LAFIT: Efficient and Reliable Evaluation of Adversarial Defenses With Latent Features

Yunrui Yu, *Student Member, IEEE*, Xitong Gao, *Member, IEEE*, and Cheng-Zhong Xu, *Fellow, IEEE*

***Abstract*—Deep convolutional neural networks (CNNs) can be easily tricked to give incorrect outputs by adding tiny perturbations to the input that are imperceptible to humans. This makes them susceptible to adversarial attacks, and poses significant security risks to deep learning systems, and presents a great challenge in making CNNs robust against such attacks. An influx of defense strategies have thus been proposed to improve the robustness of CNNs. Current attack methods, however, may fail to accurately or efficiently evaluate the robustness of defending models. In this paper, we thus propose a unified ℓ_p white-box attack strategy, LAFIT, to harness the defender's latent features in its gradient descent steps, and further employ a new loss function to normalize logits to overcome floating-point-based gradient masking. We show that not only is it more efficient, but it is also a stronger adversary than the current state-of-the-art when examined across a wide range of defense mechanisms. This suggests that adversarial attacks/defenses could be contingent on the effective use of the defender's hidden components, and robustness evaluation should no longer view models holistically.**

***Index Terms*—Adversarial robustness, white-box attacks, latent feature attack.**

I. Introduction

RECENT years have witnessed a wide scale adoption of deep learning in many safety-critical systems such as aviation [1], [2], [3], medical diagnosis [4], [5], autonomous driving [6], [7], [8]. Deep learning models, however, are highly vulnerable to adversarial attacks: adding small specially-crafted perturbations to the input image may cause drastic changes in the model outputs [9], [10], [11]. As safety-critical systems are increasingly automated by CNNs, adversarial attacks could potentially endanger the safety of such systems, it is now incumbent upon us to not only defend deep learning models against adversarial attacks, but also provide an efficient and accurate evaluation of their robustness under attack.

The strongest assumption commonly used for generating adversarial examples is the white-box ℓ^p threat model, where the adversary assumes full visibility of the defender [11] and can perturb a natural input by adding a small perturbation within the ℓ_p-norm. For instance, the model architecture, parameters, training algorithm and dataset, etc. are completely exposed to the attacker. By leveraging the gradient of the output loss w.r.t. the input image, white-box attacks, for instance projected gradient descent (PGD) [12], thus operate under this assumption to craft adversarial examples that can effectively decimate the accuracy of non-robust models. Many new techniques to improve the robustness of DL models have since been proposed to defend against such attacks. Recent years have therefore seen a tug of war between adversarial attack [10], [12], [13], [14], [15], [16], [17] and defense [12], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29] strategies. Attackers search for a perturbation that maximizes the loss of the model output, typically through gradient ascent, whereas defenders attempt to flatten the loss landscape w.r.t. the perturbation. Many defenses employ a wide range of non-trivial gradient masking techniques [30], [31], [32], [33], [34], [35], i.e., rendering gradients unusable by inducing numerical instability. Recent literatures [17] reveal that such defenses may offer a false sense of security, as high accuracies under existing gradient-based attacks, does not automatically correlate with high robustness. Reliably evaluating their robustness thus often requires manually designing specialized attacks [36].

From a human perception perspective, shallow layers of deep learning models extract simple local textures, whereas deep layers specialize in recognizing shapes and patterns in complex objects [37], [38]. Intuitively, we expect that incorrectly extracted features from shallow layers often cannot be pieced together to form correct high-level features. We illustrate this phenomenon with LPGD, which equips PGD with the ability to attack *only one* of the intermediate layers (Fig. 1). LPGD scrambles the features of an intermediate layer by training an auxiliary classifier for the layer, and then attacking a layer by maximizing the loss of this classifier. In deeper layers, we observe greater discrepancies between the pairs of features extracted from the natural images and their corresponding adversaries.

Some defense strategies [30], [31], [32], [33], [34], [35], [39] report high robustness under PGD attacks, giving a false sense of robustness. Understandably, these models specialize their defenses to counter existing conventional attacks. We

Manuscript received 15 August 2022; revised 16 August 2023; accepted 3 October 2023. Date of publication 13 October 2023; date of current version 5 December 2023. This work was supported in part by the Science and Technology Development Fund of Macao S.A.R (FDCT) under Grants 0015/2019/AKP, 0123/2022/AFJ, and 0081/2022/A2, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2020B1515130004, in part by the National Natural Science Foundation of China under Grant 62376263, and in part by the Basic Research Program of Shenzhen under Grant JCYJ20190812160003719. This work was carried out in part at SICC which is supported by SKL-IOTSC, University of Macau. Recommended for acceptance by D. Crandall. *(Yunrui Yu and Xitong Gao contributed equally to this work.) (Corresponding author: Cheng-Zhong Xu.)*

Yunrui Yu and Cheng-Zhong Xu are with the State Key Lab of IOTSC, University of Macau, Taipa, Macau 999078, China (e-mail: yb97445@um.edu.mo; czxu@um.edu.mo).

Xitong Gao is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: xt.gao@siat.ac.cn).

Digital Object Identifier 10.1109/TPAMI.2023.3323698

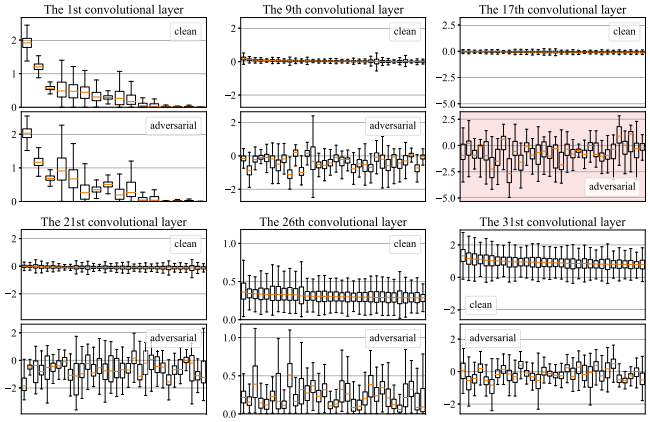


Fig. 1. Distribution of 32 most excited channels on average of convolutional layers in a naturally trained WideResNet-32-10 when being shown CIFAR-10 "airplane" images. We compare that against the corresponding channel activations under adversarial "airplane" examples with LPGD-10. Attacking the 17th layer (shaded in red) resulted in scrambled features across the entire model and incorrect final model outputs.

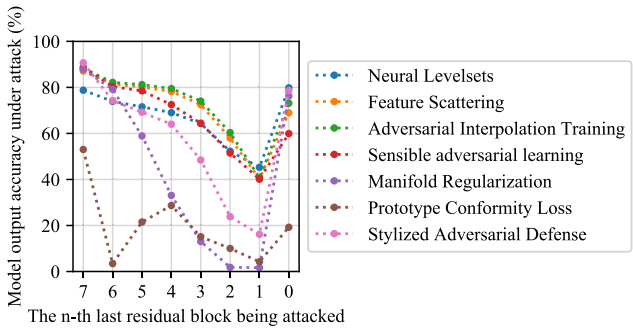


Fig. 2. We used LPGD-100, or PGD with 100 iterations to attack *only* the nth residual block of a "robust" model. Except the 0th denotes the output layer and PGD is used. To our surprise, adversaries generated by attacking an *early* layer are highly transferable to the *final* classification output of the model. Respective defending models are [30], [31], [32], [34], [35], [39], [40].

hypothesize one of the reasons why PGD fails to break through their defenses is because of its *model-holistic* nature. This notion prompts us to consider two important questions: *Can latent features be vulnerable to attacks; and subsequently, can the falsely extracted features cascade to make the model output incorrect?* It turns out that the new adversarial examples generated by LPGD can detrimentally affect the accuracies of the above defenses. Experiments in Fig. 2 show that although the "robust" models produced by these defenses can effectively defend against PGD, they fail spectacularly when faced with an attack which simply targets their latent features. This finding may also suggest that a flat model loss landscape w.r.t. the input image does not necessarily entail flat latent features w.r.t. the input (Fig. 3).

Nevertheless, existing attack and defense strategies approach the challenge of evaluating or promoting the white-box model robustness in a *model-holistic* manner. Namely, for classifiers, they regard the model as a single non-linear differentiable function $f : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^K$ that maps the input image $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ to output logits $\mathbf{y} \in \mathbb{R}^K$. While these approaches can

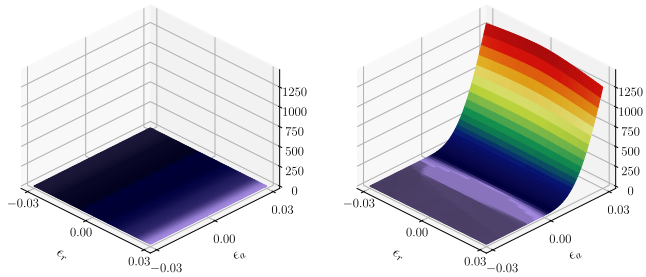


Fig. 3. Averaged loss surfaces of [39] across all samples in the image space $\mathbf{x} + \boldsymbol{g}\epsilon_{\mathrm{a}} + \boldsymbol{g}^{\perp}\epsilon_{\mathrm{r}}$ after 10 iteration of PGD (a) and LPGD (b) attacks. Here, g and $g^{\perp}$ respectively denote normalized adversarial perturbation after 10 iterations of gradient updates at the natural input $\mathbf{x}$, and its random orthogonal.

be readily applied across most defenses, they tend to ignore the latent features extracted by the intermediate layers within the model f.

Motivated by the findings above, this paper proposes a new attack strategy, LAFIT, which seeks to harness latent features in a generalized attack framework. To push the envelope of existing state-of-the-art (SOTA) in adversarial robustness assessment, it draws inspiration from recent effective techniques discovered, such as the use of momentum [15], [17], step size schedule [17], [41], and multi-targeted attacks [24], [41], [42]. An initial version of this paper can be found at [43], which makes the following contributions:

- We introduce how intermediate layers can be leveraged in adversarial attacks.
- We show that latent features provide faster convergence, and accelerate gradient-based attacks.
- By combining multiple effective attack tactics, we propose LAFIT. Empirical results show that it rivals competing methods in both the attack performance and computational efficiency. We perform extensive ablation analysis of its hyperparameters and components.

In addition, this paper improves upon [43] by including further contributions below:

- We further enhance the effectiveness of LAFIT, and expand the experiments to compare against recent SOTA attack strategies on additional defenses. Finally, we extend LAFIT to evaluate defending models under the ℓ_2-norm threat model, and CIFAR-100 defenses (Table I in Section IV).
- New extensions to LAFIT allows it to extend to ensemble-based defenses (Section III-G). Empirical results show its effectiveness (Section IV).
- Comparisons of the design of the surrogate loss function used by LAFIT against other alternatives (Section V).
- Extensive ablation analyses of individual components used by LAFIT, and sensitivity analyses of the hyperparameters searched during attack (Section V).
- We provide visualization and analyses of the challenges of gradient masking in existing attacks. Experiments show that improving the loss function alone is insufficient to

TABLE I
COMPARING ACCURACY UNDER ATTACK (%) OF LAFIT AGAINST ITERATIVE METHODS [12], [15], [49], [50], AUTOATTACK (**AA**) [17], ADAPTIVE AUTO ATTACK (**A^3**) ACROSS VARIOUS DEFENSE STRATEGIES

Defense method Worst-case complexity	**Clean** 1	**Nominal**	**PGD** 100	**BIM** 100	**FAB** 100	**LAF$_1$** 100	**AA** 8.3 k	**A^3** 5 k	**CAA** 1.9 k	**LAF$_2$** 1 k	**LAF$_3$** 1.9 k	**LAF$_4$** 4.6 k	**Δ**	***I*** $\times 10^2$
CIFAR-10, ℓ_∞, $\epsilon = 8/255$														
Uncovering limits [28]†	91.10	65.87	68.55	70.61	66.85	65.99	65.88	66.19	65.86	65.83	65.79	**65.78**	−0.09	3
Fixing data augmentation [78]†	88.50	64.64	67.71	69.25	65.25	64.80	64.58	65.37	64.63	64.62	64.54	**64.53**	−0.11	19
Fixing data augmentation [78]†	88.54	64.25	67.37	68.80	64.86	64.44	64.20	64.59	64.24	64.28	**64.20**	**64.20**	−0.05	19
Generated data [79]‡	87.50	63.44	65.65	67.53	64.15	63.81	63.38	63.95	63.42	63.41	63.34	**63.33**	−0.11	19
Proper Definition [80]‡	89.01	63.35	66.70	68.50	64.50	63.50	63.35	63.51	63.28	63.35	**63.26**	**63.26**	−0.09	9
Proper Definition [80]‡	88.61	61.04	64.68	66.51	62.07	61.18	61.04	61.25	**60.97**	60.98	60.97	**60.97**	−0.07	2
Weight perturbation [26]†	88.25	60.04	63.26	65.33	60.72	60.19	60.04	60.00	59.98	59.98	59.97	**59.97**	−0.07	2
Unlabeled data [19]†	89.69	62.50	62.17	65.77	60.72	59.77	59.53	59.49	59.54	59.46	59.44	**59.43**	−3.07	5
Misclassification-aware [25]†	87.50	65.04	62.55	65.57	57.60	56.78	56.29	56.27	56.30	56.38	56.18	**56.16**	−8.88	19
HYDRA [71]†	88.98	62.24	59.98	63.87	58.33	57.38	57.14	57.19	57.14	57.10	57.03	**57.02**	−5.22	4
Pre-training [81]†	87.11	57.40	57.54	60.25	55.58	55.15	54.92	54.89	54.87	54.85	54.79	**54.78**	−2.62	6
Hypershpere embedding [24]	85.14	62.14	62.17	63.15	54.47	53.94	53.74	53.70	53.70	53.73	53.68	**53.66**	−8.48	8
Overfitting [67]	85.34	58.00	57.24	59.49	54.30	53.74	53.42	53.40	53.36	53.39	**53.34**	**53.34**	−4.66	8
Robustness (Python library) [82]	87.03	53.29	52.32	56.52	50.67	49.82	49.21	49.23	49.23	49.35	49.14	**49.12**	−4.17	19
YOPO [83]	87.20	47.98	46.15	50.69	45.80	45.33	44.83	44.86	44.81	44.88	**44.73**	**44.73**	−3.25	19
Fast adversarial training [69]	83.34	46.06	46.52	50.44	44.25	43.91	43.41	43.44	43.43	43.51	43.33	**43.32**	−2.74	19
MMA training [84]	83.28	47.18	47.29	54.65	46.88	41.36	40.21	40.72	40.43	40.28	39.84	**39.76**	−7.42	19
Feature scattering [30]	89.98	60.60	69.01	74.54	43.42	38.62	36.64	37.02	37.17	36.29	36.14	**35.98**	−24.62	5
Interpolation [31]	90.25	68.70	73.13	74.95	43.34	36.71	36.45	37.80	37.71	35.60	35.50	**33.35**	−35.35	3
Sensible adversarial learning [32]	91.51	57.23	59.93	66.85	41.87	36.93	34.22	35.20	35.83	33.46	33.35	**33.18**	−24.05	5
Stylized adversarial defense [33]	93.29	78.68	82.87	82.50	19.14	14.18	13.42	14.71	16.47	11.75	11.61	**11.10**	−67.58	5
Manifold regularization [34]	90.84	77.68	77.63	77.30	27.18	0.46	1.35	1.10	3.89	0.26	0.26	**0.23**	−77.45	1
Prototype conformity loss [35]	89.16	32.32	19.42	38.77	5.81	0.20	0.28	1.16	1.19	0.06	0.06	**0.04**	−32.28	1
CIFAR-10, ℓ_∞, $\epsilon = 0.031$														
Self-adaptive training [65]	83.48	58.03	56.58	58.89	54.56	53.51	53.34	54.63	53.30	53.26	53.23	**53.20**	−4.83	4
TRADES [21]	84.92	56.43	55.50	58.06	53.96	53.28	53.08	53.03	53.06	52.99	**52.98**	**52.98**	−3.45	4
Neural level sets [39]	81.30	79.67	79.83	79.83	40.98	40.07	40.22	39.90	39.94	39.91	39.82	**39.77**	−39.90	1
CIFAR-10, ℓ_2, $\epsilon = 0.5$														
Uncovering limits [28]	90.90	74.50	75.68	76.23	74.89	74.59	74.50	74.57	74.50	74.49	74.48	**74.47**	−0.03	8
Weight perturbation [26]	88.51	73.66	74.93	75.33	73.82	73.68	73.66	73.66	73.66	73.65	**73.64**	**73.64**	−0.02	8
Robustness (Python library) [82]	90.83	70.11	70.51	71.59	69.61	69.28	69.24	69.24	69.24	69.24	69.21	**69.20**	−0.91	9
Overfitting [67]	88.67	71.60	69.26	70.06	68.08	67.90	67.68	67.70	67.68	67.71	67.64	**67.61**	−3.99	19
Decoupling direction & norm [85]	89.05	67.60	67.38	68.46	66.84	66.55	66.44	66.45	66.45	66.44	66.44	**66.42**	−1.18	7
MMA training [84]	88.02	66.18	67.01	68.28	66.50	63.14	66.09	63.09	63.11	63.07	63.06	**62.98**	−3.2	10
CIFAR-100, ℓ_∞, $\epsilon = 8/255$														
Weight perturbation [26]	60.38	28.86	33.68	34.70	29.25	29.65	28.86	28.80	28.82	29.08	**28.80**	**28.80**	−0.06	19
Pre-training [81]†	59.23	33.50	33.70	35.09	28.83	28.97	28.42	28.33	28.45	28.54	28.25	**28.25**	−5.25	19
Progressive Hardening [86]	62.82	24.57	26.75	29.66	25.14	25.56	24.57	**24.51**	24.58	24.85	**24.51**	**24.51**	−0.06	19
Overfitting [67]	53.83	28.10	20.95	23.10	19.49	20.08	18.95	18.90	18.96	19.10	18.87	**18.86**	−9.24	19

The "Δ" column shows the difference between the reported ("Nominal") and LAFIT accuracies, whereas *I* reports the number of iterations required by LAFIT to match these values. Models marked with † were additionally trained with unlabeled datasets [77], and models with ‡ use generated data from a DDPM trained with the original training set. Methods with multiple rows are models of different sizes.

improve robustness evaluation, hence it is pertinent to leverage latent features (Section VI).

To the best of our knowledge, LAFIT is currently the strongest against a wide variety of defense mechanisms. Compared to existing SOTAs that diversify their strategies by comprising multiple attacks, LAFIT is a *unified* algorithm that achieves faster convergence and much improved attack success rates. Since latent features are vulnerable to adversarial attacks, which could in turn break robust models. We believe the future evaluation of model robustness could be contingent on how to make effective use of the hidden components of a defending model. In short, model robustness should no longer be viewed from a holistic perspective.

II. PRELIMINARIES & RELATED WORK

A. Adversarial Attacks

Since the discovery of *adversarial examples* [9], i.e., adding a small perturbation to the original image can fool a model under attack into making an incorrect prediction, it has revealed safety concerns [44], [45] of deep learning systems, can be used to improve transfer learning [46], GAN training [47], DNN interpretability [48] and etc.

Let us assume a defending classifier model $\mathrm{f}_{\boldsymbol{\theta}}(\mathbf{x})$, with the model function $\mathrm{f}_{\boldsymbol{\theta}} : \mathcal{I} \rightarrow \mathbb{R}^K$ maps the input image $\mathbf{x} \in \mathcal{I} \subset \mathbb{R}^{C \times H \times W}$ to its classification result $\mathbf{z}$. Here, C is the number of channels, in the image (typically 3 for RGB-colored inputs), H and W respectively the height and width of the image, and $\boldsymbol{\theta}$ denotes model parameters. Finally, $\arg\max \mathbf{z} \in \mathcal{C}$, where $\mathcal{C}$ is the set of all classes, gives the model's class prediction of the input image $\mathbf{x}$.

The attacker's objective is thus to find an *adversarial example* $\hat{\mathbf{x}} \in \mathcal{I}$ of the model under attack $\mathrm{f}_{\boldsymbol{\theta}}$ by (approximately) solving the optimization problem:

$$\max_{\hat{\mathbf{x}} \in \mathcal{I} \wedge \mathrm{d}(\mathbf{x}, \hat{\mathbf{x}}) \leq \epsilon} \mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}), y), \tag{1}$$

To confine the range of perturbation, $\mathrm{d}(\mathbf{x}, \hat{\mathbf{x}}) \leq \epsilon$ limits the distance between the original $\mathbf{x}$ and the adversarial $\hat{\mathbf{x}}$ to be less than or equal some small constant ϵ. Here, $\mathcal{L} : \mathbb{R}^K \times \mathcal{C} \rightarrow \mathbb{R}$ is typically the *softmax cross-entropy* (SCE) loss taking as inputs both the output $\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$ and the ground truth label y. By maximizing the loss value, one may arrive at a $\hat{\mathbf{x}}$ such that $\arg\max \mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}) \neq y$. In other words, by slightly perturbing the natural image $\mathbf{x}$ to become $\hat{\mathbf{x}}$, the model can be fooled to produce an incorrect classification.

The distance metric $\mathrm{d}(\mathbf{x}, \hat{\mathbf{x}})$ represents the ℓ_p-distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$, i.e., $\mathrm{d}(\mathbf{x}, \hat{\mathbf{x}}) \triangleq \|\mathbf{x} - \hat{\mathbf{x}}\|_p$. Different choices of p are explored in literature. In this paper, we focus on the ℓ_p white-box scenario which completely exposes to the attacker to the inner mechanisms of the defense, i.e., it reveals to the attacker the model architecture, its parameters, training data and algorithms, and etc. In particular, we consider the standard euclidean distance ($p = 2$) as used by [9], [13], [14], and the maximum pixel perturbation distance ℓ_∞, i.e., $|\max(\mathbf{x} - \hat{\mathbf{x}})|$ as in [10], [12].

The optimal solution of (1) is, however, generally unattainable. In practice, we instead seek after approximate solution, often with gradient-based methods. A commonly used attack by defenders to evaluate white-box adversarial robustness is the *projected gradient descent* (PGD). PGD finds an adversarial example by performing the following iterative update [12]:

$$\hat{\mathbf{x}}_{i+1} = \mathcal{P}_{\epsilon,\mathbf{x}}(\hat{\mathbf{x}}_i + \alpha_i \operatorname{sign}(\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i), y))). \tag{2}$$

Initially, $\hat{\mathbf{x}}_0 = \mathcal{P}_{\epsilon,\mathbf{x}}(\mathbf{x} + \mathbf{u})$, where $\mathbf{u} \sim \mathcal{U}([-\epsilon, \epsilon])$, i.e., $\mathbf{u}$ is a uniform random noise bounded by $[-\epsilon, \epsilon]$. The function $\mathcal{P}_{\epsilon,\mathbf{x}} : \mathbb{R}^{C\times H\times W} \to \mathcal{I}$ clips the range of its input into the ϵ-ball ℓ_p-distance neighbor of the original image $\mathbf{x}$ and the $\mathcal{I}$. The term $\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i), y)$ computes the gradient of the loss w.r.t. the input $\hat{\mathbf{x}}_i$. Finally, α_i is the step size, and for each element in the tensor $\mathbf{z}$, $\operatorname{sign}(\mathbf{z})$ returns one of 1, 0 or -1, if the value is positive, zero or negative respectively. For simplicity, we define $\mathrm{PGD}_{\epsilon,\mathbf{x},\mathbf{y}}(\mathcal{L}, \boldsymbol{\alpha}, i)$ to be $\hat{\mathbf{x}}_i$ synthesized with PGD using a sequence of step sizes $\boldsymbol{\alpha}$ and the loss function $\mathcal{L}$ on the original image $\mathbf{x}$. Other gradient-based methods include fast gradient-sign method (FGSM) [10], basic iterative method (BIM) [49], momentum iterative method (MIM) [15] and fast adaptive boundary attack (FAB) [50] for ℓ_∞-norm attacks. Carlini and Wagner (C&W) [14] and DeepFool [13] can additionally carry out ℓ_2-norm attacks. Similar to PGD, they only iteratively leverage the loss gradient $\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i), y)$, as all adopt a holistic view on the model $\mathrm{f}_{\boldsymbol{\theta}}$.

B. Tactics Employed by Adversarial Attacks

Many auxiliary techniques can push the limit of existing attack methods, which we will discuss in the following section.

Surrogate Losses: Because the SCE loss function $\mathcal{L}^{\mathrm{sce}}$ in the objective (1) is highly non-linear, easily saturated, and normally evaluated with limited floating-point precision, gradient-based attacks may experience vanishing/exploding gradients and difficulty converging [17]. Recent attack methods hence instead use *surrogate losses* for gradient calculation [14], [41], and optimize an alternative objective by replacing $\mathcal{L}^{\mathrm{sce}}$ with a custom surrogate loss function. Assuming $\mathbf{z} = \mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$, and allowing the notation $\mathbf{z}_i$ to represent the ith component of $\mathbf{z}$, popular and effective choices to replace the SCE objective include the difference-of-logits (DL) [14], also known as hinge loss, where $\mathcal{C} \setminus y$ denotes the set of all classes except y:

$$\mathcal{L}^{\mathrm{cw}}(\mathbf{z}, y) \triangleq -\delta_y = -\mathbf{z}_y + \max_{\hat{y}\in\mathcal{C}\setminus y} \mathbf{z}_{\hat{y}}, \tag{3}$$

and the difference-of-logits ratio (DLR) loss [17]:

$$\mathcal{L}^{\mathrm{dlr}}(\mathbf{z}, y) \triangleq \mathcal{L}^{\mathrm{cw}}(\mathbf{z}, y)/(\mathbf{z}_{\pi_1} - \mathbf{z}_{\pi_3}), \tag{4}$$

where $\mathbf{z}_{\pi_1}$ and $\mathbf{z}_{\pi_3}$ respectively denote the 1th and 3th largest components of $\mathbf{z}$. It is clear that maximizing the alternative goals is consistent with the original objective of making $\arg\max f(\hat{\mathbf{x}}) \neq y$. Moreover, as the surrogate losses avoid the softmax operation, they are less easily saturated, allowing gradient-based optimizations to converge more efficiently.

Improving Gradient Optimization Algorithms: Momentum-based gradient methods can notably improve convergence rates, and are widely used by white-box attacks to further increase their attack success rates [15], [41]. Using the Adam optimizer [51] also shows effectiveness at quickly finding adversarial examples [14].

Step Size Schedule: A step-size schedule [17], [41] with a decaying step-size in relation to the iteration count could improve the overall success rate. With a fixed step size, gradient-based attacks such as PGD may oscillate among suboptimal solutions. For this reason, recent attacks propose to decay the step size either linearly [41], or with an adaptive schedule [17].

Multi-Targeted Attacks: Using the optimization objective (1) alone can fail to find adversarial examples. To further improve success rates, recent literatures [17], [24], [42] introduce the use of label-specific surrogate losses. Instead of maximizing the untargeted objective $\mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}), y)$, they propose to minimize $\mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}), t)$, where $t \in \mathcal{C} \setminus y$ enumerates over all possible target labels except the ground truth label y.

Multiple Restarts: In addition to the above tactics, attacks may restart gradient-based attacks multiple times, each with a random initial perturbation usually sampled from a uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$, in order to avoid converging to suboptimal local minima [11]. Output diversified sampling (ODS) [52] diversifies starting points to prevent similar local minima, and adaptive auto attack (A^3) [53] further learns to sample initial points adaptively.

Ensemble of Attacks: To further improve success rates, recent strong baselines for robustness evaluation may comprise a large arsenal of diverse attack algorithms. For instance, AutoAttack [17] incorporates PGD-based attacks with query-efficient black-box attacks such as Square Attack [54] and FAB [50]. Composite Adversarial Attacks (CAA) [55] search for optimal sequences of diverse attack steps with an evolutionary algorithm. Although these techniques are powerful adversaries to defending models, their performance is a direct result of high computational costs. In stark contrast, without resorting to a large arsenal of attacks, LAFIT achieves higher success rates with much faster convergence, while using a *single* unified attack algorithm that is easy to implement.

Leveraging Latent Features: Finally, there are a few recent publications that leverage latent features in their attacks [56], [57]. Recent publications on black-box attacks also demonstrate that latent features can enhance attack transferability [58], [59], which also motivates LAFIT to leverage latent features in white-box attacks. Unlike these methods, LAFIT considers ℓ_p-norm white-box attacks, and further differentiates itself from them as it learns to attack defending models.

Generative Methods: Generative networks that learn from the adversarial loss can synthesize adversarial examples [60], [61]. One can further enhance this approach with generative adversarial networks (GANs) [62], where the discriminator network encourages the distribution of adversarial examples to become indistinguishable from that of natural examples [16], [63].

C. Defending Against Adversarial Examples

The objective of achieving robustness against adversarial examples can be formalized as a saddle point problem, which finds model parameters that minimize the adversarial loss [12]:

$$\min_{\boldsymbol{\theta}} \mathsf{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\text{train}}} \left[\max_{\hat{\mathbf{x}}\in\mathcal{I}\wedge \mathrm{d}(\mathbf{x},\hat{\mathbf{x}})\leq\epsilon} \mathcal{L}(\mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}), y) \right], \tag{5}$$

where $\mathcal{D}_{\text{train}}$ contains pairs of input images $\mathbf{x}$ and ground truth labels y, and $\mathcal{L}$ is by default the SCE loss. A straightforward approach to approximately solving the above objective (5) is through *adversarial training* [64], which trains the model with adversarial examples computed on-the-fly using, for instance, PGD [12].

Adversarial Training Methods: Many adversarial defense strategies follow the same paradigm, but train the model with different loss objective functions in order to further boost robustness. Along with the standard classification loss, TRADES [21] minimizes the multi-class calibrated loss between the output of the original image and the one of the adversarial example. Misclassification-aware regularization [25] encourages the smoothness of the network output, even when it produces misclassified results. Self-adaptive training [40] allows the training algorithm to adapt to the noise added to the training data. Feature scattering [30] generates adversarial examples for training by maximizing the distances between features extracted from the natural and adversarial examples. Neural level-sets [39] and sensible adversarial training [32] use different proxy robustness objectives for adversarial training. Hypersphere embedding [24] normalizes weights and features to be on the surface of hyperspheres, and also normalizes the angular margin of the logits layer. Prototype conformity loss [35], and manifold regularization [34] adopt different regularization losses to allow the model to learn a smooth loss landscape w.r.t. changes in $\mathbf{x}$. Stylized adversarial defense [33] and learning-to-learn (L2L) [61] propose to use neural networks to generate adversarial examples for training. Self-training with unlabeled data [19], [20] can substantially improve robustness in a way that cannot be trivially broken by adversarial attacks. Finally, others provide practical considerations and tricks for stronger adversarial defense under adversarial training [11], [27], [65], [66], [67].

Efficient Adversarial Training: Adversarial training often requires several iterations to compute adversarial examples for each model parameter update. This process is considerably more time-consuming than traditional training with natural examples. Adversarial training for free [18] address this problem by interleaving adversary updates with model updates for efficient training of robust CNNs. Fast adversarial training [68] further accelerates the training by using a simpler FGSM-based adversary.

Efficient Robust Models: As robust models may be substantially larger than non-robust ones, there have been a recent interest in making them more space and time efficient. Alternating direction method of multipliers (ADMM) has been applied to prune and adversarial train CNNs jointly [69]. HYDRA [70] preserves the robustness of pruned models by integrating the pruning objective into the adversarial loss optimization.

Ensemble-Based Robustness: Adversarial training can easily overfit [19] and is often compute-intensive. The resulting models are often unable to achieve high clean accuracy [71]. Ensemble-based defenses thus aim to work around the above limitations by training multiple small non-robust models to form a large robust ensemble. These defense strategies [22], [23], [29] learn to reduce the transferability of adversarial examples among sub-models, presenting an obstacle for the attacker to successfully fool sub-models simultaneously. This approach allows the ensemble model to gain a certain level of robustness even without adversarial training.

III. The LAFIT Method

A. High-Level Overview

We introduce LAFIT by providing a high-level illustration (Fig. 4) of its attack procedure. First, to obtain a firm grasp on latent features, it starts by training fully-connected layers for each residual block with the training set until convergence. Note that we ensure the original model $\mathrm{f}_{\boldsymbol{\theta}}$ to remain constant during this process. To compute adversarial examples $\hat{\mathbf{x}}$, we maximize a novel surrogate loss $\mathcal{L}^{\text{lf}}(\hat{\mathbf{x}}, y)$, which leverages latent features from individual layers. For the assessment of adversarial robustness, we then transfer the generated adversarial example to the original model $\mathrm{f}_{\boldsymbol{\theta}}$ for evaluation.

B. Logit Normalization

Many defenses may rely on gradient masking techniques [72], [73], i.e., making models produce unusable gradients, to stymie adversarial attackers. Gradients computed from the SCE loss has been notoriously shown to easily underflow in floating-point arithmetic for such defending models [14], [17]. Consider the DL as defined in (3) $\delta_y = \mathbf{z}_y - \max_{\hat{y}\in\mathcal{C}\setminus y} \mathbf{z}_{\hat{y}}$, and $\mathbf{z} = f_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$, if $\delta_y \geq \lambda$, where $\lambda \approxeq 16.6, 103.3, 744.4$ respectively under half-, single-, and double-precision floating-point arithmetics, the result of the softmax operation $\mathbf{s} = \mathrm{softmax}(\mathbf{z})$ would saturate, and $\mathbf{s}_y$ thus evaluates to 1. Conversely, $\delta_y \leq -\lambda$ gives $\mathbf{s}_y = 0$. Under the SCE loss, we therefore identify that the defending model may exhibit zero-valued gradients, with the root cause of this attributed to this saturation behavior (Figs. 5 and 8).

For this reason, *surrogate loss* functions have been proposed [14], [17], [41] to work around this limitation. Despite their effectiveness in breaking through defenses, it is difficult to interpret why they work as they no longer maximize the original adversarial objective (1) directly.

Yet, it is desirable to maximize the original softmax cross-entropy loss, as it models the negative log-likelihood of the data. As a result, we propose the *logit-normalized softmax*

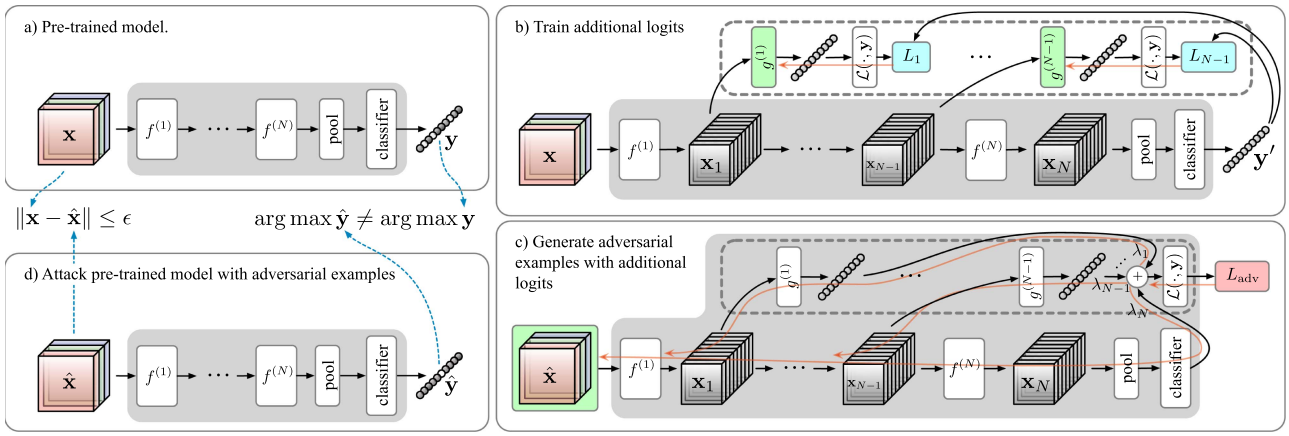


Fig. 4. High-level overview of LAFIT. Note that in each of the steps, layers in gray regions remain fixed, dotted outline regions denote new layers added by LAFIT. We train components in green to respectively minimize and maximize losses in cyan and red blocks. (a) illustrates the original pre-trained model, where $f^{(l)}$ denotes the lth layer (or residual block). (b) trains additional fully-connected layers from intermediate layers, each with a softmax cross-entropy loss until convergence. (c) computes adversarial examples iteratively with a weighted sum of surrogate losses. (d) uses the adversarial examples from (c) to evaluate the robustness of the original model.

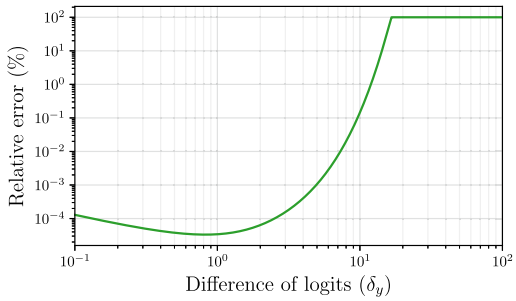


Fig. 5. Comparing the relative error in the half-precision gradient of the SCE loss with its true value as δ_y varies.

cross-entropy (LNSCE) loss, a small modification to the original SCE loss, which scales the logits adaptively before evaluating the softmax operation. For untargeted attacks, the objective is as follows:

$$\mathcal{L}^{\text{lnsce}}(\mathbf{x}, y) \triangleq \mathcal{L}^{\text{sce}}(\text{norm}(\text{f}_{\boldsymbol{\theta}}(\mathbf{x})), y), \tag{6}$$

and the targeted variant maximizes $-\mathcal{L}^{\text{lnsce}}(\mathbf{z}, k)$ to instead encourage the final prediction $\arg\max\{\mathbf{z} \triangleq \text{f}_{\boldsymbol{\theta}}(\mathbf{x})\}$ towards a predefined class target $k \in \mathcal{C} \setminus y$. Here, to prevent saturated softmax values, the logit-normalization function norm normalizes the logits by the DL $\delta_y = \mathbf{z}_y - \max_{\hat{y} \in \mathcal{C} \setminus y} \mathbf{z}_{\hat{y}}$ before applying the softmax operation:

$$\text{norm}(\mathbf{z}) \triangleq \begin{cases} \mathbf{0} & \delta_y \leq 0 \\ \mathbf{z}/(t \cdot \text{detach}(\delta_y)) & \text{otherwise}, \end{cases} \tag{7}$$

where the condition $\delta_y \leq 0$ effectively disables the contribution of $\mathbf{z}$ when the logit vector signifies a successful attack, detach prevents gradients from back-propagating to its input, and the temperature factor $t = 1$ in all of our experiments, and Section V gives a sensitivity analysis of t.

The new surrogate loss $\mathcal{L}^{\text{lnsce}}$ has three-fold advantages. First, it prevents the gradients from floating-point underflows and improves convergence performance. Second, unlike the DL or the DLR loss, it can still represent the original SCE loss faithfully, and all values in the logit vector can still contribute to the final loss. Finally, it maintains the same mathematical optimality as the SCE loss in order to avoid deviation from the objective.

C. Latent Feature Adversary

Following the footsteps of surrogate losses, in Section I we postulate that a similarly indirect loss on latent features can also effectively enhance adversarial attacks. LAFIT exploits the features extracted from intermediate layers to craft even stronger adversarial examples for $\text{f}_{\boldsymbol{\theta}}$. We assume the model architecture $\text{f}_{\boldsymbol{\theta}}$ to generally comprise a sequence of N layers (or residual blocks) and can be represented as:

$$\text{f}_{\boldsymbol{\theta}}(\mathbf{x}) = f^{(N)} \circ f^{(N-1)} \circ \cdots \circ f^{(1)}(\mathbf{x}), \tag{8}$$

where $f^{(1)}, f^{(2)}, \ldots, f^{(N)}$ denotes the sequence of intermediate layers in the model. For simplicity in notation, we omit the parameters from individual layers. We therefore formalize this proposal by generalizing the traditional PGD attack (2) with a *latent-feature PGD* (LFPGD) adversarial optimization problem:

$$\max_{\boldsymbol{g}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathcal{L}^{\text{sur}}} \quad \mathcal{L}^{\text{sce}}(\text{f}_{\boldsymbol{\theta}}(\text{PGD}_{\epsilon, \mathbf{x}, \mathbf{y}}(\mathcal{L}^{\text{lf}}, \boldsymbol{\alpha}, I)), y),$$

$$\textit{where } \mathcal{L}^{\text{lf}}(\mathbf{x}) = \mathcal{L}^{\text{sur}}(\text{f}_{\boldsymbol{\theta}}(\mathbf{x}), \{g^{(l)}(\mathbf{x}^{(l)}) : l \in [1:N]\}, y). \tag{9}$$

Here, the constant I is the maximum number of gradient-update iterations; for each layer $l \in [1:N]$, the term $\mathbf{x}^{(l)} \triangleq f^{(l)} \circ \cdots \circ f^{(1)}(\mathbf{x})$ denotes the features extracted from the lth layer, and the function $g^{(l)}$ maps the features $\mathbf{x}^{(l)}$ to logits. Finally, $\boldsymbol{\alpha}$ denotes the step-size schedule. Our goal is hence to find the right combinations of surrogate loss $\mathcal{L}^{\text{sur}}$ with model output $\text{f}_{\boldsymbol{\theta}}(\mathbf{x})$, auxiliary classifiers $(g^{(1)}, \ldots, g^{(N)})$, and the step-size schedule $\boldsymbol{\alpha}$ to use.

Solving the LFPGD optimization problem as in (9) is unfortunately infeasible in practice. For which we devise methods that could *approximately* solve it, and nevertheless enable us to generate adversarial examples stronger than competing methods.

D. Training Intermediate Auxiliary Layers

To allow the attack to utilize latent features, we begin by introducing auxiliary classifiers to the original model for individual intermediate layers $f^{(l)}$ for all $l \in [1 : N-1]$. The auxiliary classifiers can be formed by appending new branches defined as $g^{(l)} : \mathbb{R}^{C^{(l)} \times H^{(l)} \times W^{(l)}} \rightarrow \mathbb{R}^K$, with each comprising a sequence of layers. First, two 3×3 convolution operations (conv_1 and conv_2), both with stride sizes of 2 and C' output channels, reduce the feature map dimensions to $C' \times \frac{H^{(l)}}{4} \times \frac{W^{(l)}}{4}$. This is then followed by an adaptive pooling layer pool further lowering the size of resulting features to $C^{(l)} \times 2 \times 2$, and finally a fully-connected layer fc for classification:

$$g^{(l)}(\mathbf{x}^{(l)}) \triangleq \text{fc} \circ \text{pool} \circ \text{conv}_2 \circ \text{conv}_1(\mathbf{x}^{(l)}), \tag{10}$$

where $\circ$ denotes composition, and $\mathbf{x}^{(l)}$ is the features extracted from the lth layer. In practice, we let hidden channel numbers be $C' = 64$. As the final layer $f^{(N)}$ is already a classification layer, we assume $g^{(N)} \triangleq \text{id}$ is an identity function. The training process, then trains the parameters within auxiliary classifiers to minimize the SCE loss $\mathcal{L}^{\text{sce}}(g^{(l)}(\mathbf{x}^{(l)}), y_{\text{pred}})$ between the logit outputs $g^{(l)}(\mathbf{x}^{(l)})$ and the predicted labels $y_{\text{pred}} = \arg\max \text{f}_{\boldsymbol{\theta}}(\mathbf{x})$ with sampled images $\mathbf{x}$.

Depending on the availability, we could train the added layers by sampling $\mathbf{x}$ from either $\mathcal{D}_{\text{train}}$, the data samples used for attack $\mathcal{D}_{\text{attack}}$, or both together. While we used $\mathcal{D}_{\text{train}}$ in our experiments, we observed in practice negligible differences in either attack strengths given sufficient amount of training examples, as they are theoretically drawn from the same data sampling distribution. Moreover, this paper differs from the original approach described in [43], as we found in practice, training more accurate auxiliary classifiers improves LAFIT, and the new training procedure does not require ground truth labels, making LAFIT an even more versatile tool.

It is important to note that during the training procedure of $g^{(l)}$ classifiers, the original model $\text{f}_{\boldsymbol{\theta}}$ is used as a feature extractor, with all training techniques (e.g., dropout layers, parameter updates, etc.) disabled. This means that the model parameters $\boldsymbol{\theta}$, the layers $f^{(l)}$ and their parameters, batch normalization [74] statistics, and etc. remain *constant*, while only the parameters in $g^{(l)}$ functions are actively being trained. For the experiments in Section IV, the training process takes only 250 iterations, which is minuscule compared to the full robustness evaluation.

E. Surrogate Loss With Auxiliary Logits

Intuitively, the new surrogate loss should leverage the auxiliary logits $g^{(l)}(\mathbf{x}^{(l)})$ if the auxiliary classifier makes a correct prediction, and otherwise by default use the output logits $\text{f}_{\boldsymbol{\theta}}(\mathbf{x})$ only. Given a pretrained auxiliary classifier $g^{(l)}$, LAFIT then maximizes a loss:

$$\mathcal{L}^{\text{lf}}(\mathbf{x}, y) \triangleq \mathcal{L}^{\text{sce}}(h(\mathbf{x}, y), y), \tag{11}$$

which combines the model output logits and auxiliary logits with:

$$h(\mathbf{x}, y) \triangleq \beta \, \text{norm}(g^{(l)}(\mathbf{x}^{(l)})) + (1 - \beta) \, \text{norm}(\text{f}_{\boldsymbol{\theta}}(\mathbf{x})). \tag{12}$$

Here, $\beta \in [0, 1]$ is a hyperparameter that interpolates between the model output and auxiliary logits.

The above formulation enables the choices of the interpolation constant β and which auxiliary logits $g^{(l)}$ to use. As illustrated in Fig. 2, the auxiliary logits $g^{(l)}(\mathbf{x}^{(l)})$ of layers that are closer to the final layer may exhibit adversarial transferability to successfully deceive the model output. As it is impractical to enumerate over all intermediate layers and β values, we thus search the last 3 intermediate layers, i.e., $l \in [N-3 : N-1]$, and further explore $\beta \in [0.50, 0.25, 0.75]$ to find adversarial examples.

F. Helpful Tactics

In addition to the original contributions explained above, we also employ simple yet helpful tactics from previous literature. First, early stopping helps to prevent successful attacks from reverting to unsuccessful ones, and save computational resources for difficult images. Second, improving on the original linear step-size schedule, we introduce a cosine schedule $\alpha_i = \epsilon(1 + \cos(i\pi/I))$, where ϵ is the perturbation bound, I is the total number of gradient-update steps, and α_i is the step size of the ith step. i is the current iteration number and I denotes the total number of iterations. Third, we adapt momentum-based updates from [17].

To summarize, we illustrate the LAFIT algorithm in Algorithm 1. The function `LAFIT_Attack` computes an adversarial example $\hat{\mathbf{x}}_I$ as return, by accepting the following inputs: the model $f_{\boldsymbol{\theta}}$, the pretrained logits function for the lth layer to be attacked jointly, natural image $\mathbf{x}$ and its true label y, an optional target label k, the interpolation parameter β between the lth layer and the output layer, the momentum weight used ν, the perturbation boundary ϵ, and lastly the maximum iteration count I.

With the algorithm above, we can perform a simple grid search on the combined configurations of $\beta \in [0.5, 0.25, 0.75]$ and $l \in [N-1 : N-3]$ for untargeted attacks. Since using latent features results in faster convergence, the search starts from the penultimate layer ($l = N-1$) with an interpolation mid-point ($\beta = 0.5$), to minimize the number of iterations required to attack each image. Finally, to push the limit of LAFIT, we can further incorporate the *multi-targeted* attack [41]. After attempting attacks with the untargeted surrogate loss (11), we additionally enumerate $k \in \mathcal{C} \setminus y$, i.e., all possible target classes except for the ground truth y to compute the targeted adversarial variants. For computational efficiency, the above search procedure can be early-stopped upon the discovery of successful adversarial examples, until it progressively sifts through all images in $\mathcal{D}_{\text{attack}}$, the set of all natural images that require adversarial counterparts.

G. Attacking Ensemble Defenses

LAFIT further extends the original version to attack ensemble defenses. Unlike conventional defenses, ensemble defenses learn to reduce transferability of adversarial examples among multiple sub-models, in hope that the attacker may be unable to fool sub-models simultaneously, and can thus gain a certain degree of robustness.

Algorithm 1: LAFIT White-Box Attack.

function LAFIT_ Attack($\mathrm{f}_{\boldsymbol{\theta}}, g^{(l)}, \mathbf{x}, y, k, \beta, \nu, \epsilon, I$)
  $\hat{\mathbf{x}}_0 \leftarrow \mathcal{P}_{\epsilon,\mathbf{x}}(\mathbf{x}+\mathbf{u})$, ▷Random start
    where $\mathbf{u} \sim \mathcal{U}([-\epsilon, \epsilon])$
  $\boldsymbol{\mu}_0 \leftarrow 0$
  for $i \in [0 : I-1]$ **do**
    $\mathbf{z} \leftarrow \mathrm{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i)$
    $\mathbf{z}^{(l)} \leftarrow g^{(l)}(f^{(l)}(\hat{\mathbf{x}}_i))$
    $\delta \leftarrow \mathbf{z}_y - \max_{\hat{y}\in\mathcal{C}\setminus y} \mathbf{z}_{\hat{y}}$ ▷DL of output logits
    if $\delta < 0$ **then**
      return $\hat{\mathbf{x}}_i$ ▷Successful attack
    end if
    $\delta^{(l)} \leftarrow \mathbf{z}^{(l)}_y - \max_{\hat{y}\in\mathcal{C}\setminus y} \mathbf{z}^{(l)}_{\hat{y}}$ ▷DL of latent logits
    $\beta_i \leftarrow 1$ **if** $\delta^{(l)} < 0$ **else** β ▷Adaptive weight
    $\mathbf{z} \leftarrow \beta_i \mathbf{z}/\delta + (1-\beta_i)\mathbf{z}^{(l)}/\delta^{(l)}$ ▷Surrogate loss
    if $k \neq$ Null **then** ▷Sign-Gradient
      $\boldsymbol{g}_{i+1} \leftarrow -\operatorname{sign}(\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}^{\mathrm{sce}}(t\mathbf{z}, k))$ ▷Targeted
    else
      $\boldsymbol{g}_{i+1} \leftarrow \operatorname{sign}(\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}^{\mathrm{sce}}(t\mathbf{z}, y))$ ▷Untargeted
    end if
    $\alpha_i \leftarrow \epsilon(1+\cos(i\pi/I))$ ▷Step-size schedule
    $\boldsymbol{\mu}_{i+1} \leftarrow \mathcal{P}_{\epsilon,\mathbf{x}}(\boldsymbol{\mu}_i + \alpha_i \boldsymbol{g}_{i+1})$ ▷Momentum
    $\hat{\mathbf{x}}_{i+1} \leftarrow \mathcal{P}_{\epsilon,\mathbf{x}}(\hat{\mathbf{x}}_i +$ ▷Iterative update
      $\nu(\boldsymbol{\mu}_{i+1} - \hat{\mathbf{x}}_i) + (1-\nu)(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_{i-1}))$
  end for
  return $\hat{\mathbf{x}}_I$ ▷Give up after I iterations
end Function

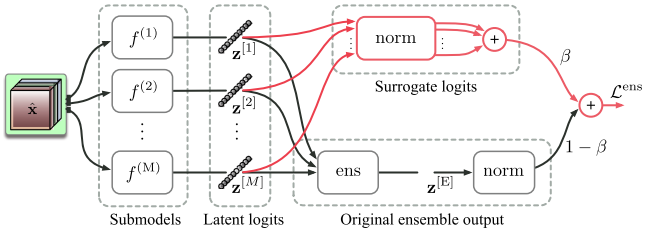


Fig. 6. High-level overview of applying LAFIT on ensemble defense strategies, with new components highlighted in red. Instead of directly attacking the ensemble output, LAFIT further considers each sub-model output as a latent feature, and leverage them jointly with the original output. Here, ens represents the prediction forming process in (13).

Typically, each sub-model in an ensemble readily provides logits outputs for its prediction, and it is thus not required to train additional auxiliary classifiers. In this case, LAFIT can simply use the outputs of each sub-model $f^{(m)}$ as the latent logits. Formally, let $f^{(\mathrm{E})}(\mathbf{x})$ be the ensemble output of M sub-models, and $f^{(m)}(\mathbf{x})$ denotes the mth sub-model output for all $m \in [1 : M]$. Ensemble strategies [22], [23], [29] examined in this paper typically forms model predictions via averaged softmax values of the sub-model outputs, namely:

$$f^{(\mathrm{E})}(\mathbf{x}) \triangleq \frac{1}{M} \sum_{m\in[1:M]} \operatorname{softmax}(f^{(m)}(\mathbf{x})). \tag{13}$$

As we identify earlier, the softmax operations are notoriously difficult for conventional PGD attacks as they can saturate easily. To tackle this, a surrogate loss the ensemble can thus be formed as follows to leverage all sub-model predictions (Fig. 6):

$$\mathcal{L}^{\mathrm{ens}}(\mathbf{x}, y) \triangleq \mathcal{L}^{\mathrm{sce}}(\beta j(\mathbf{x}) + (1-\beta)\operatorname{norm}(f^{(\mathrm{E})}(\mathbf{x})), y),$$
$$\text{where} \quad j(\mathbf{x}) = \frac{1}{M} \sum_{m\in[1:M]} \operatorname{norm}(f^{(m)}(\mathbf{x})). \tag{14}$$

It is notable that by the definition of logits normalization in (3), the norm function not only normalizes the sub-model logits by the respective DLs, but also disables the contributions from the corresponding sub-models with incorrect predictions. This allows LAFIT to concentrate its effort on the remaining robust sub-models, and in practice we observe that this approach can notably improve success rates.

IV. Experiments

To compare fairly against existing SOTA attacks in our evaluation, we test a wide variety of defense techniques that assume the ℓ_∞-norm and ℓ_2-norm threat models on the CIFAR-10 dataset, and also ℓ_∞-norm bounded perturbation on the CIFAR-100 dataset [75], and ensure the list of defenses to be as comprehensive as possible. As baselines, we reproduce and report traditional white-box attacks (e.g., PGD [12], BIM [49] and FAB [50]), each with 100 iterations and a constant step-size of $2/255$.

We examine LAFIT with 4 attack strengths: $\mathbf{LAF}_1$ chooses the penultimate layer, i.e., $l = N-1$ and $\beta = 0.5$, and performs 100 iterations of gradient updates; $\mathbf{LAF}_2$ further restarts the 100-iteration attack multiple times by varying $\beta \in \{0.25, 0.5, 0.75\}$ and the choice of latent layers $l \in \{\text{Unused}, N-1, N-2, N-3\}$, with "Unused" using no latent layers. On top of this, $\mathbf{LAF}_3$ then carries out multi-targeted attacks with 9 closest incorrect targets, all without latent layers. Finally, $\mathbf{LAF}_4$ further repeats the multi-targeted attacks with latent layer enumeration under $\beta = 0.5$. We fix the momentum of all iterative updates at $\nu = 0.75$.

A full comparison can be found in Table I. It reveals that not only is $\mathbf{LAF}_3$ an even stronger attack than the current SOTAs in the robustness evaluation of defense strategies, i.e., AutoAttack (AA) [17], Composite Adversarial Attacks (CAA) [55], and Adaptive Auto Attacks (A^3) [53], but it also improves the worst-case complexity in terms of the maximum number of forward-passes required for each image. Particularly, with 1.9 k iterations, $\mathbf{LAF}_3$ matches and even surpasses the success rates of AA in Table I, which uses 8.3 k iterations. This equates to an approximate speedup of $4.4\times$. Despite AA's diverse set of white-box and black-box attack strategies and a generous computational budget, it can still fall short in accurately evaluating certain defense mechanisms. To push the boundary of LAFIT, we also report $\mathbf{LAF}_4$, which enjoys a substantial increase in computational effort. Note that AA as an ensemble combines multiple strategies (PGD with momentum and two surrogate losses [17], square attack [54] and FAB [50]), and CAA requires an extensive search of attack

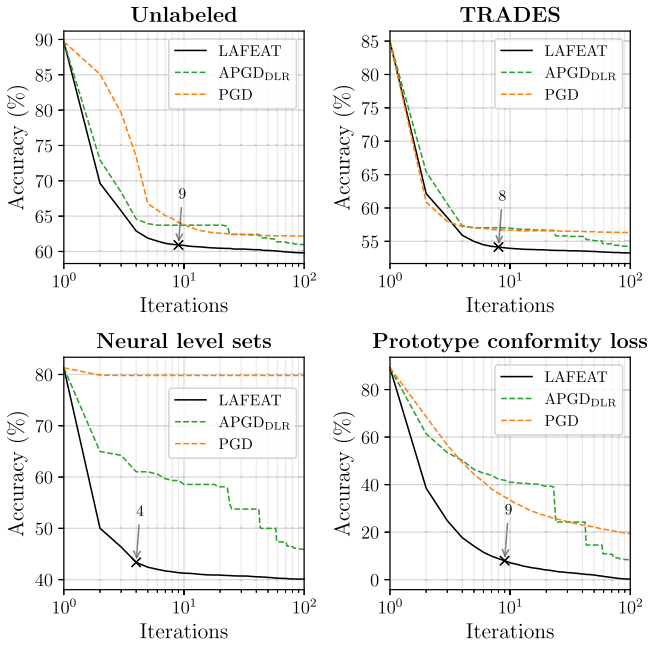


Fig. 7. Comparing the performance of LAFIT with 100 iterations against other adversarial attack methods (APGD$_{DLR}$ [17] and PGD [12]) on defenders [19], [21], [35], [39]. The horizontal and vertical axes respectively show the number of iterations used so far, and the percentage of remaining unsuccessful examples. The iteration count needed for LAFIT to defeat APGD$_{DLR}$-100 is also marked.

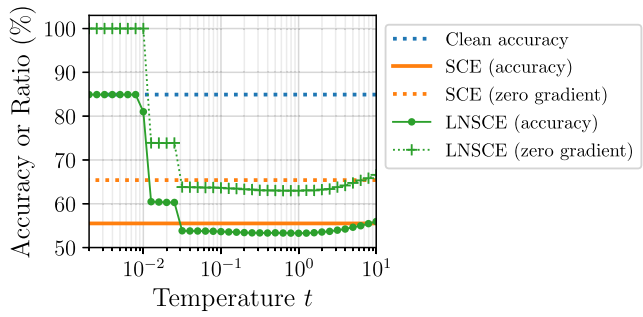


Fig. 8. Higher proportion of underflow gradients correlates with greater difficulty in defeating the defending model. Here, we attack the model obtained from [21] using PGD with 100 iterations. The horizontal axis represents the temperature t in the logit normalization (7).

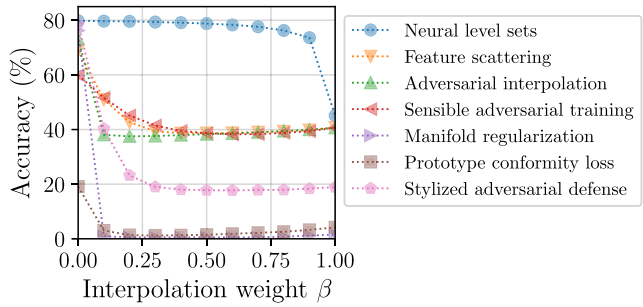


Fig. 9. Accuracy under attack *versus* different β, the interpolation between the output and latent logits for [30], [33], [35], Here, $\beta = 0$ uses only the output logits, and 1 uses only the latent logits.

TABLE II
Robustness Evaluation of Ensemble L_∞ Adversarial Defenses by LAFIT

Defense Method Worst-case complexity	Clean 1	C&W 500	PGD 500	AA 8.3 k	LAF$_4$ 4.6 k
CIFAR-10, ℓ_∞, $\epsilon = 0.01$					
ADP [22]	92.88	7.86	5.98	0.96	**0.31**
GAL [23]	89.41	11.57	8.13	0.95	**0.67**
DVERGE [29]	91.99	40.17	44.49	30.56	**25.30**

We report the clean accuracy and the robust accuracy of the C&W with 100 iterations and five restarts, AA with 8.3k iterations, and LAFITwith 4.6k iterations. The ϵ is equal to 0.01.

TABLE III
PGD-7 Adversarial Training versus **LAF**100,10, Both Explored With (+LF) and Without (–LF) Latent Features

Accuracy under attack (%)		Clean	Adversarial Attack PGD-100	–LF	+LF
Defense	–LF	79.56	47.06	41.21	41.05
	+LF	83.53	47.17	41.50	41.26

TABLE IV
Effect of LAFIT With 4.6 k Iterations on the Adversarial Trained and Compressed WideResNet-28-4 Models From HYDRA [70]

PR (%)	Clean	Nominal	LAFIT	Δ
0%	85.6	57.2	53.94	−3.26
90%	83.7	55.2	51.68	−3.52
95%	82.7	54.2	49.87	−4.33
99%	75.6	47.3	42.73	−4.57

PR denotes pruning ratio, *i.e.*, the percentage of zeros in model weights.

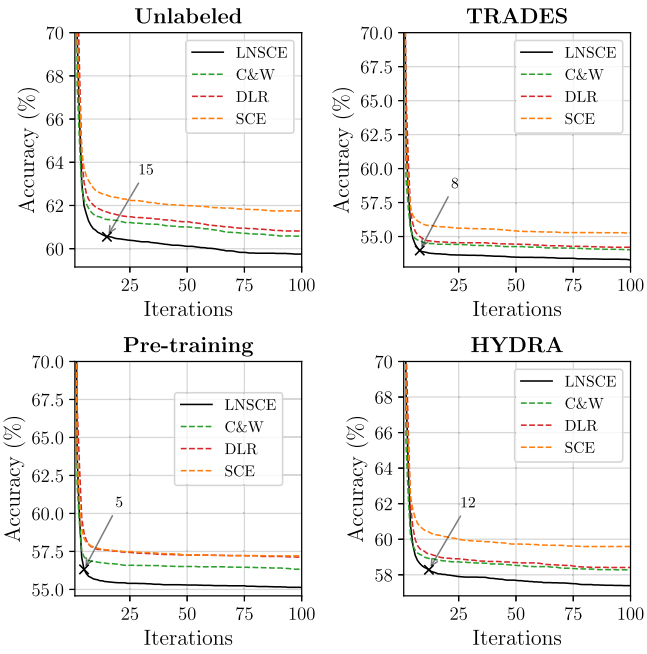


Fig. 10. Comparing the performance of new loss against the standard SCE loss and other surrogate loss functions, i.e., C&W [14], and DLR [17] on defenders [19], [21], [70], [80]. The horizontal and vertical axes respectively show the number of iterations used so far, and the percentage of remaining unsuccessful examples. The figures also mark the iteration count needed for the LNSCE loss to surpass all other loss functions. Note that the LNSCE loss converges much faster than competing losses.

TABLE V
COMPARING THE PROPOSED LNSCE LOSS ($\mathcal{L}^{\text{LNSCE}}$), AGAINST SCE ($\mathcal{L}^{\text{SCE}}$), C&W ($\mathcal{L}^{\text{CW}}$), AND DLR ($\mathcal{L}^{\text{DLR}}$) LOSSES

Defense method	Clean	SCE ($\mathcal{L}^{\text{sce}}$)	C&W ($\mathcal{L}^{\text{cw}}$)		DLR ($\mathcal{L}^{\text{dlr}}$)		LNSCE ($\mathcal{L}^{\text{lnsce}}$)	
CIFAR-10, ℓ_∞, $\varepsilon = 8/255$								
Uncovering limits [28] (large)	91.10	67.96	66.70	(-1.26)	66.78	(-1.18)	**65.96**	(-2.00)
Uncovering limits [28] (small)	89.48	65.59	63.62	(-1.97)	63.82	(-1.77)	**62.96**	(-2.63)
Adversarial weight perturbation [87]	88.25	63.18	60.51	(-2.67)	60.60	(-2.58)	**60.09**	(-3.09)
Unlabeled data [19]	89.69	61.60	60.47	(-1.13)	60.67	(-0.93)	**59.72**	(-1.88)
HYDRA [71]	88.98	59.53	58.21	(-1.32)	58.30	(-1.23)	**57.38**	(-2.15)
Misclassification-aware [25]	87.50	61.60	58.03	(-3.57)	58.73	(-2.87)	**56.88**	(-4.72)
Pre-training [81]	87.11	57.07	56.27	(-0.80)	57.07	(0.00)	**55.10**	(-1.97)
Hypersphere embedding [24]	85.14	61.43	55.35	(-6.08)	56.21	(-5.22)	**53.85**	(-7.58)
Overfitting [67]	85.34	56.85	55.22	(-1.63)	55.97	(-0.88)	**53.62**	(-3.23)
Robustness (Python library) [82]	87.03	51.56	52.07	(+0.51)	52.81	(+1.25)	**49.84**	(-1.72)
YOPO [83]	87.20	46.05	47.02	(+0.97)	47.55	(+1.50)	**45.19**	(-0.86)
Fast adversarial training [69]	83.34	45.75	45.81	(+0.06)	46.89	(+1.14)	**43.57**	(-2.18)
MMA training [84]	83.28	47.69	48.66	(+0.97)	48.69	(+1.00)	**46.03**	(-1.66)
CIFAR-10, ℓ_∞, $\varepsilon = 0.031$								
Self-adaptive training [65]	83.48	56.12	54.30	(-1.82)	54.73	(-1.39)	**53.48**	(-2.64)
TRADES [21]	84.92	55.21	53.94	(-1.27)	54.11	(-1.10)	**53.25**	(-1.96)
Neural level sets [39]	81.30	79.12	40.07	(-39.05)	45.10	(-34.02)	**40.06**	(-39.06)
CIFAR-10, ℓ_2, $\varepsilon = 0.5$								
Uncovering limits [28]	90.90	75.20	74.89	(-0.31)	74.95	(-0.25)	**74.56**	(-0.64)
Adversarial weight perturbation [26]	88.51	74.77	73.88	(-0.89)	73.89	(-0.88)	**73.67**	(-1.10)
Robustness (Python library) [82]	90.83	69.65	70.13	(+0.48)	70.25	(+0.60)	**69.29**	(-0.36)
Overfitting [67]	88.67	68.66	68.74	(+0.08)	69.03	(+0.37)	**67.87**	(-0.79)
Decoupling direction and norm [85]	89.05	66.56	67.00	(+0.44)	67.02	(+0.46)	**66.48**	(-0.08)
MMA training [84]	88.02	66.22	66.58	(+0.36)	66.60	(+0.38)	**66.16**	(-0.06)
CIFAR-100, ℓ_∞, $\epsilon = 8/255$								
Adversarial weight perturbation [26]	60.38	33.09	30.74	(-2.35)	31.13	(-1.96)	**29.35**	(-3.74)
Pre-training [81]†	59.23	32.82	30.58	(-2.24)	31.83	(-0.99)	**29.02**	(-3.80)
Progressive Hardening [86]	62.82	26.18	26.69	(+0.51)	27.26	(+1.08)	**25.17**	(-1.01)
Overfitting [67]	53.83	20.47	20.17	(-0.30)	20.30	(-0.17)	**19.40**	(-1.07)

For each surrogate loss, we use Algorithm 1 ableit without latent layers. Numbers in parentheses indicate the change w.r.t. the SCE baseline.

policies for each model. In contrast, LAFIT uniformly uses only Algorithm 1.

Faster Convergence: It is sensible to argue that models could also rely on *computational security* as one of their defense tactics. In contrast to most attack methods, the effectiveness of LAFIT is not accompanied by high computational costs. By exploiting latent features, we find that it generally leads to faster convergences to adversarial examples than competing methods. In Fig. 7, we compare the speed of convergence among different methods. The two baselines include PGD and APGD_{DLR}, the most effective attack method of the AA ensemble [17] across most defense methods in Table I. For computational fairness, LAFIT uses $\beta = 0.5$ with the penultimate latent layer to compete. All methods run for 100 iterations. The results show that for all 4 defending models, LAFIT is not only stronger, but also often orders of magnitude faster than APGD_{DLR} and PGD for successful attacks. Finally, the logits layers introduce minuscule overhead ($\leq 0.008\%$ additional FLOPs in all models), and have no discernible impact on the iteration time.

Ensemble Defenses: Following the ensemble-based latent attack introduced in Section III-G, we examine LAFIT on recent ensemble-based defenses, i.e., adaptive diversity promoting (ADP) [22], diversifying vulnerabilities (DVERGE) [29] and diversity training with gradient alignment loss (GAL) [23]. Table II shows that LAFIT consistently achieves the strongest results on the defending models.

Adversarial-Trained Latent Features Improve Model Robustness: As demonstrated earlier with LAFIT, one can exploit the latent features learned by defending models to craft powerful adversarial examples. This raises a question: *is it possible to fortify latent features against attacks to improve model robustness?* To answer this, we performed a simple experiment and trained 2 WideResNet-32-10 models, both with PGD-7 adversarial training [12]. The only difference is that in one model, we introduced additional logits layers for the residual block outputs, which were adversarially trained along with the output layer. For attacks, we likewise carry out ablation on the use of latent features. The results are in Table III. It reveals that training latent

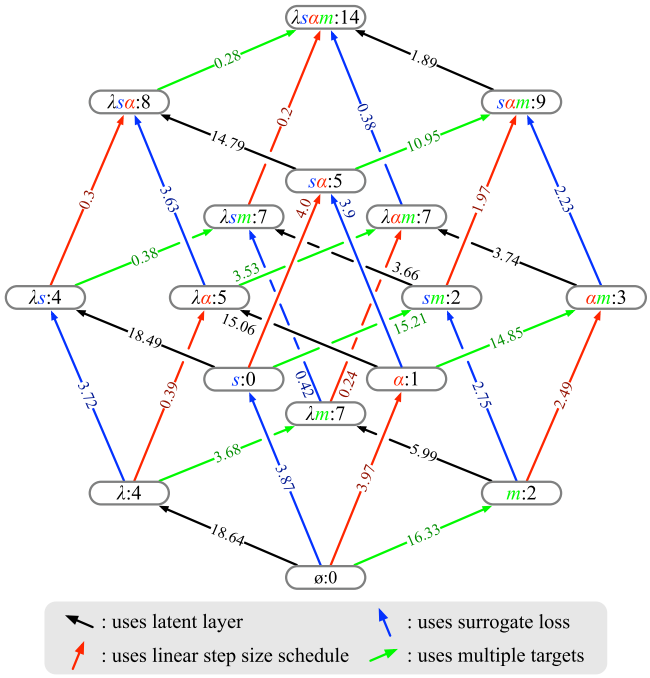


Fig. 11. We ablate LAFIT of its 4 tactics on a complete lattice. Here, the bottom node is standard PGD with 100 iterations. Each node is a unique combination of 4 tactics, where λ introduces the use of latent layers and β intopolation weights; s applies logit normalization; α uses the linear step-size schedule; and m denotes multiple targets, and the number indicates the number of methods where LAFIT is better than AA across 14 different defenses [19], [21], [25], [30], [31], [32], [33], [34], [35], [39], [40], [68], [70], [80]. The arrows , , and represent the introduction of the corresponding tactics, and each number on an arrow indicates the average accuracy degradation as a result of adding that tactic.

features to be more robust can improve the overall robustness even when faced against attacks without using latent features.

Compression versus*Robustness:* In Table IV, compressed models from HYDRA [70] display a worsening robustness degradation between the reported (PGD-50 with 10 restarts) and $\mathbf{LAF}_4$ for an increasing pruning ratio. This shows that model-holistic attacks can potentially overestimate the robustness of compressed models.

V. Sensitivity and Ablation Analyses

Temperature t in the Logit-Normalized SCE Loss: Fig. 8 explores the effect of temperature t on the LNSCE loss $\mathcal{L}^{\text{lnsce}}$ (6). Note that varying the temperature t effectively controls the magnitude of the DL δ_y after logit normalization, i.e., smaller t means larger δ_y. We find that large DL δ_y typically results in zero-valued gradients, making PGD attack more difficult to converge.

Interpolation Between Latent and Output Logits: Fig. 9 varies the interpolation weight between the most effective latent feature and the final logits output for LAFIT with 100 iterations. The experiment used $\beta \in \{0.0, 0.1, \ldots, 0.9\}$ and 100 iterations for each. The results showed that different defense strategies call for distinct β values, making the exploration of β a compelling necessity.

Comparing the SCE Loss, Surrogate Losses and Logit-Normalized SCE Loss: We compared the effectiveness of attacks using either standard SCE loss, or existing surrogate losses (C&W [14] and DLR [17]), with the newly introduced logit-normalized SCE loss $\mathcal{L}^{\text{lnsce}}$. To compare fairly, for each loss we run standard PGD with 100 iterations, but enhance it with the cosine step size schedule. The full results are provided in Table V, and Fig. 10 further compares their convergence rates. It is notable that the logit-normalized SCE loss can consistently outperform the other 3 losses in terms of attack success rates. We believe the reasons of its effectiveness are as follows. First, as it maintains numerical stability across ranges of logit values, it does not suffer from gradient masking as often shown by traditional attacks that use SCE loss function. As many defenses often use the SCE loss as their training loss, well-trained models may thus exhibit low gradient magnitudes under such loss. Second, both C&W and DLR surrogate losses do not use all output values provided by the logit output simultaneously, and could render them less effective. Finally, for certain defenses that do not obscure gradients within the loss function, the DL loss notably exhibits lower performance in comparison to the standard SCE loss. The LNSCE loss thus retains the same mathematical optimality as the SCE loss to prevent deviation from the objective.

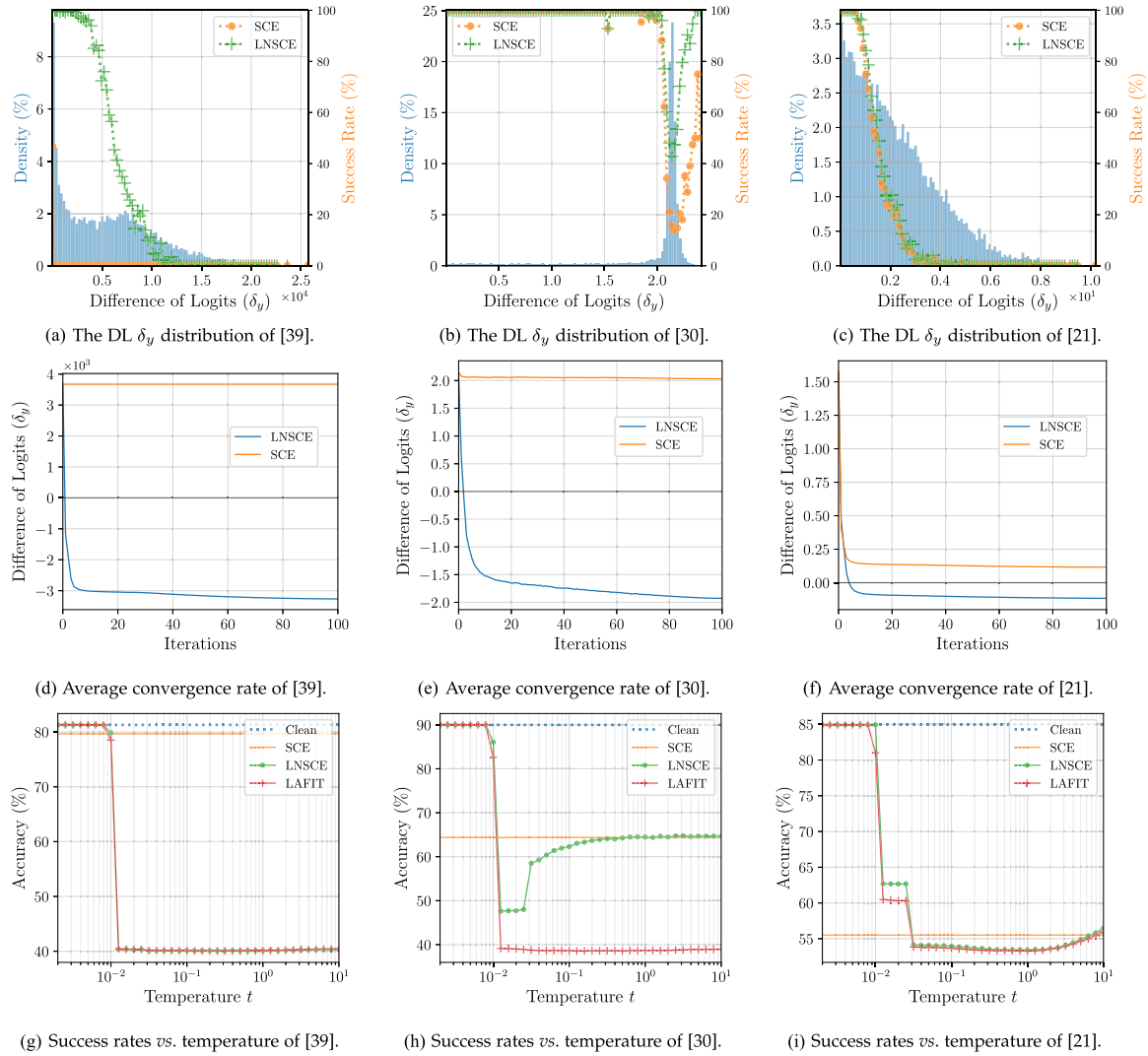
Combinatorial Ablation of Attack Tactics: Table VI presents the ablation analysis of the 4 tactics employed by LAFIT across various defending models, including the use of latent features and logit normalization introduced in the paper. The remaining two tactics, multi-targets and the cosine decay step-size schedule, were inspired by related publications. Fig. 11 further summarizes the results by showing the average accuracy degradations caused by introducing new tactics to all possible combinations. Across all 16 strategy combinations, we discovered that adding latent features to an existing combination consistently has the most significant impact on accuracy among possible choices.

VI. Understanding Gradient Masking Under the Lens of the Difference-of-Logits

In this section, we study the effect of gradient masking under the lens of the distribution of the difference-of-logits (DL) $\delta_y = \mathbf{z}_y - \max_{\hat{y} \in \mathcal{C} \setminus y} \mathbf{z}_{\hat{y}}$, where $\mathbf{z} = \mathrm{f}_{\boldsymbol{\theta}}(\mathbf{x})$ and the data points $(\mathbf{x}, y)$ span the natural images in the CIFAR-10 test set. As mentioned earlier, the DL δ_y is of great importance as a negative δ_y directly signifies a successful attack, and thus all adversarial attack strategies thus attempt to minimize δ_y with respective means to compute bounded perturbations.

By the chain rule, the gradient of the loss $\mathcal{L}^{\text{sce}}(\mathrm{f}_{\boldsymbol{\theta}}(\mathbf{x}))$ w.r.t. the input image $\mathbf{x}$ is $\nabla_{\mathbf{z}}\mathcal{L}\,\nabla_{\mathbf{x}}\mathbf{z}$, where $\nabla_{\mathbf{z}}\mathcal{L}$ and $\nabla_{\mathbf{x}}\mathbf{z}$ respectively signify the gradient of the loss function w.r.t. the logits $\mathbf{z} = \mathrm{f}_{\boldsymbol{\theta}}(\mathbf{x})$ and the logits w.r.t. the input image $\mathbf{x}$. We thus examine all the defending models tested in this paper, and identify three typical failure modes of model robustness assessment using standard PGD, namely gradient masking in: (1) $\nabla_{\mathbf{z}}\mathcal{L}$, i.e., the loss function, (2) $\nabla_{\mathbf{x}}\mathbf{z}$, the model function, and (3) general rounding errors. To illustrate, we provide representative examples to explain the above characteristics:

1) Gradient masking in the loss function: Some defending models may exhibit a high degree of gradient masking in the



(a) The DL δ_y distribution of [39]. (b) The DL δ_y distribution of [30]. (c) The DL δ_y distribution of [21].

(d) Average convergence rate of [39]. (e) Average convergence rate of [30]. (f) Average convergence rate of [21].

(g) Success rates *vs*. temperature of [39]. (h) Success rates *vs*. temperature of [30]. (i) Success rates *vs*. temperature of [21].

Fig. 12. (a, b, c) DL $\delta_y = \mathbf{z}_y - \max_{\hat{y}\in\mathcal{C}\setminus y} \mathbf{z}_{\hat{y}}$ distributions of defending models on correctly predicted natural examples. The dotted lines denote the attack success rate of the image given the initial DL. All results are averaged over 100 bins. (d, e, f) The average convergence rates of δ_y under attack iteration constraints. The vertical and horizontal axes respectively denote the average magnitudes of δ_y and the current number of attack iterations. We average the results over images which PGD-100 with the SCE loss $\mathcal{L}^{\text{sce}}$ fail to attack but $\mathcal{L}^{\text{lnsce}}$ succeeds. (g, h, i) Evaluating model robustness (vertical axes) with varying temperature constants t (horizontal axes) in the logit normalization function (norm) with 100 PGD attack iterations.

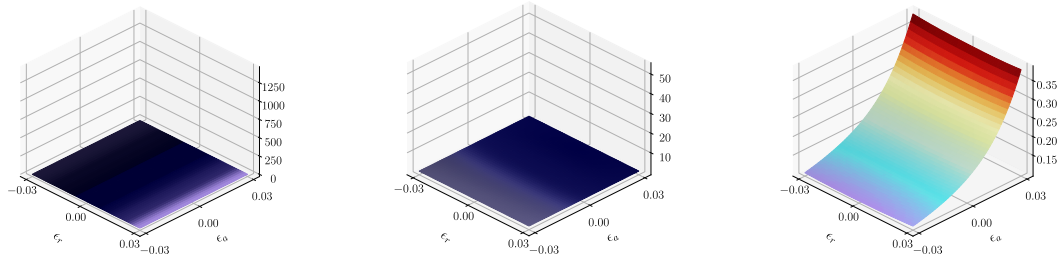
loss function. For instance, controlling the neural level sets [39] generates models that move decision boundaries a constant distance away from all training data, and can surprisingly achieve 79.8% accuracy under PGD-100 attack. Upon inspection, as shown in Fig. 12(a), the magnitudes of δ_y of correctly classified natural images can be very large, which gives a false sense of high confidence in the model's output. Interestingly, these models often exhibit a property where random perturbations to the input image are even more effective than the gradients in increasing the loss (Fig. 13(a)). Clearly, larger differences between logits δ_y lead to saturated results following the softmax operation (as explained in Section III-B), making it challenging for PGD to gather useful gradients using the conventional SCE loss. In such cases, one can address the challenge with surrogate losses [17], [41], to avoid saturated gradients. Consequently, we substitute the SCE loss with its logit-normalized variant $\mathcal{L}^{\text{lnsce}}$ as described in Section III-B, resulting in a notable reduction in model accuracy under PGD-100 to 40.0%. By normalizing logits using (7) before computing the SCE loss, gradient masking can be overcome as the new step eliminates softmax saturation, and notably improve robustness evaluation.

2) Gradient masking in the model: Differing from defenses that make the gradients from the SCE loss unusable, some defending models may mask gradients in the model output logits w.r.t. the input images, i.e., $\nabla_{\mathbf{x}}\mathbf{z}$, rendering them less effective in gradient-based white-box attack algorithms. As an example, the model trained with feature scattering [30] shows this behavior, and can resist PGD-100 attacks with a 69.0% accuracy under perturbed images. Using the logit-normalized variant $\mathcal{L}^{\text{lnsce}}$ in lieu of the SCE loss can notably improve robustness evaluation, bringing the accuracy down to 64.4%; exploring the effect of temperature t can further lower it to

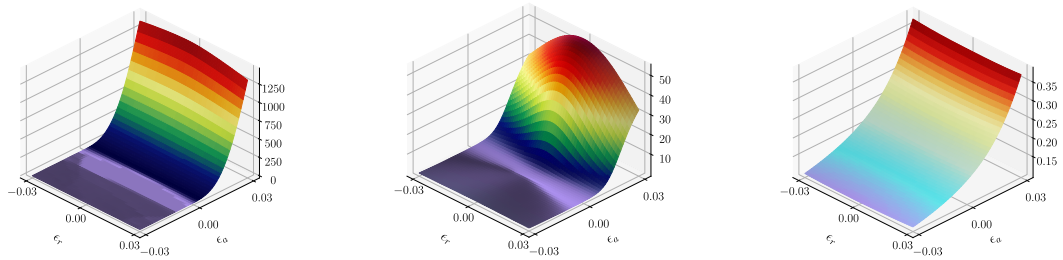
TABLE VI
DETAILED ABLATION STUDIES OF THE 4 STRATEGIES IN LAFIT ACROSS 14 DEFENSES

Defense	[19]	[71]	[25]	[81]	[40]	[69]	[39]	[31]	[32]	[34]	[34]	[34]	[35]	[33]
PGD-100	62.09	59.84	62.2	57.54	56.48	55.48	46.26	79.75	68.82	73.13	59.88	78.16	75.72	19.01
m	60.43	58.07	58.44	55.74	54.67	53.74	44.37	40.17	39.78	52.8	45.65	35.6	15.72	10.61
α	61.7	59.57	61.7	57.17	56.3	55.25	45.99	79.75	64.41	71.45	56.57	68.13	52.78	8.11
αm	60.27	57.91	58.17	55.66	54.54	53.65	44.28	39.92	38.25	49.72	41.84	27.24	6.19	3.36
s	60.01	57.52	57.37	55.28	53.69	53.42	44.03	40.39	68.98	73.17	62.63	78.21	75.41	20.08
s m	59.71	57.3	56.59	55.04	53.43	53.22	43.55	40.08	38.59	44.3	37.92	22.88	17.27	7.47
$s\alpha$	59.75	57.42	56.92	55.14	53.52	53.28	43.87	40.13	64.41	71.51	59.1	68.17	52.48	8.54
$s\alpha m$	59.52	57.15	56.28	54.84	53.28	53.07	43.4	39.88	37.33	41.43	36.55	18.27	6.64	2.12
λ	61.79	59.48	61.2	56.73	56.02	55.13	45.85	74.89	37.22	36.4	35.09	12.73	0.48	0.38
λ m	60.35	57.92	58.02	55.54	54.45	53.65	44.32	40.08	36.35	35.73	33.72	11.44	0.27	0.16
λ α	61.48	59.27	60.99	55.61	55.83	54.96	45.6	74.89	36.58	35.91	34.44	11.97	0.35	0.09
λ αm	60.21	57.78	57.84	55.49	53.4	53.55	44.22	39.86	36.04	35.32	33.38	11.2	0.22	0.04
λs	59.8	57.37	56.9	55.16	53.49	53.26	43.83	40.26	36.9	36.27	34.66	12.59	0.42	0.44
λs m	59.62	57.24	56.34	54.96	53.33	53.16	43.46	39.93	36.39	35.71	33.98	11.47	0.26	0.19
$\lambda s\alpha$	59.6	57.2	56.74	55	53.32	53.1	43.68	39.99	36.39	35.76	34.04	11.93	0.34	0.12
$\lambda s\alpha m$	59.43	57.08	56.15	54.8	53.22	53.02	43.33	39.81	36.06	35.31	33.59	11.18	0.23	0.06
AA	59.53	57.14	56.29	54.9	53.33	53.08	43.41	40.22	36.64	36.45	34.22	13.42	1.35	0.28

Here, λ denotes the use of latent layers, s applies logit normalization, α introduces cosine step schedule, and m additionally uses multi-targeted attacks. Each row denotes a combined method from 16 different possibilities. Each column is a defending model in descending order of final evaluated robustness. The numbers are accuracies (%) under ℓ_∞ attacks bounded by $\epsilon = 8/255$. For reference, we provide the results from Auto Attack (AA) [17] with 8.3 k forward passes.



(a) [39] under PGD attack. (b) [30] under PGD attack. (c) [21] under PGD attack.



(d) [39] under LAFIT attack. (e) [30] under LAFIT attack. (f) [21] under LAFIT attack.

Fig. 13. Averaged loss surfaces $\mathcal{L}^{\text{sce}}(\hat{\mathbf{x}}, y)$ w.r.t. the input space $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{g}\epsilon_{\text{a}} + \boldsymbol{g}^{\perp}\epsilon_{\text{r}}$ across all samples remaining after resisting attacks of defending models. Here, g and $g^{\perp}$ respectively denote normalized adversarial gradient after accumulating 10 iterations of gradient updates at the natural input $\mathbf{x}$, and its randomized orthogonal.

47.6% (Fig. 12(h)). Notably, modifying surrogate losses still gives an overestimated robustness, as we find that exploiting latent features can reduce the accuracy to 38.58% using only 100 iterations.

(3) *Gradient masking with rounding errors:* Such models are unable to show exaggerated robustness against PGD-100 attacks. For instance, LAFIT can only reduce the estimated robustness of the adversarially trained TRADES model [21] from 55.3% under PGD-100 to 53.3%. We find that the attacks on newly generated adversarial examples tend to stop converging as δ_y approaches 0 under PGD-100 (Fig. 12(f)). This might suggest that the added numerical stability introduced by $\mathcal{L}^{\text{lnsce}}$ and the use of latent features can help cross the $\delta_y \leq 0$ boundary, allowing them to be attacked successfully.

VII. Conclusion

LAFIT has demonstrated that exploiting latent features is highly effective against many recent defense techniques. It has efficiently outperformed the current SOTA attack methods across a wide range of defenses. We believe that the future progress in adversarial attack and defense on CNNs depends on the understanding of how latent features can be effectively used as novel attack vectors. The evaluation of adversarial robustness, therefore, cannot view the model solely from a holistic perspective. LAFIT is open-source[1], and we hope it can pave the way for gaining knowledge on robustness evaluation through the explicit use of latent features.

References

[1] T. Cheng, P. Wen, and Y. Li, "Research status of artificial neural network and its application assumption in aviation," in *Proc. 12th Int. Conf. Comput. Intell. Secur.*, 2016, pp. 407–410.

[2] S. Akçay, M. E. Kundegorski, M. Devereux, and T. P. Breckon, "Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 1057–1061.

[3] R. Madaan, et al., "AirSim drone racing lab," in *Proc. Neurips 2019 Competition Demonstration Track*, 2020, pp. 177–191.

[4] G. Wang et al., "Interactive medical image segmentation using deep learning with image-specific fine tuning," *IEEE Trans. Med. Imag.*, vol. 37, no. 7, pp. 1562–1573, Jul. 2018.

[5] Z. Li, M. Dong, S. Wen, X. Hu, P. Zhou, and Z. Zeng, "CLU-CNNs: Object detection for medical images," *Neurocomputing*, vol. 350, pp. 53–59, 2019.

[6] P. Li, X. Chen, and S. Shen, "Stereo R-CNN based 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7644–7652.

[7] M. Bojarski et al., "VisualBackProp: Efficient visualization of CNNs for autonomous driving," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1–8.

[8] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7151–7160.

[9] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[10] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[11] N. Carlini et al., "On evaluating adversarial robustness," 2019, *arXiv: 1902.06705*. [Online]. Available: https://github.com/evaluating-adversarial-robustness/adv-eval-paper

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.

[14] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[15] Y. Dong et al., "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.

[16] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3905–3911.

[17] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, Art. no. 206.

[18] A. Shafahi et al., "Adversarial training for free!," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 3358–3369.

[19] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, "Unlabeled data improves adversarial robustness," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11 192–11 203.

[20] J.-B. Alayrac, J. Uesato, P.-S. Huang, A. Fawzi, R. Stanforth, and P. Kohli, "Are labels required for improving adversarial robustness?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 12192–12202.

[21] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482.

[22] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4970–4979.

[23] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," 2019, *arXiv: 1901.09981*.

[24] T. Pang, X. Yang, Y. Dong, K. Xu, H. Su, and J. Zhu, "Boosting adversarial training with hypersphere embedding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 652.

[25] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *Proc. Int. Conf. Learn. Representations*, 2020.

[26] D. Wu, S. Tao Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 249.

[27] B. Wu, J. Chen, D. Cai, X. He, and Q. Gu, "Does network width really help adversarial robustness?," 2020, *arXiv: 2010.01279*.

[28] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," 2020, *arXiv: 2010.03593*.

[29] H. Yang et al., "DVERGE: Diversifying vulnerabilities for enhanced robust generation of ensembles," 2020, *arXiv: 2009.14720*.

[30] H. Zhang and J. Wang, "Defense against adversarial attacks using feature scattering-based adversarial training," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1831–1841.

[31] H. Zhang and W. Xu, "Adversarial interpolation training: A simple approach for improving model robustness," *OpenReview*, 2020. [Online]. Available: https://openreview.net/forum?id=Syejj0NYvr

[32] J. Kim and X. Wang, "Sensible adversarial learning," *OpenReview*, 2020. [Online]. Available: https://openreview.net/forum?id=rJlf_RVKwr

[33] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli, "Stylized adversarial defense," 2020, *arXiv: 2007.14672*.

[34] C. Jin and M. Rinard, "Manifold regularization for locally stable deep neural networks," 2020, *arXiv: 2003.04286*. [Online]. Available: https://arxiv.org/abs/2003.04286

[35] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Adversarial defense by restricting the hidden space of deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3384–3393.

[36] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," 2020, *arXiv: 2002.08347*.

[37] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. [Online]. Available: https://distill.pub/2017/feature-visualization

[38] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas," *Distill*, vol. 4, 2019, Art. no. e15. [Online]. Available: https://distill.pub/2019/activation-atlas

[39] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman, "Controlling neural level sets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 2032–2041.

[40] L. Huang, C. Zhang, and H. Zhang, "Self-adaptive training: Beyond empirical risk minimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 1624.

[41] S. Gowal, J. Uesato, C. Qin, P.-S. Huang, T. Mann, and P. Kohli, "An alternative surrogate loss for PGD-based adversarial testing," 2019, *arXiv: 1910.09338*.

[42] C. Qin et al., "Adversarial robustness through local linearization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 1240.

[43] Y. Yu, X. Gao, and C.-Z. Xu, "LAFEAT: Piercing through adversarial defenses with latent features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5735–5745.

[44] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh, "AdVersarial: Perceptual ad blocking meets adversarial machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2005–2021.

[45] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–36, 2021.

[1] Available at: https://github.com/lafeat/lafeat.

[46] F. Utrera, E. Kravitz, N. B. Erichson, R. Khanna, and M. W. Mahoney, "Adversarially-trained deep nets transfer better: Illustration on image classification," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=ijJZbomCJIm
[47] D. Bashkirova, B. Usman, and K. Saenko, "Adversarial self-defense for cycle-consistent GANs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 58.
[48] Z. Deng, L. Zhang, K. Vodrahalli, K. Kawaguchi, and J. Y. Zou, "Adversarial training helps transfer learning via better representations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 25179–25191.
[49] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," Google Inc., 2017. [Online]. Available: https://arxiv.org/abs/1607.02533
[50] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proc. Int. Conf. Mach. Learn.*, 2020, Art. no. 205.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980
[52] Y. Tashiro, Y. Song, and S. Ermon, "Diversity can be transferred: Output diversification for white- and black-box attacks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 381.
[53] Y. Liu, Y. Cheng, L. Gao, X. Liu, Q. Zhang, and J. Song, "Practical evaluation of adversarial robustness via adaptive auto attack," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15084–15093.
[54] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 484–501.
[55] X. Mao, Y. Chen, S. Wang, H. Su, Y. He, and H. Xue, "Composite adversarial attacks," in *Proc. 35th AAAI Conf. Artif. Intell. 33rd Conf. Innov. Appl. Artif. Intell. 11th Symp. Educ. Adv. Artif. Intell.*, 2021, pp. 8884–8892.
[56] N. Kumari, M. Singh, A. Sinha, H. Machiraju, B. Krishnamurthy, and V. N. Balasubramanian, "Harnessing the vulnerability of latent layers in adversarially trained models," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 2779–2785.
[57] I. Oseledets and V. Khrulkov, "Art of singular vectors and universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8562–8570.
[58] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4733–4742.
[59] N. Inkawhich, W. Wen, H. H. Li, and Y. Chen, "Feature space perturbations yield more transferable adversarial examples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7066–7074.
[60] S. Baluja and I. Fischer, "Learning to attack: Adversarial transformation networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2687–2695.
[61] Y. Jang, T. Zhao, S. Hong, and H. Lee, "Adversarial defense via learning to generate diverse attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2740–2749.
[62] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[63] P. Mangla, S. Jandial, S. Varshney, and V. N. Balasubramanian, "AdvGAN++: Harnessing latent layers for adversary generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop*, 2019, pp. 2045–2048.
[64] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Representations*, 2017.
[65] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," 2020, *arXiv: 2010.00467*. [Online]. Available: https://arxiv.org/abs/2010.00467
[66] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8093–8104.
[67] T. Chen, S. Liu, S. Chang, Y. Cheng, L. Amini, and Z. Wang, "Adversarial robustness: From self-supervised pre-training to fine-tuning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 696–705.
[68] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJx040EFvH
[69] S. Ye et al., "Adversarial robustness vs. model compression, or both?," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 111–120.
[70] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "HYDRA: Pruning adversarially robust neural networks," 2020, *arXiv: 2002.10509*.
[71] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SyxAb30cY7
[72] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.
[73] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 274–283. [Online]. Available: https://arxiv.org/abs/1802.00420
[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[75] A. Krizhevsky, V. Nair, and G. Hinton, "The CIFAR-10 and CIFAR-100 datasets," 2014. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html
[76] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.
[77] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann, "Data augmentation can improve robustness," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 29935–29948. [Online]. Available: https://openreview.net/forum?id=kgVJBBThdSZ
[78] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving robustness using generated data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 4218–4233.
[79] T. Pang, M. Lin, X. Yang, J. Zhu, and S. Yan, "Robustness and accuracy could be reconcilable by (Proper) definition," in *Proc. 39th Int. Conf. Mach. Learn.*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., 2022, pp. 17 258–17 277. [Online]. Available: https://proceedings.mlr.press/v162/pang22a.html
[80] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2712–2721.
[81] L. Engstrom, A. Ilyas, H. Salman, S. Santurkar, and D. Tsipras, "Robustness (Python library)," 2019. [Online]. Available: https://github.com/MadryLab/robustness
[82] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 227–238.
[83] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang, "MMA training: Direct input space margin maximization through adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HkeryxBtPB
[84] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based L2 adversarial attacks and defenses," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4322–4330.
[85] C. Sitawarin, S. Chakraborty, and D. Wagner, "Improving adversarial robustness through progressive hardening," 2020, *arXiv: 2003.09347*. [Online]. Available: https://arxiv.org/abs/2003.09347
[86] B. Wu, J. Chen, D. Cai, X. He, and Q. Gu, "Do wider neural networks really help adversarial robustness?," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 7054–7067.

Yunrui Yu (Student Member, IEEE) received the BSc and MSc degrees in space engineering from Beihang University, in 2016 and 2019, respectively. He is currently working toward the PhD degree in computer science and technology with the University of Macau. His research interests include adversarial attacks and defenses in deep learning.

Xitong Gao (Member, IEEE) received the MEng degree in information systems engineering and the PhD degree in electronic engineering research from Imperial College London, in 2012 and 2016. He is currently working as an associate researcher with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include efficient inference and training and adversarial robustness of deep learning models.

Cheng-Zhong Xu (Fellow, IEEE) received the BSc and MSc degrees in computer science and engineering from Nanjing University, in 1986 and 1989, respectively, and the PhD degree in computer science and engineering from the University of Hong Kong, in 1993. He is the dean with the Faculty of Science and Technology, University of Macau and a chair professor of computer and information science. His research interests include parallel and distributed computing, BigData, and cloud computing.