al., 2018), which are then (again) fooled by newer attack methods (Athalye et al., 2018).

In this work, we leverage the relationship between similar images to build a defense that replaces the last layer and softmax output of a classifier with a graph-based method that: (1) significantly reduces overconfidence; (2) survives adversarial attacks stronger than what was seen during training; (3) does not require retraining of the classifier.

Under the assumption that the neural network maps similar examples to nearby regions of the embedding space, we create a graph by connecting images whose representation are close to each other (see Fig. 2). Then, we make use of a Gossip algorithm (Boyd et al., 2005), to diffuse the labels through this graph, which improves the model's predictive uncertainty. Moreover, our defense refuses to give overconfident answers to test images which are nowhere close to the curated validation examples, such as those under strong adversarial attacks.

Note that a direct *calibration* objective under adversarial attacks would be the wrong goal, since calibration with an imperfect classifier requires knowing the distribution of the test data, i.e., knowing what the adversary will do. Calibrating existing machine learning models is a topic which has been explored for decades (Platt, 1999; Niculescu-Mizil & Caruana, 2005; Naeini et al., 2015) for traditional classifiers and recently gained focus in the context of deep learning models (Guo et al., 2017). But these approaches do not target avoiding overconfident predictions and frequently are not developed in the context of adversarial examples (where we observe an unknown distribution shift at test time).

Moreover, since our proposed framework is based on the representations learned by the deep learning model, it can easily be combined with a multitude of current (and future)



(a) Radius measured by L^p norm ($p \in \{2, \infty\}$)

(b) Points within radius are connected

(c) Given a radius, a graph is constructed

(d) Larger radius increases connectivity of the graph

Fig. 2 Images whose representations are close (as measured by L^p norm), are connected, to form a graph. The choice of the radius has a direct impact on the connectivity of the graph, which influences how we diffuse label information, to reduce overconfidence of the predictions

defenses (e.g. adversarial training), without incurring additional cost of retraining the model.

2 Related work

While the field of adversarial examples is full of interesting developments over the past few years, doing a thorough review of the literature is beyond the scope of this work. Instead, we focus on those publications which are more closely related to our work and are more relevant to the discussion we present in this paper.

The study of adversarial examples within the context of deep learning models picked up interest after the work of Szegedy et al. (2013). In the following years, the community started an arms race of developing stronger attacks and developing defenses which are then broken by even stronger attacks (Carlini & Wagner, 2017a). Despite this constant development, some of the simpler methods, such as the Projected Gradient Descent (PGD), developed by Mądry et al. (2018) are still very effective and an important tool to evaluate new defenses and techniques.

While many important improvements have been made in securing deep learning models and making them more robust to adversarial examples, most works in the field focus only on improving accuracy and few works even look at the calibration of the produced classifier. While the topic of calibration has been extensively studied for decades, from the analysis of weather forecasters in the 80 s (DeGroot & Fienberg, 1983), to the standard machine learning classifiers such as SVM (Platt, 1999), Logistic Regression, Naïve Bayes and others (Naeini et al., 2015; Niculescu-Mizil & Caruana, 2005), it was first studied in the context of deep learning by Guo et al. (2017). The Expected Calibration Error (ECE) metric we use was first proposed by Naeini et al. (2015), while the Overconfidence Error (OE)—its counterpart that only considers the miscalibration caused by overconfident predictions—was proposed by (Thulasidasan et al., 2019). The improvement of calibration (and in particular reducing overconfidence) of deep learning models under adversarial attacks is an essential step to obtain a reliable and trustworthy classifier.

Open set recognition and other approaches: a related area of research is focused on the problem setup where unknown (or new) classes are present at test time. See Geng et al. (2020) for a survey of recent advances in this area. Also related are works which not only aim at recongizing that an object is of an unknown class, but also try to continuously learn, by incorporating new classes in successive rounds of training (Dai et al., 2021). Our work, however, focuses on the adversarial setup, where the set of possible classes is fixed and an adversary modifies an input to (incorrectly) change the model prediction.

Adversarial training and other similar procedures: Perhaps, the most common defense against adversarial examples is retraining the classifier, augmenting the training data with adversarial examples (adversarial training) (Goodfellow et al., 2015; Mądry et al., 2018; Wang et al., 2019). Together with adversarial training, are other strategies which seek to harden the classifier by modifying the training procedure, usually by employing a regularization strategy that encourages some type of smoothness in the final model (preventing small perturbations of the input from producing large changes in the final output). Among such methods are Parseval Networks (Cisse et al., 2017), which seek to train networks with low (< 1) Lipschitz constants and Laplacian Networks (Lassance et al., 2021) which use tools from Graph Signal Processing to enforce smooth variations of the class boundaries. These works are not alternatives to our work: they can be combined with our proposed strategy, since we can use the (better) embeddings produced with such methods. In fact, in our experiments, we use the embeddings of an adversarially trained model, since they provide a considerable improvement over vanilla (undefended) neural networks.

Recently, a trade-off between being accurate and being robust has been shown (Tsipras et al., 2019; Zhang et al., 2019), and Hendrycks et al. (2019) show that pre-trained models can be an important step in improving robustness. A better understanding of the the geometry of adversarial examples has emerged: traditional adversarial examples are often *fragile* to random perturbations, that is, a small perturbation added to an adversarial example will revert back to the original class (Roth et al., 2019; Elliott et al., 2021; Hosseini et al., 2019; Hu et al., 2019; Guo et al., 2018). These works can be seen as simple approaches for classifier reliability, since they can output class priors whenever an adversarial example is

detected, even if reducing overconfidence or improving calibration is not their main goal. These perturbations have inspired new defenses (Hu et al., 2019; Gopalakrishnan et al., 2018; Xie et al., 2018). This robustness, however, relies on a weak adversary that can only find fragile adversarial examples. This premise has been recently put into question with trivial changes to the adversary (Hosseini et al., 2019).

Pinto et al. (2021) propose Mix-MaxEnt, a regularization approach to improve uncertainty quantification. Serban et al. (2021) propose Deep Repulsive Prototypes, a modified training proceedure, with a distance-based loss to encourage separation of classes, which is competitive with adversarial training. However, both approaches require re-training and cannot be applied post-hoc.

Other post-hoc defenses: In the realm of post-hoc defences, other works have used the embeddings to construct alternative classifiers, mostly with the use of k-Nearest Neighbor (k-NN) over the pre-trained embeddings. This approach was shown successful in simply improving accuracy in language models (Khandelwal et al., 2020), while for adversarial reliability, k-NNs have inspired defenses using the embeddings of neighbors from training data (Sitawarin & Wagner, 2019a) and embeddings of multiple distinct (Dubey et al., 2019) datasets. Papernot and McDaniel (2018), propose Deep k-NN, which uses k-NN together with principles from conformal predictions to improve reliability. Unfortunately, new attacks have been shown effective against such methods (Sitawarin & Wagner, 2019b). The use of a nearest neighbor approach to find related examples, however, could potentially consider examples very far from each other, since their notion of being related is restricted by quantity of such neighbors, rather than fixing a distance as we do in our proposed method. Moreover, the existing k-NN defenses do not address the potential Palm distribution bias, and performed poorly in our experiments, becoming increasingly overconfident when confronted with stronger attacks.

Recent approaches (Liu et al., 2020; van Amersfoort et al., 2020; Mukhoti et al., 2021) proposed replacements of the final dense layer and softmax output, to promote distance-aware uncertainty quantification, based on Gaussian processes, RBF kernels and discriminant analysis. In contrast to our work, their methods require modifications of the training procedure (e.g. spectral normalization or gradient penalty regularization) and/or architecture and their work focus on improving out-of-distribution detection, rather than investigating strong adversarial shifts. For example, in the work of Mukhoti et al. (2021), if an image is incorrectly identified as *in distribution*, their method will still use the softmax outputs, which may be overconfident.

Recently, Hess et al. (2020) and Wang and Loog (2022) propose post-hoc modifications of the softmax, based on theoretical reinterpretations of the last-layer + softmax output. We provide an analysis of these methods in the Appendix.

Certified defenses, such as Randomized Smoothing (Cohen et al., 2019) have gained attention as they provide theoretical (and practical) guarantees of robustness for perturbations within a given radius around images. But recent attacks have been effective against it (Ghiasi et al., 2020) and, in our experiments, such defense displayed poor calibration and strong overconfidence.

Gossip algorithms: The use of gossip algorithms to perform distributed computations has been widely studied in the context of sensor and computer networks (Boyd et al., 2005), where it first emerged. Gossip algorithms have been used to develop a distributed SGD training strategy for neural networks (Blot et al., 2019). GossipNet (Hosang et al., 2017) is a CNN architecture, which uses concepts from gossip algorithms by doing a message passing operation among neighboring sets of pixels in a single image. While all these works are based on Gossip algorithms, they are unrelated to our approach as they are not

concerned with reducing overconfidence of the classifier and do not build a graph from the datasets (as our approach does). Instead, they are focused on distributed training or on exchanging information within a single image, whereas our use of gossip algorithm is to diffuse label information among similar images on the dataset. Our use of the gossip algorithm, in particular as a defense strategy against adversarial examples is novel.

Graph-based algorithms: Graph based algorithms have been used before in the context of image processing, but most such works build a graph over pixels or regions of a single image (Felzenszwalb & Huttenlocher, 2004; Shekkizhar & Ortega, 2020), with the goal of denoising or performing other (single image) processing operation. In our work, instead, we use the embeddings of images to build a graph that encodes how related (similar) the images are to each other. We also note that while we experimented with a couple of ways of creating such a graph of examples, other methods could easily be incorporated into our proposed gossip framework.

Finally, we want to distinguish our work from those which are focused on attacking graph-structured data and models, such as GNNS (Dai et al., 2018; Bojchevski & Günnemann, 2019; Liao et al., 2020). In this setting, the graph is the input to their models, which produce embeddings used for downstream tasks. The attackers' goal is to extract information from the neighborhood or to influence the embeddings by modifying the input graph (adding or removing nodes or edges).

In our proposed framework, the input data is not graph-related—we build the graph ourselves based on the mapping produced by the neural network. And our graph is constructed from a curated set of validation examples, which an adversary cannot modify.

3 Classifier calibration and uncertainty quantification preliminaries

A classifier is said to be *well-calibrated* when its assessment of its own uncertainty is accurate, that is, the confidence of its predictions (e.g. softmax values) matches its accuracy (i.e. among predictions made with 80% confidence, 8 out of 10 are correct). When the classifier makes more mistakes than expected (according to the confidence values), it is overconfident.

As shown in Fig. 1, strong distribution shifts (e.g adversarial attacks) can cause models to be exceedingly overconfident (even for models adversarially trained to defend against such attacks). When such models can have their accuracy compromised by adversarial attacks, it is desirable to at least have a reliable assessment of its uncertainty, by not being overconfident, so we are not misled into making bad decision due to wrong confidence estimation.

In this work, we assess the calibration of models with the Expected Calibration Error (ECE) (Naeini et al., 2015; Guo et al., 2017), where we bin predictions based on their confidence values and average the difference between accuracy and confidence of each bin. Another important metric is the Overconfidence Error (OE), derived from ECE, where we only consider the bins for which the confidence is higher than the accuracy. More formally, consider a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ and denote by $B_1, \ldots, B_m$ the m bins of the predictions, based on their confidence. We compute the metrics as:

$$ECE = \sum_{b=1}^{m} \frac{|B_b|}{N} |\text{conf}(B_b) - \text{acc}(B_b)| \tag{1}$$

and

$$OE = \sum_{b=1}^{m} \frac{|B_b|}{N} \left[\text{conf}(B_b) \times \text{overconf}(B_b)\right], \tag{2}$$

where $\text{conf}(B_b)$ and $\text{acc}(B_b)$ are the average confidence and accuracy of the examples in bin B_b and $\text{overconf}(B_b) = \max\left(0, \text{conf}(B_b) - \text{acc}(B_b)\right)$.

4 Methodology

Having introduced the necessary concepts in the previous section, we now focus on addressing the hypothesis: *If we incorporate concepts of stochastic geometry to build a relational method, to be applied on top of the embeddings produced by a pre-trained neural network, we can reduce overconfidence of the predictions, when compared to the original fully-connected + softmax predictions, in particular for adversarial attacks stronger than used in training.*

We seek to leverage the representational power of the neural network by extracting *embeddings* of the set of *clean* validation images and using them to build a graph-based estimator, under the assumption that similar images will be have their embeddings mapped close to each other.

We denote the neural network architecture by $f(x) = h(z(x))$, where $z(x)$ is the output after all the convolutional layers (a d dimensional vector), and $h(\cdot)$ is the final dense layer and softmax. We replace $h(\cdot)$ with a graph-based estimator constructed based on the distance between the embeddings $z(x)$ of validation examples. This graph is then used in a label diffusion algorithm to obtain an estimation of $P(Y \mid X)$ for validation images, with reduced overconfidence. At test time, our predictions are based on this new estimation of the label distribution for the relevant validation images (i.e. validation images whose embeddings are close to the test image).

4.1 Building a graph from embeddings of validation examples

As our method is based on proximity of embeddings of the images, we have found that the smoothness of $z(x)$ has a direct impact on the accuracy of our defense. See Sect. B.2 for a discussion of limitations of this assumption. For this reason, we extensively employ adversarially trained embeddings in our experiments, and argue that any improvement that can produce more robust embeddings can be combined with our approach to improve its robustness.

We build a graph from the embeddings of (clean) *validation* examples, by treating each validation image as a node in our graph and connecting the images whose embeddings are within a radius r of each other (illustrated in Fig. 2). In what follows, we denote by $\mathcal{D}^{(\text{vl})} = \{(x_i, y_i)\}_{i=1}^{N^{(\text{vl})}}$ the set of $N^{(\text{vl})}$ validation images and their labels. First, for each image x_i, we identify all other *neighbor* images within the radius r:

$$\mathcal{N}_r\big(z(x_i)\big) = \{j \in [N^{(\text{vl})}] : \|z(x_j) - z(x_i)\| \leq r\}, \tag{3}$$

as shown in Fig. 2a. We then add the edges (i, j) connecting that node with each of its neighbors in $\mathcal{N}_r\big(z(x_i)\big)$, as seen in Fig. 2b.

The choice of the radius r can impact on the connectivity of the graph we build, which then impacts the diffusion of label that we will perform based on this graph. As

the radius increases, the graph becomes more connected, as seen in Fig. 2c, d. We tried two strategies to pick the radius, based on ensuring a connected graph or a minimum edge density (what fraction of all possible edges to include in the graph). We observed similar performance across both strategies. For completeness, in Sect. 5.3 discuss the results obtained with each strategy.

4.2 Diffusion of labels via Gossip algorithm

To obtain a smoother and less overconfident estimation of $P(Y|X)$, we diffuse the (one-hot encoded) true label signal through the graph, so that, for each example, we obtain a new distribution that incorporates the label information from nearby examples as well. We will denote by $p(k|x_i;r)$ this new estimation of $P(Y = k \mid X = x_i)$, for the validation example x_i, based on the graph of radius r.

The final outcome of this label diffusion, however, needs to fulfill two important desirable properties:

1. For each validation example, $p(\cdot|x_i;r)$ must describe a valid probability distribution over the classes:

$$\sum_{k=1}^{C} p(k \mid x_i;r) = 1.$$

2. The diffusion process must not change the marginal distribution of labels:

$$\frac{1}{N^{(\mathrm{vl})}} \sum_{i=1}^{N^{(\mathrm{vl})}} p(k|x_i;r) = \frac{1}{N^{(\mathrm{vl})}} \sum_{i=1}^{N^{(\mathrm{vl})}} \mathbb{1}\big(y_i = k\big).$$

To achieve such properties, we took inspiration in Gossip Algorithms (Boyd et al., 2005), a class of iterative distributed algorithms used for computing averages over networks of sensors or ad-hoc and peer-to-peer networks. At the center of a gossip algorithm is the idea that, at a given iteration, nodes will exchange information with its neighbors and replace its current value with an average of it and the values received from the neighbors. These algorithms are generally focused on reaching a point of equilibrium, but we, instead, focus on the transient phase, performing a finite number of exchanges, so that the influence of an image remains local.

The information being exchanged is the estimated label distribution $p(\cdot \mid x_i;r)$, which we initialize at time $t = 0$ with the one-hot encoding of the true label. Then, at each iteration, the nodes update p, forming the following dynamic process:

$$p^{(t)}(k \mid x_i;r) = \sum_{j \in \mathcal{N}_r(z(x_i))} \big[\Pi_r\big]_{ij} p^{(t-1)}(k \mid x_j;r),$$

with $p^{(0)}(k \mid x_i;r) = \mathbb{1}\big(y_i = k\big)$, and with $\big[\Pi_r\big]_{ij}$ denoting the weight that node i gives to the information coming from j. Section 4.2.1 describes how to construct $\Pi_r \in^{N^{(\mathrm{vl})} \times N^{(\mathrm{vl})}}$ from the graph structure.

In a more general sense, if we stack the vectors $p(\cdot \mid x_i;r)$ into a matrix $p(\mathcal{D}^{(\mathrm{vl})}, r)$, we can describe the final values, after M rounds as the following matrix multiplication:

$$p^{(M)}(\mathcal{D}^{(\mathrm{vl})}, r) = \Pi_r^M \boldsymbol{Y}_{OH}, \tag{4}$$

where $\boldsymbol{Y}_{OH} \in^{N^{(\mathrm{vl})} \times C}$ is the matrix built by stacking the one-hot encoding of the labels of each node of our graph. As before, the matrix Π_r describes, at each round, how node i incorporates the information coming from its neighbor j. With the matrix description of Eq. (4), we can implement the diffusion with matrix operations, with time complexity of $\mathcal{O}(MN^{\mathrm{vl}}C)$, which can be accelerated with a GPU. By its nature and origin, it is scalable for large datasets due to its distributed nature.

If we choose a large enough value of M in Eq. (4), it will converge to the average of the labels in the dataset (the marginal distribution $P(Y)$). While this would be perfectly calibrated, having the same (prior) distribution for all validation points is undesirable. Instead, we stop the gossip algorithm before convergence, allowing for some diffusion of the label information through the nearby examples and this number of rounds M, which we treat as a parameter to be tuned, gives us control of how much we want to smooth the label information over the dataset.

4.2.1 Construction of the diffusion matrix Π_r

An integral part of the method is the matrix Π_r, which controls how the label diffusion is done and how fast it converges. As per Eq. (4), this matrix controls how each node averages the value of p coming from the other nodes, at each round. One required property of Π_r is for it to be *doubly-stochastic*, i.e., Π_r's rows and columns both sum to one. This constraint ensures that the total probability that a given label y is predicted matches the frequency of label y in the validation data. Using a doubly stochastic matrix also guarantees the two desired properties described in the previous section.

A natural choice is to use the adjacency matrix, as it already encodes the relationship between the validation examples. To make it a doubly-stochastic matrix, we apply the Sinkhorn-Knopp algorithm (Sinkhorn & Knopp, 1967), which iteratively normalizes the rows and columns of the matrix. The advantage of this approach is that the exchange of labels happens only between neighbors in the graph. The time complexity of this step is $\mathcal{O}(kn^2)$ where k is the number of iterations and n is the size of the graph (usually $k \ll n$). Since it is implemented with matrix operations, this step can also be accelerated by using a GPU.

We also experimented an alternative strategy to construct Π_r using Personalized PageRank, as a way to offload some of the label diffusion to the PageRank computation. In Sect. 5.3 we compare this PageRank approach with the simpler approach of applying Sinkhorn-Knopp to the adjacency matrix. In practice, both performed similarly.

We also note that this method puts an excessive emphasis on the information coming from neighbors, and little reliance on the original signal of the node itself. To counter this effect, we add a final step of reinforcing the diagonal entries in the produced doubly-stochastic matrix Π'_r, by mixing it with an identity matrix: $\Pi_r = \rho I + (1 - \rho)\Pi'_r$. The lazy diffusion parameter $0 \leq \rho < 1$ controls how much we should trust the original label of the node itself versus how much to rely on the gossip diffusion process and is tuned as a hyperparameter.

4.3 Making predictions at test time

The graph-building and gossip diffusion we described above needs to be performed only once. After computing the $p(k|x_i;r)$ for all the validation examples, the predictions for a test example x are much simpler: First we identify which validation points would be neighbors of x in the graph $\mathcal{N}_r(z(x))$. If this set is empty, we simply output the marginal $P(Y)$.

When the set of neighbors $\mathcal{N}_r(z(x))$ is not empty, we output the average of their estimated distributions:

$$\hat{p}_{\text{Avg}}(\cdot|x;r) = \frac{1}{|\mathcal{N}_r(z(x))\ |} \sum_{i \in \mathcal{N}_r(z(x))} p^{(M)}(\cdot|i;r).$$

Alternatively, we also investigate an attempt to counter-act a potential finite-sample bias of $z(x)$ being more likely closer to high-degree nodes in the graph, if the test x is a clean image (since high-degree nodes are in denser regions of $P(X)$). In this case, we apply the Horvitz-Thompson estimator (Horvitz & Thompson, 1952), which corrects for estimator biases known up to a constant:

$$\hat{p}_{\text{HT}}(\cdot \mid x;r) = \left(\sum_{j \in \mathcal{N}_r(z(x))} \frac{1}{d_j} \right)^{-1} \sum_{i \in \mathcal{N}_r(z(x))} \frac{p^{(M)}(\cdot \mid x_i;r)}{d_i},$$

where d_i is the degree of node i. This version estimates the label distribution of the region defined by the hyper-sphere of radius r around the test point—the relative ratio of degrees among the validation points in this region (the neighbors of the test point) reflect their sampling probability.

The time complexity of making predictions at test time (given the pre-computed graph and the embeddings of test images) is bounded by the cost of computing the distance to validation points, which is $\mathcal{O}(N^{\text{vl}}d)$ and can be done in parallel. In our experiments, making predictions for a test set of 8000 images of size $3 \times 32 \times 32$ takes less than a second on an NVidia GeForce 1080 Ti GPU, comparable to other defenses we used.

4.4 Estimator consistency, Gilbert graphs, and palm calculus

In what follows we prove the consistency of our estimator in Eq. (4). We also show that the graph we built in Sect. 4.1 is a Gilbert graph (Penrose, 2003), whose connectivity properties follow that of continuum percolation, with the main result over a compact set $A \subset \mathcal{X}$ given by Baccelli and Błaszczyszyn (2009).

In order to prove this property, we first introduce a few concepts from stochastic geometry (see Daley and Vere-Jones (2007) for a reference). Proposition 1 below proves that our validation image embedding dataset $\{(z(x_i), y_i)\}_{i=1}^{N^{(\text{vl})}}$ can be described as $N^{(\text{vl})}$ samples of a Marked Poisson Point Process (MPPP), where the embedding $z(X)$ has an associated mark Y (the image's class).

Proposition 1 *If the $\mathcal{D}^{(\text{vl})}$ and $\mathcal{D}^{(\text{te})}$ datasets are MPPPs, and the embedding function $Z : \mathbb{R}^d \to \mathbb{R}^{d'}$ is injective, then the embedding datasets $\mathcal{D}_Z^{(vl)}$ and $\mathcal{D}_Z^{(te)}$ are also MPPPs. Finally, if the derivative $z(x)$ exists around the infinitesimal ball $\lim_{r \to 0} B_r(z(x))$, the number of neighbors $\mathcal{N}_r(X)$ in the $B_r(z(X))$ ball, $X \in \mathcal{D}^{(\text{vl})} \cup \mathcal{D}^{(\text{te})}$, is Poisson distributed with*

intensity proportional to $\lambda(X)|\det \frac{\partial z}{\partial x}|$, *where* $\lambda(X)$ *is the spatial distribution density of a clean image X in the validation data.*

The proof is in the Appendix. It then follows directly that our graph is a Gilbert graph.

Another important property is the distribution of embeddings in the neighbors $\mathcal{N}_r\big(z(x_i)\big)$ of validation image x_i, $i = 1, \ldots, N^{(\text{vl})}$, which is a Palm distribution. A *Palm distribution* (Palm, 1943) of a point process is the conditional distribution of the point process given one of its points z_0 at the origin, denoted $P^0_{z_0}$. The proof of Proposition 1 (in Appendix) also shows that the Palm distribution $P^0_{z_0}$ of the clean image embeddings observed around a validation example embedding z_0 is the true distribution of clean image embeddings as if the validation point were not there.

Finally, we use these results to show that the estimator in Eq. (4) is consistent.

Proposition 2 *Let* $p(y \mid z)$ *be uniformly continuous on* z, *that is* $\forall \epsilon > 0,\ \exists\ \delta > 0$ *such that* $\forall z_1, z_2$ *with* $\|z_1 - z_2\| \leq \delta, |\ p(y \mid z_1) - p(y \mid z_2)\ | \leq \epsilon$. *Then,* $p^{(M)}$ *in Eq.* (4) *converges to* $p(Y \mid x_i)$ *for a sufficiently large number of validation examples* $N^{(\text{vl})} \to \infty$ *and an appropriate choice of radius* $r > 0$ *in Eq.* (3) *and matrix power* $M \geq 1$ *in Eq.* (4).

The proof is in the Appendix.

Why we cannot construct the graph with training data

The consistency result is no longer true if we build our graph with the training data in lieu of the validation data. The neighboring image embeddings of a training example may be geometrically distorted (i.e., the Palm and true distributions can be different), since the embeddings no longer would form an MPPP (since $z(\cdot)$ depends on the training data). In summary, we should never use training in lieu of validation data to build our graph.

4.5 Choosing hyperparameters

For a given graph and corresponding diffusion matrix, we tune two remaining parameters for our gossip approach: the number of rounds M and the lazy diffusion parameter ρ. We considered values for M in the range [10, 200] and for ρ between (0, 1), in a grid search. We evaluated the configurations under a PGD attack with $\epsilon = {}^{8}\!/_{255}$, removed redundant ones using a Pareto frontier and selected the best configuration using an utility function that balances accuracy, calibration and the need to perform too many rounds. More extensive discussion can be found in the Appendix.

5 Results

5.1 Experimental setup

Dataset and architecture: We follow a similar experimental setup as Mądry et al. (2018), on image classification in the CIFAR-10 dataset (Krizhevsky & Hinton, 2009), using the ResNet-18 (He et al., 2016) architecture, adversarially trained using the procedure from Mądry et al. (2018) with $\epsilon = {}^{8}\!/_{255}$, which we denote as AT. We employ a 5-fold cross validation scheme, described in detail in the Appendix. These models were trained using NVidia GeForce 1080 Ti GPUs with PyTorch (Paszke et al., 2019), with learning rate of

10^{-2} and batch size of 64. Our code will be made available after acceptance. More details about the training and the resources used can be found in the Appendix.

Attacks: We evaluate the defenses under a collection of L^∞ attacks: Fast Gradient Sign Method (FGSM) (Szegedy et al., 2013), Projected Gradient Descent (PGD) (Mądry et al., 2018) and the method from Carlini and Wagner (2017a), using the implementation from the Adversarial Robustness Toolbox (ART) (Nicolae et al., 2018). For each image we select the attacked version that was most successful at fooling the defense being evaluated. We vary the maximum allowed perturbation between $^2/_{255}$ and $^{24}/_{255}$ by steps of $^2/_{255}$. The second type of attack we used is LaVAN (Karmon et al., 2018), an attack where only a small square patch of the image is changed, without restriction of how much noise is added to it. We varied the side size of the square patch from 2 to 16 pixels, by multiples of 2. More details about this can be found in the Appendix.

Note that, during training, the model is only exposed to a PGD attack with $\epsilon = {}^8/_{255}$, but in the evaluation, it is presented not only with stronger versions of the same attack, but also with different types of attack all together. We aim to prepare our defense to the *worst-case*, reducing its overconfidence even for such stronger and unforeseen attacks.

Other defenses: Besides the baseline adversarially trained (AT) model (with the original dense layer + softmax), we also compare Temperature Scaling (Guo et al., 2017), which improves calibration by adding a temperature to the logit values. We also tried a variation of temperature scaling where we set the temperature arbitrarily high (infinite), which we denote by TS_∞, to investigate a simple alternative that simply forces the predicted probabilities to resemble an uniform distribution.

We also tested two defenses which use the (adversarially trained) embeddings for k-Nearest Neighbor approaches: a simple version that averages the one-hot encoded label of neighbors and Deep k-NN (Papernot & McDaniel, 2018), which combines k-NN with conformal predictions that incorporates the dissimilarity of the test example when compared with validation and training examples. Due to the excessive memory requirements of Deep k-NN when using the outputs of all layers (42 GiB for our setting), we use only the outputs of a single layer, the same one we use as embeddings in our method, similar to the approach by Sitawarin and Wagner (2020). Finally, we also include *softRmax* (Wang & Loog, 2022), a polynomial version of softmax, aimed at reducing overconfidence.

5.2 Discussion of the experiments

Both types of attack we use have a simple knob that can be used to change the *strength* of the attack: the allowed L^∞ norm of the distortion (ϵ) or the size of the patch. We used these knobs to study how the defenses cope against a wide range of attack strengths, in particular for strong attacks which the models didn't see in training. We present our main results in Figs. 3, 4 and in Sect. 5.3 we include more results using variations of the graph and diffusion matrix, which perform similarly. We use the ECE and OE metrics (Eqs. (1) and (2)) as well as accuracy to evaluate the defenses, with a focus on *overconfidence*, as is the major cause of miscalibration under medium and severe attacks, and a great cause of concern in many applications.

To guide our analysis, we identify three regimes of the attack's strength: mild, medium and severe attacks. We emphasize that these thresholds are just visual aids to simplify our exposition. We do not claim to have a general definition of attack *strength*, but rather selected thresholds that helps describe the patterns we observe.

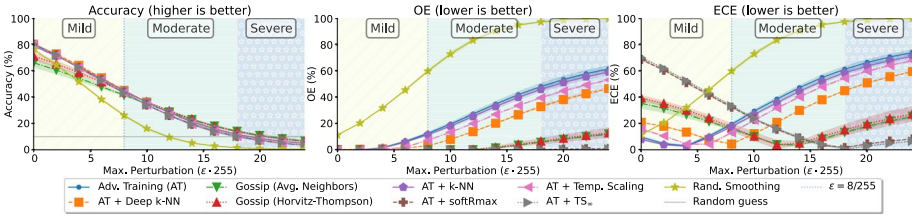


Fig. 3 Performance of defenses over varying strength of standard L^∞ attacks (worst case between PGD, FGSM and CW). Our approach is particularly effective at reducing overconfidence for stronger attacks

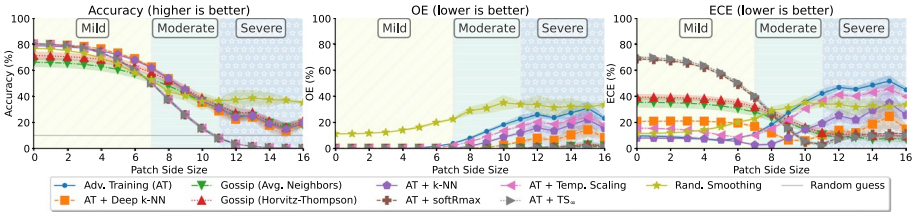


Fig. 4 Performance of defenses over varying sizes of patch attack (LaVAN). While other defenses show worse calibration as patches get larger, our approaches display lowest OE and ECE

5.2.1 Mild attacks

We name *mild attacks*, those with $\epsilon < {}^{8}/_{255}$ (the value used for adversarial training) or patch sizes smaller than 7, which we mark in the left region of Figs. 3 and 4. We observe that only randomized smoothing is significantly overconfident in this range, while most other methods display their best ECE values in this range. Since our method has the goal of reducing overconfidence (which is confirmed by achieving 0% OE), most of the miscalibration we present in this range is due to being underconfident, as we tend to give more conservative predictions than other methods. This comes at the cost of a drop in accuracy under the clean images, but as the attacks get stronger, we see similar accuracy as the other defenses.

The only defenses which reach as low OE as we do are: temperature scaling with infinite temperature (AT + TS_∞ in the plots) and softRmax. However, TS_∞ has the most uninformative answers (akin to random guess), which means it also has the higher miscalibration, precisely for that reason, showing around 80% ECE, even for clean images, 40% worse than our approach. And softRmax also suffers from the same problem (too underconfident).

5.2.2 Moderate attacks

In this range, defined by ${}^{8}/_{255} < \epsilon \leq {}^{18}/_{255}$ and patch size between 7 and 11 pixels, we start seeing attacks stronger than what was used during training, but not too strong to render the defenses worse than random guess. For the patch attacks, it is during this range that adversarially trained models (and temperature scaling variants) start to underperform as compared to others, as we see in Fig. 4 where our accuracy is almost 4 times higher and in Fig. 3 where OE and ECE is about 30% lower for our approach.

All defenses increase their overconfidence during this regime, except for our approach, TS_∞, and softRmax, as seen in Figs. 4 and 3. In particular, Randomized Smoothing is the

most overconfident, reaching close to 100% OE for L^∞ attacks (cf Fig. 3). Most defenses display between 6 and 8 times worse OE when compared with our gossip approach.

While this increase in overconfidence is reflected in the increase of ECE for most defenses, the progressive strengthening of the attacks starts justifying the reduction in overconfidence achieved by our approach, which is why we reach the lowest ECE values towards the end of this range for L^∞ attacks (Fig. 3), making it better calibrated for the moderate-severe attack regime.

5.2.3 Severe attacks

In this last range ($\epsilon > 18/255$ and patch size larger than 11 pixels), the attacks are so strong that most defenses' accuracy becomes worse than that of a random guess, as seen in Fig. 3. For LaVAN (Fig. 4), we see the original AT defense and its temperature scaling variants drop to worse-than-random accuracy, while all others sustain better accuracy, with Randomized Smoothing as the most accurate, but also the most overconfident.

The calibration in this range, however, tells an opposite story. While a true random guess would show 0% ECE, most defenses reach upwards of 70% ECE in this range. The benefits of our approach are even more remarkable in this range, as we have much better ECE and OE than other defenses. In particular, for L^∞ attacks (Fig. 3), our approach is up to 50% less overconfident than other competing approaches. Other defenses are the most confident when they are making the most mistakes, while the gossip approaches show a significant reduction in the overconfidence everywhere (See the Appendix for statistical significance tests).

While for moderate and severe attacks our method's accuracy is very much in par with other defenses, we observe a drop in accuracy for the mildest attacks, which we attribute to our intent on being conservative in our predictions. The parameters of our gossip algorithm can be tuned to improve accuracy at the expense of worse calibration and less reduction of overconfidence. If this trade-off is important for the user, our method provides the knobs to tune it.

We also observe that variants using the Horvitz-Thompson estimator show better accuracy but worse calibration for milder attacks, but similar performance to their average counterpart for stronger attacks. Under moderate to severe attacks, all our variants are still considerably more calibrated than other defenses.

Lastly, we hypothesize that the localized nature of the patch attacks does not entice large shift in the embedding, which is why all defenses which are based on the geometric proximity of the embeddings (k-NN methods and our gossip approaches) cope better than other defenses. We also note that our reduction in overconfidence is still significant, even when compared to these k-NN inspired defenses.

5.3 Further evaluation

In this section, we present further results evaluating the performance of the proposed defence based on different strategies to the radius for Eq. (3) and for building the diffusion matrix Π_r. As expressed before, we observe similar results across all configurations, indicating that our proposed method is somewhat robust to variations in the graph used in the diffusion process.

For each type of attack (L^∞ and LaVAN), we tested two strategies for choosing r: (1) choosing r that ensures a minimum edge density (we targeted density of either 15% or 25%

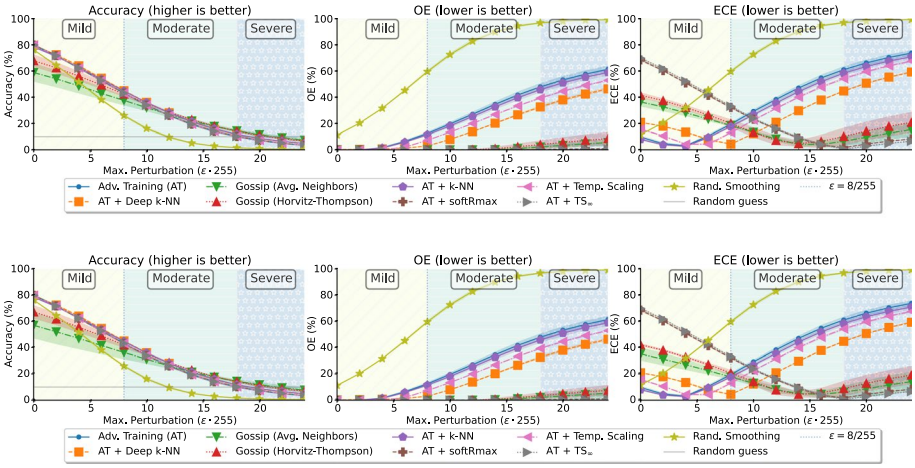


Fig. 5 Results for L^∞ attacks, with graph constructed using Single Connected Component strategy. Top is without excess radius, bottom is with 10% excess radius

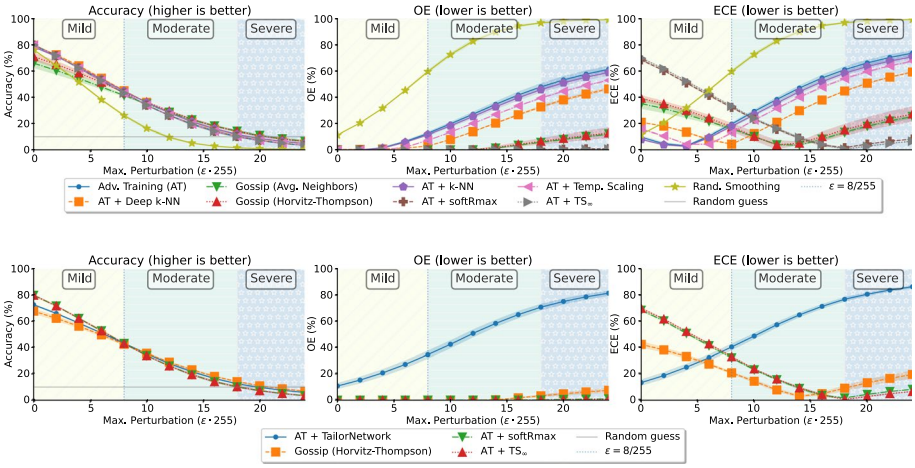


Fig. 6 Results for L^∞ attacks, with graph constructed using Edge Density strategy. Top is with target density of 15%, bottom is 25% density

of all possible edges); and (2) ensuring the resulting graph has a Single Connected Component (with and without a potential increase of the computed radius by 10%, which we named "excess radius"). This gives us a total of 4 settings. In each set of plots we include results using both versions of the diffusion matrix (based on Adjancency matrix or Personalized PageRank), as well as employing the Horvitz-Thompson estimator or not.

Figures 6 and 5 display the results for L^∞ attacks, while Figs. 8 and 7 contain the results for the LaVAN attack. The main behavior we observe is similar across all the

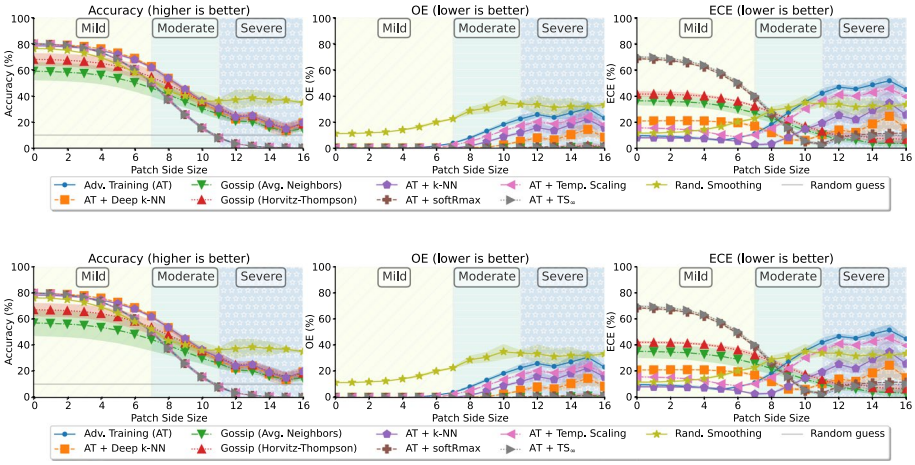


Fig. 7 Results for LaVAN attack, with graph constructed using Single Connected Component strategy. Top is without excess radius, bottom is with 10% excess radius

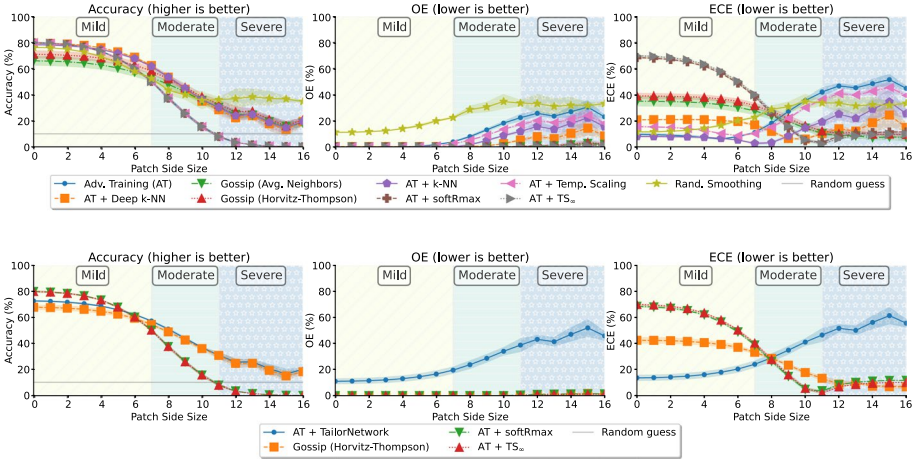


Fig. 8 Results for LaVAN attack, with graph constructed using Edge Density strategy. Top is with target density of 15%, bottom is 25% density

tested graphs. Within each strategy, the version with increased connectivity (density of 25% and excess radius of 10%) tend to show slightly lower accuracy for clean images. The increased connectivity would make the diffusion converge faster, leading to more reduced confidence levels, which could explain the lower accuracy for clean images. This drop, however, is not observed for stronger attacks and the improvements in calibration are equally remarkable, no matter what graph was used. Similarly, the versions with PPR diffusion matrix tend to show lower accuracy under mild attacks but also achieve lower calibration error for stronger attacks.

6 Conclusions

In this work we showed that existing defenses against adversarial examples, when challenged by progressively stronger attacks, get more confident about their predictions as they make more mistakes (overconfidence). We then developed a new graph-based framework to replace the last fully-connected layer + softmax with a new estimator, whose predictions rely on a gossip graph algorithm that diffuses the label from clean validation examples (nodes) to other such nodes in the graph, collectively reducing overconfidence while preserving the correct marginal distribution of labels after such diffusion.

We showed the effectiveness of our framework in drastically reducing overconfidence when compared against existing defenses, particularly under stronger attacks. Our proposed framework can be easily combined with any existing (and future) adversarial defenses, as long as these produce embeddings which we can use, and we also have access to validation data not used to learn the embedding function. We believe the use of collective graph-based predictions to defend against adversarial examples, and its understanding through stochastic geometry, brings new tools to tackle key challenges in adversarial machine learning.

Limitations: Given an embedding, the diffusion process over the Gilbert graph improves upon existing softmax, tempered scaling, and K-NN outputs. However, the approach proposed in this work depends on how well existing embeddings can cluster images of the same class under adversarial attacks.

Appendix A: Theory

Our quest to build reliable classifiers will take us through important concepts in stochastic geometry, which we introduce next. For the reader interested in a deep-dive, a good set of references are Daley and Vere-Jones (2007), Baccelli and Błaszczyszyn (2010), and Chiu et al. (2013).

Stochastic geometry primer A point process (p.p.) is a stochastic process composed of binary events that occur in a continuous space, which we assume to be some high-dimensional space $\mathbb{R}^d$, $d > 1$. Note that w.l.o.g. we can represent images as d-dimensional vectors. These events induce a counting process N that measures how many events (e.g., points, images) we observe in a region $B \subseteq \mathbb{R}^d$. A Marked Point Process (MPP) is obtained when each of these sampled events X also has an associated mark Y. For us, Y will be the class of image X.

We call N a Poisson p.p. if it satisfies the following properties:

Definition 1 (Poisson point process) Let λ be an absolutely continuous measure over subsets of $\mathbb{R}^d$, $d > 0$. A Poisson process with intensity measure λ is a point process N on $\mathbb{R}^d$ with the following two properties:

(**i**) For every $B \subset \mathbb{R}^d$ the number of events in B, defined as $N(B)$, is distributed according to a Poisson with parameter $\lambda(B)$, that is, $P(N(B) = k) = \text{Poisson}(k;\lambda(B))$ for all $k \in \mathbb{N}_0$.

(**ii**) For all collections of $m \geq 2$ *pairwise disjoint* sets $B_1, \ldots, B_m \subset X$, the random variables $N(B_1), \ldots, N(B_m)$ are independent.

Property (i) of Definition 1 is the reason for the name *Poisson process*. The Poisson process' usefulness to point processes is similar to that of Gaussian distributions to traditional statistics. This comes from the fact that most naturally occurring point processes are Poisson due to the Palm-Khintchin convergence theorem:

Theorem 1 (The Palm–Khintchin theorem (informal)) *The superposition of n independent i.i.d. nonstationary point processes in an area* $B \subset \mathbb{R}^d$, $d > 0$, *with intensities* $\lambda(B)/n$ *converges in distribution to a nonhomogeneous Poisson process with arrival rate function* $\lambda(B)$ *as* $n \to \infty$.

The Palm–Khintchin theorem allows us to claim that a dataset of images sampled i.i.d. from a large number of real-world image generation processes is approximately distributed according to a nonhomogeneous Poisson process. Hence, we will assume that the samples $\mathcal{D}^{(\text{tr})}$, $\mathcal{D}^{(\text{vl})}$, and $\mathcal{D}^{(\text{te})}$ come from a Marked Poisson Point Process (MPPP).

A Poisson p.p. is a special type of p.p., since it remains a Poisson p.p. even after an injective mapping $Z : \mathbb{R}^d \to \mathbb{R}^{d'}$, $d' \geq 1$, is applied to the points.

Theorem 2 (Mapping theorem (Kingman , 1993, Chapter 2.3)) *The transformation of the Poisson p.p. of intensity measure* λ *by an injective function* $Z : \mathbb{R}^d \to \mathbb{R}^{d'}$, $d' \geq 1$, *is a Poisson p.p. with intensity measure*

$$\lambda_Z(B) = \int_{\mathbb{R}^d} \mathbb{1}(Z(x) \in B)\lambda(dx), \quad \forall B \subseteq \mathbb{R}^{d'}.$$

The study of point processes also has a peculiarity first described by Palm (1943), from which its theory, Palm Calculus, takes its name.

Definition 2 (Palm distribution (informal)) The *Palm distribution* of a point process is the conditional distribution of the point process given one of its points z_0 at the origin, denoted $P^0_{z_0}$. For instance, if we consider the distribution of images around a given image, that distribution is a Palm distribution.

Palm distributions can be complex statistical objects to study, except when the point process is Poisson, thanks to the Slivnyak–Mecke theorem:

Theorem 3 [*Slivnyak–Mecke Theorem* (*Daley & Vere-Jones*, 2007)] *Let N be a Poisson p.p. with intensity measure* λ. *For almost all* $z_0 \in \mathbb{R}^d$,

$$P^0_{z_0}(N(B) = k) = P(N(B) = k), \quad \forall B \subset \mathbb{R}^d,$$

that is, *the Palm distribution of the Poisson p.p. is equal to its* (*original*) *distribution*.

In what follows we prove that neural network embeddings of the validation $\mathcal{D}^{(\text{vl})}$ and test $\mathcal{D}^{(\text{te})}$ examples are also guaranteed to form a Poisson p.p., while the embeddings of the training data have no such guarantees. We then use this fact to design our graph that takes into consideration the Palm distributions of the embedding MPPP.

Validation and test embeddings as MPPPs

Our graph building approach hinges on the application of the Slivnyak–Mecke theorem (Theorem 3) over the validation and test embedding datasets. This application, in turn, requires the embedding datasets to be MPPPs. Thankfully, Proposition 1 shows that these conditions hold if the embedding function $Z(\cdot)$ is injective:

Proof The first part of Proposition 1 is a consequence of the Mapping theorem (Theorem 2). We excluded $\mathcal{D}^{(\text{tr})}$ from Proposition 1 since the function Z depends on $\mathcal{D}^{(\text{tr})}$, which then creates dependencies between the mapping of the points in $\mathcal{D}^{(\text{tr})}$, violating condition (ii) in Definition 1 of a Poisson p.p.. Since, by Definition 1, ${\mathcal{D}^{(\text{vl})}}_Z$ and ${\mathcal{D}^{(te)}}_Z$ are independent of $\mathcal{D}^{(\text{tr})}$, the dependence of Z on the training data does not affect the mapping of the validation and test datasets.

To find the density of the MPPP, Theorem 3 allow us to state that the distribution of the number of neighbors around a validation/test image (point) X does not change if we have a validation/test example X or not (by Slivnyak–Mecke's theorem and the fact that the embeddings are also a MPPP).

Now, the mapping theorem (Theorem 2) states that the average density of the Poisson process in the ball $B_r(z(X))$ is equal to the density $\lambda(\{x' | z(x') \in B_r(z(X))\})$ of images whose embeddings fall into $B_r(z(X))$. If $z(x)$ is differentiable w.r.t. x, this is proportional to $\int_{x \in \{x' | z(x') \in B_r(z(X))\}} |\det \frac{\partial z}{\partial x}| \lambda(\partial x)$ from the calculation of the area for a change of variables. □

Gossip estimator consistency

Proof Choose any $\epsilon > 0$. Since $p(y|z)$ is uniformly continuous on z, obtain a radius δ such that $\forall z_1, z_2$ with $\|z_1 - z_2\| \leq \delta$, $|p(y|z_1) - p(y|z_2)| \leq \epsilon$. Let $r = \delta / M$, so that any M-length path in our graph falls from a test or validation example X falls within a δ-sized ball around $z(X)$. Assume $|\mathcal{N}_r(z)| > 0$ as $N^{(\text{vl})} \to \infty$ (we will later prove this is true almost surely), so that $\tilde{p}^{(M)}$ in Eq. (4) is well-defined. These two properties are enough to show that $\tilde{p}^{(M)}$ in Eq. (4) yields $|p(y|z(X)) - \tilde{p}^{(M)}| \leq \epsilon$, since by construction all validation examples X' that are M hops away from X in our graph will satisfy $\|z(X) - z(X')\| \leq \delta$.

Combining Theorem 3 and Proposition 1 we have that $|\mathcal{N}_r(z(X))|$ follows a Poisson distribution with intensity $\lambda_Z(B_r(z(X))) = \int_{\mathbb{R}^d} \mathbb{1}\big(z(x) \in B_r(z(X))\big) \lambda(dx)$. All we need now is to make sure $N^{(\text{vl})} \to \infty$ increases fast enough w.r.t. the decrease in radius $r \to 0$ in order to guarantee that $\lambda_Z(B_r(z(X))) \to \infty$, which implies $P(|\mathcal{N}_r(z(X))| = 0) \to 0$. This can be guaranteed by ensuring that $N^{(\text{vl})} = \Omega(r^{-d'})$, which implies that the number of validation examples in an area of the ball $B_r(z(X))$ increases as the radius r decreases. □

Appendix B: Choice of problem setup

Why not just out-of-distribution detection?

Since much of our focus is on being resilient and reliable, particularly when presented with images from a distribution other than what we had access during training, it is natural to

think of the similarities of our task with out-of-distribution detection (OODD) and covariate shift adaptation (CFA).

However, there are fundamental aspects that drive our task and these other methods apart. For instance, while methods have been developed for covariate shift adaptation that seek to improve calibration (Wang et al., 2020), it is commonly assumed to have access, during training, to sample of images from the target (test) distribution. In our task we don't assume the access to the test data and, while we can (and do) use PGD to produce adversarial images (for training or parameter selection), we test our methods under a shift not seen in training, be it a stronger version of the attack or even a completely new type of attack (LaVAN). Therefore, we have to do the best we can with the samples we have from the training and validation data and rely on the robustness of our geometrically-based graph approach to deliver a more reliable prediction under completely unseen new distributions.

When performing OODD, it is common to have some score which is used to discriminate between in and out of distribution, such as a likelihood ratio (Ren et al., 2019) or a test-statistic (Roth et al., 2019) and then choose a threshold which is used to separate the trusted "in-distribution" images from the potentially attacked "out-of-distribution" images. Choosing such parameters, as well as choosing what to do with images detected as out-of-distribution comes with its own set of challenges. For instance, once an example is detected as borderline between in and out of distribution, it is unclear which label probabilities should be assigned to it. Another example is the method of (Mukhoti et al., 2021), which uses the original softmax for in-distribution images, which could lead to overconfident answer if a shift is not detected.

In contrast, by its nature, our approach can seamlessly handle such out-of-distribution examples. By grounding our prediction on the proximity with validation examples, images which are out of distribution will be far from the clean validation data images and, thus, will receive only our most conservative (and uncertain) predictions, reflecting the fact that they are not similar to what the model was trained to handle.

Why adversarially trained models?

In our evaluation, we use adversarially trained models as the underlying classifier on top of which we apply the defenses (both our gossip defense as well as other methods). Models trained with vanilla SGD (undefended models) are very fragille: PGD attacks can easily reduce the accuracy to 0. Not only are AT models more robust, but they are also smoother functions, when compared to undefended models, which also means that, under attack, the embeddings of the images will not be moved too far from their clean counterpart and, as a result, will still be close to the validation data, as we show in Fig. 9.

Recall that our gossip method produces predictions by comparing test point with nearby validation points. Therefore, one limitation of our method is the assumption that the images (even under adversarial attack) will be mapped to nearby regions on the embedding space. This is not the case for undefended models, where the adversarial images can be mapped to distant regions on the embedding space. The implication is that, for undefended models, the attacked images will all be far from the (clean) validation images and, thus, will all receive a random (from the prior) prediction, which is trivially well calibrated.

Therefore, for an undefended model, the reduction in overconfidence is coming not from the gossip algorithm, but from the fact that our defense would give random predictions for the attacked images. Note that this would have higher accuracy—random predictions would be correct for $\frac{1}{C}$ of images. While this means our defense can still be less overconfident than

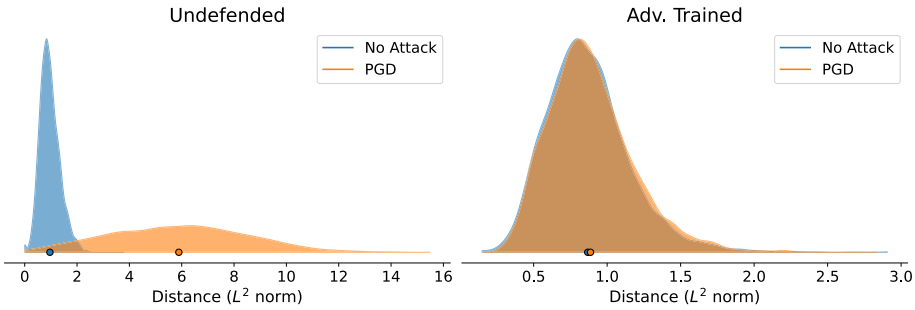


Fig. 9 Distribution of distance from each test image embedding to the closest validation embedding. Adversarially trained models are smoother, so even under attack the images are still closer to the validation data, so our defense remains useful

the alternatives, this use case is less interesting due to the low accuracy and the fact that it does not exploit the label diffusion aspect of our defense.

The choice of using adversarially trained models allows to: (1) illustrate how our defense can be combined with (and make use of) existing and future training techniques; (2) be competitive, in terms of accuracy; (3) show the impact of the gossip step, which reduces the overconfident estimations of $p(y \mid x)$.

Appendix C: Adversarial attacks and threat model

Adversarial attacks can be classified based on the assumed threat model. A common division is according to the amount of knowledge the attacker has about the model: a *white-box* attack has full knowledge of the model and its parameters and can, for example, compute exact gradients with respect to the inputs. *Black-box* attacks, on the other hand, are unaware of model parameters and usually are restricted to querying the model with images, obtaining either the predicted label only or an associated confidence score (e.g. softmax values). Furthermore, adversarial examples are usually described as imperceptible changes in the input images. However, such notion of "imperceptible" is often replaced in practice by a constraint on the magnitude of the modification, usually measured by the L^p norm of the difference, that is $\|[\|p]x - x_{\text{adv}} \leq \epsilon$.

In this work, we consider *white-box* attacks, measured by L^∞ norm. In particular, we employ the following attacks: Fast Gradient Sign Method (FGSM) from Goodfellow et al. (2015); Projected Gradient Descent (PGD) from Mądry et al. (2018); a version of the method proposed by Carlini and Wagner (2017b) which controls for the L^∞ norm; and LaVAN, a patch attack proposed in Karmon et al. (2018).

We make a distinction between the standard L^∞ norm attacks (FGSM, PGD, CW) and the patch attack, which does not limit the L^∞ norm, but can only change a limited (square) patch in the image. For the L^∞ attacks, we use a per-image worst case in our evaluation, that is, for each test image, we run all three methods and keep the adversarial image that fools the evaluated defense (if more than one image leads to misclassification, we break the tie by choosing the most wrong one—i.e. the one for which the defense gives highest confidence for the wrongly predicted class). Note that all these attacks are unaware of any defense added on top of the original classifier, so the images computed for the adversarially

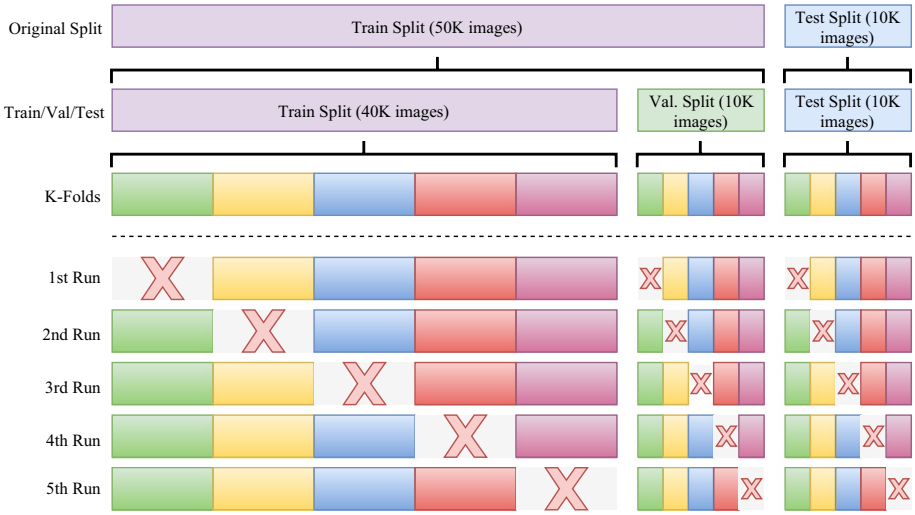


Fig. 10 Cross validation scheme used. We respect the standard Train/Test boundaries, but further split the train data to obtain a validation set and use k-fold split to produce the subsets used in each run

trained model will be the same used for the k-NN methods and or our proposed gossip methods.

For FGSM, PGD and CW attacks, we used the implementation from the Adversarial Robustness Toolbox (Nicolae et al., 2018), while for the LaVAN attack, we used our own implementation (provided with our code), since no official implementation is available. For the L^∞ attacks, we vary the maximum allowed perturbation ϵ between $^2/_{255}$ and $^{24}/_{255}$, by multiples of $^2/_{255}$, and 20 iterations for the iterative attacks. For LaVAN, we vary the patch size between 1×1 and 16×16, and use a maximum of 1500 iterations.

Appendix D: Dataset, pre-processing and cross-validation

In our experiments, we use the CIFAR-10 dataset[1] (Krizhevsky & Hinton, 2009), readily available via the `torchvision` package distributed with PyTorch (Paszke et al., 2019). The dataset is composed of 60,000 colored images (3 channels, 32 ×32), distributed along 10 classes, originally split into 50,000 training images and 10,000 test images. We further randomly split the original training set into train (80%) and validation (20%) sets, as we use the validation set to perform early stopping, tune the Temperature Scaling method and to build the graph we use in our gossip-based framework.

We wanted to provide a measure of variation from our experiments, but at the same time respect the original train/test boundary. To achieve both, we employed the following 5-fold cross-validation procedure (illustrated in Fig. 10): for each of the train/validation/test set, we divide it into $k = 5$ folds: $\{\mathcal{D}^{(\text{tr})}{}_1, \cdots, \mathcal{D}^{(\text{tr})}{}_5\}$, $\{\mathcal{D}^{(\text{vl})}{}_1, \cdots, \mathcal{D}^{(\text{vl})}{}_5\}$, $\{\mathcal{D}^{(\text{te})}{}_1, \cdots, \mathcal{D}^{(\text{te})}{}_5\}$. Then, for the i-th run ($i \in \{1, 2, 3, 4, 5\}$), we exclude the i-th fold from each split, for example, the first run will have train, validation and test folds 2 to 4. We independently run our

[1] Hosted at: https://www.cs.toronto.edu/~kriz/cifar.html.

experiments for each of the 5 runs, from training the model to hyperparameter tuning of defenses, to evaluation of the test images (clean and attacked). In our results, we average over the 5 runs and display the corresponding standard deviation as error bands.

Appendix E: Implementation details

Graph building

An important aspect of our method is the choice of the radius r, used to build the graph. We experimented with two strategies for choosing r, which we describe next. In our experiments, both strategies yielded similar results, indicating that our framework is robust (to some extent) to the structure of this graph. Further exploration of graph-building strategies and their potential impact when used in our approach is left as future work.

Both strategies we tested center around choosing the smallest radius r that satisfy some desired property. The first strategy targets a given edge density, that is, what fraction of all possible $\binom{N^{(\mathrm{vl})}}{2}$ edges do we want to include in the graph. The results presented in the main paper use this strategy. The second strategy seeks to build a graph with a single connected component. In both cases, we run a bisection search to find the first value $r \in_{>0}$ that satisfy the given condition. Each strategy has one parameter, specified by the user: the desired edge density in the first case and, for the second case, a value $\alpha > 0$, which we call *excess radius*, used as follows: Once we determined the minimum radius r' that makes the resulting graph a single connected component, we increase the radius $r = r'(1 + \alpha)$, to increase connectivity and avoid potential bottlenecks in the graph (e.g. regions connected by single edge).

The time complexity of this step is dominated by the cost of evaluating the desired property (e.g. edge density) during the bisection. For both strategies, we first compute all pairs of distances, which has cost of $\mathcal{O}(n^2 d)$. The bisection procedure will take at most $\mathcal{O}(\log n)$ steps, and at each step we threshold the distance matrix to build the graph and check the target property (e.g. edge density), costing $\mathcal{O}(n^2)$ per bisection step, bringing the total cost to $\mathcal{O}(n^2 \log n + n^2 d)$. If we assume $d \ll \log n$, we obtain the final time complexity of $\mathcal{O}(n^2 \log n)$.

For storage cost, we keep all the embeddings of validation points, since, at test time, we need them to identify the images which are neighbors to the test point, bringing the cost to $\mathcal{O}(nd)$, which in our experiments (CIFAR-10 and $d = 256$) makes this 12 times smaller than the cost of storing the original validation images. Exploring cheaper or faster alternatives to storing and searching nearest neighbors is left as future work. For our experiments (CIFAR-10 dataset), the runtime of our defense is comparable with the other tested defenses (except for randomized smoothing, which requires multiple forward passes).

Diffusion matrix

For the Gossip diffusion process, any doubly stochastic matrix is a valid choice. However, as we seek to run the gossip algorithm for a finite number steps and not let it converge, it is desirable top build a diffusion matrix that maintains the diffusion process local.

We experimented with two strategies. In the first, we get the adjacency matrix and turn it into a doubly stochastic matrix by employing the Sinkhorn-Knopp algorithm (Sinkhorn

& Knopp, 1967). This algorithm performs successive row and column normalization (dividing rows/cols by their sum), in an alternating fashion, for a fixed number of iterations (stopping early if convergence is achieved). This algorithm can utilize a GPU and, in our experiments, for a 5000×5000 matrix, it takes about 2 s (using GPU) to compute the Sinkhorn-Knopp algorithm. By definition, the resulting diffusion matrix will have the same graph topology as the adjacency matrix (same nodes will be connected).

The second strategy involves computing a Personalized PageRank (PPR) vector for each node. This is a variation of PageRank (Brin & Page, 1998) in which a restart sends the random surfer back to node i instead of uniformly choosing a node in the graph (see Gleich (2015) for a good review of PageRank and its applications). After computing the PPR vector for each node, we stack them in a matrix and then apply the Sinkhorn-Knopp algorithm. In this strategy, the matrix Π_r is dense and does not conform to the adjacency structure of the graph. It can be seen as offloading part of the smoothing process to the PPR, and then treating the data as a complete graph where the gossip exchange is controlled by the PPR values (which carry over some of the notion of locality due to the personalization).

Usual implementations of personalized PageRank commonly support a single personalization vector. In our case, we want a different personalization vector for each node (a one-hot encoding of the vector, which makes the surfer return to the original node). This means that we need to compute n PageRank vectors, since each version (starting from a different node) defines a distinct Markov Chain for the PageRank algorithm. However, we implement the personalized PageRank algorithm ourselves, as a message passing algorithm, with the help of the DGL library (Wang et al., 2019), which can use GPUs to accelerate the computations. Since this is implemented as an iterative message passing algorithm, we can store and compute all n PageRank vectors simultaneously, much more efficiently than alternative implementations available in other graph libraries.

In our graph with n nodes, computing the n PageRank vectors, using a Nvidia GeForce 1080 Ti GPU, takes around 12 s. Once the PageRank vectors are computed, the Sinkhorn-Knopp algorithm is applied as described above.

In our experiments, both alternatives performed equally well, so in the main paper we restrict our discussion and results to the version using the adjacency matrix. Similarly, our theoretical results (e.g. consistency of estimators) were developed with that version in mind (e.g. if the graph is fully connected, assuming $p(Y|X)$ to be uniformly continuous might be less reasonable than the case where the neighborhood of the point reflects the nearby images).

Gossip algorithm

Once we obtained the diffusion matrix, we adjust its values by reinforcing the diagonal, as described in Sect. 4. Then, we perform M rounds of the gossip algorithm, which amounts to raising the diffusion matrix to the M-th power and multiplying the result by a matrix with the stacked one-hot encoding of the labels of the validation examples. This, again, can be accelerated by using a GPU and in our experiments (5000 node graph), takes less than half a second to run.

After running the gossip algorithm, we obtain the final estimation of $P(Y|X)$ for the validation examples. This new estimation (a $n \times C$ matrix) and the validation points is all that needs to be stored to be used at test time. For the test predictions, we need to compute the distance between the test points and the validation points, to determine which validation points are the neighbors of each test point. If using the Horvitz-Thompson estimator, we compute a matrix of weights for these neighbor points based on their degrees, which is

then used to average the computed $P(Y|X)$ (see Sect. 4.3). All these steps can be computed with vectorized matrix operations that can make use of a GPU. In our experiments, making predictions for a set of 8000 test points, using the GPU, takes less than half a second.

Training specification

To expand on our experimental setup, we implemented our code in PyTorch (Paszke et al., 2019) and used Nvidia GeForce 1080 Ti GPUs to accelerate the training. We fixed the learning rate to 10^{-2} and batch size of 64. We use SGD with momentum factor of 0.9 and weight decay of 5×10^{-4}. We train for 200 epochs and perform early stopping, keeping the model with lowest validation loss. For the Randomized Smoothing defense, we train the model from scratch, adding independently sampled Gaussian noise with $\sigma = 0.25$ to the training images at each step, as recommended in Cohen et al. (2019).

Choosing hyperparameters

There are a set of parameters and decision to be made in our approach: how to build a graph, how to obtain a doubly-stochastic diffusion matrix from the graph, the lazy diffusion parameter ρ, the number of gossip rounds M to perform and the strategy to make test predictions.

In our experiments, we explicitly try both our graph construction methods, as well as the strategies to obtain the diffusion matrix and to make predictions for the test examples, and display the results for each combination. For each combination of those, the two remaining parameters, M and ρ were tuned according to the following procedure.

We perform a grid search, varying M between 10 and 200 and ρ between 0 and 1. For each combination, we use half of the validation examples to build our defense and, for the other half, we compute adversarial examples using PGD with $\epsilon = {}^{8}/_{255}$ and evaluate the defense over this set of attacked images, computing accuracy and calibration (OE).

Choosing the best configuration is not straight-forward for our experiments. Since we are concerned with maintaining lower calibration error, we cannot simply chose the most accurate configuration and disregard calibration. To make matters worse, it is possible to achieve perfect calibration with very low accuracy (when the predictions are like a random guess). We also observed many configurations with similar accuracy and calibration, but with very different number of rounds, for example configurations with 10 and 200 rounds having same accuracy and OE differing by less than 1%. Since doing more rounds of the gossip algorithm leads to probabilities closer to a uniform distribution, we want to avoid selecting configuration with too many rounds if they are not that different from configurations with fewer rounds, since those may give better accuracy under other distribution shift.

To reduce the search space, we discard the Pareto dominated configurations, that is, we keep only those configuration for which no other parameters achieve, at the same time, higher accuracy and lower OE, with fewer rounds. The remaining configurations are the true trade-off that a user must choose from and this would be an ideal point to have a human in the loop. But since we want to automate our tuning, we devised a utility function that balances our goals by assigning the same utility to: reducing OE by 1%, improving accuracy by 2% or performing 50 fewer rounds. This utility function converts each configuration into a utility score, which we then use to pick the best configuration. We found that using this utility score to select he best hyperparameters strikes a good balance between optimizing both accuracy and calibration, while preventing doing too many rounds of the gossip algorithm unless the benefits are very noticeable.

Appendix F: Other softmax re-interpretation

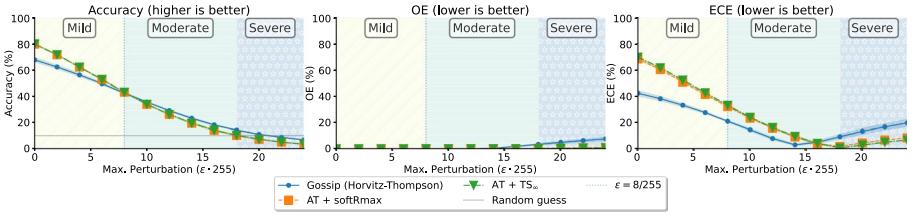
In this section, we compare two recent approaches which propose a post-hoc replacement of the softmax based on a formal re-interpretation of the softmax function applied to the last layer.

Softmax as k-means clustering: The recent work of Hess et al. (2020) shows a formal connection between the softmax predictions and k-means clustering, where the last layer followed by softmax partitions the data into clusters for each of the predicted classes. Inspired by this, they propose to replace the softmax with a k-means approach, computing the clusters centroid for each class (average representation of all datapoints in the class). They name this approach *Gauss networks*.

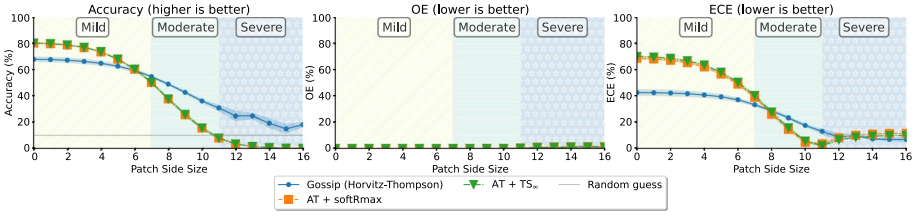
To accommodate for out-of-distribution data, this approach discards the probabilistic interpretation of the output of the neural network. The authors use a confidence value, computed using a Gaussian kernel over the distance to each cluster centroid. This value, however, is not a probability and doesn't sum to one over all the classes, which renders calibrations metrics (like ECE or OE) meaningless, as those metrics are computed from a distribution over the possible classes (i.e. the output of the neural networks sums to 1).

Polynomial version: Another recent work (Wang & Loog, 2022) shows how, by assuming a Gaussian distribution for the class conditional $p(X|y)$, the posterior $p(y|x)$ assumes the well-known softmax function. Inspired by this observation, they replace the Gaussian distribution with a Cauchy distribution, which yields a polynimial version of the softmax, which displays softer decay in the tail, reducing overconfidence. This approach is named *softRmax* by the authors.

In Fig. 11 we compare the softRmax method described above with our gossip algorithm approach, evaluating against both white-box L^∞ attacks and patch attacks. As previously mentioned, the Gauss networks do not produce probabilistic outputs, so the metrics measured (OE and ECE) cannot be computed. For this reason, we excluded the approach from



(a) L^∞ attacks, based on underlying adversarially trained model.



(b) LaVAN patch attack, based on underlying adversarially trained model.

Fig. 11 Comparison between our method and softRmax, another simple post-hoc. While softRmax reduces overconfidence, its behavior is similar to temperature scaling with infinite temperature, in that it loses accuracy and becomes too underconfident (poor ECE)

this analysis. The softRmax approach, however, reduces overconfidence, as claimed by the authors. However, it has an interesting, but unfortunate behavior: similar to temperature scaling with infinite temperature, it has worse ECE (too underconfident) and also reduced accuracy under severe attacks, in particular for patch attacks.

Appendix G: Statistical significance tests

To strengthen our claim of obtaining significantly lower Overconfidence Error (OE), we performed a multiple comparison statistical hypothesis test, using Tukey's Honest Significance Test (HS) Tukey (1949), which is an appropriate statistical test that adjusts for the multiple comparisons. In Table 1 we display the results for severe L^∞ attacks. In Fig. 12 we provide a corresponding visual representation which makes it easier to compare the results. This figure shows the hypothesis test confidence intervals for the Overconfidence

Table 1 Multiple comparison of overconfidence error for L^∞ attacks, using Tukey HS test

Alg. 1	Alg. 2	Mean diff.	p-value	CI	Reject
AT + Deep k-NN	AT + Gossip	−36.97	0.0	(−39.86, −34.07)	True
AT + Deep k-NN	AT + TS_∞	−37.70	0.0	(−40.59, −34.81)	True
AT + Deep k-NN	AT + Temp. Scaling	6.59	0.0	(3.70, 9.48)	True
AT + Deep k-NN	AT + k-NN	12.96	0.0	(10.07, 15.85)	True
AT + Deep k-NN	AT + softRmax	−37.45	0.0	(−40.34, −34.56)	True
AT + Deep k-NN	Adv. Training (AT)	15.24	0.0	(12.35, 18.13)	True
AT + Deep k-NN	Rand. Smoothing	60.17	0.0	(57.27, 63.06)	True
AT + Gossip	AT + TS_∞	−0.73	0.99	(−3.62, 2.15)	False
AT + Gossip	AT + Temp. Scaling	43.56	0.0	(40.67, 46.45)	True
AT + Gossip	AT + k-NN	49.93	0.0	(47.04, 52.82)	True
AT + Gossip	AT + softRmax	−0.48	0.99	(−3.37, 2.40)	False
AT + Gossip	Adv. Training (AT)	52.21	0.0	(49.32, 55.10)	True
AT + Gossip	Rand. Smoothing	97.14	0.0	(94.24, 100.03)	True
AT + TS_∞	AT + Temp. Scaling	44.30	0.0	(41.41, 47.19)	True
AT + TS_∞	AT + k-NN	50.66	0.0	(47.77, 53.56)	True
AT + TS_∞	AT + softRmax	0.24	1.0	(−2.64, 3.13)	False
AT + TS_∞	Adv. Training (AT)	52.94	0.0	(50.05, 55.83)	True
AT + TS_∞	Rand. Smoothing	97.87	0.0	(94.98, 100.76)	True
AT + Temp. Scaling	AT + k-NN	6.36	0.0	(3.47, 9.25)	True
AT + Temp. Scaling	AT + softRmax	−44.05	0.0	(−46.94, −41.16)	True
AT + Temp. Scaling	Adv. Training (AT)	8.64	0.0	(5.75, 11.53)	True
AT + Temp. Scaling	Rand. Smoothing	53.57	0.0	(50.68, 56.46)	True
AT + k-NN	AT + softRmax	−50.42	0.0	(−53.31, −47.53)	True
AT + k-NN	Adv. training (AT)	2.27	0.21	(−0.61, 5.16)	False
AT + k-NN	Rand. smoothing	47.20	0.0	(44.31, 50.09)	True
AT + softRmax	Adv. training (AT)	52.69	0.0	(49.80, 55.58)	True
AT + softRmax	Rand. smoothing	97.62	0.0	(94.73, 100.51)	True
Adv. Training (AT)	Rand. smoothing	44.92	0.0	(42.03, 47.81)	True

Error: overlapping CIs indicate pairs for which the hypothesis (that averages are different) cannot be rejected. Further results are shown in Figs. 13 and 14 for all the 3 regimes for all 3 regimes (mild, moderate and severe) for L^∞ and patch attacks.

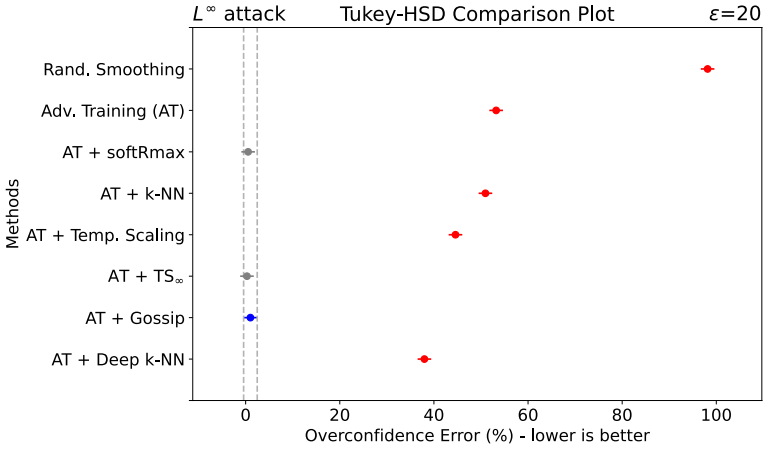
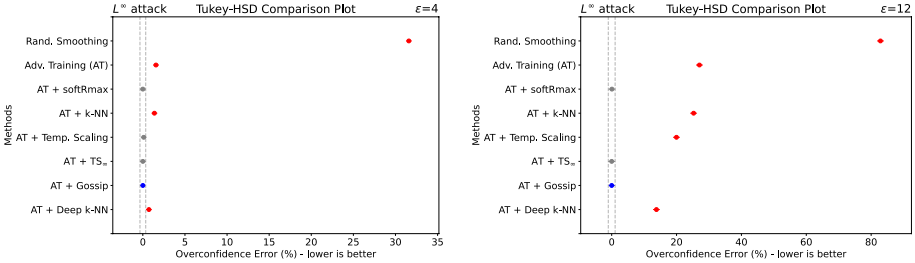
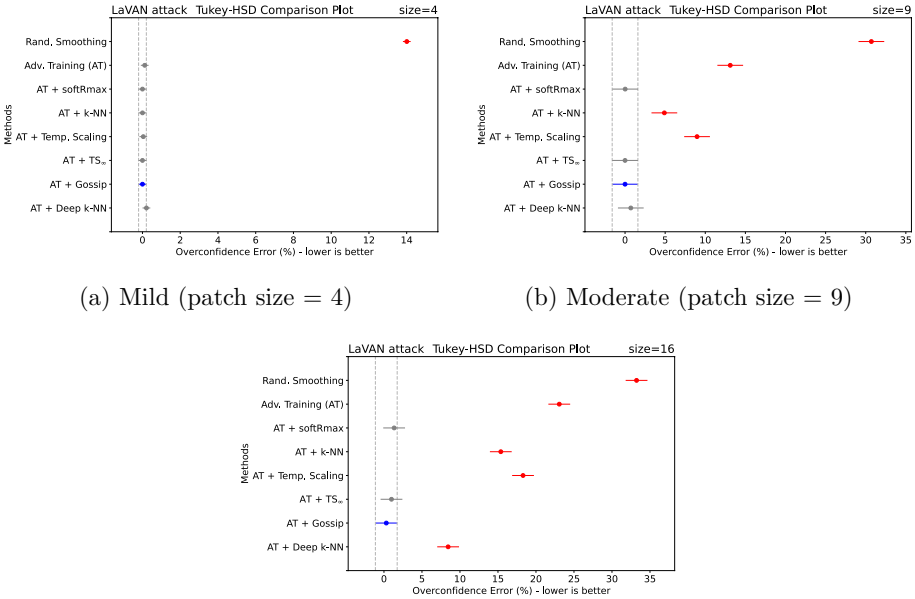


Fig. 12 Visual representation of the Tukey HSD test for severe L^∞ attacks. Highlighted in blue is our gossip approach, with the confidence interval shown. Methods whose CI does overlap represent significant difference (in red) (Color figure online)



(a) Mild ($\epsilon = 4$) (b) Moderate ($\epsilon = 12$)

Fig. 13 Visual representation of the Tukey HSD test for L^∞ attacks. Highlighted in blue is our gossip approach, with the confidence interval shown. Methods whose CI does overlap represent significant difference (in red) (Color figure online)



(a) Mild (patch size = 4)

(b) Moderate (patch size = 9)

(c) Severe (patch size = 16)

Fig. 14 Visual representation of the Tukey HSD test for LaVAN patch attacks. Highlighted in blue is our gossip approach, with the confidence interval shown. Methods whose CI does overlap represent significant difference (in red) (Color figure online)

Author contributions All authors (Leonardo Teixeira, Brian Jalaian and Bruno Ribeiro) contributed to the conception and design of this work. Code development, experiments and analysis were performed by Leonardo Teixeira. The first draft of the manuscript was written by Leonardo Teixeira and all the authors commented on previous versions of the manuscript and participated in reviewing and editing of same. All authors read and approved the final manuscript.

Funding This work was supported in part by the National Science Foundation (NSF) awards CAREER IIS-1943364 and CCF1918483, and by the ARO, under the U.S. Army Research Laboratory contract number W911NF-09-2-0053.

Availability of data and materials The data used in this work is publicly available. Any additional data and material produced as part of our analysis will be made public upon acceptance.

Code availability All code used is publicly available at https://github.com/PurdueMINDS/reducing-overconfidence.

Declaration

Conflict of interest Steve Hanneke is a member of the editorial board of the Machine Learning journal and is affiliated to the same institution as two authors of this manuscript.

Consent to participate Not applicable.

Consent for publication Not applicable.

Ethics approval Not applicable.

References

van Amersfoort, J., Smith, L., Teh, Y.W. & Gal, Y. (2020). Uncertainty estimation using a single deep deterministic neural network. In H.D. III & A. Singh (Eds.), *International conference on machine learning* (vol. 119, pp. 9690–9700). http://proceedings.mlr.press/v119/van-amersfoort20a.html.

Athalye, A., Carlini, N. & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In J. Dy & A. Krause (Eds.), *International conference on machine learning* (vol. 80, pp. 274–283). https://proceedings.mlr.press/v80/athalye18a.html.

Baccelli, F., & Błaszczyszyn, B. (2009). *Stochastic geometry and wireless networks* (vol. 1). Now Publishers Inc.

Blot, M., Picard, D., Thome, N., & Cord, M. (2019). Distributed optimization for deep learning with gossip exchange. *Neurocomputing, 330*, 287–296.

Bojchevski, A. & Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *International conference on machine learning* (vol. 97, pp. 695–704). http://proceedings.mlr.press/v97/bojchevski19a.html.

Boyd, S., Ghosh, A., Prabhakar, B., & Shah, D. (2005). Gossip algorithms: Design, analysis and applications. Computer and Communications Societies. In *Annual joint conference of the IEEE computer and communications societies*, vol. 3, 1653–1664.

Carlini, N. & Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Workshop on artificial intelligence and security ACM workshop on artificial intelligence and security* (pp. 3–14). Dallas, Texas, USAACM. https://doi.org/10.1145/3128572.3140444.

Carlini, N. & Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy* (pp. 39–57). https://doi.org/10.1109/SP.2017.49.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y. & Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In D. Precup & Y.W. Teh (Eds.), *International conference on machine learning* (vol. 70, pp. 854–863). https://proceedings.mlr.press/v70/cisse17a.html.

Cohen, J., Rosenfeld, E. & Kolter, Z. (2019). Certified adversarial robustness via randomized smoothing. In K. Chaudhuri & R. Salakhutdinov (Eds.), *International conference on machine learning* (vol. 97, pp. 1310–1320). https://proceedings.mlr.press/v97/cohen19c.html.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J. & Song, L. (2018). Adversarial attack on graph structured data. In J. Dy & A. Krause (Eds.), *International conference on machine learning* (vol. 80, pp. 1115–1124). https://proceedings.mlr.press/v80/dai18b.html.

Dai, R., Lefort, M., Armetta, F., Guillermin, M. & Duffner, S. (2021). Self-supervised continual learning for object recognition in image sequences. In *Advances on neural information processing* (pp. 239–247).

Daley, D. J., & Vere-Jones, D. (2007). *An introduction to the theory of point processes: Volume II: General theory and structure*. Springer.

DeGroot, M. H. & Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician), 321*, 212–22. http://www.jstor.org/stable/2987588.

Dhillon, G. S., Azizzadenesheli, K., Lipton, Z. C., Bernstein, J. D., Kossaifi, J., Khanna, A. & Anandkumar, A. (2018). Stochastic activation pruning for robust adversarial defense. In *International conference on learning representations*. https://openreview.net/forum?id=H1uR4GZRZ.

Dubey, A., Maatenvan der Maaten, L., Yalniz, Z., Li, Y. & Mahajan, D. (2019). Defense against adversarial images using web-scale nearest-neighbor search. In *Conference on computer vision and pattern recognition. IEEE conference on computer vision and pattern recognition*. arXiv:1903.01612.

Elliott, A., Law, S. & Russell, C. (2021). Explaining classifiers using adversarial perturbations on the perceptual ball. In *Conference on computer vision and pattern recognition IEEE conference on computer vision and pattern recognition*, (pp. 10693–10702).

Felzenszwalb, P. F. & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision, 59*, 167–181.

Geng, C., Huang, S., & Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 4310*, 3614–3631.

Ghiasi, A., Shafahi, A. & Goldstein, T. (2020). Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificate. In *International conference on learning representations*. https://openreview.net/forum?id=HJxdTxHYvB.

Goodfellow, I. J., Shlens, J. & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International conference on learning representations*. arXiv:1412.6572.

Gopalakrishnan, S., Marzi, Z., Madhow, U. & Pedarsani, R. (2018). Combating adversarial attacks using sparse representations. In *International conference on learning representations - workshop track*. https://openreview.net/forum?id=S10qYwywf.

Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321–1330). http://proceedings.mlr.press/v70/guo17a.html.

Guo, C., Rana, M., Cisse, M. & Maatenvan der Maaten, L. (2018). Countering adversarial images using input transformations. In *International conference on learning representations*. https://openreview.net/forum?id=SyJ7ClWCb.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hendrycks, D., Lee, K. & Mazeika, M. (2019). Using pre-training can improve model robustness and uncertainty. In *International conference on machine learning* (p 2712-2721). http://proceedings.mlr.press/v97/hendrycks19a.html. arXiv:1901.09960

Hess, S., Duivesteijn, W. & Mocanu, D. (2020). Softmax-based classification is k-means clustering: Formal proof, consequences for adversarial attacks, and improvement through centroid based tailoring. arXiv:2001.01987 [cs.LG].

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association, 47*(260), 663–685.

Hosang, J., Benenson, R. & Schiele, B. (2017). Learning non-maximum suppression. In *Conference on computer vision and pattern recognition IEEE conference on computer vision and pattern recognition* (pp. 4507–4515).

Hosseini, H., Kannan, S. & Poovendran, R. (2019). Are odds really odd? bypassing statistical detection of adversarial examples. arXiv:1907.12138 [cs.LG].

Hu, S., Yu, T., Guo, C., Chao, W.-L. & Weinberger, K.Q. (2019). A new defense against adversarial images: Turning a weakness into a strength. In *Advances in neural information processing systems* (pp. 1633–1644).

Karmon, D., Zoran, D. & Goldberg, Y. (2018). LaVAN: Localized and visible adversarial noise LaVAN: Localized and visible adversarial noise. In J. Dy & A. Krause (Eds.), *International conference on machine learning* (vol. 80, pp. 2507–2515). https://proceedings.mlr.press/v80/karmon18a.html.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L. & Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. In *International conference on learning representations*. https://openreview.net/forum?id=HklBjCEKvH.

Krizhevsky, A. & Hinton, G. (2009). Learning multiple layers of features from tiny images (Tech. Rep.). Toronto, Canada: Department of Computer Science, University of Toronto.

Kurakin, A., Goodfellow, I. J. & Bengio, S. (2017). Adversarial machine learning at scale. In *International conference on learning representations*. https://openreview.net/forum?id=BJm4T4Kgx.

Lassance, C., Gripon, V., & Ortega, A. (2021). Laplacian networks: Bounding indicator function smoothness for neural networks robustness. *APSIPA Transactions on Signal and Information Processing, 10*, e2.

Liao, P., Zhao, H., Xu, K., Jaakkola, T., Gordon, G., Jegelka, S. & Salakhutdinov, R. (2020). Graph adversarial networks: Protecting information against adversarial attacks.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax Weiss, T. & Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan & H. Lin (Eds.), *Advances in neural information processing systems* (vol. 33, pp. 7498–7512). Curran Associates, Inc.

Mądry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International conference on learning representations*. https://openreview.net/forum?id=rJzIBfZAb.

Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H. S. & Gal, Y. (2021). Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty.

Naeini, M. P., Cooper, G. F. & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. In *Conference on artificial intelligence AAAI conference on artificial intelligence*.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., & Edwards, B. (2018).. Adversarial robustness toolbox v1.2.0. arXiv:1807.01069 [cs.LG].

Niculescu-Mizil, A. & Caruana, R. (2005). Predicting good probabilities with supervised learning. In *International conference on machine learning* (pp. 625–632). New York, NY, USAACM. https://doi.org/10.1145/1102351.1102430.

Palm, C. (1943). *Intensitatsschwankungen im fernsprechverker*. Ericsson Technics.

Papernot, N. & McDaniel, P. (2018). Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. arXiv:1803.04765 [cs, stat].

Papernot, N., McDaniel, P., Goodfellow, I. , Jha, S., Celik, Z.B. & Swami, A. (2017). Practical black-box attacks against machine learning. In *Asia conference on computer and communications security proceedings of the 2017 ACM on Asia conference on computer and communications security* (pp. 506–519). New York, NY, USAACM. https://doi.org/10.1145/3052973.3053009.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G. & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Penrose, M. (2003). *Random geometric graphs*. Oxford: Oxford University Press.

Pinto, F., Yang, H., Lim, S.-N., Torr, P. & Dokania, P. K. (2021). Mix-maxent: Improving accuracy and uncertainty estimates of deterministic neural networks. In *Neurips 2021 workshop on distribution shifts: Connecting methods and applications*. https://openreview.net/forum?id=hlVgM8XcssV.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers, 10*(3), 61–74.

Roth, K., Kilcher, Y. & Hofmann, T. (2019). The odds are odd: A statistical test for detecting adversarial examples. In K. Chaudhuri & R. Salakhutdinov (Eds.), *International conference on machine learning* (vol. 97, pp. 5498–5507). https://proceedings.mlr.press/v97/roth19a.html.

Serban, A., Poll, E. & Visser, J. (2021). Deep repulsive prototypes for adversarial robustness. arXiv:2105.12427 [cs.LG].

Shekkizhar, S. & Ortega, A. (2020). Graph construction from data by non-negative kernel regression. In *International conference on acoustics, speech and signal processing IEEE international conference on acoustics, speech and signal processing* (pp. 3892–3896). https://doi.org/10.1109/ICASSP40776.2020.9054425

Sinkhorn, R., & Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics, 21*(2), 343–348.

Sitawarin, C. & Wagner, D. (2019a). Defending against adversarial examples with k-nearest neighbor. arXiv:1906.09525 [cs, stat].

Sitawarin, C. & Wagner, D. (2019b). On the robustness of deep k-nearest neighbors. In *Security and privacy workshops IEEE security and privacy workshops* (p. 1-7). https://doi.org/10.1109/SPW.2019.00014.

Sitawarin, C. & Wagner, D. (2020). Minimum-norm adversarial examples on KNN and KNN based models. *IEEE Security and Privacy Workshops*, pp. 34–40.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R. (2013). Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV].

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T. & Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in neural information processing systems* (pp. 13888–13899).

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A. & Mądry, A. (2019). Robustness may be at odds with accuracy. In *International conference on learning representations*. https://arxiv.org/pdf/1805.12152.pdf

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B. & Gu, Q. (2019). On the convergence and robustness of adversarial training. In K. Chaudhuri & R. Salakhutdinov (Eds.), *International conference on machine learning* (vol. 97, pp. 6586–6595). https://proceedings.mlr.press/v97/wang19i.html.

Wang, Z. & Loog, M. (2022). Enhancing classifier conservativeness and robustness by polynomiality. In *Conference on computer vision and pattern recognition IEEE conference on computer vision and pattern recognition* (pp. 13327–13336).

Xie, C., Wang, J., Zhang, Z., Ren, Z. & Yuille, A. (2018). Mitigating adversarial effects through randomization. In *International conference on learning representations*.

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L.E. & Jordan, M. (2019). Theoretically principled trade-off between robustness and accuracy. In K. Chaudhuri & R. Salakhutdinov (Eds.), *International conference on machine learning* (vol. 97, pp. 7472–7482). https://proceedings.mlr.press/v97/zhang19p.html.

Baccelli, F., & Błaszczyszyn, B. (2010). Stochastic geometry and wireless networks: Volume II, applications. *Foundations and Trends® in Networking, 4*(1–2), 1–312.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems, 30*(1–7), 107–117.

Chiu, S. N., Stoyan, D., Kendall, W. S., & Mecke, J. (2013). *Stochastic geometry and its applications*. London: Wiley.

Gleich, D. F. (2015). Pagerank beyond the web. *SIAM Review, 57*(3), 321–363. https://doi.org/10.1137/140976649

Kingman, J. (1993). *Poisson processes*. UK: Oxford University Press.

Ren, J., Liu, P.J., Fertig, E., Snoek, J., Poplin, R., Depristo, M. & Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. In *Advances in neural information processing systems* (vol. 32, pp. 14707–14718). Curran Associates, Inc.

Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 99–114.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X. & Zhang, Z. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv:1909.01315 [cs.LG].

Wang, X., Long, M., Wang, J., & Jordan, M. I. (2020). *Transferable calibration with lower bias and variance in domain adaptation advances on neural information processing systems*. Red Hook, NY: Curran Associates Inc.