Inverse Problems and Imaging doi:10.3934/ipi.2020046

ADVERSARIAL DEFENSE VIA THE DATA-DEPENDENT ACTIVATION, TOTAL VARIATION MINIMIZATION, AND ADVERSARIAL TRAINING

Bao Wang

Department of Mathematics, Scientific Computing and Imaging Institute
University of Utah, Salt Lake City, UT, 84112-0090, USA

Alex Lin and Penghang Yin

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA, 90095, USA

Wei Zhu

Department of Mathematics
Duke University
Durham, NC, 27708, USA

Andrea L. Bertozzi and Stanley J. Osher

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA, 90095, USA

Abstract. We improve the robustness of Deep Neural Net (DNN) to adversarial attacks by using an interpolating function as the output activation. This data-dependent activation remarkably improves both the generalization and robustness of DNN. In the CIFAR10 benchmark, we raise the robust accuracy of the adversarially trained ResNet20 from $\sim$ 46% to $\sim$ 69% under the state-of-the-art Iterative Fast Gradient Sign Method (IFGSM) based adversarial attack. When we combine this data-dependent activation with total variation minimization on adversarial images and training data augmentation, we achieve an improvement in robust accuracy by 38.9% for ResNet56 under the strongest IFGSM attack. Furthermore, We provide an intuitive explanation of our defense by analyzing the geometry of the feature space.

1. **Introduction.** The adversarial vulnerability [34] of Deep Neural Nets (DNNs) threaten their applicability in security critical tasks, e.g., autonomous cars [1], robotics [11], DNN-based malware detection systems [26, 10]. Since the pioneering work by Szegedy et al. [34], many advanced adversarial attacks have been devised to generate imperceptible perturbations to fool the DNN [9, 25, 7, 39, 14, 4, 8]. Not only are adversarial attacks successful in white-box attacks, i.e., when the adversary has access to the DNN parameters, but they are also successful in black-box attacks, i.e., without access to network parameters. Adversarial attacks are transferable in the sense that a perturbed image meant to be misclassified by one DNN also has

2020 *Mathematics Subject Classification.* Primary: 68T45; Secondary: 68T01.
Key words and phrases. Deep learning, Adversarial defense, Interpolation.
Please correspond to wangbaonj@gmail.com.

a significant chance to be misclassified by another DNN [28]. Due to this transferability, adversaries can attack DNN without knowing the network parameters (i.e. blackbox) [19, 6]. There even exist universal perturbations that can imperceptibly perturb any image and cause misclassification for any given network [22]. And recently, there has been much work on defending against these universal perturbations [2].

In this work, we defend against adversarial attacks by replacing the commonly used output activation of DNN with a manifold-interpolating function. Together with the Projected Gradient Descent (PGD) adversarial training [21], Total Variation Minimization (TVM), and training data augmentation, we show state-of-the-art results for adversarial defense on the CIFAR10 benchmark.

1.1. **Related work.** Defensive distillation was recently proposed to increase the robustness of DNN [27], and a related approach [35] cleverly modifies the training data to increase robustness against black-box attacks and adversarial attacks in general. To counter adversarial perturbations, Guo et al. [12], proposed to use image transformation, e.g., bit-depth reduction, JPEG compression, TVM, and image quilting. A similar idea of denoising the input was later explored in [23], where the authors divide the input into patches, denoise each patch, and then reconstruct the image. These input transformations are intended to be non-differentiable, thus making adversarial attacks more difficult, especially for gradient-based attacks. Another denoising approach is introduced by Liao et al. [18], where they proposed a High-level Representation Guided (HGD) denoiser – the idea is that while perturbations seem small in the original and adversarial images, these perturbations are amplified in higher representations. Transformation-based defenses have also been proposed by Xie et al. [40], and Luo et al. [20]. Song et al. [33], noticed that small adversarial perturbations shift the distribution of adversarial images far from the distribution of clean images. Therefore, they proposed to purify the adversarial images by PixelDefend. And Prakash et al. [29], also seek to examine image statistics in order to construct an adversarial defense – in their work, they introduce Pixel Deflection where they force images to match statistics of natural images. Lee et al. [17], have also used the distribution of images to detect adversarial examples. Adversarial training is another family of defense methods to enhance the stability of DNN [9, 21, 24]. In particular, the PGD adversarially trained DNN achieves state-of-the-art resistance to the available attacks [21]. GANs are also employed for adversarial defense [31]. In [3], the authors proposed an approximated gradient to attack the defenses that are based on the obfuscated gradient.

Instead of using the softmax function as DNN's output activation, Wang et al. [36, 38], utilized a class of non-parametric interpolating functions. This is a combination of both deep and manifold learning which causes the DNN to utilize the geometric information of the training data sufficiently. The authors show a significant amount of generalization accuracy improvement, and the results are more stable when one only has a limited amount of training data. Recently, Wang et al. [37] modeled ResNet as a transport equation, and they proposed an Feynman-Kac formalism principled adversarial robust DNN.

1.2. **Organization.** We organize this paper as follows: In section 2, we overview the DNN with a graph Laplacian-based high dimensional interpolating activation function. In section 3, we present a few adversarial attacks that will be used as benchmarks for this work. In section 4, we elaborate on adversarial defense via

interpolating activation together with TVM. In section 5, we further study the robustness of PGD adversarially trained DNN with interpolating activation. This paper ends up with concluding remarks.

2. DNN with data-dependent activation.

In this section, we summarize the architecture, training, and testing procedures of the DNN with the data-dependent activation [36]. For the standard DNN with softmax activation, the training and testing are shown in Fig. 1 (a) and (b), respectively. In the kth iteration of training, 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 the output into the softmax activation to obtain the predictions $\tilde{\mathbf{Y}}$, i.e.,

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

Then the loss is computed (e.g., cross entropy) between $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$: $\mathcal{L} \doteq \mathcal{L}^{\text{Linear}} = \text{Loss}(\mathbf{Y}, \tilde{\mathbf{Y}})$.

Backpropagation: Update weights (Θ^{k-1}, $\mathbf{W}^{k-1}$) by gradient descent with learning rate γ

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

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

Once the model is optimized, with optimal parameters being $(\Theta, \mathbf{W})$, the predicted labels for testing data $\mathbf{X}$ are

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

Wang et al [36] proposed to replace the data-agnostic softmax by an interpolating function defined below.

2.1. Manifold interpolation - a harmonic extension approach.

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ be a set of points on a high dimensional manifold $\mathcal{M} \subset \mathbb{R}^d$ and $\mathbf{X}^{\text{te}} = \{\mathbf{x}_1^{\text{te}}, \mathbf{x}_2^{\text{te}}, \cdots, \mathbf{x}_m^{\text{te}}\}$ ("te" for template) be a subset of $\mathbf{X}$ which are labeled with label function $g(\mathbf{x})$ [1]. We want to interpolate a function u that is defined on $\mathcal{M}$ and can be used to label the entire dataset $\mathbf{X}$. The harmonic extension is a natural approach to find such an interpolating function, which is defined by minimizing the following Dirichlet energy functional

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

with the boundary condition

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

[1] The minimum requirement is that the template data needs to cover all classes. In [36], we show that for an image classification task with m number of different classes, the size of the template needs to be at least $m \log m$. In practice, the size of the template set will not affect the performance much as long as the template set size is more than 1K for CIFAR10 and CIFAR100.

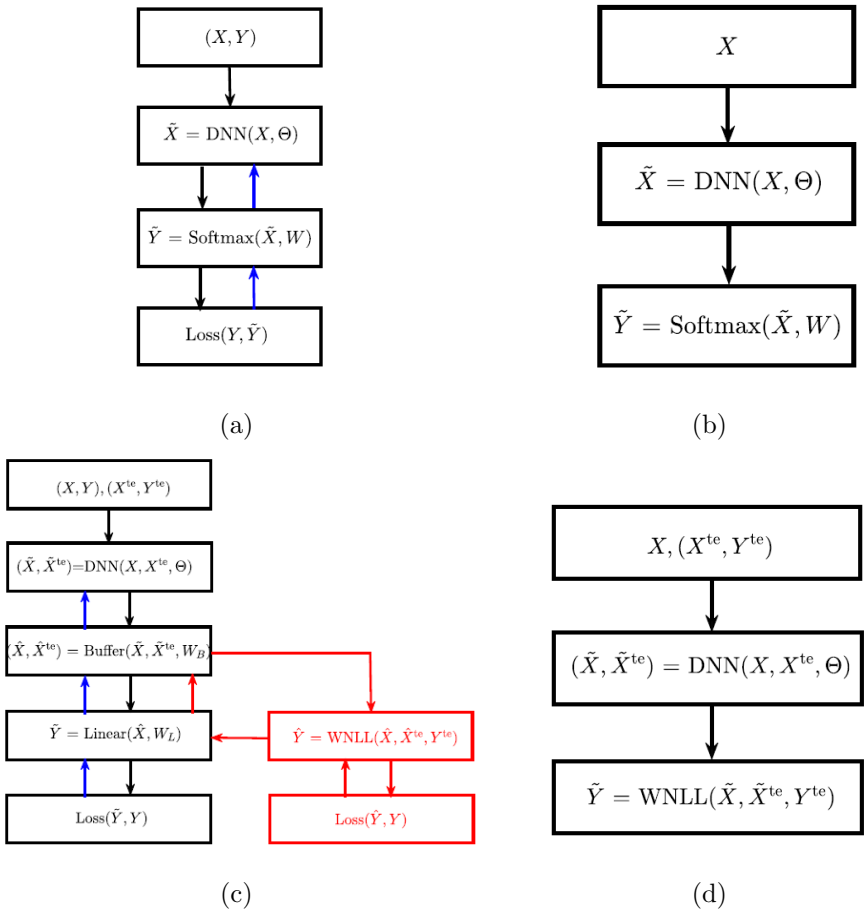

FIGURE 1. Training and testing procedures of the DNN with softmax and WNLL functions as the output activation layer. (a) and (b) show the training and testing steps for the standard DNN, respectively; (c) and (d) illustrate the training and testing procedure of the WNLL activated DNN, respectively.

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. The Euler-Lagrange equation for Eq. (1) is

$$\begin{cases} \sum_{\mathbf{y}\in\mathbf{X}} \left(w(\mathbf{x}, \mathbf{y}) + w(\mathbf{y}, \mathbf{x})\right) \left(u(\mathbf{x}) - u(\mathbf{y})\right) = 0 & \mathbf{x} \in \mathbf{X}/\mathbf{X}^{\mathrm{te}} \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \mathbf{X}^{\mathrm{te}}. \end{cases} \tag{2}$$

By solving the linear system Eq. (2), we obtain labels $u(\mathbf{x})$ for the unlabeled data $\mathbf{x} \in \mathbf{X}/\mathbf{X}^{\mathrm{te}}$. This interpolation becomes invalid when the labeled data is tiny, i.e., $|\mathbf{X}^{\mathrm{te}}| \ll |\mathbf{X}/\mathbf{X}^{\mathrm{te}}|$. To resolve this issue, the weights of the labeled data is increased in the Euler-Lagrange equation, which gives

$$
\begin{cases}
\sum_{\mathbf{y}\in\mathbf{X}} \left(w(\mathbf{x},\mathbf{y}) + w(\mathbf{y},\mathbf{x})\right)\left(u(\mathbf{x}) - u(\mathbf{y})\right) + \\
\left(\frac{|\mathbf{X}|}{|\mathbf{X}^{\text{te}}|} - 1\right) \sum_{\mathbf{y}\in\mathbf{X}^{\text{te}}} w(\mathbf{y},\mathbf{x})\left(u(\mathbf{x}) - u(\mathbf{y})\right) = 0 & \mathbf{x} \in \mathbf{X}/\mathbf{X}^{\text{te}} \\
u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \mathbf{X}^{\text{te}}.
\end{cases} \tag{3}
$$

The solution to Eq. (3) is named weighted nonlocal Laplacian (WNLL), denoted as $\text{WNLL}(\mathbf{X}, \mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$. Shi et al. [32], showed that WNLL converges to the solution of the high dimensional Laplace-Beltrami equation. For classification, $g(\mathbf{x})$ is the one-hot label for the example $\mathbf{x}$.

2.2. **Training and testing of the DNN with data-dependent activation Function.** For a standard DNN, we denote the WNLL activated one as DNN-WNLL, e.g., the WNLL activated ResNet20 is denoted as ResNet20-WNLL. In both training and testing of the DNN-WNLL, we need to reserve a small portion of data/label pairs, denoted as $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$, to interpolate the label for new data. We name the reserved data $(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$ as the template. Directly replacing softmax by WNLL has difficulties in back propagation, namely the true gradients $\frac{\partial \mathcal{L}}{\partial \Theta}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_B}$ (here $\mathcal{L} \doteq \mathcal{L}^{WNLL} = \text{Loss}(\hat{Y}, Y)$, as shown in Fig. 1(c)) are difficult to compute since WNLL defines an implicit function. Instead, to train the DNN-WNLL, a proxy via an auxiliary DNN (Fig. 1(c)) is employed. On top of the original DNN, we add a buffer block (a fully connected layer followed by a ReLU), and followed by two parallel branches, WNLL and linear (fully connected) layers. The auxiliary DNN can be trained by alternating between training the DNN with linear and WNLL activation functions, respectively. When training DNN with WNLL activation function, the training loss of the WNLL activation is backpropped via a straight-through gradient estimator [3, 5], e.g., in the kth iteration, we use the following approximated gradient descent (Eq. (4)) to update $\mathbf{W}_B$ only (when backpropagating the training loss $\mathcal{L}^{\text{WNLL}}$ we freeze the remaining part except for the buffer block, and the other parameters will be updated in training DNN with linear activation function),

$$
\begin{aligned}
\mathbf{W}_B^k &= \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}} \cdot \frac{\partial \hat{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B} \\
&\approx \mathbf{W}_B^{k-1} - \gamma \frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}} \cdot \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}} \cdot \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{W}_B},
\end{aligned} \tag{4}
$$

where $\frac{\partial \mathcal{L}^{\text{Linear}}}{\partial \tilde{\mathbf{Y}}}$ and $\frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}}$ are the gradients computed through two different activation functions. In the approximation of Eq. (4), we simply replace the value of $\mathcal{L}^{\text{Linear}}$ with that of $\mathcal{L}^{\text{WNLL}}$, which allows us to compute the value of $\frac{\partial \mathcal{L}^{\text{WNLL}}}{\partial \hat{\mathbf{Y}}}$ by leveraging the computational graph of DNN with linear activation. The detailed training procedure can be found in [36].

At test time, we remove the linear activation from the neural net and use the DNN and buffer blocks together with WNLL to classify new data (Fig. 1 (d)). Here for simplicity, we merge the buffer block to the DNN block. For a given set of testing data $\mathbf{X}$, and the labeled template $\{(\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})\}$, the predicted labels for $\mathbf{X}$ is given by

$$
\tilde{\mathbf{Y}} = \text{WNLL}(\text{DNN}(\mathbf{X}, \mathbf{X}^{\text{te}}, \Theta), \mathbf{Y}^{\text{te}}).
$$

2.3. **Computational complexity of DNN with data-dependent activation.** Using WNLL activation will lead to some extra computational overhead, which comes from the nearest neighbor searching and solving a system of linear equations. We following the same training procedure as that used in [36] to train ResNet20. In Table 1, we list the training and test time on a single Titan Xp GPU for ResNet20 on CIFAR10.

TABLE 1. Running time and GPU memory for ResNet20 with two different activation functions.

	Training time	Testing time	Memory
ResNet20	3925.6 (s)	0.657 (s)	1007 (MB)
ResNet20-WNLL	7378.4 (s)	14.09 (s)	1563 (MB)

3. **Adversarial attacks.** We consider three benchmark attacks: the Fast Gradient Sign Method (FGSM) [9], Iterative FGSM (IFGSM) [16], and Carlini-Wagner's L_2 (CW-L2) [7] attack. We denote the classifier defined by the DNN as $\tilde{y} = f(\theta, \mathbf{x})$ for a given instance $(\mathbf{x}, y)$. FGSM searchs the adversarial image $\mathbf{x}'$ with a bounded perturbation by maximizing the loss $\mathcal{L}(\mathbf{x}', y) \doteq \mathcal{L}(f(\theta, \mathbf{x}'), y)$, subject to the l_∞ perturbation constraint $||\mathbf{x}' - \mathbf{x}||_\infty \leq \epsilon$ with ϵ being the attack strength. We can approximately solve this constrained optimization problem by using the first order approximation of the loss 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 approximation, the optimal adversarial image is

$$\mathbf{x}' = \mathbf{x} + \epsilon \operatorname{sign} \cdot (\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, y)) . \tag{5}$$

IFGSM iterates FGSM to generate the enhanced attack, i.e.,

$$\mathbf{x}^{(m)} = \operatorname{Clip}_{\mathbf{x},\epsilon} \left\{ \mathbf{x}^{(m-1)} + \alpha \cdot \operatorname{sign} \left(\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{(m-1)}, y) \right) \right\}, \tag{6}$$

where $m = 1, \cdots, M$, $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}' = \mathbf{x}^{(M)}$, with M be the number of iterations. α is the step size used in each iteration, and $\operatorname{Clip}_{\mathbf{x},\epsilon}$ clips the update to be within an ϵ-ball centered at $\mathbf{x}$ in l_∞-norm.

Moreover, we consider the attack due to Carlini and Wagner. For a given image-label pair $(\mathbf{x}, y)$, and $\forall t \neq y$, CW-L2 searches the adversarial image that will be classified to class t by solving the optimization problem

$$\min_{\delta} ||\delta||_2^2, \tag{7}$$

subject to

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

where δ is the adversarial perturbation (for simplicity, we ignore the dependence of θ in f).

The equality constraint in Eq. (7) is hard to handle, so Carlini et al. considered the surrogate

$$g(\mathbf{x}) = \max \left(\max_{i \neq t} (Z(\mathbf{x})_i) - Z(\mathbf{x})_t, 0 \right), \tag{8}$$

where $Z(\mathbf{x})$ is the logit vector for an input $\mathbf{x}$, i.e., output of DNN before the output layer. $Z(\mathbf{x})_i$ is the logit value corresponding to class i. It is easy to see that

$f(\mathbf{x}+\delta)=t$ is equivalent to $g(\mathbf{x}+\delta)\leq 0$. Therefore, the problem in Eq. (7) can be reformulated as

$$\min_{\delta} ||\delta||_2^2 + c\cdot g(\mathbf{x}+\delta), \tag{9}$$

subject to

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

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

By letting $\delta=\frac{1}{2}(\tanh(\mathbf{w})+1)-\mathbf{x}$, Eq. (9) can be written as an unconstrained optimization problem. Moreover, Carlini et al. introduce the confidence parameter κ into the above formulation. Above all, the CW-L2 attack seeks the adversarial image by solving the following problem

$$\begin{aligned}\min_{\mathbf{w}} ||\frac{1}{2}(\tanh(\mathbf{w})+1)-\mathbf{x}||_2^2 + c\cdot \\ \max\left\{-\kappa, \max_{i\neq t}(Z(\frac{1}{2}(\tanh(\mathbf{w}))+1)_i) - Z(\frac{1}{2}(\tanh(\mathbf{w}))+1)_t\right\}.\end{aligned} \tag{10}$$

The Adam optimizer [15] can solve this unconstrained optimization problem efficiently. All three attacks clip the values of the adversarial image $\mathbf{x}'$ to between 0 and 1.

3.1. **Attack the DNN with WNLL activation.** For a given mini-batch of testing images $(\mathbf{X},\mathbf{Y})$ and template $(\mathbf{X}^{\text{te}},\mathbf{Y}^{\text{te}})$, we denote the DNN-WNLL as $\hat{\mathbf{Y}}=\text{WNLL}(Z(\{\mathbf{X},\mathbf{X}^{\text{te}}\}),\mathbf{Y}^{\text{te}})$, where $Z(\{\mathbf{X},\mathbf{X}^{\text{te}}\})$ is the composition of the DNN and buffer blocks as shown in Fig. 1(c). 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 attacks for DNN-WNLL are formulated below.

- **FGSM**

$$\mathbf{X}'=\mathbf{X}+\epsilon\cdot\text{sign}\left(\nabla_{\mathbf{X}}\tilde{\mathcal{L}}(\mathbf{X},\mathbf{Y},\mathbf{X}^{\text{te}},\mathbf{Y}^{\text{te}})\right). \tag{11}$$

- **IFGSM**

$$\mathbf{X}^{(m)}=\text{Clip}_{\mathbf{X},\epsilon}[\mathbf{X}^{(m-1)}+\alpha\cdot\text{sign}\left(\nabla_{\mathbf{X}}\tilde{\mathcal{L}}(\mathbf{X}^{(m-1)},\mathbf{Y},\mathbf{X}^{\text{te}},\mathbf{Y}^{\text{te}})\right)], \tag{12}$$

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

- **CW-L2**

$$\begin{aligned}\min_{\mathbf{W}} ||\frac{1}{2}(\tanh(\mathbf{W})+1)-\mathbf{X}||_2^2 + c\cdot\max[-\kappa, \\ \max_{\mathbf{i}\neq\mathbf{t}}(Z(\frac{1}{2}(\tanh(\mathbf{W}))+1)_{\mathbf{i}}) - Z(\frac{1}{2}(\tanh(\mathbf{W}))+1)_{\mathbf{t}}],\end{aligned} \tag{13}$$

  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 Fig. 1 (c), we approximate $\nabla_{\mathbf{X}}\tilde{\mathcal{L}}$ in the following way

$$\nabla_{\mathbf{X}}\tilde{\mathcal{L}}=\frac{\partial\mathcal{L}^{\text{WNLL}}}{\partial\hat{\mathbf{Y}}}\cdot\frac{\partial\hat{\mathbf{Y}}}{\partial\hat{\mathbf{X}}}\cdot\frac{\partial\hat{\mathbf{X}}}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\mathbf{X}}\approx\frac{\partial\mathcal{L}^{\text{Linear}}}{\partial\tilde{\mathbf{Y}}}\cdot\frac{\partial\tilde{\mathbf{Y}}}{\partial\hat{\mathbf{X}}}\cdot\frac{\partial\hat{\mathbf{X}}}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\mathbf{X}}, \tag{14}$$

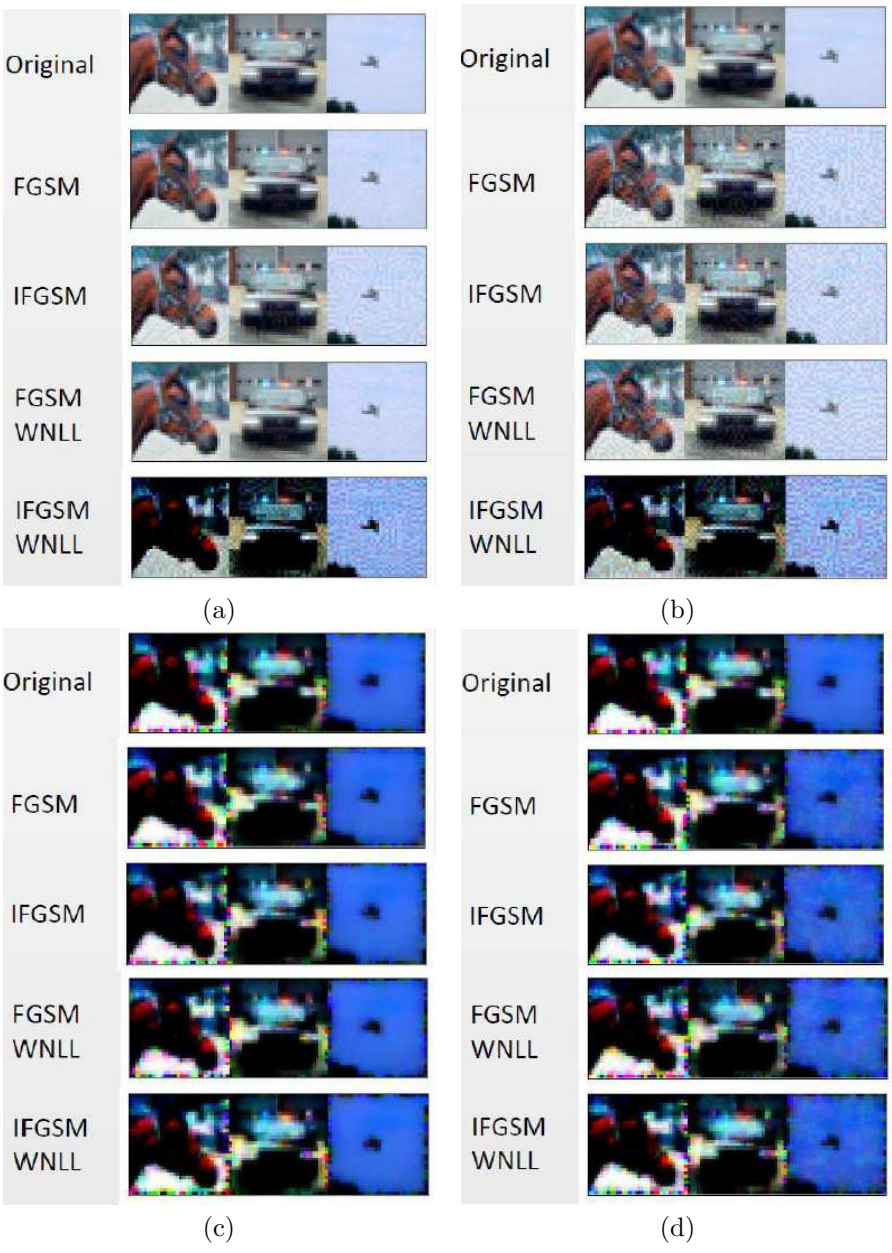

FIGURE 2. Samples from CIFAR10. Panel (a): from the top to the last rows show the original, adversarial images by attacking ResNet56 with FGSM and IFGSM ($\epsilon = 0.02$); and by attacking ResNet56-WNLL. Panel (b) corresponding to those in panel (a) with $\epsilon = 0.08$. Charts (c) and (d) corresponding to the TV minimized images in (a) and (b), respectively.

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

Based on our numerical experiments, the batch size of $\mathbf{X}$ has minimal 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.

4. **Defense by interpolating function, TVM, and training data augmentation.** To defend against adversarial attacks, we first combine the data-dependent activation with input transformation and with training data augmentation. We train ResNet56 [13] and ResNet56-WNLL, respectively, on the original training data, the TV minimized training data, and a combination of the previous two. Moreover, in testing, we apply the TVM [30] used by [12], with the same setting, to transform the adversarial images to boost classification performance. The basic idea of TVM is to reconstruct the simplest image $\mathbf{z}$ from the sub-sampled image, $X \odot \mathbf{x}$ with X the mask filled by a Bernoulli binary random variable, by solving

$$\min_{\mathbf{z}} ||(1 - X) \odot (\mathbf{z} - \mathbf{x})||_2 + \lambda_{TV} \cdot TV_2(\mathbf{z}),$$

where $\lambda_{TV} > 0$ is the regularization constant.

We apply the three attack schemes mentioned above to attack ResNet56 and ResNet56-WNLL. For IFGSM, we run 10 iterations of Eqs. (6) and (12) with $\epsilon = 0.1$ to attack the DNN with two different output activations, respectively. For the CW-L2 attack (Eqs. (10, 13)), in both scenarios we set the parameters $c = 10$ and $\kappa = 0$, and run 10 iterations of the Adam optimizer with learning rate 0.01. Figure 2 depicts three randomly selected images (horse, automobile, airplane) from the CIFAR10 dataset, as well as the perturbed images from applying different attacks on ResNet56 and ResNet56-WNLL, and the TV minimized ones. All attacks successfully fool the classifiers to classify any of them correctly. Figure 2 (a) shows that the perturbations resulted from FGSM attack with $\epsilon = 0.02$ is almost imperceptible. However, both FGSM and IFGSM attacks are powerful in fooling DNNs. Figure 2 (b) shows the corresponding images of (a) with a stronger attack, $\epsilon = 0.08$. With a larger ϵ, the adversarial images become more noisy. The TV minimized images of Fig. 2 (a) and (b) are shown in Fig. 2 (c) and (d), respectively. TVM removes a significant amount of information from the original and the adversarial images. Meanwhile, it also makes it harder for humans to classify them.

4.1. **Numerical results.** In this subsection, we first discuss the transferability of adversarial examples generated by attacking DNNs with softmax and WNLL activation functions. The transferability of adversarial examples is often used for black-box adversarial attacks. Adversarial examples of a robust DNN typically have good transferability. Next, we numerically verify the efficacy of adversarial defense by leveraging DNN with the WNLL activation function and TVM. Finally, we explain the adversarial robustness by considering the deep learning features learned by DNN with different activation functions.

4.1.1. *Transferability of the adversarial images.* Consider the transferability of adversarial examples crafted by using the above adversarial attacks to attack ResNet56 with either softmax or WNLL activation. We utilize the training strategy used in [36] to train the DNNs. To test the transferability, we classify the adversarial images by using ResNet56 with the opponent activation (the opponent activation of WNLL is softmax, and vice versa). We list the mutual classification accuracy (the accuracy of DNN with one specific activation to classify adversarial images crafted by attacking DNN with the other activation) on adversarial images resulting from

using FGSM or IFGSM in Table. 2. The adversarial images crafted by attacking ResNet56 with two types of activation functions are both transferable, as the mutual classification accuracy on adversarial images ($\epsilon \neq 0$) is significantly lower than testing on the clean images ($\epsilon = 0$). For both FGSM and IFGSM, the stronger attack (in the sense of bigger ϵ) is adapted to the opponent activation function, as the mutual classification accuracy decreases dramatically as ϵ increases. IFGSM not only fools its underlying model completely, but also significantly decreases the accuracy of the opponent DNN. The mutual classification results for the CW-L2 attack is shown in Table. 3, where Exp-I denotes classifying adversarial images resulted from attacking ResNet56-WNLL by ResNet56, and Exp-II denotes the opposite. Training data augmentation can defend CW-L2 attack very effectively.

TABLE 2. Mutual classification accuracy on the adversarial images crafted by using FGSM and IFGSM to attack ResNet56 and ResNet56-WNLL. (Unit: %)

Attack	Training data	$\epsilon = 0$	$\epsilon = 0.02$	$\epsilon = 0.04$	$\epsilon = 0.06$	$\epsilon = 0.08$	$\epsilon = 0.1$
Accuracy of ResNet56 on adversarial images crafted by attacking ResNet56-WNLL							
FGSM	Original data	93.0	69.8	56.9	44.6	34.6	28.3
FGSM	TVM data	88.3	51.5	37.9	30.1	24.7	20.9
FGSM	Original + TVM	93.1	78.5	70.9	64.6	59.8	55.8
IFGSM	Original data	93.0	5.22	5.73	6.73	7.55	8.55
IFGSM	TVM data	88.3	7.00	6.82	8.30	9.28	10.7
IFGSM	Original + TVM	93.1	27.3	28.6	29.5	29.1	29.4
Accuracy of ResNet56-WNLL on adversarial images crafted by attacking ResNet56							
FGSM	Original data	94.5	65.2	49.0	39.3	32.8	28.3
FGSM	TVM data	90.6	45.9	30.9	22.2	16.9	13.8
FGSM	Original + TVM data	94.7	78.3	68.2	61.1	56.5	52.5
IFGSM	Original data	94.5	3.37	3.71	3.54	4.69	6.41
IFGSM	TVM data	90.6	7.88	7.51	7.58	8.07	9.67
IFGSM	Original + TVM data	94.7	34.3	33.4	33.1	34.6	35.8

TABLE 3. Mutual classification accuracy on the adversarial images crafted by using CW-L2 to attack ResNet56 and ResNet56-WNLL. (Unit: %)

Training data	Original data	TVM data	Original + TVM data
Exp-I	52.1	43.2	80.0
Exp-II	59.7	41.1	80.1

TABLE 4. Testing accuracy on the adversarial/TVM adversarial CIFAR10 dataset. The testing accuracy with no defense is in red italic; and the results with all three defenses are in boldface. (Unit: %)

Training data	Original data	TVM data	Original + TVM data
ResNet56	*4.94*/32.2	11.8/54.0	15.1/52.4
ResNet56-WNLL	18.3/35.2	15.0/53.9	28/**54.5**

TABLE 5. Testing accuracy on the adversarial/TVM adversarial CIFAR10 dataset. The testing accuracy with no defense is in red italic; and the results with all three defenses are in boldface. (Unit: %)

Attack	Training data	$\epsilon = 0$	$\epsilon = 0.02$	$\epsilon = 0.04$	$\epsilon = 0.06$	$\epsilon = 0.08$	$\epsilon = 0.1$
			ResNet56				
FGSM	Original data	93.0	*36.9*/19.4	*29.6*/18.9	*26.1*/18.4	*23.1*/17.9	*20.5*/17.1
FGSM	TVM data	88.3	27.4/50.4	19.1/47.2	16.6/43.7	15.0/38.9	13.7/35.0
FGSM	Original + TVM	93.1	48.6/51.1	42.0/47.6	39.1/44.2	37.1/41.8	35.6/39.1
IFGSM	Original data	93.0	*0*/16.6	*0*/16.1	*0.02*/15.9	*0.1*/15.5	*0.25*/16.1
IFGSM	TVM data	88.3	0.01/43.4	0/42.5	0.02/42.4	0.18/42.7	0.49/42.4
IFGSM	Original + TVM	93.1	0.1/38.4	0.09/37.9	0.36/37.9	0.84/37.6	1.04/37.9
			ResNet56-WNLL				
FGSM	Original data	94.5	58.5/26.0	50.1/25.4	42.3/25.5	35.7/24.9	29.2/22.9
FGSM	TVM data	90.6	31.5/52.6	24.5/49.6	20.2/45.3	17.3/41.6	14.4/37.5
FGSM	Original + TVM	94.7	**60.5**/ 55.4	**56.7**/52.0	**55.3**/48.6	**53.2**/45.9	**50.1**/43.7
IFGSM	Original data	94.5	0.49/16.7	0.14/17.3	0.3/16.9	1.01/16.6	0.94/16.5
IFGSM	TVM data	90.6	0.61/37.3	0.43/36.3	0.63/35.9	0.87/35.9	1.19/35.5
IFGSM	Original + TVM	94.7	0.19/**38.5**	0.3/**39.4**	0.63/**40.1**	1.26/**38.9**	1.72/**39.1**

4.1.2. *Adversarial defense.* Figure 3 plots the results of adversarial defense by combining the WNLL activation, TVM, and training data augmentation. Panels (a) and (b) show the testing accuracy of ResNet56 with and without defense on CIFAR10 data for FGSM and IFGSM, respectively. It is seen that as ϵ increases, the testing accuracy decreases rapidly. FGSM is a relatively weak attack, and the accuracy remains above 20.5% even with the most potent attack ($\epsilon = 0.1$). Meanwhile, the defense raises the accuracy to 43.7%. Figure 3 (b) shows that IFGSM fools ResNet56 near completely even with $\epsilon = 0.02$. The defense maintains the accuracy above 38.5%, 54.5% under the CW-L2 and IFGSM attacks, respectively (see Tables. 4 and 5). Compared to the state-of-the-art defensive methods on CIFAR10, PixelDefend, our approach is much simpler and faster. Without adversarial training, we have shown our defense is more robust to FGSM and IFGSM attacks under the strongest attack than PixelDefend [33]. Moreover, our defense strategy is additive to adversarial training and many other defenses including PixelDefend.

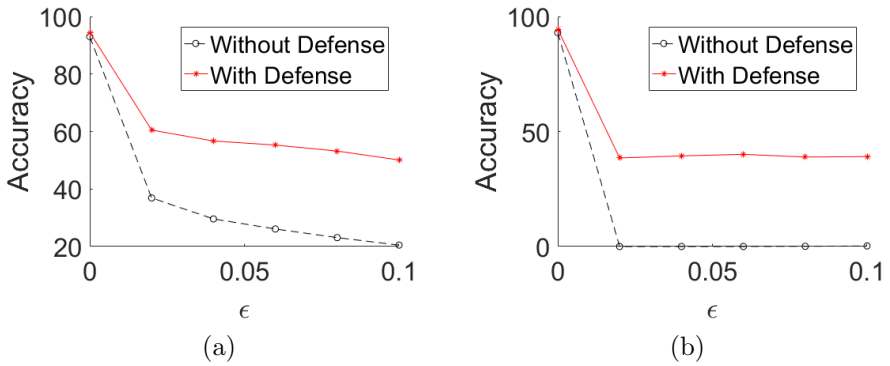

FIGURE 3. ϵ v.s. accuracy without defense, and defending by WNLL activation, TVM and augmented training. (a) and (b) plot results for FGSM and IFGSM attack, respectively.

To analyze the contribution from each component of the defensive strategy, we separate the three parts and list the testing accuracy in Tables. 4 and 5. Performing TVM on the adversarial images cannot defend FGSM attacks except when the training data contains the TV minimized images. For instance, when we attack the model by FGSM with $\epsilon = 0.02$, the accuracy on the adversarial images for ResNet56 and ResNet56-WNLL are 36.9% and 58.5%, respectively, provided the models are trained on the original training data. The accuracy reduces to 19.4% and 26.0% when testing on the TV minimized adversarial images. For ResNet56, the accuracy raises to 50.4% and 51.1% when the model is trained on the TVM and augmented data, respectively. For ResNet-WNLL, the accuracy increases to 52.6% and 55.4%, respectively. The WNLL activation improves testing accuracy of adversarial attacks significantly and persistently. Augmented training can also improve the stability consistently.

IFGSM fools the ResNet56-WNLL near completely, as the accuracy is always less than or close to 1%. These results verify the efficacy of using the approximated gradient, i.e., Eq. (14), in attacking the neural nets.

4.1.3. *Analysis of the geometry of features.* We consider features' geometry of the original and adversarial images. We randomly select 1000 training and 100 testing images from the airplane and automobile classes, respectively. We apply two visualization strategies for ResNet56: (1) Apply the principle component analysis (PCA) to reduce the 64D features from the layer before the softmax to 2D, and (2) we add a 2 by 2 fully connected (FC) layer before the softmax to learn 2D features. We verify that the newly added layer does not change the performance of ResNet56, as shown in Fig. 4, and the training and testing performance remains essentially the same.

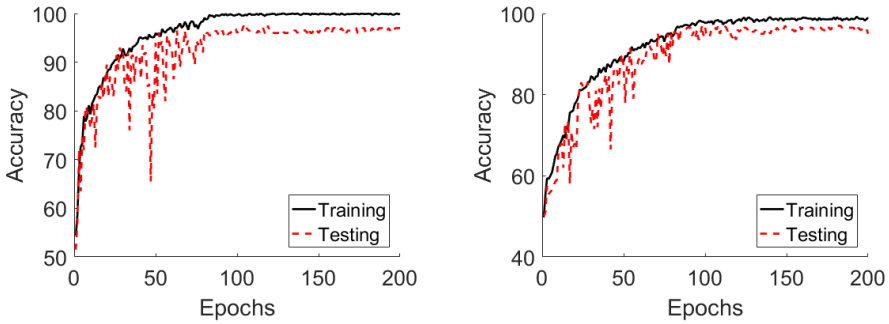

FIGURE 4. Epochs v.s. accuracy of ResNet56 on CIFAR10. (a): without the additional FC layer; (b): with the additional FC layer.

Figure 5 (a) and (b) show the 2D features generated by ResNet56 with the additional FC layer for the original and adversarial testing images, respectively, where we generate the adversarial images by using FGSM ($\epsilon = 0.02$). Before adversarial perturbation (Fig. 5 (a)), there is a line that can separate the two classes very well. The small perturbation mixes the features and there is no linear classifier that can easily separate these two classes (Fig. 5 (b)). The first two principle components (PCs) of the 64D features of the clean and adversarial images are shown in Fig. 5

Algorithm 1 PGD Adversarial Training of the DNN-WNLL

Input: Training set: (data, label) pairs $(\mathbf{X}, \mathbf{Y})$. N_1: the number of epochs in training DNN + Linear blocks. N_{IFGSM}: the number of iterations of IFGSM attack.

Output: An optimized DNN-WNLL, denoted as DNN_{WNLL}.

for iter $= 1, \ldots, N$ (where N is the number of alternating steps.) **do**

//PGD adversarial training of the left branch: DNN with linear activation.

Train DNN + Linear blocks, and denote the learned model as $\text{DNN}_{\text{Linear}}$.

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 IFGSM.

for iter$_1 = 1, \ldots, N_{\text{IFGSM}}$ **do**

Update the training image $\mathbf{X}_i = \mathbf{X}_i + \epsilon \cdot \text{sign}\left(\nabla_{\mathbf{X}_i} \mathcal{L}\right)$, where $\mathcal{L}$ is the loss of the prediction by DNN + Linear blocks w.r.t the ground truth labels $\mathbf{Y}_i$.

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., $(\mathbf{X}, \mathbf{Y}) \doteq (\mathbf{X}^{\text{tr}}, \mathbf{Y}^{\text{tr}}) \bigcup (\mathbf{X}^{\text{te}}, \mathbf{Y}^{\text{te}})$.

Partition the training data into M_2 mini-batches, i.e., $(\mathbf{X}^{\text{tr}}, \mathbf{Y}^{\text{tr}}) = \bigcup_{i=1}^{M_2} (\mathbf{X}_i^{\text{tr}}, \mathbf{Y}_i^{\text{tr}})$.

for epoch$_2 = 1, \ldots, N_2$ **do**

for $i = 1, \ldots, M_2$ **do**

//Attack the input training images by IFGSM.

for iter$_1 = 1, \ldots, N_{\text{IFGSM}}$ **do**

Update the training image $\mathbf{X}_i^{\text{tr}} = \mathbf{X}_i^{\text{tr}} + \epsilon \cdot \text{sign}\left(\nabla_{\mathbf{X}_i^{\text{tr}}} \tilde{\mathcal{L}}\right)$, where $\tilde{\mathcal{L}}$ is the loss of the prediction by DNN with WNLL activation w.r.t the ground truth labels $\mathbf{Y}_i^{\text{tr}}$.

Backpropagate the classification error of the adversarial images.

(c) and (d), respectively. Again, the PCs are well separated for clean images, while adversarial images causes overlap.

The bottom charts of Fig. 5 depict the first two PCs of the 64D features output from the layer before the WNLL. The distributions of the unperturbed training and testing data are the same, as illustrated in panels (e) and (f). The new features are better separated which indicates that DNN-WNLL are more accurate and more robust to small random perturbation. Panels (g) and (h) plot the features of the adversarial and TV minimized adversarial images in the test set. The adversarial attacks make the features move towards each other and TVM helps to eliminate the outliers. Based on our computation, the interpolating function on features shown in panels (g) and (h) are significantly more accurate than the softmax classifier as shown in panel (d). The fact that the adversarial perturbations change the features' distribution was also noticed in [33], and [18].

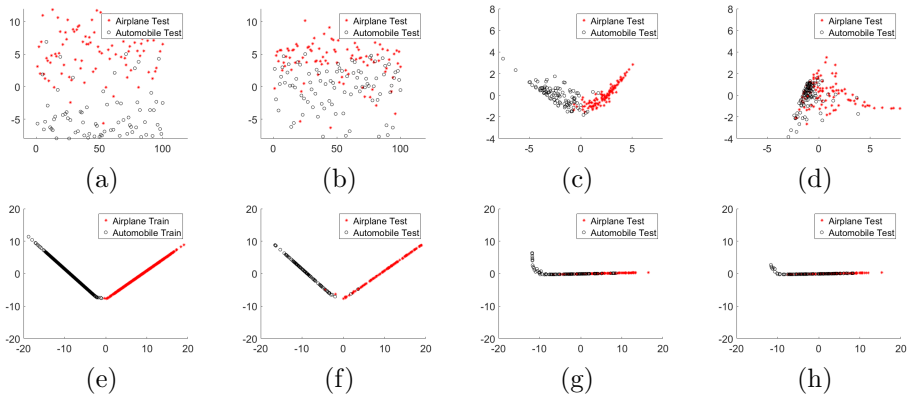

FIGURE 5. Visualization of the features learned by DNN with softmax ((a), (b), (c), (d)) and WNLL ((e), (f), (g), (h)) activation functions. (a) and (b) plot the 2D features of the original and adversarial testing images; (c) and (d) are the first two principle components of the 64D features for the original and adversarial testing images, respectively. Charts (e), (f) plot the first two components of the training and testing features learned by ResNet56-WNLL; (g) and (h) show the two principle components of the adversarial and TV minimized adversarial images for the test set.

5. **PGD adversarial training with data-dependent activation function.** Image transformation based adversarial defense has been broken recently by circumventing the obfuscated gradient [3]. To train a DNN that is most resistant to adversarial attacks, Madry et al. [21], incorporate the adversarial perturbation into the empirical risk function $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathcal{L}(\mathbf{x},y,\theta)\right]$, where $\mathcal{D}$ is the collection of the pairs of training images and labels, and θ represents the parameters of the neural nets. The idea of PGD adversarial training is that instead of feeding samples from $\mathcal{D}$ directly into the loss $\mathcal{L}$, we use the adversary to perturb the input first, and then we end up with the following saddle point problem

$$\min_{\theta}\rho(\theta)=\min_{\theta}\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max_{\delta\in S}\mathcal{L}(\theta,\mathbf{x}+\delta,y)\right], \tag{15}$$

where δ is the adversarial perturbation. To make the problem (Eq. (15)) solvable, the inner maximization problem is relaxed to a strong adversarial attack, say IFGSM. It is argued in [3], that PGD adversarial training achieves the best resistance to adversarial attacks for CIFAR10 classification. We extend the PGD adversarial training to DNN-WNLL by applying the approximated gradient, Eq. (14), to approximately resolve the interior maximization problem. We summarize the PGD adversarial training of DNN-WNLL in Algorithm 1.

5.1. **Numerical results.** We consider PGD adversarial training, respectively, for the ResNet20 and ResNet20-WNLL. Again, we train the ResNet20 with two types of activation, where we follow the strategy used in [36], and where all the hyperparameters in Algorithm 1 are referred. To approximate $\max_{\delta\in S}\mathcal{L}(\theta,\mathbf{x}+\delta,y)$, we apply the IFGSM attack with $\alpha=8/255$ in Eqs. (6, 12).

First, we fixed the attack strength $\epsilon = 1/255$ and vary the number of IFGSM iterations. As shown in Fig. 6 (a), the accuracy of ResNet20 with both activations decreases as the number of iteration increases. The vanilla ResNet20's accuracy decays much faster than the ResNet20-WNLL. The difference is $\sim 23\%$ when 10 iterations of IFGSM is applied. Second, we fixed the IFGSM iteration to be 10 and vary ϵ from 0 to 8/255 with step size 1/255. As shown in Fig. 6 (b), for different nonzero attack strengths, PGD adversarial training of the ResNet20-WNLL has $\sim 23\%$ higher accuracy than the vanilla one consistently.

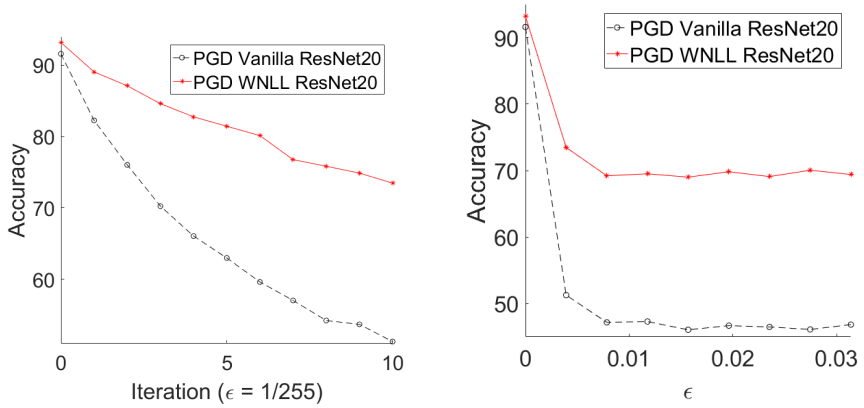

FIGURE 6. (a): #IFGSM iterations v.s. accuracy for the ResNet20 and the ResNet20-WNLL trained with PGD adversarial training. (b):ϵ v.s. accuracy for the ResNet20 and the ResNet20-WNLL trained with PGD adversarial training.

6. **Concluding remarks.** In this paper, by analyzing the influence of adversarial perturbations on the geometric structure of the DNN features, we propose to defend against adversarial attacks by using a data-dependent activation function. We further show our defenses are additive to other defenses, namely total variation minimization, training data augmentation, and projected gradient descent adversarial training. Results on ResNet20 and ResNet56 with CIFAR10 benchmark reveal that these defenses improve robustness to adversarial perturbation significantly. Total variation minimization simplifies the adversarial images, which is very useful in removing adversarial perturbation. The data-dependent activation framework raises the accuracy of PGD adversarial training around 23% under different attack strengths. An interesting direction to explore is to combine these methods with other denoising methods to remove adversarial perturbation.

Acknowledgments. This material is based on research sponsored by the National Science Foundation under grant number DMS-1924935 and DMS-1554564 (STROBE). The Air Force Research Laboratory under grant numbers FA9550-18-0167 and MURI FA9550-18-1-0502, the Office of Naval Research under grant number N00014-18-1-2527. ALB is partially supported by the Simons Math + X award.

REFERENCES

[1] N. Akhtar and A. Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, *IEEE Access*, **6** (2018), 14410–14430, arXiv:1801.00553.

[2] N. Akhtar, J. Liu and A. Mian, *Defense Against Universal Adversarial Perturbations*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[3] A. Athalye, N. Carlini and D. Wagner, *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*, International Conference on Machine Learning, 2018.

[4] A. Athalye, L. Engstrom, A. Ilyas and K. Kwok, *Synthesizing Robust Adversarial Examples*, International Conference on Machine Learning, 2018.

[5] Y. Bengio, N. Leonard and A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, arXiv preprint, arXiv:1308.3432, 2013.

[6] W. Brendel, J. Rauber and M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, arXiv preprint, arXiv:1712.04248, 2017.

[7] N. Carlini and D. A. Wagner, Towards evaluating the robustness of neural networks, *IEEE European Symposium on Security and Privacy*, (2016), 39–57.

[8] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu and J. Li, *Boosting Adversarial Attacks with Momentum*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[9] I. J. Goodfellow, J. Shlens and C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint, arXiv:1412.6275, 2014.

[10] K. Grosse, N. Papernot, P. Manoharan, M. Backes and P. McDaniel, Adversarial perturbations against deep neural networks for malware classification, arXiv preprint, arXiv:1606.04435, 2016.

[11] A. Guisti, J. Guzzi, D. C. Ciresan, F. L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Carlo and et al, A machine learning approach to visual perception of forecast trails for mobile robots, *IEEE Robotics and Automation Letters*, (2016), 661–667.

[12] C. Guo, M. Rana, M. Cisse and L. van der Maaten, *Countering Adversarial Images Using Input Transformations*, International Conference on Learning Representations, 2018.

[13] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, In *CVPR*, (2016), 770–778.

[14] A. Ilyas, L. Engstrom, A. Athalye and J. Lin, *Black-box Adversarial Attacks with Limited Queries and Information*, International Conference on Machine Learning, 2018.

[15] D. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.

[16] A. Kurakin, I. J. Goodfellow and S. Bengio, Adversarial examples in the physical world, arXiv preprint, arXiv:1607.02533, 2016.

[17] K. Lee, K. Lee, H. Lee and J. Shin, *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks*, ArXiv e-prints, 2018.

[18] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu and J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[19] Y. Liu, X. Chen, C. Liu and D. Song, Delving into Transferable Adversarial Examples and Black-box Attacks, arXiv preprint, arXiv:1611.02770, 2016.

[20] Y. Luo, X. Boix, G. Roig, T. A. Poggio and Q. Zhao, *Foveation-based Mechanisms Alleviate Adversarial Examples,* CoRR, abs/1511.06292, 2015.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, *Towards Deep Learning Models Resistant to Adversarial Attacks*, International Conference on Learning Representations, 2018.

[22] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi and P. Frossard, Universal adversarial perturbations, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] S.-M. Moosavi-Dezfooli, A. Shrivastava and O. Tuzel, *Divide, Denoise, and Defend Against Adversarial Attacks*, CoRR, abs/1802.06806, 2018.

[24] T. Na, J. Hwan Ko and S. Mukhopadhyay, *Cascade Adversarial Machine Learning Regularized with A Unified Embedding*, International Conference on Learning Representations, 2018.

[25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, The limitations of deep learning in adversarial settings, *IEEE European Symposium on Security and Privacy*, (2016), 372–387.
[26] N. Papernot, P. McDaniel, A. Sinha and M. Wellman, Sok: Towards the science of security and privacy in machien learning, arXiv preprint, arXiv:1611.03814, 2016.
[27] N. Papernot, P. McDaniel, X. Wu, S. Jha and A. Swami, *Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks*, IEEE European Symposium on Security and Privacy, 2016.
[28] N. Papernot, P. D. McDaniel and I. J. Goodfellow, *Transferability in Machine Learning: From Phenomena to Black-box Attacks Using Adversarial Samples*, CoRR, abs/1605.07277, 2016.
[29] A. Prakash, N. Moran, S. Garber, A. DiLillo and J. Storer, Deflecting adversarial attacks with pixel deflection, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
[30] L. Rudin, S. Osher and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D: Nonlinear phenomena*, **60** (1992), 259–268.
[31] P. Samangouei, M. Kabkab and R. Chellappa, *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*, In *International Conference on Learning Representations*, 2018.
[32] Z. Shi, B. Wang and S. J. Osher, Error Estimation of Weighted Nonlocal Laplacian on Random Point Cloud, arXiv preprint, arXiv:1809.08622, 2014.
[33] Y. Song, T. Kim, S. Nowozin, S. Ermon and N. Kushman, *Pixeldefend: Leveraging Generative Models to Understand and Defend Against Adversarial Examples,* In *International Conference on Learning Representations*, 2018.
[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan and I. Goodfellow, Intriguing properties of neural networks, arXiv preprint, arXiv:1312.6199, 2013.
[35] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh and P. McDaniel, *Ensemble Adversarial Training: Attacks and Defenses*, International Conference on Learning Representations, 2018.
[36] B. Wang, X. Luo, Z. Li, W. Zhu, Z. Shi and S. Osher, Deep neural nets with interpolating function as output activation, arXiv preprint, arXiv:1802.00168, 2018.
[37] B. Wang, Z. Shi and S. Osher, *ResNets Ensemble via the Feynman-Kac Formalism to Improve Natural and Robust Accuracies*, Advances in Neural Information Processing Systems, 2019.
[38] B. Wang and S. Osher, Graph interpolating activation improves both natural and robust accuracies in data-efficient deep learning, arXiv preprint, arXiv:1907.06800, 2019.
[39] X. Wu, U. Jang, J. Chen, L. Chen and S. Jha, *Reinforcing Adversarial Robustness Using Model Confidence Induced by Adversarial Training*, International Conference on Machine Learning, 2018.
[40] C. Xie, J. Wang, Z. Zhang, Z. Ren and A. Yuille, *Mitigating Adversarial Effects Through Randomization*, International Conference on Learning Representations, 2018.

Received November 2019; revised April 2020.

E-mail address: wangbaonj@gmail.com
E-mail address: atlin@math.ucla.edu
E-mail address: pyin@albany.edu
E-mail address: zhu@math.duke.edu
E-mail address: bertozzi@math.ucla.edu
E-mail address: sjo@math.ucla.edu