Euro. Jnl of Applied Mathematics (2021), *vol.* 32, *pp.* 540–569 © The Author(s), 2020. Published by Cambridge University Press.
doi:10.1017/S0956792520000406

Graph interpolating activation improves both natural and robust accuracies in data-efficient deep learning

BAO WANG[1] and STAN J. OSHER[2]

[1]*Department of Mathematics, Scientific Computing and Imaging (SCI) Institute, University of Utah, Salt Lake City, UT, USA*
email: wangbaonj@gmail.com
[2]*Department of Mathematics, UCLA, Los Angeles, CA 90095-1555, USA*
email: sjo@math.ucla.edu

(*Received 24 September 2019*; *revised 1 August 2020*; *accepted 23 November 2020*; *first published online 28 December 2020*)

Improving the accuracy and robustness of deep neural nets (DNNs) and adapting them to small training data are primary tasks in deep learning (DL) research. In this paper, we replace the output activation function of DNNs, typically the data-agnostic softmax function, with a graph Laplacian-based high-dimensional interpolating function which, in the continuum limit, converges to the solution of a Laplace–Beltrami equation on a high-dimensional manifold. Furthermore, we propose end-to-end training and testing algorithms for this new architecture. The proposed DNN with graph interpolating activation integrates the advantages of both deep learning and manifold learning. Compared to the conventional DNNs with the softmax function as output activation, the new framework demonstrates the following major advantages: First, it is better applicable to data-efficient learning in which we train high capacity DNNs without using a large number of training data. Second, it remarkably improves both natural accuracy on the clean images and robust accuracy on the adversarial images crafted by both white-box and black-box adversarial attacks. Third, it is a natural choice for semi-supervised learning. This paper is a significant extension of our earlier work published in NeurIPS, 2018. For reproducibility, the code is available at https://github.com/BaoWangMath/DNN-DataDependentActivation.

Key words: Data-dependent activation, adversarial defense, data-efficient learning

2020 Mathematics Subject Classification: 68T01 (Primary); 68T10 (Secondary)

1 Introduction

Deep learning (DL) has achieved tremendous success in both image and speech recognition and natural language processing, and it has been widely used in industrial production [29]. Improving generalisation accuracy and adversarial robustness of deep neural nets (DNNs) are primary tasks in DL research. Moreover, applying DNNs to data-efficient machine learning (ML), where we do not have a large number of training instances, is important to different research communities.

Despite the extraordinary success of DNNs in image and speech perception, their vulnerability to adversarial attacks raises concerns when applying them to security-critical tasks, e.g., autonomous cars, robotics and DNN-based malware detection systems [2]. Since the seminal

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 1. *Test errors of DNNs trained on the entire (50K), the first 10K and the first 1K instances of the training set of the CIFAR10*

Network	# of parameters (M)	50K (%)	10K (%)	1K (%)
ResNet20	0.27	9.06 (8.75 [19])	12.83	34.90
ResNet32	0.46	7.99 (7.51 [19])	11.18	33.41
ResNet44	0.66	7.31 (7.17 [19])	10.66	34.58
ResNet56	0.85	7.24 (6.97 [19])	9.83	37.83
ResNet110	1.7	6.41 (6.43 [19])	8.91	42.94

work of Szegedy et al. [47], recent research shows that DNNs are vulnerable to many kinds of adversarial attacks including physical, poisoning and inference (evasion) attacks [8, 6, 39, 14]. Physical attacks occur during data acquisition, poisoning and inference attacks happen during training and testing phases of ML, respectively.

Adversarial attacks have been successful in both white-box and black-box scenarios. In white-box attacks, the adversarial have access to the architecture and weights of DNNs. In black-box attacks, the adversarial have no access to the details of the underlying model. Black-box attacks are successful because one can perturb an image to cause its misclassification on one DNN, and the same perturbed image also has a significant chance to be misclassified by another DNN; this is known as the transferability of adversarial examples [41]. Due to this transferability, it is straightforward to attack DNNs in a black-box fashion by attacking an oracle model [32, 5]. There also exist universal perturbations that can imperceptibly perturb any image and cause misclassification for any given network [34]. Dou et al. [11] analysed the efficiency of many adversarial attacks for a large variety of DNNs.

Besides the issue of adversarial vulnerability, the superior accuracy of DNNs depends heavily on a massive amount of training data. When we do not have sufficient training data, which is often the case in many real situations, to train a high capacity deep network, performance degradation becomes a serious problem. As shown in Table 1, when ResNets are trained on 50K or 10K CIFAR10 images, as the depth of ResNet increases, the test accuracy gains. However, when ResNets are trained on only 1K images, the test accuracy decays as the model's capacity increases. For instance, the test errors of ResNet20 and ResNet110 are 34.90% and 42.94%, respectively.

1.1 Our contributions

In this paper, we propose an end-to-end framework to mitigate the aforementioned two issues of DNNs, i.e., adversarial vulnerability and generalisation accuracy degradation in the small training data scenario. At the core of our framework is to replace the data-agnostic softmax output activation with a data-dependent graph interpolating function. To this end, we leverage the weighted nonlocal Laplacian (WNLL) [45] to interpolate features in the hidden state of DNNs. In back-propagation, we linearise the WNLL activation function to compute gradient of the loss function approximately. The major advantages of the proposed framework are summarised below.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

- The naturally trained DNNs with the WNLL output activation obtained by solving the empirical risk minimisation (ERM), i.e., (1.2), are remarkably more accurate than the vanilla DNNs with the softmax output activation.
- The robustly trained DNNs with the WNLL activation obtained by solving the empirical adversarial risk minimisation (EARM), i.e., (1.1), are much more robust to adversarial attacks than the robustly trained vanilla DNNs. To the best of our knowledge, DNNs with the WNLL activation achieve the current-state-of-the-art result in adversarial defense on the CIFAR10 and MNIST benchmarks.
- In the small training data situation, the WNLL activation can regularise the training procedure. The test accuracy of DNNs with the WNLL activation increases as the network goes deeper.
- DNN with the WNLL output activation is a natural choice for semi-supervised DL.
- The proposed framework is applicable to any off-the-shelf DNNs when use the softmax as its output activation.

1.2 Related work

In this subsection, we will discuss related work from the viewpoints of improving generalisability and adversarially robustness.

1.2.1 *Improving generalisability of DNNs*

Generalisability is crucial to DL, and many efforts have been made to improve the test accuracy of DNNs [4, 20]. Advances in network architectures such as VGG networks [46], deep residual networks (ResNets) [19, 18] and recently DenseNets [22] and many others [9], together with powerful hardware, make the training of very deep networks with good generalisation capabilities possible. Effective regularisation techniques such as dropout and maxout [21, 51, 13], as well as data augmentation methods [26, 46] have also explicitly improved generalisation for DNNs. From the optimisation point of view, Laplacian smoothing stochastic gradient descent has been recently proposed to improve training and generalisation of DNNs [38].

A key component of DNN is the activation function. Improvements in designing of activation functions, such as the rectified linear unit (ReLU) [12], have led to huge improvements in performance in computer vision tasks [36, 26]. More recently, activation functions adaptively trained to the data such as the adaptive piece-wise linear unit (APLU) [1] and parametric rectified linear unit (PReLU) [16] have led to further improvements in the performance of DNNs. For output activation, support vector machine (SVM) has also been successfully applied in place of softmax [48]. Though training DNNs with softmax or SVM as output activation is effective in many tasks, it is possible that alternative activations that consider the manifold structure of data by interpolating the output based on both training and testing data can boost the performance of the deep network. In particular, ResNets can be modelled as solving control problems of a class of transport equations in the continuum limit [31, 54]. Transport equation theory suggests that using an interpolating function that interpolates terminal values from initial values can dramatically simplify the control problem compared with an ad hoc choice. This further suggests that a fixed and data-agnostic activation for the output layer may be suboptimal.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

1.2.2 *Adversarial defense*

EARM is one of the most successful mathematical frameworks for certified adversarial defense. Under the EARM framework, adversarial defense for the ℓ_∞-norm-based inference attacks can be formulated as solving the following minimax optimisation problem

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \max_{\|\mathbf{x}_i'-\mathbf{x}_i\|_\infty\leqslant\epsilon} L(f(\mathbf{x}_i', \mathbf{w}), y_i), \tag{1.1}$$

where $f(\cdot, \mathbf{w})$ is a function in the hypothesis class $\mathcal{H}$, e.g., DNNs, parameterised by $\mathbf{w}$. Here, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ are n i.i.d. data-label pairs drawn from some high-dimensional unknown distribution $\mathcal{D}$, $L(f(\mathbf{x}_i, \mathbf{w}), y_i)$ is the loss associated with f on the data-label pair $(\mathbf{x}_i, y_i)$. For classification, L is typically selected to be the cross-entropy loss; for regression, the root mean square error is commonly used. The adversarial defense for other measure based attacks can be formulated similarly. As a comparison, solving ERM is used to train models in a natural fashion to classify the clean data, where ERM is to solve the following optimisation problem

$$\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} L(f(\mathbf{x}_i, \mathbf{w}), y_i). \tag{1.2}$$

Many of the existing approaches try to defend against the inference attacks by searching for a good surrogate loss to approximate the loss function in the EARM. Projected gradient descent (PGD) adversarial training is a representative work along this line that approximates EARM by replacing $\mathbf{x}_i'$ with the adversarial data that is obtained by applying the PGD attack to the clean data [14, 33]. Besides finding an appropriate surrogate to approximate the empirical adversarial risk, under the EARM framework, we can also improve the hypothesis class to improve adversarial robustness of the trained robust models [54].

There is a massive volume of research over the past several years on defending against adversarial attacks for DNNs. Randomised smoothing transforms an arbitrary classifier f into a 'smoothed' surrogate classifier g and is certifiably robust to the ℓ_2-norm based adversarial attacks [30, 10]. Among the randomised smoothing technique, one of the most popular ideas is to inject Gaussian noise to the input image, and the classification result is based on the probability of noisy image in decision region. Wang et al. [54] modeled ResNets as a transport equation and interpreted the adversarial vulnerability of DNNs as irregularity of the transport equation's solution. To enhance its regularity, i.e., improve adversarial robustness, they added a diffusion term to the transport equation and solved the resulting convection-diffusion equation by the celebrated Feynman-Kac formula. The resulting algorithm remarkably improves both natural and robust accuracies of the robustly trained DNNs.

Robust optimisation for solving EARM has achieved tremendous success in certified adversarial defense [33, 56]. Regularisation in EARM can further boost the robustness of the adversarially trained robust models [27, 43, 57]. The adversarial defense algorithms should learn a classifier with high test accuracy on both clean and adversarial data. To achieve this goal, Zhang et al. [56] developed a new loss function named TRADES that explicitly trades off between natural and robust generalisation.

Besides robust optimisation, there are many other approaches for adversarial defense. Defensive distillation was proposed to increase the stability of DNN [40], and a related approach [49] cleverly modifies the training data to increase robustness against black-box attacks and

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

adversarial attacks in general. To counter adversarial perturbations, Guo et al. [15] proposed to use image transformations, e.g., bit-depth reduction, JPEG compression, total variation minimisation and image quilting. These input transformations are intended to be nondifferentiable, thus making adversarial attacks more difficult, especially for gradient-based attacks. GANs are also used for adversarial defense [44]. However, adversarial attacks can break these gradient mask-based defenses by circumventing the obfuscated gradient [3].

Instead of using the softmax function as DNN's output activation, Wang et al. [53] utilised a non-parametric graph interpolating function which provably converges to the solution of a Laplace–Beltrami equation on a high-dimensional manifold [45]. The proposed data-dependent activation shows a remarkable amount of generalisation accuracy improvement, and the results are more stable when one only has a limited amount of training data. This data-dependent activation is also useful in adversarial defense when combined with image transformations [52]. Verma et al. [50] simplified the interpolation procedure and generalised it to more hidden layers to learn better representations.

1.3 Organisation

This paper is structured in the following way: In Section 2, we present the generic architecture of DNNs with a graph interpolating function as its output activation. In Section 3, we present training and testing algorithms in both natural and robust fashions for the proposed DNNs with graph interpolating activation. We verify the performance of the proposed algorithm numerically in Section 4 from the lens of natural and robust generalisation accuracies and semi-supervised learning. In Section 5, we provide geometric explanations for improving generalisation and robustness by using the proposed new framework. This paper concludes with a remark in Section 6.

2 Network architecture

We illustrate the training and testing procedures of a standard DNN in Figure 1, where

- **Training** (Figure 1(a)), in the kth iteration, given a mini-batch of training data $(\mathbf{X}, \mathbf{Y})$, we perform:
  - *Forward propagation:* Transform $\mathbf{X}$ into features by the DNN block (a combination of convolutional layers, nonlinearities, etc.), and then feed these features into the softmax activation to obtain the predictions $\tilde{\mathbf{Y}}$, i.e.,

$$\tilde{\mathbf{Y}} = \text{Softmax}(\text{DNN}(\mathbf{X}, \Theta^{k-1}), \mathbf{W}^{k-1}),$$

    where $(\Theta^{k-1}, \mathbf{W}^{k-1})$ are the temporary values of the trainable weights $(\Theta, \mathbf{W})$ at the $(k-1)$th iteration. Then the loss is computed (e.g., cross entropy) between the ground-truth labels $\mathbf{Y}$ and the predicted labels $\tilde{\mathbf{Y}}$: $\mathcal{L} \doteq \mathcal{L}^{\text{Linear}} = \text{Loss}(\mathbf{Y}, \tilde{\mathbf{Y}})$.
  - *Backpropagation:* Update weights $(\Theta^{k-1}, \mathbf{W}^{k-1})$ by applying gradient descent with learning rate γ

$$\mathbf{W}^k = \mathbf{W}^{k-1} - \gamma \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \mathbf{W}},$$

$$\Theta^k = \Theta^{k-1} - \gamma \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \tilde{\mathbf{X}}} \cdot \frac{\partial \tilde{\mathbf{X}}}{\partial \Theta}.$$

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

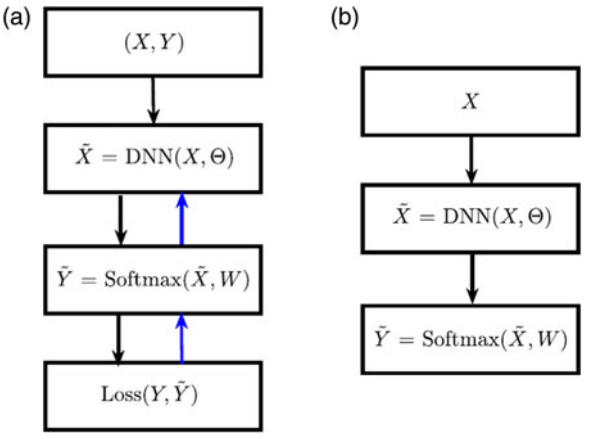


FIGURE 1. Illustration of training and testing procedures of the standard DNN with the softmax function as output activation layer. (a): Training; (b): testing.

- **Testing** (Figure 1(b)): Once the training procedure finishes with the learned parameters $(\Theta, \mathbf{W})$. The predicted labels for the testing data $\mathbf{X}$ are

$$\tilde{\mathbf{Y}} = \mathrm{Softmax}(\mathrm{DNN}(\mathbf{X}, \Theta), \mathbf{W}),$$

  for notational simplicity, we still denote the test set and the learned weights as $\mathbf{X}$, Θ and $\mathbf{W}$, respectively.

Even though this DL paradigm achieves the current-state-of-the-art success in many artificial intelligence tasks, the data-agnostic activation (softmax) acts as a linear model on the space of deep features $\tilde{\mathbf{X}}$, which does not take into consideration the underlying manifold structure of $\tilde{\mathbf{X}}$, and has many other problems, e.g., it is less applicable when we have a small amount of training data and is not robust to adversarial attacks. To this end, we replace the softmax output activation with a graph interpolating function, WNLL, which will be introduced in the following subsection. We illustrate the training and testing data flow in Figure 2 which will be discussed later.

2.1 Graph-based high-dimensional interpolating function – A harmonic extension approach

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of points located on a high-dimensional manifold $\mathcal{M} \subset \mathbb{R}^d$ and $\mathbf{X}^{\mathrm{te}} = \{\mathbf{x}_1^{\mathrm{te}}, \mathbf{x}_2^{\mathrm{te}}, \ldots, \mathbf{x}_m^{\mathrm{te}}\}$ ('te' for template) be a subset of $\mathbf{X}$ that is labelled with the label function $g(\mathbf{x})$. We want to interpolate a function u that is defined on the whole manifold $\mathcal{M}$ and can be used to interpolate labels for the entire data set $\mathbf{X}$. The harmonic extension is a natural approach to find such a smooth interpolating function which is defined by minimising the following Dirichlet energy functional

$$\mathcal{E}(u) = \frac{1}{2} \sum_{\mathbf{x},\mathbf{y} \in \mathbf{X}} w(\mathbf{x}, \mathbf{y}) \, (u(\mathbf{x}) - u(\mathbf{y}))^2 \,, \tag{2.1}$$

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

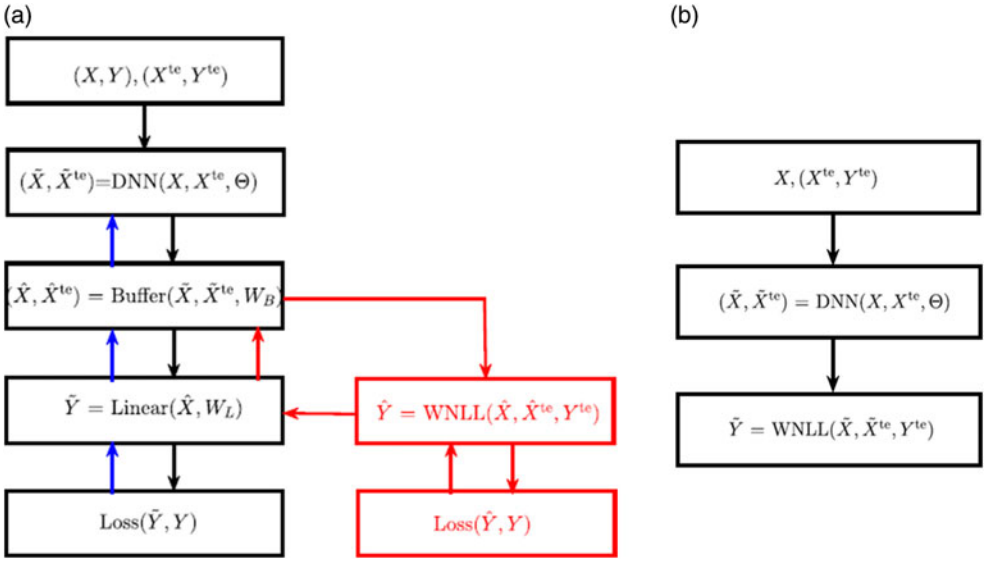


FIGURE 2. Illustration of training and testing procedures of the DNN with the WNLL interpolating function as the output activation function. (a): Training; (b): testing.

with the boundary condition

$$u(\mathbf{x}) = g(\mathbf{x}), \ \ \mathbf{x} \in \mathbf{X}^{te},$$

where $w(\mathbf{x}, \mathbf{y})$ is a weight function, chosen to be Gaussian: $w(\mathbf{x}, \mathbf{y}) = \exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{\sigma^2})$ with σ being a scaling parameter. By taking the variational derivative of the energy functional (2.1), we get the following Euler–Lagrange equation

$$\begin{cases} \sum_{\mathbf{y} \in \mathbf{X}} (w(\mathbf{x}, \mathbf{y}) + w(\mathbf{y}, \mathbf{x})) \, (u(\mathbf{x}) - u(\mathbf{y})) = 0 \mathbf{x} \in \mathbf{X}/\mathbf{X}^{te} \\ u(\mathbf{x}) = g(\mathbf{x}) \mathbf{x} \in \mathbf{X}^{te}. \end{cases} \tag{2.2}$$

By solving the linear system (2.2), we obtain labels $u(\mathbf{x})$ for the unlabelled data $\mathbf{x} \in \mathbf{X}/\mathbf{X}^{te}$. The interpolation quality becomes very poor when only a tiny amount of data are labelled, i.e., $|\mathbf{X}^{te}| \ll |\mathbf{X}/\mathbf{X}^{te}|$. To alleviate this degradation, the weight of the labelled data is increased in the above Euler–Lagrange equation (2.2), which gives

$$\begin{cases} \sum_{\mathbf{y} \in \mathbf{X}} (w(\mathbf{x}, \mathbf{y}) + w(\mathbf{y}, \mathbf{x})) \, (u(\mathbf{x}) - u(\mathbf{y})) + \\ \left(\frac{|\mathbf{X}|}{|\mathbf{X}^{te}|} - 1 \right) \sum_{\mathbf{y} \in \mathbf{X}^{te}} w(\mathbf{y}, \mathbf{x}) \, (u(\mathbf{x}) - u(\mathbf{y})) = 0 \ \ \mathbf{x} \in \mathbf{X}/\mathbf{X}^{te} \\ u(\mathbf{x}) = g(\mathbf{x}) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{x} \in \mathbf{X}^{te}. \end{cases} \tag{2.3}$$

We call the solution to (2.3) WNLL and denote it as WNLL$(\mathbf{X}, \mathbf{X}^{te}, \mathbf{Y}^{te})$. Shi et al. [45] showed that the WNLL graph interpolating function converges to the solution of the associated high-dimensional Laplace–Beltrami equation. For classification, $g(\mathbf{x})$ is the one-hot label for $\mathbf{x}$.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Remark 1 *For a given* $\mathbf{x}$*, due to the exponential decay of the kernel* $w(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/\sigma^2)$*, we do not need to compute weights for all* $\mathbf{y}$ *in* $\mathbf{X}$*. In practice, we only consider the contribution from the first m-nearest neighbours of* $\mathbf{x}$ *and let* σ *be the distance between* $\mathbf{x}$ *and its nth nearest neighbour. We use the approximate nearest neighbour* [35] *to search all the m nearest neighbours of any given data* $\mathbf{x}$*.*

2.1.1 *Theoretical guarantees for the WNLL interpolating function*

To ensure the accuracy of WNLL interpolation, the template data, i.e., the labelled data, should cover all classes of data in $\mathbf{X}$. We give a necessary condition in Theorem 2.1.

Theorem 2.1 ([53]) *Suppose we have a data set,* $\mathbf{X}$*, which consists of N different classes of data with each instance having the same probability to belong to any of the N classes. Moreover, suppose the number of instances of each class is sufficiently large. If we want to guarantee all classes of data to be sampled at least once, on average at least* $N\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}\right)$ *data needs to be sampled from* $\mathbf{X}$*. In this case, the number of data being sampled, in expectation for each class, is* $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N} \approx \ln N$*.*

We consider the convergence of the WNLL for graph interpolation and give a theoretical interpretation of the special weight selected in (2.3). We summarise some results from [45]. Consider the following generalised WNLL interpolation

$$\begin{cases} \sum_{\mathbf{y}\in\mathbf{X}} R_\delta(\mathbf{x}, \mathbf{y})\,(u_\delta(\mathbf{x}) - u_\delta(\mathbf{y})) + \mu \sum_{\mathbf{y}\in\mathbf{X}^{\text{te}}} K_\delta(\mathbf{x}, \mathbf{y})(u_\delta(\mathbf{x}) - g(\mathbf{y})) = 0, \quad \mathbf{x} \in \mathbf{X}, \\ u_\delta(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \mathbf{X}^{\text{te}}. \end{cases} \tag{2.4}$$

where $R_\delta(\mathbf{x}, \mathbf{y})$, $K_\delta(\mathbf{x}, \mathbf{y})$ are kernel functions given as

$$R_\delta(\mathbf{x}, \mathbf{y}) = C_\delta R\left(\frac{|\mathbf{x} - \mathbf{y}|^2}{4\delta^2}\right), \quad K_\delta(\mathbf{x}, \mathbf{y}) = C_\delta K\left(\frac{|\mathbf{x} - \mathbf{y}|^2}{4\delta^2}\right), \tag{2.5}$$

where $C_\delta = \frac{1}{(4\pi\delta^2)^{k/2}}$ is the normalisation factor. $R, K \in C^2(\mathbb{R}^+)$ are two kernel functions satisfying the conditions listed in Assumption 1.

Assumption 1

- *Assumptions on the manifold:* $\mathcal{M}$ *is a k-dimensional closed* C^∞ *manifold isometrically embedded in a Euclidean space* $\mathbb{R}^d$*.* $\mathcal{D}$ *and* $\partial\mathcal{D}$ *are smooth submanifolds of* $\mathbb{R}^d$*. Moreover,* $g(\mathbf{x}) \in C^1(\mathcal{D})$*.*
- *Assumptions on the kernel functions:*
  - *(a) Smoothness:* $K(r), R(r) \in C^2(\mathbb{R}^+)$*;*
  - *(b) Nonnegativity:* $R(r), K(r) \geqslant 0$ *for any* $r \geqslant 0$*.*
  - *(c) Compact support:* $R(r) = 0$ *for* $\forall r > 1$*;* $K(r) = 0$ *for* $\forall r > r_0 \geqslant 2$*.*
  - *(d) Nondegeneracy:* $\exists \delta_0 > 0$ *such that* $R(r) \geqslant \delta_0$ *for* $0 \leqslant r \leqslant 1/2$ *and* $K(r) \geqslant \delta_0$ *for* $0 \leqslant r \leqslant 2$*.*
- *Assumptions on the point cloud:* $\mathbf{X}^{\text{te}}$ *and* $\mathbf{X}$ *are uniformly distributed on* $\mathcal{M}$ *and* $\mathcal{D}$*, respectively.*

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

As the continuous counterpart, we consider the Laplace–Beltrami equation on a closed smooth manifold $\mathcal{M}$

$$\begin{cases} \Delta_{\mathcal{M}} u(\mathbf{x}) = 0, & \mathbf{x} \in \mathcal{M}, \\ u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \mathcal{D}, \end{cases} \tag{2.6}$$

where $\Delta_{\mathcal{M}} = \operatorname{div}(\nabla)$ is the Laplace–Beltrami operator on $\mathcal{M}$. Let $\Phi : \Omega \subset \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d$ be a local parametrization of $\mathcal{M}$ and $\theta \in \Omega$. For any differentiable function $f : \mathcal{M} \to \mathbb{R}$, we define the gradient on the manifold

$$\nabla f(\Phi(\theta)) = \sum_{i,j=1}^{m} g^{ij}(\theta) \frac{\partial \Phi}{\partial \theta_i}(\theta) \frac{\partial f(\Phi(\theta))}{\partial \theta_j}(\theta). \tag{2.7}$$

And for the vector field $F : \mathcal{M} \to T_{\mathbf{x}}\mathcal{M}$ on $\mathcal{M}$, where $T_{\mathbf{x}}\mathcal{M}$ is the tangent space of $\mathcal{M}$ at $\mathbf{x} \in \mathcal{M}$, the divergence is defined as

$$\operatorname{div}(F) = \frac{1}{\sqrt{\det G}} \sum_{k=1}^{d} \sum_{i,j=1}^{m} \frac{\partial}{\partial \theta_i} \left(\sqrt{\det G}\, g^{ij} F^k(\Phi(\theta)) \frac{\partial \Phi^k}{\partial \theta_j} \right) \tag{2.8}$$

where $(g^{ij})_{i,j=1,\dots,k} = G^{-1}$, $\det G$ is the determinant of matrix G and $G(\theta) = (g_{ij})_{i,j=1,\dots,k}$ is the first fundamental form with

$$g_{ij}(\theta) = \sum_{k=1}^{d} \frac{\partial \Phi_k}{\partial \theta_i}(\theta) \frac{\partial \Phi_k}{\partial \theta_j}(\theta), \quad i,j = 1, \dots, m. \tag{2.9}$$

and $(F^1(\mathbf{x}), \dots, F^d(\mathbf{x}))^T$ is the representation of F in the embedding coordinates.

We have the following high probability guarantee for convergence of the WNLL interpolating function to the solution of the Laplace–Beltrami equation on the manifold $\mathcal{M}$.

Theorem 2.2 ([45]) *Let u_δ solve (2.4) and u solve (2.6). Given Assumption 1, with probability at least $1 - 1/(2n)$, where $n = |\mathbf{X}|$, we have*

$$|u_\delta - u| \leqslant C\delta,$$

as long as

$$\mu \sum_{\mathbf{y} \in \mathbf{X}^{\text{te}}} K_\delta(\mathbf{x}, \mathbf{y}) \geqslant C \sum_{\mathbf{y} \in \mathbf{X}} R_\delta(\mathbf{x}, \mathbf{y}), \quad \mathbf{x} \in \mathbf{X} \cap \mathcal{D}_\delta, \tag{2.10}$$

where $\mathcal{D}_\delta = \{\mathbf{x} \in \mathcal{M} : dist(\mathbf{x}, \mathcal{D}) \leqslant 2\delta\}$, and $C = C(\mathcal{M}, \mathcal{D}, R, K) > 0$ is a constant that is independent of δ, $\mathbf{X}$ and $\mathbf{X}^{\text{te}}$.

In the above theorem, (2.10) actually gives a constraint on the weight μ. Note that

$$\frac{1}{n} \sum_{\mathbf{y} \in \mathbf{X}} R_\delta(\mathbf{x}, \mathbf{y}) \approx \frac{1}{|\mathcal{M}|} \int_{\mathcal{M}} R_\delta(\mathbf{x}, \mathbf{y}) d\mathbf{y} = O(1), \quad \mathbf{x} \in \mathbf{X} \cap \mathcal{D}_\delta.$$

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

$\mathbf{X}^{\text{te}}$ samples $\mathcal{D}$, if $\mathbf{X}^{\text{te}}$ is dense enough, we have

$$\frac{1}{|\mathbf{X}^{\text{te}}|}\sum_{\mathbf{y}\in\mathbf{X}^{\text{te}}} K_\delta(\mathbf{x},\mathbf{y}) \approx \frac{1}{|\mathcal{D}|}\int_{\mathcal{D}} K_\delta(\mathbf{x},\mathbf{y})d\mathbf{y}, \quad \mathbf{x}\in\mathbf{X}\cap\mathcal{D}_\delta.$$

Here, we need the assumption on K such that $K(r)\geqslant\delta_0>0, \forall\, 0\leqslant r\leqslant 2$. This implies that

$$\int_{\mathcal{D}} K_\delta(\mathbf{x},\mathbf{y})d\mathbf{y} = O(1), \quad \mathbf{x}\in\mathbf{X}\cap\mathcal{D}_\delta.$$

Hence, from (2.10), we have

$$\mu \sim \frac{|\mathbf{X}|}{|\mathbf{X}^{\text{te}}|},$$

which explains the scaling of $\frac{|\mathbf{X}|}{|\mathbf{X}^{\text{te}}|}$ in the WNLL.

2.2 DNNs with the graph interpolating function as output activation

A straightforward approach is to replace the softmax function with the WNLL in Figure 1. However, backpropagation is difficult in this case. To resolve this, we consider a new DNN architecture as shown in Figure 2 which will be discussed in detail in Section 3.

3 Algorithms

In this section, we will present training and inference algorithms for DNNs with the WNLL as the output activation in both natural and robust fashions. Natural training means to solve the ERM problem on the training data set and robust training stands for training an adversarially robust deep network by solving the EARM problem. Meanwhile, we will also adapt DNNs with the WNLL interpolating output activation to semi-supervised learning.

3.1 Natural training and inference

We abstract the natural training and testing procedures for DNNs with the WNLL activation in Figure 2(a) and (b), respectively. As a prerequisite of the WNLL interpolation, we need to reserve a small portion of data-label pairs, denoted as $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$, to interpolate labels for the unlabelled data in both training and testing procedures of DNNs with the WNLL activation. We call $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$ as the preserved template. Directly replacing the softmax by the WNLL in the architecture shown in Figure 1(a) causes difficulties in backpropagation, namely, the gradient $\frac{\partial\mathcal{L}}{\partial\Theta}$ is difficult to compute since WNLL defines a very complex implicit function. Instead, to train DNNs with the WNLL as the output activation, we propose a proxy via an auxiliary neural net (Figure 2(a)). On top of the original DNNs, we add a buffer block (a fully connected (FC) layer followed by a ReLU) and followed by two parallel branches, the WNLL and the linear (FC) layers. We train the auxiliary DNNs by alternating between the following two steps: training DNNs with linear and WNLL activation, respectively. In the following, we denote DNN with the WNLL activation as DNN-WNLL, e.g., we denote ResNet20 with WNLL activation as ResNet20-WNLL.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Train DNN-WNLL with linear activation: Run N_1 steps of the following forward and backward propagation, where in the kth iteration, we have:

- *Forward propagation:* Transform the training data $\mathbf{X}$, respectively, by DNN, Buffer and Linear blocks into the predicted labels $\tilde{\mathbf{Y}}$:

$$\tilde{\mathbf{Y}} = \text{Linear}(\text{Buffer}(\text{DNN}(\mathbf{X}, \Theta^{k-1}), \mathbf{W}_B^{k-1}), \mathbf{W}_L^{k-1}).$$

  Then compute the loss between the ground truth labels $\mathbf{Y}$ and the predicted ones $\tilde{\mathbf{Y}}$, denoted the loss as $\mathcal{L}^{\text{Linear}}$.
- *Backpropagation:* Update $(\Theta^{k-1}, \mathbf{W}_B^{k-1}, \mathbf{W}_L^{k-1})$ by stochastic gradient descent:

$$\mathbf{W}_L^k = \mathbf{W}_L^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \mathbf{W}_L},$$

$$\mathbf{W}_B^k = \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B},$$

$$\Theta^k = \Theta^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \tilde{\mathbf{X}}} \cdot \frac{\partial \tilde{\mathbf{X}}}{\partial \Theta}.$$

Train DNN-WNLL with the WNLL activation: Run N_2 steps of the following forward and backward propagation, where in the kth iteration, we have:

- *Forward propagation:* The training data $\mathbf{X}$, template $\mathbf{X}^{\text{te}}$ and $\mathbf{Y}^{\text{te}}$ are transformed, respectively, by DNN, Buffer and WNLL blocks to get predicted labels $\hat{\mathbf{Y}}$:

$$\hat{\mathbf{Y}} = \text{WNLL}(\text{Buffer}(\text{DNN}(\mathbf{X}, \Theta^{k-1}), \mathbf{W}_B^{k-1}), \hat{\mathbf{X}}^{\text{te}}, \mathbf{Y}^{\text{te}}).$$

  Then compute the loss, $\mathcal{L}^{\text{WNLL}}$, between the ground truth labels $\mathbf{Y}$ and predicted ones $\hat{\mathbf{Y}}$.
- *Backpropagation:* Update weights $\mathbf{W}_B^{k-1}$ only (here we take a similar strategy as weights fine-tuning, which is widely used in transfer learning), $\mathbf{W}_L^{k-1}$ and Θ^{k-1} will be tuned in the next iteration in training DNN-WNLL with the linear activation, by stochastic gradient descent.

$$\mathbf{W}_B^k = \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}} \cdot \frac{\partial \hat{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B} \approx \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B}. \tag{3.1}$$

We use the computational graph of the left branch (linear layer) to compute the approximated gradients for the DNN with WNLL activation. For a given loss value $\mathcal{L}^{\text{WNLL}}$, we adopt the approximation $\frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}} \cdot \frac{\partial \hat{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \approx \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}}$ where the right-hand side is also evaluated at the value of $\mathcal{L}^{\text{WNLL}}$. The heuristic behind this approximation is the following: WNLL defines a harmonic function implicitly, and the linear function is the simplest nontrivial explicit harmonic function. Empirically, we observe this simple approximation works well in training the deep network. The reason why we freeze the network in the DNN block is mainly because of the stability concerns.

The above alternating scheme is an algorithm of a greedy fashion. During training, the WNLL activation plays two roles: on the one hand, alternating between the linear and the WNLL activation benefits both which enables the neural nets to learn features that are appropriate for both linear classification and the WNLL-based manifold interpolation. On the other hand, in the case when we lack sufficient training data, the training of DNNs usually gets stuck at some bad local

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Algorithm 1 DNN with the WNLL Output Activation: Training Procedure.

Input: Training set: (data, label) pairs $(\mathbf{X}, \mathbf{Y})$. The number of alternating steps N and the number of epochs for training DNN with WNLL activation M.
Output: An optimised DNN with the WNLL activation, denoted as DNN-WNLL.
for iter $= 1, \ldots, N$ **do**
    //Train the left branch: DNN with the linear activation.
    Train DNN + Linear blocks and denote the learned model as DNN-Linear.
    //Train the right branch: DNN with the WNLL activation.
    Split $(\mathbf{X}, \mathbf{Y})$ into training data and template, i.e., $(\mathbf{X}, \mathbf{Y}) \doteq (\mathbf{X}^{\text{tr}}, \mathbf{Y}^{\text{tr}}) \bigcup (\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$.
    Partition the training data into M mini-batches, i.e., $(\mathbf{X}^{\text{tr}}, \mathbf{Y}^{\text{tr}}) = \bigcup_{i=1}^{M} (\mathbf{X}_i^{\text{tr}}, \mathbf{Y}_i^{\text{tr}})$.
    for $i = 1, 2, \ldots, M$ **do**
        Transform $\mathbf{X}_i^{\text{tr}} \bigcup \mathbf{X}^{\text{te}}$ by DNN-Linear, i.e., $\tilde{\mathbf{X}}^{\text{tr}} \bigcup \tilde{\mathbf{X}}^{\text{te}} = \text{DNN}_{\text{Linear}}(\mathbf{X}_i^{\text{tr}} \bigcup \mathbf{X}^{\text{te}})$.
        Apply WNLL (2.3) on $\{\tilde{\mathbf{X}}^{\text{tr}} \bigcup \tilde{\mathbf{X}}^{\text{te}}, \mathbf{Y}^{\text{te}}\}$ to interpolate label $\tilde{\mathbf{Y}}^{\text{tr}}$.
        Backpropagate the error between $\mathbf{Y}^{\text{tr}}$ and $\tilde{\mathbf{Y}}^{\text{tr}}$ via (3.1) to update $\mathbf{W}_B$ only.

Algorithm 2 DNN with the WNLL Output Activation: Testing Procedure.

Input: Testing data $\mathbf{X}$, template $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$. The optimised model DNN-WNLL.
Output: Predicted label $\tilde{\mathbf{Y}}$ for $\mathbf{X}$.
Apply the DNN block of the DNN-WNLL to $\mathbf{X} \bigcup \mathbf{X}^{\text{te}}$ to get the features $\tilde{\mathbf{X}} \bigcup \tilde{\mathbf{X}}^{\text{te}}$.
Apply the WNLL interpolation (2.3) on $\{\tilde{\mathbf{X}} \bigcup \tilde{\mathbf{X}}^{\text{te}}, \mathbf{Y}^{\text{te}}\}$ to interpolate label $\tilde{\mathbf{Y}}$.

minima which cannot generalise well on the new data. We use the WNLL interpolation to perturb those learned sub-optimal weights and to help to arrive at a local minimum with better generalisability. At inference (test) time, we remove the linear classifier from the neural nets and use the DNN block together with the WNLL to predict new data (Figure 2(b)). The reason for using the WNLL instead of the linear layer is simply because the WNLL interpolation is superior to the linear classifier and this superiority is preserved when applied to deep features (which will be confirmed in Section 4). Moreover, the WNLL interpolation utilises both DL features and the reserved template at the test time to guide the classifier and to enhance adversarial robustness in classification.

We summarise the training and testing for DNN-WNLL in Algorithms 1 and 2, respectively. In each round of the alternating procedure, i.e., each outer loop in Algorithm 1, the entire training data set $(\mathbf{X}, \mathbf{Y})$ is first used to train DNN-WNLL with the linear activation. We randomly separate a template, e.g., half of the entire data from the training set which will be used to perform WNLL interpolation in training DNN-WNLL with the WNLL activation. In practice, for both training and testing, we use mini-batches for both the template and the interpolated points when the entire data set is too large. The final predicted labels are based on a majority voting across interpolation results from all the template mini-batches.

Remark 2 *In Algorithm 1, the WNLL interpolation is also performed in a mini-batch manner (as shown in the inner iteration). Based on our experiments, this has a very small influence on reducing interpolation accuracy.*

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

3.2 Adversarial training

Adversarial training is one of the most generic frameworks for adversarial defense. The key idea of adversarial training is to augment the training data with adversarial versions which can be obtained by applying adversarial attacks to the clean data. In the following, we adopt the minimax formalism of the adversarial training proposed by Madry et al. [33].

3.2.1 *Adversarial attacks*

We consider three benchmark attacks: the fast gradient sign method (FGSM) and the iterative fast gradient sign method (IFGSM) in the ℓ_∞-norm [14] and the Carlini Wagner attack [6] in the ℓ_2-norm (C&W). We denote the classifier defined by a specific DNN as $\tilde{y} = f(\Theta, \mathbf{x})$ for a given instance $(\mathbf{x}, y)$. FGSM searches the adversarial image $\mathbf{x}'$ by maximising the loss $\mathcal{L}(\mathbf{x}', y) \doteq \mathcal{L}(f(\Theta, \mathbf{x}'), y)$ with a maximum allowed perturbation ϵ, i.e., $\|\mathbf{x}' - \mathbf{x}\|_\infty \leqslant \epsilon$. We can approximately solve this constrained optimisation problem by linearise the objective function, i.e.,

$$\mathcal{L}(\mathbf{x}', y) \approx \mathcal{L}(\mathbf{x}, y) + \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)^T \cdot (\mathbf{x}' - \mathbf{x}).$$

Under this linear approximation, the optimal adversarial image is

$$\mathbf{x}' = \mathbf{x} + \epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y)\right). \tag{3.2}$$

IFGSM iterates FGSM to generate the enhanced adversarial images, where the iteration proceeds as follows

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \alpha \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(m-1)}, y)\right), \tag{3.3}$$

where $m = 1, \ldots, M$, $\mathbf{x}^{(0)} = \mathbf{x}$ and α being the step size. Moreover, let the adversarial image be $\mathbf{x}' = \mathbf{x}^{(M)}$ with M being the number of iterations. To ensure the maximum perturbation to the clean image is no bigger than ϵ, in each iteration, we clip the intermediate adversarial images which results in the following attack scheme

$$\mathbf{x}^{(m)} = \operatorname{Clip}_{\mathbf{x},\epsilon}\left\{\mathbf{x}^{(m-1)} + \alpha \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(m-1)}, y)\right)\right\}, \tag{3.4}$$

where $\operatorname{Clip}_{\mathbf{x},\epsilon}(\mathbf{x}')$ limits the change of the generated adversarial image in each iteration, and it is defined as

$$\operatorname{Clip}_{\mathbf{x},\epsilon}(\mathbf{x}') = \min\left\{1, \mathbf{x} + \epsilon, \max\{0, \mathbf{x} - \epsilon, \mathbf{x}'\}\right\},$$

where we assume the pixel value of the image is normalised to $[0, 1]$.

Both FGSM and IFGSM belong to the fixed-perturbation attacks. Moreover, we consider a zero-confidence attack proposed by Carlini and Wagner. For a given image-label pair $(\mathbf{x}, y)$, and for any given label $t \neq y$, C&W attack searches the adversarial image that will be classified to class t with minimum perturbation by solving the following optimisation problem

$$\min_{\boldsymbol{\delta}} ||\boldsymbol{\delta}||_2^2, \tag{3.5}$$

subject to

$$f(\mathbf{x} + \boldsymbol{\delta}) = t, \ \mathbf{x} + \boldsymbol{\delta} \in [0, 1]^n,$$

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

where $\boldsymbol{\delta}$ is the adversarial perturbation (for the sake of simplicity, we ignore the dependence on Θ in f). The equality constraint in (3.5) is hard to tackle, so Carlini and Wagner considered the following surrogate constraint

$$g(\mathbf{x}) = \max\left(\max_{i\neq t}(Z(\mathbf{x})_i) - Z(\mathbf{x})_t, 0\right), \tag{3.6}$$

where $Z(\mathbf{x})$ is the logit vector for an input $\mathbf{x}$, i.e., output of the neural net before the output layer, and $Z(\mathbf{x})_i$ is the logit value corresponding to class i. It is easy to see that $f(\mathbf{x}+\boldsymbol{\delta}) = t$ is equivalent to $g(\mathbf{x}+\boldsymbol{\delta}) \leqslant 0$. Therefore, the problem in (3.5) can be reformulated as

$$\min_{\boldsymbol{\delta}} ||\boldsymbol{\delta}||_2^2 + c \cdot g(\mathbf{x}+\boldsymbol{\delta}), \tag{3.7}$$

subject to

$$\mathbf{x}+\boldsymbol{\delta} \in [0,1]^d,$$

where $c \geqslant 0$ is the Lagrangian multiplier.

By letting $\boldsymbol{\delta} = \frac{1}{2}(\tanh(\mathbf{w})+1) - \mathbf{x}$, (3.7) can be written as an unconstrained optimization problem. Moreover, Carlini and Wagner introduced the confidence parameter κ into the above formulation. In a nutshell, the C&W attack seeks the adversarial image by solving the following problem

$$\begin{aligned}&\min_{\mathbf{w}} \|\frac{1}{2}(\tanh(\mathbf{w})+1) - \mathbf{x}\|_2^2 + c\cdot \\ &\max\left\{-\kappa, \max_{i\neq t}(Z(\frac{1}{2}(\tanh(\mathbf{w}))+1)_i) - Z(\frac{1}{2}(\tanh(\mathbf{w}))+1)_t\right\}.\end{aligned} \tag{3.8}$$

The Adam optimiser [24] can solve the above unconstrained optimisation problem, (3.8), efficiently. All three attacks clip the values of each pixel of the adversarial image $\mathbf{x}'$ to between 0 and 1.

The only difficulty in extending the above three adversarial attacks to DNN-WNLL is again to compute the gradient in backpropagation. Similar to the training of DNN-WNLL, we compute the following surrogate gradient by linearising the WNLL activation. For a given mini-batch of test image-label pairs $(\mathbf{X}, \mathbf{Y})$ and template $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$, we denote the DNN-WNLL as $\tilde{\mathbf{Y}} = \text{WNLL}(Z(\{\mathbf{X}, \mathbf{X}^{\text{te}}\}), \mathbf{Y}^{\text{te}})$, where $Z(\{\mathbf{X}, \mathbf{X}^{\text{te}}\})$ is the composition of the DNN and buffer blocks, as shown in Figure 2(a). By ignoring dependence of the loss function on the parameters, the loss function for DNN-WNLL can be written as $\tilde{\mathcal{L}}(\mathbf{X}, \mathbf{Y}, \mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}}) \doteq \text{Loss}(\hat{\mathbf{Y}}, \mathbf{Y})$. The above three attacks for DNN-WNLL are summarised below.

- **FGSM**

$$\mathbf{X}' = \mathbf{X} + \epsilon \cdot \text{sign}\left(\nabla_{\mathbf{X}} \tilde{\mathcal{L}}(\mathbf{X}, \mathbf{Y}, \mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})\right). \tag{3.9}$$

- **IFGSM**

$$\mathbf{X}^{(m)} = \text{Clip}_{\mathbf{X},\epsilon}[\mathbf{X}^{(m-1)} + \alpha \cdot \text{sign}\left(\nabla_{\mathbf{X}} \tilde{\mathcal{L}}(\mathbf{X}^{(m-1)}, \mathbf{Y}, \mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})\right)], \tag{3.10}$$

where $m = 1, 2, \ldots, M$; $\mathbf{X}^{(0)} = \mathbf{X}$ and $\mathbf{X}' = \mathbf{X}^{(M)}$.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

- **C&W**

$$\min_{\mathbf{W}} ||\frac{1}{2}(\tanh(\mathbf{W})+1)-\mathbf{X}||_2^2+ \tag{3.11}$$
$$c\cdot\max\left[-\kappa,\max_{\mathbf{i}\neq\mathbf{t}}\left(Z\left(\frac{1}{2}(\tanh(\mathbf{W}))+1\right)_{\mathbf{i}}\right)-Z\left(\frac{1}{2}(\tanh(\mathbf{W}))+1\right)_{\mathbf{t}}\right],$$

where $\mathbf{i}$ are the logit values of the input images $\mathbf{X}$, $\mathbf{t}$ are the target labels.

In the above attacks, $\nabla_{\mathbf{X}}\tilde{\mathcal{L}}$ is required to generate the adversarial images. In the DNN-WNLL, this gradient is difficult to compute. As shown in Figure 2(b), we approximate $\nabla_{\mathbf{X}}\tilde{\mathcal{L}}$ in the following way

$$\nabla_{\mathbf{X}}\tilde{\mathcal{L}}=\frac{\partial\mathcal{L}^{\text{WNLL}}}{\partial\hat{\mathbf{Y}}}\cdot\frac{\partial\hat{\mathbf{Y}}}{\partial\hat{\mathbf{X}}}\cdot\frac{\partial\hat{\mathbf{X}}}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\mathbf{X}}\approx\frac{\partial\mathcal{L}^{\text{Linear}}}{\partial\tilde{\mathbf{Y}}}\cdot\frac{\partial\tilde{\mathbf{Y}}}{\partial\hat{\mathbf{X}}}\cdot\frac{\partial\hat{\mathbf{X}}}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\mathbf{X}}, \tag{3.12}$$

again, in the above approximation, we set the value of $\mathcal{L}^{\text{Linear}}$ to that of $\tilde{\mathcal{L}}$.

Based on our numerical experiments, the batch size of $\mathbf{X}$ has a negligible influence on the adversarial attack and defense. In all of our experiments, we choose the size of both mini-batches $\mathbf{X}$ and the template to be 500.

3.2.2 *Adversarial training*

We apply the PGD adversarial training [33] to train the adversarially robust DNNs, where we approximately solve the EARM (1.1) by using the PGD adversarial images, i.e., IFGSM attacks with an initial random perturbation on the clean images, to approximate the solution of the inner maximisation problem. We summarise the PGD adversarial training for DNNs with the WNLL activation, as shown in Figure 2(a), in Algorithm 3.

3.3 Semi-supervised learning

Semi-supervised learning is another fundamental learning paradigm, where we have access to a large amount of training data. However, most of the training data are unlabelled. Semi-supervised learning is of particular importance in, e.g., medical applications [7]. It is straightforward to extend DNNs with the WNLL activation to semi-supervised learning. Let the labelled and unlabelled training data be $\{\mathbf{X}^l,\mathbf{Y}^l\}$ and $\{\mathbf{X}^{ul},\mathbf{Y}^{ul}\}$, respectively. There are two approaches to semi-supervised learning by using DNN-WNLL.

- **Approach I:** Train DNN-WNLL on only labelled data $\{\mathbf{X}^l,\mathbf{Y}^l\}$. During testing, we feed the unlabelled data together with the labelled template data to predict labels for the testing data. This is essentially similar to the classical graph Laplacian-based semi-supervised learning on the DL features.
- **Approach II:** Train DNNs with the WNLL activation by using both labelled $\{\mathbf{X}^l,\mathbf{Y}^l\}$ and unlabelled $\{\mathbf{X}^{ul},\mathbf{Y}^{ul}\}$ data. During training, we use both labelled and unlabelled data to build a graph for WNLL interpolation, and then we backpropagate loss between predicted and true labels of the labelled data. The testing phase is the same as that in Approach I.

In this work, we focus on the Approach I.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Algorithm 3 DNN with the WNLL Output Activation: PGD Adversarial Training

Input: Training set: (data, label) pairs $(\mathbf{X}, \mathbf{Y})$, the number of PGD iterations M, PGD attack step size α, maximum PGD perturbation ϵ. The number of alternating iterations N, and the number of epochs used to train DNN with the linear activation N_1 and the WNLL activation N_2.
Output: An optimised DNN-WNLL.
for iter $= 1, \ldots, N$ **do**
  //PGD adversarial training of the left branch: DNN with linear activation.
  Train DNN + Linear blocks.
  Partition the training data into M_1 mini-batches, i.e., $(\mathbf{X}, \mathbf{Y}) = \bigcup_{i=1}^{M_1} (\mathbf{X}_i, \mathbf{Y}_i)$.
  for epoch$_1 = 1, \ldots, N_1$ **do**
    for $i = 1, \ldots, M_1$ **do**
      //Attack the input images by PGD attack.
      $\mathbf{X}_i = \mathbf{X}_i + \mathbf{U}(-\epsilon, \epsilon)$ with $\mathbf{U}(-\epsilon, \epsilon)$ be a uniform random vector.
      for iter$_1 = 1, \ldots, M$ **do**
        Attack $\mathbf{X}_i$ according to (3.4).
      Backpropagate the classification error of the adversarial images.
  //PGD adversarial training of the right branch: DNN with WNLL activation.
  Split $(\mathbf{X}, \mathbf{Y})$ into training data and template, i.e., $(\mathbf{X}, \mathbf{Y}) \doteq (\mathbf{X}^{\mathrm{tr}}, \mathbf{Y}^{\mathrm{tr}}) \bigcup (\mathbf{X}^{\mathrm{te}}, \mathbf{Y}^{\mathrm{te}})$.
  Partition the training data into M_2 mini-batches, i.e., $(\mathbf{X}^{\mathrm{tr}}, \mathbf{Y}^{\mathrm{tr}}) = \bigcup_{i=1}^{M_2} (\mathbf{X}_i^{\mathrm{tr}}, \mathbf{Y}_i^{\mathrm{tr}})$.
  for epoch$_2 = 1, \ldots, N_2$ **do**
    for $i = 1, \ldots, M_2$ **do**
      //Attack the input training images by PGD attack.
      $\mathbf{X}_i^{\mathrm{tr}} = \mathbf{X}_i^{\mathrm{tr}} + \mathbf{U}(-\epsilon, \epsilon)$.
      for iter$_1 = 1, \ldots, M$ **do**
        Attack $\mathbf{X}_i^{\mathrm{tr}}$ according to (3.10).
      Backpropagate the classification error of the adversarial images.

4 Numerical results

In this section, we will numerically verify the accuracy and robustness of DNN-WNLL. Moreover, we show that DNN-WNLL is suitable for data-efficient learning. We also provide results of semi-supervised learning by using DNN-WNLL. We implement our algorithm on the PyTorch platform [42]. All the computations are carried out on a machine with a single Nvidia Titan Xp graphics card.

To validate the classification accuracy, efficiency and robustness of the proposed framework, we test the new architecture and algorithm on the CIFAR10, CIFAR100 [25], MNIST [28] and SVHN data sets [37]. In all the experiments below, we apply the standard data augmentation that is used for the CIFAR data sets [19, 22, 55]. For MNIST and SVHN, we use the raw data without any data augmentation.

Before diving into the performance of DNNs with different output activation functions, we first compare the performance of the WNLL with the softmax on the raw input images for various data sets. The training sets are used to train the softmax classifier and interpolate labels for the test set in the WNLL interpolation, respectively. Table 2 lists the classification accuracies of the

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 2. *Accuracies of the softmax and the WNLL classifiers in classifying some benchmark data sets*

Data set	CIFAR10 (%)	MNIST (%)	SVHN (%)
softmax	39.91	92.65	24.66
WNLL	40.73	97.74	56.17

WNLL and the softmax on three data sets. For the WNLL interpolation, we only use the top 30 nearest neighbours to ensure sparsity of the weight matrix to speed up the computation, and the 15th neighbour's distance is used to normalise the weight matrix. WNLL outperforms softmax remarkably in all the three benchmark tasks especially for the MNIST (test accuracy: 92.65% vs. 97.74%) and SVHN (test accuracy: 24.66% vs. 56.17%) classification. These results indicate potential benefits of using the WNLL instead of the softmax as the output activation in DNNs.

For natural training of the DNN-WNLL: We take two passes of the alternating step, i.e., set $N = 2$ in Algorithm 1. For training of the linear activation stage (Stage 1), we train the network for $n = 400$ epochs with stochastic gradient descent. For the training of the WNLL activation stage (Stage 2) we train for $n = 5$ epochs. In the first pass, the initial learning rate is 0.05 and halved after every 50 epoch in training DNNs with linear activation, and a fixed learning rate 0.0005 is used to train DNNs with the WNLL activation. The same Nesterov momentum and weight decay as that used in [19, 23] are employed for the CIFAR and the SVHN experiments, respectively, in our work. In the second pass, the learning rate is set to be one-fifth of the corresponding epochs in the first pass. The batch sizes are 128 and 2000 when training softmax/linear and WNLL activated DNNs, respectively. For a fair comparison, we train the vanilla DNNs with the softmax output activation for 810 epochs with the same optimiser used in the WNLL activated ones.

4.1 Data-efficient learning – Small training data case

When we do not have a sufficient amount of labelled training data to train a high capacity deep network, the generalisation accuracy of the trained model typically decays as the network goes deeper. We illustrate this in Figure 3. The WNLL-activated DNNs, with its superior regularisation power and perturbation capability on bad local minima, can overcome this generalisation degradation. The left and right panels of Figure 3 plot the results of DNNs with the softmax and the WNLL activation that are trained on 1K and 10K images, respectively. These results show that the generalisation error rate decays consistently as the network goes deeper in DNN-WNLL. Moreover, the generalisation accuracy between the vanilla and the WNLL-activated DNNs can differ up to 10 percent within our testing regime.

Figure 4 plots the evolution of generalisation accuracy during training. We compute the test accuracy per epoch. Panels (a) and (b) plot the test accuracies for the ResNet50 with the softmax and the WNLL activation (1–400 and 406–805 epochs correspond to linear activation), respectively, with only the first 1K instances in the training set of CIFAR10, are used to train the models. Charts (c) and (d) are the corresponding plots with 10K training instances, using a pre-activated ResNet50. After around 300 epochs, the accuracies of the vanilla DNNs plateau and cannot improve anymore. However, the test accuracy for WNLL jumps at the beginning of

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

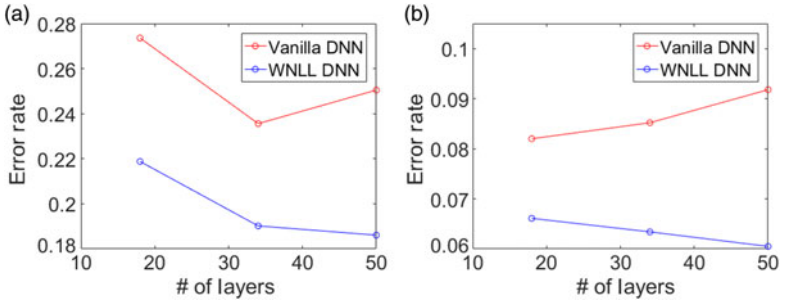


FIGURE 3. Plots of test errors when 1K (a) and 10K (b) training data are used to train the vanilla and the WNLL-activated DNNs. In each plot, we test three different deep networks: PreResNet18, PreResNet34 and PreResNet50. All tests are done on the CIFAR10 data set.

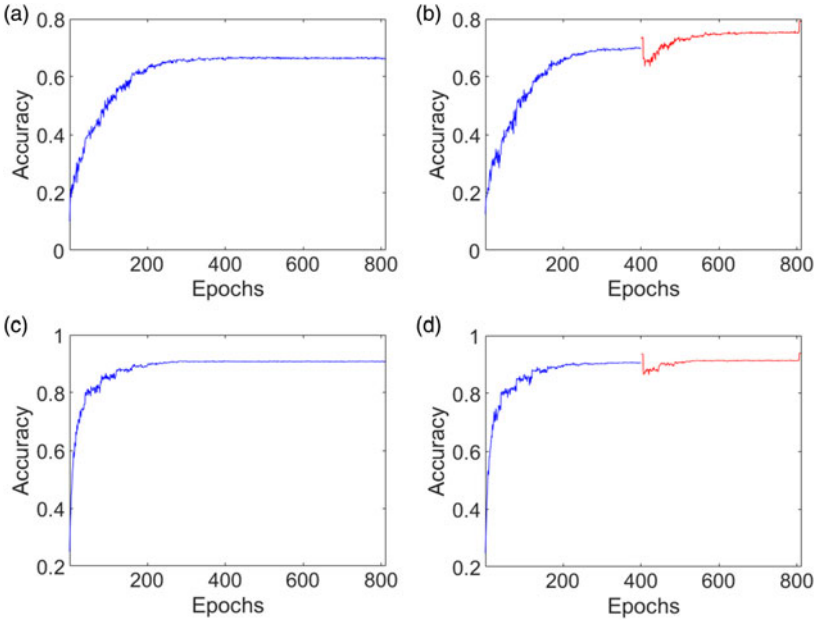


FIGURE 4. Evolution of the generation accuracy over the training procedure. Charts (a) and (b) plot the accuracy evolution of ResNet50 with the softmax and the WNLL activation trained with 1K training data, respectively. Panels (c) and (d) correspond to the case of 10K training data for PreResNet50. All tests are done on the CIFAR10 data set

Stage 2 in the first pass; during the Stage 1 of the second pass, even though initially there is an accuracy reduction, the accuracy continues to climb and eventually surpasses that of the WNLL activation in Stage 2 of the first pass. The jumps in accuracy at epoch 400 and 800 are due to switching from linear activation to WNLL for predictions on the test set. The initial decay when alternating back to the softmax is caused partially by the final layer W_L not being tuned with respect to the deep features $\tilde{\mathbf{X}}$ and partially due to predictions on the test set being made by the softmax instead of the WNLL. Nevertheless, the perturbation via the WNLL activation quickly results in the accuracy increasing beyond the linear stage in the previous pass.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 3. *Test errors of the vanilla DNNs, SVM and WNLL activated ones trained on the entire, the first* 10*K and the first* 1*K instances of the training set of the CIFAR10 data set (Median of 5 independent trials)*

	Whole			10,000		1000	
Network	Vanilla (%)	WNLL (%)	SVM (%)	Vanilla (%)	WNLL (%)	Vanilla (%)	WNLL (%)
VGG11	9.23	**7.35**	9.28	10.37	**8.88**	26.75	**24.10**
VGG13	6.66	**5.58**	7.47	9.12	**7.64**	24.85	**22.56**
VGG16	6.72	**5.69**	7.29	9.01	**7.54**	25.41	**22.23**
VGG19	6.95	**5.92**	7.99	9.62	**8.09**	25.70	**22.87**
ResNet20	9.06	**7.09**	9.60	12.83	**9.96**	34.90	**29.91**
ResNet32	7.99	**5.95**	8.73	11.18	**8.15**	33.41	**28.78**
ResNet44	7.31	**5.70**	8.67	10.66	**7.96**	34.58	**27.94**
ResNet56	7.24	**5.61**	8.58	9.83	**7.61**	37.83	**28.18**
ResNet110	6.41	**4.98**	8.06	8.91	**7.13**	42.94	**28.29**
ResNet18	6.16	**4.65**	6.00	8.26	**6.29**	27.02	**22.48**
ResNet34	5.93	**4.26**	6.32	8.31	**6.11**	26.47	**20.27**
ResNet50	6.24	**4.17**	6.63	9.64	**6.49**	29.69	**20.19**
PreResNet18	6.21	**4.74**	6.38	8.20	**6.61**	27.36	**21.88**
PreResNet34	6.08	**4.40**	5.88	8.52	**6.34**	23.56	**19.02**
PreResNet50	6.05	**4.27**	5.91	9.18	**6.05**	25.05	**18.61**

4.2 Generalisation of naturally trained DNN-WNLL

We next show the superiority of the DNN-WNLL in terms of generalisation accuracies when compared to their surrogates with the softmax or the SVM output activation functions. Besides ResNets, we also test the WNLL surrogate on the VGG networks. In Table 3, we list the generalisation errors for 15 different DNNs from VGG, ResNet, Pre-activated ResNet families trained on the entire, first 10K and first 1K instances of the CIFAR10 training set. We observe that WNLL, in general, improves more for ResNets and pre-activated ResNets, with less but still remarkable improvements for the VGGs. Except for VGGs, we can achieve a relatively 20–30% testing error rate reduction across all neural nets. All results presented here and in the rest of this paper are the median of five independent trials. We also compare with SVM as an alternative output activation and observe that the performance are still inferior to the DNN-WNLL. Note that the bigger batch size is to ensure the interpolation quality of the WNLL. A reasonable concern is that the performance increase comes from the variance reduction due to increasing the batch size. However, experiments were done with a batch size of 2000 for vanilla networks deteriorates the test accuracy.

We list the error rates of the 15 different DNNs with either the softmax or the WNLL activation on the CIFAR10 and CIFAR100 in Tables 3 and 4, respectively. On the CIFAR10, DNN-WNLL outperforms the vanilla ones with around 1.5–2.0% absolute, or 20–30% relative error rate reduction. The improvements on the CIFAR100 by using the WNLL activation are more remarkable than that on the CIFAR10. We independently ran the vanilla DNNs on both data sets, and our results are consistent with the original reports and other researchers' reproductions [19, 17, 22]. We provide experimental results of DNNs' performance on SVHN data in Table 5. Interestingly,

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 4. *Test errors of the vanilla DNNs vs. the WNLL-activated DNNs on the CIFAR100 data set (Median of 5 independent trials)*

Network	Vanilla DNN (%)	WNLL DNN (%)	Network	Vanilla DNN (%)	WNLL DNN (%)
VGG11	32.68	**28.80**	ResNet110	28.86	**23.74**
VGG13	29.03	**25.21**	ResNet18	27.57	**22.89**
VGG16	28.59	**25.72**	ResNet34	25.55	**20.78**
VGG19	28.55	**25.07**	ResNet50	25.09	**20.45**
ResNet20	35.79	**31.53**	PreResNet18	28.62	**23.45**
ResNet32	32.01	**28.04**	PreResNet34	26.84	**21.97**
ResNet44	31.07	**26.32**	PreResNet50	25.95	**21.51**
ResNet56	30.03	**25.36**			

Table 5. *Test errors of the vanilla DNNs vs. the WNLL-activated DNNs on the SVHN data set (Median of 5 independent trials)*

Network	Vanilla DNN (%)	WNLL DNN (%)	Network	Vanilla DNN (%)	WNLL DNN (%)
ResNet20	3.76	**3.44**	ResNet18	3.96	**3.65**
ResNet32	3.28	**2.96**	ResNet34	3.81	**3.54**
ResNet44	2.84	**2.56**	PreResNet18	4.03	**3.70**
ResNet56	2.64	**2.32**	PreResNet34	3.66	**3.32**
ResNet110	2.55	**2.26**			

Table 6. *Running time and GPU memory for ResNet20 and ResNet20-WNLL*

Network	Training time (50K data) (s)	Test time (10K data) (s)	Memory (MB)
ResNet20	3925.6	0.657	1007
ResNet20-WNLL	7378.4	14.09	1563

the improvement is more significant on more challenge tasks which suggest a potential for our methods to succeed on other tasks/data sets.

Table 6 compares the computational cost in both training and test between ResNet20 and ResNet20-WNLL, where the computation is performed on a single Titan Xp GPU.

4.3 Adversarial robustness

We carry out experiments on the benchmark MNIST and CIFAR10 data sets to show the efficiency of using the graph interpolating activation for adversarial defense. For MNIST, we train the Small-CNN that is used in [56] by running 100 epochs of PGD adversarial training with $\epsilon = 0.3$, $\alpha = 0.01$ and $M = 40$. We let the initial learning rate be 0.1 and decay by a factor of 10 at the 50th epoch. For CIFAR10, we consider three benchmark models: ResNet20, ResNet56 and WideResNet34. We train these models on the CIFAR10 data set by running 120 epochs of

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 7. *Natural and robust accuracies under different white-box adversarial attacks of different robustly trained models on the MNIST data set*

Model	$\mathcal{A}_{\text{nat}}$ (%)	$\mathcal{A}_{\text{rob (FGSM)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{40}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{100}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (C\&W)}}$ (%)
Small-CNN	99.33	98.17	96.27	96.09	95.31
Small-CNN-WNLL	99.39	98.35	97.36	96.90	97.55

PGD adversarial training with $\epsilon = 8/255$, $\alpha = 2/255$ and $M = 10$. The initial learning rate is set to be 0.1 and decays by a factor of 10 at the 80th, 100th and 110th epochs, respectively. After the robust models have been trained by the PGD adversarial training, we test their natural accuracies on the clean images and robust accuracies on the adversarial images crafted by attacking these robustly trained models by the aforementioned three adversarial attacks, where the parameters are set as follows

- **FGSM:** In equations (3.2) and (3.9), we let $\epsilon = 8/255$ and 0.3 to attack DNNs for CIFAR10 and MNIST classification, respectively.
- **IFGSM:** We denote the n-step IFGSM attack as IFGSM^n. To attack DNNs for CIFAR10 classification, we let $\epsilon = 8/255$ and $\alpha = 1/255$ in equations (3.4) and (3.10) for both IFGSM^{10} and IFGSM^{20} attacks. For MNIST, we let $\epsilon = 0.3$ and $\alpha = 0.01$ in equations (3.4) and (3.10) for IFGSM^{40} and IFGSM^{100} attacks.
- **C&W:** For adversarial attack on the CIFAR10 data set, we let $\kappa = 0$ and $c = 10$ in equations (3.8) and (3.11), and we run 50 iterations of the Adam optimiser with learning rate 0.006 to find the optimal C&W attack in ℓ_2-norm on the clean images. To search for the optimal C&W attack in ℓ_2-norm on the MNIST data, we run 100 iterations of the Adam optimiser with learning rate 0.003 with $\kappa = 0$ and $c = 10$ in equations (3.8) and (3.11).

We consider both white-box and black-box attacks. In the black-box attack, we apply the given adversarial attack to attack another oracle model in the white-box fashion, and then we use the target model to classify the adversarial images crafted by attacking the oracle model.

Table 7 lists both natural and robust accuracies of the PGD adversarially trained Small-CNN with either the softmax or the WNLL output activation function on the MNIST. Small-CNN with the WNLL activation is remarkably more accurate on both clean and adversarial images, e.g., for Small-CNN, the natural accuracies for the softmax and the WNLL activation functions are 99.33% and 99.39%, respectively. The robust accuracies for Small-CNN and Small-CNN-WNLL are 98.17% vs. 98.35%, 96.27% vs. 97.36%, 96.09% vs. 96.90% and 95.31% vs. 97.55%, respectively, to the FGSM, IFGSM^{40}, IFGSM^{100} and C&W attacks in the white-box scenario. We regard Small-CNN as the oracle model to perform black-box attacks on the Small-CNN-WNLL, the corresponding robust accuracies to the above four adversarial attacks are listed in Table 8. In the MNIST experiment, black-box attacks are less effective than the white-box attacks.

Next, we consider the adversarial defense capability of DNNs with the WNLL activation on the CIFAR10 data set. Table 9 lists the natural and robust accuracies, under the white-box attacks, of the standard ResNet20, ResNet56 and WideResNet34-10 and their counterpart with the WNLL activation. These results show that the robustly trained ResNets with the WNLL activation slightly improves natural accuracies on the clean images, while the robust accuracies are significantly improved. For instance, under the FGSM and C&W attacks, the WNLL activation

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 8. *Robust accuracies under different black-box adversarial attacks of different robustly trained models on the MNIST data set*

Model	Oracle	$\mathcal{A}_{\text{rob (FGSM)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{40}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{100}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (C\&W)}}$ (%)
Small-CNN-WNLL	Small-CNN	98.40	97.47	97.40	98.14

Table 9. *Natural and robust accuracies under different white-box adversarial attacks of different robustly trained models on the CIFAR10 data set*

Model	$\mathcal{A}_{\text{nat}}$(%)	$\mathcal{A}_{\text{rob (FGSM)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{20}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{100}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (C\&W)}}$ (%)
ResNet20	75.11	50.89	46.03	46.01	58.73
ResNet20-WNLL	75.53	55.76	53.31	53.26	63.82
ResNet56	79.32	55.05	50.98	50.06	61.75
ResNet56-WNLL	79.52	60.50	58.19	57.26	67.93
WideResNet34	84.05	51.93	48.93	48.32	59.04
WideResNet34-WNLL	84.95	65.50	63.03	62.25	72.37

Table 10. *Robust accuracies under different black-box adversarial attacks of different robustly trained models on the CIFAR10*

Model	Oracle	$\mathcal{A}_{\text{rob (FGSM)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{20}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{100}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (C\&W)}}$ (%)
ResNet20-WNLL	ResNet20	55.91	53.44	53.35	65.13
ResNet56-WNLL	ResNet56	60.00	57.94	57.85	70.47
WideResNet34-WNLL	WideResNet34	67.19	67.07	67.17	81.17

can boost robust accuracy by $\sim 5\%$; and under the IFGSM^{20} and IFGSM^{40} attacks, the robust accuracy improvement is up to $\sim 7\%$. For the WideResNet34-10, under the IFGSM^{20} attack, we achieve accuracy 63.03% which outperforms the results of Zhang et al. [56] (56.61%) by more than 6%. For black-box attacks on DNNs with the WNLL activation, we regard the counterpart DNNs with the softmax activation as the oracle models. The robust accuracies of ResNet20-WNLL, ResNet56-WNLL and WideResNet34-10-WNLL are listed in Table 10. Again, the black-box attacks are less effective than the white-box ones.

Furthermore, we consider the influence of the number of nearest neighbours m with the nth-nearest neighbour used to normalise the weights in (2.3) in the WNLL interpolation. Table 11 lists the natural and robust accuracies of the ResNet56-WNLL with the different number of nearest neighbours, (m, n), involved in the WNLL interpolation. The natural accuracy decays as a greater number of nearest neighbours are used for interpolation, and the robust accuracies are maximised when $(m, n) = (30, 15)$. When more nearest neighbours are used for interpolation, the robust accuracies decay. This issue might be due to the fact that these nearest neighbours are only selected from a finite number of data points and the resulting nearest neighbours are far from the real nearest neighbours.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 11. *Natural and robust accuracies under different white-box adversarial attacks of ResNet56-WNLL with different number of points, in the form* (m, n)*, used for interpolation on the CIFAR10 data set*

(m, n)	$\mathcal{A}_{\text{nat}}$ (%)	$\mathcal{A}_{\text{rob (FGSM)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{20}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (IFGSM}^{100}\text{)}}$ (%)	$\mathcal{A}_{\text{rob (C\&W)}}$ (%)
(15, 8)	79.89	59.71	57.85	56.53	67.91
(30, 15)	79.52	60.50	58.19	57.26	67.93
(45, 23)	78.92	59.50	57.94	57.06	66.26
(60, 30)	77.92	58.04	55.80	54.97	67.74

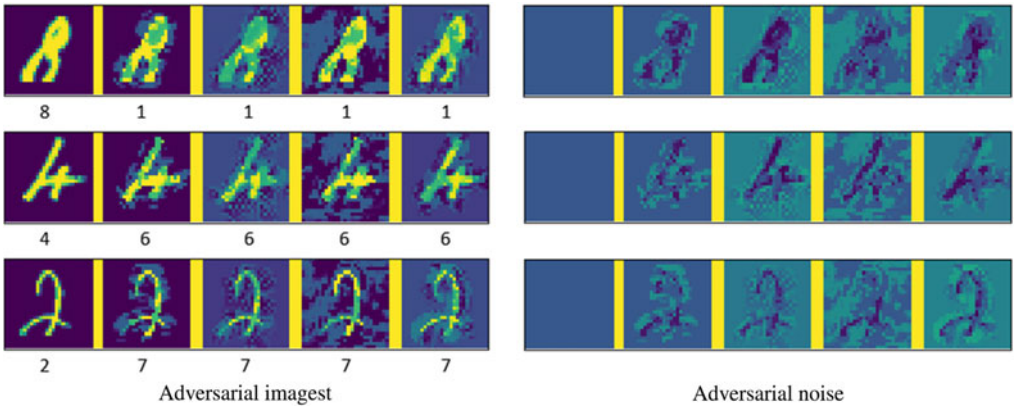


FIGURE 5. Adversarial images (left panel) selected from the MNIST data set and the corresponding adversarial noise (right panel). Column 1: cleaning image and noise (no noise in this case); Column 2–3: adversarial images and noise crafted by IFGSM^{40} and C&W attacks on the small CNN, respectively; Column 4–5: adversarial images and noise crafted by IFGSM^{40} and C&W attacks on the small CNN-WNLL, respectively. The predicted labels for the adversarial images are listed below the adversarial images in the left panel.

Finally, let us look at the adversarial images and the adversarial noise crafted by adversarial attacks on DNNs with both softmax and WNLL activation functions. Figures 5 and 6 depict adversarial images and adversarial noise of the MNIST and the CIFAR10 obtained by applying different adversarial attacks to Small-CNN and ResNet20 with both softmax and WNLL activation functions. All these adversarial images are misclassified by DNNs with both the softmax and the WILL activation. However, they can be easily classified by human beings.

4.4 Semi-supervised learning

In this subsection, we apply the DNN-WNLL to semi-supervised learning where we have access to all the training data of the CIFAR10 but only part of them are labelled. We can use the unlabelled data to build a graph for the WNLL interpolation in semi-supervised learning, while not in data-efficient learning. We list the accuracies of the semi-supervised learning when 1K and 10K training data are labelled to train DNNs in Table 12. Compared to the results in Table 3, semi-supervised learning has better accuracy with the same number of labelled training data.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

Table 12. *Test error of DNNs with the WNLL output activation for the CIFAR10 classification in the semi-supervised learning setting*

Network	1K (Labeled)/49K (Unlabelled)	10K (Labelled)/40K (Unlabelled)
ResNet20-WNLL	27.02%	9.01%
ResNet32-WNLL	26.28%	7.53%
ResNet44-WNLL	25.63%	7.25%
ResNet56-WNLL	25.53%	6.99%
ResNet110-WNLL	25.38%	6.50%

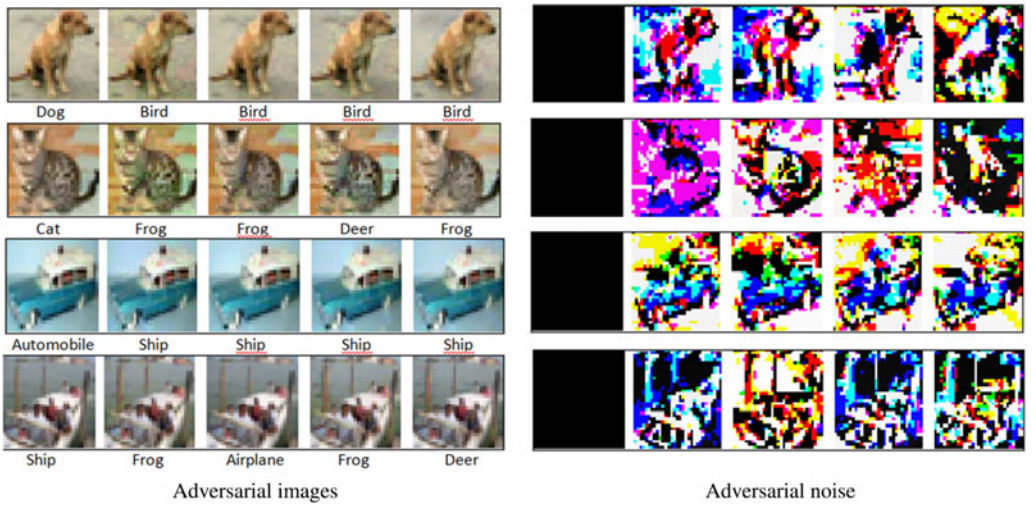


FIGURE 6. Adversarial images (left panel) selected from the CIFAR10 data set and the corresponding adversarial noise (right panel). Column 1: cleaning image and noise (no noise in this case); Column 2–3: adversarial images and noise crafted by IFGSM[40] and C&W attacks on the ResNet20, respectively; Column 4–5: adversarial images and noise crafted by IFGSM[20] and C&W attacks on the ResNet20-WNLL, respectively. The predicted labels for the adversarial images are listed below the adversarial images in the left panel.

5 Geometric explanations

In this section, we will consider the representations learned by DNNs with two different output activation functions. As an illustration, we randomly select 1000 training instances and 100 testing data each for the airplane and automobile classes from the CIFAR10 data set. We consider two different strategies to visualise the features learned by ResNet56 and ResNet56-WNLL for the above randomly selected data.

- **Strategy I:** Apply the principal component analysis (PCA) to reduce the 64D features output right before the softmax/WNLL activation to 2D.
- **Strategy II:** Add an additional FC layer before the output activation function. This FC layer will help to learn the 2D representations.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

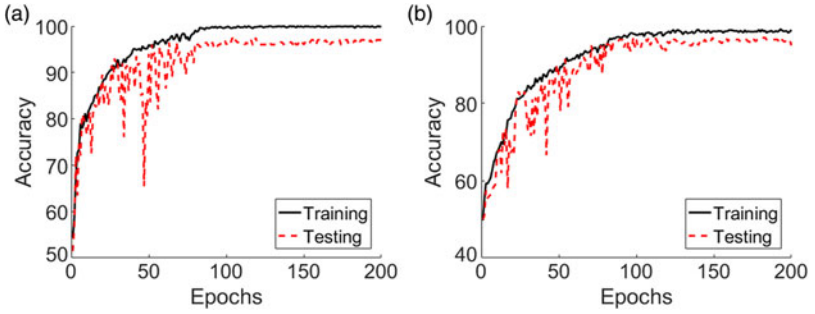


FIGURE 7. Epochs vs. accuracy in training ResNet56 on the CIFAR10. (a): Without the additional FC layer; (b): with the additional FC layer.

We first show that in Strategy II, the newly added FC layer does not affect the performance of the original ResNet56 much. We train and test the ResNet56 with and without the additional FC layer on the aforementioned randomly selected training and testing data. As shown in Figure 7, the training and testing accuracies evolution are essentially the same for ResNet56 with and without the additional FC layer.

5.1 Improving generalisation

Figure 8 plots the representations for the selected airplane and automobile data from the CIFAR10 data set. First, panels (a) and (b) show the features of the test set learned by ResNet56 visualised by the proposed two strategies. In both cases, the features are well separated, in general, with a small overlapping which causes some misclassification. Charts (c) and (d) depict the first two principal components (PCs) learned by ResNet56-WNLL for the selected training and testing data. The PCs of the features learned by ResNet56-WILL are better separated than that of ResNet56's (Figure 8), and it indicates that ResNet56-WILL is more accurate in classifying the randomly selected data.

5.2 Improving adversarial robustness

First, let us look at how the adversarial attack changes the geometry of the learned representations. We consider the simple one-step IFGSM attack, IFGSM[1], with the same parameters used before. Figure 9 shows the first two PCs of the representations learned by ResNet56 and ResNet56-WNLL for the adversarial test images. These PCs show that the adversarial attack makes the features of the two different classes mixed and therefore drastically reduces the classification accuracy.

Second, we consider how the WNLL interpolation helps to improve adversarial robustness. We randomly pick up an adversarial image that is misclassified by the standard DNNs with the softmax activation from the MNIST and the CIFAR10, respectively. The top five nearest neighbours in the deep feature space from the clean training data, of these two adversarial images,

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

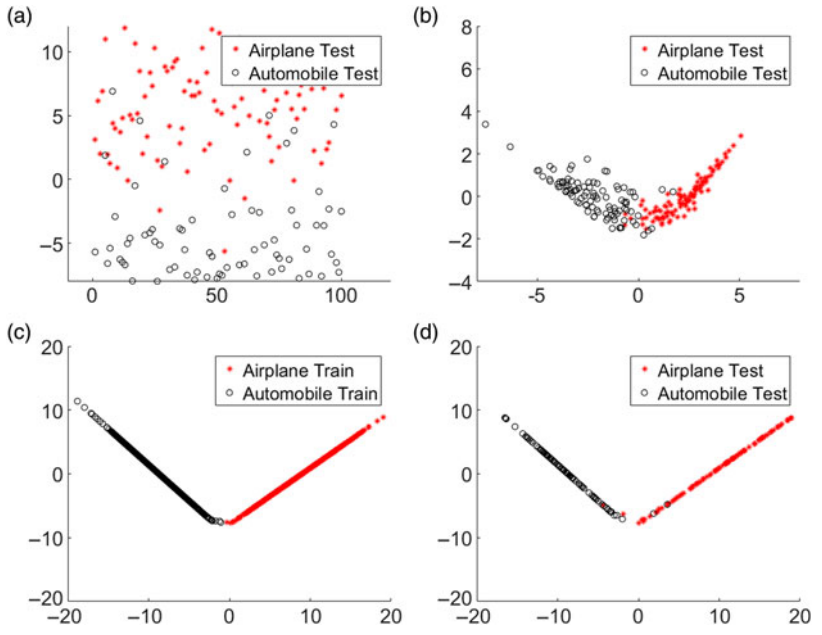


FIGURE 8. Visualisation of the features learned by ResNet56 with the softmax ((a), (b)) and the WNLL ((c), (d)) activation functions. (a): The 2D features of the airplane and automobile data in the test set learned by the ResNet56 with an additional 2×2 linear layer; (b): the first two principal components of the features of the airplane and automobile data in the test set learned by the ResNet56; (c) and (d) plot the first two principal components of features of the airplane and automobile data in the training and test set learned by the ResNet56-WNLL. All experiments are done on the CIFAR10 data set.

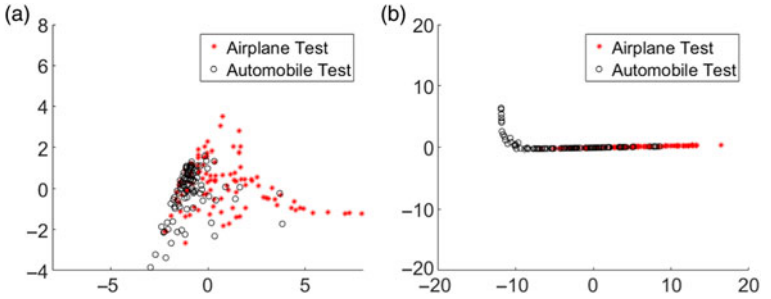
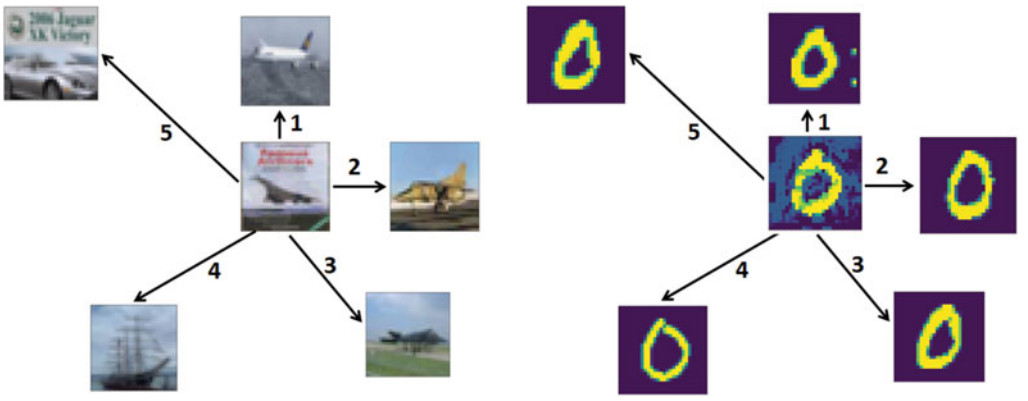


FIGURE 9. Visualisation of the first two principal components of the adversarial images' (IFGSM^1 attack) features learned by ResNet56 with the softmax (a) and the WNLL (b) activation functions, respectively.

are shown in Figure 10. For the MNIST digit, all the nearest neighbours belong to the same class as the adversarial image; and for the CIFAR10 adversarial image, the top three neighbours belong to the same category as the adversarial one. These nearest neighbours will guide DNN-WNLL to classify the adversarial images correctly.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press



MNIST CIFAR10

FIGURE 10. A randomly selected adversarial image and their top five nearest neighbours in the clean training set searched based on the distance between features output from the layer before the WNLL activation layer. Left: IFGSM^{40} attack on the small CNN-WNLL; Right: IFGSM^{20} attack on the ResNet20-WNLL

6 Concluding remarks

In this paper, we leveraged ideas from the manifold learning and proposed to replace the output activation function of the conventional deep neural nets, typically the softmax function, with a graph Laplacian-based high-dimensional interpolating function. This simple modification is applicable to any of the existing off-the-shelf DNNs with the softmax activation enables DNNs to make sufficient use of the manifold structure of data. Furthermore, we developed end-to-end and multi-staged training and testing algorithms for the proposed DNN with the interpolating function as its output activation. On the one hand, the proposed new framework remarkably improves both generalisability and robustness of the baseline DNNs; on the other hand, the new framework is suitable for data-efficient ML. These improvements are consistent across networks of different types and with a different number of layers. The increase in generalisation accuracy could also be used to train smaller models with the same accuracy, which has great potential for mobile device applications.

In this work, we utilised a special kind of graph interpolating function as DNNs' output activation. An alternative approach is to learn such an interpolating function instead of using one which is fixed. This approach is under our consideration.

Acknowledgements

This material is based on research sponsored by the NSF grant DMS-1924935 and DMS-1952339, the DOE grant DE-SC0021142, and the Air Force Research Laboratory under grant numbers FA9550-18-0167 and MURI FA9550-18-1-0502.

Conflicts of interest

None.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

References

[1] AGOSTINELLI, F., HOFFMAN, M., SADOWSKI, P. & BALDI, P. (2014) Learning Activation Functions to Improve Deep Neural Networks. arXiv preprint arXiv:1412.6830.

[2] ANONYMOUS. (2019) Adversarial Machine Learning against Tesla's Autopilot. https://www.schneier.com/blog/archives/2019/04/adversarial_mac.html.

[3] ATHALYE, A., CARLINI, N. & WAGNER, D. (2018) Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples. In: *International Conference on Machine Learning*.

[4] BENGIO, Y., LAMBLIN, P., POPOVICI, D. & LAROCHELLE, H. (2007) Greedy layer-wise training of deep networks. In: *Advances in Neural Information Processing Systems*.

[5] BRENDEL, W., RAUBER, J. & BETHGE, M. (2017) Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248.

[6] CARLINI, N. & WAGNER, D. A. (2016) Towards evaluating the robustness of neural networks. In: *IEEE European Symposium on Security and Privacy*, pp. 39–57.

[7] CHAPELLE, O., SCHOLKOPF, B. & ZIEN, A. (2006) *Semi-supervised Learning*, MIT Press, Cambridge, Massachusetts.

[8] CHEN, X., LIU, C., LI, B., LIU, K. & SONG, D. (2017a) Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv preprint arXiv:1712.05526.

[9] CHEN, Y., LI, J., XIAO, H., JIN, X., YAN, S. & FENG, J. (2017b) Dual path networks. In: *Advances in Neural Information Processing Systems*.

[10] COHEN, J., ROSENFELD, E. & KOLTER, J. Z. (2019) Certified Adversarial Robustness via Randomized Smoothing. arXiv preprint arXiv:1902.02918v1.

[11] DOU, Z., OSHER, S. J. & WANG, B. (2018) Mathematical Analysis of Adversarial Attacks. arXiv preprint arXiv:1811.06492.

[12] GLOROT, X., BORDES, A. & BENGIO, Y. (2011) Deep sparse rectifier neural networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323.

[13] GOODFELLOW, I., WARDE-FARLEY, D., MIRZA, M., COURVILLE, A. & BENGIO, Y. (2013) Maxout networks. arXiv preprint arXiv:1302.4389.

[14] GOODFELLOW, I. J., SHLENS, J. & SZEGEDY, C. (2014) Explaining and Harnessing Adversarial Examples. arXiv preprint arXiv:1412.6275.

[15] GUO, C., RANA, M., CISSE, M. & VAN DER MAATEN, L. (2018) Countering adversarial images using input transformations. In: *International Conference on Learning Representations*. https://openreview.net/forum?id=SyJ7ClWCb.

[16] HE, K., ZHANG, X., REN, S. & SUN, J. (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

[17] HE, K., ZHANG, X., REN, S. & SUN, J. (2016a) Identity mappings in deep residual network. In: *European Conference on Computer Vision*.

[18] HE, K., ZHANG, X., REN, S. & SUN, J. (2016b) Identity mappings in deep residual networks. In: *European Conference on Computer Vision*.

[19] HE, K., ZHANG, X., REN, S. & SUN, J. (2016c) Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

[20] HINTON, G., OSINDERO, S. & TEH, T. (2006) A fast learning algorithm for deep belief nets. *Neural Comput.* **18**(7), 1527–1554.

[21] HINTON, G., SRIVASTAVA, N., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R. (2012) Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. arXiv preprint arXiv:1207.0580.

[22] HUANG, G., LIU, Z., VAN DER MAATEN, L. & WEINBERGER, K. (2017) Densely connected convolutional networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*.

[23] HUANG, G., SUN, Y., LIU, Z., SEDRA, D. & WEINBERGER, K. (2016) Deep networks with stochastic depth. In: *European Conference on Computer Vision*.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

[24] KINGMA, D. & BA, J. (2014) Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.
[25] KRIZHEVSKY, A. (2009) Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/cifar.html
[26] KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.
[27] KURAKIN, A., GOODFELLOW, I. & BENGIO, S. (2017) Adversarial machine learning at scale. In: *International Conference on Learning Representations*.
[28] LECUN, Y. (1998) The MNIST Database of Handwritten Digits.
[29] LECUN, Y., BENGIO, Y. & HINTON, G. (2015) Deep learning. *Nature* **521**, 436–444.
[30] LECUYER, M., ATLIDAKIS, V., GEAMBASU, R., HSU, D. & JANA, S. (2019) Certified robustness to adversarial examples with differential privacy. In: *IEEE Symposium on Security and Privacy (SP)*.
[31] LI, Z. & SHI, Z. (2017) Deep Residual Learning and PDEs on Manifold. arXiv preprint arXiv:1708.05115.
[32] LIU, Y., CHEN, X., LIU, C. & SONG, D. (2016) Delving into Transferable Adversarial Examples and Black-Box Attacks. arXiv preprint arXiv:1611.02770.
[33] MADRY, 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.
[34] MOOSAVI-DEZFOOLI, S.-M., FAWZI, A., FAWZI, O. & FROSSARD, P. (2017) Universal adversarial perturbations. In: *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.
[35] MUJA, M. & LOWE, D. G. (2014) Scalable nearest neighbor algorithms for high dimensional data. *IEEE Pattern Anal. Mach. Intell. (PAMI)* **36**, 2227–2240.
[36] NAIR, V. & HINTON, G. (2010) Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814.
[37] NETZER, Y., WANG, T., COATES, A., BISSACCO, A., WU, B. & NG, A. (2011) Reading digits in natural images with unsupervised features learning. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.
[38] OSHER, S. J., WANG, B., YIN, P., LUO, X., PHAM, M. & LIN, A. (2018) Laplacian Smoothing Gradient Descent. arXiv preprint arXiv:1806.06317.
[39] PAPERNOT, N., MCDANIEL, P., JHA, S., FREDRIKSON, M., CELIK, Z. B. & SWAMI, A. (2016a) The limitations of deep learning in adversarial settings. In: *IEEE European Symposium on Security and Privacy*, pp. 372–387.
[40] PAPERNOT, N., MCDANIEL, P., WU, X., JHA, S. & SWAMI, A. (2016b) Distillation as a defense to adversarial perturbations against deep neural networks. In: *IEEE European Symposium on Security and Privacy*.
[41] PAPERNOT, N., MCDANIEL, P. D. & GOODFELLOW, I. J. (2016c) Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. CoRR, abs/1605.07277. http://arxiv.org/abs/1605.07277.
[42] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L. & LERER, A. (2017) Automatic Differentiation in PyTorch. https://openreview.net/forum?id=BJJsrmfCZ.
[43] ROSS, A. & DOSHI-VELEZ, F. (2017) Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients. arXiv preprint arXiv:1711.09404.
[44] SAMANGOUEI, P., KABKAB, M. & CHELLAPPA, R. (2018) Defense-GAN: protecting classifiers against adversarial attacks using generative models. In: *International Conference on Learning Representations*. https://openreview.net/forum?id=BkJ3ibb0-.
[45] SHI, Z., WANG, B. & OSHER, S. (2018) Error Estimation of the Weighted Nonlocal Laplacian on Random Point Cloud. arXiv preprint arXiv:1809.08622.
[46] SIMONYAN, K. & ZISSERMAN, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.
[47] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I. & FERGUS, R. (2013) Intriguing Properties of Neural Networks. arXiv preprint arXiv:1312.6199.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press

[48] TANG, Y. (2013) Deep Learning Using Linear Support Vector Machines. arXiv:1306.0239.
[49] TRAMÈR, F., KURAKIN, A., PAPERNOT, N., GOODFELLOW, I., BONEH, D. & MCDANIEL, P. (2018) Ensemble adversarial training: attacks and defenses. In: *International Conference on Learning Representations*. https://openreview.net/forum?id=rkZvSe-RZ.
[50] Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitiagkas, I., Courville, A., Lopez-Paz, D. & Bengio, Y. (2018) Manifold Mixup: Better Representations by Interpolating Hidden States. arXiv preprint arXiv:1806.05236.
[51] WAN, L., ZEILER, M., ZHANG, S., LECUN, Y. & FERGUS, R. (2013) Regularization of neural networks using dropconnect. In: *International Conference on Machine Learning*, pp. 1058–1066.
[52] WANG, B., LIN, A. T., SHI, Z., ZHU, W., YIN, P., BERTOZZI, A. L. & OSHER, S. J. (2018a) Adversarial Defense via Data Dependent Activation Function and Total Variation Minimization. arXiv preprint arXiv:1809.08516.
[53] WANG, B., LUO, X., LI, Z., ZHU, W., SHI, Z. & OSHER, S. (2018b) Deep neural nets with interpolating function as output activation. In: *Advances in Neural Information Processing Systems*.
[54] WANG, B., YUAN, B., SHI, Z. & OSHER, S. (2019) ResNets ensemble via the Feynman-Kac formalism to improve natural and robust accuracies. In: *Advances in Neural Information Processing Systems*.
[55] ZAGORUYKO, S. & KOMODAKIS, N. (2016) Wide residual networks. In: *British Machine Vision Conference*.
[56] ZHANG, H., YU, Y., JIAO, J., XING, E., GHAOUI, L. & JORDAN, M. (2019) Theoretically Principled Trade-off between Robustness and Accuracy. arXiv preprint arXiv:1901.08573.
[57] ZHENG, S., SONG, Y., LEUNG, T. & GOODFELLOW, I. (2016) Improving the robustness of deep neural networks via stability training. In: *IEEE Conference on Computer Vision and Pattern Recognition*.

https://doi.org/10.1017/S0956792520000406 Published online by Cambridge University Press